Proceedings of the Second International Workshop on

Knowledge Discovery from Sensor Data (Sensor-KDD 2008)

August 24, 2008, Las Vegas, Nevada, USA

held in conjunction with

SIGKDD'08

Workshop Chairs

Ranga Raju Vatsavai, Oak Ridge National Laboratory, TN, USA
Olufemi A. Omitaomu, Oak Ridge National Laboratory, TN, USA
Joao Gama, University of Porto, Portugal
Nitesh V. Chawla, University of Notre Dame, IN, USA
Mohamed Medhat Gaber, Monash University, Australia
Auroop R. Ganguly, Oak Ridge National Laboratory, TN, USA

Second International Workshop on Knowledge Discovery from Sensor Data (Sensor-KDD 2008)

Wide-area sensor infrastructures, remote sensors, and wireless sensor networks, RFIDs, yield massive volumes of disparate, dynamic, and geographically distributed data. As such sensors are becoming ubiquitous, a set of broad requirements is beginning to emerge across high-priority applications including disaster preparedness and management, adaptability to climate change, national or homeland security, and the management of critical infrastructures. The raw data from sensors need to be efficiently managed and transformed to usable information through data fusion, which in turn must be converted to predictive insights via knowledge discovery, ultimately facilitating automated or human-induced tactical decisions or strategic policy based on decision sciences and decision support systems. The challenges for the knowledge discovery community are expected to be immense. On the one hand, dynamic data streams or events require real-time analysis methodologies and systems, while on the other hand centralized processing through high end computing is also required for generating offline predictive insights, which in turn can facilitate real-time analysis. Problems ranging from mitigating hurricane impacts, preparing for abrupt climate change, preventing terror attacks and monitoring improvised explosive devices require knowledge discovery solutions designed to detect and analyze anomalies, change, extremes and nonlinear processes, and departures from normal behavior. In order to be relevant to society, solutions must eventually reach end-to-end, covering the entire path from raw sensor data to real-world decisions.

This workshop will bring together researchers from academia, government and the private sector to facilitate cross-disciplinary exchange of ideas in the following broad areas of knowledge discovery from sensor and sensor stream data.

Offline Knowledge Discovery

1. Predictive analysis from geographically distributed and temporally spread heterogeneous data.
2. Computationally efficient approaches for mining unusual patterns, including but not limited to anomalies, outliers, extremes, nonlinear processes, and changes from massive and disparate space-time data

Online Knowledge Discovery

1. Real-time analysis of dynamic and distributed data, including streaming and event-based data
2. Mining from continuous streams of time-changing data and mining from ubiquitous data
3. Efficient algorithms to detect deviations from the normal in real-time
4. Resource-aware algorithms for distributed mining
5. Monitoring and surveillance based on a single or multiple sensor feeds

Decision and Policy Aids

1. Coordinated offline discovery and online analysis with feedback loops
2. Combination of knowledge discovery and decision scientific processes
3. Facilitation of faster and reliable tactical and strategic decisions

Theory

1. Distributed data stream models
2. Theoretical frameworks for distributed stream mining

Case Studies

1. Success stories, especially about end-to-end solutions, for national or global priorities
2. Real-world problem design and knowledge discovery requirements

The first workshop was a success and attended by more seventy registered participants. This is the second workshop in this series. This year we received very high quality submissions. Each paper is reviewed by at least two program committee members. Based on the reviewers' recommendations, we accepted *seven full papers* and *six short papers*. There is no distinction between full and short papers in term of paper length, only in presentation time. All the accepted papers will be considered for LNCS post workshop proceedings.

In addition to the oral presentations of accepted papers, there will be two invited speakers – Dr. Kendra E. Moore, Program Manager, DARPA/IPTO and Prof. Jiawei Han, Department of Computer Science, University of Illinois at Urbana-Champaign.

We would like to thank the SensorNet® program (the website is available at http://www.sensornet.gov) managed by the Computational Sciences and Engineering Division at the Oak Ridge National Laboratory and other collaborators. In addition, we that the SIGKDD'08 organizers, the authors of the submitted papers, and the members of the Program Committee for their respective and collective efforts to make this workshop possible.

Ranga Raju Vatsavai, Oak Ridge National Laboratory, Oak Ridge, TN, USA
Olufemi A. Omitaomu, Oak Ridge National Laboratory, Oak Ridge, TN, USA
Joao Gama, University of Porto, Portugal
Nitesh V. Chawla, University of Notre Dame, IN, USA
Mohamed Medhat Gaber, Monash University, Australia
Auroop R. Ganguly, Oak Ridge National Laboratory, Oak Ridge, TN, USA

Organizers

Workshop Chairs

Ranga Raju Vatsavai, Oak Ridge National Laboratory, Oak Ridge, TN, USA
Olufemi A. Omitaomu, Oak Ridge National Laboratory, Oak Ridge, TN, USA
Joao Gama, University of Porto, Portugal
Nitesh V. Chawla, University of Notre Dame, IN, USA
Mohamed Medhat Gaber, Monash University, Australia
Auroop R. Ganguly, Oak Ridge National Laboratory, Oak Ridge, TN, USA

Program Committee (in alphabetical order)

1. **Michaela Black**, University of Ulster, Coleraine, Northern Ireland, UK.
2. **Andre Carvalho**, University of Sao Paulo, Brazil.
3. **Sanjay Chawla**, University of Sydney.
4. **Francisco Ferrer**, University of Seville, Spain.
5. **Ray Hickey**, University of Ulster, Coleraine, Northern Ireland, UK.
6. **Ralf Klinkenberg**, University of Dortmund, Germany.
7. **Miroslav Kubat**, University Miami, FL, USA.
8. **Mark Last**, Ben-Gurion University, Israel.
9. **Chang-Tien Lu**, Virginia Tech., VA, USA.
10. **Elaine Parros Machado de Sousa**, University of Sao Paulo, Brazil.
11. **Sameep Mehta**, IBM Research, India.
12. **Laurent Mignet**, IBM Research, India.
13. **S. Muthu Muthukrishnan**, Rutgers University and AT&T Research.
14. **Pedro Rodrigues**, University of Porto, Portugal.
15. **Josep Roure**, Carnegie Mellon University, Pittsburgh, PA, USA.
16. **Bernhard Seeger**, University Marburg, Germany.
17. **Cyrus Shahabi**, University of Southern California, USA
18. **Mallikarjun Shankar**, Oak Ridge National Laboratory, Oak Ridge, TN, USA.
19. **Alexandre Sorokine**, Oak Ridge National Laboratory, Oak Ridge, TN, USA.
20. **Eiko Yoneki**, University of Cambridge, UK.
21. **Philip S. Yu**, IBM Watson Research Center, Yorktown Heights, NY, USA.
22. **Nithya Vijayakumar**, Cisco Systems, Inc., USA.
23. **Guangzhi Qu**, Oakland University, Rochester, MI, USA.

Workshop website: http://www.ornl.gov/sci/knowledgediscovery/SensorKDD-2008/

Table of Contents

Invited Speaker Bio-Sketches

Dr. Kendra E. Moore
Program Manager
DARPA / IPTO
3701 Fairfax Drive
Arlington, VA 22203-1714
http://www.darpa.mil/IPTO/personnel/moore.asp

Dr. Moore's research interests include automatic pattern learning and change detection in complex spatio-temporal data streams. This spans learning data representations, and activity and movement models, and adapting to changes as they occur. Dr. Moore is also interested in developing technology to understand, support, and assess peer production-based information fusion systems. Dr. Moore currently manages the Predictive Analysis for Naval Deployment Activities (PANDA) program. She also managed the Fast Connectivity for Coalition Agents Program (Fast C2AP) program, which transitioned to the US Navy's GCCS-M program in October 2007.

Dr. Moore joined DARPA in 2005. Prior to joining DARPA, Dr. Moore was president and founder of Advestan, Inc., where she provided R&D consulting services to DoD customers and contractors in advanced estimation, analysis, and exploitation for large-scale information fusion applications. Before starting Advestan, Dr. Moore was the Director of Information Fusion at ALPHATECH, Inc. (now BAE Systems). She also served on the Problem-Centered Intelligence, Surveillance, and Reconnaissance (PCISR) study panel to develop recommendations for new all-source ISR architectures for a national intelligence agency. Prior to that, she developed, extended, and applied large-scale systems analysis techniques to a wide range of military command and control systems.

Dr. Moore holds Ph.D. (1998) and M.S. (1989) degrees in Operations Research from Northeastern University, and M.A. (1985) and B.A., (1981) degrees in Philosophy and Religion from Columbia University and Stephens College, respectively.

Professor Jiawei Han
Data Mining Group
Department of Computer Science
University of Illinois at Urbana-Champaign
http://www.cs.uiuc.edu/~hanj

Jiawei Han is a Professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign. His research expertise include data mining, data warehousing, database systems, data mining from spatiotemporal data, multimedia data, stream and RFID data, social network data, and biological data. He has written over 350 journal and conference publications. He has chaired or served in over 100 program committees of international conferences and workshops, including PC co-chair of 2005 (IEEE) International Conference on Data Mining (ICDM), Americas Coordinator of 2006 International Conference on Very Large Data Bases (VLDB). He is also serving as the founding Editor-In-Chief of ACM Transactions on Knowledge Discovery from Data. He is an ACM Fellow and has received 2004 ACM SIGKDD Innovations Award and 2005 IEEE Computer Society Technical Achievement Award. His book “Data Mining: Concepts and Techniques” 2nd ed., Morgan Kaufmann, 2006) has been popularly used as a textbook worldwide.

Spatio-Temporal Outlier Detection in Precipitation Data

Elizabeth Wu
School of Information Technologies
The University of Sydney
Sydney, Australia
ewu1@it.usyd.edu.au

Wei Liu
School of Information Technologies
The University of Sydney
Sydney, Australia
weiliu@it.usyd.edu.au

Sanjay Chawla
School of Information Technologies
The University of Sydney
Sydney, Australia
chawla@it.usyd.edu.au

ABSTRACT

The detection of outliers from spatio-temporal data is an important task due to the increasing amount of spatio-temporal data available and the need to understand and interpret it. Due to the limitations of current data mining techniques, new techniques to handle this data need to be developed. We propose a spatio-temporal outlier detection algorithm called Outstretch, which discovers the outlier movement patterns of the top-k spatial outliers over several time periods. The top-k spatial outliers are found using the **Exact-Grid Top-k** and **Approx-Grid Top-k** algorithms, which are an extension of algorithms developed by Agarwal et al. [2]. Since they use the Kulldorff spatial scan statistic, they are capable of discovering all outliers, *unaffected by neighbouring regions that may contain missing values.* After generating the outlier sequences, we show one way they can be interpreted, by comparing them to the phases of the El Niño Southern Oscilliation (ENSO) weather phenomenon to provide a meaningful analysis of the results.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications - Data Mining

General Terms

Algorithms

Keywords

Spatio-temporal data mining

1. INTRODUCTION

Spatio-temporal data mining is the discovery of interesting spatial patterns from data over time using data mining techniques on spatially and temporally distributed data. One such pattern is a spatio-temporal outlier.

A spatio-temporal outlier is a spatio-temporal object whose thematic (non-spatial and non-temporal) attributes are significantly different from those of other objects in its spatial and temporal neighbourhoods. An extended discussion of the definition of a spatio-temporal outlier is provided in Section 2.

The interest in utilising spatio-temporal data mining techniques for the discovery of outliers in space and time has been prompted by the increasing amount of spatio-temporal data available and the need to interpret it effectively [6]. The spatio-temporal outliers we discover are of interest because they show particular regions which have precipitation behaviour that is significantly different from nearby regions over some period of time. Understanding the patterns of such behaviour over several years could allow us to find relationships between these regional patterns, and other weather patterns such as the El Niño Southern Oscillation (ENSO). Additionally, with enough historical data, we could examine the historical behaviour of rainfall to try and predict future extreme events, their duration, and their locations.

Another driving force behind the increasing popularity of data mining tools for spatio-temporal outlier detection is the inadequacies of existing methods. Traditional data mining techniques have limitations in the discovery of outliers [10]. Such algorithms can be restrictive as outliers do not fit to a specified pattern or have a usual type of behaviour. In addition, these algorithms are often computationally inefficient since finding all the different behavioural patterns of objects may require more time and effort than that required to find outliers.

The ability to discover an increased number of patterns more accurately could enhance our understanding of many different application areas. This research focuses on the field of Hydrology, where knowledge about the behaviour of unusual precipitation could allow governments and individuals to better prepare for extreme events such as floods. Performing data mining on geographic data forms one part of the process of Geographic Knowledge Discovery (GKD), which is 'the process of extracting information and knowledge from massive geo-referenced databases' [10]. Geographic data mining is the use of computational techniques and tools to discover patterns that are distributed over geographic space and time [11], while taking into consideration data features that are specific to geographic domains [14]. In this study, we conduct our experiments on South American precipitation data provided by the National Oceanic and Atmospheric Administration (NOAA) [9], and as such, we also need to consider the geographical features of the data.

Following this, we compare our results to ENSO using the Southern Oscillation Index (SOI), since it is believed that there is a teleconnection or relationship between ENSO and precipitation. ENSO consists of two phases - El Niño and La Niña. The SOI is a measure of the strength and phase of an El Niño or La Niña event. A prolonged negative SOI is associated with an El Niño event, while a positive SOI is associated with a La Niña event. We obtained the values for the SOI from the NOAA Climate Prediction Center [12]. The calculation of the SOI by the NOAA is provided in greater detail in [17]. In South America, El Niño is associated with increased precipitation over parts of South America, so understanding the relationship between extreme precipitation events and ENSO is an important task.

We make the following contributions in this paper:

- Extend the **Exact-Grid** and the more efficient **Approx-Grid** algorithms from [1] to create the **Exact-Grid Top-k** algorithm and the **Approx-Grid Top-k** algorithms, which find the top-k highest discrepancy regions, rather than only the single highest discrepancy region.
- Develop techniques for handling missing data, and dealing with region selection problems encountered when extending these algorithms to find the top-k regions.
- Extend the above algorithms to use Outstretch to find and store sequences of high discrepancy regions over time into a tree structure.
- We provide the **RecurseNodes** algorithm to allow the extraction of all possible sequences and sub-sequences from the **Outstretch** tree-structure.
- We apply **Outstretch** to South American precipitation data, to show that it is capable of finding outlier sequences from the data set.
- We illustrate one way the data could be analysed by comparing the discrepancy of regions to the SOI.

These contributions focus on assisting the Outstretch algorithm to find the moving paths of the most significant outlier regions over several time periods. This makes analysis of the spatio-temporal nature of historical extreme events possible, and could help to predict the location, duration and timing of future extreme events. To the best of our knowledge, the discovery of such patterns has not previously been investigated.

This paper is organised as follows. Section 2 defines and describes the properties of a spatio-temporal outlier. Section 3 provides an overview of related work. Section 4 describes our method of discovering spatio-temporal outliers. Section 5 outlines the experimental setup and the results we achieve. A discussion of our technique and results, followed by a conclusion and suggestions for further research is given in Section 6.

2. SPATIO-TEMPORAL OUTLIERS

An outlier can constitute different things in different application settings, and as a result the precise definition of an outlier is difficult to capture [13]. Spatio-temporal outlier detection is an extension of spatial outlier detection. It aims to find spatial outliers, but instead of just looking at a single

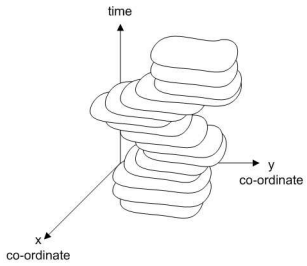
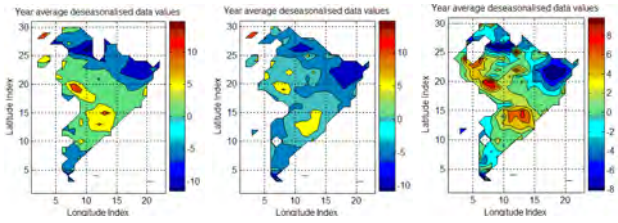


Figure 1: A moving region

snap shot in time, it considers the behaviour of these outliers over several time periods. Cheng and Li [5] define a spatio-temporal outlier to be a "spatial-temporal object whose thematic attribute values are significantly different from those of other spatially and temporally referenced objects in its spatial or/and temporal neighbourhoods". Birant and Kut [3] define a spatio-temporal outlier as "an object whose non-spatial attribute value is significantly different from those of other objects in its spatial and temporal neighborhood". From these definitions, Ng [13] notes two possible types of outliers - extreme and non-extreme. If the values under examination follow a normal distribution, then extreme values at the tail of the distribution are classified as extreme value outliers. We focus on this type of outlier as they are indicative of extreme events.

Since a spatio-temporal outlier is a spatio-temporal object, we also need to provide a definition of a spatio-temporal object. Theodoridis *et al.* [15] define a spatio-temporal object as a time-evolving spatial object whose evolution or 'history' is represented by a set of instances (o_id, s_i, t_i), where the spacestamp s_i, is the location of object o_id at timestamp t_i. According to this definition, a two dimensional region is represented by a solid in three-dimensional space. A conceptual example of a moving region is shown in Figure 1. A moving region identified from our dataset can be seen in Figure 2 in the grid whose bottom left corner identified as longitude 10 and latitude 10.



(a) 2002 (b) 2003 (c) 2004

Figure 2: Deseasonalised Precipitation Data from the South American NOAA Precipitation Dataset

3. RELATED WORK

In previous work, Birant and Kut [3] define two spatial objects (S-objects) as temporal neighbors if the "values of these objects are observed in consecutive time units such as consecutive days in the same year or in the same day in consecutive years". However, Birant and Kut regard a spatio-

temporal outlier to be a spatial outlier from a single time period that is different from its immediate temporal neighbours. This is essentially a spatial outlier in spatio-temporal data. Since we follow the definition of a spatio-temporal object provided by Theodoridis *et al.* [15], and consider that a spatio-temporal outlier may exist over more than one time period. For example, in our work if there is higher than average precipitation in Peru over the years 1998-2002, then the solid in three dimensional space is an outlier. In experiments conducted by Birant and Kut [3], they discover a region in the Mediterranean Sea that is a spatial outlier in 1998, where the years immediately preceeding and following it, 1997 and 1999, contain different values for the region. While this is also considered to be an spatio-temporal outlier by our definition, we remove the limitations imposed by Birant and Kut's definition, and are also able to discover spatio-temporal outliers that persist over several time periods and which may move or evolve in shape and size. This is important as valuable outliers such as extreme events may exist over a number of time periods.

Before moving outliers can be discovered, however, we first need to find the spatial outliers in each time period. To identify the most significant outliers, we use a measure known as discrepancy. One well-known method for determining the discrepancy of spatial outliers is the spatial scan statistic, first introduced in [8]. This statistic has been applied by [1] to spatial grid data sets using an algorithm called Exact-Grid. We extend these to find the top-k outlier regions, rather than only finding the single 'highest discrepancy' region.

The Exact-Grid algorithm finds every possible different rectangular region in the data using four sweep lines to bound them. Once found, a well-known spatial scan statistic known as Kulldorff's scan statistic is applied to give each rectangular region a discrepancy value that indicates how different it is from the rest of the dataset [2].

The Kulldorff spatial scan statistic uses two values: a measurement m and a baseline b. The measurement is the number of incidences of an event, and the baseline is the total population at risk. For example, when finding disease outliers, m would be the number of cases of the disease and b would be the population at risk of catching the disease [1].

To calculate the Kulldorff scan statistic, $d(m, b, R)$ for a region R, we first need to find the measurement M and baseline B values for the *whole* dataset, where $M = \Sigma_{p \in U} m(p)$ and $B = \Sigma_{p \in U} b(p)$, and U is a box enclosing the entire dataset.

We then use these global values to find m and b for the local region R, by letting $m_R = \Sigma_{p \in R} \frac{m(p)}{M}$ and $b_R = \Sigma_{p \in R} \frac{b(p)}{B}$.

Once these values have been found, all we need to do is perform a simple substitution into the Kulldorff scan statistic, which is given by
$d(m_R, b_R) = m_R log(\frac{m_R}{b_R}) + (1 - m_R) log(\frac{1-m_R}{1-b_R})$ if $m_R > b_R$ and 0 otherwise.

One of the most notable advantages of using the spatial scan statistic is that its ability to detect outliers is *unaffected by missing data regions.* This is particularly relevant in geographical data, which often contains a large number of missing values for regions and/or time periods.

For the Exact-Grid algorithm, one approach of dealing with missing values was to set missing value as the average value in the dataset. That would mean that the measurement m for the grid cell would be 0, while the baseline b for the grid cell would become 1. However, this large baseline measure has an impact on the data, and causes larger grids to be selected as high discrepancy regions due to the larger baseline population considered to be at risk. Instead, the aproach we have adopted for each missing grid, is to set the baseline value b, which represents the total population, to zero, as there is no population present in a missing region. In addition because the population is zero, the measurement m must be zero. This means that when we try to calculate the spatial scan statistic, $m_R > b_R$ does not hold so the resulting value is 0. As shown in the following example, zero regions are not considered as outlier regions, and so the presence of missing values has minimal effect on the ability of the spatial scan statistic to find outliers. In essence we are able to ignore the missing regions while still detecting valuable and interesting outliers.

An example of the application of the Kulldorff scan statistic is provided in Figure 3. In this example, the maximum discrepancy of the shaded area is calculated by finding $M = 6$, $B = 16$, $m_R = \frac{4}{6}$ and $b_R = \frac{4}{16}$. Substituting this into the formula gives a value of $d(m_R, b_R) = 0.3836$.

	1	2	3	4
1	1	1	0	0
2	1	1	0	0
3	0	0	0	1
4	0	0	1	0

Figure 3: An example grid for calculating Kulldorff's scan statistic

Previous work using the spatial scan statistic to detect space-time clusters in point data has been done in [7]. They use a pyramid with square cross sections at each time interval, that can either expand or contract from the start to finish of the time interval. Candidate clusters are generated in a biased random fashion using a randomised search algorithm. However, their algorithm aims to discover clusters of point data whereas our research focuses on the detection of outliers from grid data. We have adopted the concept of using rectangular cross-sections at particular time periods from [7] to finding outliers. However, the methods we have used to generate the initial rectangles, and also the subsequent rectangles are different. These changes allow us to accommodate the discovery of outliers from data located in a spatial grid, and to find all sequences of rectangular outliers that are stationary, move and/or change shape. Due to the size and dimensionality of the data, finding patterns which deviate from the normal behaviour of the dataset is a non-trivial problem. While previous methods exist to find spatial outliers in spatio-temporal data, to the best of our knowledge, no other method exists that is capable of finding spatial outliers over more than one time period.

4. OUR APPROACH

This section describes our approach taken to discover sequences of spatial outliers over time. It consists of three main steps:

1. Find the top-k outliers for each time period, using the

Exact-Grid Top-k or **Approx-Grid Top-k** algorithms.

2. Using these top-k for each time period, find all the sequences of outliers over time and store into a tree, using the **Outstretch** algorithm.

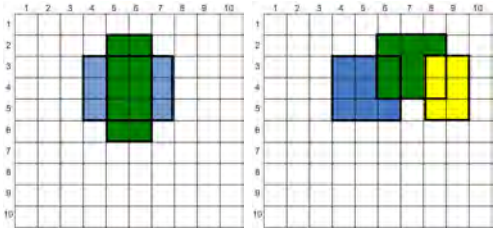
3. Extract all possible sequences from the tree using the **RecurseNodes** algorithm.

The resulting output from the above steps is a list of all sequences and subsequences of outliers found in the dataset. Each of these steps is described in the following subsections.

4.1 Exact-Grid Top-k

The Exact-Grid algorithm was proposed by Agarwal *et al.* [1]. It uses 4 sweep lines to find all possible different shaped regions that are located over a grid space of size $g \times g$. It takes $O(g^4)$ time to run the algorithm, since there are g^4 rectangles to consider. Runtime is minimised by maintaining a count of the measurement m and baseline b values for each row between the left and right scan lines. By doing this they are able to calculate the Kulldorff discrepancy value in constant time. Our extension to the Exact-Grid algorithm, called Exact-Grid Top-k, finds the top-k outliers for each time period. This is important since there may be a number of interesting outliers present in any given time period.

Since the Exact-Grid algorithm only finds the single highest discrepancy outlier, it did not have to take into account overlapping regions, as any region which had a lower discrepancy was simply replaced. When adding regions to the list of top-k algorithms however, we need to consider the case where there are overlapping regions, or else we could end up with a list of top-k regions that lie over the same area. This is illustrated in Figure 4(a) , where the green region is overlapping the blue region. The different types of overlap that we considered are shown in Figure 5.



(a) Two-region overlap problem (b) Chain overlap problem

Figure 4: Overlap problems

However, simply eliminating the lowest discrepancy region is not always the best solution, particularly if the regions only overlap slightly as this could eliminate some potentially interesting outliers. Therefore, we have introduced a threshold parameter, so we can specify the maximum amount of allowable overlap between regions.

Another issue is the scenario where there is a chain of overlaps, as shown in Figure 4(b). In this case, the discrepancy of the blue region is less than that of the green, and the discrepancy of the green is less than the yellow (i.e. $d(blue) < d(green) < d(yellow)$). If we are eliminating

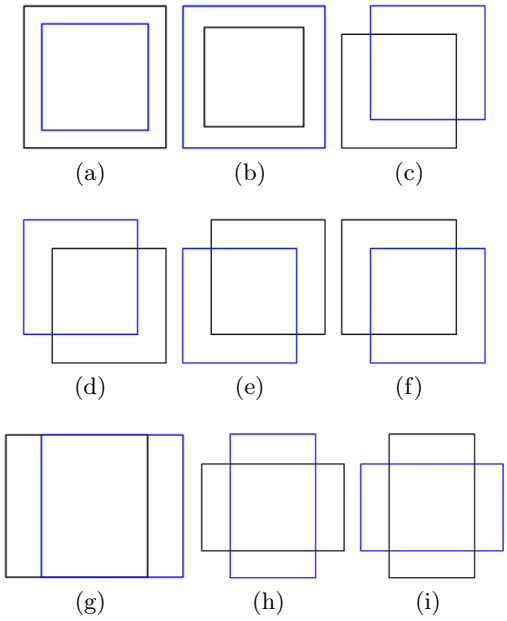


Figure 5: Possible overlap types between two top-k regions

based on the highest discrepancy, and we find the blue region first and add it to our list of top-k outliers, when we find the green region, it will replace the blue region in the top-k outlier list. Then, when we find the yellow region, it will replace the green region in the list of top-k outliers. This creates a chain effect, which is problematic as the blue region may be quite different or far from the yellow region and yet has been eliminated.

One option that was considered was to form a union between the two regions. Then if the discrepancy of the unioned region was higher, the two sub-regions would be discarded, and the union would be stored in their place in the list of $top - k$. This concept is shown in figure 6. However, this would have decreased the efficiency of our algorithm, since it would be more complex to search irregular shapes for overlaps.

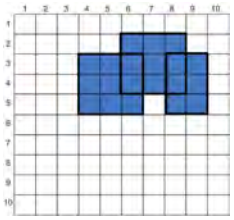


Figure 6: The union solution to the overlap problem

Instead, to deal with the overlap problem, we chose to allow some overlap between regions. This means we are able to include both spatial outliers in the top-k outlier list, which is important because despite their overlap, they represent significantly different locations. The amount of overlap is specified as a percentage, and the algorithm allows the user

to vary this to the most appropriate amount for their particular application domain. The procedure is described in the following paragraphs.

The Exact-Grid Top-k algorithm finds the top-k outliers for each time period by keeping track of the highest discrepancy regions as they are found. As it iterates through all the region shapes, it may find a new region that has a discrepancy value higher than the lowest discrepancy value ($k^{th} value$) of the top-k regions so far. We then need to determine if this region should be added to the list of top-k regions. To do this we need to determine the amount of overlap that the new region has with regions already in the top-k.

For any top-k region that this new candidate region overlaps with, we first calculate the percentage overlap between the two regions. If it overlaps more than the percentage specified by the parameter at input, such as 10%, then we compare the discrepancy values of the two regions. If the new region has a higher discrepancy value, it will replace the other region, otherwise the new region will not be added. In the case where the percentage overlap is below the parameter specified maximum allowable overlap, the region will added to the list of top-k, provided that it does not violate the overlap condition with any of the other top-k regions. The Exact-Grid Top-k algorithm is shown in Algorithm 1.

Exact-Grid Top-k computes the overlapping region in $O(k)$ using the subroutine in Algorithm 2, since it has to check the new potential top-k region against all previous regions for overlap. Because of this, the total time required by the algorithm is $O(g^4k)$.

The Update Top-k subroutine calls the **get_overlap** method, which calculates the percentage overlap between region c, the current region under examination and each of the regions *topk* in the list of top-k regions. If the overlap is less than 10%, the region will be added to *topk* and will bump the k^{th} highest discrepancy region off the list. Otherwise only the highest discrepancy region will be kept in *topk*.

Algorithm 1 Exact-Grid Top-k Algorithm

Input: $g * g$ grid with values $m(i,j)$, $b(i.,j)$, $max\_overlap$
Output: Top-k highest discrepancy regions *topk*

```
1: // Left Sweep Line
2: for i = 1 to g do
3:    Initialize m[y] = m(i,y), b[y] = b(i,y) for all y
4:    for y = 2 to g do
5:       m[y]+ = m[y − 1], b[y]+ = b[y − 1]
6:    end for
7:    // Right Sweep Line
8:    for j = i+1 TO g do
9:       m = 0, b = 0
10:      for y = 1 to g do
11:         m+ = m(j,y), b+ = b(j,y),
12:         m[y]+ = m, b[y]+ = b
13:      end for
14:      // Bottom Sweep Line
15:      for k = 1 to g do
16:         // Top Sweep Line
17:         for l = k to g do
18:            if k = 1 then
19:               m = m[k],b = b[k]
20:            else
21:               m = m[l] − m[k − 1],
22:               b = b[l] − b[k − 1]
23:            end if
24:            if (d(m,b) > topk(k)) then
25:               c=the current region,
26:               topk=update_topk(c,topk)
27:            end if
28:         end for
29:      end for
30:   end for
31: end for
```

4.2 Approx-Grid Top-k

Instead of using two horizontal lines to determine the testing area, our Approx-Grid Top-k algorithm follows Agarwal's [1] method of approximation. That is, all the points inside the cells are projected onto the right sweep line. This is done using the same linear function as in Agarwal's paper [1], which is:

$$L(m_R, b_R) = \cos(\sin(\pi^2/8))m_R - \sin(\sin(\pi^2/8))b_R$$

where R is the right sweep line. Then we find the interval r on the right sweep line which maximises the linear function.

As shown in Algorithm 3, the main difference between Exact-Grid Top-k and Approx-Grid Top-k is their approach to creating the test rectangles from the grid. The two algorithms both used two vertical lines to determine the left and right edges of the test area. However, while Exact-Grid uses two horizontal lines (line 15 and line 17 in Algorithm 1) to form the bottom and top edge, Approx-Grid Top-k projects all points inside the two vertical lines onto the right vertical sweep line (line 6 to line 9 in Algorithm 3). By doing this, Approx-Grid Top-k reduces the maximising problem down to one-dimension. Thus, instead of two sweep lines moving from the bottom to the top, it simply finds the preceding interval r which maximises the above function (line 11 in Algorithm 3). Then the ends of this interval are used as the top and bottom edges of the test rectangle (line 12 in in Algorithm 3).

The runtime of Approx-Grid Top-k is $O(g^3k)$. This includes g iterations for the left sweep line, g iterations for the right sweep line, g iterations for finding the maximized interval and k iterations for the **update_topk** routine.

4.3 The Outstretch Algorithm

To discover sequences of outliers, we developed Outstretch, which is detailed in Algorithm 4. Outstretch takes as input the top-k values for each year period under analysis, and a variable r, the region_stretch, which is the number of grids to 'stretch' by on each side of an outlier. This is shown in Figure 7.

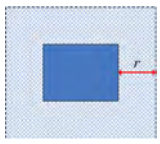


Figure 7: Region Stretch Size r

Outstretch then examines the top-k values of the second

Algorithm 2 Update Top-k Subroutine

Input: c, $topk$, $max\_overlap$
Output: Top-k highest discrepancy regions $topk$

```
1: for all tk = topk do
2:    ov=get_overlap(c, tk)
3:    if ov < max_overlap then
4:       add c to topk
5:    else
6:       if dval(c)>dval(tk) then
7:          replace tk with c in topk
8:       end if
9:    end if
10: end for
```

to last available year periods. For all the years, each of the outliers from the current year are examined to see if they are framed by any of the stretched regions from the previous year, using the function **is_framed**. If they are, the variable 'framed' will return true, and the item will be added to the end of the previous years child list. As a result, all possible sequences over all years are stored into the *outlier_tree*.

An example tree structure is shown in Figure 8. In this example we have found the top 4 outliers over 3 time periods. These are labelled with two numbers, the first being the time period, the second being an identification number of the outlier from that time period. The algorithm stores all single top-k outliers from each time period, *yrly_topkvals*, as rows in a table, where each of these rows contains the outliers children. An example table corresponding to the example tree in Figure 8 is given in Figure 9. From this table we can extract the spatial outliers that persist over several time periods in a similar location.

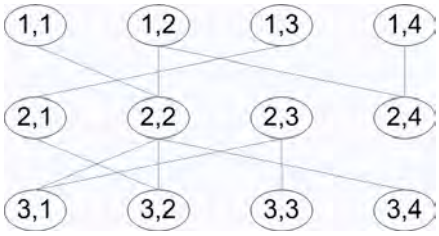


Figure 8: An example outlier tree built by the Outstretch algorithm

The example in Figure 10 shows how a sequence of outliers over three time periods can be collected. In each diagram, the outlier is represented by the solid blue region, while the stretch region is represented by the shaded blue region. In this example, the stretch size r equals 1. To begin, the outlier from the first time period is found. Then the region is extended by the stretch size on all sides, and is searched for outliers that are enclosed by it in the following year. If one is found that lies completely within the stretch region, a new stretch region is generated around the new outlier. This new stretch region is then searched for outliers in the third time period. This process continues until all time periods have been examined or there are no outliers that fall completely within the stretch region. As each of these outliers sequences

Algorithm 3 Approx-Grid Top-k Algorithm

Input: $g \times g$ grid with values $m(i,j)$, $b(i,j)$, $max\_overlap$
Output: Top-k highest discrepancy regions $topk$

```
1: // Left Sweep Line
2: for i = 1 to g do
3:    Initialize m[y] = m(i, y), b[y] = b(i, y) for all y
4:    // Right Sweep Line
5:    for j = i+1 TO g do
6:       for y = 1 to g do
7:          m[y] = m[y] + m(j, y),
8:          b[y] = b[y] + b(i, y)
9:       end for
10:      // The interval that maximizes the linear function
11:      r = arg max_{r∈R} L(r),
12:      (y_b, y_t) = r
13:      if y_b = 1 then
14:         m = m[1],b = b[1]
15:      else
16:         m = m[y_t] − m[y_b − 1],
17:         b = b[y_t] − b[y_b − 1]
18:      end if
19:      if (d(m, b) > topk(k)) then
20:         c=the current region,
21:         topk=update_topk(c,topk)
22:      end if
23:   end for
24: end for
```

are discovered they are stored into a tree.

Algorithm 4 Outstretch Algorithm

Input: $yrly\_topkvals$, region stretch (r), years (y)
Output: outlier_tree (tr)

```
1: for yr = 2 to y do
2:    c = yrly_topkvals(yr)
3:    for all c do
4:       p = yrly_topkvals(yr-1)
5:       for all p do
6:          framed = is_framed(c,p,r)
7:          if framed = true then
8:             tr(p,len(tr(p))+1) = c
9:          end if
10:      end for
11:   end for
12: end for
```

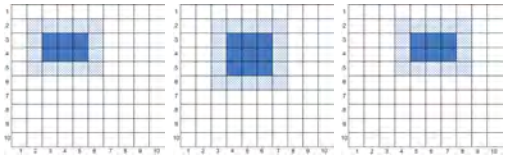
The Outstretch algorithm runs in $O(n^3)$, since for each of the time periods available it iterates through all the top-k outliers for that period, and compares them against all the outliers from the previous time period.

4.4 The RecurseNodes Algorithm

From the tree structure, all possible sequences are extracted using a simple recursive algorithm, described in Algorithm 5. RecurseNodes takes 4 input variables. *outlier_tree* contains a list of each node and its children. *sequence* contains the sequence nodes so far, excluding the

Outlier	Number of Children	Child List
(1,1)	1	(2,2)
(1,2)	2	(2,2),(2,4)
(1,3)	1	(2,1)
(1,4)	1	(2,4)
(2,1)	1	(3,2)
(2,2)	3	(3,1),(3,2),(3,4)
(2,3)	2	(3,1),(3,3)
(2,4)	0	
(3,1)	0	
(3,2)	0	
(3,3)	0	
(3,4)	0	

Figure 9: The outlier table corresponding to the outlier tree shown in Figure 8



(a) Outlier at t=1 (b) Outlier at t=2 (c) Outlier at t=3

Figure 10: Example Outstretch Algorithm Sequence

child value. *child_list* is a list of all the children of the last sequence item. *sequence_list*, is a list of all the sequences that have been generated from the outlier_tree at any point in time. The RecurseNodes algorithm has a running time of $O(n^y)$ where n is the total number of length-1 outliers, and y is the number of years of data and maximum possible length of an outlier sequence.

5. EXPERIMENTS

5.1 Data

Special features of geographical data need to be considered when performing geographical data mining. Traditional data mining algorithms often assume the data to be independent and/or identically distributed [4]. These assumptions would violate Tobler's first law of Geography, which states that 'everything is related to everything else, but near things are more related than distant things' [16]. For the purpose of our experiments, we have removed some of the temporal dependencies in the data through deseasonalisation. This allows us to discover more interesting patterns.

Secondly, we have had to consider the effects of missing data. In our dataset, there are a large number of missing values, mostly in regions that do not lie over land masses. One of the advantages of using the spatial scan statistic is that it is able to discover significant outliers despite their close proximity to regions that contain missing data. Section 3 provides further details on the techniques used to handle missing values.

The data used in our experiments is the South American precipitation data set obtained from the NOAA [9]. It is

Algorithm 5 RecurseNodes Algorithm

Inputs: outlier_tree(tr), sequence(seq), child_list(ch_list), sequence_list(seq_list)
Outputs: sequence_list (seq_list)

```
1: for all c in ch_list do
2:    new_seq = seq + c
3:    // append new_seq to the end of seq_list:
4:    seq_list(len + 1) = new_seq
5:    //get the grandchildren:
6:    gchildr = tr(c)
7:    if size(gchild) > 0 then
8:       seq_list =
9:       RecurseNodes(tr,new_seq,gchildr,seq_list)
10:   end if
11: end for
```

provided in a geoscience format, known as NetCDF. A description of the data is provided in both [9] and [17]. The data are presented in a grid from latitude $60°S$ to $15°N$ and longitude $85°W$ to $35°W$. It contains daily precipitation values from around 7900 stations, whose values are averaged for each grid between 1940-2006.

Before we apply our algorithms to the data it must be pre-processed. It is difficult to spot interesting patterns in raw precipitation values as rainfall patterns are seasonal, so our attention is first drawn to these seasonal patterns. Such patterns are not very interesting, as we are usually aware of which seasons have greater rainfall. A more interesting statistic is the deviation of precipitation from the normal amount of precipitation expected at any particular time in the year. Data in this form is deseasonalised, and the deseasonalisation process is described in [17].

Once we have completed the deseasonalisation procedure, we take the average of the deseasonalised values over each period, for each grid, before running either the Exact-Grid Top-k or Approx-Grid Top-k algorithm, which gives a discrepancy value to each region.

5.2 Experimental Setup

The effectiveness of the Outstretch algorithm is evaluated by counting the total number of spatio-temporal outliers in the outlier tree, and the length of these outliers.

For this experiment, we set the input variables as shown in Figure 11.

Allowable Overlap is the maximum allowable size of two overlapping regions. This means that if two regions are overlapping by less than the percentage specified, we will not replace the one with the lower discrepancy value with the other.

Number of top-k is the maximum number of high discrepancy regions that are to be found by the Exact-Grid Top-k and Approx-Grid Top-k algorithms.

Extreme Rainfall sets the threshold percentage that the deseasonalised average rainfall must exceed to be considered extreme. In our experiments we chose the 90th percentile of values. This means that given deseasonalised average rainfall for all regions in South America, we consider only those regions whose rainfall was significantly different from the mean of all regions by more than 90%.

Region Stretch describes the number of grids to extend

beyond the outlier to check for outliers that are bounded by it in the subsequent time period, as shown in Figure 7.

Variable	Value
Allowable Overlap	10%
Number of top-k	5
Extreme Rainfall	90%
Region Stretch	5

Figure 11: Variable Setup

The following table in Figure 12 describes the subset of South American precipitation data that we use in our experiments from [9].

Variable	Value
Num Year Periods	10
Year Range	1995-2004
Grid Size	2.5°x2.5°
Num Latitudes	31
Num Longitudes	23
Total Grids	713
Num Data Values	2,609,580

Figure 12: Subset of Precipitation Data Used

5.3 Experimental Results

The application of our algorithm to South American precipitation data involved three steps described in Section 4.

The first stage of our algorithm involved finding the top-k outliers at each time step using the Exact-Grid Top-k and Approx Grid Top-k algorithms, so we can compare the two techniques. The results of this are shown in Figure 13, which show that when we set $k = 5$, we were able to find 5 top outlier regions for all years using the Exact-Grid Top-k algorithm and usually less than 5 top outlier regions using the Approx-Grid Top-k algorithm.

Year	Exact-Grid Top-k	Approx-Grid Top-k
1995	5	5
1996	5	5
1997	5	5
1998	5	5
1999	5	5
2000	5	3
2001	5	2
2002	5	3
2003	5	2
2004	5	3

Figure 13: Number of Top-k found for each year

From the second and third stages of our algorithm, we found a total of 155 outlier sequences over 10 years when using the Exact-Grid Top-k algorithm and 94 outlier sequences for the Approx-Grid Top-k algorithm. These sequences ranged from a minimum length of 1 to a maximum length of 10, since there are 10 year periods in our data subset. The results of this are summarised in Figure 14.

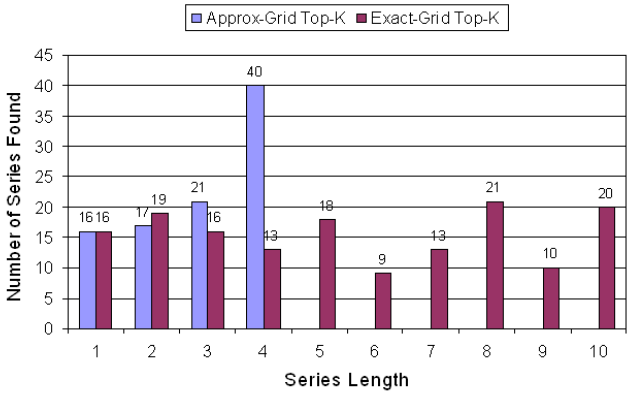


Figure 14: Length and Number of Outliers Found

Figures 15(a) and (b) show the mean discrepancy for the outlier sequences generated by the Exact-Grid Top-k and the Approx-Grid Top-k algorithms respectively. The discrepancy is plotted at the middle year of the sequence. For example, given a sequence from 1999 to 2001, the mean discrepancy for the points in the sequence will be plotted on the graphs as a point for the year 2000. Both graphs also show the mean SOI for each year. From these graphs we can see that during years where the SOI is negative, there is a lower discrepancy value, while positive SOI years coincide with a higher discrepancy value. This means that during El Niño years, identifiable by prolonged negative SOI values, the outliers are less different, as indicated by their lower discrepancy value, from surrounding regions as in non El Niño years.

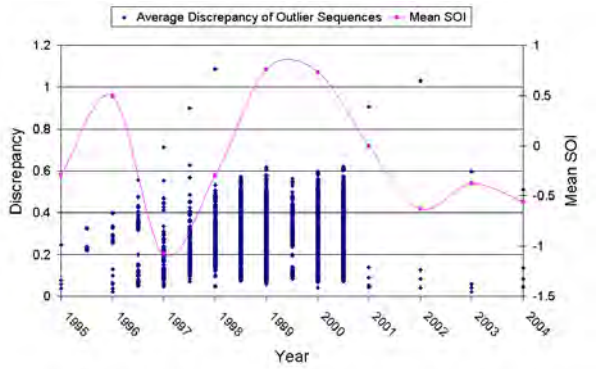
To evaluate the computational performance of Approx-Grid Top-k and Exact-Grid Top-k, we ran both algorithms over 1995 to 2004. This is shown in Figure 16, where we can see that the Approx-Grid Top-k algorithm is much faster.

6. DISCUSSION AND CONCLUSION

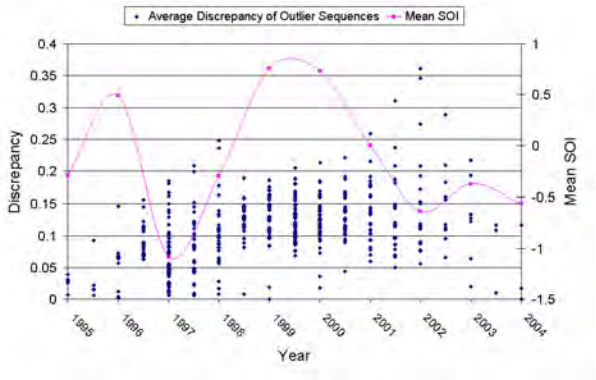
Exact-Grid Top-k and Approx-Grid Top-k algorithms are able to find the top-k high discrepancy regions from a spatial grid. We have shown that Approx-Grid Top-k is able to find a similar number of outliers from each time period, significantly faster than the Exact-Grid Top-k algorithm by approximating the discrepancy of the outlier regions. While some longer sequences could not be found by Approx-Grid Top-k, shorter sequences may be able to convey the same trend that longer sequences do. This can be seen in Figure 15 where the general trend shown against the SOI is similar for both algorithms.

We have also extended this algorithm to include the ability to discover moving spatio-temporal outlier sequences that change location, shape and size, using our Outstretch algorithm, which stores the found sequences into a tree structure. To then extract sequences from the tree, we have provided the RecurseNodes algorithm.

Our results demonstrate the successful application of our algorithm to a large precipitation data set. Given the large size of our dataset, the time taken to run the algorithm is quite good. We have shown that our algorithm is capable of finding spatial outlier sequences and subsequences that



(a) Mean discrepancy of Exact-Grid Top-k sequences and the mean SOI



(b) Mean discrepancy of Approx-Grid Top-k sequences and the mean SOI

Figure 15: Relationship between discrepancy and SOI

Exact-Grid Top-k	Approx-Grid Top-k
229s	35s

Figure 16: Time taken to discover the Top-k regions over 10 years

occur over several time periods in the South American precipitation data. In addition, we have shown one possible way of interpreting the results by comparing the behaviour of the outlier regions to the El Niño and La Niña phases of the ENSO phenomenon.

Future work could use a similar approach to ours, but instead apply it to point-data. In [1], the authors provided an algorithm called Exact to discover high discrepancy regions in point data. Additionally, with enough historical data, future work could use the historical patterns we have discovered to predict the timing, location and duration of future events.

7. REFERENCES

[1] D. Agarwal, A. McGregor, J. M. Phillips, S. Venkatasubramanian, and Z. Zhu. Spatial Scan Statistics: Approximations and Performance Study. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 24–33, 2006.

[2] D. Agarwal, J. M. Phillips, and S. Venkatasubramanian. The Hunting of the Bump: On Maximizing Statistical Discrepancy. In *Proc. 17th Ann. ACM-SIAM Symp. on Disc. Alg.*, pages 1137–1146, January 2006.

[3] D. Birant and A. Kut. Spatio-temporal outlier detection in large databases. In *28th International Conference on Information Technology Interfaces*, pages 179–184, 2006.

[4] S. Chawla, S. Shekhar, W. Wu, and U. Ozesmi. Modelling Spatial Dependencies for Mining Geospatial Data: An Introduction. *Geographic Data Mining and Knowledge Discovery, Taylor & Francis, New York, NY*, pages 131–159, 2001.

[5] T. Cheng and Z. Li. A Multiscale Approach for Spatio-Temporal Outlier Detection. *Transactions in GIS*, 10(2):253–263, 2006.

[6] J. Han, R. B. Altman, V. Kumar, H. Mannila, and D. Pregibon. Emerging Scientific Applications in Data Mining. *Communications of the ACM*, 45(8):54–58, 2002.

[7] V. Iyengar. On Detecting Space-Time Clusters. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 587–592, Seattle, WA, 2004. ACM, New York, NY.

[8] M. Kulldorff. A Spatial Scan Statistic. *Comm. in Stat.: Th. and Meth.*, 26:1481–1496, 1997.

[9] B. Liebmann and D. Allured. Daily precipitation grids for south america. *Bull. Amer. Meteor. Soc.*, 86:1567–1570, 2005.

[10] H. J. Miller. Geographic Data Mining and Knowledge Discovery. *Wilson and A. S. Fotheringham (eds.) Handbook of Geographic Information Science*, 2007.

[11] H. J. Miller and J. Han. Geographic Data Mining and Knowledge Discovery: An Overview in Geographic Data Mining and Knowledge Discovery. Taylor & Francis, New York, NY, 2001.

[12] National Oceanic and Atmospheric Administration (NOAA) Climate Prediction Center. Monthly Atmospheric & SST Indices - www.cpc.noaa.gov/data/indices [accessed: February 2, 2008].

[13] R. T. Ng. Detecting Outliers from Large Datasets *in Geographic Data Mining and Knowledge Discovery.* Taylor & Francis, New York, NY, 2001.

[14] S. Openshaw. Geographical Data Mining: Key Design Issues. In *Proceedings of GeoComputation '99*, 1999.

[15] Y. Theodoridis, J. R. O. Silva, and M. A. Nascimento. On the Generation of Spatiotemporal Datasets. In *Proceedings of the 6th International Symposium on Advances in Spatial Databases*, pages 147 – 164. Springer-Verlag, 1999.

[16] W. R. Tobler. A computer model simulation of urban growth in the Detroit region. *Economic Geography*, 46(2):234–240, 1970.

[17] E. Wu and S. Chawla. Spatio-Temporal Analysis of the relationship between South American Precipitation Extremes and the El Nino Southern Oscillation. In *Proceedings of the 2007 International Workshop on Spatial and Spatio-temporal Data Mining.* IEEE, 2007.

Probabilistic Analysis of a Large-Scale Urban Traffic Sensor Data Set

Jon Hutchins
Dept. of Computer Science
University of California, Irvine
CA 96297-3435
johutchi@uci.edu

Alexander Ihler
Dept. of Computer Science
University of California, Irvine
CA 96297-3435
ihler@ics.uci.edu

Padhraic Smyth
Dept. of Computer Science
University of California, Irvine
CA 96297-3435
smyth@ics.uci.edu

ABSTRACT

Real-world sensor time series are often significantly noisier and more difficult to work with than the relatively clean data sets that tend to be used as the basis for experiments in many research papers. In this paper we report on a large case-study involving statistical data mining of over 100 million measurements from 1700 freeway traffic sensors over a period of seven months in Southern California. We discuss the challenges posed by the wide variety of different sensor failures and anomalies present in the data. The volume and complexity of the data precludes the use of manual visualization or simple thresholding techniques to identify these anomalies. We describe the application of probabilistic modeling and unsupervised learning techniques to this data set and illustrate how these approaches can successfully detect underlying systematic patterns even in the presence of substantial noise and missing data.

Categories and Subject Descriptors

I.5 [**Pattern Recognition**]: Statistical Models; I.2.6 [**Artificial Intelligence**]: Learning—*graphical models*

General Terms

probabilistic modeling, traffic model, case study

Keywords

loop sensors, MMPP, traffic, Poisson

1. INTRODUCTION

Large-scale sensor instrumentation is now common in a variety of applications including environmental monitoring, industrial automation, surveillance and security. As one example, the California Department of Transportation (Caltrans) maintains an extensive network of over 20,000 inductive loop sensors on California freeways [1, 7]. Every 30 seconds each of these traffic sensors reports a count of the number of vehicles that passed over the sensor and the percentage of time the sensor was covered by a vehicle, measurements known as the flow and occupancy respectively. The data are continuously archived, providing a potentially rich source from which to extract information about urban transportation patterns, traffic flow, accidents, and human behavior in general.

Large-scale loop sensor data of this form are well known to transportation researchers, but have resisted systematic analysis due to the significant challenges of dealing with noisy real-world sensor data at this scale. Bickel et al. [1] outline some of the difficulties in a recent survey paper:

> ...loop data are often missing or invalid...a loop detector can fail in various ways even when it reports values...Even under normal conditions, the measurements from loop detectors are noisy...
>
> Bad and missing samples present problems for any algorithm that uses the data for analysis...we need to detect when data are bad and discard them
>
> A systematic and principled algorithm [for detecting faulty sensors] is hard to develop mainly due to the size and complexity of the problem. An ideal model needs to work well with thousands of detectors, all with potentially unknown types of malfunction.
>
> Even constructing a training set is not trivial since there is so much data to examine and it is not always possible to be absolutely sure if the data are correct even after careful visual inspection.

Similar issues arise in many large real-world sensor systems. In particular, the presence of "bad" sensor data is a persistent problem—sensors are often in uncontrolled and relatively hostile environments, subject to a variety of unknown and unpredictable natural and human-induced changes. Research papers on sensor data mining and analysis often pay insufficient attention to these types of issues; for example, our previous work [4, 5] did not address sensor failures directly. However, if research techniques and algorithms for sensor data mining are to be adapted and used for real-world problems it is essential that they can handle the challenges of such data in a robust manner.

In this paper we present a case study of applying probabilistic sensor modeling algorithms to a data set with 2263 loop

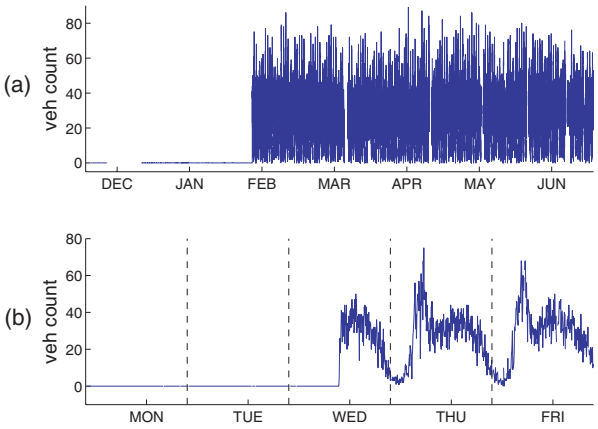


Figure 1: (a) A sensor that is stuck at zero for almost two months. (b) Five days of measurements at the end of the period of sensor failure, after which a typical pattern of low evening activity and higher activity at morning and afternoon rush hour begins to appear.

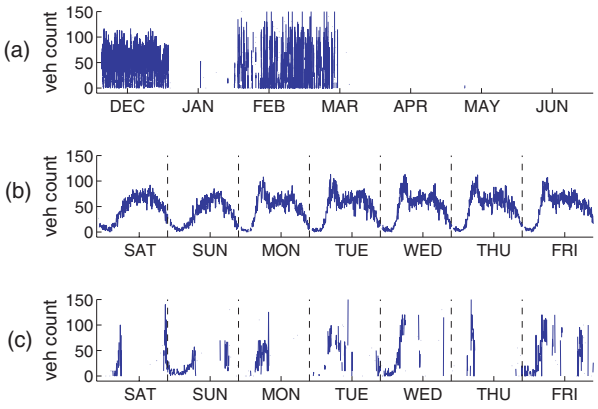


Figure 2: (a) A sensor with normal (periodic) initial behavior, followed by large periods of missing data and suspicious measurements. (b) A week at the beginning of the study showing the periodic behavior typical of traffic. (c) A week in February. Other than the missing data, these values may not appear that unusual. However, they are not consistent with the much clearer pattern seen in the first two months. The presence of unusually large spikes of traffic, particularly late at night, also make these measurements suspicious.

sensors involving over 100 million measurements, recorded over seven months in Southern California. The sensor modeling algorithms that we use are based on unsupervised learning techniques that simultaneously learn the regular patterns of human behavior from data as well as the occurrence of unusual events, as described in our previous work [4, 5].

The seven months of time-series data from the 2263 loop sensors contain a wide variety of anomalous behavior including "stuck at zero" failures, missing data, suspiciously high readings, and more. Figure 1 shows a sensor with a "stuck at zero" failure, and Figure 2 shows an example of a sensor with extended periods both of missing data and of suspicious measurements. In this paper we focus specifically on the challenges involved in working with large numbers of sensors having diverse characteristics. Removing bad data via visual inspection is not feasible given the number of sensors and measurements, notwithstanding the fact that it can be non-trivial for a human to visually distinguish good data from bad. In Figure 2, for example, the series of measurements between January and March might plausibly pass for daily traffic variations if we did not know the normal conditions. Figure 2 also illustrates why simple thresholding techniques are generally inadequate, due to the large variety in patterns of anomalous sensor behavior.

We begin the paper by illustrating the results of a probabilistic model that does not include any explicit mechanism for handling sensor failures. As a result, the unsupervised learning algorithms fail to learn a pattern of normal behavior for a large number of sensors. We introduce a relatively simple mechanism into the model to account for sensor failures, resulting in a significant increase in the number of sensors where a true signal can be reliably detected, as well as improved automatic identification of sensors that are so inconsistent as to be unmodelable. The remainder of the paper illustrates how the inferences made by the fault-tolerant model can be used for a variety of analyses, clearly distinguishing (a) the predictable hourly, daily, and weekly rhythms of human behavior, (b) unusual bursts of event traffic activity (for example, due to sporting events or traffic accidents), and (c) sequences of time when the sensor is faulty. We conclude the paper with a discussion of lessons learned from this case study.

2. LOOP SENSOR DATA

We focus on the flow measurements obtained from each loop sensor, defined as the cumulative count of vehicles that passed over the sensor. The flow is reported and reset every 30 seconds, creating a time series of count data. As shown in Figures 1 and 2, the vehicle count data is a combination of a "true" periodic component (e.g., Figure 2(b)) and a variety of different types of failures and noise.

We collected flow measurements between November 26, 2006 and July 7, 2007 for all of the entrance and exit ramps in Los Angeles and Orange County. The data were downloaded via ftp from the PeMS database [1, 7] maintained by U.C. Berkeley in cooperation with Caltrans. Of the 2263 loop sensors, 566 sensors reported missing (no measurement reported) or zero values for the entire duration of the seven month study. The remaining 1716 sensors reported missing measurements 29% of the time on average. Missing data occurred either when PeMS did not report a measurement (either due to a faulty detector or a faulty collection system), or when our own system was unable to access PeMS. Such measurements are explicitly flagged as missing and are known to our model.

Aside from missing measurements and sensor failures, the periodic structure in the data reflecting normal (predictable) driving habits of people can be further masked by periods of unusual activity [5]; including those caused by traffic accidents or large events such as concerts and sporting events.

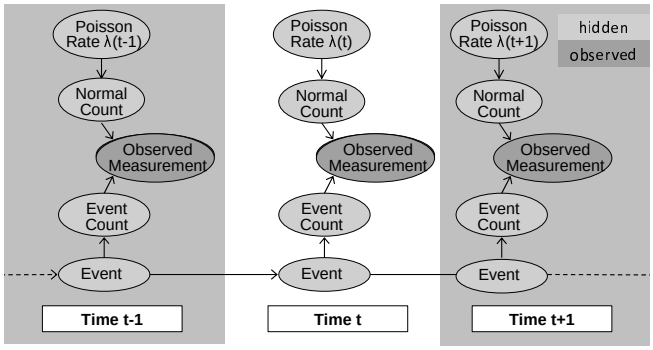


Figure 3: Graphical model of the original approach proposed in [4, 5]. Both the event and rate variables couple the model across time: the Markov event process captures rare, persistent events, while the Poisson rate parameters are linked between similar times (arrows not shown). For example, the rate on a particular Monday during the 3:00 to 3:05pm time slice is linked to all other Mondays at that time.

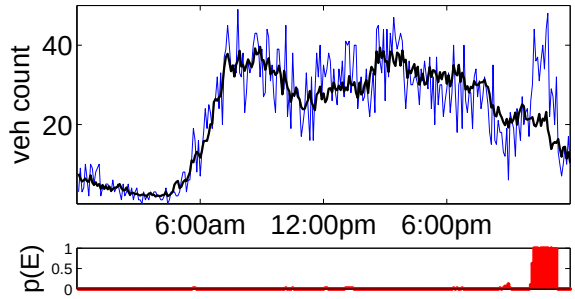


Figure 4: Example inference results with the model from Figure 3. The blue line shows actual flow measurements for one sensor on one day, while the black line is the model's inferred rate parameters for the normal (predictable) component of the data. The bar plot below shows the estimated probability that an unusual event is taking place.

Noisy measurements, missing observations, unusual event activity, and multiple causes of sensor failure combine to make automated analysis of a large number of sensors quite challenging.

3. ORIGINAL MODEL

In our earlier work [5] we presented a general probabilistic model (hereafter referred to as the original model) that can learn patterns of human behavior that are hidden in time series of count data. The model was tested on two real-world data sets, and was shown to be significantly more effective than a baseline method at discovering both underlying recurrent patterns of human behavior as well as finding and quantifying periods of unusual activity. Our original model consisting of two components: (a) a time-varying Poisson process that can account for recurrent patterns of behavior, and (b) an additional "bursty" Poisson process, modulated by a Markov process, that accounts for unusual events.

If labeled data are available (i.e. we have prior knowledge of the time periods when unusual events occur), then estimation of this model is straightforward. However, labeled data are difficult to obtain and are likely to be only partially available even in a best-case scenario. Even with close visual inspection it is not always easy to determine whether or not event activity is present. Supervised learning (using labeled data) is even less feasible when applying the model to a group of 1716 sensors.

Instead, in our earlier work [4, 5], we proposed separating the normal behavior and the unusual event behavior using an unsupervised Markov modulated Poisson process [9, 10]. The graphical model is shown in Figure 3. The normal (predictable) component of the data is modeled using a time-varying Poisson process, and the unusual event activity is modeled separately using a Markov chain. The event variable can be in one of three states: no event, positive event (indicating unusually high activity), or negative event (unusually low activity).

In the model, the Poisson rate parameter defines how the normal, periodic behavior counts are expected to vary, while the Markov chain component allows unusual events to have persistence. If the observed measurement is far from the rate parameter, or if event activity has been predicted in the previous time slice, the probability of an event increases.

Given the model and the observed historical counts, we can infer the unknown parameters of the model (such as the rate parameters of the underlying normal traffic pattern) as well as the values of the hidden states. Note that in addition to the event state variables being connected in time, the rate parameters $\lambda(t)$ are also linked (not shown in Figure 3). This leads to cycles in the graphical model, making exact inference intractable. Fortunately, there are approximation algorithms that are effective in practice. We use a Gibbs sampler [3] for learning the hidden parameters and hidden variables [4, 5]. The algorithm uses standard hidden Markov recursions with a forward inference pass followed by a backwards sampling pass for each iteration of the sampler. The computational complexity of the sampler is linear in the number of time slices, and empirical convergence is quite rapid (see [5] for more details).

Figure 4 shows an example of the results of the inference procedure. The measured vehicle count for this particular day follows the model's inferred time-varying Poisson rate for the normal (predictable) component of the data for most of the day. In the evening, however, the observed measurements deviate significantly. This deviation indicates the presence of unusual event activity and is reflected in the model's estimated event probability (bottom panel). The output of the model also includes information about the magnitude and duration of events.

The event activity in this example looks obvious given the inferred profile of normal behavior (the bold curve in Figure 4); however, simultaneously identifying the normal pattern and unusual event activity hidden within the measurements is non-trivial. In our earlier work [4, 5] we found that the Markov-modulated Poisson process was significantly more accurate at detecting known events than simpler baseline

Event Fraction	Number of Sensors
0 to 10%	912
10 to 20%	386
20 to 50%	265
50 to 100%	153

Table 1: Original model's fit. The study's 1716 sensors are categorized using a measure of the model's ability to find a predictable periodic component in the sensor measurements (if present). The *event fraction* is defined as the fraction of time a sensor's measurements are classified as a positive or negative event. For sensors with lower event fractions, the model has found a strong periodic component with fewer periods of unusual event activity.

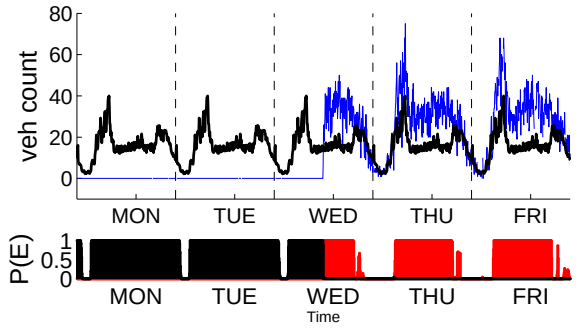


Figure 5: Original model output for the sensor in Figure 1. Shown are the observed measurements (blue) for one week (the same week as Figure 1(b)) along with the model's inferred Poisson rate (black). With a long period stuck at zero, a poor model is inferred for normal behavior in the middle of the day. This is reflected in the event probabilities (bottom), where unusual event activity is predicted for most of each day.

methods such as threshold-type detectors based on Poisson models.

4. SCALE-UP CHALLENGES

After the initial successes described in Section 3, we wanted to test the model on a much larger network of sensors. The model can generally be applied to various types of sensors that record count data, but the loop sensor data set was a particularly appealing choice for a case study. As mentioned earlier, we had access to the measurements of a large selection of loop sensors in the Southern California area. We also had additional information about the sensors that could prove useful during analysis, such as their geographic location and whether each sensor was on an exit or entrance ramp. In addition, there are many data analysis problems specific to traffic data, including accident detection, dynamic population density estimation, and others. In our work we were motivated by the challenge of extracting useful information from this large data set to provide a basic framework for addressing these questions.

We applied the original model to the data from our seven month study involving 1716 sensors and over 100 million hidden variables. The model was subjected to much greater levels of variability than experienced in our earlier studies. Several weaknesses of the original model were identified as a result.

Table 1 shows one method for judging how well the model fit the data. The table shows the fraction of time that the model inferred unusual event activity for each of the sensors during our seven month study, i.e. the fraction of time slices in which the event variable in Figure 3 was inferred to be in an unusual event state and are thus not explained by the periodic component of the data.

There is reason to be suspicious when the model infers unusual event activity for a large fraction of the time, especially in cases where unusual event activity is more common than normal activity (as in the last row of the table). A review of the sensors where event activity was inferred over 50% of the time revealed some weaknesses of the original model. Some sensors in this category were apparently faulty throughout the study. Another group of sensors recorded non-missing measurements for only a very small fraction of the study, which were not enough to form a good model. However, there were many sensors which appeared to have an underlying periodic behavior pattern that was missed by the original model.

The sensor with the "stuck at zero" failure (Figure 1) is an example of a sensor with a clear periodic pattern that the original model missed. Figure 5 shows the model's attempt to fit the data from this sensor. The model is able to learn early morning and late night behavior, but an inaccurate profile is inferred for normal behavior in the middle of the day. Examples such as this were observed across many other sensors, and in many cases where a poor model was inferred for normal behavior there appeared to be long periods where the sensor was faulty.

We experimented with a number of modifications to the model, including adjusting the priors on the parameters of the Markov process, avoiding poor initialization of the Gibbs sampler which sometimes occurred when extensive periods of failure were present, and dealing with missing data differently. These adjustments improved the performance of the model in some cases. But in many cases (particularly in sensors with extensive periods of sensor failure) inaccurate profiles were still inferred for normal behavior.

5. FAULT-TOLERANT MODEL

It is clear from Section 4 that in order to make the original model more general and robust, sensor failures should be addressed directly instead of bundling them together with unusual event activity. We note that heuristic approaches to sensor fault detection in traffic data have been developed in prior work [2, 6], but these techniques are specific to loop detectors and to certain types of sensor failures. Our focus in this paper is on developing an approach that can handle more general types of faults, not only in loop sensor data but also in other sensors that measure count data.

One possible approach to solve these problems would be to modify the model to broaden the definition of "events" to include sensor failures. However, sensor failures and events

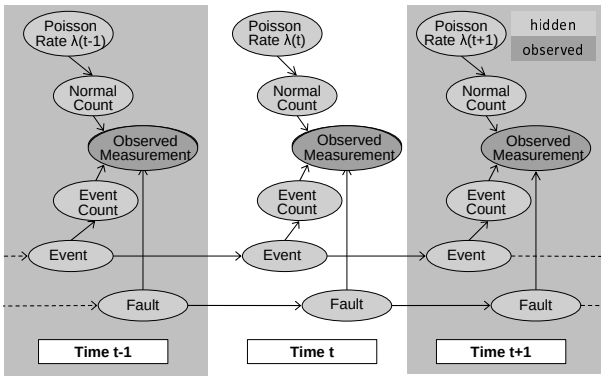


Figure 6: Graphical model of the fault-tolerant method. The fault-tolerant model is very similar to the original model (Figure 3), but with an additional Markov chain added to model sensor failure.

(as we define them) tend to have quite different characteristic signatures. Events tend to persist for a few hours while failures often have a much broader range of temporal duration. Events also tend to be associated with a relatively steady change (positive or negative) in count rates over the duration of the event, while failures can have significant variability in count rates during the duration of the fault. Ultimately, while there is not a crisp boundary between these two types of non-normal measurements, we will show in Sections 6 and 7 that both types are sufficiently different and prominent in the data to merit separate treatment.

When we detect sensor failures visually, we are in essence recognizing extensive periods of time where the periodic structure that we have come to expect is not present. This reasoning is built into a new model, defined by the graphical model in Figure 6. The original model (Figure 3) has been modified to include an additional Markov chain for failure modes. This failure state is a binary variable indicating the presence or absence of a sensor failure.

If the state variable in the event chain is in an event state, the observed measurement is accounted for by both the normal and event count components. If the state variable in the fault chain is in the fault state, however, the observed measurement is treated as if it were missing. This allows our model to ignore the faulty part of the data when inferring the time-varying Poisson (normal) component of the data.

In a Bayesian framework, our belief about the relative duration of events and failures can be encoded into the priors that are put on the transition parameters for the two Markov chains. We expect events to be short, ranging from a few minutes to a couple of hours; in contrast, we expect failures to be relatively lengthy, ranging from days to months.

The modular nature of graphical models makes it possible to extend the original model without starting from scratch. Some modifications were made to the inference calculations when the fault sensor process was added to the original model, but learning and inference proceeds in the same general fashion as before (see [4, 5] for details). From a practical viewpoint, relatively few changes to the software code were needed to extend the model to include a failure state. Details of these changes are given in the Appendix.

Event Fraction	Original Model Number of Sensors	Fault-tolerant Model Number of Sensors
0 to 10%	960	1285
10 to 20%	375	242
20 to 50%	244	117
50 to 100%	137	72

Table 2: Comparison of the fraction of time in the event state for the original and fault-tolerant models. We have excluded missing data and times detected as faulty from the percentage calculation for both models. While there is a slight shift to the upper rows of the table for the original model (compared to Table 1), we see a significantly greater shift for the fault-tolerant model, indicating that it has done a better job of inferring the true periodic structure underlying the data.

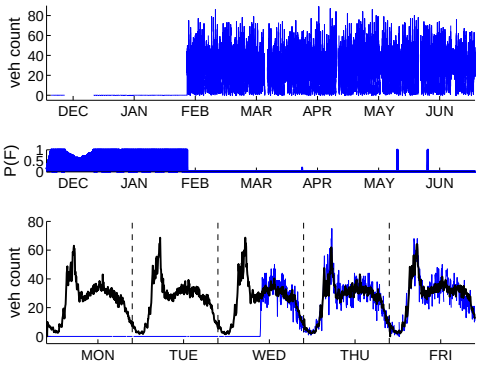


Figure 7: Fault-tolerant model results for the "stuck at zero" example (refer to Figures 1 and 5 and Section 4). Shown are the raw data (top), the model's estimate of the probability of a faulty sensor (center), and the inferred time-varying Poisson rate of the normal component (bottom). The model detects the "stuck at zero" failure, and the model's rate fits the periodic signal in the data that was missed by the original model (Figure 5).

6. EXPERIMENTAL RESULTS

Table 2 gives a sense of the gains made by the fault-tolerant model compared to the original model. We compare the percentage of time spent in an event state for sensors under each model. In order to provide a fair comparison, missing measurements as well as measurements detected as faulty by the fault-tolerant model were removed for *both* models before calculating the event fraction of the remaining measurements predicted by each model. The fault-tolerant model is able to infer a better model of the periodic structure hidden within the flow data. This is apparent in the shift of sensors to the upper rows of the table where unusual activity is detected less frequently and thus more of the data is explained by the time-varying Poisson process. Of the 381 sensors in the >20% range for event fraction with the original model, only 189 remain under the fault-tolerant model, a 50% reduction.

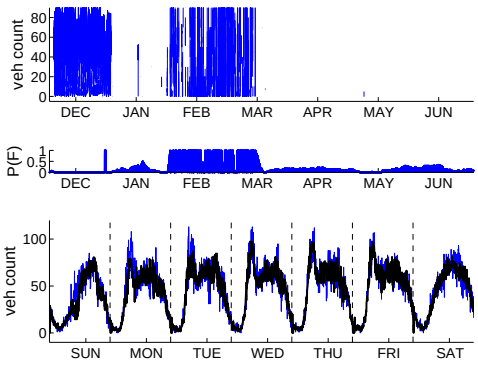


Figure 8: Fault-tolerant model results for the corrupted signal example (refer to Figure 2). The corrupted portion of the signal is detected (center), and the model's inferred time-varying Poisson rate (bottom) fits the periodic signal present in the first months of the study.

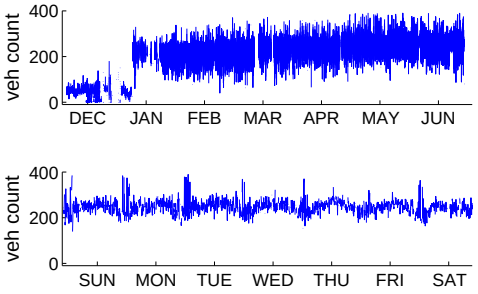


Figure 9: Sensor with a corrupt signal. This sensor appears to be faulty for the entire duration of our study. There is no consistent periodic pattern to the signal, and large spikes often occur in the middle of the night when little traffic is expected.

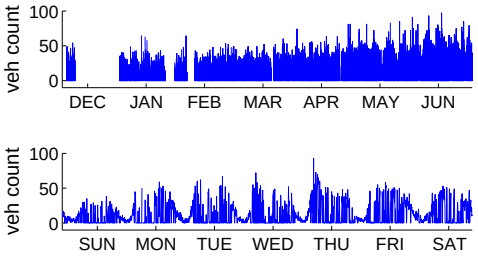


Figure 10: Sensor signal with no consistent periodic component. There may be some periodic structure within a particular week (bottom panel), but there appears to be no consistent week-to-week pattern.

Figure 7 is a plot of the same sensor as in Figures 1 and 5, where the original model was not able to find the normal traffic behavior during the middle of the day. The fault-tolerant model was able to detect the "stuck at zero" failure at the beginning of the study and find a much more accurate model of normal behavior.

Figure 8 shows the performance of the fault-tolerant model for a different type of failure. This is the same sensor shown earlier in Figure 2, where the measurements display periodic behavior followed by a signal that appears to be corrupted. During this questionable period, the measurements are missing more often than not, and unusually large spikes (many 50% higher than the highest vehicle count recorded during the first two months of the study) at unusual times of the day are often observed when the signal returns. The fault-tolerant model can now detect the corrupted signal and also in effect removes the faulty measurements when inferring the time-varying Poisson rate.

In the 50% to 100% row of the table, there are still a number of sensors where the fault-tolerant model is not able to discover a strong periodic pattern. About half of these sensors had large portions of missing data with too few non-missing measurements to form a good model. Others such as seen in Figures 9 and 10, had no consistent periodic structure. Figure 9, is an example of a sensor that appears to be faulty for the duration of our study. The measurements for the sensor in Figure 10, on the other hand, appear to have some structure; morning rush hour with high flow, and low flow in the late evening and early morning as expected. However, the magnitude of the signal seems to alter significantly enough from week to week so that there is no consistent "normal" pattern. Even though the non-zero measurements during the day could perhaps be accurate measurements of flow, the unusual number of measurements of zero flow during the day along with the weekly shifts make the sensor output suspicious.

Before performing our large-scale analysis, we pruned some highly suspicious sensors. With most sensors, the fault-tolerant model makes a decent fit, and can be used to parse the corresponding time-series count data into normal, event, fault, and missing categories, and the results can be used in various analyses. When the model gives a poor fit (Figures 9 and 10 for example), the parsed data can not be trusted, and may cause significant errors in later analysis if included. So, the outputs of such models (and the corresponding sensors) need to be excluded.

We used the information found in Table 2 to prune our sensor list, and limited our evaluation to the 89% of the sensors that predicted less than 20% unusual event activity. The retained sensors sufficiently cover the study area of Los Angeles and Orange County, as seen in Figure 11. Removing sensors with questionable signals visually, without the use of a model, is not practical. Our model allows us to prune away sensors for which the model can not make any sense in an automated way.

7. LARGE-SCALE ANALYSIS

After pruning the sensor list, 1508 sensor models remain, which together have learned normal, predictable, traffic behavior for approximately 9 million vehicle entrances and exits to and from the freeways of Los Angeles and Orange County. During the seven month study, these models detected over 270,000 events and almost 13,000 periods of sensor failure. Sensors saw unusual event activity approximately once every 30 hours on average, and saw sensor failure once every 26 days on average.

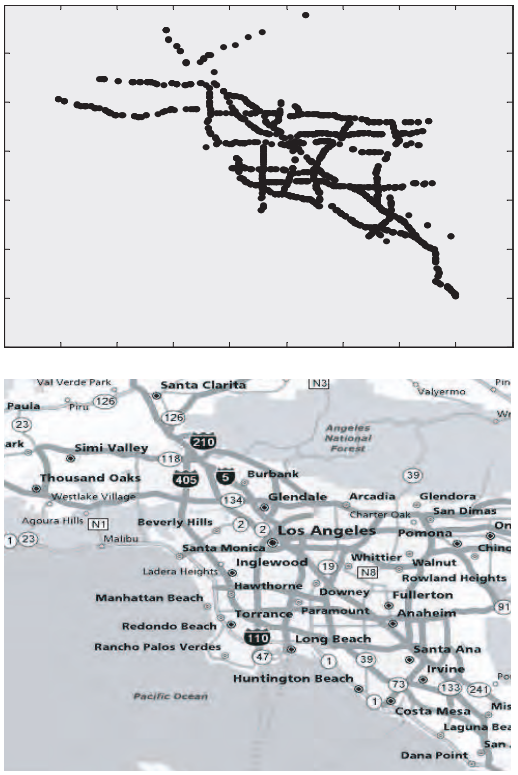


Figure 11: The locations of the sensors used in our large-scale analysis which remain after pruning "suspicious" sensors as described in Section 6 (top), and a road map (bottom) of our study area, Los Angeles and Orange County.

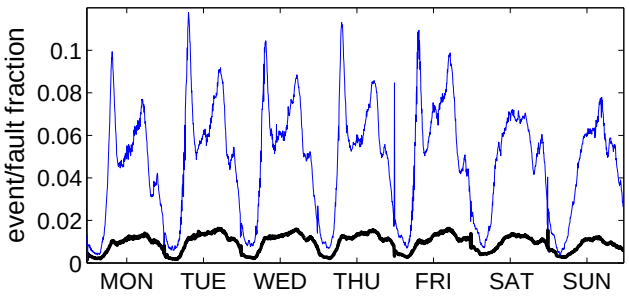


Figure 12: Unusual event frequency and fault frequency. The thin blue line with the greater magnitude shows the fraction of time that events were detected as a function of time of day, while the thick black line shows the fraction of time that faults were detected.

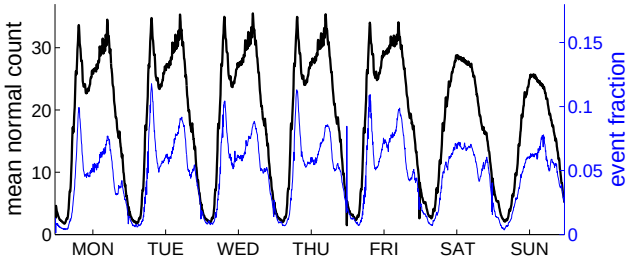


Figure 13: The event fraction (thin blue line, smaller magnitude, right y-axis) is plotted alongside the mean normal vehicle flow profile (the inferred Poisson rate averaged across the sensors) shown as the thick black line and using the left y-axis. The profiles are similar, with a correlation coefficient of 0.94.

After observing almost 300,000 periods of unusual and faulty activity, the first question we ask is: On what day of the week and at what time of the day is it most common to see unusual event activity? Figure 12 shows a plot of the frequencies of unusual events and of sensor failures as a function of time of day and day of week. Sensor failures do not appear to have much of a pattern during the day. The troughs at nighttime reflect a limitation of our fault model to detect failures at night when there is little or no traffic. Chen et al. [2] also found it difficult to reliably detect failure events in loop sensor data at night and as a consequence limited fault detection to the time-period between 5am and 10pm.

Of more interest, the frequency of unusual event activity in Figure 12 does have a strong pattern that appears proportional to normal traffic patterns. That is, weekdays have spikes in unusual activity that appear to correspond to morning and afternoon rush hour traffic. The event pattern and the normal traffic flow pattern are compared in Figure 13. There is a strong relationship between the two (correlation coefficient 0.94), although there are significant bumps in the event activity each evening, particularly on weekend evenings, that depart from the normal flow pattern.

To explain the shape of the event fraction curve in Figure 13, it is reasonable to consider two types of event activity: events correlated with flow and events independent of flow. Traffic accidents might fall into the correlated event type, because one would expect an accident on the freeway or on an artery close to a ramp to affect traffic patterns more when there is already heavy traffic. Much less of a disruption is expected if the accident occurs in the middle of the night. Traffic from large sporting events, which often occur in the evening, might fit the second type of event that is not correlated with traffic flow since the extra traffic is not primarily caused by people trying to escape traffic congestion.

Also of note in Figure 13 is that the flow patterns for weekdays look very similar. In Figure 14(a), the inferred time-varying Poisson rate profile for normal activity, averaged across all 1508 sensors, for each week day are plotted on top of each other. This figure shows that the average normal traffic pattern does not vary much between Monday and Friday. Note that in the fault-tolerant model used for the scale up experiments, there is no information-sharing between weekdays, so there is nothing in the model that would influence one weekday to look similar to another. The similarity is much less clear in the raw data (Figure 14(b)).

In Figure 14(a) there is also evidence of bumps occurring at regular intervals, especially in the morning and late afternoon. To investigate if the model was accurately reflecting

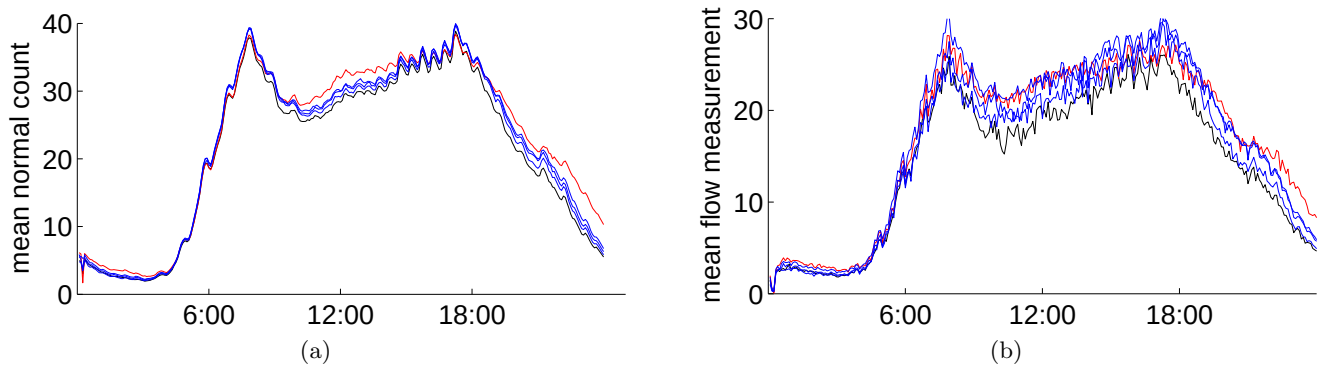


Figure 14: (a) The average Poisson rate (across all sensors) for each weekday, superimposed. Although nothing links different weekdays, their profiles are quite similar, and the oscillation during morning and afternoon rush hour is clearly visible. (b) The mean vehicle flow rates for each weekday (average of raw measurements over all sensors), superimposed. The similarity in patterns is far less clear than in panel (a).

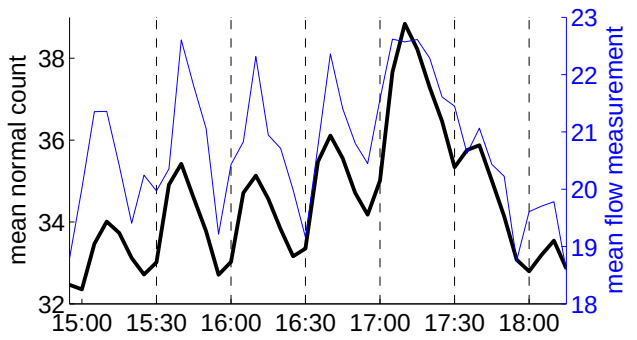


Figure 15: The mean normal vehicle profile (as in Figure 14) shown by the thick black line (using the left y-axis), is plotted against the actual mean flow (light blue line, right y-axis) for Mondays between 3pm and 5pm. The bumps that occur regularly at 30-minute intervals in the model's inferred time-varying Poisson rate are also present in the raw data.

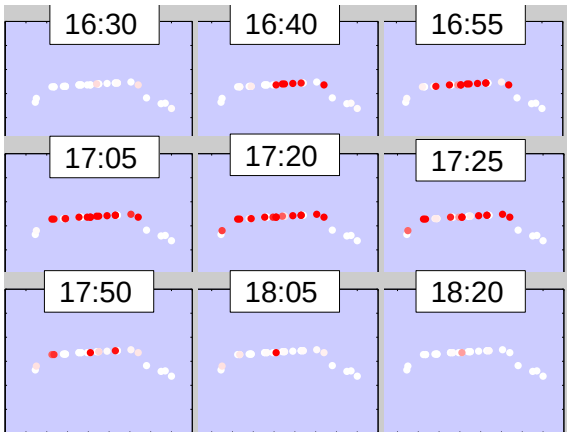


Figure 16: Example of a spatial event that occurs along a stretch of Interstate 10 in Los Angeles. Each circle is a sensor on an exit or entrance ramp. It is light colored when no unusual event activity was inferred by the sensor's model over the past 5 minutes, and is darker as the estimated probability of an unusual event (inferred by the model) increases. The 9 snapshots span a nearly two hour period where unusual activity spreads out spatially then recedes.

a true behavior, we plotted the raw flow measurements for each weekday to compare with the model prediction. Figure 15 shows the plot of raw data and model profile for Monday, zoomed in on the afternoon period where the phenomenon is more pronounced. The raw data generally follows the same pattern as the model, confirming that these oscillations are not an artifact of the model. Interestingly, weekend days do not experience this behavior; and when individual ramps were examined, some showed the behavior and some did not. The peaks of the bumps appear regularly at 30 minute intervals. One plausible explanation [8] is that many businesses are located close to the highway, and people generally report to work and leave work on the half hour and on the hour; the bumps are caused by people getting to work on time and leaving work.

Note that this type of discovery is not easy to make with the raw data. In Figure 14(b), the mean flow profiles for the weekdays appear to be potentially different because events and failures corrupt the observed data and mask true patterns of normal behavior. It is not easy to see how similar these daily patterns are, and the half hour bumps in common between the days (Figure 15) are less likely to be spotted. An important point here is that the model (in Fig 14(a)) has automatically extracted a clear signal of normal behavior, a signal that is buried in the raw data (Fig 14(b)).

Lastly, we present an example of spatial analysis of the model output. Figure 16 shows an example of a "spatial event". The series of plots span a two hour period beginning with a plot of one ramp seeing unusual activity, followed by plots showing a spread of unusual activity detection. At its height, the unusual event activity spans a seven mile stretch of Interstate 10 in Los Angeles, which is followed by a gradual reduction of unusual event activity. One can imagine using information such as this to find the extent of disruption caused by an accident.

8. CONCLUSIONS

We have presented a case study of a large-scale analysis of an urban traffic sensor data set in Southern California. 100 million flow measurements from 1700 loop detectors over a period of seven months were parsed using a probabilistic model into normal activity, unusual event activity, and sensor failure components. The model provides a useful and general framework for systematic analysis of large noisy sensor data sets. In particular, the model was able to provide useful insights about an urban traffic data set that has been considered difficult to analyze in the past. Future work could include linking the sensors spatially, and extending the model to detect the spatial and temporal effect of events such as traffic accidents. Other future work could include use of the occupancy values measured by loop sensors in addition to the flow measurements, or making use of census information for dynamic population estimation.

Acknowledgements

This material is based upon work supported in part by the National Science Foundation under Award Numbers ITR-0331707, IIS-0431085, and IIS-0083489.

9. REFERENCES

[1] P. Bickel, C. Chen, J. Kwon, J. Rice, E. van Zwet, and P. Varaiya. Measuring traffic. *Statistical Science*, 22(4):581–597, 2007.

[2] C. Chen, J. Kwon, J. Rice, A. Skabardonis, and P. Varaiya. Detecting errors and imputing missing data for single-loop surveillance systems. *Transportation Research Record*, 1855:160–167, 2003.

[3] A. Gelman. *Bayesian Data Analysis*. CRC Press, 2004.

[4] A. Ihler, J. Hutchins, and P. Smyth. Adaptive event detection with time-varying Poisson processes. In *ACM Int'l Conf. Knowledge Discovery and Data mining*, pages 207–216, 2006.

[5] A. Ihler, J. Hutchins, and P. Smyth. Learning to detect events with Markov-modulated poisson processes. *TKDD*, 1(3), 2007.

[6] L. N. Jacobson, N. L. Nihan, and J. D. Bender. Detecting erroneous loop detector data in a freeway traffic management system. *Transportation Research Record*, 1287:151–166, 1990.

[7] PeMS. Freeway Performance Measurement System. http://pems.eecs.berkeley.edu/.

[8] W. Recker and J. Marca. Institute of Transportation Studies, UC Irvine, personal communication.

[9] S. Scott. *Bayesian Methods and Extensions for the Two State Markov Modulated Poisson Process*. PhD thesis, Harvard University, 1998.

[10] S. Scott and P. Smyth. The Markov modulated Poisson process and Markov Poisson cascade with applications to web traffic data. *Bayesian Statistics*, 7:671–680, 2003.

APPENDIX

Using the notation and inference procedure described in our earlier work [5], we explain below the necessary modifications for the fault-tolerant extension of the model.

We use a binary process $f(t)$ to indicate the presence of a failure, i.e., $f(t) = 1$ if there is a sensor failure at time t, and 0 otherwise. We define the probability distribution over $f(t)$ to be Markov in time, with transition probability matrix

$$M_f = \begin{pmatrix} 1-f_0 & f_0 \\ f_1 & 1-f_1 \end{pmatrix}.$$

We put Beta distribution priors on f_0 and f_1:

$$f_0 \sim \beta(f; a_0^F, b_0^F) \qquad f_1 \sim \beta(f; a_1^F, b_1^F).$$

In the sampling procedure for the hidden variables given the parameters, the conditional joint distribution of $z(t)$ (the event process) and $f(t)$ is computed using a forward-backward algorithm. In the forward pass we compute $p(z(t), f(t)|\{N(t'), t' t\})$ using the likelihood functions

$$p(N(t)|z(t), f(t)) = \begin{cases} \mathrm{P}(N(t); \lambda(t)) & z(t) = 0, f(t) = 0 \\ \sum_i \mathrm{P}(N(t) - i; \lambda(t)) \, \mathrm{NBin}(i) & z(t) = +1, f(t) = 0 \\ \sum_i \mathrm{P}(N(t) + i; \lambda(t)) \, \mathrm{NBin}(i) & z(t) = -1, f(t) = 0 \\ \mathrm{U}(N(t); N_{\max}) & \text{otherwise} \end{cases}$$

where $N_{\max}$ is the largest observed flow measurement and $\mathrm{U}(N(t); N_{\max})$ is the uniform distribution over $[0 \ldots, N_{\max}]$.

If a failure state is not sampled ($f(t) = 0$), $N_0(t)$ and $N_E(t)$ are sampled as in [5]. However, if a failure state is sampled ($f(t) = 1$), the observed data is treated as missing.

In our previous work [5], $N_0(t)$ and $N_E(t)$ were sampled if the measurement was missing. The fault-tolerant model does not sample $N_0(t)$ and $N_E(t)$ when the data is missing to avoid slow mixing of the Gibbs sampler for sensors with extensive periods of missing data.

By not sampling $N_0(t)$ for missing time slices, the time-varying Poisson rate parameter can no longer be decomposed into day, week, and time-of-day components as in [5]. Instead, a rate parameter is learned for each of the 2016 unique 5-minute time periods of the week. The Poisson rate parameters have prior distributions

$$\lambda_{i,j} \sim \Gamma(\lambda; a_{i,j}^L = 0.05, b_{i,j}^L = 0.01)$$

where i takes on values $\{1, \ldots, 7\}$ indicating the day of the week and j indicates the time-of-day interval $\{1, \ldots, 288\}$.

We used Dirichlet priors for the rows of the Markov transition matrix for the event process (Z):

$$\begin{pmatrix} a_{00}^Z & a_{01}^Z & a_{02}^Z \\ a_{10}^Z & a_{11}^Z & a_{12}^Z \\ a_{20}^Z & a_{21}^Z & a_{22}^Z \end{pmatrix} = \begin{pmatrix} .999 & .0005 & .0005 \\ .14 & .85 & .01 \\ .14 & .01 & .85 \end{pmatrix} \times 10^6$$

The Beta parameters for the transition matrix of the fault process (F) were:

$$\begin{pmatrix} a_0^F & b_0^F \\ a_1^F & b_0^F \end{pmatrix} = \begin{pmatrix} .00005 & .99995 \\ .0005 & .9995 \end{pmatrix} \times 10^6$$

Strong priors are used for the Markov transition parameters in MMPPs [9] to prevent the model from trying to explain normal sensor noise with the Markov component. The priors above ensure reasonable frequencies and durations for inferred events and faults.

Monitoring Incremental Histogram Distribution for Change Detection in Data Streams

Raquel Sebastião
LIAAD - INESC Porto, L.A.
and Faculty of Science,
University of Porto
Rua de Ceuta, 118, 6
4050-190 Porto, Portugal
raquel@liaad.up.pt

João Gama
LIAAD - INESC Porto, L.A.
and Faculty of Economics,
University of Porto
Rua de Ceuta, 118, 6
4050-190 Porto, Portugal
jgama@liaad.up.pt

Pedro Pereira Rodrigues
LIAAD - INESC Porto, L.A.
and Faculty of Science,
University of Porto
Rua de Ceuta, 118, 6
4050-190 Porto, Portugal
pprodrigues@fc.up.pt

João Bernardes
Faculty of Medicine, University
of Porto and INEB, Porto
Alameda Prof. Hernâni
Monteiro
4200 - 319 Porto, Portugal
joaobern@med.up.pt

ABSTRACT

Histograms are a common technique for density estimation and they have been widely used as a tool in exploratory data analysis. Learning histograms from static and stationary data is a well known topic. Nevertheless, very few works discuss this problem when we have a continuous flow of data generated from dynamic environments.

The scope of this paper is to detect changes from high-speed time-changing data streams. To address this problem, we construct histograms able to process examples once at the rate they arrive. The main goal of this work is continuously maintain a histogram consistent with the current status of the nature. We study strategies to detect changes in the distribution generating examples, and adapt the histogram to the most recent data by forgetting outdated data. We use the Partition Incremental Discretization algorithm that was designed to learn histograms from high-speed data streams.

We present a method to detect whenever a change in the distribution generating examples occurs. The base idea consists of monitoring distributions from two different time windows: the reference window, reflecting the distribution observed in the past; and the current window which receives the most recent data. The current window is cumulative and can have a fixed or an adaptive step depending on the distance between distributions. We compared both distributions using Kullback-Leibler divergence, defining a threshold for change detection decision based on the asymmetry of this measure.

We evaluated our algorithm with controlled artificial data sets and compare the proposed approach with nonparametric tests. We also present results with real word data sets from industrial and medical domains. Those results suggest that an adaptive window's size exhibit high probability in change detection and faster detection rates, with few false positives alarms.

ACM SIGKDD '08 Las Vegas, Nevada USA

Keywords

Change detection, Data streams, Machine learning, Learning histograms, Monitoring data distribution, Adaptive Cumulative Windows

1. INTRODUCTION

Nowadays, the scenario of finite stored data sets is no longer appropriated because information is gathered assuming the form of transient and infinite data streams. As a large massive amount of information is produced at a high-speed rate it is no longer possible to use algorithms which require to store, in the main memory, the full historic data. In Data Streams the data elements are continuously received, treated and discarded. In this context processing time, memory and sample size are the crucial constraints in knowledge discovery systems [3]. Due to the exploratory nature of data and to time restrictions an exact answer may be not required: a user may prefer a fast but approximate answer to an exact but slow answer. Methods to deal with these issues consist of applying synopsis techniques, such as histograms [12, 16, 19, 27], sketches [8] and wavelets [7, 15]. Histograms are one of the techniques used in data stream management systems to speed up range queries and selectivity estimation (the proportion of tuples that satisfy a query), two illustrative examples where fast but approximate answers are more useful than slow and exact ones.

In the context of open-ended data streams, as we never observe all values of the random variable, it is not appropriate to use the traditional histograms to construct a graphical representation of continuous data, because they require the knowledge of all data. Thus, there is still missing algorithms

to address conveniently this issue. The Partition Incremental Discretization [12, 27] and the V-Optimal Histograms [14, 16, 18] are two examples. A key characteristic of a data stream is its dynamic nature. The process generating data is not strictly stationary and evolves over time. The target concept may gradually change over time. Moreover, when data is collected over time, at least for large periods of time, it is not acceptable to assume that the observations are generated at random according to a stationary probability distribution. Several methods in machine learning have been proposed to deal with concept drift [11, 17, 21, 23, 27, 29, 30]. Drifting concepts are often handled by time windows or weighted examples according to their age or utility. Another approach to detect drift concepts is monitoring distributions on two different time windows, which monitors the evolution of a statistical function between two distributions: from past data in a reference window and in a current window of the most recent data points [20, 28].

1.1 Previous work

In a previous work [28], we present a method to detect changes in data streams. In that work, we construct histograms using the two layer structure of the Partition Incremental Discretization (PiD) algorithm and address the detection problem by monitoring distributions using a fixed window model. In this work, we propose a new definition of the number of histogram's bins and the use of an adaptive-cumulative window model to detect changes. We also perform studies on the distance measures and advance a discrepancy measure based on the asymmetry of the Kullback-Leibler Divergence (KLD). We support this decision in previous results. The results of the [28] suggest that the KLD achieve faster detection rates than the other tested distances measures (a measure based on entropy and the cosine distance).

1.2 Motivation, challenges and paper outline

The motivation for studying time-changing high-speed data streams comes from the emergence of temporal applications such as communications networks, web searches, financial applications, and sensor data, which produces massive streams of data. Since it is impractical to store completely in memory all data, new algorithms are needed to process data online at the rate it is available. Another challenge is to create compact summaries of data streams. Histograms are in fact compact representations for continuous data. They can be used as a component in more sophisticated data mining algorithms, like decision trees [17].

As the distribution underlying the data elements may change over time, the development of methods to detect when and how the process generating the stream is evolving, is the main challenge of this study. The main contribution of this paper is a new method to detect changes when learning histograms using adaptive windows. We are able to detect a time window where change has occurred. Another contribution is an improved technique to initialize histograms satisfying user constraint on the admissible relative error.

The proposed method has potential use in industrial and medical domains, namely, in monitoring biomedical signals and production processes (respectively).

The paper is organized as follows. The next section presents an algorithm to continuously maintain histograms over a data stream. In Section 3 we extend the algorithm for change detection. Section 4 presents preliminary evaluation of the algorithm in benchmark datasets and real-world problems. Last section concludes the paper and presents some future research lines.

2. HISTOGRAMS

Histograms are one of the most used tools in exploratory data analysis. They present a graphical representation of data, providing useful information about the distribution of a random variable. A histogram is visualized as a bar graph that shows frequency data. The basic algorithm to construct a histogram consists of sorting the values of the random variable and places them into *bins*. Next we count the number of data samples in each bin. The height of the bar drawn on the top of each bin is proportional to the number of observed values in that bin.

A histogram is defined by a set of k non-overlapping intervals and each interval is defined by its boundaries and a frequency count. The most used histograms are either *equal width*, where the range of observed values is divided into k intervals of equal length $(\forall i, j : (b_i - b_{i-1}) = (b_j - b_{j-1}))$, or *equal frequency*, where the range of observed values is divided into k bins such that the counts in all bins are equal $(\forall i, j : (f_i = f_j))$.

When all the data is available, there are exact algorithms to construct histograms [26]. All these algorithms require a user defined parameter k, the number of bins. Suppose we know the range of the random variable (domain information) and the desired number of intervals k. The algorithm to construct equal width histograms traverses the data once; whereas in the case of equal frequency histograms a sort operation is required.

One of the main problems of using histograms is the definition of the number of intervals. A rule that has been used is the Sturges' rule: $k = 1 + log_2 n$, where k is the number of intervals and n is the number of observed data points. This rule has been criticized because it is implicitly using a binomial distribution to approximate an underlying normal distribution [1]. Sturges rule has probably survived because, for moderate values of n (less than 200) produces reasonable histograms. However, it does not work for large n. Scott gave a formula for the optimal histogram bin width which asymptotically minimizes the integrated mean square error. Since the underlying density is usually unknown, he suggest using the Gaussian density as a reference standard, which leads to the data-based choice for the bin width of $a \times s \times n^{-1/3}$, where $a = 3.49$ and s is the estimate of the standard deviation.

In exploratory data analysis, histograms are used iteratively. The user tries several histograms using different values of k (the number of intervals), and choose the one that better fits his purposes.

2.1 The Partition Incremental Discretization (PID)

The *Partition Incremental Discretization* algorithm (*PiD* for short) that was designed to provide a histogram represen-

[1] Alternative rules for constructing histograms include Scott's (1979) rule for the class width: $k = 3.5sn^{-1/3}$ and Freedman and Diaconis's (1981) rule for the class width: $k = 2(IQ)n^{-1/3}$ where s is the sample standard deviation and IQ is the sample interquartile range.

tation of high-speed data streams. It learns histograms using an architecture composed by two layers. The first simplifies and summarizes the data, the algorithm transverses the data once and incrementally maintains an equal-width discretization; the second layer constructs the final histogram using only the discretization of the first phase. The first layer is initialized without seeing any data. As described in [12], the input for the initialization phase is the number of intervals (that should be much larger than the desired final number of intervals) and the range of the variable.

Consider a sample $x_1, x_2, ...$ of an open-ended random variable with range R. In this context, and allowing to consider extreme values and outliers, the histogram is defined as a set of break points $b_1, ..., b_{k-1}$ and a set of frequency counts $f_1, ..., f_{k-1}, f_k$ that define k intervals in the range of the random variable:

$$]-\infty, b_1],]b_1, b_2], ...,]b_{k-2}, b_{k-1}],]b_{k-1}, \infty[.$$

In a histogram, all x_i in a bin is represented by the correspondent middle point, which means that this approximation error is bounded by half of the length (L) of the bin. As the first layer is composed by equal-width histogram, we obtain:

$$x_i - m_j \leq \frac{L}{2} = \frac{R}{2k},\ b_j \leq x_i < b_{j+1} \text{ and } \forall j = 1, .., k.$$

Considering the set of middle-break points $m_1, ..., m_k$ of the histogram, we define the mean square error in each bin as the sum of the square differences between each point in that bin and their correspondent middle-break points: $\sum_i (x_i - m_j)^2 \leq n_j R^2/4k^2,\ b_j \leq x_i < b_{j+1},\ \forall j = 1, .., k$ and n_j is the number of data points in each bin. The quadratic error (QE) is defined as the sum of this error along all bins:

$$QE(k) = \sum_j^k \sum_i (x_i - m_j)^2.$$

From the above equations it follows that the quadratic error is bounded, in the worst case, by: $nR^2/4k^2$, where n denotes the number of observed variables.

The definition of the number of intervals is one of the main problems of using histograms. The number of bins is directly related with the quadratic error. How different would the quadratic error be if we consider just one more bin? To study the evaluation of the quadratic error of a histogram with the number of bins, we compute the following ratio, which we refer to as the relative error: $\epsilon = \frac{QE(k)}{QE(k+1)}$.

In order to bound the decrease of the quadratic error, we define the number of bins of the first layer as dependent on the upper bound on the relative error (ϵ) and on the fail probability (δ):

$$N_1 = O(\frac{1}{\epsilon} ln \frac{1}{\delta}). \quad (1)$$

Establishing a bound for relative error, this definition of the number of bins ensure that the fail probability will converge to zero when N_1 increases. So, setting ϵ and δ and using this definition we control the decrease of the quadratic error. Figure 1 shows that the number of bins increases when the error decreases and the confidence increases. Figure 1 (top) represents the number of bins of $layer_1$ in function of ϵ and δ. The bottom figures give a projection of the number of bins according with the variables ϵ and δ (respectively).

So, differing from [12] the input for the initialization phase is a pair of parameters (that will be used to express accuracy guarantees) and the range of the variable:

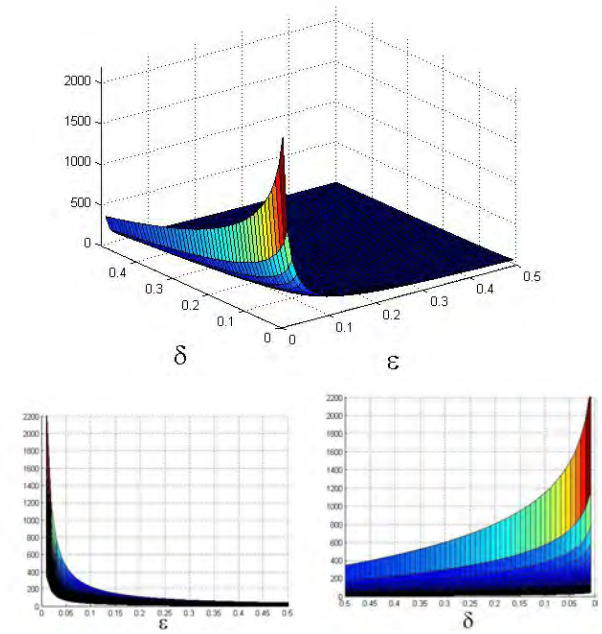


Figure 1: Representation of the number of bins of $layer_1$. The top figure shows the dependency from ϵ and δ and bottom figures show it according to only one variable.

- The upper bound on relative error ϵ.
- The desirable confidence level $1 - \delta$.
- The range of the variable.

The range of the variable is only indicative. It is used to initialize the set of breaks using an equal-width strategy. Each time we observe a value of the random variable, we update $layer_1$. The update process determines the interval corresponding to the observed value, and increments the counter of this interval. The process of updating $layer_1$ works online, performing a single scan over the data stream. It can process infinite sequences of data, processing each example in constant time and space. The second layer merges the set of intervals defined by the first layer. The input for the second layer is the breaks and counters of $layer_1$, the type of histogram (equal-width or equal-frequency) and the desirable final number of intervals. The algorithm for the $layer_2$ is very simple. For equal-width histograms, it first computes the breaks of the final histogram, from the actual range of the variable (estimated in $layer_1$). The algorithm traverses the vector of breaks once, adding the counters corresponding to two consecutive breaks. For equal-frequency histograms, we first compute the exact number F of points that should be in each final interval (from the total number of points and the number of desired intervals). The algorithm traverses the vector of counters of $layer_1$ adding the counts of consecutive intervals till F.

The two-layer architecture divides the histogram problem into two phases. In the first phase, the algorithm traverses the data stream and incrementally maintains an equal-width discretization. The second phase constructs the final histogram using only the discretization of the first phase. The

computational costs of this phase can be ignored: it traverses once the discretization obtained in the first phase. We can construct several histograms using different number of intervals and different strategies: equal-width or equal-frequency. This is the main advantage of PiD in exploratory data analysis. We use PiD algorithm to create compact summaries of data, and along with the improvement of the number of bins definition, we also accomplished it with a change detection technique.

3. CHANGE DETECTION

The algorithm described in the previous section assumes that the observations came from a stationary distribution. When data flows over time, and at least for large periods of time, it is not acceptable to assume that the observations are generated at random according to a stationary probability distribution. At least in complex systems and for large time periods, we should expect changes in the distribution of the data.

3.1 Related Work

When monitoring a stream is fundamental to know if the received data comes from the distribution observed so far. It is necessary to perform tests in order to determine if there is a change in the underlying distribution. The null hypothesis is that the previously seen values and the current observed values come from the same distribution. The alternative hypothesis is that they are generated from different continuous distributions.

There are several methods in machine learning to deal with changing concepts [21, 22, 23, 30]. In general, approaches to cope with concept drift can be classified into two categories: *i*) approaches that adapt a learner at regular intervals without considering whether changes have really occurred; *ii*) approaches that first detect concept changes, and next, the learner is adapted to these changes. Examples of the former approaches are *weighted examples* and *time windows* of fixed size. Weighted examples are based on the simple idea that the importance of an example should decrease with time (references about this approach can be found in [22, 24, 30]). When a time window is used, at each time step the learner is induced only from the examples that are included in the window. Here, the key difficulty is how to select the appropriate window's size: a small window can assure a fast adaptability in phases with concept changes but in more stable phases it can affect the learner performance, while a large window would produce good and stable learning results in stable phases but can not react quickly to concept changes.

In the latter approaches, with the aim of detecting concept changes, some indicators (e.g. performance measures, properties of the data, etc.) are monitored over time (see [21] for a good classification of these indicators). If during the monitoring process a concept drift is detected, some actions to adapt the learner to these changes can be taken. When a time window of adaptive size is used these actions usually lead to adjusting the window's size according to the extent of concept drift [21]. As a general rule, if a concept drift is detected the window's size decreases; otherwise the window's size increases.

3.1.1 Windows models

Most of the methods in this approach monitor the evolution of a distance function between two distributions: from past data in a *reference window* and in a *current window* of the most recent data points. An example of this approach, in the context of learning from Data Streams, has been present by [20]. The author proposes algorithms (statistical tests based on Chernoff bound) that examine samples drawn from two probability distributions and decide whether these distributions are different.

In this work, we monitor the distance between the distributions in two time windows: a reference window that has a fixed size and refers to past observations and an adaptive-cumulative window that receives the actual observations and could have a fixed or an adaptive step depending on the distance between distributions. For both windows, we compute the relative frequencies: a set of empirical probabilities $p(i)$ for the reference window and $q(i)$ for the adaptive-cumulative window.

3.1.2 Adaptive-cumulative window model

In a previous work [28] we defined the windows sizes as dependent on the number of intervals of the $layer_1$, being half of these ones: $\frac{N_1}{2}$. In this work, in order to evaluate the influence of the number of examples required to detect a change, we defined cumulative window (the current one) using an adaptive increasing step that dependents on the distance between data distributions. Starting with a size of $\frac{N_1}{2}$, the step is incremented if the distance between data distributions increases and is decremented otherwise, according to the following relation:

$$WindowStep = \frac{N_1}{2}(1 - \frac{1}{\alpha} * |KLD(p||q) - KLD(q||p)|), \quad (2)$$

where α is related with the change detection threshold (introduced further in this paper).

Figure 2 shows the dependency of the window's step on distributions' distance.

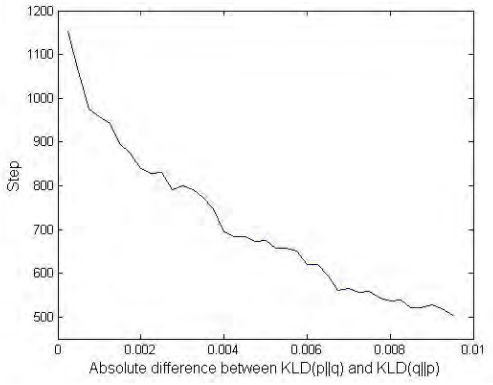


Figure 2: Representation of the windows' step with respect to the absolute difference between KLD(p||q) and KLD(q||p). An illustrative example showing that the window's step decreases when the absolute difference between distances increases.

[2]KLD stands for Kullback-Leibler Divergence. This measure is introduced in the next subsection.

3.2 Distance between distributions - Kullback-Leibler Divergence

Assuming that sample in the *reference window* has distribution p and that data in the *current window* has distribution q, we use as a measure to detect whether has occurred a change in the distribution the Kullback-Leibler Divergence.

From information theory [4], the Relative Entropy is one of the most general ways of representing the distance between two distributions [10]. Contrary to the Mutual Information this measure assesses the dissimilarity between two variables. Also known as the Kullback-Leibler (KL) distance or divergence, it measures the distance between two probability distributions and so it can be used to test for change.

Considering two discrete distributions with empirical probabilities $p(i)$ and $q(i)$, the relative entropy of p with respect to q is defined by:

$$KLD(p||q) = \sum_i p(i) log_2 p(i)/q(i).$$

The KL divergence is not a real metric since is asymmetric $KLD(p||q) \neq KLD(q||p)$. Nevertheless, it satisfies many important mathematical properties: is a nonnegative measure, it is a convex function of $p(i)$ and equals to zero only if $p(i) = q(i)$.

Given a reference window with empirical probabilities $p(i)$, and a sliding window with probabilities $q(i)$: lower values of $KLD(p||q)$, corresponds to smaller dispersion between the distributions of the two variables, meaning that them are closer. A higher value of the distance represents distributions that are further apart. Due to the asymmetric property, if the distributions are similar, the difference between $KLD(p||q)$ and $KLD(q||p)$ is small.

3.3 Decision Rule

According to the distance above, we define that had occurred a change in the distribution of the current window relatively to the reference distribution using a high quantile, the 99th percentile (or the 95th percentile, depending on the data length), as a boundary. If the absolute difference of the Kullback-Leibler divergence (KLD) between the distributions of the reference and the current windows and the KLD between the distributions of the current and the reference windows is greater than 1% (or 5%) we assign a change. If no change occurs, we maintain the reference distribution and consider more data points in the current window, and start a new comparison. If we detect any anomalies and/or deviations from what is expected, we can trigger an alert alarm and we clean the reference data set and initialize the process of search for changes. This decision rule is directly related with the window's step presented previously. From Eq. 2it is clear that the closer the distributions are to the considered significance level, the smaller the step will be.

4. EXPERIMENTAL EVALUATION

In this section we evaluate our proposed technique against statistics for nonparametric change detection. In order to compare results we used artificial data, where we can control the changes in generating distributions. We also present detection results obtained with our method in real world datasets to reveal its applicability and usage.

For both kinds of datasets, the data is received at any time producing an equal-with histogram. The number of bins is defined according to Eq. 1, setting both the variables ϵ and δ as 1% (or 5%, depending on the data length). We considered that the initial data points should be used as a representation of data and that the number of initial points should be chosen according to the number of intervals of the $layer_1$. So we decided that the first $10 * N_1$ data points are part of a stabilization process and that no change occurs in this range. For the decision rule, we established that if the absolute difference of the KLD between the distributions of the reference and the current windows and the KLD between the distributions of the current and the reference windows is greater than 1% (or 5%) we assign a change.

We estimate the delay time using the number of examples between the real change point and the detected point.

4.1 Controlled Experiments with Artificial Data

We compare the presented method, using a Fixed-Cumulative Window Model (FCWM) and an Adaptive-Cumulative Window Model (ACWM), with statistics for nonparametric change detection, the two-sample Kolmogorov-Smirnov Test (KST) and the Wilcoxon Rank Sum Test (WRST).

We perform tests using data underlying Log Normal distributions with different parameters, we simulated changes in mean and in standard deviation (StD). Distribution changes are created as follows: we generated 10 streams with 60K points each, the first and second 30K points of each stream are generated from $P_0 = LogN(0, 1)$ and $P1$, respectively. For changes in the mean parameter $P1 = LogN(\Delta p, 1)$ and for changes in the standard deviation parameter $P1 = LogN(0, 1 + \Delta p)$, with $\Delta p = 0.1, .., 1$. The goals of these experiments are:

1. Ability to Detect and React to drift.
2. Resilience to False Alarms when there is no drift, which is not detect drift when there is no change in the target concept.
3. The number of examples required to detect a change after the occurrence of a change.

Figure 3 shows the delay time of the change detection tests using the described artificial data, as a function of Δp. For datasets with changes in the mean parameter it can be observed that the ACWM achieve, for all values of Δp, better results than the FCWM. We can also conclude that, except for small values of Δp, the nonparametric tests outperforms the ACWM. However, the KST and WRST, have miss detections for some datasets.

The WRST tests if two datasets are independent samples from identical continuous distributions with equal medians against the alternative hypothesis that they do not have equal medians. The median of median of a LogNormal distribution is defined as e^{μ}. In the datasets with change in the standard deviation parameter the mean parameter remains the same ($\mu = 0$), meaning that the median is the same in all stream, so it does not make sense to use the WRST in this kind of datasets. For the other three tests we can observed that ACFW is the method that presents better results, achieving them with less delay time and without false alarms (the KST had 29 miss detections).

For both kinds of changes in distribution parameters, as Δp increases, one can observes that the delay time decreases, which is consistent with the design of experiments because as greater the Δp the more abrupt is the created change between distributions.

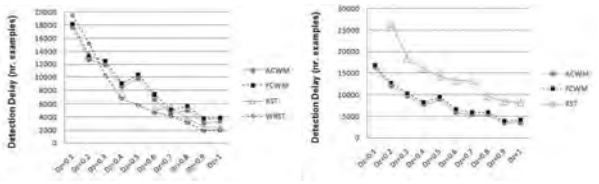


Figure 3: Detection delay (nr. of examples) for datasets with changes in mean and in StD parameters as a function of Δp.

To evaluate the performance of the two algorithms we also use quality metrics such as *Precision* and *Recall*. The *Precision* gives a ratio between the correct detected changes and all the detected changes and *Recall* is defined as a ratio between the correct detected changes and all the occurred changes:

$$Precision = \frac{TP}{TP+FP} \quad Recall = \frac{TP}{TP+FN}$$

For both quality metrics, the closest to one, the better are the results. For each detection test, we considered the sum of false alarms (TP and TN) and the sum of correct detections (TP). For the two kinds of changes, the precision achieved by the change detection tests were equal to one. Table 1 shows the recall achieved by the four detection tests, for all the datasets with changes in the mean and standard deviation parameters, respectively. For each detection test, we considered the sum of false alarms (TP and TN) and the sum of correct detections (TP).

Table 1: Recall, for changes in mean and standard deviation parameters, of the four change detection tests.

Method	Change in mean	Change in StD
ACWM	1.00	1.00
FCWM	1.00	1.00
KST	0.86	0.71
WRST	0.90	-

In spite of the recall values, that suggest the use of a window model to detect changes, we should also point out that both nonparametric tests compare distribution of all observed values in two different datasets. In the context of high-speed streams, data manipulations tend to become more laborious. Also for nonparametric tests, the critical values must be calculated for each distribution and these values may not always be generated by computer software. These are two reasons why nonparametric tests work only for low-dimensional data.

Comparing obtained results with ACWM and FCWM, the advantage of using a window's step depending on the distributions' distance can be easily observed. For all datasets, the number of examples required to detect a change decreased, allowing a faster answer and a quicker adaptation of the algorithm in drift context. The results obtained with those datasets were very consistent and precise, supporting the use of a window's step depending on the distributions' distance improves the accuracy of the change detection algorithm.

4.2 Data and experimental setup

In order to evaluate our algorithm in real-world problems, we considered a dataset from an industrial environment and data sets from two different medical domains.

4.2.1 Industrial dataset

To obtain data, tests were carried out in a Kondia HS1000 machining centre equipped with a Siemens 840D open architecture CNC. The blank material used for the tests was a 170 mm profile of Aluminum with different hardness. The maximum radial depth of cut was 4.5 mm using Sandvik end-mill tools with two flutes and diameter 8, 12 and 20 mm. Tests were done with different cutting parameters, using sensors for registry vibration and cutting forces. A multi-component dynamometer with an upper plate was used to measure the in-process cutting forces and piezoelectric accelerometers in the X and Y axis for vibrations measure. A Karl Zeiss model Surfcom 130 digital surface roughness instrument was used to measure surface roughness.

Each record includes information on the following seven main variables used in a cutting process:

- ***Fz*** - feed per tooth
- ***Diam*** - tool diameter
- ***ae*** - radial depth of cut
- ***HB*** - hardness on the type of material
- ***Geom*** - tools geometry
- ***rpm*** - spindle speed
- ***Ra*** - average surface roughness

This factors was proceeding the Design of Experiment explained in [9] was used for validation a Bayesian model for prediction of surface roughness. We use the sensor measure of the cutting speed on X axes to detect when a change had occurred in the experiments. We must point out that the measures for each test were saved individually. Then we jointed 9 of them sequentially in order to have only one dataset with eight changes. The goal is to study the effect of an adaptive window's step in change detection in an industrial problem.

Table 2 shows the obtained results and the time delay (we presented the average results for the required number of examples to detect the 8 changes). In spite of the window's step, the algorithm detects the 8 changes, but with an adaptive window's step (ACWM) the number of examples required to detect a change decreased.

Table 2: Results, for real data, using the FCWM and ACWM.

Industrial Dataset	TP	FP	DelayTime(average)
FCWM	8	0	1760
ACWM	8	0	1365

4.2.2 Medical dataset - CTGs

We have evaluated our detection algorithm on five Fetal Cardiotocographic (CTG) problems, collected at Hospital de São João, Porto. Fetal Cardiotocography is one of the most important means assessment fetal well-being. CTG signals

contain information about the fetal heart rate (FHR) and uterine contractions (UC).

Five antepartum FHR with a median duration of 70.8 min (SD: 18.3) were obtained and analyzed by the SisPorto® system. These cases corresponded to a mean gestational age of 39.7 weeks (SD: 2.6).

The SisPorto® system, developed at INEB (Instituto Nacional de Engenharia Biomédica), starts the computer processing of CTG features automatically after 11 min of tracing acquisition and its updated every minute [1], providing estimation of FHR baseline, identifying accelerations and decelerations and quantifying short- and long-term variability according to algorithms described in [2]. Along with this features the system also triggers alerts, such as 'Normality criteria met' alert, 'Non-reassuring alerts' and 'Very non-reassuring alerts' (further details can be founded in [2]). However, the system usually takes about 10 min to detect these different behaviors. In the 'Normal' stage of FHR tracing can be classified according to four different patterns: *A*) corresponding to calm or non-eye movement (REM) sleep, *B*) active or rapid eye movement (REM) sleep, *C*) calm wakefulness and *D*) active wakefulness [13].

Figure 4 shows an example of the analysis of a CTG exam exactly as it is produced by the SisPorto® system. The top tracing is the FHR and the bottom the UC. The FHR baseline estimation, accelerations and decelerations and different alerts stages also can be observed in this figure. The 'Normal' stage is represented with a green bar, the 'Suspicious' is represented with yellow and orange bars and the 'Problematic' with a red bar.

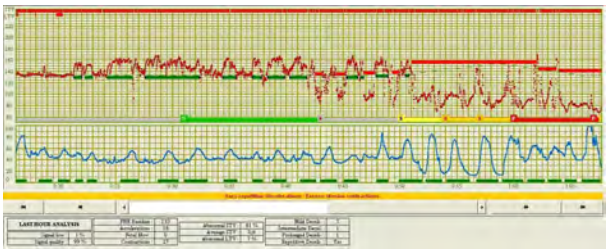


Figure 4: FHR (top) and UC (bottom) tracings. This window also includes the FHR baseline estimation, accelerations and decelerations and patterns classification.

Our aim is to detect the concept changes detected by SisPorto, if possible faster. We applied our detection algorithm to the FHR tracings. Because the records contained few observations, we set the input parameters ϵ and δ as 5% and for the decision rule we established the 95th percentile as boundary.

The achieved results are consistent with the system analysis and our algorithm detects the changes between the different stages earlier than the SisPorto® system. Further than the analysis of this program, our algorithm is able to detect some changes between different patterns of the 'Normal' stage. Due to difficulty of ascertain the exact change points between these behaviors we could not perform a delay evaluation. However the preference of an adaptive window's step is again supported by detections results in this dataset.

4.2.3 *Numerical high dimensional dataset: MAGIC Gamma Telescope Benchmark*

In order to simulate a data stream with concept changes, we modified the UCI MAGIC Gamma Telescope [6], which consists of 19020 data points in 2 classes with 10 numerical (real) attributes ('fLength', 'fWidth', 'fSize', 'fConc', 'fConc1', 'fAsym', 'fM3Long', 'fM3Trans', 'fAlpha' and 'fDist').

In order to rank the attributes' importance, we create classification trees (based on the algorithm described in [5]). We started by creating a classification tree using all the attributes. Then we take out the attribute that was chosen for the split test at the root and create another classification tree, and repeat the process until the rank is finished. We obtained the following ranking: 'fAlpha', 'fLength', 'fWidth', 'fM3Long', 'fAsym', 'fM3Trans', 'fConc', 'fSize', 'fConc1' and 'fDist'.

For each attribute, we created a single vector composed first for the examples of class 'gamma' (12332 data points) and followed by the examples of class 'hadron' (6688 data points). Figure 5 shows the modified data for each attribute in this dataset.

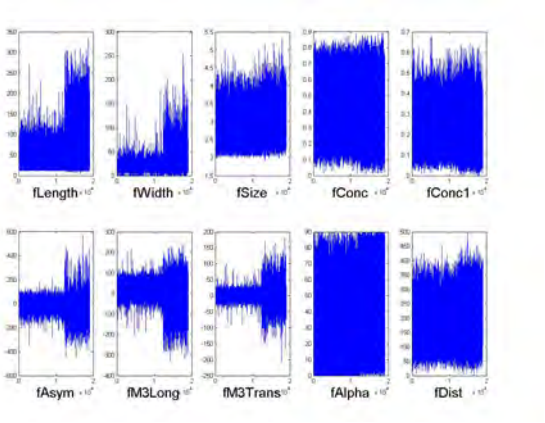


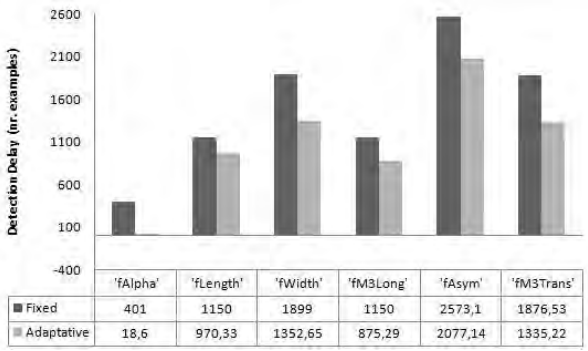
Figure 5: Data distribution of each attribute in the MAGIC Gamma Telescope dataset.

Because the data is not 'time labeled' and to obtain results independent from the examples order, for each attribute, we shuffled the examples of class 'hadron'. We repeated this strategy obtaining 100 sets for each attribute.

We evaluated both detection algorithms (with a fixed and adaptive cumulative window) in all datasets. Performing the change detection test (using ACWM and FCWM) we expected to detect changes in the top-ranked attributes, around the class change point (12332).

Both methods (ACWM and FCWM) detect the change point for attributes 'fLength', 'fWidth', 'fAsym', 'fM3Long' and 'fM3Trans', in all the 100 datasets of each one. For the rest of the attributes, none of the algorithms detected any change point. Figure 6 shows the achieved results for these attributes. In spite of the approximated delay time, the ACWM requires less data points to detect the change point for all the mentioned attributes. As expected, both algorithms require fewer examples to detect the change in the top-ranked attributes, which is consistent with the tree classification results, since the algorithm described chooses the best attribute to split the tree.

From data characteristics one may try to explore the de-



	'fAlpha'	'fLength'	'fWidth'	'fM3Long'	'fAsym'	'fM3Trans'
Fixed	401	1150	1899	1150	2573,1	1876,53
Adaptative	18,6	970,33	1352,65	875,29	2077,14	1335,22

Figure 6: Detection delay (nr. of examples) for different attributes using the ACWM and FCWM.

tected change points. Figure 7 shows the absolute difference between descriptive statistics of examples from class 'gamma' and class 'hadron' (these values are presented as percentage of the corresponding statistic of classe 'gamma'). The descriptive statistics shown are: mean, standard deviation and median. The 6 top-ranked attributes presented higher differences than the rest.

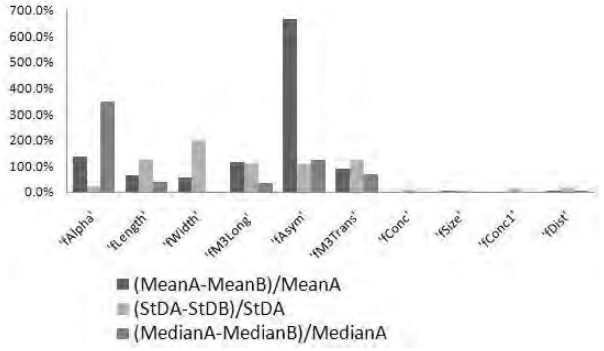


Figure 7: Absolute percentage difference between descriptive statistics of different classes.

5. CONCLUSION AND FUTURE RESEARCH

In this work we address the problem of detecting changes when constructing histograms from time-changing high-speed data streams. Histograms are a widely used tool in exploratory analysis from static data, providing useful graphical information about the distribution of a random variable. They also can be used as a component in more sophisticated data mining algorithms, like pre-processing (discretization), Bayesian classifiers and decision trees. However, in the context of open-ended data there are few contributions and still missing algorithms to address conveniently the data representation.

The method we present here is capable to understand how the process generating the stream is evolving; providing information of a time window where changes have occurred and adapting the histogram representation to the most current status of nature. For both artificial and real datasets, the results sustain that the algorithm with an adaptive window's step is capable to faster detection rates, using fewer examples to detect changes and reaching better performances. Finally, we must point out that our algorithm can be applied in a large variety of data stream problems, detecting and reacting to changes using fewer examples with the capacity of being resilient to false alarms when there are no drifts.

As a final conclusion, one can say that the results achieved so far are quite encouraging and motivating to continue this research line. Improvements of this algorithm and applications in more kinds of medical and industrial domains shall be considered. The use of different synopsis techniques and the adaptation of the proposed change detection algorithm to multivariate problems are future research steps.

6. ACKNOWLEDGMENTS

The work of Raquel Sebastião is supported by the Portuguese Foundation for Science and Technology (FCT) under the PhD Grant SFRH/BD/41569/2007.

The work of Pedro P. Rodrigues is supported by the Portuguese Foundation for Science and Technology (FCT) under the PhD Grant SFRH/BD/29219/2006.

The authors also thank to the financial support given by the FEDER, the Plurianual support attributed to LIAAD, project ALES II (POSC/EIA/55340/2004).

7. REFERENCES

[1] D. Ayres-de-Campos, P. Sousa, A. Costa and J. Bernardes. Omniview-SisPorto® 3.5 - a central fetal monitoring station with online alerts based on computerized cardiotocogram+ST event analysis. J. Perinat. Med. 2008; 36 - Article in press.

[2] D. Ayres-de-Campos, J. Bernardes, A. Garrido, J. Marques-de-Sá, L. Pereira-Leite. SisPorto 2.0: a program for automated analysis of cardiotocograms. J Matern Fetal Med. 2000; 9:311Ű8.

[3] D. Barbará. Requirements for clustering data streams. *SIGKDD Explorations (Special Issue on Online, Interactive and Anytime Data Mining)*, 3(2):23-27, 2002.

[4] M. Berthold and D. Hand. Intelligent Data Analysis - An Introduction. Springer Verlag, 1999.

[5] Breiman, L., et al.. Classification and Regression Trees. Chapman & Hall, Boca Raton, 1993.

[6] R. K. Bock and P. Savicky , 2007. MAGIC Gamma Telescope Benchmark. http://archive.ics.uci.edu/ml/datasets.html.

[7] M. Chabert, D. Ruiz, J-Y. Tourneret. Optimal wavelet for abrupt change detection in multiplicative noise. *IEEE International Conference on Acoustics Speech and Signal Processing*, pages: 1089-1092, May 2004.

[8] G. Cormode and M. Garofalakis. Sketching probabilistic data streams. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pages:281-292, June 2007.

[9] M. Correa, M. de J. Ramírez, C. Bielza, J. Pamies, and J.R. Alique. Prediction of surface quality using probabilistic models. *7th Congress of the Colombian Association of Automatic*, Cali, Colombia, 21–24 Mar, 2007. (in Spanish)

[10] T. Dasu, S. Krishnan, S. Venkatasubramanian and K. Yi. An Information-Theoretic Approach to Detecting Changes in Multi-Dimensional Data Streams. In Interface 2006 (Pasadena, CA) Report.

[11] J. Gama, P. Medas, G. Castillo and P. P. Rodrigues. Learning with Drift Detection. *Advances in Artificial Intelligence - SBIA 2004*, Vol.3171 of Lecture Notes in Computer Science, pages:286-295, São Luiz, Maranhão, Brazil, October 2004. Springer Verlag.

[12] J. Gama and C. Pinto. Discretization from Data Streams: applications to Histograms and Data Mining. In *Proceedings of the 2006 ACM Symposium on Applied Computing*, pages:662-667, 2006.

[13] H. Gonçalves, J. Bernardes J, A. Paula Rocha et al. Linear and nonlinear analysis of heart rate patterns associated with fetal behavioral states in the antepartum period. *Early Human Development*, 83(9):585–591, 2007.

[14] S. Guha, N. Koudas and J. AndWoo. REHIST: Relative error histogram construction algorithms. In *Proceedings of the VLDB Conference*, pages:300-311, 2004.

[15] S. Guha and B. Harb. Wavelet synopsis for data streams: minimizing non-euclidean error. In *Proceedings of the eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages:88-97, August 2005.

[16] S. Guha, N. Koudas and K. Shim. Approximation and streaming algorithms for histogram construction problems. *ACM Transactions on Database Systems (TODS)*, 31(1):396-438, 2006.

[17] G. Hulten, L. Spencer and P. Domingos. Mining Time-Changing Data Streams. ACM SIGKDD 2001. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages:97-106, 2001. ACM Press.

[18] H. V Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel. Optimal Histograms with Quality Guarantees. In *Proc. of the VLDB Conference*, pages: 275-286, 1998.

[19] P. Karras, D. Sacharidis, and N. Mamoulis Exploiting Duality in Summarization with Deterministic Guarantees. Approximating a Data Stream for Querying and Estimation: Algorithms and Performance Evaluation. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages:380-389, 2007.

[20] D. Kifer, S. Ben-David and J.Gehrke. Detecting change in data streams. In *VLDB 04: Proceedings of the 30th International Conference on Very Large Data Bases*, pages:180-191, 2004. Morgan Kaufmann Publishers Inc.

[21] R. Klinkenberg and I. Renz. Adaptive information filtering: Learning in the presence of concept drifts. In *Learning for Text Categorization*, pages:33-40, 1998. AAAI Press.

[22] R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. *Proceedings of ICML-00, 17th International Conference on Machine Learning*, pages 487–494, Stanford, US, 2000. Morgan Kaufmann Publishers.

[23] R. Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, 8(3):281-300, 2004.

[24] M. Maloof and R. Michalski. Selecting examples for partial memory learning. *Machine Learning*, 41:27–52, 2000.

[25] T. Mendonça, A. Marcal, A. Vieira, J. Nascimento, M. Silveira, J. Marques, J. Rozeira. Comparison of Segmentation Methods for Automatic Diagnosis of Dermoscopy Images. *Proceedings of IEEE Engineering in Medicine and Biology Society Annual Int. Conf.*, EMBS 2007, Lyon, France, 2007.

[26] D.D. Pestana and S.F. Velosa. *Introdução à Probabilidade e à Estatística*. Fundação Calouste Gulbenkian, 2002.

[27] C. Pinto and J. Gama. Incremental discretization, application to data with concept drift. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, pages:467-468, March 2007.

[28] R. Sebastião and J. Gama Change Detection in Learning Histograms from Data Streams. In *Proceedings of Portuguese Conference on Artificial Intelligence*, Guimarães, Portugal, December 2007.

[29] E. J. Spinosa, A. Carvalho and J. Gama. OLINDDA: A cluster-based approach for detecting novelty and concept drift in data streams. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, pages:448-452, March 2007.

[30] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23:69-101, 1996.

Unsupervised Plan Detection with Factor Graphs

George B. Davis
School of Computer Science
Carnegie Mellon University
gbd@cs.cmu.edu

Jamie Olson
School of Computer Science
Carnegie Mellon University
jolson@cs.cmu.edu

Kathleen M. Carley
School of Computer Science
Carnegie Mellon University
carley@cs.cmu.edu

ABSTRACT

Recognizing plans of moving agents is a natural goal for many sensor systems, with applications including robotic pathfinding, traffic control, and detection of anomalous behavior. This paper considers plan recognition complicated by the absence of contextual information such as labeled plans and relevant locations. Instead, we introduce 2 unsupervised methods to simultaneously estimate model parameters and hidden values within a Factor graph representing agent transitions over time. We evaluate our approach by applying it to goal prediction in a GPS dataset tracking 1074 ships over 5 days in the English channel.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

1. INTRODUCTION

Many real world sensor networks are capable of simultaneously tracking many agents as they navigate a space. Examples include network-linked GPS devices in phones and vehicles, passive tracking mechanisms such as radar (when paired with technology to distinguish agent identities), and entry-point systems such as keycard and RFID scanners. (Many additional examples emerge when we consider agents navigating virtual spaces such as the World Wide Web, but this paper will concentrate on focus on a physical system). As the volume and complexity of data produced by these systems has grown, human monitors are increasingly dependent on algorithms that can efficiently extract relevant patterns for their analysis.

One successful approach to pattern mining in this domain has been to presume the existence of hidden variables which mediate the transitions observed by sensors. For example, there may be a hidden activity that explains the time an agent spends at a certain location, or an underlying plan that explains the choice to travel from one place to another. Probabilistic relationships between hidden variables and observed variables can be encoded with graphical models such as Conditional Random Fields (CRFs), which support efficient algorithms for inferring missing values. Previous work have used this general approach to predict future agent actions and detect surprising deviations from typical behavior [8]. Applications have been discussed in contexts ranging from robotic and human planning [2] to assistance of seniors and disabled individuals [7].

Inference from this type of model is generally preceded by a training phase, in which model parameters are optimized against a dataset for which "true" values of hidden variables have been provided. In previous experiments, training data were drawn from journals of experiment participants, hand-coded by human oservers, or extracted from existing databases (*e.g.* maps of known locations or obstacles). In this paper, we examine a case where such data would be useful but is unavailable. Our dataset tracks the movement of 1700 merchant marine vessels servicing ports in the English channel over a 5 day period. Maritime navigation is influenced by "soft" conventions, such as shipping lanes and waypoints, rather than fixed constraints, such as roads or walls. Because some conventions differ between nationalities, companies, and ship types, there is no single collation that could be built directly into our model. The same diversity of backgrounds and conventions would make conducting a survey with reasonable accuracy and breadth cost-prohibitive.

We believe our domain is one of many for which assuming the existence of training data is unrealistic. Others include covert applications where subjects cannot be polled, large scale applications where surveys would be expensive or inaccurate, and evolving environments in which training data quickly becomes outdated. As a solution we propose and unsupervised approach to graphical models, allowing the user to exploit knowledge of the structure of hidden variables in a system without observing them - even during training. Instead, we introduce 2 algorithms that simultaneously assigns variable values and optimizes model parameters in a way that is self-consistent given the model structure. Our model and algorithm are simpler than the most advanced supervised methods, but they nonetheless give compelling results on a goal prediction task, and serve to demonstrate a variety of challenges involved in creating an unsupervised approach.

The rest of this paper is organized as follows. In section 2, we review undirected graphical models of distributions, includ-

SENSOR-KDD 2008, Las Vegas, Nevada, USA.

ing the relationship between our chosen representation, the factor graph (FG), and the more commonly discussed conditional random field (CRF). We also review prior applications of CRFs to plan detection and introduce our notation and dataset. In section 3 we discuss new challenges involved in unsupervised inference and introduce our own factor graph and algorithms. In section 4, we empirically evaluate our approach with results regarding the intermediate variables, the goal prediction task, and convergence rates of the algorithm. In section 5 we summarize our contributions and discuss future work in this area.

2. BACKGROUND

2.1 Factor Graphs

What follows is a terse introduction to the rich topic of undirected probabilistic graphical models (PGMs). As our primary goal is to introduce our notation, we refer the reader to [6] for a more thorough discussion. We restrict discussion of supervised learning and inference to a brief comparison in section 3, when we describe our unsupervised approach. For deeper background on that topic, we defer to [12].

Consider a set X of random variables, each of which can take one of at most n discrete states. A joint assignment to X is notated with the integer vector $\vec{x} \in \{\mathbb{Z}^+ \leq n\}^{|X|}$. We can parameterize the joint PMF $P(X = \vec{x})$ with a vector $\vec{w} \in \mathbb{R}^d$ whose entries specify the probability of each combination of value assignments to X. This naive representation would require $d = n^{|X|} - 1$ parameters, a quantity too great to store, fit to a dataset, or conduct inference from unless X is trivially small. The goal of a factor graph (FG) is to allow the required number of parameters to scale better with $\mid X \mid$ by identifying a set of functions with smaller domains which factorize the PMF.

Formally, a factor graph G on X can be represented by the tuple $G = \langle F, D \rangle$ where each $f \in F$ is a potential function $f : \mathbb{Z}^{|D(f)|} \rightarrow \mathbb{R}$ giving the local likelihood of a vector of values for the variables retrieved by the set-valued domain function $D : f \in F \rightarrow S \subset X$. Notating as $\vec{x^f}$ the integer vector extracting values for variable in $D(f)$ from $\vec{x}$, we can rewrite the PMF via its factorization,

$$P\left(X = \vec{x}\right) = \frac{1}{z} \prod_{f \in F} f\left(\vec{x^f}\right) \tag{1}$$

where z is a normalizing constant. The "graph" in factor graph is the bipartite network between factors and the variables in their domains. There is a corresponding single mode network between variables, the Markov network for G, whose neighborhood function (which also gives the *Markov blanket*) is

$$N(x) = \left(\bigcup_{f \in D^{-1}(x)} D(f) \right) \setminus x \tag{2}$$

The precise relationship between the factor graph and this Markov network is that they induce the same set of conditional independences,

$$\forall_{x,x' \in X},\ x \perp x' \mid N(x) \tag{3}$$

In fact, any Markov network can be written as a factor graph in which there is one factor per variable, with domain equal to that variable plus its neighborhood. However, by breaking a neighborhood into multiple factors, a factor graph can describe additional structure in potential functions, yielding a tighter bound on the number of parameters necessary. In the worst case, representing a factor f requires a parameter vector $\vec{w^f}$ with dimension $n^{|D(f)|}$ to enumerate potentials for all combinations of variable values in its domain. Definition (2) ensures that there exists a value k satisfying

$$\max_{f \in F} |D(f)| \ = \ k \ \leq \ \max_{x \in X} |N(x)| \ , \tag{4}$$

which allows us to bound the dimensionality of the full parameter vector to be exponential only in k:

$$d = \sum_{f \in F} n^{|D(f)|} \ \leq \ \mid F \mid n^k \tag{5}$$

Some distributions can be further abbreviated using *parameter sharing* between factors. For example, most hidden Markov models (HMMs) apply the same set of transition probabilities to any pair of sequential states. To annotate this we replace the domain function $D(f)$ with an instantiation function $I(f)$ which returns a set of variable sets, the *factor instance domains* on which the factor is instantiated. Repeated instantiation of factors does not affect the number of parameters necessary, but the PMF becomes

$$P\left(X\right) = \frac{1}{z} \prod_{f \in F} \prod_{I \in I(f)} f\left(\vec{d^I}\right) \tag{6}$$

In some applications, there is a clear division between the set X of *hidden* variables, whose values must be inferred, and a second set Y of *observed* variables. Better performance can often be achieved for these cases using *discriminative* models, which represent only the conditional distribution $P(X \mid Y)$ and not relationships between observable variables. Incorporating observed variables into our notation requires no immediate adjustments, but each factor instance domain $D \in I(f)$ may now include observed variables (but must still contain at least one hidden variable). Furthermore, observed variables may be continuous so long as each factor involving them has a potential function with finite parameterization (preferably, one compact enough to maintain the bound in (5)).

The discriminative version of the Markov network given by (2) is known as a Conditional Random Field (CRF), and is the most common model for previous work in plan detection. We use factor graphs in this paper because they

generalize many other models and allow simpler notation for our methods.

2.2 Sensor Logs and PGMs

This paper builds on a growing body of work applying PGMs to *sensor logs*. In our context, a sensor log is an observation set O, in which each member $\vec{o_t^a} \in O$ is a vector tagged with timestamp t and subject agent a. The vector itself includes all state information observed by the sensors, which is generally spatial context such as position and speed.

Records of this form obviously record only data visible to sensors, and in so doing break into discrete observations what agents experience continuously. Graphical models of sensor logs attempt to restore hidden state and continuity by learning relationships between co-temporal and sequential variables. The prototypical PGM of this form is the Hidden Markov model (HMM), which can be encoded as two-factor FG. The first factor, $f_S(s_t^a, \vec{o_t^a})$ measures the likelihood of $\vec{o_t^a}$ being observed if s_t^a is the underlying state. The second factor, $f_T(s_t^a, s_{t+1}^a)$, measures the likelihood of transitioning between two discrete hidden states. Work in the past decade has built from this skeleton into more advanced models that discriminate many types of hidden state and exploit a variety of local information.

Ashbrook and Starner [2] fit HMMs to GPS sensor logs of users in their daily travel routines, and confirmed non-random patterns in the transition probabilities. Nguyen *et al.* [9][10], and later [4], developed successively more complex models using additional hidden layers to represent higher level activities (those lasting multiple observations) in a controlled kitchen environment. Our own work follows most closely on that of Liao *et al.*, who have used CRFs [7] and hybrid graphical models [8] to estimate activities, place types, transportation modes, goal coordinates, and occurrence of behavioral deviations ("novelty") in agents navigating an urban environment.

Since neither our work nor prior models involve relationships or interactions between agents, the corresponding factor graphs can be broken down into separate components for each agent. Previous approaches have generally chosen to train separate models for each agent, save for parameters learned off-line such as sensor error models. [7] experimented with making predictions for one agent based on a model trained on another agent, with some success. Because our dataset has only a 5-day duration, it was necessary for us to smooth results over all agents.

Notably, there are several examples of previous work in which the space of hidden states is drawn from data rather than directly provided. In [2], GPS observations were initially clustered to derive a finite state space (though they were spot-validated by test subjects for accuracy). In [7], training data provided examples of some locations labeled with types, but additional locations were identified by adding them when inference suggested their existence (according to a hard-coded criterion). The distinction between these methods and the unsupervised learning we propose in this paper is that they treat state identification as a separate clustering step specific to their model, distinct from the general purpose algorithms used during supervised training. Although there are many other instances in literature of unsupervised learning algorithms for specific models, this paper is the first work of which we are aware discussing unsupervised algorithms applicable to factor graphs in general.

2.3 AIS Data

For 5 days in June 2005, a sensor network queried Automated Identification System (AIS) transponders on merchant marine vessels navigating the English Channel. The Automated Identification System (AIS) is a communication standard for ocean vessels used by ships and ground stations to coordinate shipping traffic. AIS transponders on compliant vessels are integrated with the ship's radio, GPS, and navigational control systems. When pinged (via broadcast messages from other boats or ground stations), the transponder replies with a radio packet containing ship identity, current GPS coordinates, heading, speed, and various other fields describing navigational state, destination, and more. AIS compliance is required on ships over a certain size by most commercial ports, making it essential for most sizable merchant vessels operating worldwide.

In total, the sensor sweep captured movements of over 1700 vessels were recorded, with activities ranging from simple shipping lane traversals to apparently complex itineraries with stops at multiple ports of call. The reasons for the collection of the data are primarily security related. The global shipping system plays a prominent role in a variety of terrorist attack scenarios, both in the United States and abroad: in any country, the ports are both the most likely means of entry for bombs and other weapons, and themselves a prime economic and symbolic target. In addition to being an attractive target, ports are currently considered unsecure – for example, it has been suggested that only 3% of shipping containers entering the United States are directly inspected by customs officials. The sheer volume of commerce conducted via international shipping makes nave attempts at greater security infeasible, as neither the direct costs associated with detailed surveillance nor the indirect costs incurred by reducing industry efficiency are easily absorbed. If automated techniques such as those designed above can give insight into the behavioral patterns and structural features of the merchant marine population, then limited budgets for surveillance and interdictions can be more precisely targeted to have the greatest impact on overall security. The data under analysis here is especially promising as it represents the result of a relative inexpensive, passive, and consensual surveillance effort.

In many sensor datasets, physical limitations of the sensors are a primary source of error; for example, an error of 10m in a car-installed GPS system can introduce ambiguity as to which street the car is on. In the case of AIS data, the physical error of sensors is so small compared to the scale of navigation (some tankers are themselves 400m long) that a sensor error model is less relevant. Instead, a primary source of error comes from creative utilization of user-input fields such as destination and navigational status. We chose to focus only on numeric fields that would be drawn directly from navigational computers. Even on this set, there were many cases of misconfigurations which, for example, reported 0 latitude and 0 longitude for the study duration. We preprocessed to eliminate all ships with constant values for any

numeric field.

AIS responses in the original dataset were intermittent with inconsistent inter-arrival times. Although work exists regarding the use of temporally irregular observations(e.g. [4]), we avoid these issues. Instead, we filter the data to produce streams of observations in which at least 45 minutes and at most 180 minutes passes between observations. We also remove ships that make fewer than 5 consecutive movements, yielding a dataset of 10935 tracks of 576 ships. We also remove 140 erroneous responses sent by malfunctioning or otherwise corrupted responders. Figure 1 shows the final dataset visualized in Google Earth [1].

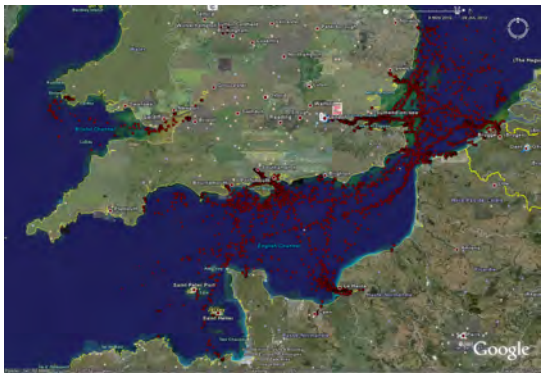


Figure 1: AIS sensor data from the English channel

3. METHODS

3.1 Factor Graphs for AIS

The trend in previous work has been to provide increasingly complex graphical models to incorporate additional sensor data (e.g. use of street maps in [8]) or knowledge regarding relationship structure (e.g. modeling of activity duration by [4]). In order to concentrate on unsupervised learning, we employed the relatively simple, two-layer model shown in figure 2. The variables in our factor graph include:

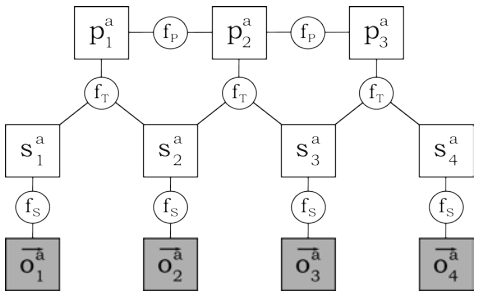


Figure 2: Plan Prediction Factor Graph

- $\vec{o_t^a}$ is an *observed* vector in $\mathbb{R}^3$ containing the latitude, longitude, and speed of agent a at time t,
- s_t^a is a discrete *state* variable representing the instantaneous state of an agent. It takes on integer values $0 \leq s_t^a < n_s$.
- p_t^a is a discrete *plan* variable capturing an internal agent state persisting over several time periods. It takes on integer values $0 \leq p_t^a < n_p$

The following factors model relationships in our graphs.

- $f_S(s_t^a, \vec{o_t^a})$ is a *state compatibility* factor which measures the likelihood of observation $\vec{o_t^a}$ being generated when within state s_t^a. f_S is implemented by maintaining n_s Gaussians, so that $f_S(c, \vec{o_t^a})$ is equal to the probability density at $\vec{o_t^a}$ of distribution $\mathcal{N}(\mu_c, \Sigma_c)$ where the mean vectors and covariance matrices for each Gaussian comprise the factor parameters. To avoid overfitting the Gaussians corresponding to infrequently observed states, each one is initialized with a a mean prior drawn from a uniform distribution over the range of latitudes, longitudes and speeds. The prior covariance is the covariance matrix for the same uniform distribution.
- $f_T(s_t^a, s_{t+1}^a, p_t^a)$ is a *state transition* factor which measures the likelihood of transitioning from s_t^a to s_{t+1}^a. This likelihood is mediated by the plan state p_t^a, representing (for example) the propensity of an agent to select a different route when targeting a different destination. This factor is parameterized as a likelihood table for all possible transitions, and is initialized with a uniform prior to ensure that a minimal probability remains for unobserved transitions.
- $f_P(p_t^a, p_{t+1}^a)$ is a *plan transition* factor which measures the likelihood of switching from p_t^a to p_{t+1}^a. Whereas the state transition factors capture physical constraints (the need to move between continuous states), the primary purpose of the plan transition factor is to model a time scale on how frequently agents are expected to change objectives. This factor has a single parameter, the change probability, which we initialize to .2 to indicate an expected time scale of plans being maintained for approximately 5 hours (the average time-at-sea we observed in ships that went to port). Although this parameter (and therefore the time-scale of a plan) can be changed during training, this initial setting plays an important role in determining which maximal labeling we will reach. This is discussed further in section 3.2.

3.2 Unsupervised Learning for Factor Graphs

During *supervised* learning, factor parameters are generally found maximizing the expectation of some training set $T = \vec{t}$.

$$\vec{w}^*(\vec{t}) = \underset{\vec{w}}{\operatorname{argmax}} P_{\vec{w}}(T = \vec{t}) \tag{7}$$

Maximum likelihood estimation (MLE) can then be performed by finding the assignment to hidden variables X that has maximum likelihood under factor parameters $\vec{w}^*$. [1]

[1]In most applications the training sets X and T may be different sizes or even "shapes" in terms of relations between variables. However, if a generator is provided for instantiating the same factors on both sets, parameter sharing allows us to reuse a single parameter vector.

$$\vec{x}^*(\vec{w}) = \operatorname*{argmax}_{\vec{x}} P_{\vec{w}^*(\vec{t})}(X = \vec{x}) \tag{8}$$

In the unsupervised case, no true values $\vec{t}$ are provided, preventing sequential learning and inference. As an alternative goal, we seek to find an assignment satisfying the fixed point of (7) and (8):

$$\vec{x}^* = \operatorname*{argmax}_{\vec{x}} P_{\vec{w}^*(\vec{x}^*)}(X = \vec{x}) \tag{9}$$

To compare the many possible solutions to (9), we introduce the *self-consistency likelihood*, a scoring function favoring assignments which receive high probability under their own optimal parameter values:

$$\bar{L}(\vec{x}) = P_{\vec{w}^*(\vec{x})}(X = \vec{x}) \tag{10}$$

The global maximum of $\bar{L}$ is the fixed point with maximum self-consistency. However, finding it is challenging on several levels. First, the space of possible assignment vectors is far too large (size $n^{|X|}$) to enumerate or sample meaningfully. Second, evaluating $\bar{L}(\vec{x})$ is expensive: one must first compute the parameter values $\vec{w}^*(\vec{x})$, and then the partition constant z for the corresponding distribution.

Algorithms for supervised inference on PGMs face the same challenges above, and most overcome them using local search informed by the graphical structure. For example, the max-residual belief propagation (MRBP) algorithm maintains a set of messages corresponding to graphical ties, and incrementally update message values in a way that it is guaranteed to reduce a free energy quantity. Unfortunately, these methods cannot be directly applied to maximize our target, $\bar{L}$. Whereas changing the value of a variable x in a graph with fixed factor parameters affects only local likelihoods, it can potentially effect *all* factor instances used to calculate $\bar{L}$. This is because the change may affect the optimal parameter settings for all factors for which x participates in an instance. An alternate way to describe this effect is that the distribution $\bar{P}$ achieved by normalizing $\bar{L}$ no longer induces the independences given in (3) – the Markov blanket for x under $\bar{P}$ includes all variables with which it shares a *factor*, not an instance.

However, we can offer a preliminary argument regarding a bound on the impact of these "long range effects". Let $w^{f*}(\vec{x})$ be the optimal parameter assignments for a single factor under assignments $\vec{x}$, and let $\vec{x} \leftarrow (x, c)$ be an operator returning an updated assignment vector with variable x set to state c. Now consider the condition

$$\forall_{x,c} \lim_{|I(f)|\to\infty} w^{f*}(\vec{x}) - w^{f*}(\vec{x} \leftarrow (x, c)) = 0 \tag{11}$$

In other words, as the number of instances of a factor grows, the incremental change to optimal parameters caused by changing the value of a single variable approaches zero. Many common factor parameterizations satisfy this condition, including those we use and list in section 3.1 (modulo the assumption that we observe all Gaussians and state transitions a sufficient number of items). Under this condition, the effect under $\bar{P}$ that changing x has on local likelihoods outside $N(x)$ becomes negligible as our graph becomes larger.

Armed with this intuition, we define a local search with an operator $\delta : \mathbb{Z}^{|X|} \to \mathbb{Z}^{|X|}$, which produces a sequence of assignment vectors following $\vec{x^i} = \delta(\vec{x^{i-1}})$. If δ is such that

$$P_{\vec{w}^*(\vec{x})}(X = \delta(\vec{x})) \geq P_{\vec{w}^*(\vec{x})}(X = \vec{x}) \tag{12}$$

then its fixed point must satisfy (9) as well (assuming that it does not trivially self-cycle). In the following subsections we introduce two operators that satisfy this condition, but have different properties in terms of convergence rate and susceptibility to local maxima while maximizing $\bar{L}$.

Asynchronous EM

One way graphical models support efficient computation is by defining marginal distributions $P(x \mid N(x))$ that can be efficiently computer. This allows quick updates to be designed for Gibbs samplers and belief propagation algorithms [12]. Our first local search method exploits this to improve an assignment vector incrementally by setting one variable at a time to the value with maximum expectation under the current state. The successor is

$$\delta_A(\vec{x}^t) = \vec{x}^t \leftarrow \left(x^t, \operatorname*{argmax}_c P_{\vec{w}*(\vec{x}^t)}\left(X = \vec{x}^t \leftarrow (x^t, c)\right)\right), \tag{13}$$

where x^t is drawn from a round robin schedule established in advance. This operator is easy to implement for our graph because our factors support incremental updates: changing the value of x changes only factor instances $I^{-1}(x)$, and each of our factors can be readjusted to give maximum expectation to a new instance assignment in constant time. When describing an iteration of the algorithm we include one update for each variable, in order to standardize the unit of work by graph size. Pseudocode for an implementation of δ_A can be found as Algorithm 1.

The initial assignments $\vec{x}^0$ are selected in the following way. First, a vector of random assignments $\vec{x}'$ is established. Then, each variable is set to its maximum likelihood value with neighbor variables assigned according to $\vec{x}'$ using the prior factor parameters. This "stacking" of the initial state assures that initial factor parameters fully explore the range of possible values they can take on. In testing, we found that making sure that initial parameters were distributed was essential to avoiding bad local maxima. For example, maximizing initial factor parameters against a random allocation vector tended to intialize all Gaussians in state factors to have means near the actual mean coordinates for the data. This initial clustering resulted in poor exploration of the state space, with most clusters remaining near the map center even after many iterations.

Algorithm 1 ASYNCHRONOUS

$\vec{w^0} \leftarrow$ Draws from prior
$\vec{x^0} \leftarrow$ Random allocation
$t \leftarrow 1$
loop
  $\vec{w}^t \leftarrow \vec{w}^{t-1}$
  $\vec{x}^t \leftarrow \vec{x}^{t-1}$
  for all $x \in X$ **do**
    $\vec{x}^t \leftarrow (\vec{x}^t \leftarrow x, \text{argmax}_c\, P_{\vec{w}^t}\left(X = \vec{x}^t \leftarrow (x^t, c)\right))$
    $\vec{w}^t \leftarrow \text{argmax}_{\vec{w}}\, P_{\vec{w}}(X = \vec{x}^t)$ (local updates)
  end for
  $t \leftarrow t + 1$
end loop

Algorithm 2 SYNCHRONOUS

$\vec{w^0} \leftarrow$ Draws from prior
$t \leftarrow 0$
loop
  $\vec{x}^t \leftarrow \text{MLE}_{\vec{w}}(\vec{x})$ (calls MLBP)
  $\vec{w}^{t+1} \leftarrow \text{argmax}_{\vec{w}}\, P_{\vec{w}}(X = \vec{x}^t)$
  $t \leftarrow t + 1$
end loop

Synchronous EM

Our second operator synchronously updates all variable values to a maximum likelihood estimate under the current factor parameters:

$$\delta_S(\vec{x}^t) = \text{MLE}_{\vec{w}^*(\vec{x}^t)}(X) \tag{14}$$

This is analogous to a standard EM algorithm, in which cluster assignments and cluster parameters are updated in an alternating fashion. We hypothesized that taking larger steps in the space of assignment vectors might make us less susceptible to local minima. However, by changing assignments to many variables at once, we may be less protected by the guarantee in (11).

Pseudocode for this method is listed as Algorithm 2. We initialize cluster parameters with priors as we did for the asynchronous method, but it is unnecessary to initiate the first state as we will be using maximum likelihood belief propagation, which depends only on observed variables and factor parameters. Then, at each step, we conduct inference with the current factor parameters using MLBP. Finally, we re-optimize factor parameters to the new assignment.

3.3 Plan Projection Experiment

We designed an experiment to simulate our system's performance at a fundamental task: using the estimated plan of an agent at sea to predict where it will next make port. Our experiment proceeds in two phases. First, we perform unsupervised learning on a test set representing sequences that would have occurred prior to some test sequences, as well as on the first potion of the test sequences themselves. Then, using the labels and factor parameters assigned during the first phase, we sample a distribution of future states for the test set, in order to estimate its next stop.

To create a dataset of sequences appropriate to this task, we developed the following test. First, we included only observations from ships with five consecutive observations "in motion" (reported velocity over 1 km / h) to eliminate a large percentage of ships that did not move often enough to assist training of transition probabilities. Since our model does not explicitly address the duration between observations, we standardized this quantity by eliminating sequences whose inter-observational interval was outlying (over 3 hours). A total of 13715 individual observations fell into sequences in this category. Then, for the test set, we isolated the 457 subsequences within the test set that consisted of routes beginning in motion and ending stopped, with at least 5 segments in between. The criterion on test sequence length is the only one of these filter that could not be applied without full knowledge of the test set, but was necessary to ensure that each test sequence A) was long enough for us to instantiate a factor graph with all factors on, and B) had a buffer beyond this so that we would be forced to predict multiple state transitions.

To calculate a maximum likelihood estimate for the next portfall of a particular test ship, we appended 100 additional hidden plans and states (along with associated factors) to the section of the ship's factor graph which was optimized during training. We then ran Gibbs a sampler on these hidden states using the factor parameters learned during training. Every 1000 iterations we would extract a prediction from the sampler by recording the mean position of the first state whose expected velocity was under 1 km / h.

4. RESULTS AND ANALYSIS

Visual inspection of the locations and transition probabilities learned by our algorithm confirms that it produces a coarse but credible summary of traffic flow in the channel. Figure 3 shows one model trained with our asynchronous algorithm and visualized using Google Earth. Vertexes are placed at the Gaussian mean for each state, with edges placed between transitions with high probability under any plan.

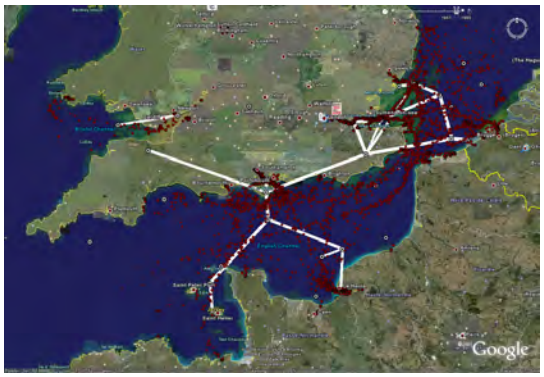


Figure 3: Learned plans and hidden states overlaid on AIS observations

To measure accuracy on our portfall prediction task, we computed the surface distance between the predicted destination and the actual portfall associated with each prediction and plotted the inverse cumulative density for this

figure as Figure 4. The curve summarizes a set of probably approximately correct (PAC) bounds for the estimator. For example, models trained with the asynchronous algorithm achieved accuracy under 100km 71% of the time. Synchronous models had only a 53% chance of acheiving this accuracy.

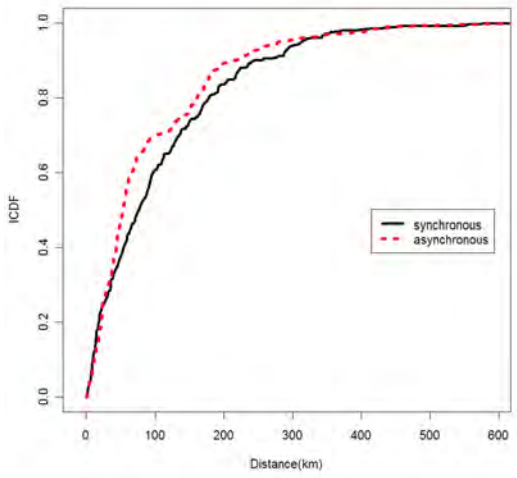


Figure 4: Inverse cumulative density function for error

Another important factor in algorithm choice for probabilistic graphical models is time to convergence. We measured this by counting the number of variables updated in each iteration of the algorithm. To minimize the impact of random starting configuration, we ran 5 trials to 20 iterations with each algorithm, producing the mean updates and error bars shown in figure 5.

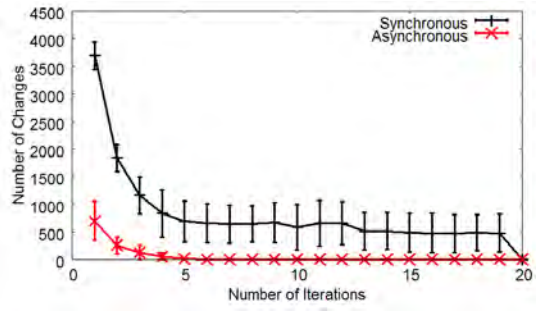


Figure 5: Convergence rates for the two learning algorithms

Overall, the predictions made by the model were well short of the accuracy needed for most real world applications of this system. For example, if the goal was to meet the ship upon portfall, then in many parts of the English channel there would be several potential ports within the 100km radius mentioned above. However, the results do show that asynchronous updates dominate synchronous updates in terms of both probable approximate correctness and convergence rate. We were surprised to find that after only 4 cycles of updates the asynchronous algorithm reached a fixed point in most cases. In contrast, the synchronous algorithm seemed prone to cycles in which each iteration toggled significant number of predictions even after 20 iterations.

5. CONCLUSION AND FUTURE WORK

We presented synchronous and asynchronous expectation maximization algorithms for unsupervised learning in factor graphs. We used these algorithms with a factor graph interpreting AIS data in order to simultaneously detect a map, hidden plans, and transition frequencies between plans. To our knowledge, this was the first report applying general purpose unsupervised algorithms for graphical models to conduct learning with real data. We used the learned models to make projections of portfalls for ships in motion. Although these preliminary results were not accurate enough for real-world application, both prediction accuracy and direct inspection of learned locations and transition probabilities suggested that a reasonable model was being inferred. Our asynchronous method significantly outperformed our synchronous method in terms of both convergence rate and probability of achieving high accuracy in portfall prediction.

In section 3.2 we laid out some principle objectives for unsupervised learning in factor graphs, which we hope can assist future research on this topic. In particular, we would like to make more rigorous the argument associated with (11), so that in future work we can explore properties of the self-consistency probability $\bar{P}$ rather than focusing only on local search. Our appendix contains some initial work in this direction, as we will be exploring $\bar{P}$ as a distribution over partitionings rather than assignment vectors.

There are many areas in which we believe our work can be extended to take advantage of recent developments in supervised learning on PGMs. We are particularly interested in creating unsupervised variants of belief propagation algorithms, where the theory regarding convergence has advanced significantly. The residual belief propagation of [5], in which messages are updated on a schedule that prioritizes "poorly fit" variables, seems especially relevant to our clustering application. In our experiments we saw consistently that some regions of the map stabilized very quickly in terms of cluster locations and transition probabilities, while others were left to slowly improve over many iterations. The result was that our algorithms spent significant computation considering updates for already stable assignments.

The primary focus of our current research is adaptation of algorithms for hierarchical Bayes models [11] to general factor graphs. These models support unsupervised learning with even fewer assumptions: the number of states for a variable class is derived from data and a Dirichlet prior rather than specified in advance. The infinite hidden Markov model of Beal *et. al.* [3] is especially similar in structure to our temporal factor graph, and supports a Gibbs sampler to estimate marginal probabilities in addition to a maximum likelihood assignment.

To improve our AIS application, we are working on algorithms which detect significant locations with higher granularity. This involves both computational challenges and issues of insufficient data, as some regions of the map are far better represented than others in our dataset. Our current

experiments in this direction involve hierarchical Gaussian mixtures to allow finer grained notions of location only in areas where there is supporting data. Another important direction is to expand our model to include additional information, such as ship class, heading, or even ownership. Doing so will give us an opportunity to examine how our algorithms scale with more and different types of factors.

6. ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under the IGERT program (DGE-9972762) for training and research in CASOS, and the Office of Naval Research under Dynamic Network Analysis program (N00014-02-1-0973, ONR N00014-06-1-0921, ONR N00014-06-1-0104). Additional support was provided by CASOS - the Center for Computational Analysis of Social and Organizational Systems at Carnegie Mellon University.

7. REFERENCES

[1] Google earth. http://earth.google.com/.

[2] D. Ashbrook and T. Starner. Using gps to learn significant locations and predict movement across multiple users. *Personal Ubiquitous Comput.*, 7(5):275–286, 2003.

[3] M. J. Beal, Z. Ghahramani, and C. E. Rasmussen. The infinite hidden Markov model. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

[4] T. V. Duong, H. H. Bui, D. Q. Phung, and S. Venkatesh. Activity recognition and abnormality detection with the switching hidden semi-markov model. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 838–845, Washington, DC, USA, 2005. IEEE Computer Society.

[5] G. Elidan, I. McGraw, and D. Koller. Residual belief propagation: Informed scheduling for asynchronous message passing. In *Proceedings of the Twenty-second Conference on Uncertainty in AI (UAI)*, Boston, Massachussetts, July 2006.

[6] D. Koller, N. Friedman, L. Getoor, and B. Taskar. Graphical models in a nutshell. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[7] L. Liao, D. Fox, and H. Kautz. Extracting places and activities from gps traces using hierarchical conditional random fields. *Int. J. Rob. Res.*, 26(1):119–134, 2007.

[8] L. Liao, D. J. Patterson, D. Fox, and H. Kautz. Learning and inferring transportation routines. *Artif. Intell.*, 171(5-6):311–331, 2007.

[9] N. Nguyen, H. Bui, S. Venkatesh, and G. West. Recognizing and monitoring high level behaviours in complex spatial environments. In *CVPR Conference Proceedings*. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2003.

[10] N. Nguyen, D. Phung, S. Venkatesh, and H. Bui. Learning and detecting activities from movement trajectories using the hierarchical hidden markov model. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2005.

[11] Y. Teh, M. Jordan, M. Beal, and D. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[12] J. Yedida, W. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.

8. APPENDIX

Unsupervised learning as maximal partitioning

For any assignment vector $\vec{x}$, there is a corresponding partitioning $\beta(\vec{x})$ which divides X into n unordered clusters, each corresponding to a hidden value. If we assume that factors permit exchangeability between state indexes (which is reasonable, as there is no training data to justify a bias), then the equivalence sets induced by β are also equivalent with respect to optimal parameters:

$$\beta(\vec{x}) = \beta(\vec{x'}) \Rightarrow \vec{w}^*(\vec{x}) = \vec{w}^*(\vec{x'}) \tag{15}$$

β-equivalence extends to whether assignment vectors satisfy (9) and the likelihood the vectors achieve under optimized parameters. Our search for $\vec{x}^*$ can therefore be reduced to a search for $\beta^* = \beta(\vec{x}^*) \in B$, where B is the set of all possible partitionings. Unfortunately, $|B|$ (given by $S(|X|, s)$ where S gives Stirling's number of the second kind) is still far too large to enumerate or sample meaningfully. However, a sensible search for assignment vectors should avoid evaluating multiple vectors in the same β-equivalence class.

An Adaptive Sensor Mining Framework for Pervasive Computing Applications

Parisa Rashidi
Washington State University
Pullman, WA
US, 99164
001-5093351786
prashidi@wsu.edu

Diane J. Cook
Washington State University
Pullman, WA
US, 99164
001-5093354985
cook@eecs.wsu.edu

Abstract
Considering the wealth of sensor data in pervasive computing applications, mining sequences of sensor events brings unique challenges to the KDD community. The challenge is heightened when the underlying data source is dynamic and the patterns change. In this work, we introduce a new adaptive mining framework that detects patterns in sensor data, and more importantly, adapts to the changes in the underlying model. In our framework, the frequent and periodic patterns of data are first discovered by the Frequent and Periodic Pattern Miner (FPPM) algorithm; and then any changes in the discovered patterns over the lifetime of the system are discovered by the Pattern Adaptation Miner (PAM) algorithm, in order to adapt to the changing environment. This framework also captures vital context information present in pervasive computing applications, such as the startup triggers and temporal information. In this paper, we present a description of our mining framework and validate the approach using data collected in the CASAS smart home testbed.

Categories and Subject Descriptors
H.2.8 [**Database Management**]: Database Applications– *data mining;* I.2.6 [**Artificial Intelligence**]: Learning– *knowledge acquisition;* H.4.m [**Information Systems**]: Information system Applications– *Miscellaneous.*

General Terms
Algorithms, Design, Experimentation, Human Factors.

Keywords
Sensor data mining, Sequential mining, Pervasive computing applications, Smart environments, Adaptation.

2nd International Workshop on Knowledge Discovery from Sensor Data (Sensor-KDD 2008), 2008, Las Vegas, Nevada, US.

1. INTODUCTION
With remarkable recent progress in computing power, networking equipment, sensors, and various data mining methods, we are steadily moving towards ubiquitous and pervasive computing, into a world entangled with abundant sensors and actuators. As a result, there is a wealth of sensor data that can be analyzed with the goal of identifying interesting patterns. For example, by discovering repetitive sequences (frequent or periodic), modeling their temporal constraints and learning their expected utilities, we can intelligently automate a variety of tasks such as repetitive daily tasks in homes, or assembly sequences in manufacturing floors. Mining sequences of sensor events brings unique challenges to the KDD community, and the challenge is even heightened more when the underlying data source is dynamic and the patterns change.

In this work, we introduce a new adaptive mining framework for use with pervasive computing applications, which will detect and adapt to changing patterns in the sensor data. These patterns can be expressed as a set of time ordered sequences, and discovering and adapting to the changes in such sequences can be achieved by using a sequence mining algorithm tailored to the special domain requirements of pervasive computing applications. The pervasive computing special requirements include utilizing context information such as startup triggers and temporal information, a unified framework for discovering periodic and frequent patterns, and most importantly adaptation over the lifetime of the system. Startup triggers are events that can trigger another action, e.g. a person entering a room can act as a potential startup trigger for the light in the room to be turned on. Triggers, which are absent in most traditional data mining methods, are a key data aspect in pervasive computing environments and need to be processed accordingly. At the same time, as already mentioned, discovering and adapting to the changing patterns over the lifetime of the system is a fundamental part of most pervasive computing applications. In this work, we introduce a framework to address these requirements. Providing such an adaptive framework is a significant advantage over previous sequential mining algorithms applied to pervasive computing applications, which assume the learned model is static over the lifetime of the system [1, 6, 16]. In addition, utilizing context information such as startup triggers helps to better model the complex environment.

In our framework, the frequent and periodic patterns are first discovered by the Frequent and Periodic Pattern Miner (FPPM) algorithm which discovers patterns of arbitrary length and inexact

periods. The core component of our model, the Pattern Adaptation Miner (PAM) algorithm, detects changes in the discovered patterns over the lifetime of the system and adapts to the dynamic environment. PAM is able to detect changes autonomously or initiate the detection process by receiving feedback from the user. The startup triggers for discovered patterns are detected and updated accordingly during discovery and adaptation process. Also, the proposed mining framework is able to capture and utilize temporal information by modeling the discovered patterns in a hierarchal model called Hierarchal Activity Model (HAM).

In order to show a tangible application of our model, we evaluate it in the context of a popular pervasive computing application: smart environments. In recent years, smart environments have been a topic of interest for many researchers who have the goal of automating resident activities in order to achieve greater comfort, productivity, and energy efficiency [1, 6, 16]. A smart home uses networked sensors and controllers to try to make residents' lives more comfortable by acquiring and applying knowledge about the residents and their physical surroundings. In our discussions, we define an event as a single action such as turning on the light, while an activity is a sequence of such events, e.g. turning on the light – turning on the coffee maker, which is composed of two events. We also use the terms pattern, sequence, and activity interchangeably, depending on the context. The events can represent data generated by different sensors such as motion sensors, or by a device that is manipulated by a power-line controller, such as home appliances. Discovering how the resident performs routine activities in daily life facilitates home automation and makes it possible to customize the automation for each person. In this work we primarily consider discovering frequent and periodic activities, as automating these activities makes the environment responsive to the resident and removes the burden of repetitive tasks from the user.

In the next sections, we will describe our model in more detail. We also show the results of our experiments where we validate our approach on data generated by a synthetic data generator as well as on real data collected in a smart workplace testbed located on campus at Washington State University.

2. RELATED WORKS

Despite increasing progress in pervasive computing applications and especially smart environments, the crucial issue of mining sensor data in a changing environment and maintaining an adaptive model of the environment still has not been explored in depth by the research community. An adaptive solution is especially important for smart environments, as humans often change their habits and lifestyle over time. In the smart home research community, very few adaptive solutions have been proposed, such as the simple reactive fuzzy method proposed in [15], which does not consider any complex sequential tasks, and also does not discuss how such changes can be discovered from daily activity data. In data mining community, some adaptive mining techniques have been proposed for data streams, such as the adaptive associative rule mining algorithm [9]; or methods for adapting to the memory, time or data stream rate constraints [5]. The constraint adaptive methods [5] do not consider adapting to the change in the data; in additions, mentioned works [5, 9] do not address the adaptation problem for sequence mining methods where many different aspects of a sequence can be changed over time. We try to address the adaptation problem in smart environments by using the PAM algorithm to discover any changes in the resident's activities data.

Another significant sensor mining requirement in some pervasive computing applications such as smart environments is detecting the contextual information that describes features of the discovered activities. Context information is valuable in order to better understand the activities and potentially automate them. In our model we will capture important context information such as startup triggers as well as temporal information including event durations, and start times. Previous sensor mining methods applied to pervasive computing applications do not deal with startup triggers' concept systematically, and they typically do not differentiate between real sensor data (startup triggers) and actuator data (data obtained from appliances through power-line controllers). As we will see, this concept plays an important role in pervasive computing applications. In addition, other related techniques treat events in the sequence as instantaneous and ignore the conveyed temporal information. Laxman and Sastry [7] do model some temporal information, by incorporating event duration constraints into the episode description. A similar idea is proposed by Lee et al. [8], where each item in a transaction is associated with a duration time. Bettini et al. [3] place particular emphasis on the support of temporal constraints on multiple time granularities where the mining process is modeled as a pattern matching process performed by a timed finite automaton. Our work is different from these previous approaches, as we mostly focus on estimating time distributions for different time granules by utilizing combinations of local Gaussian distributions for modeling temporal information of each pattern, rather than merely considering temporal granules. Using a combination of multiple Gaussian and several temporal granules allows us to more accurately express and model duration and start times.

The basic step of our model is to first find patterns of interest, to find frequent activities, we exploit a subset of the temporal knowledge discovery field, that usually is referred to as "discovery of frequent sequences" [12], "sequence mining" [2] or "activity monitoring" [4]. In this area, the pioneering work of Agrawal to design the Aprori algorithm [2] was the starting point. Since then, there have been a number of extensions and variations on the Apriori algorithm [10]. We use a variant of the Apriori algorithm to find frequent patterns. However, in pervasive computing applications, such as smart environments, not only it is important to find frequent activities, but also those that are the most regular, occurring at predictable periods (e.g., weekly, monthly). If we ignore periodicity and only rely on frequency to discover patterns, we might discard many periodic events such as the sprinklers that go off every other morning or the weekly house cleaning. Therefore we need a unified framework that can simultaneously detect both frequent and periodic activity patterns. There are a number of earlier works that try to handle periodicity, such as the Episode Discovery (ED) algorithm [16]; however, the problem with ED and most other approaches is that they look for patterns with an exact periodicity, which is in contrast with the erratic nature of most events in the real world. Lee, et al. [8] define a period confidence for the pattern, but they require the user to specify either one or a set of desired pattern time periods. In our approach, we do not require users to specify predefined period confidence for each of the periods, as it is not realistic to force users to know in advance which time periods will be

appropriate. In addition, the periodicity of patterns for many practical applications is dynamic and change over time as internal goals or external factors change. To overcome this problem, we define two different periodicity temporal granules, to allow for the necessary flexibility in periodicity variances over two different levels. A fine grained granule is defined for hourly periods which can span several hours up to 24 hours, and a coarse grained granule is defined for daily periods which can span any arbitrary number of days. None of these periodicity granules require a period to be exact: fine grained periods provide a tolerance of up to one hour and coarse grained periods provide a tolerance of up to one day. A more detailed description of the framework is provided in the next section of the paper.

3. MODEL DESCRIPTION

We assume that the input data is a sequence of individual event tuples. Each tuple is in the form $<d_i, v_i, t_i>$ where d_i denotes a single data source like a motion sensor, light sensor, or appliance; v_i denotes the state of the source such as on or off; and t_i, denotes the occurrence time for this particular event. We assume that data is not in stream format, rather it is sent to a storage media, and the data mining process is carried out offline at regular time intervals such as weekly, or on demand as will be described later. Table 1 shows a sample of collected data.

Table 1. Sample of collected data.

Source (d_i)	**State** (v_i)	**Timestamp** (t_i)
Light_1	ON	05/15/2007 12:00:00
Light_2	OFF	05/15/2007 12:02:00
Motion_Sensor_1	ON	05/15/2007 12:03:00

Our FPPM algorithm, similar to the Apriori method, takes a bottom-up approach. However, unlike the Apriori algorithm, not only does it discover frequent sequences, but it also tries to find periodic sequences and their periodicity. In the first iteration, a window of size ω (initialized to 2) is passed over the data and every sequence of length equal to the window size is recorded together with its frequency and initial periodicity. Frequency is computed as the number of times the sequence occurs in the dataset, and periodicity represents the regularity of occurrence, such as every three hours or weekly. After the initial frequent and periodic sequences have been identified, FPPM incrementally builds candidates of larger size. FPPM extends sequences that made the cutoff in the previous iteration by the two events that occur before and after the current sequence instances in the data. For simplicity, we call the event right before the current sequence as its prefix, and the event right after it as its suffix. FPPM continues to increase the window size until no frequent or periodic sequences within the new window size are found or a limit on the window size is reached (if declared by user). Each discovered sequence is considered as being either periodic or frequent. At the end of the mining session, if a specific sequence is found to be both frequent and periodic, for convenience and simplicity we report it as frequent.

The FPPM algorithm is used to discover the initial patterns. In order to detect the changes in the discovered patterns, PAM provides two options. First, a fast demand-based option called *explicit request* allows users to highlight a number of activities to be monitored for changes. Second, a slower automatic option called *smart detection* automatically looks for potential changes in all patterns, based on a regular mining schedule. The explicit request mechanism detects changes in specified patterns, such that whenever a pattern is highlighted for monitoring, PAM collects data and tries to find potentially-changed versions of the specific pattern. These changes may consist of new activity start times, durations, startup triggers, periods, or structure changes (the sequence's events and their order). Structure change is detected by finding new patterns that occur during the times that we expect the old pattern to occur. Other parameters changes are detected by finding the same structure with other different parameters (e.g., different timing or startup triggers). All changes above a given threshold will be considered as different versions of the pattern and will be shown to the user through our user interface. In addition, our smart detection mechanism automatically mines collected data at periodic intervals (e.g., every three weeks) to update the activity model. The smart detection method adapts slower than the explicit request method; however it removes the burden of specifying activities to be monitored. Figure 1 shows the overall architecture of the framework.

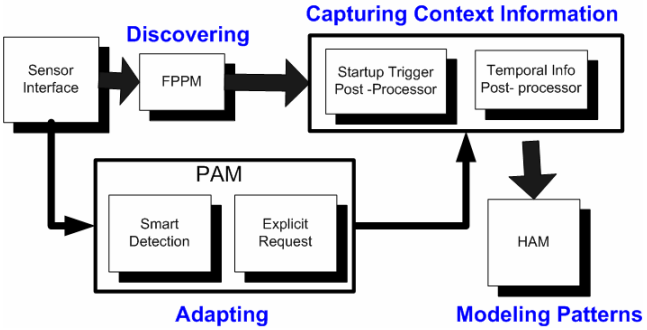


Figure 1. Components of our data mining framework.

3.1 Basic Frequent Pattern Finding

The algorithm finds frequent patterns by visiting the data iteratively. In the first pass, the whole data is visited to calculate initial frequencies and periods, and in the next passes only a small portion of data is revisited. In the first pass, a window ω of size *2*, is slid through the entire input data and every sequence of length ω, along with its frequency, *f*, is recorded. The frequency of each sequence refers to the number of times the sequence is encountered in the data. In order to mark a sequence as a frequent sequence, its frequency should satisfy certain conditions.

Drawing on results from information theory, we evaluate the frequency of a sequence based on its ability to compress the dataset, by replacing occurrences of the pattern with pointers to the pattern definition [11]. Calculating this compression is tricky for smart home data, as the size of the dataset may not be fixed due to varying activity levels (e.g. an active day will generate a lengthy event data stream). We compute the compression according to Equation 1, where f_a represents the frequency of sequence *a*, *t* represents the input data length in hours and *C* represents the compression threshold. For a sequence to be considered as frequent, the following condition should hold:

$$\frac{|a| * f_a}{t} > C \quad (1)$$

Previous approaches [3] have calculated the input data length in numbers of tuples, rather than in units of time, which results in making discoveries of frequent patterns dependent on the resident's activity level. For example, if the resident has a very active day, the input data will contain more tuples and therefore the input length will take on a high value, but if resident is not active that day, the input length will have a low value. Therefore for an activity with the same frequency such as making coffee twice a day, the compression value will be dependent on the resident's activity level. In our approach, the input length is measured in time units rather than tuples, and an activity such as making coffee twice a day will be discovered as a frequent pattern, independent of the activity level of the resident.

3.2 Basic Periodic Pattern Finding

In addition to finding frequent patterns, FPPM also discovers periodic patterns. Calculating periods is a more complicated process. To calculate the period, every time a sequence is encountered, we will compute the elapsed time since its last occurrence. More precisely, if we denote the current and previous occurrence of a sequence pattern as s_c and s_p, and their corresponding timestamps as $t(s_c)$ and $t(s_p)$, then the distance between them is defined as $t(s_c) - t(s_p)$. This distance is an initial approximation of a candidate period. To determine periodicity, as mentioned before, two different periodicity granules are considered: coarse grained and fine grained period granules. A fine grained granule is defined for hourly periods which can span several hours up to 24 hours, and a coarse grained granule is defined for daily periods which can span any arbitrary number of days. None of these periodicity granules require a period to be exact, fine grained periods provide a tolerance of up to one hour and coarse grained periods provide a tolerance of up to one day. One can claim that only a fine grained period can be sufficient to show periodicity of an activity; for example, every Sunday can be represented by a period of 7 × 24 hours instead of a coarse period of 7 days. This claim is not substantiated in practice, as taking such an approach will require the activity to happen every 7 × 24 hours with a tolerance of just 1 hour. This is not a realistic assumption, as we want to allow for more tolerance in coarse grained periods. For example, consider the scenario when a resident might watch TV every Sunday, but at different times; in this case, a fine grained period is not able to catch periodicity as its tolerance is just one hour, while a coarse grained period is easily able to catch such a periodicity as it allows for a tolerance of up to one day. The same claim can be made about other time granules, but for sake of simplicity and demonstrating the basic idea, we will just consider the two levels of temporal granules.

To construct periods, a lazy clustering method is used, such that as long as an activity's period can be matched with the previous periods (with a tolerance of one hour for fine grained, and one day for coarse grained), no new period is constructed. If the new activity has a period different than previous periods, a new period is constructed and is added to the tentative list of fine grained or coarse grained periods. In order to make sure that candidate periods are not just some transient accidental pattern, they are kept in a tentative list until they reach a confidence frequency value. If the periodicity for a sequence is consistent a threshold number of times (e.g. 90%), the pattern is reported as periodic, and it is moved to the consolidated period list. Updating tentative and consolidated lists is performed dynamically and a period can be moved from one list to another several times. Such a schema helps to eliminate any transient periods based on current or future evidence, resulting in an adaptive evolvable mining approach. In this approach, whenever more data becomes available as a result of regularly scheduled mining sessions, the periods are revisited again, and if there is any period that does not meet periodicity criteria anymore, it will be moved from the consolidated list into the tentative list. Later if we find again more evidence that this period can be consolidated, it will be moved back into the consolidated list; this results in a more robust model that can evolve and adapt over time.

In contrast to frequent pattern finding which uses a single frequency (compression) threshold for detecting all frequent sequences, we can not use a single confidence threshold for detecting all periodic sequences. Rather we need to tailor the confidence threshold to each specific periodic pattern, because the number of times that an activity occurs can vary depending on its period, making a low frequency sequence a valid periodic candidate in some cases, and a relatively high frequency sequence an invalid candidate in other cases. For example, consider a scenario where our input file contains two weeks of resident activity data and there are two periodic activities: a_1 with a period of one hour and a_2 with a period of 4 days. In this scenario, the number of times we expect to see a_1 would be much more than a_2. Therefore, a single confidence value can not work for both cases. To work around this problem, as we scan through the data we calculate and update the expected number of occurrences, $E(f_a)$, for an activity a, up to current point in data. For example, if our initial period estimate for activity a is 5 hours and so far we have scanned through 10 hours of data, we expect to see two occurrences of activity a in an ideal case. Considering the erratic nature of real-world events, not all sequences will be repeated ideally. To deal with this problem, for each new occurrence of a we check it against the following equation where f_a is actual number of occurrences observed so far and ζ is a predefined threshold that determines what percentage of expected occurrences is sufficient to move a candidate periodic sequence from tentative list into consolidated list (e.g. a rigorous approach can consider it to be above 95%).

$$\frac{E(f_a)}{f_a} > \zeta \quad (2)$$

The predefined threshold can be different for coarse grained and fine grained periods, which we denote as ζ_f and ζ_c. In the next section, we will describe in more detail how the iterative model constructs sequences of larger sizes from the basic sequences.

3.3 Iterative Pattern Discovery

After the initial frequent and periodic sequences of length two have been identified in the first iteration, the next iteration begins. In repetitive iterations, FPPM does not revisit all the data again. Instead, the algorithm attempts to extend the window by the two events before and after the discovered frequent or periodic sequence instances (as we mentioned before, the prefix and suffix) and determines if the extended sequences again satisfy the

periodicity or frequency criteria. If we denote the length of the sequence before extending as *l*, then the extension process might result in finding a frequent or periodic sequence of length *l+1*, if it has been extended by either prefix or suffix; and it might result in finding a frequent or periodic sequence of length *l+2*, if it has been extended by both its prefix and suffix. For example, consider Figure 2, where "BC" is a frequent pattern. Now if it is extended by its prefix (Figure 3) , then it results in another frequent pattern "ABC". However, extending it by its suffix (Figure 4) or by both suffix and prefix (Figure 5) does not result in a frequent pattern.

Figure 2. Frequent pattern "BC"

Figure 3. Extending "BC" pattern by its prefix.

Figure 4. Extending "BC" pattern by its suffix.

Figure 5. Extending "BC" pattern by both suffix and prefix.

Incrementing the window size will be repeated until no more frequent (periodic) sequences within the new window size are found, or a limit on the window size is reached (if declared by the user). The symmetric expansion of window allows for patterns to grow both forward and backward in time, and the incremental expansion allows for discovery of variable-length patterns.

3.4 Startup Trigger Information

An important notion that can be used to improve activity prediction in real world data mining applications such as smart environments is the discovery of startup triggers in sequences. Basically, a trigger is an event which causes an activity to start. A startup trigger paradigm can be compared to the event programming paradigm, in which for example a mouse click event (startup trigger) can trigger an action (such as a menu appearing). In pervasive computing and smart environments, the same paradigm applies; for example, if a person enters a dark room, it can be considered as a startup trigger for turning on the light; or as another example, running out of milk in the refrigerator can be a trigger to initiate a supermarket purchase reminder. These startup triggers will also appear in the collected data and therefore it is necessary to augment the data mining model with a trigger processing component that is able to recognize triggers, in order to facilitate automation of activities.

A trigger is typically part of an FPPM's discovered sequence. For example, if a person turns on the light every time s/he enters the room, FPPM will discover the sequence "enter room – turn on light" from the sensor and power-line controller data. By examining this sequence, we will find out that a startup trigger is not actually part of an automated activity; rather it is a condition that starts an automated activity (in this case, turning on light). We also can see that the startup triggers are in fact the sensor data, while automated activities represent data from actuators (power–line controllers attached to appliances). In our model, we will process the discovered sequences from FPPM in such a way that a sequence merely represents automations and only contains data from actuators, though it can have several startup triggers assigned to it. For example, the previous sequence "enter room – turn on light" will be converted to a single event sequence "turn on light" with the "enter room" triggers assigned to it. A sequence can have several triggers assigned to it. We adopt the following general policy for processing triggers:

- If a trigger appears at the beginning of a sequence, it should be removed from the beginning of the sequence and be added to the list of assigned start up triggers.
- If a trigger appears at the end of a sequence, it has no effect on the sequence; we simply remove it from the sequence.
- If several triggers occur consecutively, we will just consider the last one, discarding the other ones.
- If a trigger occurs in the middle of a sequence, we will split the sequence into two sequences and the trigger will be assigned to the second sequence (see Figure 6).
- If a sequence contains more than one trigger, the above steps are repeated recursively.

Note that we assume that frequency and period would be the same for split sequences as the original sequence; but, the compression value may change as it depends on the sequence's length. So, the compression value is computed for recently split sequences and if it does not satisfy the frequency criteria, recently split sequences will be removed from the frequent patterns' list. Also during the sequence splitting process, sequence might reduce to one of the already existent sequences. In this case, one approach is to repeat the data mining process again to find any existing relation between these two sequences (e.g., they might have different periods). However, for the sake of simplicity and also efficiency, we do not mine the data again; rather we will choose the sequence with the highest compression value and simply discard the other.

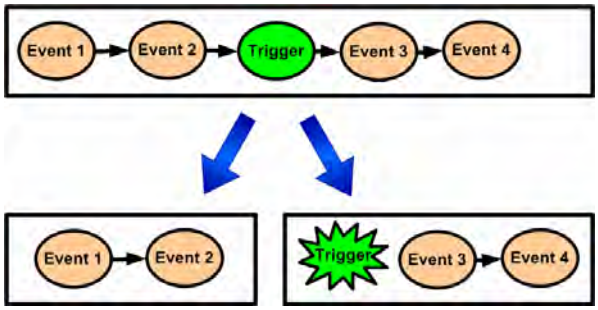


Figure 6. A trigger causing a sequence to be split.

3.5 Temporal Information

After activity structure and periods have been discovered by FPPM, the sequences will be organized in a Hierarchal Activity Model (HAM) structure, which filters out activities according to two temporal granule levels of day and hour (see Figure 7). In addition to finding frequent and periodic patterns, FPPM records duration and start times of events by processing their timestamps. These durations and start times are revisited by PAM when looking for changes. HAM captures the temporal relationships

between events in an activity by explicitly representing sequence ordering as a Markov chain [13]. Each activity will be placed in a HAM leaf node (sensor level) according to its day and time of occurrence, and will have a start time and event durations assigned to it. There have been earlier approaches to modeling durations of states in a Markov chain such as the approach by Vaseghi [14] in which state transition probabilities are conditioned on how long the current stated has been occupied. In our model, for each activity at each time node, we describe the start time and the duration of each individual event using a normal distribution that will be updated every time new data is collected. If an activity is located in different time/day nodes within the HAM model, multiple Gaussian distributions for that activity will be computed, allowing HAM to more accurately approximate the start time and duration values, by using multiple simple local normal functions that represent a more complex function.

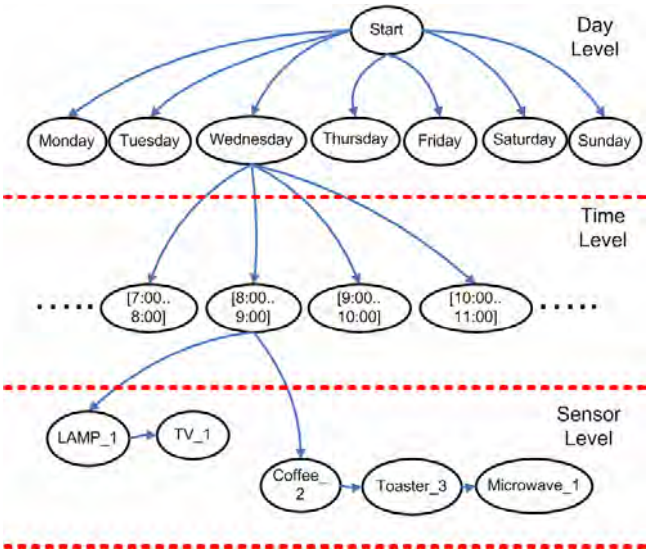


Figure 7. Example HAM model.

3.6 Adaptation

Most pervasive computing mining algorithms generate a static model, by assuming that once the desired patterns from the sensor data have been learned, no changes need to be applied to maintain the model over time. The challenge is heightened when the underlying data source is dynamic and the patterns change. In the case of smart environments, as we know, humans are likely to change their habits and activity patterns over time depending on many factors, such as social relations, seasonal and weather conditions and even emotional states. Therefore, a static model cannot serve the purpose of a long-term solution for a smart environment. Instead, we need to find a way to adapt to the changes that occur over time.

As mentioned earlier, there are two options for detecting changes in discovered patterns. The fast explicit request option allows users to highlight a number of patterns to be monitored for changes; and the smart detection option automatically looks for potential changes of all patterns, based on a regular mining schedule.

For every activity, we maintain a potential value, *Q*, which reflects the amount of evidence against or for an activity as being frequent or periodic, in other words the degree to which it should be considered for automation. The potential value can be increased or decreased through a compensation effect or a decay effect, as will be described. If potential value falls below a certain activity threshold, the activity is discarded from the model, in other words it will be forgotten. Maintaining a potential value for each discovered activity can help us distinguish transient changes from long-term changes that still might be accompanied by a noise element. The potential value is increased by using the following formula:

$$Q = (Q + \alpha * r) \qquad (3)$$

In the above equation, r denotes the evidence value (predefined, e.g. 0.5), and $\alpha \in [0,1]$ denotes the learning rate which is a small value to imply a gradual history-preserving nature of learning as there is no strong evidence or user guidance that a change is a permanent change and not a temporary one. In order to avoid infinite growth of potential, we allow value functions to grow only in the range of [-1...1]. Note that when updating the potential value, we do not differentiate between different events that comprise an activity; instead, we assign just a single value to an activity.

In addition to the compensation effect, we also employ a decay effect which subtracts a small value ε from all activities' values at each time step θ. By applying a decay function, the value of any activity during an arbitrary time interval Δt is decreased by:

$$Q = Q - \frac{\varepsilon * \Delta t}{\theta} \qquad (4)$$

The decay effect allows for those activity patterns that have not been perceived over a long period of time to descend toward a vanishing value over time, or in an intuitive sense to be forgotten. This helps to adapt to changing environment in smart detection method. The effect of the decay function is compensated through compensation effect in a way that the potential value remains bounded.

3.6.1 Explicit Request

In this method, whenever a pattern is highlighted to be monitored, PAM analyzes recent event data and looks for changes in the pattern, such as the pattern's start time, durations, periods, or the pattern structure (the component events with their temporal relationships). Without loss of generality, we refer to two different categories of changes: changes that preserve the structure and changes that alter the structure.

Structure change is detected by finding new patterns that occur during the same times we expect the old pattern to occur; assuming that the start time can act as a discriminative attribute. First, PAM looks for a pattern, a, such that its start time, s_a, is contained within the interval $\Delta\delta = \mu_a \pm \sigma_a$, where μo and σ_a denote the mean and standard deviation of the original pattern's start time distribution. These locations are marked by the algorithm in order to avoid looking at all data. PAM is looking for different patterns within these start time intervals, in which we expect to see the original pattern. It moves a sliding window of size ω (initially set to 2) over the interval and incrementally increases the window size at every iteration. The window size does not increase when no more frequent or periodic patterns of length ω can be

found. A frequent pattern can easily be extended beyond the marked point, as we require only its start time to be contained within the marked interval. This process results in finding a new pattern which may be longer, shorter, or have other different properties than the original one.

In the case where structure is preserved, we first mark all the occurrences of the original activity in the data, and based on these occurrences calculate properties such as new durations, new start times or new periods. After results from both cases have been collected, the PAM algorithm reports the list of changes that can be accepted or rejected by the user.

3.6.2 Smart Detection

In the smart detection method, PAM automatically mines the data regularly to update the model and uses the decay and compensation effects to adapt to changes. This approach is slower than the explicit request method, because the changes might not be detected until the next scheduled mining session. After every mining session, the discovered patterns will include a mixture of new and previously-discovered patterns. For new patterns, we simply can add them to the model. For previously existing patterns, if the pattern shows no change, then PAM applies the compensation effect to indicate observation of more evidence for this pattern (Equation 3). However, if the pattern shows some changes, we will add the modified patterns to the model, while also preserving the original pattern, as there is no explicit evidence that this change is a permanent change. To achieve adaptation in this case, we will leave it to the compensation effect and decay functions to decide over time which version is more likely. The compensation effect will increase the value of the more frequently-observed version of the pattern while the decay function dominates for a less-observed pattern. As a result, the value of patterns that have not been observed for a long time will fall below the activity threshold; and will eventually be removed from the model. This again results in adapting to the changes in the environment.

In order to find out when a changed pattern replaces the original pattern, we can use the following analysis. If we denote the original pattern as O and the modified version of the pattern as M, then we can calculate the number of times the decay function should be applied, in order for O to be dominated by M. Assume at time t_i, pattern M is discovered for the first time and its potential value is assigned an initial value of Q_i^M. Also consider that the potential value for pattern O is initially Q_i^O. After time t_i, the decay function will periodically decrease the value of both patterns, while Q_i^M also increases each time M is observed. The potential value for pattern O, Q_u^O, after j applications of the decay function and at time t_u, will be:

$$Q_u^O = Q_i^O - \frac{\varepsilon * \Delta t}{\theta} \qquad (5)$$

We also know that in order for the original pattern to be perfectly forgotten, its value function should be below an activity threshold, i.e. $Q_u^O < \sigma$. Substituting Equation 5 into $Q_u^O < \sigma$ leads to:

$$\frac{Q_i^O - \sigma}{\varepsilon} < j \qquad (6)$$

The above inequality shows the minimum number of times that the decay function should be applied to a pattern before it's forgotten. At the same time, if we consider l observation due to regular mining sessions, Q_i^M will be changed as following:

$$Q_u^M = Q_i^M + l\alpha_o r_o - j\varepsilon \qquad (7)$$

In order for Q_u^M to have a value greater than the activity threshold, we require that $Q_u^M > \sigma$. If we consider ΔT as an arbitrary time interval, and p as the period of regular mining (e.g. every week), then l can be defined in terms of m and ΔT, as $\Delta T/p$. Substituting Equation 7 into $Q_u^M > \sigma$ and considering ΔT and p leads to:

$$p < \frac{\alpha_o r_o * \Delta T}{\sigma + j\varepsilon - Q_i^M} \qquad (8)$$

Equation 8 shows how frequently the old patterns will be replaced by new patterns.

4. EXPERIMENTATION

Our goal is to develop a sensor mining model that unifies frequent and periodic pattern mining for pervasive computing applications. We desire that the model, as is essential for these applications, adapt to changes in those patterns over time and automatically discover startup triggers and necessary temporal information. Here we evaluate our model using synthetic and real data collected in CASAS, our smart environment testbed located at Washington State University.

4.1 FPPM Evaluation

To provide a controlled validation of the FPPM algorithm, we implemented a synthetic data generator that simulates sensor events corresponding to a set of specified activities. Timings for the activities can be varied and a specified percentage of random events are interjected to give the data realism. In addition to synthetic data, to evaluate FPPM on real world data, we tested it on data obtained through sensors located in one room of the CASAS smart workplace environment. This physical test-bed is equipped with motion sensors, rug sensors, light sensors, and controllers for the lamps (Figure 8).

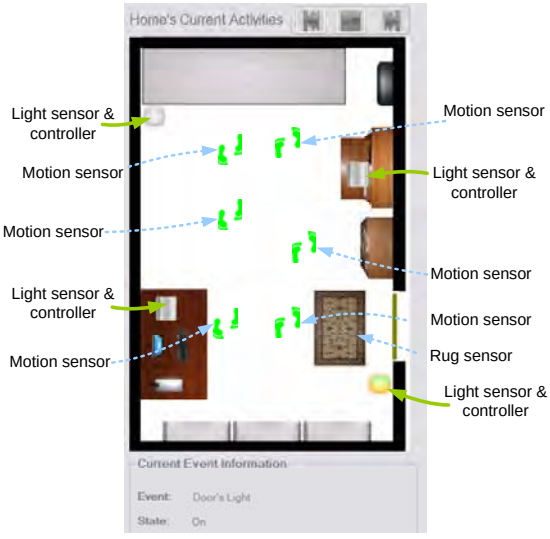


Figure 8. Testbed floor plan and sensor layout.

We initially generated one month of synthetic data that contained events for six activities. Five of the activities were event sequences of length two and the remaining activity was an event sequence of length three. One activity included an "open the door" trigger at the end and another included an "open the door" trigger in the middle of the sequence. The periods for these activities ranged from two to five hours. The devices and sensors that were simulated in this scenario were the same as those installed in WSU's AI Lab. Details of the six simulated activities are listed here by start time, period, and components events with a duration of five minutes imposed for each event.

Table 2. Generated activities.

	Start Time	**Period** (hourly)	**Events**
1	13:00	2	DoorLight ON, LeftDeskLamp ON
2	13:30	3	WhiteboardLight ON, OpenDoor
3	14:25	5	RightDeskLamp ON, WhiteboardLight ON
4	14:55	2	LeftDeskLamp ON, OpenDoor, RightDeskLamp ON
5	15:35	3	LeftDeskLamp ON, WhiteboardLight ON
6	15:55	3	RightDeskLamp ON, DoorLight ON

Our expectation was that FPPM would correctly identify all six activities with their corresponding periods, triggers, start times, and durations. In fact, FPPM was able to find all of the activities with their correct periods. In addition, FPPM identified "open the door" trigger where it existed and accurately divided the sequence with a trigger in the middle into two subsequences where the trigger initiated the second subsequence. When faced with two events that are scheduled to occur at the same time, the synthetic data generator randomly picks just one of the events to simulate. Because of this design feature, the frequency of some of the detected activities was not as high as expected and thus the compression rates were lower than anticipated. For example, activity 1 had a compression rate of 6.7 instead of 8.0 and activity 5 had a compression rate of 6.0 instead of 6.6. However, in some cases the compression rates were higher than expected due to generation of similar random events. For example, split activities of activity 4 had a compression rate of 4.8 instead of 4.0. In other cases, the compression rates were as expected with minimal differences.

In addition to these synthetic data experiments, we also tested FPPM on real smart environment data. Because we needed to validate that the discovered patterns align with what we knew existed in the data, we recruited a participant to execute a simple script in the smart workplace environment. The participant moved through the environment shown in Figure 5 for about an hour, repeating the script ten times. In order to inject some randomness into the data, the participant was asked to perform random activities for about one minute in step three of the script. The script is defined as follows:

1. Turn on right desk light, wait 1 minute.
2. Turn off right desk light.
3. Perform random activities for 1 minute.

Because the testbed area was fairly small, the participant inadvertently created patterns in the random actions themselves. In addition, the motion sensors picked up slight motions (such as hand movements) which resulted in a randomly-triggered pattern that occurred between steps 1 and 2 in the script, which FPPM then split into two subsequences. The light sensor also occasionally fired in the lab after the desk light was turned on. Despite these issues that occur in a real-world situation, FPPM was able to accurately discover the following patterns:

- Right desk lamp on, Compression: 12, Trigger: Walking nearby
- Right desk lamp off, Compression: 9
- Left desk lamp on, Compression: 2
- Right desk lamp on, Right desk lamp off, Compression: 10
- Whiteboard light on, Compression: 2

The first and second patterns are the result of splitting the sequence "Right desk lamp off, Random event, Right desk lamp on" into two subsequences. The "walking" trigger is correct because after turning the light off, the participant performs a random action and heads back to the right desk to turn on the light, which usually involves walking across the room to reach the desk. The difference in compression values between the first and second sequences is due to multiple triggers from the light sensor for a single light on or light off action. The third sequence is the result of a random activity; the compression value is relatively small compared to the main script activities. The fourth sequence reflects the embedded activity, and the last sequence is a frequent activity associated with random events, again with a smaller compression value. These results support our claim that FPPM can detect patterns correctly.

4.2 PAM Evaluation

In order to evaluate PAM's ability to adapt to new patterns, we again tested it on both synthetic and real data. We hypothesize that PAM can adapt to changes in discovered patterns. To test the hypothesis, for our first experiment we created one month of synthetic data with six embedded scenarios, the same as in previous experiment with FPPM. After FPPM found corresponding activity patterns, we highlighted the third activity to be monitored for changes. We then changed the activity description in the data generator such that all event durations were set to 7 minutes, instead of 5 minutes. PAM detected the changes accordingly by finding a new duration of 6.44 minutes, which is quite close to the actual 7 minute change. The data generator does have an element of randomness, which accounts for the discrepancy between the specified and detected time change. In similar tests, PAM was also able to detect start time changes from 13:00 to 13:30, and structure changes (omission or addition).

In the next step, we tested PAM on real world data using our smart environment testbed. A volunteer participant entered the room and executed two different activities:

- Turn on right lamp(1 min), perform random actions
- Turn on left lamp(1 min), perform random actions

The first activity was repeated 10 times over the course of two hours with random events in between. Then the participant highlighted the activity for monitoring and performed the second scripted version by changing the duration from 1 to 2 minutes. PAM detected the duration change as 1.66 minutes. The change was made to the correct parameter and in the correct direction, but did not converge on an accurate new value due to the detection of

other similar patterns with different durations. These experiments validate that PAM can successfully adapt to resident changes even in real-world data. We also found that in addition to changes in duration, AAM detected some changes in start time. This is another correct finding by AAM. As in the second dataset, we changed the duration of all events in all scenarios which resulted in a shifted start time for all scenarios, in our case 14:55 instead of original 14:25.

We also empirically validated our theoretical analysis to see how fast original patterns will be replaced by modified versions. To evaluate this, we designed an experiment in which we generated two sets of synthetic data similar to the first experiment. We then validated the adaptation capability for different decay values and different initial value function (see Figure 9). Our findings are consistent with our expectation, validating that PAM can successfully adapt to resident changes even in real-world data.

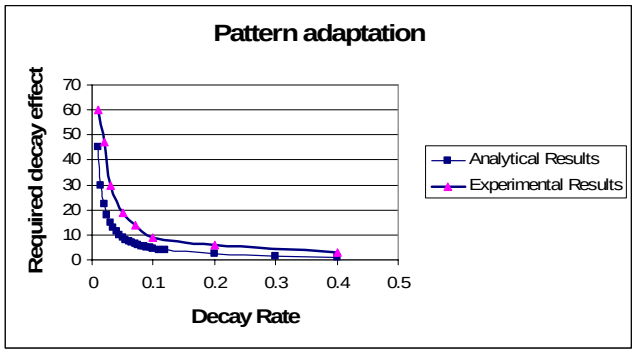


Figure 9. Changes in decay rate.

5. CONCLUSIONS

In this paper, we introduced a new adaptive mining framework to detect sequential patterns in sensor event data and to adapt to changes in the underlying model. This framework uses the FPPM algorithm to discover frequent and periodic event patterns; and FPPM is complemented by using the PAM algorithm, which detects changes in the discovered patterns. Our framework also detects startup triggers in addition to temporal information. We presented the results of our experiments on both synthetic and real data. In our ongoing work, we plan to extend the FPPM method to include discovery of parallel activities, besides detection of abrupt changes. We also intend to discover additional types of contextual information that allow the model to better generalize over discovered sequences.

6. REFERENCES

[1] G.D. Abowd and E.D. Mynatt. Designing for the human experience in smart environments. In Smart Environments: Technology, Protocols and Applications, pages 153-174, 2005.

[2] R. Agrawal and R. Srikant. Mining Sequential Patterns, Proc. 11th Int'l Conf. Data Engineering, pp. 3-14, 1995.

[3] C. Bettini, S.X. Wang, S. Jagodia, and J.-L. Lin. Discovering Frequent Event Patterns with Multiple Granularities in Time Sequences, IEEE Transactions on Knowledge and Data Engineering, vol. 10, no. 2, pp. 222-237,1998.

[4] T. Fawcett and F. Provost. Activity Monitoring: Noticing Interesting Changes in Behavior, Proc. Fifth Int'l Conf. Knowledge Discovery and Data Mining, S., pp. 53-62, 1999.

[5] Gaber, M, M., Krishnaswamy, S., and Zaslavsky, A., Adaptive Mining Techniques for Data Streams Using Algorithm Output Granularity, The Australasian Data Mining Workshop (AusDM 2003), Canberra, Australia.

[6] S. Helal, W. Mann..The Gator Tech Smart House: A programmable pervasive space. IEEE Computer, 38(3):50-60, 2005.

[7] S. Laxman and P.S. Sastry. A survey of temporal data mining, Sadhana Vol. 31, Part 2, pp. 173–198, 2006.

[8] C.H. Lee, M.-S. Chen, and C.-R. Lin. Progressive pattern miner: An efficient algorithm for mining general temporal association rules. IEEE Transactions on Knowledge and Data Engineering 15: 1004–1017, 2003.

[9] W Lin, SA Alvarez, C Ruiz - Efficient Adaptive-Support Association Rule Mining for Recommender Systems. Data Mining and Knowledge Discovery, 2002 – Springer.

[10] H. Mannila and H. Toivonen. Discovering Generalised Episodes Using Minimal Occurences, Proc. Second Int'l Conf. Knowledge Discovery and Data Mining, pp. 146-151, 1996.

[11] Rissanen, J. Modeling by shortest data description. Automatica, 14:465--471, 1978.

[12] John F. Roddick, Myra Spiliopoulou. A Survey of Temporal Knowledge Discovery Paradigms and Methods, IEEE Transactions on Knowledge and Data Engineering, vol. 14, No. 4, 2002.

[13] Sutton, R.S. On the significance of Markov decision processes . Artificial Neural Networks -- ICANN'97, pp. 273-282. Springer.

[14] S.V. Vaseghi. State duration modeling in hidden Markov models. Signal Processing, 41(1):31-41, 1995.

[15] V Vainio, A.-M., Vanhala, J. Continuous-time Fuzzy Control and Learning Methods. ISCIT 2007. October2007. Sydney, Australia.

[16] G. M. Youngblood and D. J. Cook. Data mining for hierarchical model creation. IEEE Transactions on Systems, Man, and Cybernetics, Part C, 2007.

Dense Pixel Visualization for Mobile Sensor Data Mining

Pedro Pereira Rodrigues
LIAAD - INESC Porto L.A.
Faculty of Sciences - University of Porto
Rua de Ceuta, 118 - 6 andar
4050-190 Porto - Portugal
pprodrigues@fc.up.pt

João Gama
LIAAD - INESC Porto L.A.
Faculty of Economics - University of Porto
Rua de Ceuta, 118 - 6 andar
4050-190 Porto - Portugal
jgama@fep.up.pt

ABSTRACT

Sensor data is usually represented by streaming time series. Current state-of-the-art systems for visualization include line plots and three-dimensional representations, which most of the time require screen resolutions that are not available in small transient mobile devices. Moreover, when data presents cyclic behaviors, such as in the electricity domain, predictive models may tend to give higher errors in certain recurrent points of time, but the human-eye is not trained to notice this cycles in a long stream. In these contexts, information is usually hard to extract from visualization. New visualization techniques may help to detect recurrent faulty predictions. In this paper we inspect visualization techniques in the scope of a real-world sensor network, quickly dwelling into future trends in visualization in transient mobile devices. We propose a simple dense pixel display visualization system, exploiting the benefits that it may represent on detecting and correcting recurrent faulty predictions. A case study is also presented, where a simple corrective strategy is studied in the context of global electrical load demand, exemplifying the utility of the new visualization method when compared with automatic detection of recurrent errors.

Keywords

dense pixel visualization, mobile devices, sensor data

1. INTRODUCTION

Sensor data is usually represented by streaming time series, which are sometimes hard to interpret, especially if visualization is produced in low-resolution screens usually embedded in small mobile devices. When applying data mining methods to extract knowledge or to learn a given concept from the sensor data series, visualizing the data mining results is also an important feature for human experts. Human expert knowledge is most of the time difficult to include in learning processes. Moreover, if data is not easily visualized, the experts may have even more difficulties in analyzing the complete process.

Another key issue is the fact that these series tend to represent usual physical phenomena, or human interaction with the environment, or even human behavior, which are most of the times cyclic in some extent. Often, when data presents cyclic behaviors, predictive models learned for these series may also tend to give higher errors in certain recurrent points of time. However, the human-eye is not trained to notice these cycles in a long stream, yielding the need for new visualization techniques.

The approach we present here is to give the human-expert the possibility of visualizing the sensor data and the absolute errors of a given predictive process for a recent past period, enabling him with tools to detect recurrent periods of higher errors. Nowadays applications are being requested to answer queries in low-resolution mobile devices [12,13]. A simplified dense pixel approach is used in order to cope with mobile resources restrictions. After the expert detects the period of time (hours of the day) where higher errors are recurrent, a simple technique can be used to improve the prediction in that periods in the forthcoming data, or decide on more sophisticated mining techniques for those periods.

This paper is organized as follows. Next section presents an overview on visual data mining and how our contribution relates with its different dimensions. Section 3 motivates the reader to sensor data produced in human-related environments, presenting the example domain of electricity demand. Section 4 presents our main contribution, a simple dense pixel display which helps the human-expert on identifying recurrent errors in long-term time series, even from low-resolution mobile devices. In section 5, a simple case-study is presented, where our visualization method is used to detected recurrent errors, comparing it with automatic procedures in the automatic correction of following predictions. Section 6 concludes the paper with future directions.

2. VISUAL DATA MINING

For data mining to be effective, it is important to include the human in the data exploration process and combine the flexibility, creativity, and general knowledge of the human with the enormous storage capacity and the computational power of today's computers [15]. Visualization techniques enable the user to overcome major problems of automatic machine learning methods, such as presentation and interpretation of results, lack of acceptance of the discovered findings, or

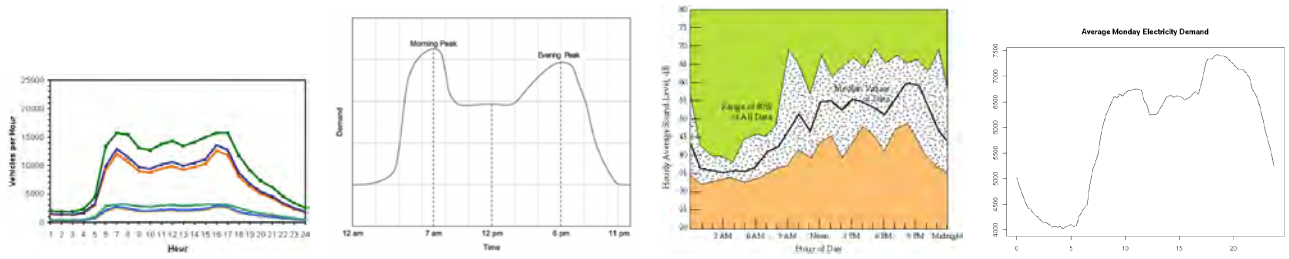


Figure 1: Real-world examples of how human behavior directly influences sensor data time patterns: highway traffic volume [3], home water demand [18], indoor residential sound level [1] and electricity demand [6]. Not only the evolution is cyclic but also highly correlated among different daily-life dimensions.

limited confidence in these [2]. Moreover, they represent a major improvement in learning systems as they can combine human expert knowledge, human perception and human reasoning, with computational power, computational data mining and machine learning.

Visual data mining is the task of discovering implicit, complex and massive but useful knowledge from large data sets, using data and/or knowledge visualization techniques based on computer graphics [8]. The main motivation to use visual data mining techniques is to take advantage of the human highly parallel processor, which combines a sophisticated reasoning machine with a huge knowledge base, even when there is no 2D or 3D semantics in the data set. However, it lacks automation and induces intrinsic bias in the process. According to Keim [15], visual data mining techniques can be classified on three orthogonal dimensions: data type to be visualized, vizualization technique and interaction/distortion technique. Please see [15]for a classified overview of visualization techniques.

2.1 Data Type to be Visualized

Data types differ mostly on the way they are structured: data with light structures, being one-dimensional (usually temporal, e.g. sensor data), two-dimensional (e.g. geographical), or multi-dimensional (e.g. combined sensor network data); and data with heavy structures, such as text and hypertext, hierarchies and graphs, or even algorithms [15].

With respect to the problem at hands in this paper, we are addressing one-dimensional data produced by one sensor over time. Further work will focus on the integration of visual data mining for multi-dimensional data gathered by the whole sensor network.

2.2 Visualization Technique

Visualization techniques have been widely proposed for different domains, with lots of highly sophisticated methods. From the standard 2D/3D displays (e.g. bar charts and line plots) to geometrically transformed displays (e.g. landscapes), with high emphasis on dense pixel (e.g. recursive pattern) and stacked (e.g. treemaps) displays [15].

This is the main area of action of our work. We believe that, for mobile sensor data analysis, systems should evolve from the current 2D/3D standards into a simplified version of dense pixel displays [14]. The following sections present our proposal.

2.3 Interaction and Distortion Technique

These techniques empower the users with the ability to interact with the visualization, changing visual parameters according to current needs. They include projection, filtering, zooming, distortion and linking [15].

Our contribution in this dimension is thin. Nevertheless, since we wander along dense pixel approaches, we foresee that future systems will enable interactive zooming and filtering as part of the graphical user interface, to allow the user with different insights on the results.

3. SENSOR DATA TIME PATTERNS

Sensor data is usually produced at high rate, in a stream. A data stream is an ordered sequence of instances that can be read only once or a small number of times using limited computing and storage capabilities [5]. These sources of data are characterized by being open-ended, flowing at high-speed, and generated by non stationary distributions. However, in several human-related domains, sensor data tend to follow certain time patterns, according to cultural habits and society organization. Figure 1 presents some examples of domains where this happens. In city surroundings, highway traffic volume presents usual time patterns [3]. Given its impact in people location, this is also related with home demand for resources, such as water [18] and electricity [6],and the time pattern of hourly indoor residential sound levels [1]. In these domains, not only the evolution of sensed data is cyclic, but also highly correlated among different daily-life dimensions. In the following, we will focus on a precise scenario, where sensors produce data of electrical load demand.

3.1 Electricity Demand Sensor Data

One example of a sensor network where time series data streams are of major importance to human experts is the electricity distribution network. Electricity distribution companies usually set their management operators on SCADA/ DMS products (Supervisory Control and Data Acquisition / Distribution Management Systems). One of their important tasks is to forecast the electrical load (electricity demand) for a given sub-network of consumers. Load forecast is a relevant auxiliary tool for operational management of an electricity distribution network, since it enables the identification of critical points in load evolution, allowing necessary corrections within available time. In SCADA/DMS systems, the load forecast functionality has to estimate, on a hourly basis, and for a near future, certain types of measures which are representative of system's load: active power, reactive

power and current intensity [6]. Given its practical application and strong financial implications, electricity load forecast has been targeted by innumerable works, mainly relying on the non-linearity and generalizing capacities of neural networks, which combine a cyclic factor and an autoregressive one to achieve good results [10].

3.2 Interpretation of Data Mining Results

In the context of forecasting it is relevant to present results in a meaningful way. Results show themselves to be of high importance to criticize the forecasting method. In this way, a good visualization technique should present the forecasting errors as clearly as possible, presenting relevant information. Usually, the closest period one needs to predict is the next hour load. This way, experts may inspect errors for previous predictions in a long period in the past, trying to find recurrent errors and improve next predictions. If a good predictive model is used for electrical load forecast, the error is expected to have a normal distribution [17]. The usual error measure used in electricity domains is the *Absolute Percentage Error* (APE) for which the mean is usually computed: $MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{y_i}$. The advantage of this measure is its easy interpretation as a percentage of the real value, which is clearer to understand than, for example, the *Mean Squared Error*. By analyzing the error, one can have some type of measure on the quality of the learned model.

4. VISUALIZATION OF SENSOR DATA

The main purpose of data visualization is to gain insight about the information space, mapping data onto graphical features, providing qualitative overview of large data sets, while searching for patterns or trends, or other artifacts in data. Data in a database or data warehouse can be viewed in various visual forms at different levels of abstraction, with different combinations of attributes or dimensions [8].

A usual way of exploring data is to extract the main dimensions where it is observed. Another way is to extend its dimensions. For example, n-dimensional data can be reduced into k principal components [11] or one specific dimension may be separated into different components. This is especially useful if high-dimensional data needs to be projected in two or three dimensions for visualization.

Usually, in sensor operation systems, standard 2D line plots such as the one presented in Figure 2 are used to display the values of electrical load along some time interval. It is simple to observe how the forecast values are distributed along time by looking at the same image, but a more valuable information would be to inspect the error of the corresponding forecast. By analyzing the error, one can have some type of measure on the quality of the learned model. However, forecast error, in the bottom red line, is hard to be correctly analyzed in such a graphic.

A special feature of these series is that the standard profile for a week load demand is well defined [7]: five similar week days, followed by two weekend days. One way to find usual flaws of the forecasting process would be to analyze days side by side instead of a continuous representation. Prediction errors that could be recurrent to some part of the day could be identified more easily this way. Current state-of-the-art

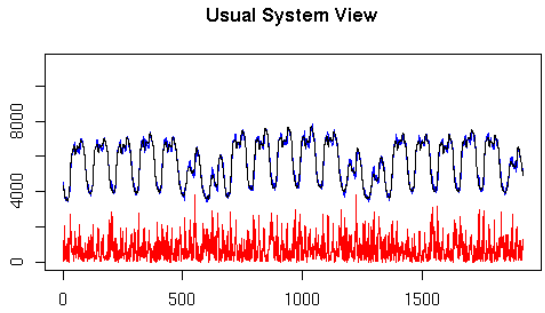


Figure 2: Usually, the forecast system presents the electrical load value and corresponding prediction (top). The absolute percentage error is presented in the bottom. For this plot, high-level resolutions are needed to observe real data with reduced loss.

in visualization of electrical load is focused on interactive three-dimensional views [16]. 3D graphics such as the ones presented in Figure 3 would increase the ability of the human expert to find regularities in both the data and the predictive error. It usually groups the days side by side, showing: the time of the day in the xx axis, the date of the day in the yy axis, and the value of the measured variable in the zz axis.

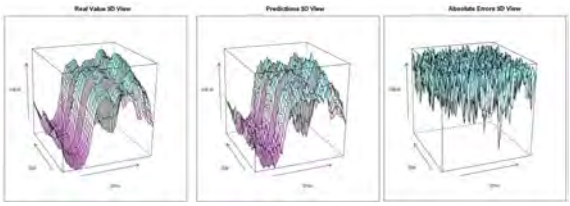


Figure 3: The electrical load is shown in a daily fashion in three dimensions. Resolution requirements are nearly out-of-reach for transient mobile devices.

These three-dimensional representations can be used to find regularities but might be less clear to the human eye. Graphics are two-dimensional and a three-dimensional representation has to be flatten in some perspective, to be represented in a two-dimensional way, covering up some parts of the visualization. This may, in fact, lead to some parts of the image being impossible to see. Moreover, 3D representations usually imply higher resolution requirements than 2D plots, which limit their use in small mobile devices.

4.1 Simple Dense Pixel Display

The method we propose here is to use a simpler 2D approach in a representation similar to a three-dimensional graphic but in a two-dimensional platform, creating a dense pixel display. The basic idea of *dense pixel* techniques is to map each dimension value to a colored pixel and group the pixels belonging to each dimension into adjacent areas [14]. This technique is especially relevant for time series, where data is temporally connected, usually creating multi-dimensional visualizations.

We create a simple dense pixel display, considering only one dimension, where the values of the measured variable are substituted by a color within a certain gradient. For sim-

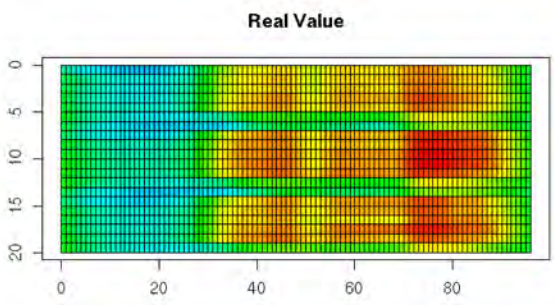


Figure 4: Simple Dense Pixel Display: one line per day. In this example one value is shown each 15 minutes, producing a 96×20 pixmap, clearly within current mobile screen standards.

plicity of this example the rainbow gradient is used where the warmest colors, longer wavelengths such as red, represent higher values and cooler colors, shorter wavelengths such as blue, represent lower values. Any spectrum can be used considering the expert's decision or taste on colors and spectres.

The arrangement of data points is directly related with the temporal dimension, as rows represent days. Horizontally-adjacent pixels are consecutive points, while vertically-adjacent pixels are values for sensor readings at same time in consecutive days. Figure 4 presents our simple dense pixel display for same data plotted in Figure 2 and left plot of Figure 3.

4.2 Recurrent Error Detection

One motivation for visualizing data mining results is to help human users to find interesting regions and suitable parameters for further quantitative analysis. Also, this process provides visual proof of the computer representations derived from the mining process.

Current load forecast rates are small. Usually, the closest period one needs to predict is the next hour load. This way, experts may inspect errors for previous predictions in a long period in the past, trying to find recurrent errors in order to improve next predictions. An automatic procedure could detect error peaks by evaluating extreme values in the distribution of errors per period of the day, using statistical techniques for outlier detection [9]. From preliminary analysis, this technique can, in fact, detect some errors. If the errors occur mostly at certain times of the day, a pattern can be found and a correction can therefore be induced at that point. Human-based approaches, however, can represent higher improvements as expert knowledge can be used to infer the recurrence of the errors.

We propose to use our plot in the analysis of recurrent errors. As seen on right plot of Figure 5, errors become easy to identify: as days are on horizontal lines, if a forecast method has a larger error at some given time, during a period, *vertical areas of warmer colors* become easy to spot. This translates to a *recurrent fail* on the predictive system, which one might correct if it occurs always in specific conditions. What is found to be especially useful in this approach is how simple it may become in certain cases to find erroneous patterns, specific periods of the day where the prediction error is higher than usual. As a showcase purpose, it is easy to show if a system is mostly reliable along the day or if and where it has drawbacks. Of course, this is valid for experts' analysis but also to non-experts'. With a simple observation of the color draw, time of day and day of week, the good and bad characteristics of a system become clear.

Overall, field-working experts could benefit from these visualization techniques in their own mobile devices in order to inspect the behavior of sensor data. Perhaps the best way to point out the errors would be to find the periods of the day, and the days, where the errors tend to be higher. This could be the most appropriate information to use when analyzing the results of a predictive model while we try to change the model to improve its predictions in the referred periods.

4.3 Applying Specific Corrections

Detecting the errors in a predictive model can be done automatically or by humans. In the same way, correcting these errors by adjusting the predictor may be performed using these two type of procedures. Mostly, one would apply, to a certain time of the day, some corrective factor in the prediction. To achieve this, we would consider knowing beforehand, by using automatic or human detection, where and how does the prediction present a larger error. In the data analyzed here, in just two weeks some patterns of high errors can be found, so the correction could get some good results.

If an automatic method is unable to find a specific corrective factor to apply to a prediction, a human expert might be able to. Using specific graphics, and after finding the patterns in the errors' series, an acceptable and somehow reliable adjustment could be applied to the forecasted values. This adjustment would benefit from the expert's knowledge on the specific problem. If the spectrum of the colors in the errors' graphic follows the error distribution, when the graphics' color becomes more and more coherent the errors' peaks can be considered annulled. Still, the connection between the correction and the detection of the error becomes strong. The purpose of this work now becomes clear, because with a better error detection, corrections can get more reliable and faster to perform even by human experts.

5. CORRECTING RECURRENT ERRORS

The main contribution of this work is to propose a simple visualization strategy which helps human-experts in the analysis of sensor data and in the detection of recurrent errors in predictive processes. Nevertheless, in this section we present a case-study where we apply simple automatic and human-based techniques for recurrent error detection and correction, in the context of global country electricity load demand. The idea is to give the human-expert the possibility of visualizing the absolute errors for a recent past period, enabling him with tools to detect recurrent periods of higher errors. After the expert has defined the period of time (hours of the day) where higher errors are recurrent, a simple technique can be used to improve the prediction in that periods in the forthcoming data.

5.1 System Description

For this initial testing, we are trying to predict the next-hour load demand, using a feed-forward neural network with

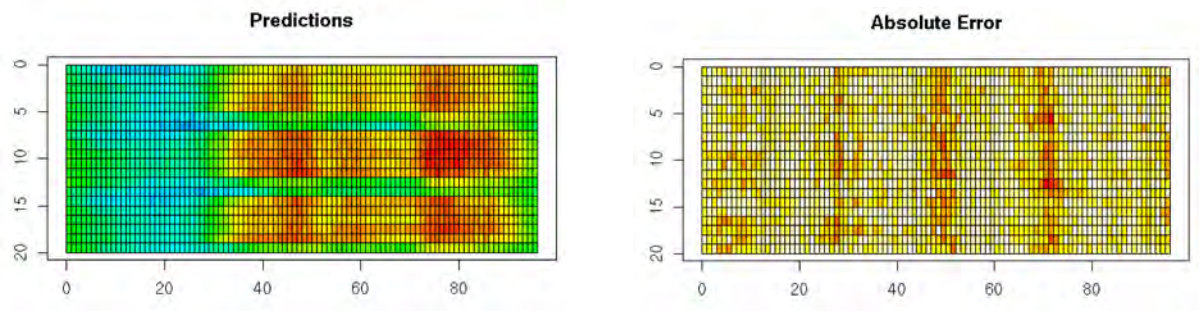


Figure 5: The predicted electrical load can also be shown in the same periodical fashion. More interesting is the visualization of the absolute errors. If a forecast method has a larger error at recurrent points in time, during a period, *vertical areas of warmer colors* become easy to spot. In this example, a human-expert can easily detect at least three recurrent periods where the system is producing higher errors.

10 inputs, 4 hidden neurons (*tanh*-activated) and a linear output. The input vector for predicting time t is t minus $\{1, 2, 3, 4\}$ hours and t minus $\{7, 14\}$ days. As usual [6], we consider also 4 cyclic variables, for hourly and weekly periods (*sin* and *cos*). The choice of the networks topology and inputs was mainly motivated by experts suggestion.

To perform a first experience on the applicability of the method to real-world scenarios, we use a global series with the entire load demand of the country, with four values per hour.

5.2 Detection Mechanisms

First two months are used to learn the model until it has satisfyingly converged. Afterwards, we use three weeks (Figure 5) to detect erroneous periods and apply the corrections in the following week. We consider three different types of detection:

Automatic Periods are defined by looking at the distribution of average errors, spotting those presenting average APE above percentiles 95, 75 and 50.

Human Periods are chosen by a human expert, after seeing the errors in the three weeks. This resulted in four periods: first three hours of the day, 6:00 to 9:00, 12:00 to 13:00 and 16:00 to 19:00.

None All points are subject to corrective measures.

5.3 Corrective Mechanism

Our target function is a continuous and derivable function over time. For these type of time series, one simple prediction strategy, reported elsewhere to work well, consists of predicting for time t the value observed at time $t - k$ [6]. Previous work with load demand data revealed that one of the most simple, yet effective, strategies is to use the corresponding value one week before ($k = 168$ hours) as a *naïve* estimate to combine with our estimate.

In this approach we do not consider error evolution or estimates reliability. A simple strategy, where we don't use any prior knowledge on the estimators, is to simply use the average of the two estimators. This way, the prediction for a time point within a spotted period is the average between the value predicted by the model (neural network) and the corresponding real value one week before. Formally, if y is the time series, producing l points per hour, and for time t, $\dot{y}_t$ is the model's prediction, the combined prediction is

$$\hat{y}_t = \lambda\, \dot{y}_t + (1 - \lambda)\, y_{t-k} \tag{1}$$

where $k = 168l$ and $\lambda \in [0, 1]$ is the associated weight for the estimates (the simplest strategy would have $\lambda = 0.5$).

5.4 Comparison of Different Strategies

For each detection strategy we vary λ from a set L of values, reproducing the idea of using only the *naïve* estimate ($\lambda = 0$), only the model estimate ($\lambda = 1$), or weighted combinations of these two estimates. This way, we try to assess if the definition of problematic regions by the human-expert is valuable to the forecast system. We compare the different strategies using the *Mean Squared Error*, given by:

$$MSE = \frac{1}{n|L|} \sum_{i=1}^{n} \sum_{j \in L} (y_i - \hat{y}_{ij})^2 \tag{2}$$

where n is the size of the tested data, and $\hat{y}_{ij}$ is the prediction of the model with $\lambda = j$. We can also compute the MSE for a single run with fixed λ.

Left plot of Figure 6 shows MSE results with respect to λ, for all approaches. The right-most point in the plot ($\lambda = 1$) uses only the model, while left-most points ($\lambda = 0$) consider complete *naïve* predictions in the chosen periods (depending on the strategy). First, we should see if our model is predicting better than the *naïve*. The reader can easily note that the error is higher for the *naïve* approach applied to all predictions. The fact that all plots are concave reflect the idea that the combination of the two estimates represents an improvement in the predictions. We should, however, inspect how this combination should be done. As we increase the weight of the *naïve* estimate (decreasing λ) we see that applying the combination to every prediction seems to outperform the advantage of expert-defined periods. Interestingly, when more weight is given to this *naïve* estimate than to our model's estimate, we clearly notice that we should only consider combinations in problematic periods. Automatic detection techniques failed to consistently lower the

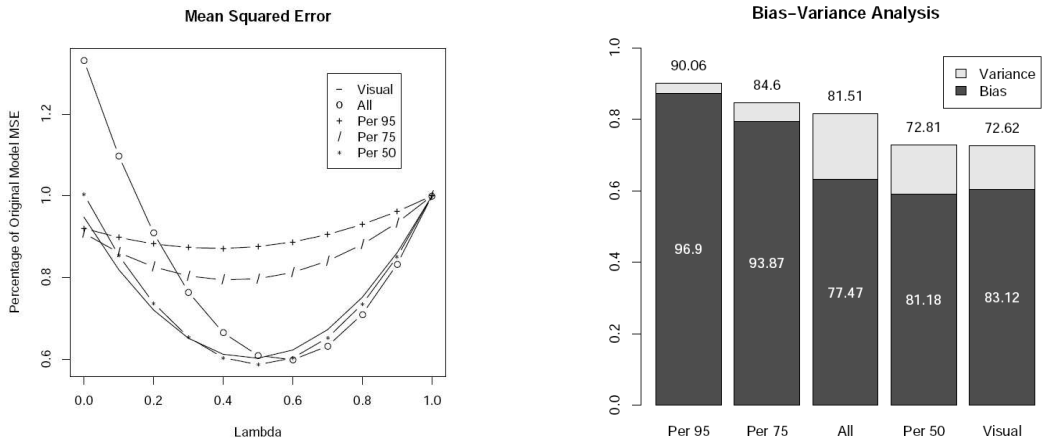


Figure 6: The plots present the MSE in percentage of the original model's MSE. Left plot presents values with respect to λ, for all approaches: straight lines represent the application of the combined predictions to chosen periods; circle-dashed lines represent the combination in all predictions; remaining lines represent the application of corrections to points above percentile 95 ("+"), 75 ("/") and 50 ("*"). The plot on the right presents the *Bias-Variance* decomposition of the five strategies. Top values represent the MSE across all values of λ, while white values within the barplot represent the percentage of the error explained by the *Bias* component.

MSE, except for the percentile 50 strategy, which nevertheless implies corrections in half of the predictions.

5.5 Sensitivity to λ

Although the λ parameter can be defined differently for each application (and even automatically defined by online strategies) we also inspect how the different strategies are sensitive to changes in this parameter. We can inspect whether the gain in the predictive MSE is due to *bias* or *variance* reduction. We can study bias and variance in exact forms as $MSE = Bias^2 + Variance$, with

$$Bias^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y}_i)^2 \quad (3)$$

and

$$Variance = \frac{1}{n|L|}\sum_{i=1}^{n}\sum_{j\in L}(\bar{y}_i - \hat{y}_{ij})^2 \quad (4)$$

where $\bar{y}_i = \sum_{j\in L}\hat{y}_{ij}/|L|$ is the average model calculated as the average of all predictions (with different λ) for input data point i.

Right plot of Figure 6 plots the variation of the error for different strategies. Low variance actually means that the strategy is less sensitive to λ. It becomes clearer that automatic detection techniques which take outliers into account (percentiles 95 and 75), present slight (reduction below 20%) but robust improvements as they are less sensitive to λ. However, they actually failed to consistently reduce the global error. The simple strategy of correcting all predictions presented slightly higher reduction but at the cost of high variance with respect to λ. Visual detection achieved the best averaged result, which is nonetheless similar to the automatic correction applied to half the predictions. However, visual detection presented slightly less sensitivity to λ, being therefore more robust to include in graphical user interfaces where the λ parameter can be arbitrarily chosen. Figure 7 presents the original and enhanced errors for the studied week with $\lambda = 0.5$.

Right plot of Figure 6 plots the variation of the error for different strategies. Low *variance* actually means that the strategy is less sensitive to λ. It becomes clearer that automatic detection techniques which take outliers into account (percentiles 95 and 75), present slight (reduction below 20%) but robust improvements as they are less sensitive to λ. However, they actually failed to consistently reduce the global error. The simple strategy of correcting all predictions presented slightly higher reduction but at the cost of high *variance* with respect to λ. Visual detection achieved the best averaged result, which is nonetheless similar to the automatic correction applied to half the predictions. However, visual detection presented slightly less sensitivity to λ, being therefore more robust to include in graphical user interfaces where the λ parameter can be arbitrarily chosen. Figure 7 presents the original and enhanced errors for the studied week with $\lambda = 0.5$. While strategies using percentiles 95 and 75 did not deliver in reducing the recurrent high-error regions, correcting all predictions created a smoother image, with no clear high-error regions but with greater overall error. Visually-detected and percentile 50 clearly outperformed the remaining strategies, but extra care should be taken when comparing these two, given the sensitivity to λ they express. Further experiments should concentrate on this result to clearly explain connections between visual detection and percentile 50.

5.6 Discussion on Study Results

We have compared the human-based visualization detection with automatic detection of recurrent errors. The comparison is done across different but simple corrective strategies. We do not seek to present this techniques as a breakthrough corrective process, rather a simple example of how visual de-

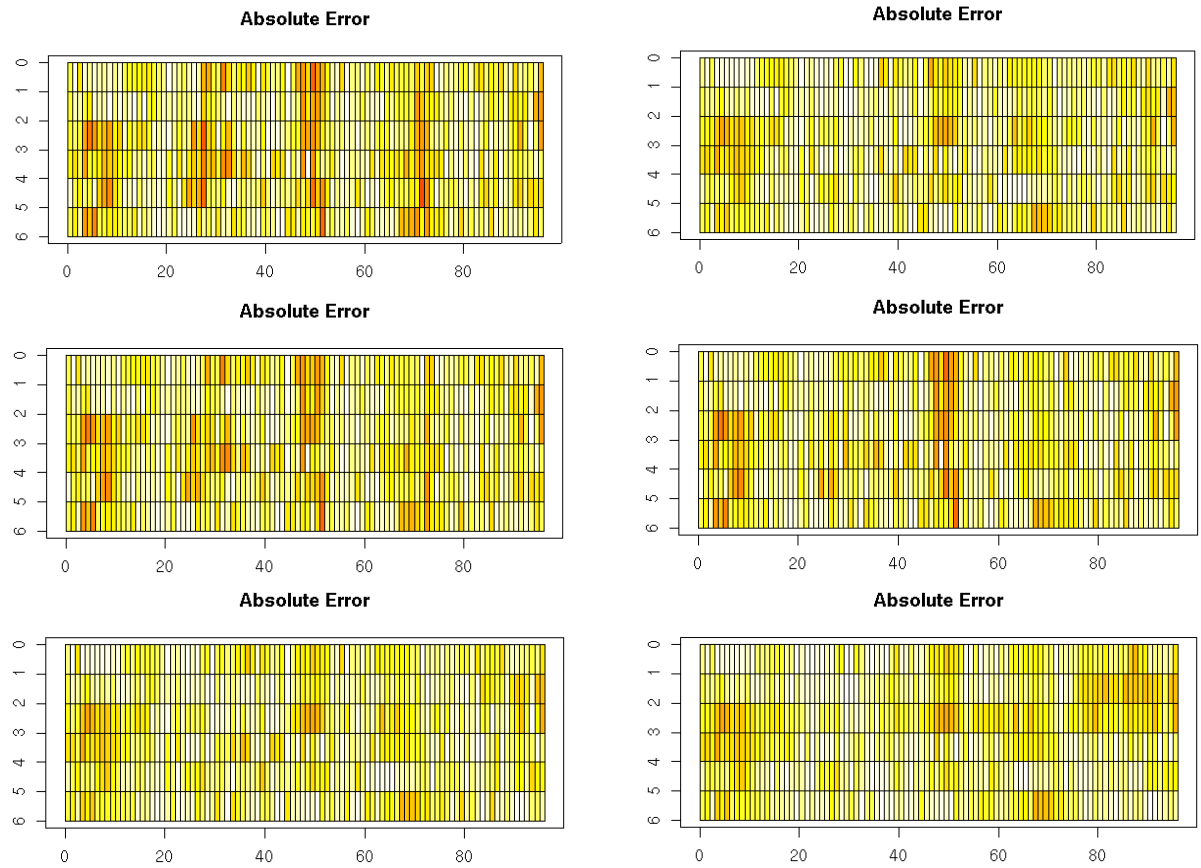


Figure 7: Visualization of the absolute errors on test week ($\lambda = 0.5$). Top-left plot presents errors for the original model ($MSE = 100\%$) while top-right plot presents errors with *visually* detected periods ($MSE = 60.24\%$). Following plots present errors for automatic detection techniques: *Per 95* ($MSE = 87.58\%$), *Per 75* ($MSE = 79.71\%$), *Per 50* ($MSE = 58.67\%$) and *All* predictions ($MSE = 60.93\%$).

tection may improve field-workers perception and support sensors' predictive models. We show that, in this study, not only the corrective measure should be applied only to visually detected periods, but also that visual detection is less sensitive to parameter tuning of the corrective mechanism than automatic detection. The fact that results of visually-spotted regions are quite similar to those of using percentile 50 could be related with the fact that human brain is trained to notice unusual patterns, hence spotting errors above the average, but this discussion should be taken only after further tests and experiments with different data and users. Overall, the major advantage of visual detection techniques is that expert knowledge is introduced in the system as complementary information, creating possibly more representative models.

6. CONCLUDING REMARKS

A human society in a given country tends to behave in a constant way and thus to consume resources and live according to similar profiles. By using visualization techniques better analysis can be performed, allowing better corrections to usual predictive errors which we found to happen in real data. In such financially relevant problems, such as the electrical consumption, every step towards more reliable predictions becomes a valuable asset. In this paper we have proposed a simple dense pixel visualization technique for sensor data, focusing on the application domain of electrical load demand, which enhances the detection of recurrent periods where the prediction error is higher.

What is found to be especially useful in this approach is how simple it may become in certain cases to find erroneous patterns, specific periods of the day where the prediction error is higher than usual. Of course, this is valid for experts' analysis but also to non-experts'. Nevertheless, these plots were considered by electrical load forecast experts as a major breakthrough, since a stacked view on load forecast enhances the ability to inspect longer periods of data and recurrent errors become easier to spot.

Moreover, expert knowledge is introduced in the system as complementary information, creating possibly better representative models. The extension of this view to the entire sensor network is still a future issue, and it could benefit from ambient intelligence techniques [4] since this research area already considers colours as an important mean of communication with human users. Parallel to these human-based techniques, the quality of predictive processes can also be enhanced by deeper sensor network comprehension tech-

niques such as sensor data clustering.

Future directions include the exhaustive validation of such a recurrent error detection technique, integrated in a graphical user interface of the electrical network management system. That way, experts can directly operate on the forecast process, defining recurrent periods of erroneous predictions, possibly defining the set of corrective techniques they are willing to apply to their predictions.

Acknowledgments

The work of Pedro P. Rodrigues is supported by the Portuguese Foundation for Science and Technology (FCT) under the PhD Grant SFRH/BD/29219/2006. The authors thank FCT's Plurianual financial support attributed to LIAAD and the participation of Project ALES II under Contract POSC/EIA/55340/2004. Pedro P. Rodrigues is also partially supported by FCT through Project CALLAS under Contract PTDC/EIA/71462/2006. Additionally, the authors also wish to thank Tiago Caxias for his help on a initial version of this work.

7. REFERENCES

[1] Environmental Protection Agency. Information on levels of environmental noise requisite to protect public health and welfare with an adequate margin of safety, March 1974. EPA/ONAC 550/9-74-004.

[2] Jesús S. Aguilar-Ruiz and Francisco J. Ferrer-Troyano. Visual data mining. *The Journal of Universal Computer Science*, 11(11):1749–1751, 2005.

[3] Michael Claggett and Terry L. Miller. A methodology for evaluating mobile source air toxic emissions among transportation project alternatives. In *Air Quality 2006*, pages 32–41, 2006.

[4] José L. Encarnação. Ambient intelligence - the new paradigm for computer science and for information technology. *it - Information Technology*, 50(1):5–6, 2008.

[5] João Gama and Pedro Pereira Rodrigues. Data stream processing. In *Learning from Data Streams - Processing Techniques in Sensor Networks*, chapter 3, pages 25–39. Springer Verlag, 2007.

[6] João Gama and Pedro Pereira Rodrigues. Stream-based electricity load forecast. In *Procs of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2007)*, LNAI 4702, pages 446–453, 2007. Springer Verlag.

[7] D. Gerbec, S. Gasperic, I. Smon, and F. Gubina. An approach to customers daily load profile determination. In *Power Engineering Society Summer Meeting*, pages 587–591. IEEE Computer Society, 2002.

[8] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2001.

[9] D. M. Hawkins. *Identification of Outliers*. Chapman and Hall, 1980.

[10] H. S. Hippert, C. E. Pedreira, and R. C. Souza. Neural networks for short-term load forecasting: a review and evaluation. *IEEE Transactions on Power Systems*, 16(1):44–55, 2001.

[11] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24, 1933.

[12] Hillol Kargupta, Ruchita Bhargava, Kun Liu, Michael Powers, Patrick Blair, Samuel Bushra, James Dull, Kakali Sarkar, Martin Klein, Mitesh Vasa, and David Handy. VEDAS: A mobile and distributed data stream mining system for real-time vehicle monitoring. In *Proceedings of the Fourth SIAM International Conference on Data Mining*, 2004.

[13] Hillol Kargupta, Byung-Hoon Park, Sweta Pittie, Lei Liu, Deepali Kushraj, and Kakali Sarkar. MobiMine: Monitoring the stock market from a PDA. *SIGKDD Explorations*, 3(2):37–46, 2002.

[14] Daniel A. Keim. Designing pixel-oriented visualization techniques: Theory and applications. *IEEE Transactions on Visualization and Computer Graphics*, 6(1), January-March 2000.

[15] Daniel A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 7(1), January-March 2002.

[16] T. J. Overbye and J. D. Weber. New methods for the visualization of electric power system information. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 131–16c, 2000.

[17] Pedro Rodrigues and João Gama. Forecast adaptation to charge transfers. *WSEAS Transactions on Circuits and Systems*, 3(8):1693–1699, October 2004.

[18] Thomas M. Walski, Donald V. Chase, Dragan Savic, Walter M. Grayman, Stephen Beckwith, and Edmundo Koelle. *Advanced Water Distribution Modeling and Management*. Haestead Press, 2003.

Anomaly Detection from Sensor Data for Real-time Decisions

Olufemi A. Omitaomu
Oak Ridge National Laboratory
1 Bethel Valley Road
Oak Ridge, TN 37831, USA
+1 (865) 576-7597
omitaomuoa@ornl.gov

Yi Fang*
Department of Computer Science
Purdue University
West Lafayette, IN 47907
yfang@cs.purdue.edu

Auroop R. Ganguly**
Oak Ridge National Laboratory
1 Bethel Valley Road
Oak Ridge, TN 37831, USA
+1 (865) 241-1305
gangulyar@ornl.gov

ABSTRACT
The detection of unusual profiles or anomalous behavioral characteristics from multiple types of sensor data is especially complicated in security applications where the threat indicators may or may not be known in advance. Predictive modeling of massive volumes of historical data can yield insights on usual or baseline profiles, which in turn can be utilized to isolate unusual profiles when new data are observed in real-time. Thus, a two-stage knowledge discovery process is proposed, where offline approaches are utilized to design online solutions that can support real-time decisions. The profiles of the unusual, delineated in real-time, are held aside for secondary inspections, where domain knowledge may characterize these as non-threat conditions. The domain knowledge developed incrementally in this fashion, as well as embedded within the anomaly detection profile themselves at the start, can contribute to enhancing the anomaly detection process, which in turn can reduce false alarms and missed detections in real-time. The approach is illustrated in the context of ensuring safety and security of commercial trucks on the US interstate system through sensor-based anomaly detection at truck weigh stations. First, the overall problem is described in detail with an emphasis on the need to assimilate heterogeneous sensor data for risk-informed decisions in real-time. Second, a focused case study is presented with static scale data, based on prior and ongoing work in the area of situational awareness and threat detection for real-time decision support.

Categories and Subject Descriptors
I.5.2 [**Pattern Recognition**]: Design Methodology – *classifier design and evaluation, pattern analysis.*

Keywords
Radioactive materials, transportation security, anomaly detection.

*This author contributed to this work while he was a post-Master student at the Oak Ridge National Laboratory.
** **CORRESPONDING AUTHOR**

Sensor-KDD '08, August 24, 2008, Las Vegas, NV, USA.

1. INTRODUCTION
The effective detection of unusual profiles or anomalous behavioral characteristics from multiple sensor data for real-time decisions is a problem in several security-related applications including truck weigh stations. At truck weigh stations, the ability to effectively detect commercial trucks that are transporting illicit radioactive materials using radiological and nuclear (RN) data is a challenge. The current strategy uses radiation portal monitor (RPM) to analyze one type of RN data called the gross count (radiation) data during the primary inspection; the gross count data measures the total radiation counts in truck cargoes. When a truck is identified as a possible security risk during this inspection, the inspection officer requests a secondary inspection and possibly a tertiary inspection. The additional inspection procedures include collecting some supplementary data such as spectroscopy data for further analyses, analyzing the cargo manifest, collecting driver's information, and possibly conducting manual inspection. These additional procedures cause truck delays and increase the operating costs of the weigh stations. A flow chart representation of the inspection procedures are depicted in lower half of Figure 1.

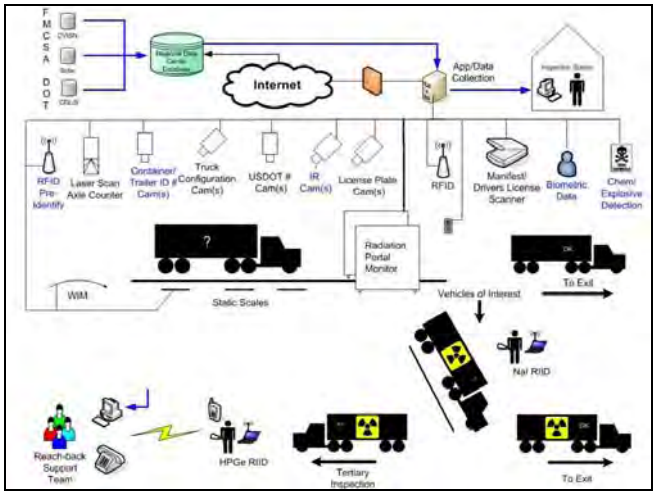


Figure 1. Truck inspection flowchart at weigh station test-beds

Generally, the initial alarm is a false alarm and this has been attributed to several factors. Firstly, there are several legitimate cargoes, such as kitty-litter and fertilizer in commerce and medical isotopes in hospitals that contain naturally occurring radioactive material (NORM) or technologically-enhanced naturally occurring radioactive material (TENORM) respectively. These cargoes trigger alarms during the primary inspection, and constitute false alarms, even though they are generally not the target [1]. Secondly, the RPM uses plastic scintillator material (PVT), which makes it applicable to only the gross count data [2]. However, the use of only the gross count data may be insufficient to determine whether or not a vehicle is a security risk. Other problems associated with the current strategy have been discussed and investigated; for example, see [1] – [5]. Due to the high number of false alarms during the primary inspection, the essence of the secondary and tertiary inspections becomes unclear under the current strategy because they have become the common events at weigh stations.

One approach to improve on the current strategy is to use multiple sensor data such as static scale, gross count, and spectroscopy data during the primary inspection; this improvement could significantly reduce the number of false alarm during the procedure and eliminate (or drastically reduce) unnecessary truck delays as well as the costs associated with such delays during the secondary and tertiary procedures [5]. Consequently, some of the other possible sources of additional data are depicted in the upper half of Figure 1. The implementation of the improved strategy requires techniques for analyzing data from each of the sensors, a framework for combining the possible massive volume of data from these multiple sensors, a methodology for quantifying the outcomes from each sensor data in terms of security risk, and an approach to fuse these outcomes into a single global measure of risk for real-time decisions. In addition, the final decision must be available to aid the inspection officers within a few seconds because commerce cannot be interrupted or delayed beyond the normally allowed time to process a truck. These requirements motivate the generalized knowledge discovery framework and the anomaly detection approach presented in this paper.

The rest of the paper is organized as follows. A generalized knowledge discovery framework for identifying anomalous commercial trucks from multiple sensors is presented in Section 2. In Section 3, we present our anomaly detection approach for one of the sensor data – static scale data. In Section 4, we discussed the results of the application of our anomaly detection approach to simulated and benchmark data. An application of the anomaly detection approach to static scale data is presented in Section 5. A discussion of the results and some conclusions about this paper is presented in Section 6.

2. THE GENERALIZED FRAMEWORK

In this section, we present our generalized framework; prior to that, we describe some of the challenges associated with anomaly detection in weigh station application and possible approaches to addressing these challenges. These approaches are the foundation for the proposed framework.

2.1 Challenges of Anomaly Detection in Weigh Station Application

The ability to detect commercial trucks transporting illicit radioactive materials poses several challenges for real-time decisions. Firstly, trucks transporting illicit radioactive materials are rare; therefore, no known cases or threat indicators of anomalous trucks are available. In other words, there is an inherent difficulty in precisely defining and quantifying what constitutes anomalies. Therefore, an incremental approach for anomaly detection may be more appropriate; so that as more examples of normal and anomalous trucks become available, the anomaly detection process is enhanced. Secondly, there are several legitimate cargoes, such as fertilizer and kitty-litter in commerce and medical isotopes used in hospitals that contain NORM and TENORM respectively. It is a known fact that most of the initial alarms that are false alarms are attributed to the presence of these common cargoes in the affected trucks. This then indicates that these legitimate cargoes are readily available and easy to procure; furthermore, the high background level of some of them including kitty-litter makes them a likely material for shielding illicit radioactive materials during transportation. Hence, there is a need for a methodology for detecting anomalies with minimum false alarms and zero missed detection. This approach will enhance the overall inspection procedures by reducing the number of secondary and tertiary inspections.

Thirdly, there are several truck features that may or may not have direct relationships with the presence of illicit materials in a truck. These features include the truck sizes, truck length, license plate, and loading patterns. However, the use of these truck features along with RN data may enhance the detection process and provide useful insights for classifying the data of the existing trucks as well as the new ones. There should then be an approach for making this connection. The availability of some domain rules may be useful in this situation. Fourthly, the inspection officer must make a decision within a few seconds using all the available data because commerce cannot be interrupted or delayed beyond the normally allowed time to process a truck. The number of sensor data and the volume of data collected from trucks of various features and cargoes cannot be evaluated in real-time without disrupting commerce through unnecessary truck delays. One way to address this challenge is to perform a part of the analyses offline using all or most of the available data – offline knowledge discovery models; the statistics from the offline models can then be used in real-time models for evaluating new trucks on a daily basis. The offline mode should be maintained, updated, and/or enhanced as often as each sensor data demands.

In order to address some of these challenges, we present a generalized knowledge discovery framework that is a two-stage knowledge discovery (KD) methodology. It consists of both offline and real-time KD approaches. The offline KD approach utilizes all the truck data available based on the respective requirements of each sensor. The real-time KD approaches uses the statistics obtained during the offline KD process to determine the risk level of a new truck. The offline KD model is updated on a regular basis to guarantee consistency in the detection process.

2.2 Distributed Framework for Anomaly Detection in Weigh Stations

The presented framework is a parallel configuration of N sensors as depicted in Figure 2. The framework assumes that all the sensors observe a single system-of-interest (e.g., a new truck into the weigh station) but each sensor focuses on different subset of the system (e.g., truck cargoes versus truck license plate). Each sensor has an associated offline and real-time knowledge discovery engines. The real-time knowledge discovery engines pass on their decisions to a fusion center where a global decision is reached. Let S_i denote either a single observation or multiple observations at the ith sensor ($i = 1,\ldots,N$). The observation S_i is passed to the *real-time knowledge discovery* ($RT\text{-}KD_i$) engine node to obtain the *tentative local decision* (LD_i), which uses statistics from the *offline knowledge discovery* (OKD_i) engine. The fusion center combines these tentative local decisions to obtain a *global decision* that is used to categorize new trucks using some domain rules and time-sensitive domain inputs. This framework assumes that the sensors do not communicate with each other and that there is no feedback from the fusion center or the human in the loop to any sensor. This is a modification of the popular parallel topology with fusion center used in distributed sensor stream data [e.g. 6].

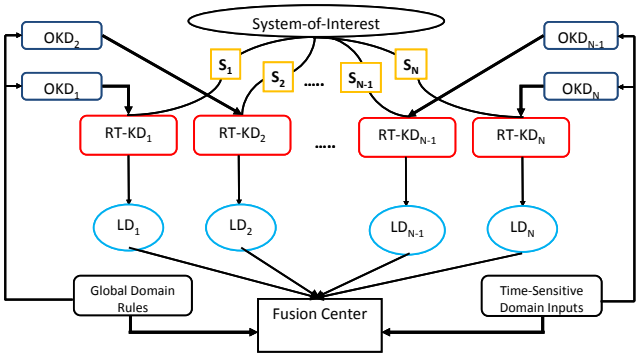


Figure 2. Parallel configuration for weigh station applications

The offline KD is necessary because the processing of all the trucks data usually takes time and may not be useful for real-time decision. This is a serious issue for knowledge discovery in security applications. Some of the techniques available in data mining literature include designing a faster algorithm, using a more efficient data representation, and partitioning the data [7]. In order not to delay trucks beyond the allowable time for inspection, the idea of data partitioning, which is also called incremental knowledge discovery, is the basis for this framework.

However, the data partitioning is implemented differently. For each sensor, the incremental knowledge discovery is defined as a process for maintaining the discovered patterns of the trucks passing through the weigh stations over time as more observations become available. Therefore, a two-stage knowledge discovery approach – offline and real-time knowledge discovery – is proposed. In the real-world, experts provide appropriate thresholds to obtain useful knowledge from massive data. The offline knowledge discovery engine in this framework provides such thresholds in this application with inputs from domain experts and other security-sensitive domain rules; the real-time knowledge discovery engine uses the threshold for evaluating and classifying a new observation. The threshold is maintained as long as the model is stable; if a model drift is detected, a new threshold (or classification) is created. The operations for updating the model include checking the validity of new observations, examining the replacement of old observations with some new observations, and adding new observations to the already existing observations. Each of the model updates is aimed at reducing false alarm of the already existing observations when using it for classifying new observations. This framework is applicable to all the sensors as data requirements permit. However, the requirements for designing the respective offline KD and real-time KD engines vary from sensor to sensor. For example, the requirements for static scale data (discussed later in this paper) is to generate a lower dimension that makes a sense of the seven truck features; for spectroscopy and gross count data, the requirements include identifying the common cargoes in commerce and developing a discriminant analysis techniques; a methodology for addressing these requirements was recently developed [3], [4]. Other sensor datasets have their requirements as well. As an illustration, a case study of the framework using observations from static scale data is presented in the following sections.

3. ANOMALY DETECTION APPROACH USING STATIC SCALE DATA

The truck static scale data describes vehicle attributes that may be useful for quantifying anomalous trucks in case the illicit materials are shielded using either some of the legitimate cargoes in commerce or physical objects such as a metal box. These attributes are vehicle length, vehicle weights at three locations, number of axles, vehicle speed as it approaches the primary inspection booth, and the distance of the vehicle from the sensor. One challenge with this data is the extraction of features that may be used to identify anomalies in the commercial trucks especially since none of these attributes is directly related to detecting the presence of illicit radioactive materials. However if such illicit materials are shielded using common legitimate cargoes in commerce or physical objects such as thick metal box in order to preclude the detection of any radiation source. The extra weight due to the shielding materials may affect the attributes of the truck. Furthermore, the attitude of the truck driver as the truck approaches the weigh station may also be a factor if the truck contains some illicit materials. This is under the assumption that the driver is aware of the content of his/her truck. The driver can go so slow in order to impress the inspection officers or he can go so fast in order to introduce more noise into the sensor readings. In any of these cases, the vehicle speed may then provide some useful insights. In summary, if the truck contains some illicit materials, the loading pattern, the weight pattern, or the driving pattern of the truck may not directly provide useful insights but the concomitant use of the insights from these data with the insights from other sensor data can provide more useful and effective global insights for more reliable global decisions. The use of all the static scale data attributes may not give consistent decisions. Therefore, one approach is to find a lower dimension

for characterizing the truck features in such a way that the inherent classifications and differences between trucks remain unaffected.

Some approaches have been suggested for finding appropriate low-dimensional feature representations for anomaly detection problems. In particular, [8] applies PCA to build a profile of normal events, computed the chi-square statistic of new events, and generates an alarm when the chi-square statistic exceeds the threshold. The use of nonlinear manifold embedding methods such as isometric feature mapping has also been proposed [9]; an application of this approach to weigh station static scale data was also presented using data collected over four months at one of the truck weigh stations. In this paper we combine the strength of these two approaches in order to reduce or eliminate their drawbacks. Specifically, our approach applies probabilistic principal components analysis (PPCA) to build a profile of normal events using iterative approach, computes chi-square statistic of new events, and generates an alarm when the chi-square statistic exceeds the 95% threshold. For this application, we used 2-dimensional feature because it can give valuable information about the structure of the data and can be directly fed into any standard anomaly detection method [8].

3.1 Review of Techniques

In this section, we review the concept of Latent variable models and discuss our proposed chi-squared statistic based anomaly detection approach.

3.1.1 Latent variable models

A latent variable model is a statistical model that investigates the dependence of observed variables on a set of latent variables [10]. The most well-known latent variable model is factor analysis, which was initially developed by psychologists. Recently, it has been found that many popular multivariate statistical techniques are closely related to latent variable models. These include vector quantization, independent component analysis models (ICA), Kalman filter models and hidden Markov models (HMMs) [11]. The general latent variable model has the following form:

$$p(\mathbf{x}) = \int p(\mathbf{x} \mid \boldsymbol{\theta}) h(\boldsymbol{\theta})\, d\boldsymbol{\theta}, \tag{1}$$

where $\mathbf{x} = [x_1, \ldots, x_M]^T$ represents the observable variables and $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_P]^T$ represents the latent variables. The number of latent variables, *P*, is usually much less than the number of observable variables, *M*. In essence all latent variable models assume that **x** has a joint probability distribution conditional on $\boldsymbol{\theta}$, denoted by $p(\mathbf{x} \mid \boldsymbol{\theta})$. Based on some assumptions, we can infer the density functions (*p* and *h*) from the known or assumed density of **x** in order to discover how the manifested variables depend on the latent variables. The key assumption of latent variable models is that of conditional independence; that is, the observable variables are independent of one another given the values of the latent variables. In other words, the observed interdependence among the observable variables totally comes from their common dependence on the latent variables; once the latent variables are fixed, the behavior of the observable variables is essentially random and this can be expressed as:

$$p(\mathbf{x}) = \int h(\boldsymbol{\theta}) \prod_{i=1}^{M} p(\mathbf{x}_i \mid \boldsymbol{\theta})\, d\boldsymbol{\theta}. \tag{2}$$

3.1.2 Probabilistic PCA

Principal component analysis (PCA) is a widely used statistical technique in data analysis. Due to its attractiveness for linear dimension reduction problems, it has recently been expressed as a maximum likelihood solution for a generative latent variable model. This recent idea is called probabilistic PCA [12] and defined as follows:

$$\mathbf{x} = \mathbf{W}_x \mathbf{t} + \mu_x + \varepsilon_x, \tag{3}$$

where $\mathbf{t} \in \Re^P$ are the latent variables, $\mathbf{W}_x$ is an $M \times P$ matrix called the factor loadings, and ε_x defines a noise process. In addition, parameters such as μ_x allow for non-zero means for the data. In this model, latent variables **t** are conventionally assumed as a standard Gaussian distribution; that is, $\mathbf{t} \approx N(\mathbf{0}, \mathbf{I})$ and ε_x takes an isotropic Gaussian form as $\varepsilon_x \approx N(\mathbf{0}, \boldsymbol{\sigma}_x^2 \mathbf{I})$. The maximum likelihood solution of $\mathbf{W}_x$ is given as:

$$\mathbf{W}_x = \mathbf{U}_P \left(\mathbf{E}_P - \boldsymbol{\sigma}_x^2 \mathbf{I}_P \right)^{\frac{1}{2}} \mathbf{R}, \tag{4}$$

where $\mathbf{U}_P$ is the matrix of the *P* principal eigenvectors of the sample covariance matrix $S_x = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu_x)(x_i - \mu_x)^T$, $\mathbf{E}_p \in \Re^{P \times P}$ is the diagonal matrix of the corresponding eigenvalues, $\mathbf{I}_p \in \Re^{P \times P}$ is the *P*-dimensional unit matrix, and **R** is an arbitrary $P \times P$ orthogonal matrix. It can be shown that PCA is a special case of PPCA as $\sigma_x^2 \to 0$. The probabilistic formulation of the PPCA provides additional advantages over PCA as discussed by Tipping [12]. These advantages include a principled method of handling missing values, the availability of a Bayesian framework, and a fast Expectation Maximization (EM) learning procedure.

3.1.3 Mixture of PPCA for feature reduction

Given the probabilistic formulation of PCA, it is quite straightforward to construct a mixture model with probabilistic principal component analyzers, whose parameters can be determined by the EM algorithm [13]. As Tipping has shown [14], the log-likelihood of the observed data for a mixture model is given as:

$$L = \sum_{i=1}^{N} \ln\left[\sum_{j=1}^{K} \lambda_j p\left(x_j \mid j\right)\right], \quad (5)$$

where $p\left(x_j \mid j\right)$ is a single PPCA model, λ_j is the corresponding mixing proportion with $\lambda_j \geq 0,$ and $\sum_{j=1}^{K} \lambda_j = 1$ with *K* as the number of mixture clusters. Since there is no close form solution for maximizing likelihood for this mixture model, a tractable iterative EM algorithm has been proposed for optimizing its parameters (see Appendix). The computational complexity of the iterative EM scheme for the mixture model is $O\left(M^3\right)$, compared to $O\left(MPN\right)$ for PCA, and $O\left(MN^2 + PN^3\right)$ for isometric feature mapping (ISOMAP). Considering its computational efficiency and its ability to handle nonlinearity, the mixture model can then be regarded as a trade-off strategy between global linear models such as PCA and nonlinear manifold learning such as ISOMAP.

3.2 Chi-square statistic for anomaly detection

Based on the mixture model of PPCA, we can characterize observed variables or latent variables by probability distributions. It can be shown that the observed variable **x** satisfies a mixture of Gaussian distributions under the mixture model as follows:

$$\begin{aligned} p\left(\mathbf{x} \mid j\right) &= \int p\left(\mathbf{x} \mid \mathbf{t}, j\right) p\left(t\right) dt \\ &= \left(2\pi\right)^{-M/2} \left|\mathbf{W}_j \mathbf{W}_j^T + \boldsymbol{\sigma}\mathbf{I}\right|^{-1/2} \\ &\quad \exp\left\{-\frac{1}{2}\left(\mathbf{x} - \boldsymbol{\mu}_j\right)\left(\mathbf{W}_j \mathbf{W}_j^T + \sigma\mathbf{I}\right)^{-1}\left(\mathbf{x} - \boldsymbol{\mu}_j\right)\right\}, \end{aligned} \quad (6)$$

The chi-square statistic for a new observation is then given as:

$$D = \left(\mathbf{x} - \boldsymbol{\mu}_j\right)^T \left(\mathbf{W}_j \mathbf{W}_j^T + \sigma\mathbf{I}\right)^{-1} \left(\mathbf{x} - \boldsymbol{\mu}_j\right). \quad (7)$$

If the distribution of $\mathbf{x} \in \Re^M$ is multivariate Gaussian, the statistic is approximately chi-square distributed with *M* degrees of freedom $\left(\chi_M^2\right)$. Multivariate outliers can now simply be defined as observations having a large squared Mahalanobis distance *C*. For our analyses, a 95% quantile is considered for the chi-square distribution.

3.3 The anomaly detection approach

It has been shown that Gaussian distribution is a more accurate density model for one-, two-, and three-dimensional data than for higher data dimensions [9]; we monitor the low dimensional latent variables **t**, which satisfies isotropic Gaussian distribution. Therefore, our anomaly detection approach is summarized in the following steps:

1. Assign each observation $\mathbf{x}_i$ a class label *j* based on the maximal posterior distribution R_{ij} computed by the EM algorithm for the mixture model of PPCA.
2. Calculate the chi-square statistic D_i for the corresponding latent variable $\mathbf{t}_i$ given the labeled class *j* defined as:

$$D = \mathbf{t}^T \mathbf{t}. \quad (8)$$

3. Detect anomalies based on the threshold of the 95% quantile of chi-square distribution with a degree of freedom that is equal to the dimension of the latent variable **t**.

This approach has an implicit assumption that for each class the majority of its members are normal and only a small fraction within the class is anomalies. In some real applications, the anomalies may consist of a whole class because they are other different classes.

4. EXPERIMENTAL EVALUATIONS

In this section, we evaluate the performance of our approach using synthetic and benchmark data respectively.

4.1 Simulated Data

In this experiment, 100 samples each were generated from two multivariate two-dimensional Gaussian distributions as $C_1 \approx N\left(\mu_1, \Sigma_1\right)$ and $C_2 \approx N\left(\mu_2, \Sigma_2\right)$ respectively, where $\mu_1 = \left[-5, -5\right]^T$, $\mu_2 = \left[5, 5\right]^T$, and $\Sigma_1 = \Sigma_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Two outliers, $n_1 = \left[5, -5\right]^T$ and $n_2 = \left[0, 0\right]^T$ replace the first and second observations in the first Gaussian cluster and another outlier $n_3 = \left[-5, 5\right]^T$ replaces the first observation in the second Gaussian. The data are shown in Figure 3 and the anomalies are identified with a circle. Using the proposed anomaly detection approach, Figure 4 shows the chi-square statistic for each observation and also the 95% quantile threshold. The figure shows that the observation 1, 2, and 21 are well above the threshold and thus clearly identified as anomalies. This result is consistent with our prior knowledge about these three observations.

4.2 Benchmark Data

We also illustrate the performance of our proposed approach on an example introduced in [15]. This dataset includes 45 observations of fish. The input variables consist of highly correlated spectra at nine wavelengths. The single output variable is the fat concentration of the fish. The objective in this case is to identify the relationship between the spectra and the fat

concentration. The input variables are shown in Figure 5, where observation 1 and 39 – 45 are highlighted.

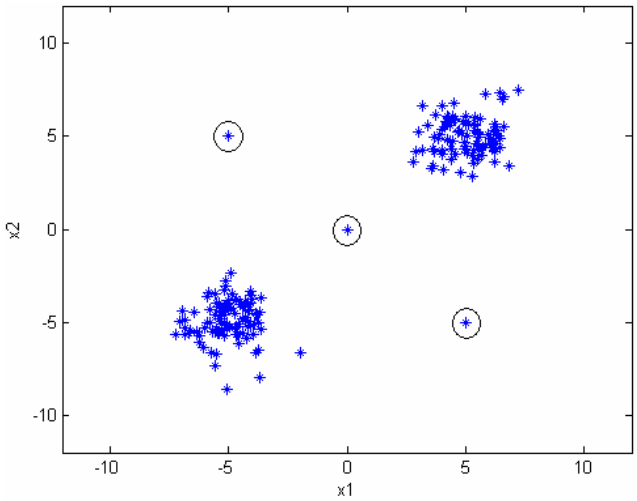


Figure 3. Simulated data from two dimensional Gaussian distributions with different mean and identical covariance. The circled data are the artificial anomalies.

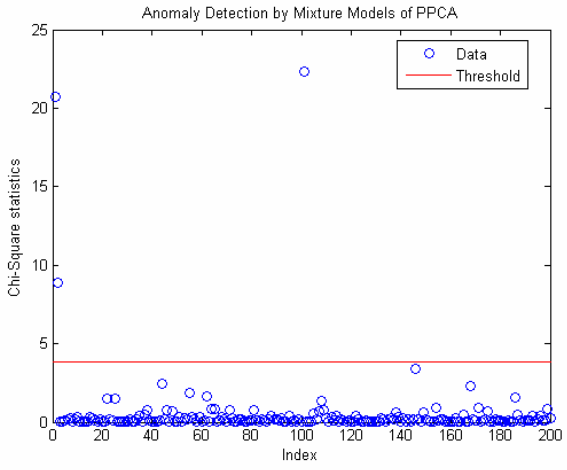


Figure 4. Outlier detection results for the 2-D synthetic dataset by mixture of PPCA; the red line indicates the threshold of the 95% quantile of chi-square distribution with a degree of freedom of 1.

It is reported in [15] that observations 39 – 45 are outliers; by observation of the plots in Figure 5 these observations except 42 have a spectrum that apparently deviates from the majority.

Figure 6 shows the results of our approach. The plot clearly shows that observations 1, 39 – 41, and 43 – 45 are identified as outliers. Compared to our prior knowledge about the data, only observation 42 was not detected. It should be noted that in this unsupervised framework, we discard the output variable. Hubert reported in [16] that robust PLS approach considers both the input and output information and still could not detect observation 42 as an outlier; furthermore, their approach classified observation 12 as an outlier. Compared to the previous approaches, our results are quite satisfactory.

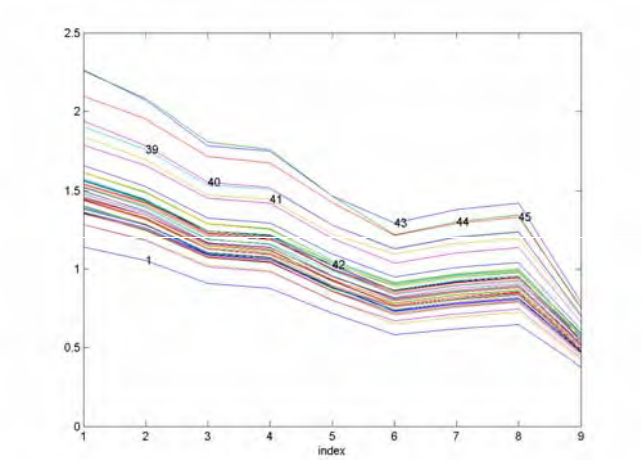


Figure 5. Observations of the benchmark data.

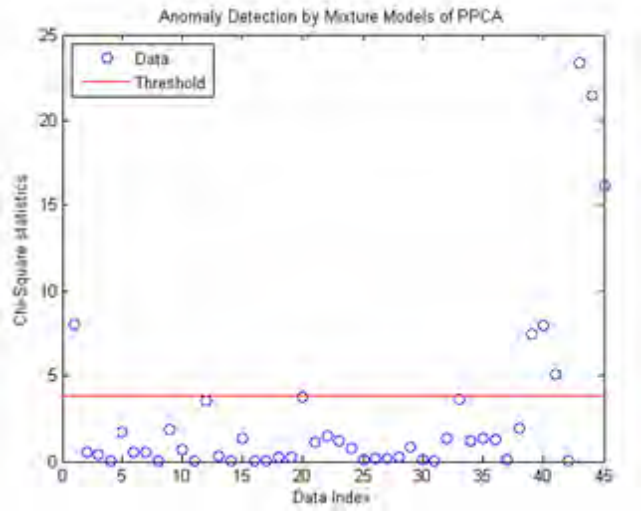


Figure 6. Outlier detection results for the fish dataset by mixture of PPCA; the red line represents the 95% quantile of the chi-square distribution with a degree of freedom of 1.

5. APPLICATION TO STATIC SCALE DATA

In this case study, we analyze the static scale data from the Watt Road weigh station on Interstate 40 in Knoxville. This data consists of seven features – truck lengths, truck weights at 3 locations, number of axles, vehicle speeds at the weigh station and the distance of the vehicle from the sensor; the data is collected over four months.

The dimension of latent variable **t** is predefined as two based on the previous studies [9]. Figure 7 shows the two dimensional latent variable for one cluster. Actually, the whole observations in this class represent anomalies because the number of its members is much less than that of another class. The original data for trucks in this "anomalous" class include trucks with vehicle speed over 70 mph, which validates our preconception. Within the "normal" class, we can still identify outliers based on the 95% quantile of chi-square statistic for this class (see Figure 8). They are the anomalies which are not as different as those in the "anomalous" class, but may still need a further investigation. To compare the chi-square statistic, we focus our analysis on the first 500 observations. Figure 9 shows the results with degree of freedom equals one ($P = 1$).

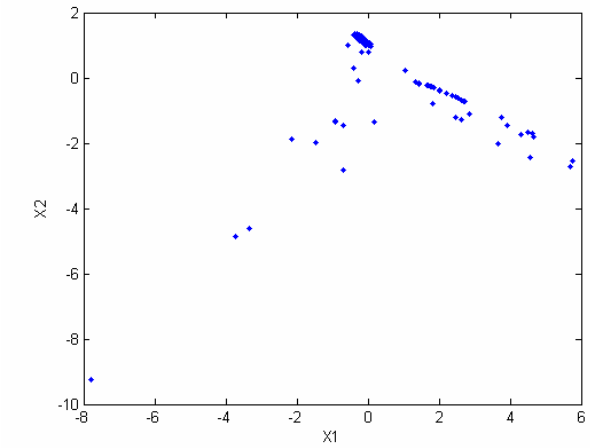


Figure 7. Two dimensional data projection for one cluster using mixture of PPCA.

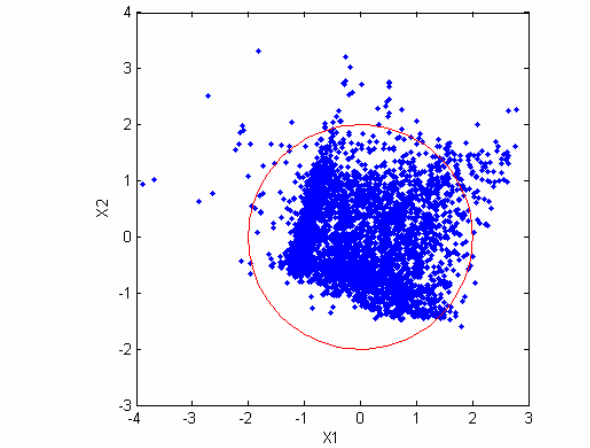


Figure 8. Two dimensional data projection for another cluster using mixture of PPCA; the red circle indicates the 95% confidence bound.

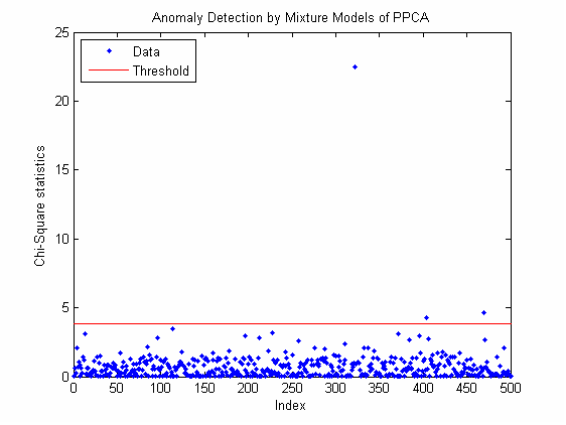


Figure 9. Outlier detection for 500 observations using mixture of PPCA; circles represent the chi-square statistics of the observations; the red line indicates the threshold.

These results indicate that the chi-square statistics of three trucks are above the 95% threshold; however, the statistics of two of the three trucks are closer to the threshold than the statistics of the third truck. One implication of these results is that if the threshold is increased to 99%, those two trucks will fall below the threshold but the other truck will still remain an anomaly. On the other hand, using a 90% threshold will definitely push more trucks in the anomaly class as could be infer from Figure 9. Therefore, a one size fits all approach may not be applicable here. Further investigations of the data show that only one of these three trucks could be described as anomaly in terms of its weight to length ratio. This is one area in which domain rules may be incorporated. Furthermore, multiple thresholds may be used in such a way that if a truck is classified as anomaly for most of the threshold levels, then there is more confidence to label the truck as anomaly and assign it a higher probability. That is, a probability is assigned to a truck based on the number of times it is classified as anomaly with respect to the number of possible times it could have been classified as anomaly.

6. DISCUSSION AND CONCLUSION

In this paper, we have presented a two-stage knowledge discovery framework for real-time anomaly detection from multiple sensor data with application to transportation security. The framework is motivated by the high number of false alarms usually encounter at weigh station test-beds using only one sensor data during the primary inspection procedure. The framework proposes using multiple sensor data including static scale data, gross count data, and spectroscopy data during the primary inspection. Furthermore, the knowledge discovery process is achieved in two stages with the first stage being the offline knowledge discovery stage and the second stage being the real-time (online) knowledge discovery stage. The OKD stage uses all available data from each sensor to extract significant statistics that can be used for characterizing the trucks. The extracted statistics are then used in the RT-KD stage to determine if new trucks to the weigh station are anomalies. The OKD stage is updated as more data becomes available. This framework is applicable to all the sensors and a case study using static scale data is presented.

We also presented an anomaly detection approach for static scale data. The approach was tested with simulated and benchmark data; it was found to detect anomalies using limited information. Our approach also shows that the computation of mixture model of PPCA is quite efficient. Moreover, the use of EM algorithm provides a straightforward extension to the real-time computation setting. The approach also consistently identified trucks with anomalous features in all scenarios investigated.

7. ACKNOWLEDGMENT

This research was funded by the Laboratory Directed Research and Development (LDRD) Program of the Oak Ridge National Laboratory (ORNL), managed by UT-Battelle, LLC, for the U.S. Department of Energy under Contract DE-AC05-00OR22725. We would like to gratefully acknowledge synergistic research activities by the SensorNet® Program managed by the Computational Sciences and Engineering Division of the Oak Ridge National Laboratory. We specifically thank the following people in the SensorNet® group: Frank DeNap, David Hill, David Feaker, Randy Walker, and Steven Saavedra. We also thank Dr. Brian Worley for his contributions to the Transportation Corridor research. We are grateful to the following scientists from ORNL who have reviewed this manuscript, either formally or informally: Mallikarjun (Arjun) Shankar, Raju Vatsavai, Randy Walker, Cy Smith, Bryan Gorman and Frank DeNap. This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains, and the publisher by accepting the article for publication, acknowledges that the United States Government retains, a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

8. REFERENCES

[1] Kouzes, R.T., Ely, J.H., Geelhood, B.D., Hansen, R.R., Lepel, E.A., Schweppe, J.E., Siciliano, E.R., Strom, D.J., and Warner, R.A. (2003). Naturally occurring radioactive materials and medical isotopes at border crossings, *IEEE Nuclear Science Symposium Conference Record*, vol. 2, pp. 1448-1452.

[2] Ely, J.H., Kouzes, R.T., Geelhood, B.D., Schweppe, J.E., and Warner, R.A. (2004). Discrimination of naturally occurring radioactive material in plastic scintillator materials, *IEEE Transactions on Nuclear Science*, vol. 51, no. 4, pp. 1672-1676.

[3] Olufemi A. Omitaomu, Auroop R. Ganguly, Bruce W. Patton, and Vladimir A. Protopopescu (in revision). Anomaly Detection in Radiation Sensor Data with Application to Transportation Security. *IEEE Transactions on Intelligent Transportation Systems*.

[4] Olufemi A. Omitaomu, Auroop R. Ganguly, and Vladimir A. Protopopescu (in review). Empirical Mode Decomposition Approach for Denoising Signals. *IEEE Transactions on Systems, Man, and Cybernetics: Part C.*

[5] Auroop R. Ganguly, Olufemi A. Omitaomu, and Randy M. Walker (2007). Knowledge Discovery from Sensor Data for Security Applications. In J. Gama and M. Gaber (Eds.), *Learning from Data Stream - Processing Techniques in Sensor Networks*, 187-204. Springer, NY.

[6] Viswanathan, R. and Varshney, P.K. (1997). Distributed detection with multiple sensors: Part 1 – Fundamentals. *Proceedings of the IEEE*, 85, 1, 54 – 63.

[7] Provost, F. and Kolluri, V. (1999). A survey of methods for scaling up inductive algorithms. *Data Mining and Knowledge Discovery*, 3, 131 – 169.

[8] Ye, N. and Chen, Q. (2001). An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems, *Quality and Reliability Engineering International*, 17:105 – 112.

[9] Agovic, A., Banerjee, A., Ganguly, A.R., and Protopopescu, V.A. (2008). Anomaly Detection in Transportation Corridors Using Manifold Embedding. In A.R. Ganguly, J. Gama, O.A. Omitaomu, M. Gaber, and R.R. Vatsavai (Eds.), *Knowledge Discovery from Sensor Data*, CRC Press, (To appear).

[10] Everitt, B.S. (1984). *An Introduction to Latent Variable Models*, London: Chapman & Hall.

[11] Roweis, S. and Ghahramani, Z. (1999). A unifying review of linear Gaussian models, *Neural Computaion*, 11(2), 305 - 345.

[12] Tipping, M. E., and Bishop, C. M. (1999a). Probabilistic principal component analysis, *Journal of the Royal Statistical Society* B, 61, 611–622.

[13] Dempster, A., Laird, N. and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society*, Series B, 39(1), 1-38.

[14] Tipping, M. E., and Bishop, C. M. (1999b). Mixtures of probabilistic principal component analyzers, *Neural Computation*, 11, 443–482.

[15] Naes, T. (1985). Multivariate calibration when the error covariance matrix is structured, *Technometrics*, 27, 301–311.

[16] Hubert, M., Rousseeuw, P., Verboven, S. (2002). A fast method for robust principal components with applications to chemometrics, *Chemometrics and Intelligent Laboratory Systems*, 60, 101–111.

9. APPENDIX

The Expectation (E) step leads to the sufficient statistics which are needed to update the parameters in the M (Maximization) step. The expectations are given by:

$$\left\langle \mathbf{t}_{ij} \right\rangle = \left(\sigma_j^2 \mathbf{I} + \mathbf{W}_j^T \mathbf{W}_j \right)^{-1} \mathbf{W}_j^T \left(\mathbf{x}_i - \boldsymbol{\mu}_j \right) \qquad (9)$$

and

$$\left\langle \mathbf{t}_{ij} \mathbf{t}_{ij}^T \right\rangle = \sigma_j^2 \left(\sigma_j^2 \mathbf{I} + \mathbf{W}_j^T \mathbf{W}_j \right)^{-1} + \left\langle \mathbf{t}_{ij} \right\rangle \left\langle \mathbf{t}_{ij} \right\rangle^T \qquad (10)$$

Then, in the M-step, the parameters are updated by the following equations respectively:

$$\lambda_j = \frac{1}{N}\sum_{i-1}^{N} R_{ij} \tag{11}$$

$$\boldsymbol{\mu}_j = \frac{\sum_{i=1}^{N} R_{ij}\left(\mathbf{x}_i - \mathbf{W}_j \left\langle \mathbf{t}_{ij} \right\rangle\right)}{\sum_{i=1}^{N} R_{ij}} \tag{12}$$

$$\mathbf{W}_j = \left[\sum_{i=1}^{N} R_{ij}\left(\mathbf{x}_i - \boldsymbol{\mu}_j\right)\left\langle \mathbf{t}_{ij}^T \right\rangle\right]\left[\sum_{i=1}^{N} R_{ij}\left\langle \mathbf{t}_{ij}\mathbf{t}_{ij}^T \right\rangle\right]^{-1} \tag{13}$$

$$\sigma_j^2 = \frac{1}{M\sum_{i=1}^{N} R_{ij}}(Q) \tag{14}$$

and

$$Q = \left\{ \begin{array}{l} \sum_{i=1}^{N} R_{ij} \left\| \mathbf{x}_i - \boldsymbol{\mu}_j \right\|^2 - \\ 2\sum_{i=1}^{N} R_{ij} \left\langle \mathbf{t}_{ij}^T \right\rangle \mathbf{W}_j^T \left(\mathbf{x}_i - \boldsymbol{\mu}_j\right) + \\ \sum_{i=1}^{N} R_{ij} tr\left[\left\langle \mathbf{t}_{ij}\mathbf{t}_{ij}^T \right\rangle \mathbf{W}_j^T \mathbf{W}_j\right] \end{array} \right\} \tag{15}$$

where R_{ij} is the posterior of mixture *j* for producing the observation $\mathbf{x}_i$ and is updated at the beginning of the E step. This EM algorithm is the same as that for a standard Gaussian mixture and similar to that of a single PPCA, except having R_{ij} as the local weighted term.

Network Service Disruption upon Natural Disaster: Inference Using Sensory Measurements and Human Inputs

Supaporn Erjongmanee
Georgia Institute of Technology
Atlanta, GA 30332
gtg730d@mail.gatech.edu

Chuanyi Ji
Georgia Institute of Technology
Atlanta, GA 30332
jic@ece.gatech.edu

ABSTRACT

Natural disasters cause large-scale network service interruption which corresponds to unreachability of networks. This problem relates to how networks respond under extreme conditions, and it is neither well-studied nor well-understood. To infer network service disruption, challenges arise, i.e., how to use heterogeneous data that include sensory measurements and human inputs?

This work shows an important role of data mining and machine learning in inferring large-scale network service disruption upon Hurricane Katrina. We present a joint use of large-scale sensory measurements from Internet and a small number of human inputs for effective network inference. Specifically, data mining includes (a) unsupervised learning, i.e., clustering and feature extraction of sensory measurements and (b) semi-supervised learning of both sensory measurements and human inputs.

The approaches are evaluated on network service disruption induced by Hurricane Katrina at subnet level. Our result shows that clustering reduces the spatial dimensionality by 81%, and sensory measurements are temporally extracted down to two features. The subnet statuses inferred by the classifier derived from semi-supervised learning show interesting facts of network resilience and provide the spatial and the temporal maps of network service disruption that can be used to assist disaster response and recovery.

To our understanding, this is the first work of data mining and machine learning using sensory measurements and human inputs for inference of large-scale network service disruption upon a large-scale natural disaster.

Categories and Subject Descriptors

J.2.8 [**Computer Applications**]: Internet Applications; H.2.8 [**Database Management**]: Database Application—*data mining, feature extraction*

1. INTRODUCTION

Internet is composed of a large number of heterogeneous sub-networks (subnets). Subnets can become unreachable after the occurrence of natural disasters, resulting in large-scale network service disruption. To provide the reliability and the reachability of networks, measurements of subnets are collected for performance and service monitoring. In a general setting, devices that perform data collection, e.g., border routers, can be regarded as "sensors," and measurements collected can be considered as sensory measurements.

Besides sensory measurements, this work introduces a novel use of human inputs to aid the inference of network service disruption. Human inputs correspond to human reports on network outages. While sensory measurements can be plenty, human inputs are generally available in a small number. This work shows how to apply data mining and machine learning to sensory measurements and human inputs to perform knowledge discovery of network service disruption upon natural disasters.

1.1 Challenges and Contribution

As analytical models of network services are unavailable, sensory measurements are imperative for inferring service disruption. However, several challenges arise and hinder the advance of this inference application.

The first challenge is that sensory measurements are large-scale. A monitored network generally consists of thousands of subnets, resulting in measurements of a high spatial dimension. For example, in this work, we consider 1009 time-series sensory measurements from 1009 subnets.

Another challenge is complex temporal patterns in sensory measurements. Networks exhibit unknown transient behaviors in response to a disaster, and the corresponding measurements generally have bursty temporal characteristics that are complex for inference.

The third challenge is the heterogeneity of data. In addition to sensory measurements, human inputs provide a distinct type of data. A human input is a "network-911-call" that a disaster responder makes to report network outages. In general, a report is made at a particular time instance but aftermath and delayed. Human inputs are usually available in a small number of subnets. The other data in this work are geographic locations and network addresses of subnets.

The current state of art in inferring service disruption relies solely on sensory measurements. Human inputs, although available, have been excluded from inference. An open issue is how to jointly use a large number of sensory measurements and a small number of human inputs for more effective inference of large-scale service disruption.

This work uses offline sensory measurements and human inputs from a large-scale natural disaster, i.e., Hurricane Katrina. We first provide a problem formulation in the context of machine learning. We then apply unsupervised learning, i.e., clustering and feature extraction to unlabeled data that are time-series sensory measurements. This reduces the spatial dimension of time-series by more than 80% and the temporal dimension down to two features. After that, semi-supervised learning is performed by combining human inputs with unlabeled data for inference. Because human inputs are delayed and thus not in a usable form of labels, we show how human inputs are converted into labeled data by indexing the unreachability pattern in time-series measurements. We then apply semi-supervised learning algorithm to both labeled and unlabeled data to infer service disruption.

The main contributions of this work lie into two aspects. First is the application of data mining and machine learning to a novel networking problem, i.e., inference of large-scale network service disruption upon a natural disaster. Second is the demonstration of the need and the effectiveness of learning from both heterogeneous sensory- and human-data.

The paper is organized as followed. The rest of Section 1 presents background and heterogeneous data. Section 2 provides problem formulation. Sections 3 and 4 respectively show the use of unsupervised and semi-supervised learning to sensory measurements and human inputs. Results are presented in Section 5. Section 6 discusses related works, and Section 7 concludes the paper.

1.2 Hurricane Katrina

Hurricane Katrina was the most severe hurricane that flooded Louisiana, Mississippi, and Alabama in 2005 and caused large-scale disruption in telecommunication networks. Network connectivity was critical but either unavailable or unstable experienced by disaster responders [1, 2]. There were a few public reports that showed the disaster impact to communications at Internet scale [3, 4] but there has not been any study on detailed service disruption at subnet level.

1.3 Network Monitoring

Network service disruption can be characterized as unreachability of subnets. This service disruption has been studied for day-to-day network operations [5] or by using simulation to infer large-scale network failures [6]. But the questions arise pertaining to service disruption caused by a real large-scale natural disaster. How to remotely infer unreachable subnets? What measurements can be used?

Sensory measurements from Internet routing infrastructure can be used for remote monitoring and service disruption inference [7, 8]. Internet consists of interconnected autonomous systems (AS), and the routing protocol among ASes is the Border Gateway Protocol (BGP) [9]. Each AS is served by at least one Internet service provider (ISP) and is composed

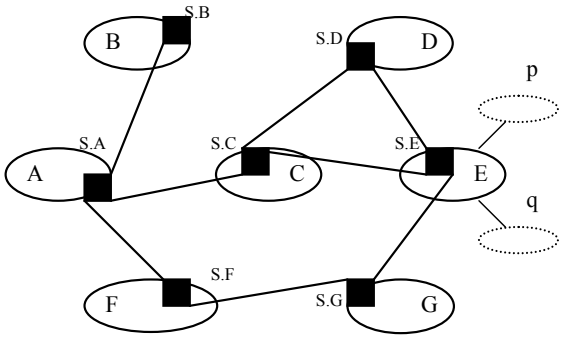


Figure 1: Example of AS network.

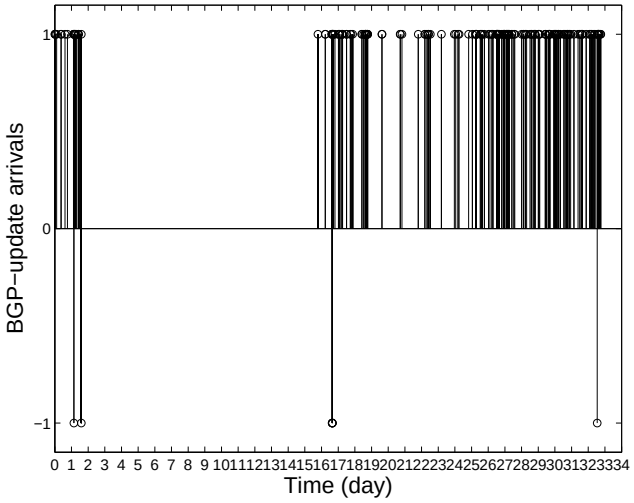


Figure 2: Example of time-series sensory measurements. (1 = BGP announcement, -1 = BGP withdrawal)

of one or several subnets identified by prefixes (network addresses)[1]. In order to route traffic from one AS to a specific subnet, a BGP router at each AS collects streams of routing messages from peering BGP routers of its neighbor ASes. These messages are called BGP update messages and are regarded as raw Internet sensory measurements in this work. Figure 1 shows the example of AS network where X is an AS, S.X is the BGP router of AS X, and X $\in$ {A,B,...,G}. AS E has two prefixes p and q. It also shows that the BGP router S.C collects Internet sensory measurements from peering BGP routers S.A, S.D, and S.E.

There are two types of BGP update messages: BGP withdrawal and BGP announcement. When a subnet becomes unreachable, all BGP routers that can no longer route Internet traffic to this subnet send BGP withdrawals to notify all of their peering routers the unreachability. When a subnet becomes reachable again, there would be new BGP announcements for this subnet. Note that besides network service disruption, multiple withdrawals followed by new announcements can also be caused by other network events, e.g., a change of routes or routing policies. Hence, a burst

[1]We shall use subnet and prefix interchangeably.

of multiple withdrawals followed by new announcements is a symptom rather than a one-to-one mapping of network service interruption [7, 8].

BGP update messages in this work are collected and stored by Oregon Route Views [10] and are publicly-available. In 2005, Oregon Route Views had about 35 geographically-distributed peering BGP routers. Oregon Route Views provides about 96 files of BGP update messages available per day, and the size of each file is approximately 8 megabytes.

1.4 Large-Scale and Heterogeneous Data

We obtain the real sensory measurements and the real human inputs from Hurricane Katrina. In particular, we choose sensory measurements to be BGP update messages that can provide remote monitoring of service disruption when local measurements are not directly available due to the evacuation and the limited accessibility to the disaster area.

Geographic locations are pertinent for selecting subnets in the disaster area to study. We identify geographic locations of subnets from Whois database [11] and select 1009 subnets from 48 ASes in the disaster area. This results in 1009 time-series sensory measurements, one per subnet. Figure 2 shows an example of time-series sensory measurements.

We choose our study duration as the Katrina interval to be between August 28 and September 4, 2005. Note that the mandatory evacuation was announced on August 28, 2005, one day prior to the Katrina landfall (August 29, 2005, 6:00 a.m., Central Daylight Time (CDT)), and most of network damage assessment, reported by our collaborating ISP, occurred within the first week after the landfall. In addition, we also select BGP update messages belong to the same subnets but between August 1-28, 2005 for comparison; this study period is called the pre-Katrina interval.

With 1009 subnets and eight-day duration, our sensory measurements are both spatially and temporally large-scale. As a burst of BGP messages is a symptom rather than a one-to-one mapping of service disruption, sensory measurements alone are insufficient to infer unreachability of all subnets.

Human inputs are reports of "this network is down". We collect total 37 human inputs from two sources. The first 28 human inputs are from the online message on NANOG mailing list posted by Todd Underwood from Renesys Corporation [4]. The other nine human inputs are network outage reports from customers of our collaborating ISP. Human inputs provide valuable and mostly accurate information on network outage status but can be delayed from the exact time that outage occurs. Thirty-seven human inputs are unlikely sufficient for inferring statuses for the other nearly 1000 subnets. Hence, sensory measurements and human inputs complement each other in inference of service disruption.

2. PROBLEM FORMULATION

Consider an underlying network with n nodes, where a node corresponds to a subnet. Let $Z_i(t)$ be a binary state of node i, $Z_i(t) = 1$ if node i is outage (unreachable); $Z_i(t) = -1$ if node i is normal (reachable); $1 \leq i \leq n$, and $t \in [0, T]$ is a time duration of interest. The state of a network is a collection of all n states, $Z(t) = \{Z_i(t)\}_{i=1}^{n}$, $t \in [0, T]$, and considered to be unknown. For our case, $n = 1009$, and $T = 8$ days (August 28-September 4, 2005). Service disruption is defined to be the same as unreachability of an individual subnet[2].

Let $X(t) \in R^n$ be an n-dimensional random vector that can be viewed as "response variables" corresponding to an underlying state $Z(t)$. Intuitively, $X(t)$ shows symptoms of $Z(t)$ and is related to both outage and normal states. A set D of m samples is assumed to be available on $X(t)$ and constitutes indirect observations on $Z(t)$. Hence, D is called unlabeled measurements. From Section 1.4, D corresponds to sensory measurements. In general, D is large-scale and insufficient for determining an underlying network state $Z(t)$ unless D is empowered by discriminative information.

Human inputs provide discriminative information. A set of k human inputs are assumed to be available for a fraction of nodes, i.e., $0 \leq k \leq n$. The simplest form of a human input is a symbol that takes binary values, 1 and -1, at time $t^{'}$. Let $t^{'}$ be the time that human reports the unreachability and t be the exact time that a network becomes unreachable. Generally, it is assumed that human reports unreachability of a subnet correctly[3], but a report can be delayed, i.e., $t^{'} > t$. Thus, a human input can be regarded as a direct but delayed observation on one specific nodal state $Z_i(t)$. A set of k human inputs is D_l, where k can be small, i.e., $0 \leq k \ll m$. In this work, we use 24 human inputs (65%) to be training data and the other 13 for validation. Hence, for our case, $k = 24$, $m = n - k = 985$.

Problem: Given a set of unlabeled sensory measurements, D, and a set of human inputs, D_l, how to infer $Z(t)$ for $t \in [0, T]$?

This is an inference problem where dichotomies between outage and normal states of subnets can be learned from sensory measurements and human inputs. Hence, we resort to data mining and machine learning approaches outlined below.

- We apply unsupervised learning algorithms that are clustering and feature extraction. Clustering is used to reduce the spatial dimension of time-series sensory measurements. We then extract the temporal features from time-series measurements to a fewer observations in a low-dimensional feature space and use these features as unlabeled data.
- We apply semi-supervised learning algorithm. We first convert human inputs to human labels by assigning dichotomies to a small number of the temporal features in the low-dimensional feature space. After that, a large set of unlabeled data and a small set of labeled data are combined to infer the statuses of subnets.
- We provide an initial understanding of network service disruption upon Hurricane Katrina and discuss

[2]We shall use unreachability, outage, and service disruption interchangeably.
[3]Note that this is a natural assumption as human only reports when a network is outage.

Table 1: List of prefix subsets with (a) Geographic location (LA = Louisiana, MS = Mississippi, AL = Alabama) (b) Number of prefixes, and (c) Reduction percentage

Set	1	2	3	4	5	6	7
(a)	LA	LA	LA	LA	LA	MS	AL
(b)	166	53	49	115	180	232	214
(c)	84.3	56.6	67.4	81.8	76.7	81.5	90.7

Table 2: Example of geographic location and time-series pattern belong to two subnets in the same cluster.

Sub net	Geographic Location	Initial t where $r(t) = -1$	Duration of $r(t) = -1$
1	Hammond, LA	8/30 18:53:42	2 hrs 53 mins
		9/3 23:39:09	17 mins
		9/4 00:25:10	10 mins
2	Hammond, LA	8/30 18:53:42	2 hrs 53 mins
		9/3 23:39:09	17 mins
		9/4 00:10:24	10 mins
		9/4 00:25:10	10 mins

the further use of the results and the applications in future network study.

3. UNSUPERVISED LEARNING

We now perform unsupervised learning to extract features from 1009 time-series sensory measurements belong to our selected 1009 subnets. The first step is to cluster these time-series to reduce the spatial dimension. The second step is to extract temporal features from patterns in the time-series.

3.1 Spatial Clustering

Features can be extracted directly from time-series measurements of each individual subnet. However, 1009 subnets are large-scale, and subnets may have experienced correlated service disruption caused by the same disaster. Therefore, we first reduce the spatial dimension of time-series measurements by grouping similar time-series into clusters.

To measure the similarity of time-series from different subnets, we change the discrete time-series of BGP update messages for a subnet i to be the continuous waveform $r_i(t)$ such that: when BGP announcement arrives at time t, $r_i(t) = 1$; otherwise, for BGP withdrawal, $r_i(t) = -1$. Consider time t, suppose two consecutive BGP updates arrive at time t_1 and t_2, $r_i(t) = r_i(t_1)$ for $t_1 \leq t < t_2$. For a subnet i without BGP update arrival, $r_i(t) = 1$ for all $t \in [0, T]$.

The similarity between $r_i(t)$ and $r_j(t)$ of subnet i and subnet j is measured by the average distance $d(r_i(t), r_j(t))$, where $d(r_i(t), r_j(t)) = \frac{1}{T}\int_{t=0}^{T} |r_i(t) - r_j(t)|dt$ for $1 \leq i, j \leq n$. The set of similarity measures, $L = \{d(r_i(t), r_j(t))\}$, where $1 \leq i, j \leq n$, is used as the input for clustering.

We choose the average-linkage hierarchical clustering algorithm since a number of clusters does not need to be pre-chosen. After clustering, we further post-process to obtain a fewer clusters by merging any two clusters if the similarity between them is smaller than a parameter $\hat{T}$. The range of $\hat{T}$ values is varied and tested using the Davies-Bouldin index [12] to determine cluster compactness. The suggested values of $\hat{T}$ are between 45-90 minutes. This can also be interpreted such that two time-series are merged into the same cluster if their similarity measure is smaller than $\hat{T}$.

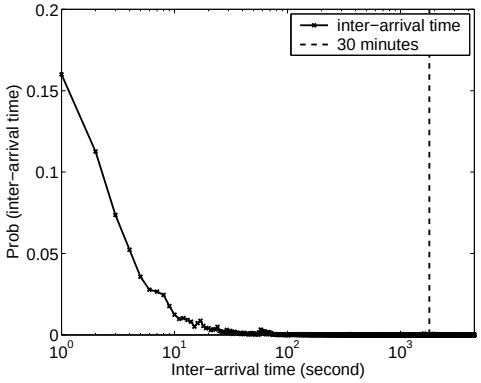


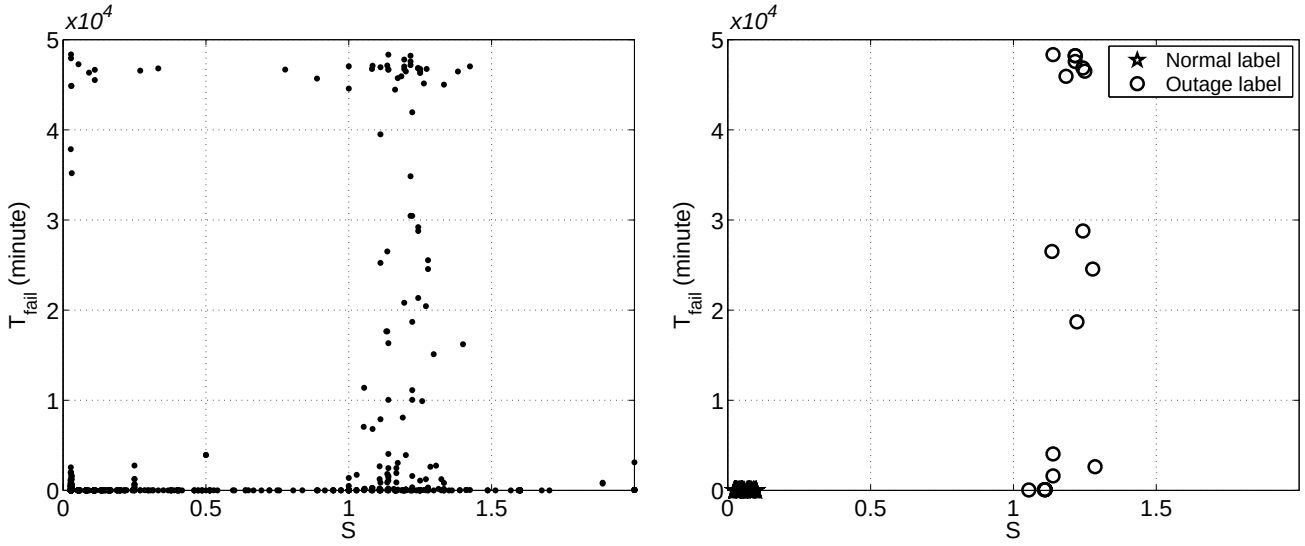
Figure 3: Empirical distribution of BGP withdrawal inter-arrival time.

Clustering spatially reduces 1009 time-series to 191 clusters, resulting in 81% reduction. Although, the simple hierarchical clustering algorithm gives the reasonably good performance, other advanced clustering algorithms can be applied to handle measurements with small similarity measures. The reduction percentages are also obtained for smaller prefix sets by separating 1009 prefixes into seven subsets based on the customers of seven local ISPs in the disaster area, and the reduction percentage of each subset is shown in Table 1. In details, each cluster contains the prefixes that have a correlation coefficient of $r_i(t)$'s between 0.9986-1.000. Table 2 shows the example of two subnets from the same cluster. This shows that subnets from the same cluster have a highly similar pattern of BGP updates, and the geographic locations belong to these subnets are similar.

3.2 Temporal Feature Extraction

Because the resulting clusters have correlation coefficient almost one, we randomly choose one representative prefix per cluster and use this much smaller set of 191 representative prefixes to extract temporal features of time-series.

As described in Section 1.3, a burst of multiple BGP withdrawals followed by new BGP announcements is a symptom of network service disruption. Thus, there are two features of this symptom. The first is a burst of withdrawals. A burst characterizes a number of withdrawal messages that peering BGP routers send in a given time-duration. The second is the length of an unreachable duration between the last withdrawal of a burst and the new announcements after a burst. This duration can be used to infer whether a subnet is unreachable upon a disaster or not. Thus, a burst



(a) Unlabeled data (b) Labeled data

Figure 4: S and T_{fail} of unlabeled and labeled data.

of withdrawals followed by new announcements and a succeeding unreachable duration form a BGP-burst pattern.

The inference of network events from a BGP-burst pattern has been studied for day-to-day network operations [8]. For instance, a BGP-burst pattern with a short unreachable duration can be caused by a temporary service disruption, i.e., a change of routes or routing policies, and a prefix becomes reachable soon after. However, a BGP-burst pattern with a long unreachable duration is mostly caused by major service disruption. But questions arise: how many withdrawals are considered to be a burst, and how long is an unreachable duration of service disruption upon a large-scale disaster? Hence, we formally define features corresponding to a BGP-burst pattern.

Definition: Burst ratio S and unreachable duration T_{fail}

Let v be a time-duration in which a burst of BGP withdrawals is characterized. Let n_v be a number of accumulative BGP withdrawals belong to a subnet that peering BGP routers send within v time-duration, and n_p be a number of peering BGP routers that could reach this subnet prior to the Katrina interval. Note that a peering BGP router can send more than one BGP withdrawal after a disruption.

The burst ratio is defined as $S = \frac{n_v}{n_p}$, and S measures percentage of BGP withdrawals from peering BGP routers. The unreachable duration T_{fail} is defined as the time period between the last BGP withdrawal of a burst in v-duration and the first new BGP announcement after a burst. Therefore, S is the spatial variable indicating how many peering BGP routers fail to reach a subnet. T_{fail} is the temporal variable that characterizes an unreachable duration.

The parameter v is a time window such that if the inter-arrival time between two BGP withdrawals is larger than v minutes, these two withdrawals are not considered to be in the same burst. It is reported that, in day-to-day network operations, a burst generally lasts for 3 minutes [13] but can be up to 15 minutes [14]. However, there was no prior result on a burst caused by natural disasters. We derive the empirical distribution of BGP withdrawal inter-arrival time after Katrina as shown in Figure 3. We select v = 30 minutes that is large enough not to partition a burst. However, such a large v, a time window may include more than one burst. This shows a disadvantage of using a fixed-size time window to locate a burst. To be more precise in locating a burst, instead of monitoring only a number of BGP withdrawals, we can explicitly examine the content of every BGP withdrawal to check subnet reachabilities.

3.3 Feature Statistics

Statistics of S and T_{fail} belong to time-series measurements are collected from the Katrina interval, and the result is shown in Figures 4(a). We also collect S and T_{fail} statistics from the pre-Katrina interval and find that there are less features with large T_{fail} values in the pre-Katrina than in the Katrina interval. This lack of large T_{fail} in the pre-Katrina interval results in the difficulty to determine the appropriate unreachable duration of Katrina service disruption. Section 4 shows how to use human inputs to derive the threshold to determine the suitable duration of this service disruption.

4. SEMI-SUPERVISED LEARNING

We extract 217 (S, T_{fail}) features from time-series measurements belong to 191 representative subnets; these features can be used as unlabeled data. Note that subnets can have more than one (S, T_{fail}) feature while some subnets do not have (S, T_{fail}) features at all. However, can we use delayed human inputs to identify a BGP-burst pattern and to obtain labeled (S, T_{fail}) features? If so, sensory measurements and human inputs can be jointly used to infer service disruption.

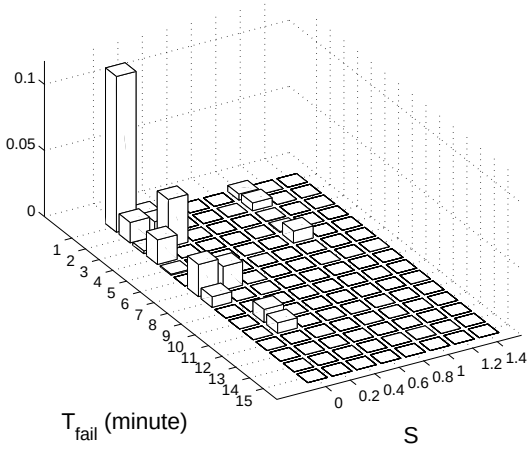


Figure 5: Empirical probability distribution of S and T_{fail} from the pre-Katrina interval.

4.1 Labeling Human Inputs

As a human input can be delayed, there can be more than one network service disruption and thus more than one BGP-burst pattern in the time-series measurements of this human input prior to the time of the human report. This shows that converting a delayed human report to a human label can be a complex process such that which BGP-burst pattern, if there is more than one, exactly corresponds to a service disruption that human reports. This work selects, for simplicity, the BGP-burst pattern immediately preceding a human report[4]. With 24 human inputs, we have 24 (S, T_{fail}) features that are labeled with "1" (outage).

To classify subnets into two dichotomies, outage and normal, we obtain (S, T_{fail}) features labeled as "-1" (normal) by using the pre-Katrina statistics. The assumption is made such that the majority of (S,T_{fail}) features in the pre-Katrina interval are normal. Figure 5 shows the empirical probability distribution of S and T_{fail} from the pre-Katrina interval. Small values, $S < 0.1$ and $T_{fail} < 3$ minutes, occurred with a large probability. This means that only a small (10%) percentage of peering BGP routers send out BGP withdrawals pertaining to a prefix while the rest of peering BGP routers can still reach this prefix. Moreover, with small T_{fail}, this can be interpreted that a prefix quickly becomes reachable after a BGP burst. Hence, prefixes with $S < 0.1$ and $T_{fail} < 3$ minutes are considered to be reachable. We extract 460 features of such values and then label these features as normal. Figure 4(b) shows (S, T_{fail}) features labeled as normal and outage.

In summary, we have 217 unlabeled features, $\{(S_i, T_{fail_i})\}_{i=1}^{217}$, 24 features labeled as outage $\{(S_i, T_{fail_i}), 1\}_{i=1}^{24}$, 460 features labeled as normal, $\{(S_i, T_{fail_i}), -1\}_{i=1}^{460}$.

4.2 Learning Labeled and Unlabeled Data

Labeled and unlabeled data have been jointly used and studied in prior works as semi-supervised learning. Prior work showed that learning with a small number of labeled data along with unlabeled data can reduce classification error from using only unlabeled data [15]. There are three major algorithms used in semi-supervised learning (see [16] and references in there), i.e., the generative models, the transductive support vector machine, and the graph-based methods. The generative models and the graph-based methods require probabilistic models. Thus, these two algorithms are infeasible because the human inputs we obtained are too few to estimate prior probability of outages accurately. Hence, we use the transductive support vector machine (TSVM) by Joachims [17] that only relies on labeled and unlabeled data.

Our goal is to train the (S, T_{fail}) classifier to determine whether prefixes are unreachable or not. To avoid overfitting, we choose the simple semi-supervised learning that applies TSVM to S and to T_{fail} separately. The resulting two one-dimensional linear classifiers (one for S and the other for T_{fail}) are used together as the two-dimensional classifier to infer the statuses of subnets.

Let x_i be labeled data and x_j^* be unlabeled data where $1 \leq i \leq k$, and $1 \leq j \leq m$, x_i or x_j^* is a generic variable in the algorithm that corresponds to either S or T_{fail}. Let y_i be the class label for x_i that is assigned as in Section 4.1, y_j^* be an unknown class label for x_j^* that is to be assigned by the classifier, and y_i, $y_j^* \in \{1, -1\}$. Let ξ_i be the so-called slack variable of x_i and ξ_j^* be the slack variable of x_j^*. The use of slack variables allows misclassified samples (see [18]).

Let w be the weight and b be the bias of a linear classifier to be obtained from minimizing

$$\frac{||w||^2}{2} + C\sum_{i=1}^{k}\xi_i + C_-^*\sum_{j:y_j^*=-1}\xi_j^* + C_+^*\sum_{j:y_j^*=+1}\xi_j^* \qquad (1)$$

subject to

$$y_i(w{\cdot}x_i + b) \geq 1 - \xi_i, \qquad (2)$$

$$y_j^*(w{\cdot}x_j^* + b) \geq 1 - \xi_j^*, \qquad (3)$$

$$\xi_i \geq 0,\ \xi_j^* \geq 0 \qquad (4)$$

where $\frac{2}{||w||}$ is the margin width of the classifier where $\sum_{i=1}^{k}\xi_i$ and $\sum_{j=1}^{m}\xi_j^*$ are bounds of classification error. C, C_-^* and C_+^* are tradeoff parameters between the margin width and the classification error (see [17] for details).

The outputs of the algorithm are w and b; $\frac{-b}{w}$ is a threshold for either S or T_{fail} to determine the class labels, $\{y_j^*\}_{j=1}^{m}$.

4.3 Experimental Setting and Validation

As unlabeled data is abundant, we separate the unlabeled features into 10 different subsets. Hence, 10 different classifiers are trained, and each training uses one separated subset of 21 (or 22) unlabeled features, all 24 features labeled as outage, and one subset of 30 randomly-chosen features labeled as normal. Other parameters used in the TSVM algorithm are initialized such that $C = 0.1$, $C^* = 0.1$, and $num_+ = 0.5$ (these parameters are related to convergence of the TSVM algorithm, and see [17] for details on a choice of parameters).

Let S^* and T_{fail}^* be the thresholds such that if any prefix has features $S > S^*$ and $T_{fail} > T_{fail}^*$, this prefix is inferred

[4]That is, humans are prompt in reporting a network outage.

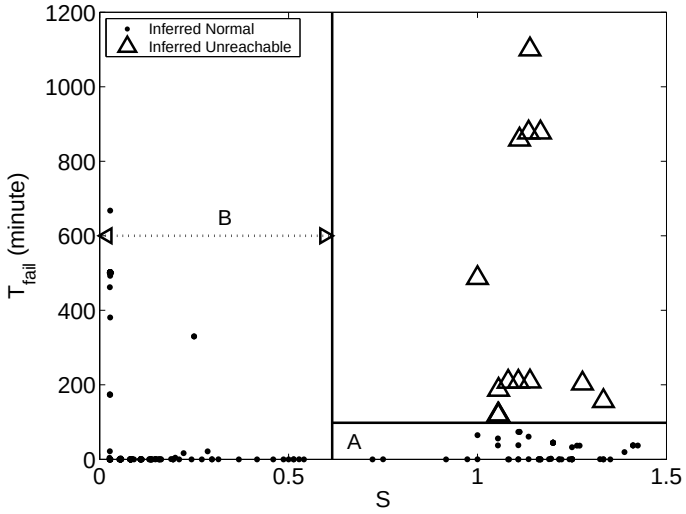


Figure 6: Scatter plot of inferred S and T_{fail}. (Solid vertical line: $S = S^*$, Solid horizontal line: $T_{fail} = T^*_{fail}$.) Plot only shows values of T_{fail} up to 1200 minutes.

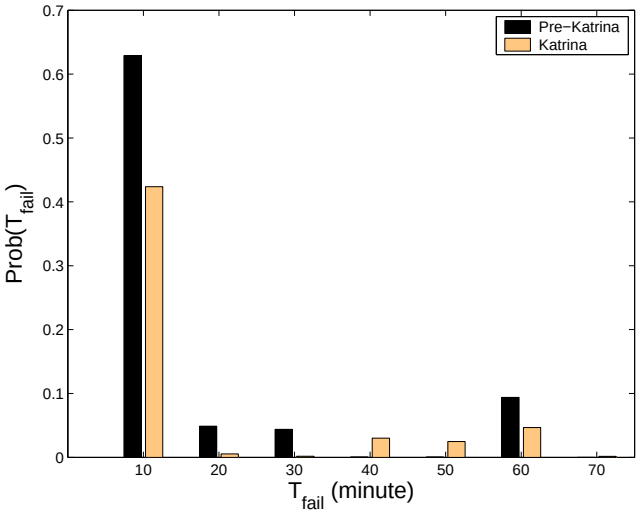


Figure 7: Empirical probability distributions of T_{fail} from the pre-Katrina and the Katrina intervals.

as unreachable upon Katrina. Ten thresholds of S resulting from training 10 different classifiers are averaged to yield S^*. We follow the same process to find the value of T^*_{fail}. This results in $S^* = 0.6153$ and $T^*_{fail} = 1$ hour 38 minutes.

We use the rest of 13 human inputs for validation. The result shows that the features belong to these 13 human inputs have $S > S^*$ and $T_{fail} > T^*_{fail}$ and thus are inferred as unreachable. The inferred unreachable statuses of these human inputs are consistent to the reports that these subnets were outages. Hence, the values of S^* and T^*_{fail} to infer unreachable prefixes are valid.

5. INFERRED SERVICE DISRUPTION

The thresholds learned are now used to infer service disruption caused by Katrina for the other 985 subnets.

5.1 Statistics of Subnet Statuses

The decision boundaries, $S = S^*$ and $T_{fail} = T^*_{fail}$, partition the feature space into two main regions shown in Figure 6:

- Outage region where $S{>}S^*$ and $T_{fail}{>}T^*_{fail}$ (upper right region in Figure 6). This region contains S and T_{fail} belong to the inferred unreachable subnets.
- Normal region that has either $S{\leq}S^*$ or $T_{fail}{\leq}T^*_{fail}$. This region contains S and T_{fail} belong to the inferred reachable subnets.

In normal region, there are two sub-regions marked as regions A and B in Figure 6. These two sub-regions contain the features that are inferred as normal but show the interesting characteristics of network resilience and responses upon Hurricane Katrina.

Region A is located where $S > S^*$ and $T{\leq}T^*_{fail}$. The features in this region correspond to the prefixes that after Katrina, multiple peering BGP routers responded with bursty BGP withdrawals, but these prefixes only experienced brief T_{fail} and resumed reachability soon after. The empirical probability distribution of T_{fail}, where $T_{fail}{\leq}T^*_{fail}$, presented in Figure 7, shows that there are significantly more T_{fail} with moderate values, 35-55 minutes, in the Katrina interval while T_{fail} of such values were scarce during the pre-Katrina interval[5]. This shows that Katrina caused network to respond differently from day-to-day network operations.

Region B is located where $S{\leq}S^*$. The features in this region correspond to the prefixes that only a small number of peering BGP routers responded to Katrina. Comparing among S statistics, we find that there are more S with values between $[0.1, 0.5]$ in the Katrina than the pre-Katrina interval. We also study some corresponding prefixes and find that these prefixes maintained the reachability statuses; hence, there might have been parts of Internet that were not highly affected and responded to Katrina.

We quantify the percentages of prefixes in these four regions as shown in Table 3. The results show that 25% of prefixes are inferred as outages, and there are 42% of prefixes from both regions A and B. With prefixes that maintained reachabilities or responded with brief disruption duration, this provides the signs of network resilience and suggests the meaningful direction to investigate the prefixes in regions A and B. This can result into an in-depth understanding of network resilience and responses upon a large-scale disaster.

5.2 Spatial-Temporal Damage Maps

We now obtain the spatial damage map presented in Figure 8. The spatial map shows network service disruption of different degree, based on the average disruption duration of the inferred unreachable prefixes in each geographic location. The worst service disruption occurred at subnets near the coast of Louisiana. Nevertheless, our results show

[5]For both intervals, probabilities of T_{fail} >80 minutes and $T_{fail} < T^*_{fail}$ are very small.

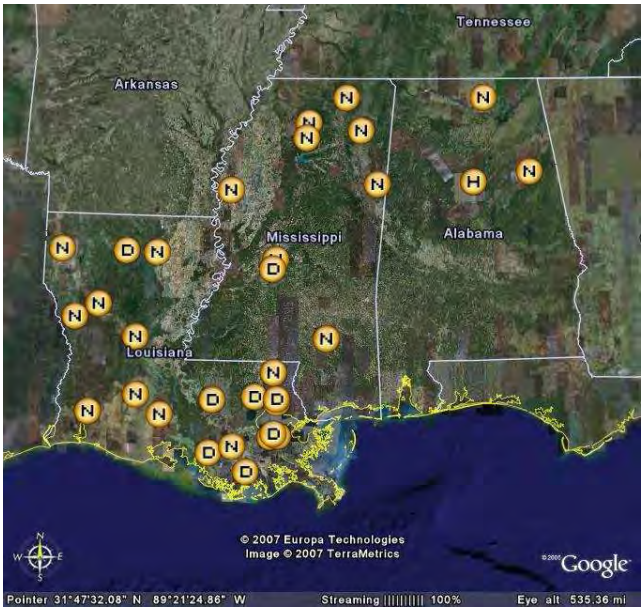


Figure 8: Impact degree of network service disruption. (N): T_{fail}<T^*_{fail}, (H): $T^*_{fail} < T_{fail} <$ 24 hours, and (D): $T_{fail} \geq$ 24 hours.

Table 3: Percentages of prefixes in four regions.

Region	Percentage of prefixes
Normal	75
Outage	25
A	12
B	30

that not all subnets in the entire disaster area suffered from service disruption. This suggests that there were available network resources in the area that could have been utilized if this information was shared among disaster responders.

We use T_{fail} to identify the initial time when service disruption started and the duration of service disruption. This results in the temporal map shown in Figure 9. The temporal map shows that 49.21% of service disruption occurred after the landfall while only 5.12% occurred on August 28, 2005 (the mandatory evacuation day). There were substantial service disruption (45.67%) occurred on August 29, 2005 before the landfall, and this service disruption will be discussed in a future study.

Communications are critical after disasters. The application of this work can be used in the future to infer network service disruption upon other disasters. The use of remotely-monitoring sensory measurements gives the advantage of the service disruption inference when the disaster area is physically inaccessible. Moreover, because ISPs do not generally disclose information on unreachability of their service networks, this application can be used to determine service disruption across different ISPs. Furthermore, this application can be further developed to use with online sensory measure-

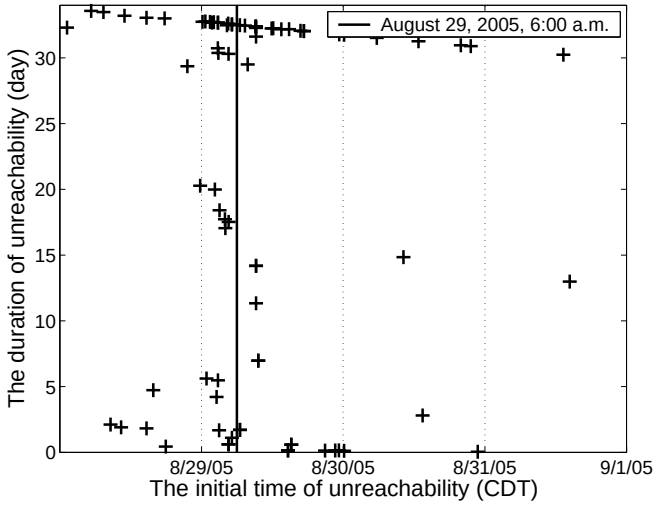


Figure 9: Initial and duration time of inferred network service disruption upon Katrina. ("+" = an inferred service disruption)

ments, and thus, it can infer service disruption in a timely fashion to assist rescue and recovery effort after disasters.

6. RELATED WORK

There have been studies of BGP update messages related to the widely affected network service disruption; the examples of these studies are the September 11 attack in 2001 [19], and the Code-Red and the Nimda worm attacks in 2003 [20]. In [3], Cowie et al. presented that some disrupted networks after Hurricane Katrina were not recovered after 10 days had passed. Among these studies, little has been done on detailed study of service disruption at subnet level using public available sensory measurements [3, 19]. Furthermore, human data has not been used in these prior works.

There have been studies of machine learning applications to BGP update messages. They were done either for day-to-day network operations or with different methods. For example, Andersen et al. applied the clustering algorithm to BGP update messages to infer a BGP topology [21] while Chang et al. temporally and spatially clustered ASPATHs to identify the cause of path changes [22]. Xu et al. proposed the algorithm to infer significant BGP events by applying the principal component analysis (PCA) to BGP updates [8].

Semi-supervised learning has been widely studied [15, 16] and has been applied in many applications such as text classification [17, 23], remote sensing [24], and image processing [25]. Nonetheless, semi-supervised learning has yet been applied in networking problem in previous studies.

7. CONCLUSION

This work has introduced data mining and machine learning to a new networking application as inference of large-scale network service disruption caused by Hurricane Katrina using sensory measurements and human inputs.

We have found that data mining has played a vital role in learning large-scale and complex sensory measurements in two aspects. First is that clustering has reduced the spatial

dimension of sensory measurements by 81%, and feature extraction has reduced the temporal dimension down to two informative features. Second is that semi-supervised learning makes use of a large number of sensory measurements and a small number of human inputs to derive the classifier of network service disruption upon Katrina.

The results show that 25% of subnets are inferred as unreachable. We also present the spatial and the temporal damage maps that are practical values to disaster response and recovery. A large fraction, i.e., 42% of prefixes are found to be either maintained or briefly resumed reachability after Katrina. This suggests the interesting directions for obtaining a deeper understanding of network resilience and responses under a large-scale disaster. These results would have been difficult to obtain without data mining, and this shows the usefulness of our approaches. Our application that is based on publicly available sensory measurements can be used to remotely monitor and localize the reachable network resources after large-scale disasters in the future.

This network application has presented challenges to the existing data mining as well as networking approaches. For example, how to use data mining with a large and complex data set in real time? How to in-depth study network resilience in response to a large-scale disaster? These provide some of future directions for our study and motivate developments of more advanced data mining applications.

8. ACKNOWLEDGEMENTS

The authors would like to thank Jere Stokely and Neale Hightower for their technical help with data and comments, Cheng Guang, Derrick Dy and Phong Do for help with data processing, Anwar Walid and Zesheng Chen for many helpful discussions. This paper is supported by NSF SGER-Katrina and ECS 0334759.

9. REFERENCES

[1] U.S. House of Representatives. A Failure of Initiative: Final Report of the Select Bipartisan Committee to Investigate the Preparation for and Response to Hurricane Katrina. Congressional Reports H. Rpt. 109-377, Washington, D.C., 2005.

[2] K. J. Martin. Written Statement of Kevin J. Martin, Chairman Federal Communications Commission, at the Hearing on Public Safety Communications from 9/11 to Katrina: Critical Public Policy Lessons, before the Subcommittee on Telecommunications and the Internet. U.S. House of Representatives, 2005.

[3] J. Cowie, A. Popescu, and T. Underwood. Impact of Hurricane Katrina on Internet Infrastructure. Renesys Corporation, 2005.

[4] T. Underwood. http://www.merit.edu/mail.archives/nanog/2005-08/msg00938.html.

[5] N. Feamster, et al. Measuring the Effects of Internet Path Faults on Reactive Routing. In *Proceedings of ACM SIGMETRICS on Measurements and Modeling of Computer*, pages 126-137, 2003.

[6] A. Sahoo, K. Kant, and P. Mohapatra. Characterization of BGP Recovery Time under Large-Scale Failures. In *Proceedings of IEEE International Conference on Communications,* pages 949-954, 2006.

[7] A. Feldmann, et al. Locating Internet Routing Instabilities. *ACM SIGCOMM Computer Communication Review,* 3(4): 205-218, 2004.

[8] K. Xu, J. Chandrashekar, and Z.-L. Zhang. Inferring Major Events from BGP Update Streams. Technical Report 04-043, University of Minnesota, 2004.

[9] Y. Rekhter, T. Li, and S. Hares. Border Gateway Protocol 4 (RFC 1771).

[10] University of Oregon. Route Views Project. http://archive.routeviews.org.

[11] Whois Database. http://www.arin.net/whois.

[12] D. L. Davies and D. W. Bouldin. A Cluster Separation Measure. *IEEE Transactions on Pattern Recognition and Machine Intelligence,* 1(2): 224-227, 1979.

[13] C. Labovitz, G. R. Malan, and F. Jahanian. Internet Routing Instability. *IEEE/ACM Transactions on Networking,* 6(5): 515-528, 1998.

[14] C. Labovitz, et al. Delayed Internet Routing Convergence. *IEEE/ACM Transactions on Networking,* 9(3): 293 - 306, 2001.

[15] V. Castelli and T. Cover. The Relative Value of Labeled and Unlabeled Samples in Pattern Recognition with an Unknown Mixing Parameter. *IEEE Transactions on Information Theory,* 42(6): 2101-2117, 1996.

[16] O. Chapelle, B. Scholkopf, and A. Zien. *Semi-Supervised Learning*. MIT Press, 2006.

[17] T. Joachims. Transductive Inference for Text Classification using Support Vector Machines. In *Proceedings of International Conference on Machine Learning,* pages 200-209, Bred, Slovenia, 1999.

[18] C. J. C. Burges. A Tutorial on Support Vector Machine for Pattern Recognition. *Data Mining and Knowledge Discovery,* 2(2): 121-167, 1998.

[19] Committee on the Internet under Crisis Conditions: Learning from September 11 National Research Council. *The Internet under Crisis Conditions*. The National Academies Press, 2003.

[20] L. Wang, et al. Observation and Analysis of BGP Behavior under Stress. In *Proceedings of ACM SIGCOMM on Internet Measurement Workshop,* pages 183-195, 2002.

[21] D. Andersen, et al.. Topology Inference from BGP Routing Dynamics. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop,* pages 243-248, Marseille, France, 2002.

[22] D. F. Chang, R. Govindan, and J. Heidemann. The Temporal and Topological Characteristics of BGP Path Changes. In *Proceedings of IEEE International Conference on Network Protocols,* pages 190-199, 2003.

[23] K. Nigam. Using Unlabeled Data to Improve Text Classification. Doctoral Thesis CMU-CS-01-126, Carnegie Mellon University, 2001.

[24] B. Shahshahani and D. Landgrebe. The Effect of Unlabeled Samples in Reducing the Small Sample Size Problem and Mitigating the Hughes Phenomenon. *IEEE Transactions on Geoscience and Remote Sensing,* 32(5): 1087-1095, 1994.

[25] J. Li and C. S. Chua. Transductive Inference for Color-based Particle Filter Tracking. In *Proceedings of International Conference Image Processing,* pages III 949-952, 2003.

WiFi Miner: An Online Apriori-Infrequent Based Wireless Intrusion Detection System

Ahmedur Rahman
School of Computer Science
University of Windsor
Windsor, Ontario N9B 3P4
rahma21@uwindsor.ca

C.I. Ezeife
School of Computer Science
University of Windsor
Windsor, Ontario N9B 3P4
cezeife@uwindsor.ca

A.K. Aggarwal
School of Computer Science
University of Windsor
Windsor, Ontario N9B 3P4
akshaia@uwindsor.ca

ABSTRACT

Intrusion detection in wireless networks has become a vital part in wireless network security systems with wide spread use of Wireless Local Area Networks (WLAN). Currently, almost all devices are Wi-Fi (Wireless Fidelity) capable and can access WLAN. This paper proposes an Intrusion Detection System, WiFi Miner, which applies an infrequent pattern association rule mining Apriori technique to wireless network packets captured through hardware sensors for purposes of real time detection of intrusive or anomalous packets. Contributions of the proposed system includes effectively adapting an efficient data mining association rule technique to important problem of intrusion detection in a wireless network environment using hardware sensors, providing a solution that eliminates the need for hard-to-obtain training data in this environment, providing increased intrusion detection rate and reduction of false alarms.

The proposed system, WiFi Miner solution approach is to find frequent and infrequent patterns on pre-processed wireless connection records using infrequent pattern finding Apriori algorithm proposed by this paper. The proposed Online Apriori-Infrequent algorithm improves the join and prune step of the traditional Apriori algorithm with a rule that avoids joining itemsets not likely to produce frequent itemsets as their results, there by improving efficiency and run times significantly. An anomaly score is assigned to each packet (record) based on whether the record has more frequent or infrequent patterns. Connection records with positive anomaly scores have more infrequent patterns than frequent patterns and are considered anomalous packets.

Keywords

Data mining, wireless intrusion, network intrusion detection, hardware sensors, infrequent patterns, no training data

1. INTRODUCTION

Security of computer networks has become a very crucial issue. Traditionally, the firewall is considered as the first line of defense, but the unsophisticated firewall policy cannot meet the requirements of some organizations, which need high security. Existing data mining based intrusion detection systems include ADAM [4], MADAMID [9], MINDS [5], DHP [10], LERAD [12], ENTROPY [18], but all these systems are designed for wired network environment. In the last few years, wireless technology has advanced rapidly, providing convenience and flexibility but few studies have been done on intrusion detection of wireless networks. Data mining has been applied successfully to wired network intrusion detection since early 2000. ADAM [4] was one of the early research that featured a system applying data mining techniques to the problem of network intrusion detection, using association rule mining Apriori algorithm [8]. Other systems include MADAMID and MINDS. MADAMID [9] focused on efficiency and automation of the process of network connection feature constructions. One limitation of these systems is that some of them are currently off-line but a more effective intrusion detection system should be real time, to minimize chances of compromising network security. Another limitation in some models is that they compute only frequent patterns in connection records. However, many intrusions like those that embed all activities within a single connection, do not have frequent patterns in connection data. These types of intrusions might go undetected in these models. A limitation of MINDS [5] is that it needs training data to learn the classifier and another limitation of MINDS is that it only analyzes the header parts of data and does not pay attention to payload. As a result, U2R (User To Root) or R2U (Root To User) attacks may go undetected in their system.

Our studies show that current wireless IDSs are still dependent on training data and without prior training these systems cannot detect intrusions in real time and some wireless IDS based on Association rule mining technique [11] detect intrusions only for ad-hoc network and are not applicable for infrastructure based WLAN. This paper proposes a network intrusion detection system (WiFi Miner) for wireless environment, which uses wireless hardware sensors to capture wireless traffic, which a newly proposed real time and online Apriori-Infrequent based data-mining algorithm promptly analyzes to detect new attacks. Wireless Fidelity (WiFi) is used to represent 802.11 wireless networks capable of transmitting data over short distances. Our WiFi Miner's proposed Real-time Online Apriori-Infrequent algorithm is introducing for the first time, the technique for analyzing incoming datasets to find infrequent patterns without any

.

prior training with safe data. The proposed technique can detect new types of wireless attacks efficiently with a reduced time complexity in comparison to traditional Apriori based systems and can flag anomalous connections in real time on the fly.

Types of Wireless Attacks

Wireless intrusions belong to four broad categories [17], namely: (1) passive attacks, (2) active attacks, (3) man-in-the-middle attack and (4) jamming attacks. A passive attack (e.g., war driving) occurs when someone listens to (or eavesdrops) on network traffic. Armed with a wireless network adaptor that supports promiscuous mode, the eavesdropper can capture network traffic for analysis using easily available tools, such as Network Monitor in Microsoft products, or (Transmission Control Protocol) TCPdump in Linux-based products, or AirSnort in Windows or Linux. War driving is the act of searching unsecured Wi-Fi networks by a person with a Wi-Fi equipped computer. As long as somebody is sniffing the network packets and trying to discover some useful information from gathered packets (e.g., WEP key used in the network or available open ports), we classify these activities as passive attacks. Once this information is discovered through passive attacks, then hackers can launch some active attacks. Active attacks launched by hackers who access the network to launch these active attacks include unauthorized access, Denial of Service (DoS) and Flooding attacks like (SYNchronized) SYN Flood attacks, and (User Datagram Protocol) UDP Flood attacks. DoS attack attempts to engage a host of computer resources so that these resources are not available to other users. DoS is an attack in which the attacker keeps the resource too busy or too full to handle other legitimate requests, and thus, it denies legitimate users access to a machine [14]. In SYN Flood attack, the attacker sends a lot of TCP packets, where both SYN and (ACKnowledgment) ACK flags in the header are set to 1 using tools like Engage Packet Builder [16]. The attacker's IP address is fake and destination IP address is the server victim's address. Receiving so many packets from attacker prevents victim from accepting new legitimate requests and may crash the victim server. Man-in-the-middle attack entails placing a rogue AP (Access Point) within range of wireless stations. If the attacker knows the SSID in use by the network (which is easily discoverable) and the rogue AP has enough strength, wireless users have no way of knowing that they are connecting to an unauthorized AP. Because of their undetectable nature, the only defense against rogue APs is vigilance through frequent site surveys using tools such as Netstumbler and AiroPeek, and physical security. Jamming is a special kind of DoS attack specific to wireless networks. Jamming occurs when spurious RF (Radio Frequency) frequencies interfere with the operation of the wireless network. Intentional and malicious jamming occurs when an attacker analyzes the spectrum being used by wireless networks and then transmits a powerful signal to interfere with communication on the discovered frequencies. Fortunately, this kind of attack is not very common because of the expense of acquiring hardware capable of launching jamming attacks and it leads to a lot of time and effort being expended merely to disable communications.

1.1 Contributions and Outline

This paper proposes a wireless intrusion detection system called WiFi Miner, with the following two objectives:

1. Eliminating the need for hard-to-get training data. This it does with a proposed Online Apriori-Infrequent algorithm, which does not use the confidence value parameter and does not create any rules, but efficiently uses only frequent and non-frequent patterns in a record to compute an anomaly score for the record to determine whether this record is anomalous or not on the fly.
2. Real-Time Detection of Intrusions: This our system does by integrating proprietary hardware sensors, where streams of wireless packets (e.g., Media Access Control or MAC frames) from Access Points (AP) are promptly captured and processed with the proposed Online Apriori-Infrequent algorithm. Our proposed Real-time Online Apriori-Infrequent algorithm improves the join and prune steps of the traditional Apriori algorithm, detects frequent and infrequent patterns in connection records, assigns anomaly scores to connection records without generating association rules from frequent patterns, and increases the efficiency and run times significantly. The proposed system targets mostly active, passive and main-in-the-middle wireless attacks, which are not easily detected by existing wired attacks.

Section 2 presents related work, Section 3 presents the proposed system: WiFi Miner, Section 4 describes the experimental results, while section 5 concludes the paper.

2. RELATED WORK

ADAM [4] is a wired Apriori based network intrusion detection system. First, ADAM collects normal, known frequent datasets through mining as training datasets. Secondly, during detection, it runs an on-line algorithm to find last frequent connections, which it compares with known mined training normal datasets and it discards those recent connections which seem to be normal. With suspicious records, it then uses a classifier, previously trained to classify and label suspicious connections as a known type of attack, unknown type of attack or a false alarm. The central theme of MADAMID [9] approach is to apply data mining programs to the extensively gathered audit data to compute models that accurately capture the actual behavior or patterns of intrusions and normal activities. In MADAMID they have used association and frequent episode rule for sequence analysis. Another research [12] presented an efficient algorithm called LERAD (Learning Rules for Anomaly Detection). Another important research in this field is MINDS [5], which uses a suite of data mining techniques to automatically detect attacks against computer networks and systems. In their research they presented two specific contributions: (1) an unsupervised anomaly detection technique that assigns a score to each network connection that reflects how anomalous the connection is, and (2) an association pattern analysis based module that summarizes those network connections that are ranked highly anomalous by the anomaly detection module. An Online K-means algorithm (KMO) was used in [20], where authors analyzed network traffic data streams collected and recorded from a WLAN system and detected all types of attack behaviors through data mining clustering technique. The log they used is specifically for wireless traffic and they extracted these data from several access points (APs). The main limitation of their approach was that they used training data which is hard to get. Another hybrid anomaly detection approach is proposed in [11], which uses association rule mining technique and cross fea-

Table 1: Example Database Records

TID	Items
1	A B D
2	A C E F
3	B C D F
4	A B C D
5	A B C E

ture mining to build normal behavior profiles of network activities for an individual node.

Data Mining Association Mining Related Algorithms Association rule can be used to find correlation among items in a given transaction. A well-known example is market basket analysis, which analyzes customer buying habits by finding associations between the different items that customers place in their shopping baskets. If most customers who buy milk also buy bread, we can put milk and bread in the same shelf to increase sales and profit. Association rule mining was proposed in [8], where the formal definition of the problem is presented as: Let L = $\{i_1, \ldots, i_n\}$ be a set of literals, called items. Let database, D be a set of transaction records, where each transaction T is a set of items such that $T \subseteq L$. Associated with each transaction is a unique identifier, called its transaction id (TID). We say that a transaction T contains X, a set of some items in L, if $X \subseteq L$. An association rule is an implication of the form $X \rightarrow Y$, where $X \subseteq L$, $Y \subseteq L$, and $X \cap Y = \emptyset$. The rule $X \rightarrow Y$ holds in the transaction set D with confidence c if c% of transactions in D that contain X also contain Y. The rule $X \rightarrow Y$ has support s in the transaction set D if s% of transactions in D contain $X \cup Y$. An example is shown in Table 1. Here, there are five transactions with TID 1, 2, 3, 4 and 5. Rule $\{A\} \rightarrow \{C\}$ is an association rule because with a given minimum support of 60% or 3 out of 5 transactions, the 2-itemset AC which, this rule is generated from, has a support of 4/5 or 80%. The confidence for this rule is 4/4=100%.

According to [8] and [1], the problem of mining association rules can be decomposed into the following two steps:
1) Discovering the large itemsets or frequent patterns, i.e., the sets of itemsets that have transaction support above a pre-determined minimum supports.
2) Using the large itemsets (frequent patterns) to generate association rules for the database that have confidence above a pre-determined minimum confidence.

Several important association rule mining algorithms including the Apriori [8], [1], [13] and Fp-growth [7], some of which are commonly used in network intrusion detection systems, exist. The basic idea behind the Apriori algorithm [8], [1], is to level-wise, use shorter frequent k-itemsets (L_k) to deduce longer frequent (k+1)-itemsets (L_{k+1}) starting from candidate 1-itemsets consisting of single items in the set L defined above, until either no more frequent itemsets or candidate itemsets can be found. Thus, the Apriori algorithm finds frequent k-itemsets L_k from the set of frequent (k-1)-itemsets L_{k-1} using the following two main steps involving Joining the L_k with L_k Apriori-gen way to generate candidate k-itemsets C_k, and secondly, pruning the C_k of itemsets not meeting the Apriori property or not having all their subsets frequent in previous large itemsets. To obtain the next frequent L_k from candidate C_k, the database has to be scanned for support counts of all itemsets in C_k. Another Apriori algorithm based algorithm, Signature-Apriori is proposed in [19], which analyzes the previously known signatures to find the signature of related attacks quickly. The only limitation of their system is that it is a misuse detection system and is unable to detect totally new types of attacks.

Since level-wise candidate generation as well as numerous scans of the database had been seen as a limitation of this approach, many optimization techniques of this approach had appeared in the literature and alternative tree-based solution proposal with Frequent pattern tree growth FP-growth [6], [7] had also been used. The FP-growth approach scans the database once to build the frequent header list, then, represents the database transaction records in descending order of support of the F_1 list so that these frequent transactions are used to construct the FP-tree. The FP-tree are now mined for frequent patterns recursively through conditional pattern base of the conditional FP-tree and suffix growing of the frequent patterns. Concepts of infrequent pattern computation and use of record anomaly scores computed from both frequent and infrequent patterns can also be applied with the efficient tree-based FP-tree algorithm for association pattern mining in application domains and could be explored in the future.

3. THE PROPOSED WIRELESS INTRUSION DETECTION SYSTEM

Section 3.1 presents definitions relevant to the proposed WiFi Miner IDS system, section 3.2 presents the overall WiFi Miner system architecture and algorithm, section 3.3 presents the Apriori-Infrequent algorithm used by the WiFi Miner system, while section 3.4 provides an example application of the Online Apriori-Infrequent algorithm.

3.1 Definitions and Properties

The following definitions and properties are used in the discussion of the proposed IDS system.

DEFINITION 3.1. A record has a maximal level of n*: if the record, R_i, has its largest frequent itemset being an n-itemset or containing* n *distinct items.* ■

DEFINITION 3.2. A maximal level n record has a set of frequent and infrequent itemsets*: consisting of all its 1-itemsets to n-itemsets that are frequent and infrequent respectively.* ■

DEFINITION 3.3. A Frequent k-itemset*: is a k-itemset which has support greater than or equal to the given minimum support with respect to the entire database stream of records.* ■

DEFINITION 3.4. An Infrequent k-itemset*: is a k-itemset which has support less than the given minimum support with respect to the entire database stream of records and has all its subsets frequent in levels* k-1 *and lower. This type of itemset is also called negative border in some work.* ■

DEFINITION 3.5. A maximal level n Record's Frequent Itemsets, F_R*: consists of the set of all its 1-itemsets to n-itemsets, which have supports greater than or equal to the*

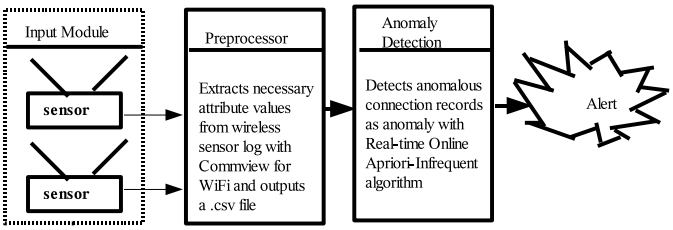


Figure 1: WiFi Miner Wireless IDS

given minimum support with respect to the entire database stream of records. ■

DEFINITION 3.6. A maximal level n Record's Infrequent Itemsets, IF_R*: consists of the set of all its 1-itemsets to n-itemsets, which have supports less than the given minimum support with respect to the entire database stream of records. All subsets of each level* k *Infrequent set are frequent in the levels* k - 1 *and lower.* ■

DEFINITION 3.7. A k-itemset Anomaly Score*: The anomaly score of a level* k *itemset is -k if the itemset is frequent but +k if the itemset is infrequent.* ■

DEFINITION 3.8. A Record's Anomaly Score*: The anomaly score of a maximal level* n *record is the sum of all its levels 1 to* n *frequent and infrequent itemsets' anomaly scores.* ■

PROPOSITION 3.1. A Normal/Anomalous Record Property*: A normal record has more frequent than infrequent itemsets and has a negative total record anomaly score, while an anomalous record has more infrequent than frequent itemsets and has a positive or zero total record anomaly score.* ■

3.2 The WiFi Miner Framework

The proposed WiFi Miner system framework comprises of three main modules. They are: Input Module, Preprocessor Module, and Anomaly Detection Module as shown in Figure 1. The proprietary Network Chemistry wireless hardware sensors [15] first need to be properly installed and configured before they can be used to capture wireless network packets. Installing the sensors entails installing both a sensor server and sensor client software systems and logging on to the sensor client software console system to initialize and configure the sensors. Input Module consisting of properly configured hardware sensors, collects network traffic data from hardware wireless sensors attached to the system, which capture data from airwaves as most of the wireless attacks may occur before data are in wired network and Access Points. The Preprocessor Module converts the raw data to readable format with the help of CommView for WiFi software, which is used to extract sensed data from the hardware sensor's firebird database and saved in a .csv file (csv stands for Comma Separated Values where attributes values are simple text separated by commas). With CommView, necessary features can be extracted for analyses to detect anomalies and extracted records stored as text file are processed directly by our WiFi Miner system. These records may also be logged into database tables for more offline processing and possible tracking of anomalous records. The focus of our approach is online processing, that is independent of training data. After the data are preprocessed, they are sent to the Anomaly Detection Module, which includes the core algorithm (Online Apriori-Infrequent) for finding infrequent patterns or anomalies.

The proposed Online Apriori-Infrequent algorithm contributes by

1. Providing a mechanism for computing the anomaly scores of a record, that is based on the relative sizes and numbers of infrequent and frequent itemsets contained in just this record without the need for hard-to-get training data. This is based on the premise that infrequent itemsets are likely anomalous as is the case with many wireless attacks.

2. Providing a smart-join mechanism that improves the Aprior-gen join step and prune steps when computing candidate itemsets, which speeds up infrequent and frequent pattern generations.

3. Providing a mechanism that eliminates the need to generate association rules from frequent patterns in order to detect anomalies.

Given a record, an anomaly score is computed from all its level *1* to level n patterns (both frequent and non-frequent patterns), where n is the largest number of items in the maximal frequent pattern as presented in the definitions. To compute the anomaly score of a record, each level k frequent pattern in the record is assigned an anomaly score of -k, while each level k infrequent pattern is assigned an anomaly score of +k, and the anomaly score of a record is the sum of the anomaly scores of all its frequent and infrequent patterns. If a record's total anomaly score becomes positive, then, this record has more infrequent than frequent patterns and is considered anomalous. On the other hand, if a record's anomaly score is negative, then, the record has more frequent than non-frequent patterns and is considered normal. If a record has zero anomaly score, it means it has the same number of frequent and infrequent patterns, and for increased security, the proposed system treats such a record as anomalous since it is safer to have a false alarm than harmful undetected intrusion. This anomaly detection module generates anomaly alerts for records with positive anomaly scores. The simple logic behind anomaly score weight assignment to frequent and infrequent itemsets is that the more the number of items in an infrequent itemset, the lower the chances of this itemset being in an arbitrary record. Thus, the presence of an infrequent 3-itemset is more rare than the presence of an infrequent 2-itemset in a record. Therefore, the anomaly weights of infrequent itemsets are proportionately increased with their size levels, while those of frequent itemsets are decreased with increasing number of items in the itemset. For example, while an infrequent 2-itemset like AC would have an anomaly score of +2, a frequent 2-itemset like AF would have anomaly score of -2. However, an infrequent 3-itemset would have an anomaly score of +3, while a frequent 3-itemset would have an anomaly score of -3.

The WiFi Miner algorithm is presented as Algorithm 1. The proposed scheme finds anomaly/infrequent patterns without training classifiers offline with safe data. Instead of finding frequent patterns at first and then comparing these patterns with incoming data to detect the anomalies during third step, our method finds the infrequent data/anomalies

during the first step with an online Apriori-Infrequent algorithm, which tries to find both infrequent patterns and frequent patterns, improves candidate set generation scheme in one step by improving the runtime complexity of Joining and Pruning. The rest of the section describes both the Online Apriori-Infrequent algorithm and the Anomaly scoring scheme adopted by the proposed WiFi Miner system.

ALGORITHM 1. *(WiFi Miner: Wireless IDS)*

Algorithm WiFi Miner()
Input: *Network connection packets (P), sensors (S), access points (AP)*
Output: *Anomalous connections (A)*
begin
While (true)
(1) Capture wireless packets from AP using sensors (S)
(2) Extract connection packets (P) from sensors S with Commview for WiFi software and save as .csv file
(3) Call Apriori-Infrequent Algorithm with "Incoming-connection" .csv file records as input and output anomalous records as alerts.
end

3.3 The Proposed Apriori-Infrequent Algorithm

The goal of the Apriori-Infrequent Algorithm is to generate all frequent patterns as well as all infrequent patterns at every level, and be able to use this knowledge to compute anomaly scores for records. In order to compute frequent and non-frequent itemsets efficiently, the proposed algorithm argues that the Apriori's method for computing candidate (i+1)-itemsets by joining all frequent i-itemsets (L_i) with themselves, if their first (i - 1) items are the same and the first itemset comes before the second itemset in the L_i list, can be improved on, with a third condition. The third join condition introduced by the Apriori-Infrequent algorithm states that an itemset in the L_i list will only be used to join other items in the L_i list that meet the first two conditions if this itemset's last item (or ith item) appears in a joinable item list called Z list, consisting of all (i-1)th item of L_i. The purpose of the Z list is to prevent ahead of time, the need to join itemsets which produce itemset results that have no chance of being frequent because their subsets are not frequent. Such itemsets in the Apriori algorithm are pruned during this step but we avoid both creating them in the first place, computing their subsets and pruning them. Our algorithm looks for infrequent patterns (which were frequent in the previous level but when they are combined with some other attributes, they become infrequent). These infrequent itemsets are similar to negative borders [13], but is computed in a more efficient fashion in our online Apriori algorithm. This concept of fast detection of infrequent pattern is useful for intrusion detection domain because suppose for example, in connection record, Flag ACK (ACKnowledgment) is frequent but when ACK is combined with Flag SYN (SYNchronized), it may be an attack. The formal Apriori-Infrequent algorithm is given as Algorithm 3 and the Smart-Join technique it uses is also given as Algorithm 2.

ALGORITHM 2. *(Apriori-SmartJoin:Computing Candidate C_k from L_{k-1})*

Algorithm Apriori-SmartJoin()
Input: *A list of large (k-1)-itemsets: L_{k-1},*
Output: *A list of candidate k-itemsets: C_k,*
Other variables: *Z-list for smart join*
begin
$C_k = \emptyset$
Z = the set of all (k-2)th item in L_{k-1}.
For each pair of itemsets M and P $\in L_{k-1}$ do
begin
M joins with P to get itemset $M \cup P$
if the following conditions are satisfied.
(a) itemset M comes before itemset P in L_{k-1}
(b) the first k-2 items in M and P (excluding just the last item) are the same.
(c) the last item (or (k-1)th item) of each itemset in L_{k-1} is joinable only if this item is in the Z list
If M and P are joinable then
$C_k = C_k \cup M \cup P$
end
end

ALGORITHM 3. *(Apriori-Infrequent:Computing Infrequent Patterns)*

Algorithm Apriori-Infrequent()
Input: *A list of candidate itemsets: C_1, minimum support count λ*
Output: *A list of frequent itemsets: L, Anomaly score for each record.*
Other Variables: *A list of Infrequent itemsets: S,*
begin
k = 1
1.Compute frequent L_k and infrequent S_k with minimum support λ from C_k.
2. While ($L_k \neq \emptyset$) do
begin
2.1. k = k+1
2.2. Compute the next candidate set C_k from L_{k-1} as L_{k-1} Apriori-smart join L_{k-1}.
2.3. For each itemset in C_k do
2.3.1. Calculate all possible subsets and prune if not previously large.
2.4.If $C_k = \emptyset$ then break and go to step 3
2.5.Compute frequent L_k and infrequent S_k with minimum support λ from C_k.
2.6.Update Anomaly Score for Connection Record by calling Anomaly Score function with L_k and infrequent S_k
end
3.Compute all Frequent patterns as $L = L_1 \cup \ldots L_k$
end

Anomaly Score Calculation

The proposed WiFi Miner system is able to calculate or give each connection packet an anomaly score on the fly. This is an important step as it eliminates the need to generate association rules from frequent patterns as done by many existing approaches in order to identify intrusions. The simple anomaly score rule assigns a positive anomaly score of $+n$ to every n-itemset infrequent pattern in a record that is equal to the number of items in the infrequent pattern but assigns a negative anomaly score of -n to a frequent pattern with n items. This rule is based on the premise that certain anomalies are infrequent events that embed themselves in frequent or normal packets. The anomaly score of each database transaction is computed in parallel with support counting of each level candidate set of the Apriori-Infrequent algorithm and this utilizes the records while they are still in memory without incurring additional I/O costs. Thus, the total anomaly score of a record is computed as the sum of all the anomaly scores of this record's itemset level *1* to level

Table 2: Database Records Anomaly Scores

TID	Items	Anomaly Score		
		Pass 1	Pass 2	Final Score
1	A B D	-3+0 = -3	-4+2 = -2	-5
2	A C E F	-2+2 = 0	-2+10 = 8	+8
3	B C D F	-3+1 = -2	-4+8 = 4	+2
4	A B C D	-4+0 = -4	-8+4 = -4	-8
5	A B C E	-3+1 = -2	-6+6 = 0	-2

n frequent and infrequent patterns, where n is the last non-empty level of frequent patterns for the record. A record is declared anomalous if its total anomaly score is zero or positive but normal if its total anomaly score is negative.

3.4 An Application of the Apriori-Infrequent and Anomaly Score

Assume that wireless network connection records were captured and preprocessed to produce a database transaction table similar to columns one and two of Table 2, with candidate 1-items as {A, B, C, D, E, F}. In pre-processed wireless packets or records, the attributes depicted as A to F above would represent connection features like: connection date and time, source and Destination MAC address, packet size in bytes, access point MAC address(BSSID), Frame Type/Subtype, transmission rate, Client/AP sequence number, signal power, access point name, source type (station or access point), channel, etc.

Example 1: Using the WiFi Miner Apriori-Infrequent and Anomaly score counting technique, identify the anomaly or alert records from Table 2 (first two columns) if the minimum support threshold is 60% or 3 out of 5 transactions.

Solution 1: Applying Algorithm 3, C_1 = {A:4, B:4, C:4, D:3, E:2, F:2}, and L_1 = {A, B, C, D} with anomaly score each of -1 and S_1 = {E, F} with anomaly score each of +1. The anomaly scores of the transactions in the database table are computed at this level as: TID 1, ABD has an anomaly score of -1(A) -1(B) -1(D) = -3. TID 2, ACEF has an anomaly score of -1(A) -1(C) +1(E) +1(F) = 0. The anomaly scores of transactions 3, 4 and 5 are respectively: -2, -4, and -2. Next, we compute C_2 as L_1 Apriori-gen join L_1 since the Z list at this level is still empty set. Thus, C_2 = {AB:3, AC:3, AD:2, BC:3, BD:3, CD:2}. L_2 is computed as {AB, AC, BC, BD} with anomaly score of -2 each, while S_2 is computed as {AD,CD} with anomaly score of +2 each. The anomaly scores of the database transactions are updated as: Tid 1 (ABD) = -3(score from previous step) -2(AB) +2(AD) -2(BD) = -5. Tid 2 (ACEF) = 0(score from previous step) - 2(AC) +2(AE) +2(AF) +2(CE) +2(CF) +2(EF) = +8. The rest of the anomaly scores are updated as shown in column 3 of Table 2. During iteration 3, to create C_3 list, the Z list is first created from L_2 as item (2 -1) or the first item in each L_2 itemset. Thus, Z = {A, B}. To join an L_2 itemset, if the last element of the itemset is not in the Z list, then, we should not perform the join. This means that we first reduce our L_2={AB, AC, AD, BC, BD, CD} to {AB} since AC, AD, BC, BD and CD do not have their last elements in the Z list. Thus, our C_3 = {AB} Apriori-gen join {AB} = $\emptyset$. Since $C_3 = \emptyset$ as well as $L_3 = \emptyset$, the algorithm ends without computing the anomaly score for this iteration. All records with negative anomaly scores are normal while those with positive or zero anomaly scores are alerts. The final anomaly scores of the example connection records are as given in column 3 of Table 2.

Table 3: Attack Signatures Used

Attack Name	Attack Signature Used
SYN Flood	flag = S, dest-host = victim (same), dest-service = vulnerable port (same)
UDP Flood	dst-host = victim (same), dst-service = vulnerable port/random port
Port Scanning	(flag = S, src-host = attacking machine, dst-service = vulnerable port) (flag = R, dest-host = attacking machine, src-service = dest-vulnerable port)

4. EXPERIMENTS AND PERFORMANCE ANALYSIS

To test the proposed system prototype, we installed Network Chemistry sensors in one PC from where we scanned all APs in ranges and selected the AP for our wireless network and started capturing packets from our Access Point. We created a wireless network with two other PCs where one was the victim and another one was the attacker PC. Within 5 minutes time window we have captured around 19,500 wireless packets, which were generated as a result of some innocent activities. Then we gathered around 500 anomalous packets which contained different kinds of crafted attacks like passive attacks (packets used for WEP cracking and packets used in Port scanning attack), active attacks (packets used for SYN Flood/ UDP Flood Attack), Man-In-the-Middle attack (packets used for establishing a rogue AP). Then, we launched these attacks from the attacker PC to the victim PC. Description of how we gathered these anomalous packets is provided next. Attack signatures used for these attacks are summarized in Table 3.

To crack a WEP (Wired Equivalent Privacy) key, at first, we spoofed a client's MAC address, which was known from previously generated innocent packets. Using this spoofed MAC address, we generated fake ARP packets and sent these packets to the AP (Access Point) using Aireplay [3]. In response to these fake ARP packets, the AP sent back reply packets, which were captured and used by Aircrack [2] to decrypt the WEP key. Our sensors captured all these fake ARP packets and these packets were considered and gathered as anomalous packets. Attack packets for SYN Flood attack, UDP Flood attack and Port Scanning attack can be created with tools like Engage Packet Builder [16]. To gather attack packets for Man-In-the-Middle type of attack, we set up a rogue AP with the same SSID (Service Set Identifier) as the legitimate one in a place nearer than the legitimate AP. Then, using the spoofed client's MAC address we sent de-authentication packets using Aireplay. As a result, the targeted client is disconnected from the legitimate AP and is connected to the rogue AP because of the stronger signal. These de-authentication packets were captured and gathered as anomalous packets. Then, we tested these combined dataset with our system WiFi Miner to ver-

ify the detection rate and total runtime. We also tested this input dataset with traditional Apriori based systems like ADAM and Snort Wireless to get a comparative performance analysis of our WiFi Miner system with existing mostly wired IDSs.

At first, we have compared the runtime of our algorithm: Real-time Online Apriori-Infrequent Algorithm with traditional Apriori algorithm concept used in ADAM and noticed an around 35% increase in execution time efficiency in our algorithm as shown in Figure 2. This is because we are not generating association rules with confidence value and also we have improved the join and prune sections of the algorithm with our Smart-Join approach. It should be stated here that from analysis and experiments, the proposed Online Apriori with smart join produces complete and correct frequent and infrequent patterns as the regular Apriori algorithm given the same datasets.

After we collected these 500 anomalous packets, then, we tested the combined (anomalous + innocent) dataset with our system WiFi Miner to verify the detection rate and total runtime. We also tested this input dataset with traditional Apriori based system like ADAM and Snort Wireless to get a comparative view of our system with existing ones. At first, we have compared the runtime of our algorithm: Real-time Online Apriori-Infrequent Algorithm with traditional Apriori algorithm concept used in ADAM and noticed an around 35% increase in execution time efficiency in our algorithm as shown in Figure 2. This is because our WiFi Miner system is not generating association rules with confidence value and also has improved the join and prune sections of the Apriori technique with a more efficient Smart-Join approach while still keeping the algorithm simple. We used around 19,500 innocent wireless packets along with 500 anomalous attack packets and tested them in WiFi Miner, Snort Wireless and Apriori based system in ADAM to see how many anomalous packets get detected in all three systems. We also calculated the false alarms produced by each system. In our testing model we had 500 anomalous packets. So, after each system flags connection packets as anomalous, we verify if that packet belongs to the class of our 500 anomalous packets. If the packet is not an anomalous packet, then it is counted as a false alarm. The total attack detection rate and false alarm comparative analysis of all systems is given in Table 4 and Figure 3 while a more detailed analysis of their detection of specific classes of attacks is provided in Table 5 and Figure 4. It can be seen from the tables that the proposed WiFi Miner system consistently detects more attacks than both Snort Wireless and ADAM in all categories of attacks. The proposed WiFi Miner system also records the lowest amount of false alarms.

Currently, the proposed WiFi Miner system has no mechanism for detecting Jamming Wireless attacks. Also, if the minimum support is set too low, there may be large number of frequent itemsets and fewer infrequent itemsets. As a result, attacks may go undetected. Experiments show that for this wireless intrusion detection domain, a good choice of minimum support is 60% or more. Future work should explore improving efficiency of the system, handling more types of attacks and further reduction of false alarms.

5. CONCLUSIONS AND FUTURE WORK

This paper proposes a wireless intrusion detection system: WiFi Miner, which uses Apriori-Infrequent based algorithm

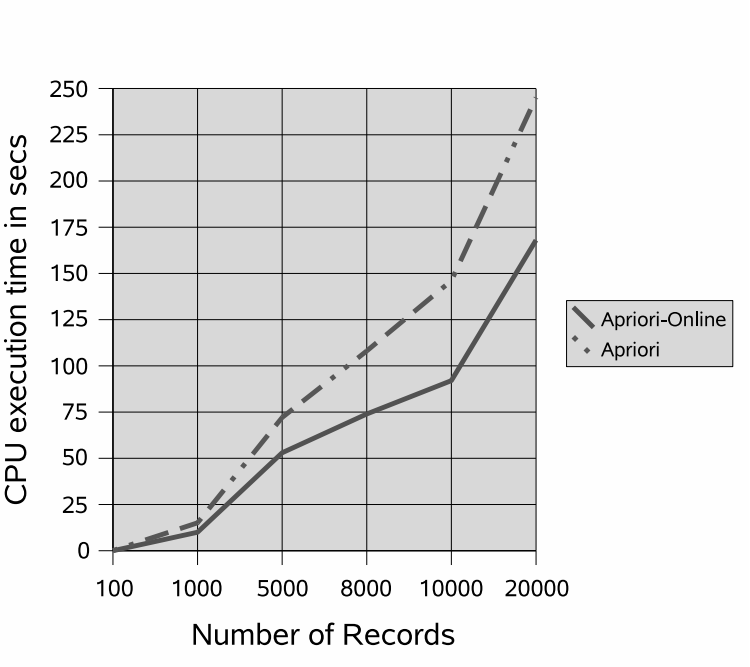


Figure 2: Comparison of Apriori-Infrequent with Apriori Algorithm

Table 4: Attacks Detected and False Alarm Comparison

Detected	(Out of 500 attacks)		
	WiFi Miner	Snort Wireless	ADAM
Attacks Detected	433	335	377
False Alarm	180	292	248

Table 5: Specific Attacks Detected Comparisons

Detected	3 Algorithms		
	WiFi Miner	Snort Wireless	ADAM
Passive Attacks (200 attacks)	179 (89.5%)	138 (69%)	161 (80.5%)
Active Attacks (200 attacks)	171 (85.5%)	145 (72.5%)	151 (75.5%)
Man-In-Middle (100 attacks)	83 (83%)	52 (52%)	65 (65%)

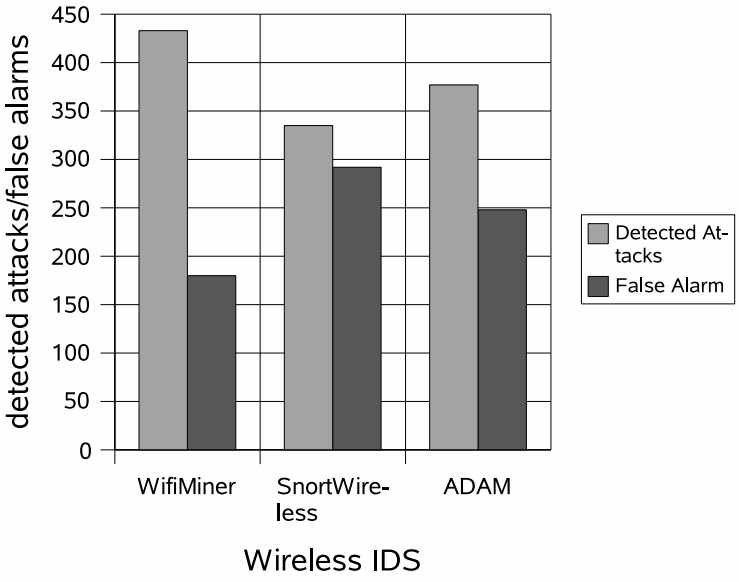


Figure 3: Comparisons of WiFiMiner with SnortWireless and ADAM

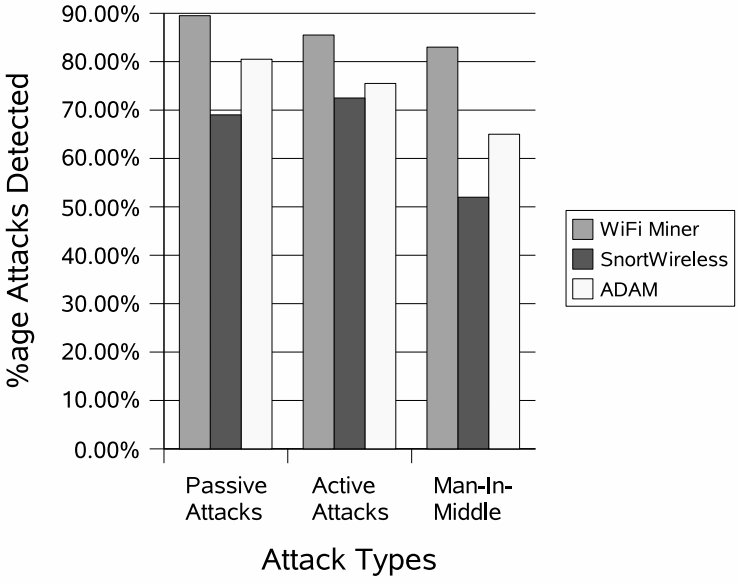


Figure 4: Specific Attacks Comparisons of WiFiMiner with SnortWireless and ADAM

to detect infrequent patterns, then our algorithm designed for Anomaly Score Calculation, assigns a score to each wireless packet. Positive or zero anomaly score in a specific connection record means that more infrequent/anomalous patterns are found in that record than frequent patterns while a negative anomaly score indicates a normal packet. We have also used proprietary Network Chemistry hardware sensors to capture real-time traffic in order to improve intrusion response time. Our system is different from existing wireless intrusion systems, since it eliminates the need for hard-to-get training data and detects intrusions in real time. Also, like other existing wireless intrusion systems, it captures the packets from airwaves while wired IDSs use net-flow data from routers. Thus, the major contribution of our system is that it can detect anomalous packets in real time without any training phase. We have tested our system with crafted intrusions and compared it with other two systems and found our system to be more efficient. Another major contribution is that we have introduced Smart-Join, which is an improved version of Join and Pruning steps in original Apriori algorithm.

In the future, we plan to enhance our system to work with many access points, currently it is capable of handling wireless connection records from one access point although our sensors are capable of finding all APs in their ranges. We are also working towards making our system generalized so that it can be used for both wired and wireless intrusion detection. Other future work include applying this online intrusion detection system approach to other domains like environment pollution monitoring systems where excessive levels of pollution can quickly raise alerts as anomalies from sensor captured data.

6. ACKNOWLEDGMENTS

This research was supported by the Natural Science and Engineering Research Council (NSERC) of Canada under an operating grant (OGP-0194134) and a University of Windsor grant.

7. REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on very Large Databases Santiago, Chile*, pages 487–499, 1994.

[2] Aircrack. Airdump web page. http://airdump.net/papers/packet-injection-windows, 2007.

[3] Aireplay. Airdump web page. http://airdump.net/papers/packet-injection-windows, 2007.

[4] D. Barbara, J. Couto, S. Jadodia, and N. Wu. Adam: A testbed for exploring the use of data mining in intrusion detection. *ACM SIGMOD RECORD: Special Selection on Data Mining for Intrusion Detection and Threat Analysis*, 30(4), 2001.

[5] L. Ertoz, E. Eilertson, A. Lazarevic, P. Tan, J. Srivastava, V. Kumar, and P. Dokas. *The MINDS - Minnesota Intrusion Detection System in Next Generation Data Mining*, chapter 3. MINDs, 2004.

[6] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, New York, 2000.

[7] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *International Journal of Data Mining and Knowledge Discovery*, 8(1):53–87, Jan 2004.

[8] T. Imielinski, A. Swami, and R. Agarwal. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD conference on management of data*, pages 207 – 216. ACM, 1993.

[9] W. Lee and S. J. Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM Transaction on Information and System Security*, 3(4):227–261, Nov. 2000.

[10] Q.-H. Li, J.-J. Xiong, and H.-B. Yang. An efficient algorithm for frequent pattern in intrusion detection. In *Proceedings of the International Conference on Machine learning and cybernatics*, pages 138–142, Nov. 2003.

[11] Y. Liu, Y. Li, H. Man, and W. Jiang. A hybrid data mining anomaly detection technique in ad hoc networks. *International Journal of Wireless and Mobile Computing*, 2(1):37–46, 2007.

[12] V. Mahoney and P. K. Chan. Learning rules for anomaly detection of hostile network traffic. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM)*, pages 601 – 604. IEEE, 2003.

[13] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *International Journal of Data Mining and Knowledge Discovery*, 1(3):241–258, Jan 2004.

[14] V. Marinova-Boncheva. Applying a data mining method for intrusion detection. In *ACM International Conference Proceeding Series*. ACM, June 2007.

[15] NetworkChemistry. Network chemistry wireless security business. http://www.networkchemistry.com, 2007.

[16] E. Security. Engage security web page. http://www.engagesecurity.com, 2007.

[17] R. J. Shimonski. Wireless attacks primer. A whitepaper published on windowssecurity.com section: Articles: Wireless security, July 2004.

[18] K. Yoshida. Entropy based intrusion detection. In *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, PACRIM*, pages 28–30. IEEE, August 2003.

[19] H. Zhengbing, L. Zhitang, and W. Junqi. A novel intrusion detection system (nids) based on signature search of data mining. In *1st international conference on Forensic applications and techniques in telecommunications, information, and multimedia and workshop*, January 2008.

[20] S. Zhong, T. Khoshgoftaar, and S. Nath. A clustering approach to wireless network intrusion detection. In *Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 190–196. IEEE, 2005.

Mobile Visualization for Sensory Data Stream Mining

Pari Delir Haghighi, Brett Gillick, Shonali Krishnaswamy, Mohamed Medhat Gaber, Arkady Zaslavsky
Centre for Distributed Systems and Software Engineering
Monash University, Caulfield Campus
900 Dandenong Rd, Caulfield East, VIC - 3145, Australia
Tel: IDD + 61 3 9903 1402, Fax: IDD + 61 3 9903 1077
{Pari.DelirHaghighi, Brett.Gillick, Shonali.Krishnaswamy, Mohamed.Gaber, Arkady.Zaslavsky}@infotech.monash.edu.au

ABSTRACT
With the emergence of ubiquitous data mining and recent advances in mobile communications, there is a need for visualization techniques to enhance the user-interactions, real-time decision making and comprehension of the results of mining algorithms. In this paper we propose a novel architecture for situation-aware adaptive visualization that applies intelligent visualization techniques to data stream mining of sensory data. The proposed architecture incorporates fuzzy logic principles for modeling and reasoning about context/situations and performs gradual adaptation of data mining and visualization parameters according to the occurring situations. A prototype of the architecture is implemented and described in the paper through a real-world scenario in the area of healthcare monitoring.

Categories and Subject Descriptors
H.5.2 [User Interfaces].

General Terms
Design, Experimentation.

Keywords
Data Stream Mining, Fuzzy logic, Context-aware, Visualization.

1. INTRODUCTION
There is a range of emerging applications that use mobile devices for data analysis and processing of sensory data. Examples of such applications include mobile fieldworkers (firefighters, healthcare professionals, salespeople, police, etc.), intrusion detection and Intelligent Transportation Systems. A very significant problem in these applications is that it is imperative that the vast amounts of sensory generated data need to be processed, analyzed and displayed on mobile devices such as a PDA in a smart, cost-efficient way to leverage the benefits that this type of technology provides.

Ubiquitous Data Stream Mining (UDM) [11] is the process of analyzing data emanating from distributed and heterogeneous sources and sensors with mobile devices and has the potential to perform real-time analysis of data streams onboard resource-constrained devices. However to perform intelligent and cost-efficient analysis of data and visualization of the mining results, it is imperative for adaptation strategies to be autonomous and agile with respect to current context of the running application and to factor in contextual/situational information.

As a meta-level concept over context we define the notion of a situation that is inferred from contextual information [25]. Situation-awareness provides applications with a more general and abstract view of their environment rather than focusing on individual pieces of context. Situation-aware adaptation of data stream mining and visualization parameters enhances streaming and visualization tasks and enables continuity and consistency of the running operations.

In this paper we introduce an architecture that integrates situation-aware adaptive processing (SAAP) of data streams mining with mobile visualization (MobileVis). Situation-awareness is achieved using a Fuzzy Situation Inference (FSI) model that combines fuzzy logic principles with the Context Spaces (CS) model [25], a formal and general context modeling and reasoning approach. SAAP provides dynamic adaptation of data stream processing parameters in real-time according to the changes of context. MobileVis applies situation-aware visualization techniques to the data stream mining process in a mobile environment and enhances the ability of users to make 'on the move' decisions. The decision making process occurs in real-time as streamed data, which is processed by a UDM algorithm, arrives and subsequently visualized according to the current context and situation [15].

Our proposed architecture is generic and could be used in a range of applications. The prototype system that we have developed focuses on the area of mobile healthcare. Mobile health monitoring applications are recently gaining popularity among practitioners and patients as they provide a convenient and affordable way to monitor vital signs remotely and generate warnings and emergency calls. Data collected from wireless biosensors are processed and analyzed locally on a mobile device (i.e. a smart phone or a PDA) or transferred to a central server for further analysis. One of the main biosensors used for monitoring heart patients is ECG (Electrocardiogram) sensors that send ECG data as a continuous data stream to the mobile device. Real-time monitoring of heart beats and other vital signs on a mobile device provides a safe and noninvasive way for early detection of chronic diseases [22, 29].

This paper is structured as follows: Section 2 describes the architecture for situation-aware adaptive visualization. Section 3 discusses the SAAP component that provides situation-awareness and adaptation of data stream mining parameters. Section 4 describes the MobileVis (Mobile Visualization) framework. Section 5 discusses the implementation and evaluation of the architecture. Section 6 reviews the related work and finally section 7 concludes the paper and discusses the future work.

2. AN ARCHITECTURE FOR SITUATION-AWARE ADAPTIVE VISUALIZATION

In this section, we introduce an architecture for situation-aware adaptive visualization of sensory data. The architecture consists of two main components as shown in Figure 1. The first component is Situation-Aware Adaptive Processing (SAAP) of data streams that consists of two engines including Fuzzy Situation Inference Engine (FSIE) and Adaptation Strategy Engine (ASE). FSIE provides situation-awareness using fuzzy logic principles and includes three subcomponents including fuzzifier, rules and situation inference. The Adaptation Strategy Engine (ASE) is responsible for adjusting data stream processing parameters according to the occurring situations. The second component of the architecture is MobileVis (Mobile Visualization), a framework which provides adaptive visualization techniques to mobile applications.

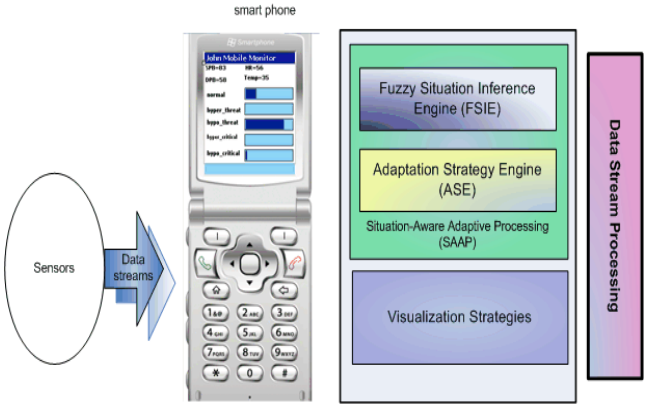


Figure 1. An Architecture for situation-aware adaptive visualization for Ubiquitous Data Mining

The next section describes the two engines of FSIE and ASE for situation-aware adaptive processing (SAAP) of sensory data.

3. SITUATION-AWARE ADAPTIVE PROCESSING (SAAP) OF DATA STREAMS

3.1 Fuzzy Situation Inference Engine (FSIE)

The Fuzzy Situation Inference (FSI) model integrates fuzzy logic principles into the Context Spaces (CS) model [25]. The CS model is a formal and general context reasoning and modeling specifically developed for pervasive computing environments that addresses inaccuracies of sensory originated information. The FSI technique incorporates the CS model's underlying theoretical basis [26] for supporting context-aware and pervasive computing environments while using fuzzy logic to model and reason about vague and uncertain situations.

Fuzziness is defined in [32] as uncertainty and vagueness related to description of the semantic meaning of events and phenomena. To enable situation-awareness in pervasive applications, it is imperative to address the issue of uncertainty [3]. The CS deals with uncertainty mainly associated with sensors' inaccuracies. Yet there is another aspect of uncertainty in human concepts and real-world situations that needs to be represented in a context model and reflected in the results of situation reasoning. Fuzzy logic has the benefit of representing this level of uncertainty using membership degree of values.

The FSIE consists of three subcomponents of fuzzifier, rules and situation inference that are discussed in the following subsections.

3.1.1 Fuzzifier

In FSIE, crisp inputs are context attribute values such as temperature degree or light level that are obtainable by the application. Fuzzifier, as a software component, uses membership functions to map crisp inputs (i.e. context attribute values) into fuzzy sets. Prior to fuzzification, we define linguistic variables and their terms and then use a trapezoidal membership function for mapping crisp inputs into fuzzy sets.

Linguistic variables [32] are defined for each context attribute that is used in the situation reasoning process. These linguistic variables are then broken into terms or fuzzy variables. Each term of a linguistic variable is characterized by a fuzzy set. An input x is related to a fuzzy set A by a membership function μ and the relation is denoted as $\mu_A(x)$. A membership function maps the input x to a membership grade between 0 and 1 [18, 33].

3.1.2 Rules

In FSIE, rules represent situations of interest and each rule consists of multiple conditions/antecedents joined with the AND operator but a condition can itself be a disjunction of conditions. An example of a FSI rule for a health-related situation considering the variables of systolic and diastolic blood pressure (SBP and DBP) and heart rate (HR) can be expressed as follows:

Rule1: if SBP is 'high' and DBP is 'high' and HR is 'fast' then situation is 'hypertension'

In many cases, there are certain variables that are more important than others in describing a situation. For example, low blood pressure is a strong indication of 'hypotension' in a person while heart rate may not be equally important. To model the importance of variables and conditions, we assign a pre-defined weight *w* to each condition with a value ranging between 0 and 1. The sum of weights is 1 per rule. A weight represents the importance of its assigned condition relative to other conditions in defining a situation.

3.1.3 Situation Inference

To interpret a fuzzy rule we need to evaluate its antecedents and produce a single output that determines the membership degree of the consequent. We use the function maximum for the conditions joined with the OR operator; however, we apply the weighted sum function [26] to evaluate the conditions joined with the AND operator. The situation inference technique that we use to evaluate the rule and compute the confidence level in the occurrence of a situation is expressed as follows.

$$Confidence = \sum_{i=1}^{n} w_i \mu(x_i)$$

where $\mu(x_i)$ denotes the membership degree of the element x_i and w_i represents a weight assigned to a condition (discussed in

subsection 3.1.2). The result of $\mu(x_i)w_i$ represents a weighted membership degree of x_i for the condition.

Table 1 shows an example of computing the level of confidence in the occurrence of 'hypertension' (defined as Rule 1 in subsection 3.1.2) based on the input values of SBP, DBP and HR. Membership degree of variables are computed using a trapezoidal membership function considering the pre-defined thresholds for each variable.

Table 1. An Example of Evaluation of Rule 1

Antecedent	**Input Value**	**Weight**	**Weighted Membership Degree**
1: SBP is 'high'	129 mm Hg	0.35	0.9*0.35=0.315
2: DBP is 'high''	93 mm Hg	0.4	0.8*0.4=0.32
3: HR is 'fast'	102 bpm	0.25	1*0.25=0.25
Confidence= 0.315+0.32+0.25=0.885			

The result of rule evaluation is applied to the consequent and the output determines a situation's membership degree $\mu_{s_i}(x)$ that suggests the level of confidence in the occurrence of the situation. If the output of a rule evaluation for the 'hypertension' situation yields a membership degree of 0.885, we can suggest that the level of confidence in the occurrence of hypertension is 0.885. The level of confidence can be compared to a confidence threshold ε between 0 and 1 (i.e. predefined by the application's designers) to determine whether a situation is occurring [26] as follows.

$$\mu_{s_i}(x) \geq \varepsilon$$

The output of the FSIE component consists of fuzzy situations that represent the current situation. These results are used as an input to the ASE for adjustments of data stream processing parameters. The following section describes the adaptation strategy engine.

3.2 Adaptation Strategy Engine (ASE)

The Adaptation Strategy Engine (ASE) is responsible for gradual tuning and adjusting data stream processing parameters in real-time according to the results of situation inference from the FSIE. These results are multiple situations with different level of confidence.

Light weight algorithms such as LWC and RA-Cluster [12, 27] are the data stream mining algorithms that their certain parameters can be adjusted according to memory availability, battery charge or CPU utilization. These parameters control output, input or the process of the algorithm. The LWC (LightWeight Clustering) algorithm is based on the AOG (Algorithm Output Granularity) [10] approach that controls the output of the data stream mining according to the available memory. LWC considers a threshold distance measure for clustering of data. Increasing this threshold discourages forming of new clusters and in turn reduces resource consumption. We have used the concepts of AOG for situation-aware adaptation by adjusting data stream mining parameters according to the occurring situations.

To provide a fine-grained adaptation and reflecting the level of confidence of each situation in the adaptation process, we use a weighted average function for adjusting the parameter value that reflects all the results of situation inference in the adaptation of parameter values. The function format is as follows.

$$\hat{p}_j = \sum_{i=1}^{n} \mu_i p_j / \sum_{i=1}^{n} \mu_i$$

where p_j represents the set value of a parameter, μ_i denotes the membership degree of situation S_i where $1 \leq i \leq n$ and n represents the number of pre-defined situations, and $\hat{p}_j$ represents aggregated value of the parameter. The next section discusses visualization of fuzzy situations and the output (i.e. clusters) of data stream mining.

4. MOBILEVIS (MOBILE VISUALIZATION)

Creating a visualization that takes advantage of the human vision system capabilities can provide a user with insights into, and interpretation of, the data being presented [4, 21]. The use of an appropriate visualization technique gives a mobile user the ability to make 'on the move' decisions. A visualization which presents a user in a simple, uncluttered manner can decrease the time taken for a user to interpret data mining results. The increasing processing power and graphical capabilities of mobile devices means that visualization operations can be carried out entirely on the device. The MobileVis framework, seen in Figure 2, consists of three modules of Visualization Builder (VB), Adaptive Visualization Controller (AVC) and Graphics Abstraction Layer (GAL).

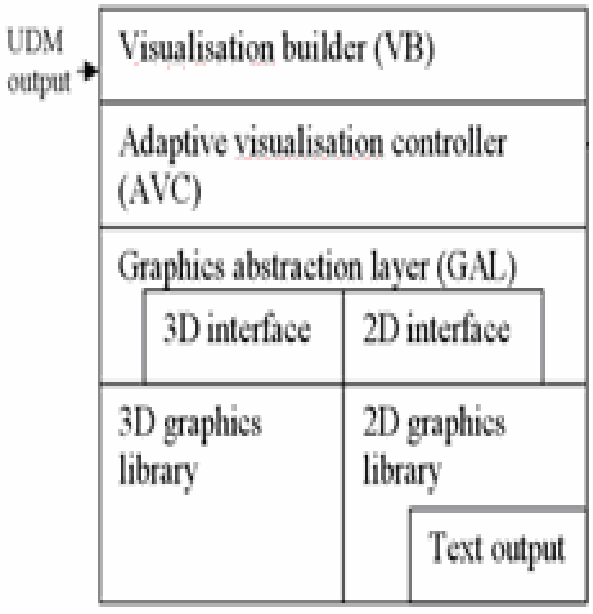


Figure 2. MobileVis framework

The visualization builder (VB) module of the framework takes as its input, the set of results generated by the UDM algorithm. The VB creates the necessary graphical components, such as text labels or lines on a chart, which are passed through the adaptive visualization controller. The adaptive visualization controller (AVC) is an autonomous process which automatically adapts the visualization to the current context/situation.

Due to the pluggable nature of the MobileVis framework, the monitoring data from SAAP and the output of data stream mining using algorithms such as LWC are used as an input to the VB component of the framework to drive the visualization. The next section discusses the implementation and evaluation of our architecture.

5. IMPLEMENTATION AND EVALUATION

A prototype of the situation-aware adaptive visualization is implemented in J2ME to represent different real-world scenarios in the area of healthcare monitoring. The prototype incorporates the SAAP and MobileVis frameworks and uses the LWC algorithm for data stream mining. Figure 3 shows the results of fuzzy situation inference on the emulator. Figure 4 shows similar results on a Nokia N95 mobile phone. The figure displays the results in real-time for situations of 'normal', 'Pre-Hypotension', 'Hypotension', 'Hypertension' and '"Pre-Hypertension' based on contextual information of SBP, DBP, HR and room temperature. Status bars show level of confidence and certainty in the occurrence of each situation.

The results of fuzzy situation inference are used for adaptation of the threshold parameter, which is one of the adjustable parameters of the LWC algorithm and controls the output of the data mining algorithms (i.e. clusters).

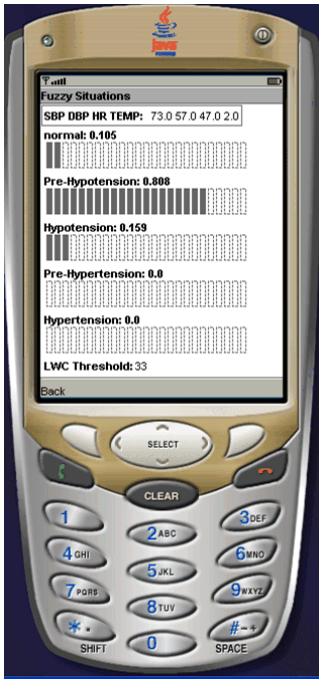


Figure 3. Fuzzy situation inference on the emulator

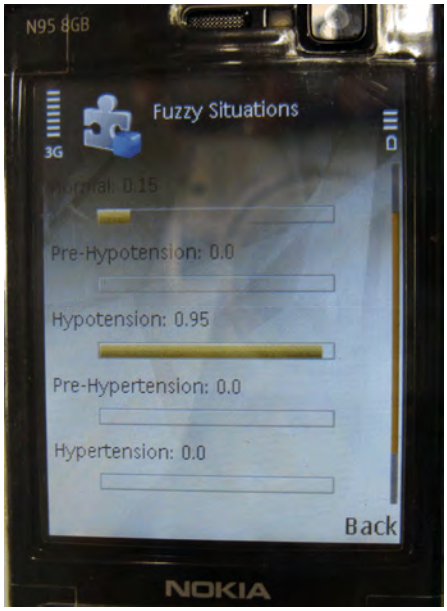


Figure 4. Fuzzy situation inference results on Nokia N95

The visualization of the inferred situation uses a simple image of a body which is coloured according to the threat level of the situation (See Figure 5). For the normal situation, the image is coloured green (top-left image in Figure 5), for a threat situation the image is coloured orange (top-right image in Figure 5), while in the hypertension and hypotension situations the image is coloured red (bottom-left image in Figure 5). The colour change of the image gives users an 'at a glance' representation of their current health situation.

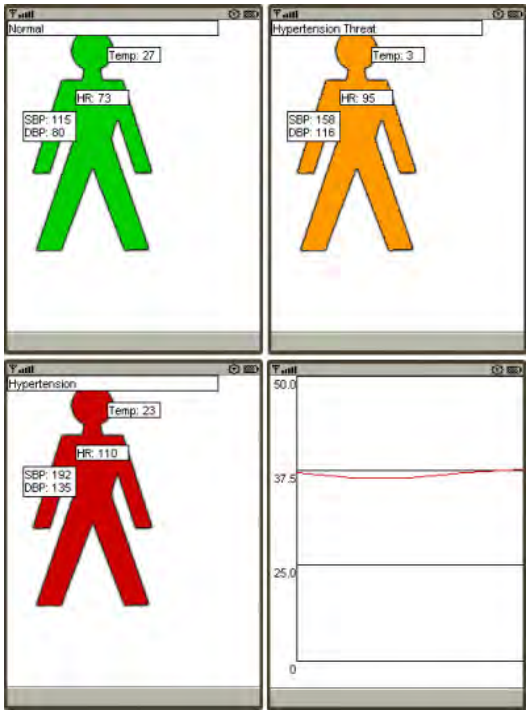


Figure 5. The application window on the emulator

Each of the images containing the body image in Figure 5 shows how the current states of the monitored streams (SBP, DBP, HR, temperature) are overlaid on the image to give the user an extra level of information when they have time to take a longer look at the screen.

As well as the simple overview displayed on the main screen, the application allows the user to 'drill down' into each of the displayed variables. The bottom-right image in Figure 5 shows the 'drill down' view of the temperature information. This allows the user or doctor to get a more detailed view of each sensor's data stream.

The top image in Figure 6 shows similar visualization on Nokia N95 mobile phone. The bottom image of the figure shows our prototype using a two-lead ECG biosensor from Alive Technologies [2] that transmits ECG signals using Bluetooth to the mobile phone. For the blood pressure and room temperature, we have used randomly generated data that simulates blood pressure and temperature fluctuations.

Figure 6. The application window on Nokia N95

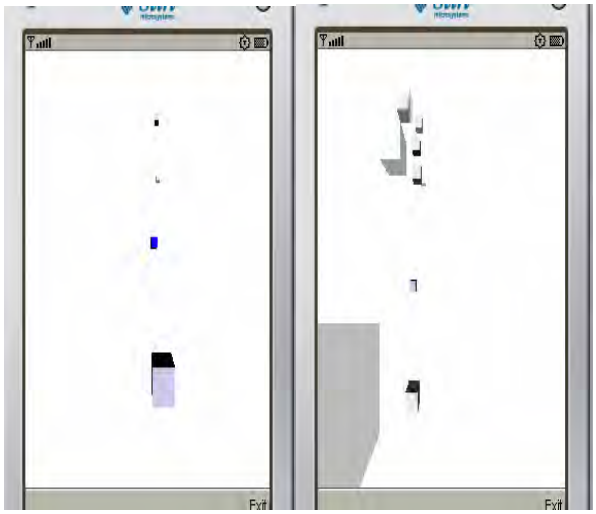


Figure 7. Visualization of the output of LWC for the normal and hypertension situations respectively

Figure 7 shows the visualization of the output of data stream mining using LWC (LightWeight Clustering) algorithm that has been adjusted according to the occurring situations of 'normal' and 'hypertension' respectively. Once we have the clustering results, we can use them in different ways. Detecting changes in clusters over time can indicate an event of interest as shown in the results obtained in [13]. Another way of using the clusters is by annotating them using knowledge-based systems [16]. Having a sufficient number of clusters is crucial to accurately perform both change detection and cluster annotation. Figure 7 shows that for a healthy individual, the number of clusters is less than for a hypertension one. This leads to a similar variation in the accuracy of change detection or cluster annotation.

In the implementation of our situation-aware adaptive data stream mining we have used LWC algorithm that operates using the AOG principals and provides adaptability by adjusting the parameter of threshold distance measure according to the available memory on a device such as a PDA. The threshold determines the distance between the center of a cluster and a new incoming data record. In our evaluation, we have used the parameter of threshold for the situation-aware adaptation.

We have set the threshold value for the situation of 'normal' to 42, 'pre-hypotension' to 36, 'hypotension' to 26, 'pre-hypertension' to 18 and 'hypertension' to 10. For critical situations the threshold needs to be decreased and for normal situations the threshold needs to be increased. This is because these values are acceptable given a variation of 12 (i.e. 42 divided by 3) for any of the context attributes of SBP, DBP and HR has no significant impact on a healthy individual while a variation of 3 for 'hypertension' can be significant.

To evaluate the adaptation process closely for each situation, we have simulated a 5-day scenario for a patient that experiences fluctuations of blood pressure. The dataset used for the evaluation is drawn from uniform distribution with different ranges for each context attribute of SBP, DBP and HR. The data is generated at a rate of 30 records/minute. The rate was chosen according to the application needs.

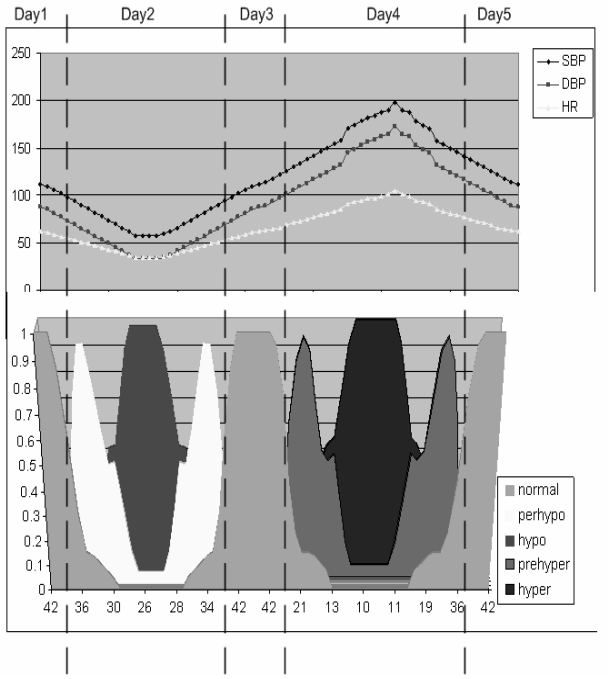


Figure 8. Results of evaluation for situation-aware adaptation of the threshold parameter of LWC

The purpose of the evaluation is to demonstrate gradual adaptation of the distance threshold parameter in real-time and according to the level of confidence in the occurrence of situations. The results of our evaluation are presented in Figure 8. The top graph shows changes of context attribute values for each day. The bottom graph illustrates how context attribute values are mapped into fuzzy situations and how the occurring situations are used for tuning the values of the distance threshold.

Figure 8 shows as the values of SBP, DBP and HR decrease, the membership degrees of 'hypotension' and 'pre-hypotension' situations increase and as these values increase, the membership degrees of 'hypertension' and pre-hypertension' increase. The figure also shows that the value of the threshold is dynamically adjusted according to the fuzziness of each situation at run-time. Decreasing the threshold value increases the number and accuracy of the output (clusters) that is required for closer monitoring of more critical situations such as hypertension. This output is passed to the next step of change detection and cluster annotation discussed earlier in this section.

To evaluate our prototype with a different parameter, we have performed a second evaluation considering the window size parameter as the adjustable parameter of LWC. We have assigned the set values for the situation of 'normal' as 60sec, 'pre-hypotension' as 45sec, 'hypotension' as 40sec, 'pre-hypertension' as 35sec and 'hypertension' as 30sec. Decreasing the window size parameter provides more frequent updates. This is needed when the situation is critical such as the 'hypertension' situation. Figure 9 shows the results of adaptation of window size. The figure illustrates gradual tuning of LWC's window size according to the results of situation inference and membership degree of situations (i.e. level of confidence of the occurring situations).

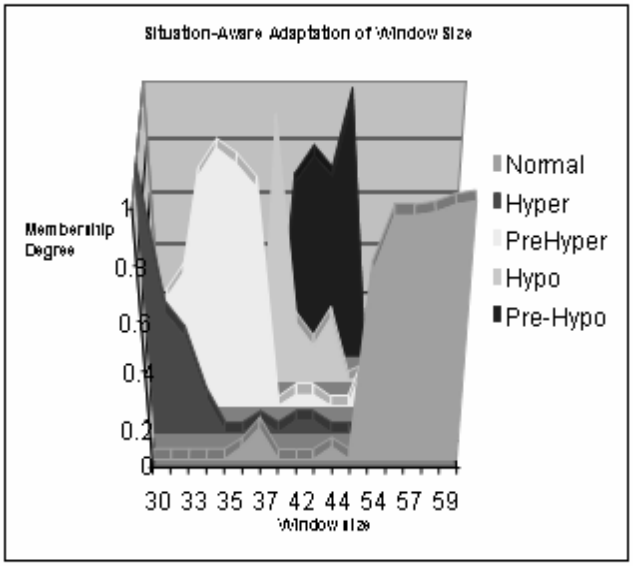


Figure 9. Situation-Aware Adaptation of window size

6. RELATED WORK

Our motivations for developing the architecture of situation-aware adaptive visualization techniques for mobile devices have led us to analyze recent publications focusing on three main research areas. These fields include ubiquitous data stream mining, context/situation modeling and reasoning using fuzzy logic, and finally visualization of data produced by data mining algorithms on mobile devices.

Data streams generated in wireless sensor networks can be processed on sensor nodes called as in-network processing [7], on a high performance computing facility at a central stationary site [17] or on mobile devices such as smart phone or PDA. Kargupta et al. [19] have developed a client/server data stream mining system called *MobiMine* which focuses on data stream mining applications for stock market data but the mining component is located at the server side rather than the PDA. In the Vehicle Data Stream Mining System (*VEDAS*) [20] the mining component is located at the PDA on-board the moving vehicle. A clustering technique has been used for analyzing and detecting the driver behavior. In [14], real-time mining of information from in-vehicle sensors to minimize driver distraction is proposed through adaptation of the instance-selection process based on changes to the data distribution. While this research recognizes the implicit need for adaptation, it is focused on intelligent sampling with little consideration for resource availability. In [30], context factors are used to filter datasets used for mining (rather than awareness of the application environment) and prune the user query to produce more accurate results. The final results are sent to a PDA but all the processing is performed on the server side.

One of the adaptive works in resource-aware data stream mining is Algorithm Output Granularity (AOG) [10] with its following works including Algorithm Input Granularity (AIG) and Algorithm Process Granularity (APG) [27] that provide adaptation of output and input streaming data and the process based on memory, battery and CPU cycles of mobile devices. However, most of these approaches have limited levels of

adaptations (mainly focusing on battery or memory) and they do not incorporate context-awareness into the ubiquitous data stream mining.

In [24], a fuzzy representation of context is introduced for adaptation of user interface application on mobile devices and the same fuzzy concept has been used in [5] for providing the user with an explicit and meaningful explanation for the system's proactive behavior while enabling user feedback for refining adaptation rules induced from context history. Alternatively, in [6, 8] fuzzy logic is used for defining the context situations and the rules for adaptation of the policies implementing services based on their fitness degree. The concept of situational computing using fuzzy logic presented in [3] is based on pre-developed ontologies and a similarity-based situation reasoning, and the situation-aware adaptation applies to applications according to user past reactions. Ranganathan et al. [28] represent context using predicates and use probabilistic and fuzzy logic for context reasoning. Probabilistic logic is used when there is precise knowledge of event probabilities and fuzzy logic is applied when this knowledge is not available.

Review of the proposed fuzzy approaches indicates that there is a lack of a general and formal fuzzy context modeling and reasoning approach that provides support for pervasive context-aware computing scenarios. The Context Spaces (CS) model [25], that we have based our model provides a Context Spaces Algebra and heuristically-based sensor data fusion algorithms specifically developed for reasoning about context in pervasive computing environments. Our fuzzy situation inference technique combines the strengths of fuzzy logic for modeling and reasoning of imperfect context and vague situations with the CS model's underlying theoretical basis for supporting context-aware pervasive computing scenarios.

Visualization is a useful tool in traditional data stream mining applications as it is able to represent interesting or important information to users while they are performing data mining operations manually [1, 9, 23]. The speed and volume of the data arriving at a device during data stream mining operations means that visualizing this data becomes a difficult task. In the application shown in [31] only a subset of IP addresses are able to be shown due to the limitations of standard screens. These kinds of problems increase in mobile visualization where screens are much smaller than those used with desktop data mining operations.

The MobiMine [19] application provides an example of visualization being used on a mobile device to assist users with interpretation of data mining results. However, in this system, data processing occurs on a central server with the results being sent to the user's device for visualization. The application does not take context considerations into account.

From our review of the literature relating to mobile and context-aware visualization of ubiquitous data mining results, we have found that there is no significant work which has been done in situation-aware adaptive visualization for mobile applications.

7. CONCLUSION AND FUTURE WORK

The growth and proliferation of technologies in the field of wireless networks and communication have led to emergence of diverse applications that process and analyze sensory data and display results on mobile devices such as a smart phone. However, the real power to make a significant impact on the area of developing these applications rests not merely on deploying wireless technology but on the ability to perform real-time, intelligent analysis and visualization of the data streams that are generated by the various sensors.

In this paper we introduced an integrated architecture of situation-aware adaptive data mining and mobile visualization techniques for ubiquitous computing environments. Situation-aware adaptive processing (SAAP) of data streams provides a fuzzy approach for situation modeling and reasoning and enables situation-aware adaptation of parameters of data stream mining algorithms. The MobileVis framework provides context-aware visualization services to mobile applications. This integrated approach significantly enhances and enables a range of ubiquitous applications including mobile healthcare systems.

Since SAAP and MobileVis are integrated with resource-aware data stream mining algorithms such as LWC, they are capable to support resource-awareness. We are adding a Resource-Monitor (RM) component to our architecture that constantly reports availability of resources (i.e memory availability, battery charge or CPU utilization) to the adaptation engine. We are extending our adaptation strategies with a set of situation-aware, resource-aware and integrated adaptation strategies that consider both the results of situation inference and availability of resources in the adaptation of parameters of data stream mining algorithms.

8. REFERENCES

[1] Aggarwal, C. C. 2003. A Framework for Diagnosing Changes in Evolving Data Streams. Proceedings of the ACM SIGMOD Conference.

[2] Alive Technologies, http://www.alivetec.com

[3] Anagnostopoulos, C.B., Ntarladimas Y., and Hadjiefthymiades, S. 2007. Situational Computing: An Innovative Architecture with Imprecise Reasoning, the Journal of Systems and Software 80, 2007, 1993-2014.

[4] Burkhard, R. 2004. Learning from Architects. 2004. The Difference between Knowledge Visualization and Information Visualization, Eight International Conference on Information Visualization (London, July 2004) IV04, 519 – 524.

[5] Byun, H, and Keith, C. 2003. Supporting Proactive 'Intelligent' Behaviour: the Problem of Uncertainty, Proceedings of the UM03 Workshop on User Modeling for Ubiquitous Computing. (Johnstown, PA, 2003), 17–25,.

[6] Cao, J., Xing, N., Chan, a., Feng, Y., and Jin, B. 2005. Service Adaptation Using Fuzzy Theory in Context-aware Mobile Computing Middleware, In Proceedings of the 11th IEEE Conference on Embedded and Real-time Computing Systems and Applications (RTCSA'05), 2005.

[7] Chen, Y., H., Leong, H., Xu, M., Cao, J., Chan, K., Chan, A., 2006. In-network Data Processing for Wireless Sensor Networks, Proceedings of the 7th International Conference on Mobile Data Management, MDM'06.

[8] Cheung, R. 2005. An Adaptive Middleware Infrastructure Incorporating Fuzzy Logic for Mobile computing. In

Proceedings of the International Conference on Next Generation Web Services Practices, NWeSP'05.

[9] de Oliveira, M. C. F. and Levkowitz, H. (2003). From visual data exploration to visual data mining: A survey. IEEE Trans. on Visualization and Computer Graphics, 9(3), 378–394.

[10] Gaber, M., Krishnaswamy, S., and Zaslavsky, A., 2003. Adaptive Mining Techniques for Data Streams Using Algorithm Output Granularity, The Australasian Data Mining Workshop, Held in conjunction with the 2003 Congress on Evolutionary Computation (Canberra, Australia, December 2003) Springer Verlag, Lecture Notes in Computer Science (LNCS), AusDM '03.

[11] Gaber, M.M., Krishnaswamy, S., and Zaslavsky, A. 2004. Ubiquitous Data Stream Mining, Current Research and Future Directions Workshop Proceedings held in conjunction with The Eighth Pacific-Asia Conference on Knowledge Discovery and Data Mining (Sydney, Australia, May 26, 2004).

[12] Gaber, M., Krishnaswamy, S., and Zaslavsky, A. 2005. "On-board Mining of Data Streams in Sensor Networks", A Book Chapter in Advanced Methods of Knowledge Discovery from Complex Data, (Eds.) S. Badhyopadhyay, U. Maulik, L. Holder and D. Cook, Springer Verlag., 2005.

[13] Gaber, M.M. and Yu, Ph. S. 2006. Detection and Classification of Changes in Evolving Data Streams. International Journal of Information Technology and Decision Making 5(4), 659-670.

[14] Galan, M., Liu, H., Torkkola, K. 2005. Intelligent Instance Selection of Data Streams for Smart Sensor Applications. SPIE Defense and Security Symposium, Intelligent Computing: Theory and Applications III. 108-119.

[15] Gillick, B., Krishnaswamy S., Gaber M., and Zaslavsky A. 2006. Visualisation of Fuzzy Classification of Data Elements in Ubiquitous Data Stream Mining. IWUC'06, 29-38.

[16] Horovitz, O., Krishnaswamy, S., Gaber, M.M. 2007. A fuzzy approach for interpretation of ubiquitous data stream clustering and its application in road safety. Intell. Data Anal. 11(1), 89-108.

[17] Hossain, A. 2002. An intelligent sensor network system coupled with statistical process model for predicting machinery health and failure, Sensors for Industry Conference.

[18] Jang, J., Sun, C., and Mizutani, E. 1997. Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence. Prentice-Hall: Upper Saddle River, NJ, 1997.

[19] Kargupta, H., Park, B., Pittie, S., Liu, L., Kushraj, D., Sarkar, K. 2002. MobiMine: Monitoring the Stock Market from a PDA. SIGKDD Explorations 3(2), 37-46.

[20] Kargupta, H., Bhargava, R., Liu, K., Powers, M., Blair, P., Bushra, S., Dull, J., Sarkar, K., Klein, M., Vasa, M., Handy, D. 2004. VEDAS: A Mobile and Distributed Data Stream Mining System for Real-Time Vehicle Monitoring. In: Proceedings of the SIAM International Data Mining Conference, SDM'04.

[21] Keim, D. A. 2002. Information visualization and visual data mining. IEEE Transactions On Visualization And Computer Graphics, 8(1), 1-8.

[22] Leijidekkers, P., and Gay, V. 2005. Personal Heart Monitoring and Rehabilitation System using Smart Phones. In Proceedings of the International Conference on Mobile Business, ICMB'05.

[23] Liu, D., Sprague, A. P., and Manne, U. 2004. JRV: an interactive tool for data mining visualization. ACM Southeast Regional Conference 2004: 442-447.

[24] Mäntyjärvi, J., and Seppanen, T. 2002. Adapting Applications in Mobile Terminals Using Fuzzy Context Information, In the Proceedings of 4th International Symposium on Mobile HCI2002, Italy, 2002, 95-107.

[25] Padovitz, A., Loke, S., and Zaslavsky, A. 2004. Towards a Theory of Context Spaces. In Proceedings of the 2nd IEEE Annual Conference on Pervasive Computing and Communications, Workshop on Context Modeling and Reasoning (Orlando, Florida, March 2004) CoMoRea'04. IEEE Computer Society.

[26] Padovitz, A., Zaslavsky, A. and Loke, S. 2006. A Unifying Model for Representing and Reasoning About Context under Uncertainty, 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (Paris, France, July 2006) IPMU'06.

[27] Phung, N., Gaber, M., and Roehm, U. 2007. Resource-aware Distributed Online Data Mining for Wireless Sensor Networks, Proceedings of the International Workshop on Knowledge Discovery from Ubiquitous Data Streams (IWKDUDS07), in conjunction with ECML and PKDD 2007 (Warsaw, Poland, September 17, 2007).

[28] Ranganathan, A., Al-Muhtadi, J., and Campbell, R. 2004. Reasoning about Uncertain Contexts in Pervasive Computing Environments. In IEEE Pervasive Computing, 3(2), (April-June 2004), 62-70.

[29] Rubel, P., Fayn, J., Nollo, G., Assanelli, D., Li, B., Restier, L., Adami, S., Arod, S., Atoui, H., Ohlsson, M., Simon-Chautemps, L., Te´lisson, D., Malossi, C., Ziliani, G., Galassi, A., Edenbrandt, L., and Chevalier, P. 2005. Toward Personal eHealth in Cardiology: Results from the EPI-MEDICS Telemedicine Project. Journal of Electrocardiology, 2005, 38, 100-106.

[30] Vajirkar, P., Singh, S., Lee, Y. 2003. Context-Aware Data Mining Framework for Wireless Medical Application, Lecture Notes in Computer Science (LNCS), Volume 2736, Springer-Verlag. ISBN 3-540-40806-1, 381 – 391.

[31] Wegman, E., Marchette, D. 2003. On some techniques for streaming data: A case study of Inter-net packet headers, Journal of Computational and Graphical Statistics, 12(4), 893-914.

[32] Zadeh, Z. 1975. The Concept of a Linguistic Variable and Its Application to Approximate Reasoning. Information Systems, 1975, 199-249.

[33] Zimmermann, H., 1996. Fuzzy Set Theory - and Its Applications. Kluwer Academic Publishers, Norwell, Massachusetts.

Spatiotemporal Neighborhood Discovery for Sensor Data

Michael P. McGuire
Center for Urban Environmental Research and Education, University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250, USA
mcguire1@umbc.edu

Vandana P. Janeja
Information Systems Department, University of Maryland Baltimore County,, 1000 Hilltop Circle, Baltimore, MD 21250, USA
vjaneja@umbc.edu

Aryya Gangopadhyay
Information Systems Department, University of Maryland Baltimore County, , 1000 Hilltop Circle, Baltimore, MD 21250, USA
gangopad@umbc.edu

ABSTRACT

The focus of this paper is the discovery of spatiotemporal neighborhoods in sensor datasets where a time series of data is collected at many spatial locations. The purpose of the spatiotemporal neighborhoods is to provide regions in the data where knowledge discovery tasks such as outlier detection, can be focused. As building blocks for the spatiotemporal neighborhoods, we have developed a method to generate spatial neighborhoods and a method to discretize temporal intervals. These methods were tested on real life datasets including (a) sea surface temperature data from the Tropical Atmospheric Ocean Project (TAO) array in the Equatorial Pacific Ocean and (b)highway sensor network data archive. We have found encouraging results which are validated by real life phenomenon.

1. INTRODUCTION

Sensors are typically used to measure a phenomenon and result in a time series of measurements associated with a specific location. For example environmental sensors monitor quality, temperature etc. in air, water or land, traffic sensors monitor congestion on highways, and comparative vacuum monitoring sensors monitor the structural stability of bridges. Such sensors can be considered as *spatial* objects generating measurements over a period of time (*temporal*). A key to effective knowledge discovery tasks (such as outlier detection, pattern discovery etc.) is to first identify a group of sensors which may be characterized similarly based on their spatial proximity and temporal measurements. For instance, an outlying sensor in a set of traffic sensors is one which is unusual with respect to its nearby sensors. This characterization of similar sensors where the data is spatiotemporal in nature is termed as the *spatiotemporal neighborhood*. In this paper our focus is the discovery of spatiotemporal neighborhoods which, consists of three components:

- Defining spatial neighborhoods
- Discretizing temporal intervals
- Combining spatial neighborhoods with temporal intervals to generate spatiotemporal neighborhoods.

Our notion of a spatiotemporal neighborhood is distinct from the traditional notions since we not only consider a spatial characterization but also a temporal characterization to form our spatiotemporal neighborhoods.

Traditionally, spatial neighborhoods are defined as a group of objects that are in spatial proximity to each other that have similar non-spatial attributes [21] [6]. A particular challenge in this research is to extend this definition to include non-spatial attribute values in the formation of the neighborhoods and to account for neighborhood boundaries that are not crisp.

If there is a vast number of measurements over a period of time associated with each spatial object it is not feasible to analyze every value in such a complex time series. Thus, a temporal characterization must discretize [13] a time series in such a way that the resulting intervals represent distinct temporal features within which knowledge discovery can be focused. Therefore, we define a temporal interval as a segment of time that has similar measurement characteristics. The method to generate temporal intervals must be able to handle the complexity that is often found in real world datasets. This is particularly a challenge in situations where divisions between intervals are not easily deduced and the number of temporal intervals is not known before hand.

The individual challenges of generating spatial neighborhoods and temporal intervals are compounded when combined to form spatiotemporal neighborhoods. A particular challenge is to be able to track spatial change over time. Just as it is not feasible to analyze every value in a complex time series, it is even more problematic to analyze spatial patterns at every time step in a dataset. Because of this, a major challenge of the spatiotemporal neighborhood approach will be to find the temporal intervals where changes in spatial patterns occur. This research is applicable to a number of domains including transportation planning, climatology, meteorology, hydrology, and others. We next present two motivating examples:

EXAMPLE 1. *Climatology: The TAO array [19] consists of sensors installed on buoys positioned in the equatorial region of the Pacific Ocean. The sensors collect a wide range of meteorological and oceanographic measurements . Sea Surface Temperature (SST) measurements are reported every five minutes. Over time, this results in a massive dynamic spatiotemporal dataset. This data played an integral part of characterizing the 1997-98 El Nino [17] and is currently being used to initialize models for El Nino prediction.*

There have been a number of studies which assimilate meteorological and oceanographic data to offer a description of the phenomena associated with the events of the 1982-83 El Nino [4] [20] and the 1997-1998 El Nino [17]. These analyses show a particular importance in the spatiotemporal patterns of meteorological variables and SST anomalies that characterize El Nino events.

El Nino events are most often characterized by anomalously high values of SST in the eastern Pacific from 160 degrees west eastward to the coast of South America. Daily anomalies are typically calculated using a combination of in situ and satellite measurements where the degree of the anomaly is based on the difference between the current SST analysis value and SST monthly climatology. This method finds global outliers at a relatively high spatial resolution. However, if a scientist would like to see outliers at higher temporal resolutions than the daily average, a dataset with a higher temporal frequency, such as data from the TAO / TRITON network, is needed. This data consists of a vast time series collected at 44 sensors across the equatorial Pacific Ocean. The challenge from the scientist's perspective is first to find the sensors in the TAO network that are proximal and have similar SST measurements. To make the analysis more efficient, the scientist would like to automatically find areas in the data where changes to the spatial patterns are most likely to occur and focus the analysis on finding anomalies in these areas.

EXAMPLE 2. *Traffic Monitoring: Traffic congestion is a common problem in urban areas. The duration and intensity of congestion has grown over the last 20 years [2]. Because of this, transportation planners are continually devising strategies to combat congestion. Many highway systems are now employing Intelligent Transportation Systems (ITS) and have sensors which monitor traffic conditions. These sensors allow traffic engineers to understand the dynamics of traffic in multiple locations on the highway network and in turn offer insight into the spatiotemporal patterns of congestion. There are a number of traffic control measures that can be employed to reduce congestion. But to arrive at an optimal solution, traffic engineers must understand where congestion exists in order to determine locations to introduce traffic control measures. In this situation, knowing the spatiotemporal pattern of congestion would be extremely useful. Furthermore knowing the spatiotemporal characterization would allow the traffic engineer to identify anomalies that occur during peak period and off peak period hours and provide a better understanding of the dynamics that cause congestion and result in new strategies to deal with congestion problems.*

Key Contributions: From these motivating examples we can identify the following key contributions of our work in discovering the spatiotemporal characterization which we refer to as the spatiotemporal neighborhood for complex sensor data.

***Spatial Neighborhoods*:** While generating spatial neighborhoods it is essential to find the spatial distribution of measurements at individual locations in combination with the spatial relationships between locations. One important challenge in identifying the spatial neighborhoods in real world datasets is that they do not have crisp boundaries. Thus a key contribution of this work is to accommodate for overlapping neighborhoods.

***Temporal Intervals*:** These intervals embody the concept of neighborhoods in time (similar to spatial neighborhoods in space). A major contribution of this work is to create unequal width or unequal frequency intervals that are robust in the presence of outliers.

***Spatiotemporal Neighborhoods*:** There have been a number of approaches in the literature which model spatiotemporal patterns using a graph-based approach [14] [8] [5]. However, our approach is the first approach to pinpoint temporal intervals where the spatial pattern changes. In this case, it becomes critical to accommodate the individual properties of spatial and temporal neighborhoods to identify points in time where the spatial pattern is most likely to change and identify temporal patterns at many spatial locations.

In this paper we propose a method to generate spatiotemporal neighborhoods. This is accomplished by first performing a spatial characterization of the data; then defining distinct temporal intervals; and finally by defining spatial neighborhoods at each interval. We discuss experiments on real world datasets on SST and traffic data with promising results in finding spatial neighborhoods and distinct temporal intervals in both datasets.

The rest of the paper is organized as follows. In section 2 we discuss our approach. In section 3 we outline our experimental results. Section 4 discusses related work and finally in section 5 we conclude and discuss some challenges for future research.

2. APPROACH

The overall approach is outlined in figure 1, which comprises of the following distinct steps.

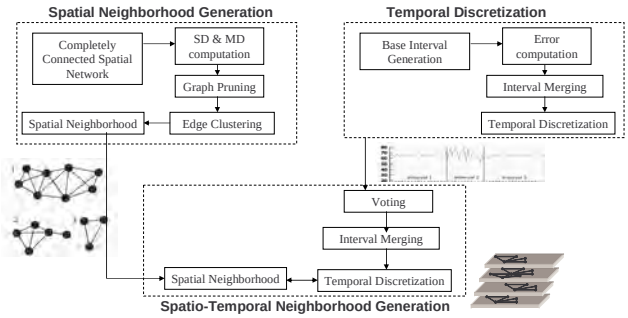


Figure 1: Spatiotemporal Neighborhood Generation

1) Spatial Neighborhood Generation: We begin by creating the spatial neighborhoods using a graph-based structure derived from the relationships between the spatial nodes in terms of their spatial proximity and measurement similarity.

2) Temporal Interval Generation: We use agglomerative clustering to generate temporal intervals in a time series dataset comprised of measurements collected at a spatial node. For this we start with temporal intervals of a preset small size and merge contiguous intervals with similar within-interval statistics resulting in a set of unequal width intervals representing distinct sections of the time series.

3) Spatiotemporal Neighborhood Generation: Using these building blocks of spatial neighborhoods and temporal intervals we next generate the spatiotemporal neighborhoods.

2.1 Spatial Neighborhood Generation

A spatial neighborhood is defined as a group of spatial nodes that are within proximal distance of each other and exhibit similar characteristics. Before we formally define our concept of spatial neighborhood we define some spatial primitives:

DEFINITION 1 (SPATIAL NODE). *Let S be a set of spatial nodes $S = \{s_1, ..., s_n\}$ where each $s_i \in S$ has a set of coordinates in 2D Euclidean space (s_{ix}, s_{iy}) and a set of attributes $A_i = \{s_{ia1}, \ldots, s_{iam}\}$.*

To define a spatial neighborhood we first consider the spatial proximity as defined by spatial relationships:

DEFINITION 2 (SPATIAL RELATIONSHIP). *Given two spatial nodes $(s_p, s_q) \in S$ a spatial relationship $sr(s_p, s_q)$ exists if there exists a distance, direction or topological relationship between them.*

For instance the spatial relationships may be qualified using a distance relationship based on the following concept of Spatial distance:

DEFINITION 3 (SPATIAL DISTANCE). *The spatial distance $sd(s_p, s_q)$ is calculated as the Euclidean distance between two spatial coordinates such that*

$$sd = \sqrt{(s_{px} - s_{qx})^2 + (s_{py} - s_{qy})^2}$$

In addition to the spatial relationship we also quantify the similarity between nodes based on the distance between the measurement values(or the non-spatial attributes) of the spatial nodes as follows:

DEFINITION 4 (MEASUREMENT DISTANCE). *The measurement distance $md(s_p, s_q)$ is the Euclidean distance between the set of normalized numerical attributes A_p and A_q at s_p and s_q such that*

$$md = \sqrt{\sum_{1}^{m} (s_{pam} - s_{qam})^2}$$

for m attributes measured at each spatial node.

We next define our notion of spatial neighborhood:

DEFINITION 5 (SPATIAL NEIGHBORHOOD). *Given a set of spatial nodes $S = \{s_1, \ldots, s_n\}$ a spatial neighborhood $spn = \{sp_1, \ldots, sp_l\}$ such that $spn \subset S$ where $\forall sp_i \in spn$ exhibits $sd(sp_i, sp_j) < d$, where d is a spatial distance threshold and $md(sp_i, sp_j) < \delta$ where δ is a measurement distance threshold.*

Our spatial neighborhood method uses a graph-based structure to model the data such that a spatial neighborhood graph $SG = sg, < e >$ where sg is a set of nodes $\in spn$ such that for all pair of nodes $(s_i, s_j) \in sg$ there exists an edge $< e_i, e_j > \in e$.

In this neighborhood graph, the edges form relationships between the spatial nodes such as the spatial distance between two nodes or the distance between measurements taken at two nodes. For example, this could be the distance between SST measurements taken at two neighboring sensors. Figure 2 shows an illustrative example of graph-based spatial neighborhoods.

On the left, the measurement of spatial nodes is shown and all possible relationships between the nodes are shown as edges. The right shows three neighborhoods that are formed after applying the distance and measurement thresholds. Neighborhood 1 shows a contiguous group of sensors that are connected by being close in proximity and having similar measurement values. Neighborhoods 2 and 3,

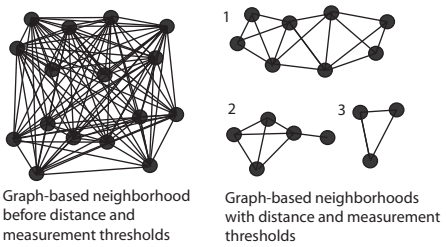


Figure 2: Graph-based Spatial Neighborhoods

while proximal to each other are divided by the measurement threshold. Notice the sensor that falls in the middle of neighborhood 2 and 3. This sensor is close in proximity to nodes in both neighborhoods however, because of the measurement threshold, it is more similar to the nodes in neighborhood 2.

Algorithm 1 Procedure: Graph-based Spatial Neighborhood Generation

Require: A set of spatial nodes $S = \{s_1, \ldots, s_n\}$
Require: A spatial distance threshold d
Require: A measurement distance threshold δ
Require: Number of clusters C
Ensure: A set of spatial neighborhoods $spn = [NodeID, NeighborhoodID]$

```
//Initialize the graph and calculate pairwise euclidean distance
for i = 1 to n do
    COUNT = n-i
    for j = 1 to COUNT do
        Add edges i,i + j to create n(n − 1)/2 edge matrix
        CALCULATE sd, md, and mean measurement between node
        i and i + j and add to edge matrix
    end for
end for
//Apply distance and measurement thresholds to graph
SelectedEdges = edges(sd < d AND md < δ)
//Cluster edges based on measurement values
CIndex = K-Means(mean measurement,C)
EdgeCluster = CONCATENATE(SelectedEdges,CIndex)
//Assign nodes to neighborhoods based on CIndex
for each selected edge s do
    for each cluster C do
        if EdgeCluster(s) = C then
            Membership(s) = Nodes in EdgeCluster(s)
            Remove duplicate Node IDs from Membership(s)
        end if
    end for
end for
for each neighborhood N do
    CALCULATE nq //Calculate neighborhood quality
end for
```

The ultimate goal of this approach is to find spatial groups in the data that are also based on non-spatial attributes. To do this we apply clustering to the non-spatial attributes of the remaining edges. Clustering is also used because in some cases, the edges that remain after applying the d and md thresholds do not form discrete neighborhood divisions. For example, if a node is within d of two neighborhoods and has a md that is less than δ from a node in each neighborhood, this node will connect the two neighborhoods and therefore finds non-crisp neighborhood boundaries. Clustering can address this if crisp boundaries are required because it will assign edges to neighborhoods based on the mean measurement value between the two nodes. The nodes of the resulting clusters are then extracted to form the spatial neighborhoods. The neighborhood quality is then measured where the measurement values of the nodes are compared to

the mean measurement value of the spatial neighborhood.

DEFINITION 6 (NEIGHBORHOOD QUALITY). *We use a within-neighborhood sum of squared error (SSE)function applied to the set of attributes A_i for each s_i to measure the spatial neighborhood quality nq such that:*

$$nq = \sum_{i=1}^{n} (s_{iam} - \mu^{spn}))^2/n$$

where s_{iam} are the attribute values for each s_i and μ^{spn} is the mean measurement value for the entire spatial neighborhood. The nq is divided by n to normalize the value so that it can be compared across neighborhoods of varying sizes.

The Spatial Neighborhood generation is outlined in Algorithm 1. The algorithm requires a set of spatial nodes and corresponding attributes and threshold values for the spatial and measurement distance between spatial nodes. These thresholds are used as heuristics to control the relationships between spatial nodes. For example if two spatial nodes are too far apart but have similar measurement values, the edge would be removed from the clustering.

2.2 Temporal Interval Generation

In this section, we present an agglomerative approach to generate temporal intervals from a set of temporal measurements. Figure 3 gives an illustrative example of this approach.

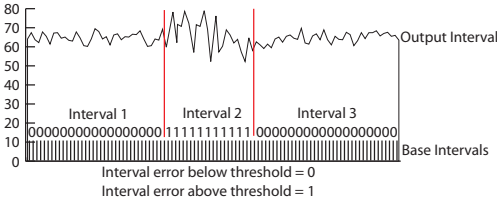


Figure 3: Agglomerative Temporal Discretization

The agglomerative approach first divides the time series into a set of base equal frequency temporal intervals. In general, a temporal interval is defined as:

DEFINITION 7 (TEMPORAL INTERVAL). *Given a set of temporal measurements $T = \{t_1, \ldots, t_n\}$ a temporal interval $int = \{t_1, \ldots, t_m\}$ is a division of T such that $int \subset T$ and $int_1 < int_2, \ldots, < int_k$, where each $int^i =< int^i_{start}, int^i_{end} >$ such that the size $int^i_{size} = (int_{start} - int_{end})$.*

We would like to create intervals where the size of the interval is variable (unequal width intervals). In order to create such intervals we begin first with base intervals int^{base} where the size int^{base}_{size} is fixed to begin with and is a user defined parameter which largely depends on the domain and granularity of the analysis. We calculate SSE for each base interval as follows:

DEFINITION 8 (SSE). *The SSE is the sum of the squared differences of each value within int^{base} from the mean of all values in int^{base} such that:*

$$SSE = \sum_{bn=1}^{BN} dist(int^{base}_{bn} - \mu^{base}_{int})^2$$

Here bn is each temporal reading in the total BN readings for the base interval. Then for each base interval, the SSE value is given a binary classification which assigns base intervals as having either a high or low within-interval error. The binary interval error is defined as follows:

DEFINITION 9 (BINARY INTERVAL ERROR). *A binary interval error $\epsilon = (1, 0)$ such that if $SSE(int) > \lambda$ then $\epsilon = 1$ else $\epsilon = 0$*

Here $\epsilon = 1$ is a high error and $\epsilon = 0$ is a low error. this error is applied to each int^{base} by using an error threshold λ such that if $SSE(int) > \lambda$ the interval is classified as 1 and if $SSE(int) < \lambda$ the interval is classified as 0.

Based on the binary interval error we merge the base intervals into larger intervals such that consecutive groups have similar error classification. This results in a set of variable width temporal intervals defined by the within-interval error. This method is flexible in that any statistical measure can be used for within-interval error. Currently as an example, we have used SSE.

The agglomerative method is formalized in Algorithm 2. The algorithm requires as input a time series ts, a base temporal interval size, and a minimum error threshold λ that is used to merge intervals. The output of the algorithm is a set of variable width temporal intervals defined by columns representing the interval start, interval end, and interval error.

Algorithm 2 Procedure:Temporal Interval Generation

```
Require: Time series measurements ts and its instances
  t_1, t_2, ..., t_n
  where t ∈ ts and t_1 < t_2 < t_n
Require: base temporal interval size I
Require: error threshold λ
Ensure: Set of variable width temporal intervals I = i_1, ..., i_n
  where each i = start, end, error
  //Create base temporal intervals and calculate SSE
  Interval Start = 1
  Interval End = Interval Start + I
  while Interval Start < length(ts) do
    CALCULATE SSE for interval
  end while
  //Apply Binary Error Classification
  for each i in I do
    if interval SSE < λ then
      ErrorGroup(t) = 0
    else
      ErrorGroup(t) = 1
    end if
  end for
  //Merge binary classification to create temporal intervals
  for each i in I do
    if ErrorGroup(t) ≠ ErrorGroup(t+1) then
      Add Interval Start and Interval End to output
    end if
  end for
```

2.3 Spatiotemporal Neighborhood Generation

Space and time are most often analyzed separately rather than in concert. Many applications collect vast amounts of data at spatial locations with a very high temporal frequency. For example, in the case of SST, it would not be possible to comprehend 44 individual time series across the equatorial Pacific Ocean. Furthermore, to look at the change in spatial pattern at each time step would also be confusing because it would require a large number of map overlays. The challenge in this case is to find the temporal intervals where the spatial neighborhoods are likely to experience the most change in order to minimize the number of spatial configurations that need to be analyzed.

In our method for spatiotemporal neighborhoods we have incorporated both of the above approaches into an algorithm that generates the temporal intervals where spatial patterns are likely to change and for each interval generates

Algorithm 3 Algorithm for Spatiotemporal Neighborhoods

Require: A set of spatial nodes $S = [s_1, \ldots; s_n]$ where each s_i has a time series of measurements T and its instances $[t_1, t_2, \ldots, t_n]$ where $t \in T$ and $t_1 < t_2 < t_n$
Require: A spatial distance threshold d
Require: A measurement distance threshold δ
Require: A base temporal interval size I
Require: An interval error threshold λ
Require: A minimum number of votes threshold mv
Require: Number of clusters C
Ensure: A set of spatiotemporal neighborhoods STN = [IntervalStart,IntervalEnd,NodeID,NeighborhoodID] //Procedure: Graph-based Spatial Neighborhood Generation //Procedure: Temporal Interval Generation //Procedure: Create spatiotemporal graph
  for each t in ts **do**
    if SUM(ErrorGroup(t))<mv **then**
      IntervalError(t) = 0 //Apply voting function
    else
      IntervalError(t) = 1
    end if
  end for
  for each interval i = 1 to number of intervals **do**
    if IntervalError(i) $\neq$ IntervalError($i+1$) **then**
      Add Interval Start and Interval End to output matrix IntInterest //Merge binary classification to create temporal intervals
    end if
  end for
  //Form spatial neighborhoods for each interval
  for each IntInterest I **do**
    for each proximal edge p **do**
      p_{md} = MEAN(md) //Calculate mean md for each interval
      if $p_{md} < \delta$ **then**
        SelectedEdges = ProximalEdges //Apply δ to mean md of edges at each temporal interval
      end if
    end for
  end for
  for each IntInterest I **do**
    CIndex = K-Means(edge mean measurement value,C) //Cluster edges based on measurement values
    EdgeCluster = CONCATENATE(SelectedEdges,CIndex)
  end for
  for each IntInterest$I$ **do**
    for each selected edge s **do**
      for each C **do**
        if EdgeCluster(s) = C **then**
          Membership(C) = Nodes in EdgeCluster(s) //Assign nodes to neighborhoods based on CIndex
          Remove duplicate values from Membership(C)
        end if
        CALCULATE nq //Calculate neighborhood quality
      end for
    end for
  end for

spatial neighborhoods. The combined result of this algorithm is a characterization of the spatiotemporal patterns in the dataset.

Because of the addition of a time series to the spatial dataset, the spatiotemporal algorithm has a number of subtle differences from the above approaches. The first is that a long time series makes it less efficient to calculate the md and mean measurement value at the same time as sd. Therefore threshold d is applied first and the md and mean measurement values are calculated only for the proximal edges.

The spatiotemporal algorithm also requires an additional step to deal with time series at many spatial nodes. After the binary error classification is created for each time series at each spatial node, the time series has to be combined to form temporal intervals that can be applied to all spatial nodes. To accomplish this task, we have implemented a voting function to count for each base temporal interval, the number of spatial nodes that have an error classification. The voting function counts for each int the number of spatial nodes that have a binary error classification of 1. This results in the total number of base intervals that have high error values.

A threshold mv is then applied to the result of the voting algorithm where mv represents the minimum number of votes for a temporal interval to be considered a high error interval for all spatial nodes. The application of mv converts the result of the voting algorithm back to a binary matrix by giving each $int_{votes} > mv$ a value of 1 and each $int_{votes} < mv$ a value of 0. These intervals are then merged using the same method as in the agglomerative temporal interval algorithm. This results in a set of temporal intervals for which the md and measurement values for each edge are averaged. Once the temporal intervals are created, the δ threshold is applied to the mean md for each edge in each interval resulting in a selected set of edges for each temporal interval. Then the edges are clustered for each interval and the spatial nodes are assigned to their respective spatial neighborhoods. The spatiotemporal neighborhood generation algorithm is presented in Algorithm 3.

3. EXPERIMENTAL RESULTS

Our experimental results are organized as follows:

- Spatial Neighborhood discovery
- Temporal Interval discovery
- Spatiotemporal Neighborhood discovery

We utilized two datasets Sea Surface Temperature Dataset(SST) and Maryland Highway Taffic Dataset. We next outline these two datasets. Subsequently we discuss the results obtained in these two datasets.

3.1 Datasets

SST Data The algorithms were tested on sea surface temperature data from the Tropical Atmospheric Ocean Project (TAO) array in the Equatorial Pacific Ocean [19]. These data consisted of measurements of sea surface temperature (SST) for 44 sensors in the Pacific Ocean where each sensor had a time series of 1,440 data points. The format of the SST data shown in Table 1 has columns for latitude, longitude, data, time (GMT), and SST in degrees Celsius. The temporal frequency of the data is 15 minutes. The SST data was used to demonstrate methods for spatial neighborhoods, temporal intervals, and spatiotemporal neighborhoods.

Table 1: Sea Surface Temperature Data Format

Latitude	Longitude	Date	Time	SST(degrees C)
0	-110	20040101	000001	24.430
0	-140	20040101	000001	25.548
0	-155	20040101	000001	25.863
...	...	...	...	...

Traffic Data The algorithms were also tested using average traffic speed from a highway sensor network data archive operated by the Center for Advanced Transportation Technology Laboratory at the University of Maryland, College Park [7]. The format of the traffic data shown in Table 2 consists of columns for date and time, direction, location, and average speed in miles per hour. The temporal frequency of the data is 5 minutes and consisted of approximately 2,100 data points for each sensor. This data was used to test graph-based spatial neighborhood, agglomerative temporal interval, and spatiotemporal neighborhood algorithms.

Table 2: Average Traffic Speed Data Format

Date Time	Direction	Location	Speed(mph)
1/2/2007 0:01	East	US 50 @ Church Rd	79
1/2/2007 0:06	East	US 50 @ Church Rd	81
1/2/2007 0:11	East	US 50 @ Church Rd	61
...	...	...	...

3.2 Spatial Neighborhood discovery

The graph-based spatial neighborhood algorithm was applied to both SST and traffic data. In this section the preliminary results of this analysis are presented.

SST Data: Figure 5 shows the edge clustering of the spatial neighborhood for the TAO array.

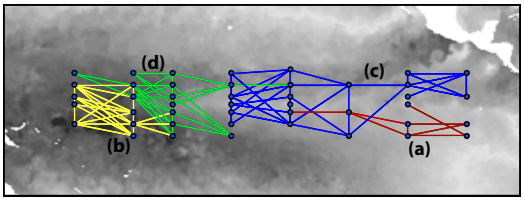


Figure 4: Result of edge clustering for SST in the Equatorial Pacific

Validation: The resulting edge clustering is validated by the satellite image of SST where the light regions represent cooler temperatures and the dark regions represent warmer temperatures. The edges in Figure 4(a) represent cooler water that extends from the southwestern Pacific shown in lower right part of the SST image and extends westward along the equator. The cluster shown in Figure 4(b) represents the warm waters of the southwestern Pacific shown in the lower left part of the image. The clusters in Figure 4(c) and (d) represent more moderate temperature regions that fall in between the extremes of clusters (a) and (b). A depiction of the nodes colored by neighborhood is shown in Figure 5.

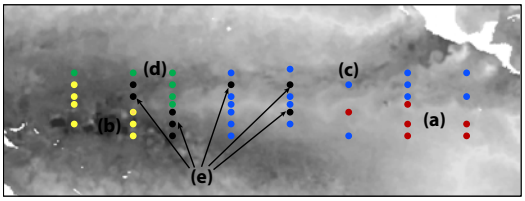


Figure 5: Graph-based neighborhoods for SST in the Equatorial Pacific

The neighborhoods shown in Figure 5(a), (b), (c), and (d) directly reflected the result of the edge clustering and thus were also validated by the pattern of SST shown in the satellite image background. Figure 5(e) refers to nodes that had edges that are connected to nodes from multiple neighborhoods. These nodes represent locations where the neighborhoods overlap and, as would be expected, typically occur along neighborhood boundaries. This illustrates the continuous nature of SST data and a major challenge to defining spatial neighborhoods in that the spatial patterns are more represented by gradual changes in SST rather than well defined boundaries.

The last step in the algorithm was to calculate the neighborhood quality using the SSE/n of the measurements taken at the nodes within the neighborhood. The neighborhood quality for the above neighborhoods is shown in Table 3.

Table 3: Graph-based Neighborhood Quality for SST Data

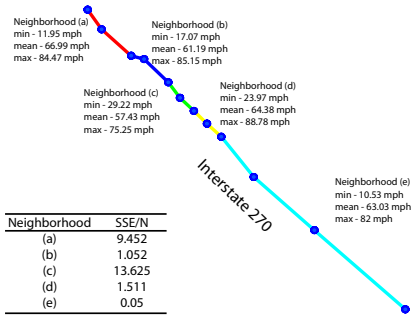
Neighborhood	SSE/n
(a)	0.338
(b)	0.169
(c)	0.286
(d)	0.116

The quality values show that the within-neighborhood error was relatively low and that neighborhoods (b) and (d) had less error than neighborhoods (a) and (c). This suggests that there is more variability in neighborhoods (a) and (c) and that the higher error values suggest that the inner spatial structure of the neighborhoods requires further investigation.

Traffic Data: The graph-based approach also lends itself well to data that is distributed along a directional network such as traffic data. A few modifications had to be made to the algorithm to find distinct neighborhoods in the network data. First, because the nodes and edges are predefined, only linear edges need to be created to successively connect the nodes. To do this, the edges are sorted by the order that they fall on the directional network so that the nodes are connected in sequential order. This removes the complexity of the first step in the algorithm in that a pairwise distance function is not needed to calculate the *sd*, *md*, and mean measurement value. Also, because the edges are predefined by a network, there is no need for thresholds to prune edges that have high spatial and measurement distances. Moreover, because the nodes are connected by only one segment, two similar neighborhoods that are separated by a neighborhood that is not similar are not connected and thus should be represented as separate neighborhoods. Because of this, the result of the clustering algorithm had to be post-processed to assign a new neighborhood ID to similar but unconnected edges. To do this, we looped through the cluster index and assigned nodes to a new neighborhood each time the cluster ID changed.

The algorithm was run on traffic data from 12 sensors located on Interstate 270 South from Frederick, Maryland to the Washington D.C. Beltway (Interstate 495). A one month period of data was used. This consisted of approximately 3,000 records for each sensor. Weekends and holidays were excluded because we wanted the spatial neighborhoods to reflect the peak periods found in the data. Peak periods are typically absent during weekends and holidays. Because of the nature of traffic patterns in terms of periods of jams and free flow, the k-means clustering was run on the minimum, mean, and maximum speed along each edge. The result of the algorithm and the neighborhood quality is shown in Figure 6.

Validation: According to the results the I-270 corridor is characterized by five traffic neighborhoods. Starting in Frederick to the northwest, the first two neighborhoods appear to have a much lower minimum speed. This indicates the presence of at least one very severe traffic jam. As traffic moves to neighborhood (c), the minimum speed speeds up and continues into neighborhood (d) because the highway goes from two to four lanes in this area. Finally in neighborhood (e), the minimum speed indicates the presence of



Neighborhood	SSE/N
(a)	9.452
(b)	1.052
(c)	13.625
(d)	1.511
(e)	0.05

Figure 6: Graph-based neighborhoods for traffic data - I-270 south from Frederick to Washington Beltway

a severe traffic jam neighborhood which reflects congestion in this area caused by the Washington D.C. Beltway. The neighborhood quality is very interesting in this example. It shows that neighborhoods (a) and (c) are different in terms of their within-neighborhood error. This indicates that these neighborhoods need to be investigated further to determine the cause of this result.

3.3 Temporal Interval discovery

The agglomerative temporal interval algorithm was tested on both the SST and traffic datasets. For the traffic and SST data we used an error threshold(λ) of 1 standard deviation from the mean SSE for all intervals and the base interval size was 20.

SST Data The sea surface temperature data was collected at one sensor in the TAO array located at 0 degrees north latitude and 110 degrees west longitude. For this sensor, SST is measured every 15 minutes and in this demonstration, a 10 day period was used from 01/01/2004 to 01/10/2004. This consisted of approximately 1400 measurements. The result of the agglomerative algorithm for the SST data is shown in Figure 7.

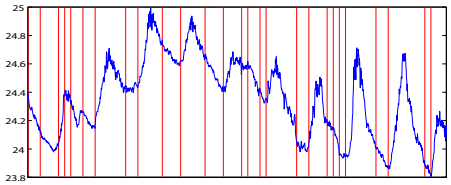


Figure 7: Agglomerative temporal intervals for SST data

Validation: The temporal intervals are validated by the SST time series in the figure. It is evident that the algorithm was able to differentiate peak periods in the SST data from more stable periods. However, it is also evident that in some cases noise in the data causes a 1-0-1 pattern in the binary error classification whereby the base temporal intervals are exposed.

Traffic Data The traffic data was taken from the intersection of east bound US Route 50 and Church Road in Maryland. This data consisted of average speed at 5 minute intervals for the period of 11/03/2007 to 11/10/2007. The size of the dataset was approximately 2100 measurements. The intervals for the traffic data are shown in Figure 8.

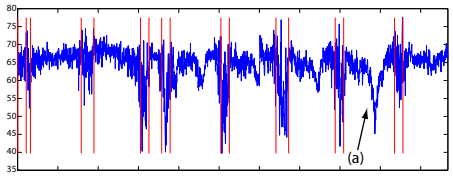


Figure 8: Agglomerative temporal intervals for traffic data

Validation: The algorithm was extremely effective in identifying periods of traffic jams and periods of free flowing traffic. However, the algorithm was not able to isolate the traffic jam in the interval shown in figure 8 (a). This is because this particular period is characterized by a slowly decreasing average speed and thus the SSE for each interval does not exceed λ.

3.4 Spatioemporal Neighborhood discovery

SST Data: Due to the limitation of space we only discuss the results found in SST data. We have employed the spatiotemporal neighborhood algorithm on a ten day time series of SST measurements for 44 sensors in the equatorial Pacific Ocean, totalling 63360 observations. The objective of the analysis is to determine if the algorithm can allow for the discovery of spatiotemporal patterns of sea surface temperature. In this section the preliminary results of this analysis are presented. We first discuss the temporal intervals, spatial neighborhoods and then the Spatiotemporal neighborhoods for some relevant intervals. The temporal intervals discovered by our approach are shown in Figure 9

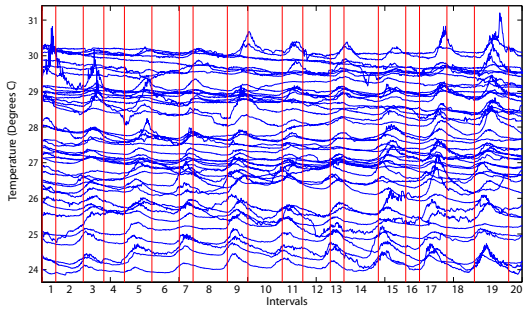


Figure 9: Temporal Intervals for Time Series at all SST Measurement Locations

Validation: The algorithm divided the time series into 20 temporal intervals. In Figure 9 the intervals are plotted as vertical lines on top of the SST time series for all 44 sensors. The intervals show the ability to capture the diurnal pattern of the SST data by generally following the daily warming and cooling pattern that is evident in each time series. However, it can be noticed from the result that there are some sensors where there exists a lag in the diurnal pattern. This is likely a result of the locations being distributed across the Pacific Ocean and time is reported in GMT and thus there exists a delay in the warming of the water based on the rotation of the earth from east to west. From a data mining standpoint, where the peak SST occurs during the interval could then be a predictor of the longitude of the sensor location.

The next part of the algorithm created spatial neighborhoods for each interval. Figure 10 shows the neighborhood

quality for the four resulting neighborhoods at each temporal interval.

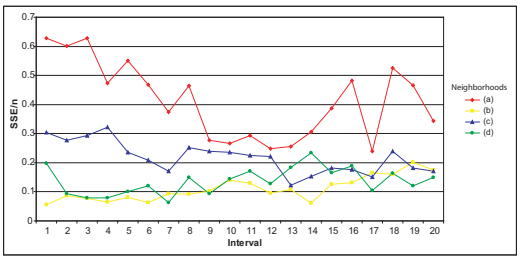


Figure 10: Neighborhood Quality for each Interval

The neighborhood quality changes quite a bit for each interval with neighborhood (a) having the highest within-neighborhood error and neighborhood (b), (c), and (d) generally having a low within-neighborhood error. This indicates that there may be more than one natural grouping in neighborhood 1 during a number of intervals. However from intervals 9 to 13 the error in neighborhood (a) was comparable with neighborhoods (b), (c), and (d). This identifies a challenge in that there may not always be the same number of neighborhoods in a dataset and furthermore, the number of neighborhoods may not always be known a priori. One interesting pattern in the graph occurs between intervals 16 and 19 where the within-neighborhood error of neighborhood 1 goes from very high to low and back to very high. We will use these four intervals to demonstrate the results of the spatiotemporal neighborhoods. Figure 11 shows the neighborhoods formed for these intervals accompanied by a SST satellite image for the approximate time of the interval.

The formation of the spatiotemporal neighborhoods are validated by the pattern of sea surface temperature shown by the satellite image. Figure 11(a),(b),(c), and (d) show the neighborhood formation for each time step. Neighborhood (a) represents the cooler temperature water coming from the south east part of the image. Neighborhood (b) represents the area dominated by the very warm water in the south west part of the image, neighborhood (c) represents the moderate temperature water that is wrapped around neighborhood (a), and neighborhood (d) represents the warmer temperatures that lie between neighborhoods (c) and (d). There are a number of locations where the neighborhoods overlap. Figure 11(e) points out the areas of overlap for each temporal interval. The overlapping areas typically take place along neighborhood boundaries where steep gradients of SST exist. The result also shows areas where change in SST occurs most. The most change occurs in the western four columns of sensors. This trend is validated by the satellite imagery in that it shows that this area is the boundary zone between warm water in the western Pacific and cooler water that travels along the equator.

4. RELATED WORK

Spatial neighborhood formation is a key aspect to any spatial data mining technique ([6, 11, 12, 16, 21, 22]etc.), especially outlier detection. The issue of graph based spatial outlier detection using a single attribute has been addressed in [21]. Their definition of a neighborhood is similar to the

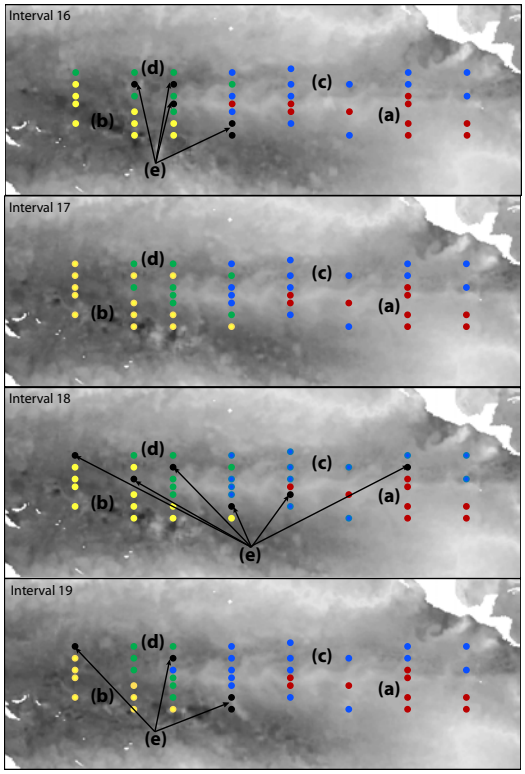


Figure 11: Spatiotemporal Neighborhoods for Intervals 16 - 19 with AVHRR Satellite SST Image

definition of neighborhood graph as in [6], which is primarily based on spatial relationships. However the process of selecting the spatial predicates and identifying the spatial relationship could be an intricate process in itself. Another approach generates neighborhoods using a combination of distance and semantic relationships [1]. In general these neighborhoods have crisp boundaries and do not take the measurements from the spatial objects into account for the generation of the neighborhoods.

The concept of a temporal neighborhood is most closely related to the literature focused on time series segmentation. The purpose of which is to divide a temporal sequence into meaningful intervals. Numerous algorithms [3, 10, 13, 15, 18] have been written to segment time series. One of the most common solutions to this problem applies a piecewise linear approximation using dynamic programming [3]. Three common algorithms for time series segmentation are the bottom-up, top-down, and sliding window algorithms [13]. Another approach, Global Iterative Replacement (GIR), uses a greedy algorithm to gradually move break points to more optimal positions [10]. This approach starts with a k-segmentation that is either equally spaced or random. Then the algorithm randomly selects and removes one boundary point and searches for the best place to replace it. This is repeated until the error does not increase. Nemeth et al. (2003) [18] offer a method to segment time series based on fuzzy clustering. In this approach, PCA models are used to test the homogeneity of the resulting segments. Most recently Lemire [15] developed a method to segment time series using polynomial degrees with regressor-based costs. These approaches primarily focus on approximating

a time series and do not result in a set of discrete temporal intervals. Furthermore, because the temporal intervals will be generated at many spatial locations, a more simplified approach is required.

There has been some work to discover spatiotemporal patterns in sensor data [5,8,9,14,21]. In [21] a simple definition of a spatiotemporal neighborhood is introduced as two or more nodes in a graph that are connected during a certain point in time.There have been a number of approaches that use graphs to represent spatiotemporal features for the purposes of data mining. Time-Expanded Graphs were developed for the purpose of road traffic control to model traffic flows and solve flow problems on a network over time [14]. Building on this approach, George and Shekhar devised the time-aggregated graph [9]. In this approach a time-aggregated graph is a graph where at each node, a time series exists that represents the presence of the node at any period in time. Spatio-Temporal Sensor Graphs (STSG) [8] extend the concept of time-aggregated graphs to model spatiotemporal patterns in sensor networks. The STSG approach includes not only a time series for the representation of nodes but also for the representation of edges in the graph. This allows for the network which connects nodes to also be dynamic. Chan et al. [5] also use a graph representation to mine spatiotemporal patterns. In this approach, clustering for Spatial-Temporal Analysis of Graphs (cSTAG) is used to mine spatiotemporal patterns in emerging graphs.

Our method is the first approach to generate spatiotemporal neighborhoods in sensor data by combining temporal intervals with spatial neighborhoods. Also, there has yet to be an approach to spatial neighborhoods that is based on the ability to track relationships between spatial locations over time.

5. CONCLUSION AND FUTURE WORK

In this paper we have proposed a novel method to identify spatiotemporal neighborhoods using spatial neighborhood and temporal discretization methods as building blocks. We have done several experiments in SST and Traffic data with promising results validated by real life phenomenon.

In the current work we have focused on the quality of the neighborhood which has led to a tradeoff in efficiency. In our future work we would like to extend this work to find high quality neighborhoods in an efficient manner. We will also perform extensive validation of our approach using spatial statistics as a measure of spatial autocorrelation and study the theoretical properties in the neighborhoods we identify. We also intend to use knowledge discovery tasks such as outlier detection to validate the efficacy of our neighborhoods. We will also explore the identification of *critical temporal intervals* where most dramatic changes occur in the spatial neighborhoods.

Acknowledgements
This article has been funded in part by the National Oceanic and Atmospheric Administration (Grants NA06OAR4310243 and NA07OAR4170518). The statements, findings, conclusions, and recommendations are those of the author(s) and do not necessarily reflect the views of the National Oceanic and Atmospheric Administration or the Department of Commerce.

6. REFERENCES

[1] N. R. Adam, V. P. Janeja, and V. Atluri. Neighborhood based detection of anomalies in high dimensional spatio-temporal sensor datasets. In *Proc. ACM SAC*, pages 576–583, New York, 2004.

[2] F. H. Administration. Traffic bottlenecks: A primer focus on low-cost operational improvements. Technical report, United States Department of Transportation, 2007.

[3] R. Bellman and R. Roth. Curve fitting by segmented straight lines. *Journal of the American Statistical Association*, 64(327):1079–1084, 1969.

[4] M. Cane. Oceanographic events during el nino. *Science*, 222(4629):1189–1195, 1983.

[5] J. Chan, J. Bailey, and C. Leckie. Discovering and summarising regions of correlated spatio-temporal change in evolving graphs. *Proc. 6th IEEE ICDM*, pages 361–365, 2006.

[6] M. Ester, H. Kriegel, and J. Sander. Spatial data mining: A database approach. In *5th International Symposium on Advances in Spatial Databases*, pages 47–66, London, UK, 1997. Springer-Verlag.

[7] C. for Advanced Transportation Technology Laboratory. Traffic data extraction software (web based).

[8] B. George, J. Kang, and S. Shekhar. Spatio-temporal sensor graphs (stsg): A sensor model for the discovery of spatio-temporal patterns. In *ACM Sensor-KDD*, August 2007.

[9] B. George and S. Shekhar. Time-aggregated graphs for modeling spatio-temporal networks. *Lecture Notes in Computer Science*, 4231:85, 2006.

[10] J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmaki, and H. Toivonen. Time series segmentation for context recognition in mobile devices. In *ICDM*, pages 203–210, 2001.

[11] Y. Huang, J. Pei, , and H. Xiong. Co-location mining with rare spatial features. *Journal of GeoInformatica*, 10(3), 2006.

[12] Y. Huang, S. Shekhar, and H. Xiong. Discovering colocation patterns from spatial data sets: A general approach. *IEEE TKDE*, 16(12):1472–1485, 2004.

[13] E. Keogh and P. Smyth. A probabilistic approach to fast pattern matching in time series databases. *Proc. 3rd ACM KDD*, pages 24–20, 1997.

[14] E. Kohler, K. Langkau, and M. Skutella. Time-expanded graphs for flow-dependent transit times. *AlgorithmsŮESA*, 2:599–611, 2002.

[15] D. Lemire. A better alternative to piecewise linear time series segmentation. *SIAM Data Mining 2007*, 2007.

[16] C. Lu, D. Chen, and Y. Kou. Detecting spatial outliers with multiple attributes. In *15th IEEE International Conference on Tools with Artificial Intelligence*, page 122, 2003.

[17] M. McPhaden. Genesis and evolution of the 1997-98 el nino. *Science*, 283:950–954, 1999.

[18] S. Nemeth, J. Abonyi, B. Feil, and P. Arva. Fuzzy clustering based segmentation of time-series, 2003.

[19] NOAA. Tropical atmosphere ocean project (http://www.pmel.noaa.gov/tao/jsdisplay/).

[20] E. Rasmusson and J. Wallace. Meteorological aspects of the el nino/southern oscillation. *Science*, 222(4629):1195–1202, 1983.

[21] S. Shekhar, C. Lu, and P. Zhang. Detecting graph-based spatial outliers: algorithms and applications (a summary of results). In *7th ACM SIG-KDD*, pages 371–376, 2001.

[22] P. Sun and S. Chawla. On local spatial outliers. In *4th IEEE ICDM*, pages 209–216, 2004.

Real-Time Classification of Streaming Sensor Data

Shashwati Kasetty, Candice Stafford[a], Gregory P. Walker[a], Xiaoyue Wang, Eamonn Keogh

Department of Computer Science and Engineering
Department of Entomology[a]

University of California, Riverside
Riverside, CA 92521

{kasettys, xwang, eamonn}@cs.ucr.edu, {staffc01, gregory.walker}@ucr.edu

ABSTRACT

The last decade has seen a huge interest in classification of time series. Most of this work assumes that the data resides in main memory and is processed offline. However, recent advances in sensor technologies require resource-efficient algorithms that can be implemented directly on the sensors as real-time algorithms. In this work we show how a recently introduced framework for time series classification, time series bitmaps, can be implemented as ultra efficient classifiers which can be updated in constant time and space in the face of very high data arrival rates. We demonstrate our results from a case study of an important entomological problem, and further demonstrate the generality of our ideas with examples from robot and cardiology data.

Categories and Subject Descriptors

H.2.8 [**DATABASE MANAGEMENT**]: Database Applications —Data mining

Keywords

Sensor Data, Data Mining, Time Series Classification, Streaming Data, Real-time Algorithms

1. INTRODUCTION

The last decade has seen a huge interest in classification of time series [12][8][14]. Most of this work assumes that the data resides in main memory and is processed offline. However recent advances in sensor technologies require resource-efficient algorithms that can be implemented directly on the sensors as real-time algorithms. In this work we show how a recently introduced framework for time series classification, *time series bitmaps* [14], can be implemented as ultra efficient classifiers which can be updated in constant time in the face of very high data arrival rates. Moreover, motivated by the need to be robust to concept drift, and to spot new behaviors with minimal lag, we show that our algorithm can be amnesic and is therefore able to discard outdated data as it ceases to be relevant.

In order to motivate our work and ground our algorithms we begin by presenting a concrete application in entomology which we will use as a running example in this work. However in our experiments we will consider a broader set of domains and show results from applications across various fields.

1.1 Monitoring Insects in Real Time

In the arid to semi-arid regions of North America, the beet leafhopper (*Circulifer tenellus*), shown in Figure 1, is the only known vector (carrier) of curly top virus, which causes major economic losses in a number of crops including sugarbeet, tomato, and beans [7]. In order to mitigate these financial losses, entomologists at the University of California, Riverside are attempting to model and understand the behavior of this insect [19].

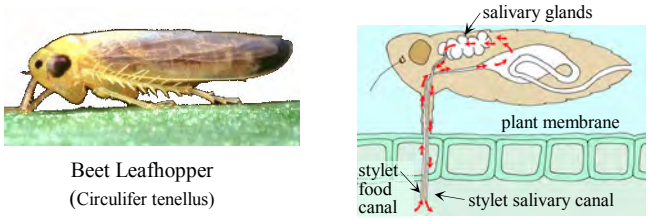


Figure 1: *left*) The insect of interest. *right*) Because of the circulatory nature of the insects feeding behavior, it can carry disease from plant to plant

It is known that the insects feed by sucking sap from living plants, much like the mosquito sucks blood from mammals and birds. In order to understand the insect's behaviors, entomologists can glue a thin wire to the insect's back, complete the circuit through a host plant and then measure fluctuations in voltage level to create an Electrical Penetration Graph (EPG) as shown in Figure 2.

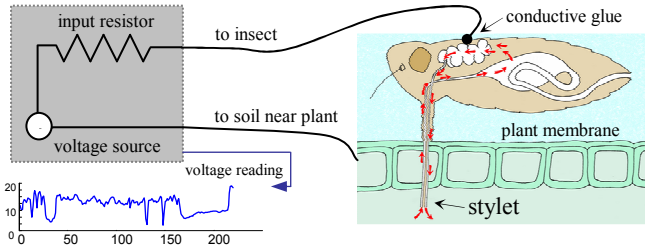


Figure 2: A schematic diagram showing the apparatus used to record insect behavior

This method of recording the insect's behavior appears to be capturing useful information. That is to say, skilled entomologists have been able to correlate various behaviors they have observed by directly watching the insects, with simultaneously recorded time series. However the abundance of such data opens a plethora of questions, including:

- Can we automatically classify the insect's behavior into the correct class? Can we detect when the beet leafhopper is in the non-ingestion pathway phase, the phloem ingestion phase, the xylem and mesophyll ingestion phase, or the non-probing phase when it is resting on the leaf surface?

Being able to detect these phases automatically would save many hours of time spent by entomologists to analyze the EPGs manually. This could open avenues for detecting new behaviors

that entomologists have not been able to model thus far. Detecting these patterns in real-time could also eliminate the need for storing large amounts of sensor data to process and analyze at a later time. To be truly real-time, the scheme must be algorithmically time and space efficient in order to deal with the high data rates sensed by the sensors. It should be able to detect patterns in data as the data is sensed.

We propose to tackle this problem using Time Series Bitmaps (TSB) [14]. In essence TSBs are a compact summary or signature of a signal. While TSB's have been shown to be useful for time series classification in the past [14][13], the fundamental contribution of this work is to show that we can maintain TSBs in constant time, allowing us to deal with very high rate data.

We defer a detailed discussion of TSBs until Section 3; however, Figure 3 gives a visual intuition of them, and their utility.

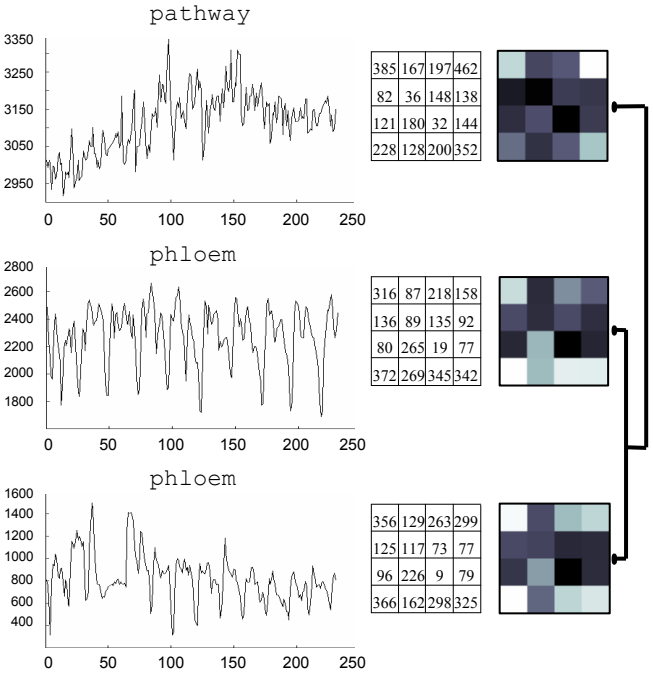


Figure 3: Three traces of insect behavior clustered using time series bitmaps. The square matrices are raw counts. The bitmaps correspond to these raw counts scaled between 0 and 256 then mapped to a colormap. Note that the y-axis values are relative, not absolute, voltage values.

The raw signals have information extracted from them regarding the frequencies of short "sub-patterns". These raw counts of sub-patterns are recorded in a square matrix, and the Euclidean distance between these matrices can effectively capture the similarity of the signals, which can then be used as an input into classification, clustering and anomaly detection algorithms [13][14]. While it is not necessary for classification algorithms, we can optionally map the values in the matrices to a colormap to allow human subjective interpretation of similarity, and higher level interpretation of the data.

Time Series Bitmaps require only a small amount of memory to store the values of the square matrix. Since these square matrices are updated in real-time, the amount of memory needed is a small constant. Furthermore, as we shall show, the operations on these matrices are also done in constant time.

The rest of the paper is organized as follows. Section 2 describes background and related work in this field. In Section 3, we provide a review of SAX and TSBs. In Section 4, we describe our algorithm in detail. Section 5 contains results from experiments. Section 6 briefly describes future work in this area. Section 7 contains the conclusion.

2. BACKGROUND AND RELATED WORK

To the best of our knowledge, the proposed method of maintaining Time Series Bitmaps (TSBs) in constant time and space per update is novel. Work has been done towards deploying algorithms on sensors that use the Symbolic Aggregate Approximation (SAX) representation [22][17], and as we shall see, SAX is a subroutine in TSBs, however, neither of the two works uses TSBs.

TSBs are aggregated representations of time series. Another aggregation scheme is presented in [15], where data maps are created that represent the sensory data as well as temporal and spatial details associated with a given segment of data. However, these data maps are not analyzed in real-time, but deposited at sink nodes that are more powerful for pattern analysis.

In [21], the authors introduce an anomaly detection program using TSBs and SAX. However, the authors do not update the TSBs in constant time, but recalculate them from scratch for every TSB. In our work, we introduce a way to maintain these TSBs in constant time without having to recalculate them from scratch, saving time that makes our algorithm truly real-time. Moreover, we tackle the problem of classification, while [21] provides an algorithm for anomaly detection.

Finally, there are dozens of papers on maintaining various statistics on streaming, see [4] and the references therein. However none of these works address the task maintaining a class prediction in constant time per update.

3. A REVIEW OF SAX/BITMAPS

For concreteness we begin with a review of the time series bitmap representation. For ease of exposition, we begin with an apparent digression: How can we summarize long DNA strings in constant space?

Consider a DNA string, which is a sequence of symbols drawn from the alphabet {**A**, **C**, **G**, **T**}. DNA strings can be very long. For example the human mitochondrial DNA has 16,571 such symbols, beginning with **GATCACAGGTCTATCACCC**… and ending with …**ACATCACGATG**. Given the great length of DNA strings a natural question is how can we summarize them in a compact representation? One approach would be to map a DNA sequence to a matrix of four cells based on the frequencies of each of the four possible base pairs. This produces a numeric summary; we can then further map the observed frequencies to a linear colormap to produce a visual summary as shown in Figure 4.

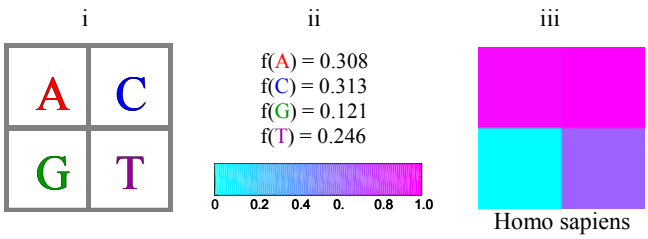


Figure 4: i) The four DNA base pairs arranged in a 2 by 2 grid. ii) The observed frequencies of each letter can be indexed to a colormap as shown in iii.

Note that in this case the arrangement of the four letters is arbitrary, and that the choice of colormap is also arbitrary.

We begin by assigning each letter a unique key value, k:

$\mathbf{A} \rightarrow 0$	$\mathbf{C} \rightarrow 1$	$\mathbf{G} \rightarrow 2$	$\mathbf{T} \rightarrow 3$

We can control the desired number of features by choosing l, the length of the DNA words. Each word has an index for the location of each symbol, for clarity we can show them explicitly as subscripts. For example, the first word with $l = 4$ extracted from the human mitochondrial DNA is $\mathbf{G_0A_1T_2C_3}$. So in this example we would say k_0 is **G**, k_1 = **A**, k_2 = **T** and k_l = **C**.

To map a word to a bitmap we can use the following equation to find its row and column values:

$$col = \sum_{n=0}^{l-1} (k_n * 2^{l-n-1}) \bmod 2^{l-n}, \quad row = \sum_{n=0}^{l-1} (k_n div 2) * 2^{l-n-1}$$

Figure 5 shows the mapping for l = 1, 2 and (part of) 3.

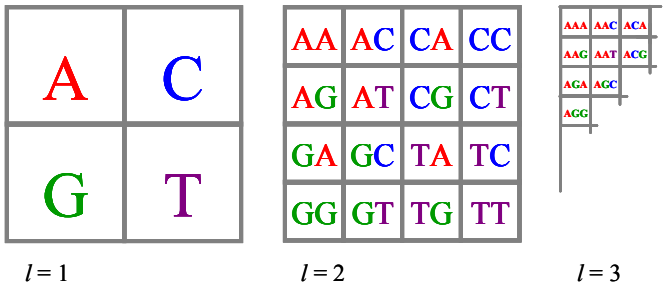


Figure 5: The mapping of DNA words of l = 1, 2 and 3. (The colors of the text are just to allow visualization of the mapping).

If one examines the mapping in Figure 5, one can get a hint as to why a bitmap for a given species might be self-similar across different scales. For example note that for any value of l, the top column consists only of permutations of A and C, and that the two diagonals consist of permutations of A and T, or G and C. Similar remarks apply for other rows and columns.

In the rest of this paper, we use the alphabet {a,b,c,d} and we choose to use bitmaps of size 4x4 or l = 2. Figure 6 below was created using this alphabet, and a different colormap than the DNA example. The icons shown here were generated from a subsequence of a non-probing behavior waveform of the beet leafhopper. Refer to section 4 for details on each beet leafhopper behavior.

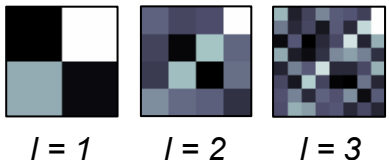


Figure 6: The icons created for a subsequence of a non-probing behavior waveform for the beet leafhopper at every level from l = 1 to 3.

Having shown how we can convert DNA into a bitmap, in the next section we show how we can convert real-valued time series into pseudo DNA, to allow us to avail of bitmaps when dealing with sensors.

3.1 Converting Time Series to Symbols

While there are at least 200 techniques in the literature for converting real valued time series into discrete symbols [1], the SAX technique of Lin et. al. is unique and ideally suited for our purposes [16]. The SAX representation is created by taking a real valued signal and dividing it into equal sized sections. The mean value of each section is then calculated. This produces a reduced dimensionality piecewise constant approximation of the data. This representation is then discretized in such a manner as to produce a word with approximately equi-probable symbols. Figure 7 shows the first 64 data points of the phloem phase waveform in the bottom of Figure 3 after converting it to a discrete string.

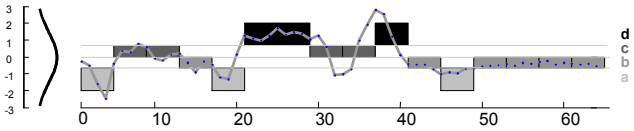


Figure 7: A real valued time series being discretized into the SAX word accbaddccdbabbbb.

Note that because we can use SAX to convert time series into a symbolic string with four characters, we can then trivially avail of any algorithms defined for DNA, including the bitmaps introduced in the last section.

SAX was first introduced in [16], and since then it has been used to represent time series in many different domains including automated detection and identification of various species of birds through audio signals [9], and analysis of human motion [2], telemedicine and motion capture analyses.

4. OUR ALGORITHM IN CONTEXT

We demonstrate our results on a case study for an important entomological problem, and then further demonstrate the generality of our ideas with examples from robot and cardiology data.

Our algorithm uses SAX (Symbolic Aggregate Approximation), a symbolic representation for time series data [10], to summarize sensor data to a representation that takes up much less space, yet captures a signature of the local (in time) behavior of the time series. We further compact the data by aggregating the SAX representations of segments of data to create a square matrix of fixed length or Time Series Bitmaps (TSBs) [14][13].

We introduce novel ways to maintain these TSBs in constant time. These optimizations make our algorithm run significantly faster, use very little space, and produce more accurate results, while being amnesic and using the most recent and relevant data to detect patterns and anomalies in real-time. With these

improvements in time and space requirements, this algorithm can be easily ported to low-power devices and deployed in sensor networks in a variety of fields.

4.1 Entomology Case Study

As noted in Section 1.1, entomologists are studying the behavior of beet leafhopper (*Circulifer tenellus*) by gluing a thin wire to the insect's back, completing the circuit through a host plant and then measuring fluctuations in voltage level to create an Electrical Penetration Graph (EPG). This method of recording the insect's behavior appears to be capturing useful information. Skilled entomologists have been able to correlate various behaviors they have observed by directly watching the insects, with simultaneously recorded time series. However, the entomologists have been victims of their own success. They now have many gigabytes of archival data, and more interestingly from our perspective, they have a need for *real-time* analyses. For example, suppose an entomologist has a theory that the presence of carbon dioxide can suppress a particular rarely seen but important insect behavior. In order prove this theory, the entomologist must wait until the behavior begins, then increase the concentration of carbon dioxide and observe the results. If we can successfully classify the behavior in question automatically, we can conduct experiments with hundreds of insects in parallel, if we cannot automate the classification of the behavior, we are condemned to assigning one entomologist to watch each insect – an untenable bottleneck. Before giving details of our algorithm in Section 4.2, we will provide some examples and illustrations of the types of behaviors of interest.

4.1.1 Characteristic Behaviors to Classify

The beet leafhopper's behavior can be grouped into 3 phases of feeding behavior and 1 phase of non-probing or resting behavior, making this a classification problem of 4 classes. There are several other behaviors of the beet leafhopper which have not yet been identified, and we will exclude these from our experiments. The original measurements were made in terms of voltage. However, only the relative values matter, so no meaning should be attached to the absolute values of the plots. Note that we have made an effort to find particularly clean and representative data for the figures. In general the data is very complex and noisy.

4.1.1.1 Class 1 - Pathway

There is no ingestion in this phase but it is believed that it occurs prior to other ingestion behaviors. During the initial stages of feeding, pathway waveforms are produced. There are several variations of pathway phase waveforms, each of which have varied characteristics. One variation of pathway is quite similar to phloem ingestion and non-probing behavior in that it is characterized by low amplitude fluctuations, which makes this variation difficult to classify. In our work, we will consider all the variations together as 1 general pathway phase behavior. An example pathway phase waveform is shown in Figure 8.

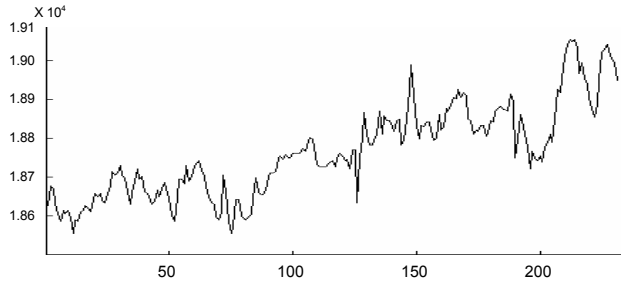


Figure 8: A Pathway Phase Waveform

4.1.1.2 Class 2 – Phloem Ingestion

In this phase, the beet leafhopper is seen to be ingesting phloem sap. The waveforms in this phase are known to have low amplitude fluctuation and occur at low voltage levels. There are varied behaviors among phloem ingestion phase waveforms; however, in this work we only classify the general phloem ingestion phase which encompasses all sub-behaviors. An example phloem ingestion phase waveform is shown in Figure 9. Note that this particular waveform has characteristic reoccurring "spikes", but the mean value can wander up and down in a very similar manner to the *wandering baseline* effect in cardiology [3]. This drift of the mean value has no biological meaning, neither in cardiology nor here. However, as we shall see, this wandering baseline effect can seriously degrade the performance of many classic time series classification techniques.

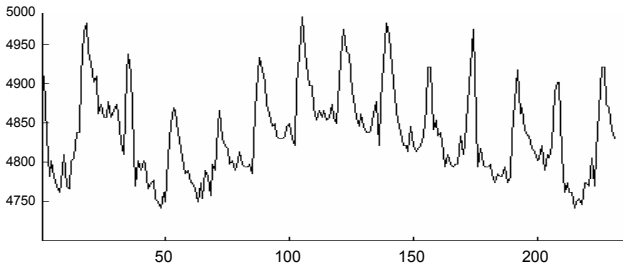


Figure 9: A Phloem Ingestion Phase Waveform

4.1.1.3 Class 3 – Xylem / Mesophyll Ingestion

In this phase, the beet leafhopper is seen to be ingesting xylem sap. Occasionally, it is seen to be ingesting mesophyll sap. However, the waveforms of the two are indistinguishable. For entomologists, this phase is easiest to recognize since visually the waveform has very characteristic and typical features. The voltage fluctuation has high amplitude and a very regular repetition rate. An example xylem / mesophyll ingestion waveform is shown in Figure 10.

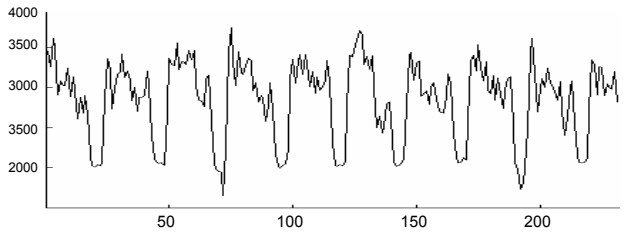


Figure 10: A Xylem / Mesophyll Ingestion Waveform

4.1.1.4 Class 4 – Non-Probing / Resting

In this phase, the beet leafhopper is resting on the surface of the leaf. Sometimes the beet leafhopper may move around, and the insect's walking and grooming behaviors can cause large

fluctuations in voltage level. Usually, when the insect is resting, the fluctuation levels are low and somewhat flat. An example non-probing phase waveform is shown in Figure 11.

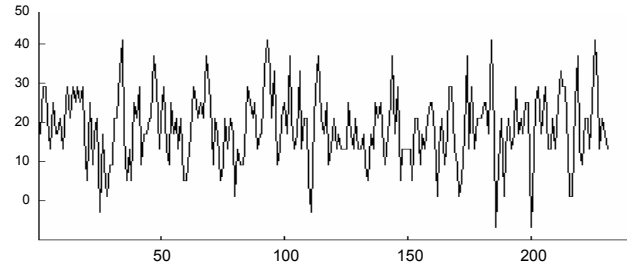


Figure 11: A Non-Probing / Resting Ingestion Waveform

4.1.2 Classification in Real-time

With the abundance of data available, it becomes impractical and time consuming for a human to analyze and classify each of these behaviors visually. If we can automatically classify this behavior using an efficient classification algorithm, it could save many hours of the entomologists' time. The benefits multiply if the behaviors can be captured in real-time as they occur, without having to record many hours of data for offline processing later.

Our algorithm is able to handle this data despite it being erratic and unpredictable, which is the case with most sensor data. We outline and describe our algorithm below, and in Section 5, we show results from experiments in which we consider streaming data, and classify the behavior that is occurring in real-time.

4.2 Maintaining TSBs in Constant Time

While SAX [10] forms the basis of our algorithm and we use TSBs [14] to aggregate the raw data, the novel contribution of our work is the way we maintain the TSBs in constant time, enabling tremendous improvement in the input rate we can handle, as well as opening up the possibility of creating an efficient classifier that can be deployed in low-power devices. These improvements allow us to process data at a high rate, while still classifying and producing results in real-time.

Because long streams of data kept in memory will become outdated and meaningless after some time, and must be discarded periodically, our algorithm is amnesic, maintaining a certain constant amount of history of data at all times. This is especially useful when there is a continuous stream of data in which there are transitions from one class to another. Our algorithm can capture these changes since the classifier is not washed out by hours of outdated data that is not relevant to the current state. Furthermore, by choosing an appropriate window length it can capture these changes with minimal lag. The pseudocode for maintaining TSBs in constant time and classifying them is outlined in Table 1.

Table 1: Maintaining TSBs in Constant Time

```
   Function classifyTSBs(N,n,a,historySize)
1  historyBuffer[historySize][n] // Holds SAX words
2  curTimeSeries[N]  // Holds current sliding window
3  curTSB[a times a] = 0 // Initialize curTSB to 0s
4  input = getInput()
5  while curTimeSeries.size() < N and input != EOF:
6     curTimeSeries.append(input)
7     input = getInput()
8  curSAXWord = sax(curTimeSeries,N,n,a) // Alphabet size a
9  incrementTSB(curTSB,curSAXWord)
10 historyBuffer.append(curSAXWord)
11 while historyBuffer.size() < historySize and input != EOF:
12    curTimeSeries.pop() // Pop() removes oldest element
13    curTimeSeries.append(input)
14    curSAXWord = sax(curTimeSeries,N,n,a)
15    incrementTSB(curTSB,curSAXWord)
16    historyBuffer.append(curSAXWord)
17    input = getInput()
18 classify(curTSB) // Classifies TSB after history collected
19 while input != EOF:
20    curTimeSeries.pop()
21    curTimeSeries.append(input)
22    curSAXWord = sax(curTimeSeries,N,n,a)
23    removedWord = historyBuffer.pop() // Remove oldest word
24    decrementTSB(curTSB,removeWord)
25    historyBuffer.append(curSAXWord) // Append newest word
26    incrementTSB(curTSB,curSAXWord)
27    classify(curTSB) // Classifies TSB after each new input
28    input = getInput()
```

The input parameters to this algorithm are the 3 SAX parameters of `N`, `n` and `a`, along with the `historySize` parameter. The algorithm begins by creating two circular arrays that will hold the current data being processed and stored (lines 1-2). The `curTSB` array of size `a times a` will hold the Time Series Bitmap counts.

The `curTimeSeries` array (line 2) holds the current sliding window of data that will be converted to a SAX word. The SAX parameters that we use in the beef leafhopper problem are `N=32` and `n=16`. This corresponds to a sliding window size of 32 data points that will be converted to a SAX word of 16 characters. These parameters are fixed constants in our algorithm but can be changed for other applications if necessary, although fine-tuning parameters too much could lead to a problem of overfitting the data [12]. The alphabet size we use is `a=4` (lines 8, 14, 22) in order to produce a square time series bitmap of size 4x4 stored here as an array of size 16 (line 3).

The `historyBuffer` (line 1) is a two dimensional array that will hold the most recent SAX words in it. The number of SAX words it holds is specified by `historySize`. We have fixed this to be 200 in our implementation for the beet leafhopper. We estimated this by visually inspecting graphs and noticing that it is a large enough timeframe of data to indicate the type of behavior being exhibited. If more points are needed to classify, the `historySize` can be increased. Conversely, if less points are needed or if memory is severely scarce, the `historySize` can be decreased. We have refrained from fine tuning this parameter to prevent overfitting but our experiments suggest that we can obtain good results over a relatively large range of values [12].

Initially, the two buffers need to be filled as long as there is input. This is done in the first two while loops. Every time the

curTimeSeries buffer is filled, it needs to be converted to a SAX word, and curTSB needs to be updated by incrementing the appropriate positions. Once the historyBuffer is filled, we can proceed with real-time classification. The first TSB representing the initial historyBuffer is classified (line 18). Then for each new input, the TSB is computed and classified (line 27). The classifier we use is one nearest neighbor with Euclidean distance between the TSBs as the distance measure [14].

4.2.1 Optimizations in Time and Space

The optimizations we propose to save time and space arise from the observation that a new TSB need not be created for each new input. We can update the TSB by removing the oldest SAX word in historyBuffer, decrementing the appropriate fields in the TSB for the removed word, appending the newest word to the historyBuffer, and updating the TSB by incrementing its fields for the new word. Similarly, curTimeSeries need not be refilled each time there is a new input. The oldest value in curTimeSeries can be removed and the new input can be added. Note that in the implementation, curTimeSeries and historyBuffer need to be circular arrays in order to perform updates in constant time. Figure 12 illustrates lines 19-28.

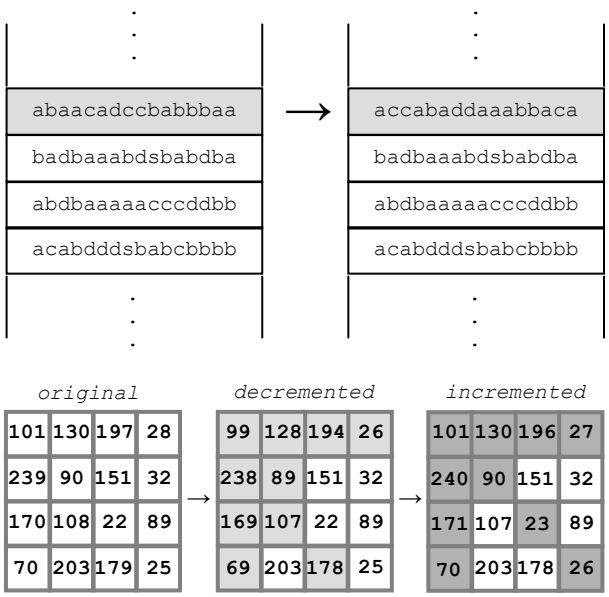


Figure 12: Maintaining TSBs in Constant Time

Figure 12 shows the status of the historyBuffer before and after a new input is processed. The new input mapped to the SAX word accabaddaaabbaca, takes the place of the oldest SAX word in the circular array historyBuffer, which in this case is the word abaacadccbabbbaa. This change needs to be reflected in the array curTSB. Figure 12 shows the status of the curTSB after the change. The appropriate values of the substring counts for the substrings in the SAX word being removed need to be decremented. Then, the values need to be incremented for the substrings of the new SAX word added.

4.2.2 A Special Case – Standard Deviation 0

Theoretically, the sum curTSB should remain the same after the oldest SAX word in removed and the new SAX word is inserted into the historyBuffer. This is because the two words have the same length which means that they have the same number of 2-letter substrings. However, practically, the sum might decrease. This happens when the standard deviation of the values in curTimeSeries is 0. For real valued time series a standard deviation of zero clearly corresponds to a system failure of some kind, for example a broken wire. Our algorithm simply ignores such data.

4.2.3 Training the Classifier

To classify a given segment of data into a particular class, we need to create reference TSBs. After preprocessing the data and performing the necessary conversions of format to plain ASCII text, we proceed to convert the streams of data to bitmaps using the same algorithm as in Table 1, and store these bitmaps in a file.

We could use all the annotated data as training data, however this has two problems. First, the sheer volume of data would make both the time and space complexity of nearest neighbor classification untenable. Second, the training data is noisy and complex and it may have mislabeled sections. Our solution is to do data editing, also known as prototype selection, condensation, etc [18]. In essence, our idea is to cluster the data and choose a small number of cluster centers as prototypes for the dominant class in that cluster.

We begin by randomly sampling the data to reduce the training set size while still maintaining the original distribution of the data. Once we have randomly sampled, we can cluster the data to find good representative models for each class to use in the classification. Since the Time Series Bitmap is an aggregation of time series subsequences, it is necessary that the bitmaps be randomly selected to avoid the problems that arise with clustering time series subsequences [11]. We proceed to cluster the bitmaps by using KMeans. The best centroids (i.e the ones that have the purest class labels) are computed for each class and these centroids make up the final training classifiers that are provided to the real-time algorithm in Table 1. The pseudocode for finding the training TSBs for each class X is presented in Table 2.

Table 2: Finding Training TSBs for Each Class

```
    Function findOptimalClusters(TSBsClassX)
1   TSBsClassX = minMaxNormalize(TSBsClassX)
2   distances[99] // Sums of distances to nearest centroid
3   centroidLists[99] // Array of lists of centroids
4   for k = 2 : 100 // k is the number of clusters
5      [clusters,sumd] = kmeans(TSBsClassX,k)
6      minDist = sum(sumd)
7      minClusters = clusters
8      for i = 2 : 5 // Running kmeans on a given k 5 times
9         [clusters,sumd] = kmeans(TSBsClassX,k)
10        if sum(sumd) < minDist
11           minDist = sum(sumd)
12           minClusters = clusters
13     distances.append(minDist) // Best distances
14     centroidLists.append(clusters) // Best centroid lists
15  for i = 1 : 98 // Stops when change in distance is < 1%
16     curDist = distances[i]
17     if(curDist - distances[i+1]) / curDist < 0.01
18        return i,centroidLists[i]
19  return null,null
```

Before beginning the KMeans clustering, we first normalize the data using min-max normalization, in which every TSB is scaled to be between 0 and 255 (line 1). For this part of the algorithm, it would suffice to scale between 0 and 1, but since the TSBs could potentially be mapped to a colormap for visualization, it is more

suitable to scale it between 0 and 255. The results would be the same regardless of which scaling is used as long as the data is min-max normalized.

Cluster sizes from 2 to 100 are tested (lines 4-14), and each test is run 5 times (lines 8-12). For each cluster size, the resulting centroids from the best test run are recorded (lines 11-14). The best test run is the one in which the sum of the distances from each instance to the nearest centroid is the lowest. The next step is to find the best number of clusters. Although the sum of the distance to the nearest centroid will decrease monotonically as the number of clusters increases, the distance change becomes negligible after some time. It is more efficient to choose fewer number of clusters since it reduces the size of the final training set created from the centroids of these clusters. To find the best cluster size, we compute the difference in distance sum between two consecutive cluster sizes starting from cluster size 2, and terminate the search when this difference is less than 1% (lines 17-18). The best cluster size and the corresponding centroids for that cluster size are returned.

There may be rare cases when the algorithm does not find such a cluster size, and in that case the return values would be null. In such a case, the difference threshold of 1% can be increased, and the algorithm can be run again.

5. EXPERIMENTAL RESULTS

In this section we describe detailed results from the beet leafhopper problem, and in order to hint at the generality of our work, we briefly present results from experiments on robot and cardiology datasets. To aid easy replication of our work, we have placed all data at: http://acmserver.cs.ucr.edu/~skasetty/classifier.

5.1 Beet Leafhopper Behavior Dataset

The classification results of the beet leafhopper behavior problem largely agree with expectations. Our algorithm classifies classes known to be easy with high accuracy, and does reasonably well on more difficult classes. The results are presented in Table 3.

Table 3: Classification Accuracies
Beet Leafhopper Problem

Class	Accuracy	# of TSBs
Pathway	42.56%	610,538
Phloem	64.93%	1,241,560
Xylem/Mesophyll	71.94%	1,432,586
Non-Probing	95.03%	412,130
Overall	**67.31%**	**3,696,814**
Default (Overall)	*38.75%*	*3,696,814*

It is important to note that we classified all 3,696,814 examples available, without throwing out the difficult or noisy cases.

As described in Section 4, the pathway phase behavior has several variations, and in our classification, we grouped all of these sub-behaviors together as a single pathway phase behavior. The waveforms of these sub-behaviors vary quite a bit, so it is expected that the classification accuracy may not be as high as the other classes. Similarly, the phloem ingestion phase behavior has several varieties, and we grouped these together as well. However, the phloem phase behavior is classified correctly 64.93% of the time, which is much higher than the phloem phase accuracy of 42.56%.

The xylem/mesophyll ingestion phase is easiest for entomologists to detect, and as expected, our classifier mirrors this, classifying accurately 71.94% of the data. The non-probing behavior is clearly different from the other three behaviors, because the insect is simply resting on the leaf, moving around or grooming during this phase. As expected, it was easiest to detect this behavior, with a classification accuracy of 95.03%.

We compared our algorithm with several competitors, including the following using the min-max normalized time series subsequences: Euclidean distance [11], the distance between the energy of the two Fourier coefficients with the maximum magnitude [6], the distance between the energy of the first 10 Fourier coefficients [6], and the difference in variance. We made every effort to find the best possible settings for competitors that have parameters.

Due to the slow running times of the other algorithms, we reduced the size of the test set by randomly selecting 1% of the testing data from each class. Since the training data was smaller (with 15,015 instances), we selected 10% of this data randomly to create the new training set. The results of all the classes together are presented in Table 4.

Table 4: Accuracy Comparison
Beet Leafhopper Problem (All Classes Together)

Classifier	Accuracy
Default Rate	*38.75%*
TSBs with KMeans	**68.94%**
Euclidean Distance	41.84%
distFFT with maxMag	41.59%
distFFT with firstCoeff	42.41%
Variance Distance	40.65%

It is clear that overall, across all classes, our algorithm performs much better than the other algorithms we tested it against. It beats the second best algorithm by more than 26%. The distribution of the 4 classes in the test set is not equal. After sampling 1% from each class, we have 6,105 instances for pathway, 12,416 instances for phloem ingestion, 14,326 instances for xylem or mesophyll ingestion and 4,121 instances for non-probing. Therefore, the default classification rate is 38.75%. Clearly, the other algorithms are only marginally better than default. Our algorithm beats the default rate by more than 30%.

We used this downsampled data on our algorithm to create the confusion matrix in Figure 13.

Test Class

Predicted Class	Pathway	Phloem	Xylem / Mesophyll	Non-Probing
Pathway	**34.37%**	12.22%	18.74%	2.84%
Phloem	23.54%	**72.72%**	6.76%	2.81%
Xylem / Mesophyll	6.96%	4.55%	**73.12%**	0.07%
Non-Probing	35.14%	10.51%	1.39%	**94.27%**

Figure 13: Confusion matrix showing test class versus predicted class

The diagonal shows the accuracy of our classifier on each class. The pathway phase is the only behavior our algorithm does not accurately classify above the default rate. The rest of the classes classify well above the default rate.

This confusion matrix illuminates several complexities and characteristics of this dataset that make classifying it particularly challenging. It is interesting to note that very rarely is a class misclassified as the xylem / mesophyll ingestion behavior. This is because the xylem / mesophyll ingestion phase has a very distinct waveform characterized by constant repetition and high voltage fluctuations as we described in Section 4.

The pathway waveforms are particularly difficult to model, and therefore, difficult to classify correctly due to the high variations in the sub-behaviors of this class. They are misclassified at a very high rate as non-probing or phloem ingestion. This is because there is a particular variation of pathway that has the low amplitude voltage fluctuation characteristic of non-probing (during resting) and phloem ingestion waveforms, and this variation of pathway is the most frequent in our dataset.

It is natural that the non-probing or resting behavior waveforms classify correctly at a very high percentage (94.27% in this case) since the other behaviors are all related to feeding. Although pathway waveforms are some times misclassified as non-probing behavior, the converse is not true. We attribute this to the low number of variations within the non-probing behavior class. The algorithm only needs to model two types of waveforms for this class. The waveforms are somewhat flat with low voltage fluctuations when the insect is resting, and high fluctuations are typical when the insect is grooming. On the other hand, the pathway class has four distinct sub-behaviors making it much more difficult to model. Moreover, as mentioned above, the most confusing variation of pathway waveforms is the most frequent variation in our dataset.

The data used to generate the results in Figure 13 follow a similar overall trend as the results in Table 3 generated from running our algorithm on a test set 100 times as large and a training set 10 times as large. The larger the dataset, the more difficult it is to classify due to the unpredictable and erratic behavior in sensor data. Here, we can see that our algorithm scales well and maintains accuracy rates overall as the dataset grows in size.

5.2 Robot Dataset

To illustrate the generality of our algorithm, we have run additional experiments on different datasets using a similar setup and procedure as for the beet leafhopper behavior classification problem. The same parameters were used as well.

The Sony AIBO is a small quadruped robot that comes equipped with a tri-axial accelerometer. We obtained accelerometer data for the AIBO moving on various surfaces: on a field, on a lab carpet and on cement [20]. We applied our algorithm to this dataset to see if it could detect which surface the robot was moving on for a given waveform. Like the beet leafhopper dataset, we passed the streams of data for each surface to generate the TSBs, ran the training algorithm in Table 2 on randomly sampled TSBs from the training data streams, and classified the TSBs using the one nearest neighbor algorithm with Euclidean distance between the TSBs as the distance measure. Table 5 shows the results.

Table 5: Classification Results from Robot Dataset

	Accuracy
Default Rate across 3 classes	***38.42%***
X-Axis Data across 3 classes	73.36%
Y-Axis Data across 3 classes	60.84%
Z-Axis Data across 3 classes	62.97%

Like the beet leafhopper dataset, the distribution of the number of data points in each class of the robot accelerometer dataset is also unequal. The default accuracy rate was calculated to be 38.42%. For all three, the x-axis, y-axis and z-axis acceleration data, our algorithm clearly beats the default rate, with the x-axis data being most easy to classify.

5.3 Cardiology Dataset

In this experiment, we used two time series of heartbeat data from two different individuals (*www.physionet.org)*. These two heartbeat time series clearly look different if examined visually. We tested our algorithm on this data to see how well it performs. In this case, the number of data points is equal in the two classes. Therefore, default classification accuracy is 50%. Our algorithm was able to get an accuracy rate of 97.29%. This is a relatively easy dataset to classify compared to the robot and beet leafhopper datasets.

Since it is easy to visualize, we use this dataset to illustrate how our algorithm classifies streaming sensor data in real-time. We concatenated together one subsequence taken from each of the two heartbeat waveforms to create one single stream of data. We then run our algorithm on this stream, and the resulting classification labels it produces are captured. Figure 14 shows the visualization of our algorithm in action.

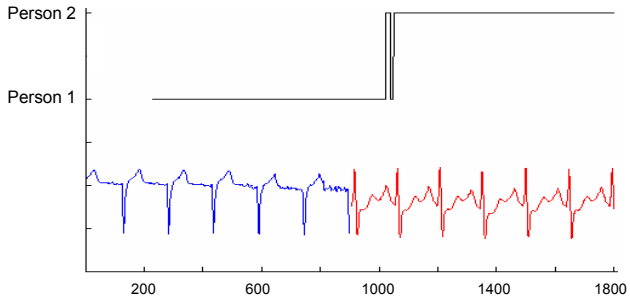


Figure 14: Visualization of our algorithm running in real-time. Person 1 data is indicated in blue and Person 2 data is indicated in red. The labels produced by our classifier are shown in black. The y-axis of the heartbeat data stream is meaningless since the data has been repositioned and resized for better visualization.

As described in section 4, we store 200 SAX words in the `historyBuffer` and convert 32 raw data points to a single SAX word. This means that the first classification does not occur till 200+32-1 = 231 points after the data begins streaming. During the first 230 points, the `historyBuffer` is still being populated, therefore, no classification occurs. There is a delay of 230 data points before the window of SAX words in the `historyBuffer` contains words entirely from the current class. Therefore, the lag period in the transition from one class to the next is 230 data points. Figure 14 illustrates this lag.

6. FUTURE WORK

There are several applications for our algorithm. We have shown that TSBs can be maintained in constant time and space, and can be highly accurate classifiers. In the near future, we intend to deploy this algorithm on a wireless sensor network to monitor buildings, running it in real-time on the hardware. We intend to explore how our algorithm performs in a distributed environment.

7. CONCLUSION

In this work, we have introduced a novel way to update Time Series Bitmaps in constant time. We have demonstrated that an amnesic algorithm like the one we propose can accurately detect complicated patterns in the erratic sensor data from an important entomological problem. Our algorithm is fast and accurate, and optimized for space. We have also described the wide range of applications for our algorithm by demonstrating its effectiveness in classifying robot and cardiology data.

8. ACKNOWLEDGEMENTS

Thanks to David C. Keith, Martin Gawecki, Anoop Mathur, Kyriakos Karenos, Dr. Vana Kalogeraki, Dragomir Yankov and Dr. Michail Vlachos. Further thanks to Dr. Manuela Veloso and Douglas Vail for donating the robotic data.

9. REFERENCES

[1] J. Almeida, J. Carrico, A. Maretzek, P. Noble, and M. Fletcher. Analysis of genomic sequences by Chaos Game Representation. In *Bioinformatics*, 17(5):429-37, 2001.

[2] V. Bhatkar, R. Pilkar, J. Skufca, C. Storey, and C. Robinson. Time-series-bitmap based approach to analyze human postural control and movement detection strategies during small anterior perturbations. *ASEE, St. Lawrence Section Conference*, 2006.

[3] L. Burattini, W. Zareba, and R. Burattini, Automatic detection of microvolt T-wave alternans in Holter recordings: effect of baseline wandering. *Biomedical Signal Processing and Control* 1(2), pp. 162-168, 2006.

[4] M. Datar, A. Gionis, P. Indyk, R. Motwani. Maintaining Stream Statistics over Sliding Windows. In *Proc. SIAM-ACM Symp. on Discrete Algorithms*, pp. 635-644, 2002.

[5] C. Daw, C. Finney, and E. Tracy. A review of symbolic analysis of experimental data. In *Review of Scientific Instruments*, volume 74, no. 2, pages 915-930, 2003.

[6] G. Janacek, A. Bagnall, and M. Powell. A likelihood ratio distance measure for the similarity between the Fourier transform of time series. *PAKDD*, 2005.

[7] S. Kaffka, B. Wintermantel, M. Burk, and G. Peterson. Protecting high-yielding sugarbeet varieties from loss to curly top, 2000. `http://sugarbeet.ucdavis.edu/Notes/Nov00a.htm`

[8] K. Kalpakis, D. Gada, and V. Puttagunta. Distance measures for effective clustering of ARIMA time-series. In *Proc. of the 1st IEEE International Conference on Data Mining*, 2001.

[9] E. Kasten, P. McKinley, and S. Gage. Automated Ensemble Extraction and Analysis of Acoustic Data Streams. *ICDCS Systems Workshops*, 2007.

[10] E. Keogh, J. Lin, and A. Fu. HOT SAX: Efficiently finding the most unusual time series subsequence. In *Proc. of the 5th IEEE International Conference on Data Mining*, pp. 226-233, 2005.

[11] E. Keogh, J. Lin, and W. Truppel. Clustering of Time Series Subsequences is Meaningless: Implications for Past and Future Research. In *Proc. of the 3rd IEEE International Conference on Data Mining*, pp. 115-122, 2003.

[12] E. Keogh, S. Lonardi, and C. Ratanamahatana. Towards Parameter-Free Data Mining. In *Proc. of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

[13] E. Keogh, L. Wei, X. Xi, S. Lonardi, J. Shieh, and S. Sirowy. Intelligent icons: Integrating lite-weight data mining and visualization into GUI operating systems. *ICDM,* 2006.

[14] N. Kumar, N. Lolla, E. Keogh, S. Lonardi, C.A. Ratanamahatana, and L. Wei. Time-series bitmaps: A practical visualization tool for working with large time series databases. In *Proc. of SIAM International Conference on Data Mining (SDM '05)*, 2005.

[15] M. Li, Y. Liu, and L. Chen. Non-threshold based event detection for 3D environment monitoring in sensor networks. *ICDCS*, 2007.

[16] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proc. of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2003.

[17] D. Minnen, T. Starner, I. Essa, and C. Isbell. Discovering characteristic actions from on-body sensor data. *10th International Symposium on Wearable Computer*, 2006.

[18] E. Pekalska, R. Duin, and P. Paclik. Prototype selection for dissimilarity-based classifiers. *Pattern Recognition*, Vol. 39, No.2, pages 189-208, 2006.

[19] C. Stafford and G. Walker. Characterization and correlation of DC electrical penetration graph waveforms with feeding behavior of beet leafhopper (Homoptera: Cicadellidae). *Under submission*, 2008.

[20] D. Vail and M. Veloso. Learning from accelerometer data on a legged robot. In *Proc. of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, 2004.

[21] L. Wei, N. Kumar, V. Lolla, E. Keogh, S. Lonardi, and C. Ratanamahatana. Assumption-Free Anomaly Detection in Time Series. In *Proc. of the 17th International Scientific and Statistical Database Management Conference*, pp. 237-240, 2005.

[22] M. Zoumboulakis and G. Roussos. Escalation: Complex event detection in wireless sensor networks. In *Proc. of 2nd European Conference on Smart Sensing and Context*, 2007.

Exploiting Spatial and Data Correlations for Approximate Data Collection in Wireless Sensor Networks

Chih-Chieh Hung Wen-Chih Peng York Shang-Hua Tsai
National Chiao Tung University
Hsinchu, Taiwan, ROC
{hungcc, wcpeng, tsaish}@cs.nctu.edu.tw

Wang-Chien Lee
The Pennsylvania State University
State College, PA 16801, USA
wlee@cse.psu.edu

ABSTRACT

Sensor nodes with similar readings can be grouped such that only readings from representative nodes need to be reported. However, efficiently identifying the sensor groups and their representative nodes is a very challenging task. In this paper, we propose an algorithm, namely DCglobal, to determine a set of representative nodes that have high energy levels and wide data coverage ranges, where a data coverage range of a sensor node is the set of sensor nodes whose reading vectors are very close to the sensor node. Furthermore, a maintenance mechanism is proposed to dynamically select alternative representative nodes when the representative nodes have less energy or representative nodes can no longer capture spatial correlation within their data coverage ranges. Through experimental studies on both synthesis and real datasets, we found that DCglobal is able to effectively and efficiently provide approximate data collection while prolonging the network lifetime.

Keywords:Approximate data collection, wireless sensor networks, spatial and data correlation.

1. INTRODUCTION

Recent advances in micro-sensing MEMS and wireless communication technologies have motivated the development of wireless sensor networks (WSN). Due to the form factor, sensor nodes are typically powered by small batteries. As a result, a major research issue in wireless sensor network is how to extend the *network lifetime* of WSN. To address this issue, various energy-saving techniques have been proposed in the literature [2][4][5][9][11][13]. Some argue that it is sufficient to support approximate data collection by tolerant certain error in readings from sensor nodes [4][9]. Moreover, sensor nodes nearby are usually expected to have similar readings due to their spatial correlation. Hence, a set of representative sensor nodes (denoted as r-nodes) can be used to approximate the readings of a monitored region. By rotating the role of representative nodes, the network lifetime of WSN can be extended.

Some research efforts have been elaborated on exploiting spatial and data correlation for approximate data collections [7][9]. Explicitly, to capture similar reading results from spatial correlation between sensor nodes, clustering techniques can be used. A distance function in the data space of sensor readings can be used to model the similarity between readings of two sensor nodes. The smaller a distance between two readings, the more similar they are. Meanwhile, sensor nodes located spatially close to each other can be identified by their communication connectivity. Given a specified distance threshold, nearby sensor nodes with similar readings are grouped. In [7], the authors proposed a snapshot query in which only the r-nodes need to report their readings. However, in [7], only one hop neighbor are involved in the similarity calculation. Moreover, in [9], the authors proposed to cluster sensor nodes nearby and formulated the problem as a CLIQUE-COVER problem, where a clique in the communication graph refers a group of sensor nodes having strong data and spatial correlation[1]. For example, Figure 1 shows a graph that models the connectivity (representing closeness) and readings of sensor nodes in the network. Assume that the Manhattan distance (i.e., the absolute difference value in their sensing readings) is used as the similarity function and an error threshold is 0.5. As shown in Figure 1(a), the number of r-nodes under snapshot queries is 6 (i.e., the black nodes are r-nodes). In addition, it can be seen in Figure 1(b), there are eight disjoint cliques covering the whole set of vertices in the graph. Consequently, r-nodes will be used to sample the readings in the network.

In this paper, we argue that selecting r-nodes by solving a SET-COVER problem can further reduce the number of selected r-nodes and thus extend the network lifetime. The SET-COVER problem is defined as follows:

***SET-COVER Problem**: Given a graph $G = (V, E)$ and a set $S = \{S_1, S_2, ..., S_n\}$, where $S_i \subseteq V$ for each i, find*

[1]Given a graph $G = (V, E)$, find the minimal number of disjoint cliques $\{C_1, C_2, ..., C_k\}$ such that $\bigcup_{i=1}^{k} C_i = V$.

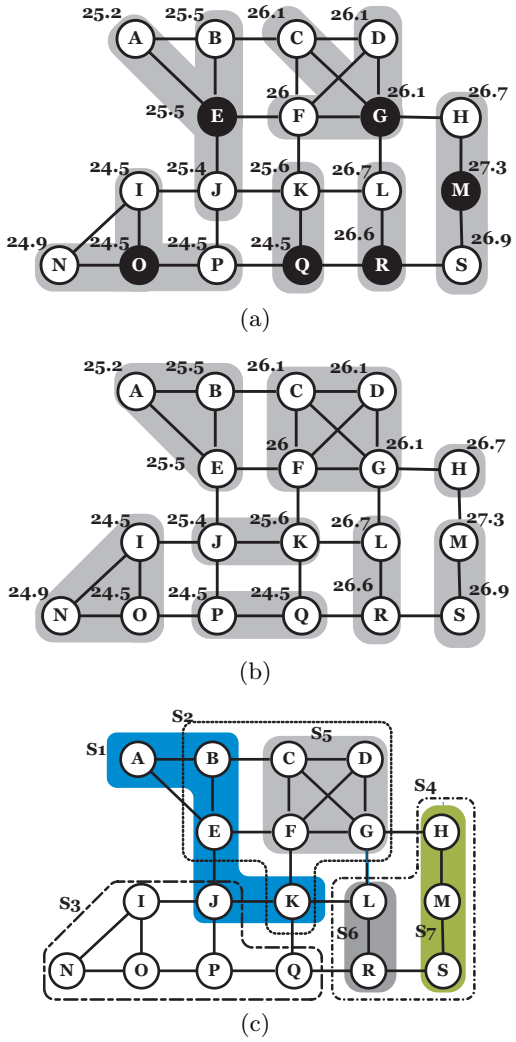


Figure 1: An illustrative example: (a)Snapshot, (b) CLIQUE-COVER and (c) SET-COVER

the subset of S, denoted as T, with minimal cardinality such that $\bigcup_{t \in T} t = V$.

In a SET-COVER problem, given a graph and a set of subsets of vertices, the goal is to find a minimal number of subsets to cover all vertices in the graph. By formulating our problem into a SET-COVER problem, we select one r-node for each subset of nodes in the sensor network. As shown in Figure 1(c), there are seven given sets S_1, S_2, ..., and S_7. We can select S_1, S_2, S_3, and S_4 (with A, F, N and S as r-nodes, respectively) to cover the whole vertices in the graph. As the wireless sensor network can be easily modeled as a connectivity graph, the main challenge is to obtain a proper input set S, which is the basic for solving SET-COVER problem, to select r-nodes.

To address this issue, we exploit a notion of *data coverage range* for each sensor node. The data coverage range of a sensor node s_i is the set of sensor nodes which are connected via a sequence of one-hop neighbors with readings similar to s_i's (governed by a threshold value ϵ). Each sensor node in data coverage range of a sensor node s_i is said to be *data-covered* by s_i. For the rest of this paper, we use the term "cover" in short to represent the term "data-cover". To conserve energy, as few r-nodes should be selected as possible such that the union set of their data coverage ranges covers the whole sensor nodes. Following the example in Figure 1 by using Manhattan distance as the similarity function with an error threshold being 0.5, it can be observed that the data coverage ranges of A, F, N, and S are {A,B,E,J,K}, {B,C,D,E,F,G,K}, {I,J,N,O,P,Q} and {H,L,M,R,S}, respectively. If nodes A, F, N and S are selected as r-nodes, all readings of other sensor nodes could be represented/approximated by these four sensor nodes. For example, since sensor node E is covered by sensor node A, the sensing reading of sensor node E (i.e., 25.5) can be approximated by reading of sensor node A (i.e., 25.2). Furthermore, the union set of their data coverage ranges covers the whole sensor nodes. As such, the network lifetime can be extended since only four sensor nodes are reported their readings and the error between the readings obtained and the ground truth is tolerable.

Nevertheless, to realize the proposed idea, several design issues need to be addressed. First, the data coverage range of each sensor node needs to be derived. Second, in order to extend network lifetime, the selected r-nodes should also have abundant energy and large data coverage ranges. Third, an effective maintenance mechanism needs to be in place to adapt to the energy budgets of nodes and the dynamics of sensor readings. It is a non-trivial task to select a substitutable r-node because sensor nodes in the data coverage range of the original r-nodes do not necessarily cover each other. For example, as shown in Figure 1(c), both node M and node R are within the data coverage range of node S, but they do not cover each other. Similar to many prior works [8][9], we assume that sensor nodes send all sensor readings to the sink for analysis. Given a pre-specified error threshold and a set of sensor nodes with their readings, we propose *DCglobal* (Standing for Data Coverage with global information) to determine a set of r-nodes to maximize network lifetime in WSN. In algorithm DCglobal, by taking both sensing readings and energy levels of all sensor nodes collected from WSN into account, the sink will select a set of r-nodes which cover all sensor nodes and maximize the network lifetime. Specifically, algorithm DCglobal is able to derive a set of representative nodes with high availability of energy and wide data coverage ranges. Furthermore, the maintenance mechanism we proposed for DCglobal dynamically select substitutes for r-nodes when r-nodes run low on energy or r-nodes no longer capture the correlations within their data coverage ranges. Experiments based on both synthesis and real datasets show that the proposed DCglobal significantly outperforms previous works in terms of extending the network lifetime of WSN.

The rest of this paper is organized as follows. Related works are discussed in Section 2. Preliminaries are given in Section 3. In Section 4, we develop DCglobal to select r-nodes for WSN. Performance studies are conducted in Section 5. This paper concludes with Section 6.

2. RELATED WORKS

There is a number of researches on data collection in WSN. These works can be roughly classified into three categories. In the first category, in-network data aggregation is

performed for data collection [11][13], which uses a routing tree to partially aggregate measurements (e.g., MAX, MIN) on their way to their destination such that the amount of transmitted data is reduced. However, these works preserve energy only on some specified operations. In the second category, approximate data collection is performed by building probabilistic models [3][4]. The authors in [3] explored a model-driven architecture in which a centralized probabilistic model is used to estimate the readings of sensor nodes by generating an observation plan to collect appropriate readings of sensor nodes. Also, the authors of [4] exploited spatial correlation for approximate data collection, where a replicated dynamic probabilistic model is built for each clique to minimize the communication from sensor nodes to the sink. However, in these works, users must define a probabilistic model to fit collected readings. It is hard to define when sensor nodes are deployed in an unfamiliar environment. In the last category, approximation data collection is performed without building probabilistic models [7][9]. In [7], by utilizing spatial correlation, the author derived an extension of declarative query in sensor networks, called snapshot queries. The snapshot queries can be answered through a data-driven approach by using a linear regression model to predict the readings of 1-hop neighbors. The authors in [9] proposed an algorithm, named as EEDC, that is executed in a centralized server. Based on spatial correlation, EEDC partitions sensor nodes into disjoint cliques such that sensor nodes in the same clique have similar surveillance time series. Round-robin scheduling has been employed to share workload of data collection. In our paper, we exploit data and spatial correlation for approximate data collection. To the best of our knowledge, this is the first work to formulate a SET-COVER problem to achieve energy saving in support of approximate data collection.

3. PRELIMINARIES

To model spatial correlation among sensor nodes, we first define the reading behaviors of sensor nodes and then the similarity between two reading behaviors of sensor nodes. Thus, we have the following definitions:

Definition 1. Reading Vector: *Assume that the readings of a sensor node s_i consists of a series of readings in a sliding window ℓ. The readings vector of s_i is $\overrightarrow{v_i}(t) = < x_i(t-\ell+1), x_i(t-\ell+2), ..., x_i(t) >$, where $x_i(t)$ is the reading sensed by s_i at the time t.*

Clearly, the readings of a sensor node within a sliding window is represented as a *reading vector*. Therefore, we can define the similarity of two sensor nodes in terms of distance of their reading vectors. There are a lot of existing distance functions, such as Euclidean distance, cosine distance and so on, which are usually application specific and task dependent [6]. To simplify our discussions, we employ the *Manhattan distance* (i.e., $d(s_i, s_j) = |\overrightarrow{v_i}(t) - \overrightarrow{v_j}(t)|$) in the following examples.

Given a distance function between two sensor nodes, we can formally define data coverage range as follows:

Definition 2. Data Coverage Range: *Given an error threshold ϵ, the data coverage range of sensor node s_i, denoted as C_i, is the set of sensor nodes such that sensor node s_j in C_i if and only if there exists a sequence of sensor nodes $< s_i = s_0, s_1, ..., s_k = s_j >$ for $k \geq 0$, s_t directly communicates with s_{t+1} and $d(s_i, s_t) \leq \epsilon$ for $0 \leq t \leq k-1$.*

For example, in Figure 1(b), suppose ϵ to be 0.5. E is in the data coverage range of sensor node A because there exists a sequence of sensor node $< A, B, E >$ such that A can communicate with B and B can communicate with E. In addition, $d(A, B) = 0.3 \leq 0.5$, and $d(A, E) = 0.3 \leq 0.5$. Consequently, the data coverage range of sensor node A is $\{A, B, E, J, K\}$.

To analyze the guideline for r-nodes selection to extend network lifetime, it is necessary to derive a network lifetime model. Here, the same concept in [1] is used for modeling. Let $G = (V, E)$ be the communication graph of a WSN, S be the set of sensor nodes, E_i be the energy level of sensor node s_i, and N_i be the set of neighbor sensor nodes of sensor node s_i. Assume that the transmission energy required for sensor node s_i to transmit an information unit to its neighboring sensor node s_j is p_{ij}. To indicate whether sensor node s_i and s_j involve into data collection or not, $I_{ij}(k)$ is defined as the indication random variable, where $I_{ij}(k) = 1$ if s_i transmits an information to s_j when involving into data collection of the kth time. Otherwise, $I_{ij}(k) = 0$.

Following the notation above, given the frequency for data collection f, the lifetime of a sensor node s_i , $T(i)$, is given by $\frac{E_i}{\sum_{k=1}^{f} \sum_{j \in N_i} p_{ij} \times I_{ij}(k)}$. In addition to the energy level of sensor node s_i, the main factor affecting the lifetime of a sensor node is how many transmission during data collection. When involving a lot of transmission for data collection (i.e., there are a lot of k such that $I_{ij}(k) = 1$), a sensor node will soon exhaust its energy and thus its lifetime will be short. The network lifetime is defined as the length of time until the first sensor node runs out its battery. Based on the definitions above, the problem of selecting r-nodes can be formulated as follows:

Problem: Let the error threshold for data collection be ϵ and data collection frequency be f. Given a set of sensor node $S = \{s_1, s_2, ..., s_n\}$ with the associated data coverage range $\{C_1, C_2, ..., C_n\}$ and energy level of each sensor node: $\{E_1, E_2..., E_n\}$[2], find the set of r-nodes $R \subseteq S$ to maximize the network lifetime of WSN under the constraint $\bigcup_{s_i \in R} C_i = S$.

Intuitively, the size of R will significantly affect the lifetime because only r-nodes have to report their readings. Moreover, to extend the network lifetime, the sensor nodes with higher energy levels should have higher priority to become r-nodes. Thus, a strategy for selecting r-nodes is to give high priority to the sensor nodes with high energy levels.

4. DCGLOBAL: ALGORITHM FOR DETERMINING R-NODES WITH GLOBAL INFORMATION

In Section 4.1, we propose algorithm *DCglobal* to select r-nodes for extending network lifetime. The sink first requires sensor nodes to report their readings and energy levels. Once collecting global information at the sink, DCglobal is able to determine the set of r-nodes. Then, the sink broadcasts the

[2]In this paper, we discretize remaining energy into levels.

information of r-nodes to each sensor node in the network. Correspondingly, our maintenance mechanism is developed in Section 4.2.

4.1 Design of DCglobal

After collecting readings and energy levels from sensor nodes, the sink has energy levels of sensor nodes as $E = \{E_1, E_2, ..., E_n\}$. In addition, since the sink owns the topology of the network and readings of sensor nodes, it is straightforward to compute data coverage range of sensor nodes as $C = \{C_1, C_2, ..., C_n\}$ by the definition above. Given C and E as inputs, algorithm DCglobal can then determine r-nodes in the network. In a nutshell, DCglobal sorts sensor nodes in descending order of energy level and data coverage range of each sensor node. The higher order a sensor node is, more energy or larger data coverage range a sensor node has. Then, DCglobal can select r-nodes iteratively by the orders of sensor nodes until the union set of data coverage ranges of r-nodes covers all sensor nodes.

To rank sensor nodes, we first define a partial order relation $\preceq_{EnC}$ (Energy and Coverage) between two sensor nodes as follows:

Definition 3. Partial Order Relation $\preceq_{EnC}$***:*** *Let* $S = \{s_1, \ldots, s_n\}$ *be the set of sensor nodes, the data coverage range of* s_i *is* C_i *and the energy level of* s_i *is* E_i*. A partial order relation* $\preceq_{EnC}$ *is a binary relation over* $S \times S$*. For all* $s_i, s_j \in S$*,* $s_i \preceq_{EnC} s_j$ *if and only if* $E_i < E_j$ *or* $(E_i = E_j$ *and* $C_i \subseteq C_j)$*.*

The philosophy of the partial order relation $\preceq_{EnC}$ is to assign priorities to sensor nodes. Since our goal is to extend network lifetime, which refers the period of time till the first sensor node dies, sensor nodes are ranked based on their energy first and use their data coverage ranges as a tie-breaker. More specifically, in partial order relation $\preceq_{EnC}$, two sensor nodes are related if one of them is inferior to the other sensor node according to the energy level and the data coverage range. In $\preceq_{EnC}$, the maximal sensor node is referred to the sensor node that no other sensor node could be inferior to.

Algorithm 1 DCglobal

Input: S, the set of sensor nodes; C, Data coverage ranges; E, Energy levels;
Output: R, the set of r-nodes

```
1: P ← partial ordered set built by ⪯EnC using C and E;
2: S' ← S
3: R ← φ
4: while S' ≠ φ do
5:    begin
6:    s ← the maximal sensor node in P;
7:    for each sensor node j ∈ Ci do
8:       begin
9:       Remove j from S';
10:      Remove all elements related to j in P ;
11:      Remove j from M for other sensor nodes;
12:      end
13:   Add s into R;
14:   Remove all elements related to s in P ;
15:   end
16: Return R
```

There are two properties about $\preceq_{EnC}$ to be clarified. First, one sensor node s_i may not be inferior to another sensor node s_j when their energy levels are the same and the data coverage range of one sensor node does not cover the other's data coverage range. In other words, s_i and s_j cannot be totally replaced by each other when selecting r-nodes. Thus, sensor nodes which are not inferior to each other have the same priority to become r-nodes. Second, the partial order relation $\preceq_{EnC}$ ranks sensor nodes according to their energy levels first. It is possible that the maximal sensor nodes may own smaller data coverage ranges than other sensor nodes and thus the number of r-nodes may increase. However, since r-nodes in each round may most likely consume more energy than sensor nodes which are not r-nodes, the sensor nodes with boarder data coverage ranges and more energy will become maximal sensor nodes in the following rounds. Thus, based on $\preceq_{EnC}$, the number of r-nodes is minimized.

An illustrative example of DCglobal is shown in Figure 2(b), where the number above a sensor node represents the energy level of this sensor node, black sensor nodes are r-nodes, and white sensor nodes are covered by r-nodes. To illustrate this example, assume that the length of reading vector is 1, the values of readings are the same as Figure 1(b) and the error threshold ϵ is set to 0.5. The running procedure is shown in Figure 2(a), where R is the set of selected r-nodes and sequences in P are sensor nodes listed in a decreasing order in terms of $\preceq_{EnC}$. Sensors with higher order have higher priority to become r-nodes. Those sensor nodes in the same bucket are not inferior to each other and thus own the same priority to become r-nodes. In the first round, as shown in Figure 2(b), sensor node N is selected as a r-node since N is the maximal sensor node in P. Then, P list is updated in that N and sensor nodes covered by N are removed from the list. In the second round, sensor node A is selected as a r-node because A is the maximal sensor node in P. In a similar way, sensor node F and S are selected to be r-nodes. When r-nodes selected are able to cover the whole set of sensor node, DCglobal terminates. As a result, the set of r-node R is $\{N, A, F, S\}$ in this example.

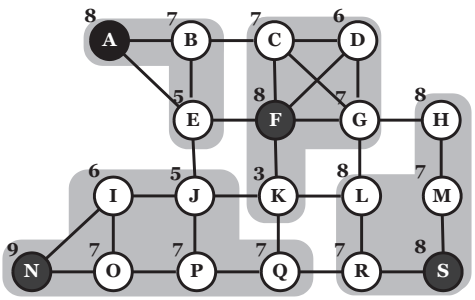
4.2 Maintenance Mechanism

Here we describe our design of the maintenance mechanism for DCglobal. The maintenance mechanism substitutes those r-nodes that run low on energy and those that no more represent nodes in their data coverage ranges. Since r-nodes may likely consume more energy than sensor nodes which are not r-nodes. Thus, to extend network lifetime, r-nodes should be substituted by those sensor nodes with more energy. On the other hand, a r-node remains valid only if it can cover all sensor nodes in its data coverage. Each r-node should be valid in each time slot. With time passing by, however, some r-nodes may no longer cover sensor nodes in their own data coverage ranges. In this case, r-nodes should be re-selected to represent the readings of the whole network precisely.

The operation of our maintenance mechanism hybrids reactive and proactive approaches to address the two issues. First, the sink periodically collects readings and energy levels of sensor nodes to update the data coverage ranges C and energy levels E of sensor nodes for every ℓ time slots. According to the information of sensor nodes, updated data coverage ranges are compared with the previous data coverage ranges to make sure which r-nodes are valid. On the other hand, during the periodical maintenance periods, the

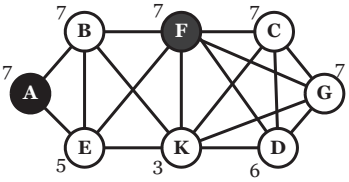
	P list	R
1	N, (A,F,S), (L,H), (B,C,G,M,O,P,Q,R), (D,I), E, J, K	N
2	(A,F,S), (L,H), (B,C,G,M,R), D, E, K	N,A
3	(F,S), (L,H), (C,G,M,R), D, K	N,A,F
4	S, (L,H), (M,R)	N,A,F,S

(a)

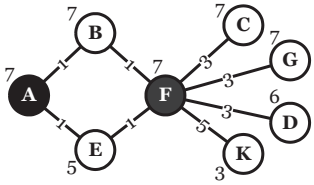


(b)

Figure 2: An illustrative example: (a) running procedure and (b) results



(a)



(b)

Figure 3: An illustrative example: (a) DC graph (b) SRN graph

sink does not collect information from the network. Thus, every sensor node in the network should keep monitoring the variation of the environment. Specifically, every sensor node s_i maintains its current reading vector $\overrightarrow{v_i}(t)$ at the current time t and the reading vector $\overrightarrow{v_i}(t_p)$ at the time t_p when announced by its r-node. Once discovering the distance between $\overrightarrow{v_i}(t)$ and $\overrightarrow{v_i}(t_p)$ larger than ϵ, a sensor node s_i will notify its r-node. The r-node asks each sensor node in its data coverage range to send their reading vectors at time t_p to the sink. Thus, the sink can verify whether the r-node is valid or not. Depending on the validity of an r-node, the different maintenance operations will be executed as follows.

Finding substituting r-nodes

For a valid r-node, it still represents the sensor nodes in its data coverage range. However, if the energy level of a valid r-node is one level lower than the previous energy level (when it became a r-node), this r-node should be substituted. Note that if the data coverage ranges of r-nodes to be replaced are adjacent to each other, we can select substitutes for these r-nodes at the same time in order to reduce the number of substituting r-nodes. To simplify our discussion, a valid r-node needed to be replaced is abbreviated as a valid r-node for the rest of this paper.

Here, we discuss a notion of *Shared Representative Neighbor* (abbreviated as SRN). The SRN represents the number of common sensor nodes covered by a sensor node and a r-node. For example, in Figure 1(b), E can cover {A,B,E,F,J,K} and F can cover {B,C,D,E,F,G,K}. The SRN between E and F is 1 since B is the common sensor node covered by E and F (note that the SRN between E and F does not count E and F themselves). The SRN measures the degree of data coverage range overlapping between a sensor node and a r-node. If a sensor node has a large SRN value to a r-node, their data coverage ranges are highly overlapping. Intuitively, the sensor node with high SRN to a r-node is suitable for replacing the r-node and thus assigned to substitute a r-node.

To exploit SRN in selection of substituting r-nodes, a *DC graph* is constructed to reflect the data coverage of sensor nodes in adjacent data coverage ranges of valid r-nodes. A DC graph is defined as follows:

Definition 4. DC Graph: *Let $\{s_1, s_2, ..., s_m\}$ be the valid r-nodes with their data coverage ranges adjacent to each other. The DC graph for $\{s_1, s_2, ..., s_m\}$ is a graph $G_{DC} = (V_{DC}, E_{DC})$, where $V_{DC} = \bigcup_{i=1}^{m} C_i$ and $(s_i, s_j) \in E_{DC}$ if s_i covers s_j.*

Following the example in Figure 2(b), suppose that valid r-nodes needed to be replaced are A and F, where their energy levels are reduced to level 7. The corresponding DC graph is shown as Figure 3(a). Once obtaining a DC graph, we can construct the SRN graph for valid r-nodes with adjacent data coverage ranges:

Definition 5. SRN Graph: *Let $\{s_1, s_2, ..., s_m\}$ be the valid r-nodes with their data coverage ranges adjacent to each other. The SRN graph for $\{s_1, s_2, ..., s_m\}$ is defined to be a weighted graph $G_{SRN} = (V_{SRN}, E_{SRN})$, where $V_{SRN} = \bigcup_{i=1}^{m} C_i$ and $(s_i, s) \in E_{SRN}$ with weight w if s has w common neighbors with a r-node s_i in the DC Graph G_{DC}.*

For example, as shown the DC graph in Figure 3(a), sensor node K has five common neighbors, (i.e., B, C, D, E, and G), to the valid r-node F. Thus, there is an edge (F, K) with weight 5 in the SRN graph. In a similar way, the corresponding SRN graph can be constructed as shown in Figure 3(b).

Based on the SRN graph and the energy levels of sensor nodes, the substituting r-node(s) can be selected as follows: among the sensor nodes with the highest energy levels, the sensor node with the largest total weight of each edge connected to it in the SRN graph is selected to be the substituting r-node. Then, sensor nodes covered by it are removed from the DC graph. This step is repeated until the DC graph is empty.

For example, in Figure 3(b), among the sensors with the highest energy level 7, G owns the largest SRN value 3,

Round	V_{DC}	r-nodes
1	A,B,C,D,E,F,G,K	Φ
2	A,B,E	G
3	Φ	G,B

Table 1: An execution scenario of selecting substituting r-nodes

which is larger than SRN value of B (i.e., 1+1=2). Therefore, G is selected to be the substituting r-node and C, D, F, G, and K are removed from the DC graph. In the next round, B is selected to be the substituting r-node and removed from the DC graph. The procedure terminates since the DC graph is empty. The whole procedure is shown in Table 1.

Determining new r-nodes

Since the environment may change, some of r-nodes become invalid because sensor nodes in their data coverage range should be covered by new r-nodes. In this case, the sink first checks whether any sensor node can be covered by existing r-nodes or not. If so, sensor nodes are assigned to the data coverage range of some existing r-nodes. For those sensor nodes that cannot be covered by existing r-nodes, the sink executes localized DCglobal for them.

5. PERFORMANCE EVALUATION

In this section, our experiments are conducted by both synthesis dataset and real dataset. In both datasets, we compare our proposed DCglobal with EEDC [9], snapshot [7] and Naive (no clustering).

5.1 Datasets

For synthesis datset, we randomly deploy $1,000$ sensor nodes in a $[0 \dots 1000) \times [0 \dots 1000)$ area (the unit - meters) and the sink is at $(0, 0)$. Following the transmission range of mica2 [14], each sensor node is set to be 100 meters. Twenty events are placed in the field in uniform distribution; the initial value of events are normal distribution $N(25°\text{C}, 1°\text{C})$. To make the reading more volatile, the value changes $[-1 \dots 1]°\text{C}$ follow a normal distribution every 20 time units. The reading of a sensor node at time t is the weighted average of all events at time t. The weight of an event is the inverse of the square of the distance between the sensor node and the event. We submit $1,000$ range queries with range $100m \times 100m$ at random in time interval $[0 \dots 2000]$. The lifetime of each sensor node is set to be able to transmit $1,000$ messages. The error threshold is 3 and the window size is 50 time units. All experiments are preformed $1,000$ times and the result is the average performance. In real dataset, we use the publicly available Intel Lab dataset [10] that consists 54 sensor nodes which measure various attributes. We use light attribute from this dataset. In this dataset, due to missing readings and asymmetric communication between two sensor nodes, we fill missing readings by previous readings and consider two sensor nodes to be connected if the probability of packet loss is less than 50%. All experimental results are the average performance from readings of ten days from this dataset.

5.2 Network Lifetime

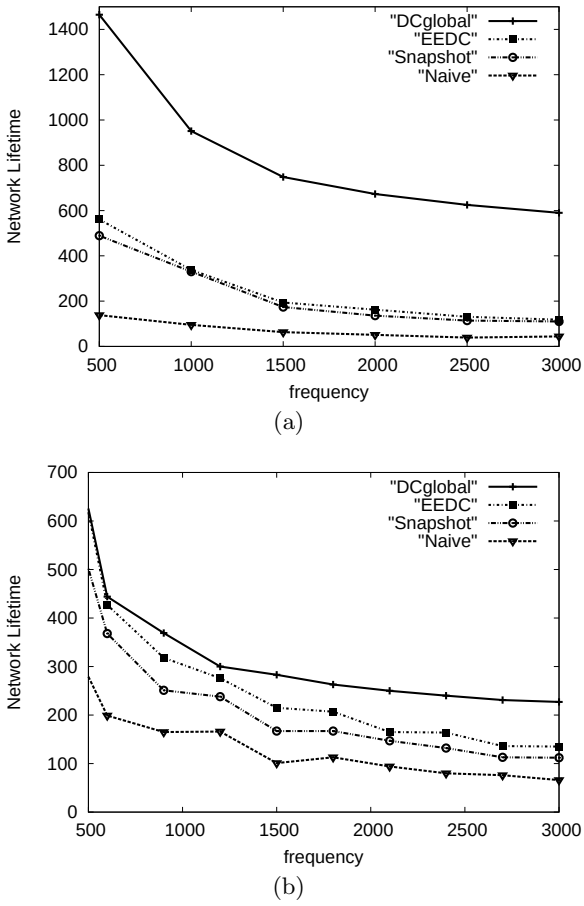


Figure 4: Network lifetime over the number of queries varied under (a)synthesis and (b)real datasets

We first investigate the network lifetime of DCglobal, Snapshot, EEDC, and the Naive scheme (no clustering). Figure 4(a) and Figure 4(b) depict the network lifetime by varying the number of queries. Snapshot builds a linear regression prediction model to estimate the reading of sensor node's a one-hop neighbors and select one-hop cluster head. EEDC is a centralized algorithm that find all cliques in the network base on both connectivity and data coverage in each clique. In each clique, cluster head is selected in round-robin. Note that the network lifetime is the time when the first sensor node depletes its energy. For synthesis dataset, Figure 4(a) shows that the network lifetime of DCglobal is much longer than other algorithms. From real dataset, as the number of queries increases, DCglobal has longer network lifetime than other approaches. By considering both data coverage size and remaining energy of each sensor node, DCglobal extends the network lifetime by 66% in both datasets, showing the effectiveness of DCglobal.

5.3 Impact of Error Thresholds

DCglobal selects r-nodes to report their reading for approximate data collection. Thus, the error threshold will has an impact on the network lifetime as well. For syn-

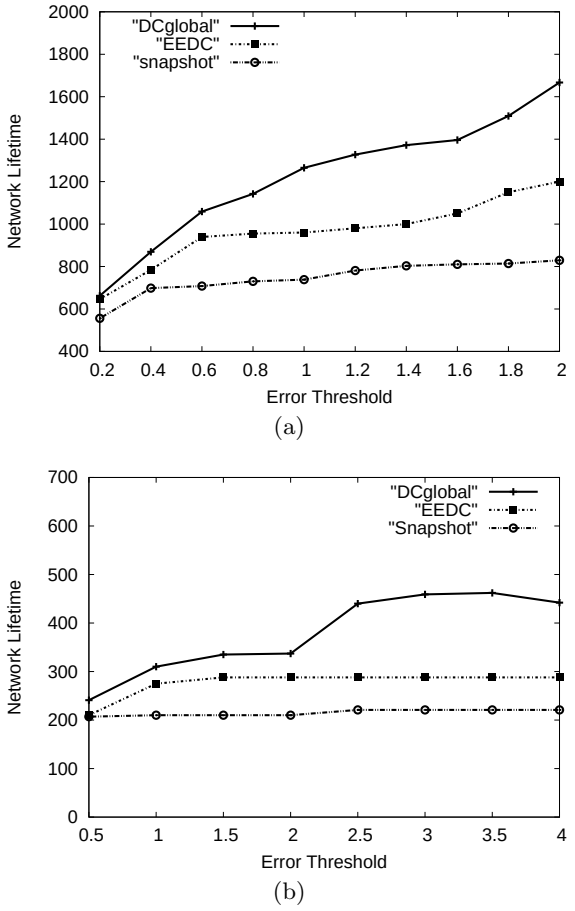


Figure 5: Network lifetime over variant error threshold under (a)synthesis and (b)real datasets

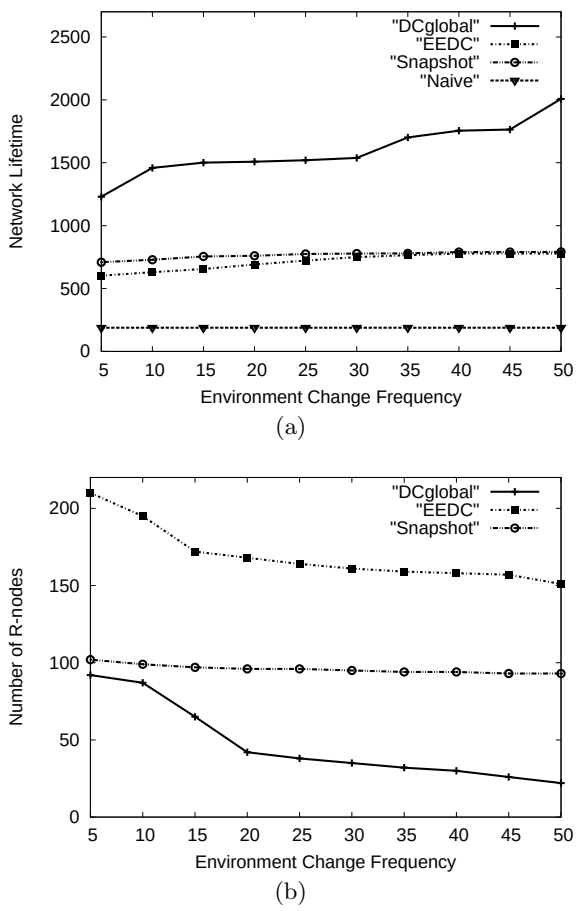


Figure 6: (a) Network lifetime under the environmental change (b) Number of r-nodes over variant environment change frequency

thesis dataset, Figure 5(a) shows that the network lifetime increases when the error threshold increases. This is because that when the error threshold is loose, the number of r-nodes will decrease and thus the network lifetime is extended. On the other hand, for real data, Figure 5(b) indicates that the lifetime for DCglobal increases when error threshold increases. In this case, the lifetime of EEDC and Snapshot do not increase significantly. Note that when the error threshold is between 0.5 to 2, the trend of the lifetime is not significant for EEDC and DCglobal. However, when the error threshold exceeds to 2.5, the trend of all algorithms is stable. Thus, in this environment, a reasonable guess to error threshold is around 2.5.

5.4 Environmental Variation

Finally, we evaluate the performance of these algorithms under environmental changes. To control the environmental variation, we use synthesis data only to show the results. In Figure 6(a) and 6(b), the impact of varied environments is shown for each algorithm by differing the event reading changing interval, from unstable to stable. DCglobal performs well, resulting longer network lifetime whether the environment is unstable or stable. We can see that the lifetime of other algorithms do not change too much. However, DCglobal can extend the network lifetime when the environment becomes stable. Also, it can be seen that the number of r-nodes selected by DCglobal is less than EEDC and Snapshot. It also supports our claim that DCglobal, an algorithm based on solving a SET-COVER problem, is effective to reduce the number of r-nodes. Thus, we can conclude that DCglobal can adapt to dynamically environment changing.

6. CONCLUSIONS

In this paper, we addressed the problem of selecting a set of r-nodes for approximate data collection in WSN. Specifically, we argued that the number of r-nodes can be reduced by solving a SET-COVER problem. Moreover, by exploiting spatial and data correlation, data coverage range of each sensor is determined. By taking energy level and data coverage range of each sensor node into account, DCglobal can further reduce the number of r-nodes, thereby extending the network lifetime. In addition, the maintenance mechanism for DCglobal was proposed to efficiently select substituting r-nodes for r-nodes with less energy and to reflect the change of environments. Experimental results on both synthesis and real datasets show that DCglobal is able to significantly

extend the network lifetime and outperform prior works.

Acknowledgement

Wen-Chih Peng was supported in part by the National Science Council, Project No. NSC 95-2221-E-009-061-MY3 and by Taiwan MoE ATU Program. Wang-Chien Lee was supported in part by the National Science Foundation under Grant no. IIS-0328881, IIS-0534343 and CNS-0626709.

7. REFERENCES

[1] J.-H. Chang and L. Tassiulas. Energy conserving routing in wireless ad-hoc networks. In *Proc. of IEEE INFOCOM*, pages 22–31, 2000.

[2] C. fan Hsin and M. Liu. Network coverage using low duty-cycled sensors: random & coordinated sleep algorithms. In *Proc. of IPSN*, pages 433–442, 2004.

[3] A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *Proc. of VLDB*, pages 588–599, 2004.

[4] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong. Approximate data collection in sensor networks using probabilistic models. In *Proc. of ICDE*, pages 48–59, 2006.

[5] C. Guestrin, P. Bodík, R. Thibaux, M. A. Paskin, and S. Madden. Distributed regression: an efficient framework for modeling sensor network data. In *Proc. of IPSN*, pages 1–10, 2004.

[6] H. V. Jagadish, A. O. Mendelzon, and T. Milo. Similarity-based queries. In *Proc. of ACM PODS*, pages 36–45, 1995.

[7] Y. Kotidis. Snapshot queries: Towards data-centric sensor networks. In *Proc. of ICDE*, pages 131–142, 2005.

[8] L. Krishnamurthy et al. Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea. In *Proc. of ACM SenSys*, pages 64–75, 2005.

[9] C. Liu, K. Wu, and J. Pei. A dynamic clustering and scheduling approach to energy saving in data collection from wireless sensor networks. In *Proc. of IEEE SECON*, pages 374–385, 2005.

[10] S. Madden. *Intel Lab Data 2004*. http://berkeley.intel-research.net/labdata.

[11] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: A tiny aggregation service for ad-hoc sensor networks. In *Proc. of OSDI*, 2002.

[12] A. Meka and A. K. Singh. Distributed spatial clustering in sensor networks. In *Proc. of EDBT*, pages 980–1000, 2006.

[13] Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD Record*, 31(3):9–18, 2002.

[14] Crossbow Technology Co. *Mica2 Datasheet*. http://www.xbow.com.