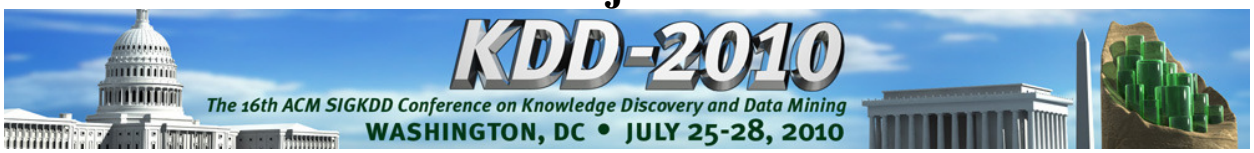
Proceedings of the Fourth International Workshop on Knowledge Discovery from Sensor Data (SensorKDD'10)

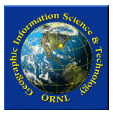
Held in conjunction with



July 25, 2010
Washington, DC

Workshop Chairs

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



Fourth International Workshop on Knowledge Discovery from Sensor Data (SensorKDD'10)

Description

Wide-area sensor infrastructures, remote sensors, RFIDs, phasor measurements, and wireless sensor networks 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 adaptability to national or homeland security, critical infrastructures monitoring, disaster preparedness and management, greenhouse emissions and climate change, and transportation. 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 are dynamic data streams or events that require real-time analysis methodologies and systems, while on the other hand are static data that require high end computing for generating offline predictive insights, which in turn can facilitate real-time analysis. The online and real-time knowledge discovery imply immediate opportunities as well as intriguing short- and long-term challenges for practitioners and researchers in knowledge discovery. The opportunities would be to develop new data mining approaches and adapt traditional and emerging knowledge discovery methodologies to the requirements of the emerging problems. In addition, emerging societal problems require knowledge discovery solutions that are designed to investigate anomalies, rare events, hotspots, changes, extremes and nonlinear processes, and departures from the normal.

According to the data mining and domain experts present at the NSF-sponsored Next Generation Data Mining Summit (NGDM '09) held in October 2009, "finding the next generation of solutions to these challenges is critical to sustain our world and civilization" [1]. Some of the organizers of the SensorKDD workshop are part of the summit. The 4th International Workshop on Knowledge Discovery from Sensor Data (SensorKDD-2010) is the first step in bringing researchers together to address these challenges and moving toward the development of the next generation data mining solutions require to address these challenges.

Therefore, the SensorKDD-2010 seeks to bring together researchers from academia, government, and the industry working in the following areas and applications:

1. Offline Knowledge Discovery
    a. Predictive analysis from geographically distributed and heterogeneous data
    b. Computationally efficient approaches for mining unusual patterns, specifically, anomalies, extremes, nonlinear processes and change, from massive and disparate space-time data
2. Online Knowledge Discovery
    a. Real-time analysis of dynamic and distributed data, including streaming and event-based data
    b. Mining from continuous streams of time-changing data and mining from ubiquitous data
    c. Efficient algorithms to detect deviations from the normal in real-time
    d. Resource-aware algorithms for distributed mining
3. Decision and Policy Aids

    a. Coordinated offline discovery and online analysis with feedback loops
    b. Combination of knowledge discovery and decision scientific processes
    c. Facilitation of faster and reliable tactical decisions as well as prudent and insightful longer term policies
4. Theory
    a. Distributed data stream models
    b. Theoretical frameworks for distributed stream mining
5. Case Studies
    a. Success stories in national or global priority applications
    b. Real-world problem design and knowledge discovery requirements

Motivation

The motivation for SensorKDD (http://www.ornl.gov/sci/knowledgediscovery/SensorKDD-Workshop/) in conjunction with the ACM SIGKDD Conference on Knowledge Discovery and Data Mining stems from the increasing need for a forum to exchange ideas and recent research results, and to facilitate collaboration and dialog between academia, government, and industrial stakeholders. The expected ubiquity of sensors in the future, combined with the critical roles they are expected to play in high priority application solutions, point to an era of unprecedented growth and opportunities. The requirements described earlier imply immediate opportunities as well as intriguing short- and long-term challenges for practitioners and researchers in knowledge discovery. In addition, the knowledge discovery and data mining (KDD) community would be called upon, again and again, as partners with domain experts to solve critical application solutions in business and government, as well as in the domain sciences and engineering.

The first workshop was organized in 2007. Based on the positive feedback from the previous workshop attendees and our own experiences and interactions with the government agencies such as the United States Department of Homeland Security, United States Department of Defense, and involvement with numerous projects on knowledge discovery from sensor data, we strongly believe in the continuation of this workshop. We believe that the ACM SIGKDD conference is the right forum to organize this workshop as it brings the KDD community together in this important area to establish a much needed leadership position in research and practice in the near term, as well as in the long term.

Success of Previous SensorKDD Workshops

The previous three workshops – SensorKDD-2007 [2], SensorKDD-2008 [3], and SensorKDD-2009 [4] – held in conjunction with the 13th, 14th, and 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining respectively attracted several participants as well as many high quality papers and presentations. The 2007 workshop was attended by more than seventy registered participants. The workshop program included presentations by authors of six accepted full papers and four invited speakers. The invited speakers were Prof. Pedro Domingos of the University of Washington, Prof. Joydeep Ghosh of the University of Texas, Austin, Prof. Hillol Kargupta of the University of Maryland, Baltimore County, and Dr. Brian Worley of the Oak Ridge National Laboratory (ORNL). There were also poster presentations by authors of six accepted short papers. The extended versions of papers presented at the workshop were developed into a book [5], the first book published in this specific discipline. The four top accepted papers were awarded certificates and cash prize of $500 each. The prize money was donated by the Computational Sciences and Engineering Division (CSED) of the Oak Ridge National Laboratory and Information Society Technology Project KDUbiq-WG3 of the European Union. The top papers were also published in a special issue of the *Journal of Intelligent Data Analysis*. This workshop was partially sponsored by the Geographic Information Science and Technology Group of CSED at ORNL.

The SensorKDD-2008 workshop was attended by more than 60 registered participants. There were presentations by authors of seven accepted full papers and six accepted short papers; the workshop

program also included presentations by two invited speakers – Prof. Jiawei Han of the University of Illinois at Urbana-Champaign and Dr. Kendra Moore of the Defense Advanced Research Projects Agency. The extended versions of papers presented at the 2008 workshop were recently published as Springer's LNCS post-proceedings in 2009 [6]. The two top accepted papers were awarded certificates and cash prize of $500 each; the prizes were donated by the Computational Sciences and Engineering Division of the Oak Ridge National Laboratory. This workshop was partially sponsored by the Geographic Information Science and Technology Group of CSED at ORNL.

The 2009 workshop was attended by several registered participants. There were presentations by authors of eight accepted full papers, eight accepted short papers, two entries for the SensorKDD-2009 cup, and three invited speakers. The invited speakers were Prof. Carlos Guestrin of Carnegie Mellon University, Dr. Aurelie Lozano of IBM T.J. Watson Research Center, and Mr. Alessandro Donati of the European Space Agency. The extended versions of papers presented at the 2009 workshop are scheduled for publication as Springer's LNCS post-proceedings in 2010. The best paper, two best student papers, and two SensorKDD-2009 cup entries were awarded certificates and cash prizes. The prizes were donated by the Computational Sciences and Engineering Division of the Oak Ridge National Laboratory and Cooperating Objects Network of Excellence of the European Union. The workshop was partially sponsored by the Geographic Information Science and Technology Group of CSED at ORNL.

Workshop Sponsors

The SensorKDD-2010 workshop is sponsored by the Geographic Information Science and Technology (GIST) Group at Oak Ridge National Laboratory and the Computational Sciences and Engineering (CSE) Division at the Oak Ridge National Laboratory.

Appreciation

We would like to thank our sponsors for their kind donations. In addition, we thank the SIGKDD'10 organizers, the authors of the submitted papers, the invited speakers, and the members of the Program Committee for their respective and collective efforts to make this workshop possible.

The workshop proceedings were compiled by Dr. Olufemi A. Omitaomu of the Computational Sciences and Engineering Division at Oak Ridge National Laboratory. The workshop proceedings have been co-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.

References

[1] Bhat, C., Ganguly A.R., Gehrke, J., Giannella, C., McGranaghan, M., and Melby, P. (2009). National Science Foundation Summit on the Next Generation of Data Mining for Dealing with Energy, Greenhouse Emissions, and Transportation Challenges (NGDM '09), Committee Report, November 2009 – With contributions from Olufemi A. Omitaomu (Unpublished).

[2] Ganguly, A., J. Gama, O. Omitaomu, M. Gaber, R. Vatsavai. Proceedings of the First International Workshop on Knowledge Discovery from Sensor Data, 13th ACM SIGKDD Conference, 2007, (Available at http://www.ornl.gov/sci/knowledgediscovery/SensorKDD-2007).

[3] Vatsavai, R. O. Omitaomu, J. Gama, M. Gaber, N. Chawla, A. Ganguly. Proceedings of the Second International Workshop on Knowledge Discovery from Sensor Data, 14th ACM SIGKDD Conference, 2008, (Available at http://www.ornl.gov/sci/knowledgediscovery/SensorKDD-2008).

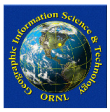
[4] Omitaomu, O., A. Ganguly, J. Gama, R. Vatsavai, N. Chawla, M. Gaber. Proceedings of the Third International Workshop on Knowledge Discovery from Sensor Data, 14th ACM SIGKDD Conference, 2009, (Available at http://www.ornl.gov/sci/knowledgediscovery/SensorKDD-2009).
[5] Ganguly, Auroop R., Joao Gama, Olufemi A. Omitaomu, Mohamed M. Gaber, and Ranga Raju Vatsavai (2009). *Knowledge Discovery from Sensor Data*. New York, NY: CRC Press, January.
[6] Mohamed Medhat Gaber, Ranga raju Vatsavai, Olufemi A. Omitaomu, Joao Gama, Nitesh V. Chawla, and Auroop R. Ganguly (Editors) (2010). Knowledge Discovery from Sensor Data. *Lecture Notes in Computer Science*. Springer.
[7] SensorNet® Program (Program website: http://www.sensornet.gov/).
[8] U.S. Department of Energy - Smart Grid Program (Website: http://www.oe.energy.gov/smartgrid.htm).
[9] Oak Ridge Climate Change Science Institute (Website: http://climatechangescience.ornl.gov/).

Dr. Olufemi A. Omitaomu (omitaomuoa@ornl.gov; +1-865-241-4310)
Dr. Varun Chandola (chandolav@ornl.gov); +1-865-576-6192)
Dr. Auroop R. Ganguly (gangulyar@ornl.gov; +1-865-241-1305)
Dr. Ranga Raju Vatsavai (vatsavairr@ornl.gov; +1-865-576-3569)
Oak Ridge National Laboratory, 1 Bethel Valley Rd, MS-6017, Oak Ridge, TN, USA.

Prof. Joao Gama (jgama@fep.up.pt; +351-22-339-2094)
LIAAD-INESC Porto LA, University of Porto, Rua de Ceuta, Porto, Portugal.

Prof. Mohamed Medhat Gaber (mohamed.m.gaber@gmail.com; +44 23 92 84 6416)
University of Portsmouth, School of Computing, Hampshire, UK

Prof. Nitesh V. Chawla (nchawla@cse.nd.edu; +1-574-631-8716)
University of Notre Dame, Dept. of Computer Science & Engineering, IN, USA.



Organizers

Workshop Chairs:

1. **Olufemi A. Omitaomu,** Oak Ridge National Laboratory, TN, USA.
2. **Varun Chandola,** Oak Ridge National Laboratory, TN, USA.
3. **Auroop R. Ganguly,** Oak Ridge National Laboratory, TN, USA.
4. **Joao Gama,** University of Porto, Portugal.
5. **Ranga Raju Vatsavai**, Oak Ridge National Laboratory, TN, USA.
6. **Nitesh V. Chawla**, University of Notre Dame, IN, USA.
7. **Mohamed Medhat Gaber,** Monash University, Australia.

Program Committee (In alphabetical order of last name):

1. **Adedeji B. Badiru**, Air Force Institute of Technology, Dayton, OH, USA.
2. **Budhendra L. Bhaduri**, Oak Ridge National Laboratory, TN, USA.
3. **Eric Auriol**, CLIMPACT, Paris, France.
4. **Albert Bifet**, University Polytechnica, Catalunya, Spain.
5. **Michaela Black**, University of Ulster, Coleraine, Northern Ireland, UK.
6. **Jose del Campo-Avila**, Universidad de Malaga, Spain.
7. **Andre Carvalho**, University of Sao Paulo, Brazil.
8. **Sanjay Chawla**, University of Sydney, Australia.
9. **Diane Cook**, Washington State University, Pullman, WA, USA.
10. **Alfredo Cuzzocrea**, University of Calabria, Italy.
11. **Jing (David) Dai**, IBM Watson Research Center, USA.
12. **Christie Ezeife**, University of Windsor, Canada.
13. **David J. Erickson III**, Oak Ridge National Laboratory, TN, USA.
14. **Yi Fang**, Purdue University, West Lafayette, IN, USA.
15. **Francisco Ferrer**, University of Seville, Spain.
16. **James H. Garrett**, Carnegie Mellon University, Pittsburgh, PA, USA.
17. **Joydeep Ghosh**, University of Texas, Austin, TX, USA.
18. **Bryan L. Gorman**, Oak Ridge National Laboratory, TN, USA.
19. **Sara Graves**, University of Alabama, Huntsville, AL, USA.
20. **Ray Hickey**, University of Ulster, Coleraine, Northern Ireland, UK.
21. **Forrest Hoffman**, Oak Ridge National Laboratory, TN, USA.
22. **Luke (Jun) Huan**, University of Kansas, Lawrence, KS, USA.
23. **Volkan Isler**, University of Minnesota, Minneapolis, MN, USA.
24. **Vandana Janeja**, University of Maryland, Baltimore County, MD, USA.
25. **Yu (Cathy) Jiao**, Oak Ridge National Laboratory, TN, USA.
26. **Ralf Klinkenberg**, University of Dortmund, Germany.
27. **Miroslav Kubat**, University Miami, FL, USA.
28. **Vipin Kumar**, University of Minnesota, Minneapolis, MN, USA.
29. **Mark Last**, Ben-Gurion University, Israel.
30. **Chang-Tien Lu**, Virginia Tech., VA, USA.
31. **Elaine Parros Machado de Sousa**, University of Sao Paulo, Brazil.
32. **Sameep Mehta**, IBM Research, India.
33. **Laurent Mignet**, IBM Research, India.

34. **S. Muthu Muthukrishnan**, Rutgers University and AT&T Research, NJ, USA.
35. **George Ostrouchov**, Oak Ridge National Laboratory, TN, USA.
36. **Guangzhi Qu**, Oakland University, Rochester, MI, USA.
37. **Rahul Ramachandran**, University of Alabama, Huntsville, AL, USA.
38. **Pedro Rodrigues**, University of Porto, Portugal.
39. **Josep Roure**, Carnegie Mellon University, Pittsburgh, PA, USA.
40. **Bernhard Seeger**, University Marburg, Germany.
41. **Cyrus Shahabi**, University of Southern California, USA.
42. **Shashi Shekhar**, University of Minnesota, Minneapolis, MN, USA.
43. **Mallikarjun Shankar**, Oak Ridge National Laboratory, Oak Ridge, TN, USA.
44. **Lucio Soibelman**, Carnegie Mellon University, Pittsburgh, PA, USA.
45. **Alexandre Sorokine**, Oak Ridge National Laboratory, Oak Ridge, TN, USA.
46. **Eduardo J. Spinosa**, University of Sao Paulo, Brazil.
47. **Karsten Steinhaeuser**, University of Notre Dame, IN and Oak Ridge National Laboratory, TN, USA.
48. **Nithya Vijayakumar**, Cisco Systems, Inc., USA.
49. **Gary Weiss**, Fordham University, NY, USA.
50. **Peng Xu**, ExxonMobil Corporate Strategic Research, NJ, USA.
51. **Eiko Yoneki**, University of Cambridge, UK.
52. **Philip S. Yu**, IBM Watson Research Center, Yorktown Heights, NY, USA.

Workshop website: http://www.ornl.gov/sci/knowledgediscovery/SensorKDD-2010

Table of Contents

FULL RESEARCH PAPERS

Activity Recognition using Cell Phone Accelerometers

Jennifer R. Kwapisz, Gary M. Weiss, Samuel A. Moore

Department of Computer and Information Science
Fordham University
441 East Fordham Road
Bronx, NY 10458
{kwapisz, gweiss, asammoore}@cis.fordham.edu

ABSTRACT

Mobile devices are becoming increasingly sophisticated and the latest generation of smart cell phones now incorporates many diverse and powerful sensors. These sensors include GPS sensors, vision sensors (i.e., cameras), audio sensors (i.e., microphones), light sensors, temperature sensors, direction sensors (i.e., magnetic compasses), and acceleration sensors (i.e., accelerometers). The availability of these sensors in mass-marketed communication devices creates exciting new opportunities for data mining and data mining applications. In this paper we describe and evaluate a system that uses phone-based accelerometers to perform *activity recognition*, a task which involves identifying the physical activity a user is performing. To implement our system we collected labeled accelerometer data from twenty-nine users as they performed daily activities such as walking, jogging, climbing stairs, sitting, and standing, and then aggregated this time series data into examples that summarize the user activity over 10-second intervals. We then used the resulting training data to induce a predictive model for activity recognition. This work is significant because the activity recognition model permits us to gain useful knowledge about the habits of millions of users passively—just by having them carry cell phones in their pockets. Our work has a wide range of applications, including automatic customization of the mobile device's behavior based upon a user's activity (e.g., sending calls directly to voicemail if a user is jogging) and generating a daily/weekly activity profile to determine if a user (perhaps an obese child) is performing a healthy amount of exercise.

Categories and Subject Descriptors

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

General Terms

Algorithms, Design, Experimentation, Human Factors

Keywords

Sensor mining, activity recognition, induction, cell phone, accelerometer, sensors

SensorKDD '10, July 25, 2010, Washington, DC, USA.

1. INTRODUCTION

Mobile devices, such as cellular phones and music players, have recently begun to incorporate diverse and powerful sensors. These sensors include GPS sensors, audio sensors (i.e., microphones), image sensors (i.e., cameras), light sensors, temperature sensors, direction sensors (i.e., compasses) and acceleration sensors (i.e., accelerometers). Because of the small size of these "smart" mobile devices, their substantial computing power, their ability to send and receive data, and their nearly ubiquitous use in our society, these devices open up exciting new areas for data mining research and data mining applications. The goal of our WISDM (Wireless Sensor Data Mining) project [19] is to explore the research issues related to mining sensor data from these powerful mobile devices and to build useful applications. In this paper we explore the use of one of these sensors, the accelerometer, in order to identify the activity that a user is performing—a task we refer to as activity recognition.

We have chosen Android-based cell phones as the platform for our WISDM project because the Android operating system is free, open-source, easy to program, and expected to become a dominant entry in the cell phone marketplace (this is clearly happening). Our project currently employs several types of Android phones, including the Nexus One, HTC Hero, and Motorola Backflip. These phones utilize different cellular carriers, although this is irrelevant for our purposes since all of the phones can send data over the Internet to our server using a standard interface. However, much of the data in this work was collected directly from files stored on the phones via a USB connection, but we expect this mode of data collection to become much less common in future work.

All of these Android phones, as well as virtually all new smart phones and smart music players, including the iPhone and iPod Touch [2], contain tri-axial accelerometers that measure acceleration in all three spatial dimensions. These accelerometers are also capable of detecting the orientation of the device (helped by the fact that they can detect the direction of Earth's gravity), which can provide useful information for activity recognition. Accelerometers were initially included in these devices to support advanced game play and to enable automatic screen rotation but they clearly have many other applications. In fact, there are many useful applications that can be built if accelerometers can be used to recognize a user's activity. For example, we can automatically monitor a user's activity level and generate daily, weekly, and monthly activity reports, which could be automatically emailed to the user. These reports would indicate an overall activity level,

which could be used to gauge if the user is getting an adequate amount of exercise and estimate the number of daily calories expended. These reports could be used to encourage healthy practices and might alert some users to how sedentary they or their children actually are. The activity information can also be used to automatically customize the behavior of the mobile phone. For example, music could automatically be selected to match the activity (e.g., "upbeat" music when the user is running) or send calls directly to voicemail when the user is exercising. There are undoubtedly numerous other instances where it would be helpful to modify the behavior of the phone based on the user activity and we expect that many such applications will become available over the next decade.

In order to address the activity recognition task using supervised learning, we first collected accelerometer data from twenty-nine users as they performed activities such as walking, jogging, ascending stairs, descending stairs, sitting, and standing. We then aggregated this raw time series accelerometer data into examples, as described in Section 2.2, where each example is labeled with the activity that occurred while that data was being collected. We then built predictive models for activity recognition using three classification algorithms.

The topic of accelerometer-based activity recognition is not new. Bao & Intille [3] developed an activity recognition system to identify twenty activities using bi-axial accelerometers placed in five locations on the user's body. Additional studies have similarly focused on how one can use a variety of accelerometer-based devices to identify a range of user activities [4-7, 9-16, 21]. Other work has focused on the applications that can be built based on accelerometer-based activity recognition. This work includes identifying a user's activity level and predicting their energy consumption [8], detecting a fall and the movements of user after the fall [12], and monitoring user activity levels in order to promote health and fitness [1]. Our work differs from most prior work in that we use a commercial mass-marketed device rather than a research-only device, we use a single device conveniently kept in the user's pocket rather than multiple devices distributed across the body, and we require no additional actions by the user. Also, we have generated and tested our models using more users (twenty-nine) than most previous studies and expect this number to grow substantially since we are continuing to collect data. The few studies that have involved commercial devices such as smart phones have focused either on a very small set of users [21] or have trained models for particular users [4] rather than creating a universal model that can be applied to any user.

Our work makes several contributions. One contribution is the data that we have collected and continue to collect, which we plan to make public in the future. This data can serve as a resource to other researchers, since we were unable to find such publically available data ourselves. We also demonstrate how raw time series accelerometer data can be transformed into examples that can be used by conventional classification algorithms. We demonstrate that it is possible to perform activity recognition with commonly available (nearly ubiquitous) equipment and yet achieve highly accurate results. Finally, we believe that our work will help bring attention to the opportunities available for mining wireless sensor data and will stimulate additional work in this area.

The remainder of this paper is structured as follows. Section 2 describes the process for addressing the activity recognition task, including data collection, data preprocessing, and data transformation. Section 3 describes our experiments and results. Related work is described in Section 4 and Section 5 summarizes our conclusions and discusses areas for future research.

2. THE ACTIVITY RECOGNITION TASK

In this section we describe the activity recognition task and the process for performing this task. In Section 2.1 we describe our protocol for collecting the raw accelerometer data, in Section 2.2 we describe how we preprocess and transform the raw data into examples, and in Section 2.3 we describe the activities that will be predicted/identified.

2.1 Data Collection

In order to collect data for our supervised learning task, it was necessary to have a large number of users carry an Android-based smart phone while performing certain everyday activities. Before collecting this data, we obtained approval from the Fordham University IRB (Institutional Review Board) since the study involved "experimenting" on human subjects and there was some risk of harm (e.g., the subject could trip while jogging or climbing stairs). We then enlisted the help of twenty-nine volunteer subjects to carry a smart phone while performing a specific set of activities. These subjects carried the Android phone in their front pants leg pocket and were asked to walk, jog, ascend stairs, descend stairs, sit, and stand for specific periods of time.

The data collection was controlled by an application we created that executed on the phone. This application, through a simple graphical user interface, permitted us to record the user's name, start and stop the data collection, and label the activity being performed. The application permitted us to control what sensor data (e.g., GPS, accelerometer) was collected and how frequently it was collected. In all cases we collected the accelerometer data every 50ms, so we had 20 samples per second. The data collection was supervised by one of the WISDM team members to ensure the quality of the data.

2.2 Feature Generation & Data Transformation

Standard classification algorithms cannot be directly applied to raw time-series accelerometer data. Instead, we first must transform the raw time series data into examples [18]. To accomplish this we divided the data into 10-second segments and then generated features that were based on the 200 readings contained within each 10-second segment. We refer to the duration of each segment as the example duration (ED). We chose a 10-second ED because we felt that it provided sufficient time to capture several repetitions of the (repetitive) motions involved in some of the six activities. Although we have not performed experiments to determine the optimal example duration value, we did compare the results for a 10-second and 20-second ED and the 10-second ED yielded slightly better results (as well as twice as many training examples).

Next we generated informative features based on the 200 raw accelerometer readings, where each reading contained an x, y, and z value corresponding to the three axes/dimensions (see Figure 1). We generated a total of forty-three summary features, although these are all variants of just six basic features. The features are

described below, with the number of features generated for each feature-type noted in brackets:

- Average[3]: Average acceleration (for each axis)
- Standard Deviation[3]: Standard deviation (for each axis)
- Average Absolute Difference[3]: Average absolute difference between the value of each of the 200 readings within the ED and the mean value over those 200 values (for each axis)
- Average Resultant Acceleration[1]: Average of the square roots of the sum of the values of each axis squared $\sqrt{(x_i^2 + y_i^2 + z_i^2)}$ over the ED
- Time Between Peaks[3]: Time in milliseconds between peaks in the sinusoidal waves associated with most activities (for each axis)
- Binned Distribution[30]: We determine the range of values for each axis (maximum – minimum), divide this range into 10 equal sized bins, and then record what fraction of the 200 values fell within each of the bins.

The "time between peaks" feature requires further explanation. The repetitive activities, like walking, tend to generate repeating waves for each axis and this feature tries to measure the time between successive peaks. To estimate this value, for each example we first identify all of the peaks in the wave using a heuristic method and then identify the highest peak for each axis. We then set a threshold based on a percentage of this value and find the other peaks that met or exceed this threshold; if no peaks meet this criterion then the threshold is lowered until we find at least three peaks. We then measure the time between successive peaks and calculate the average. For samples where at least three peaks could not be found, the time between peaks is marked as unknown. This method was able to accurately find the time between peaks for the activities that had a clear repetitive pattern, like walking and jogging. Certainly more sophisticated schemes will be tried in the future.

The number of examples generated per user for each activity varies. These differences are due to the time limitations that some users may have or physical limitations that impact the time they spend on each activity. Our data set is summarized in Section 3.1.

2.3 The Activities

In this study we consider six activities: walking, jogging, ascending stairs, descending stairs, sitting, and standing. We selected these activities because they are performed regularly by many people in their daily routines. The activities also involve motions that often occur for substantial time periods, thus making them easier to recognize. Furthermore, most of these activities involve repetitive motions and we believe this should also make the activities easier to recognize. When we record data for each of these activities, we record acceleration in three axes. For our purposes, the z-axis captures the forward movement of the leg and the y-axis captures the upward and downward motion. The x-axis captures horizontal movement of the user's leg. Figure 1 demonstrates these axes relative to a user.

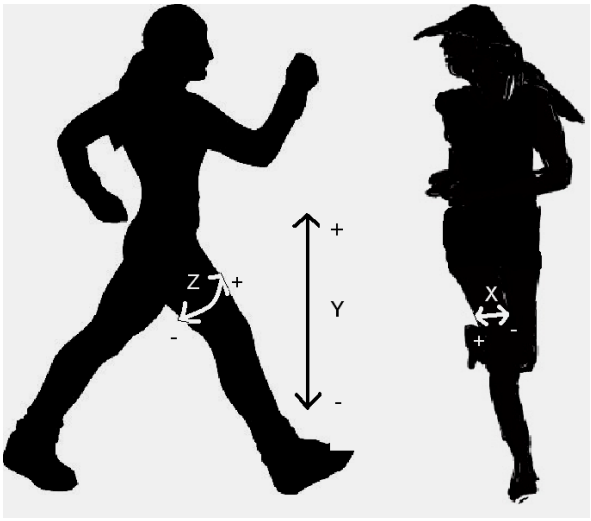
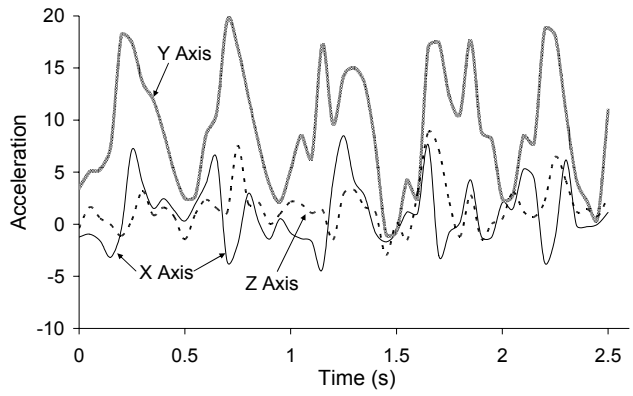


Figure 1: Axes of Motion Relative to User

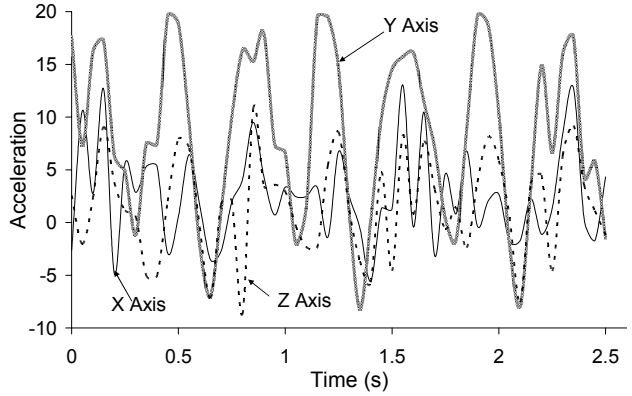
Figure 2 plots the accelerometer data for a typical user, for all three axes and for each of the six activities. It is clear that sitting and standing (Figure 2e,f) do not exhibit periodic behavior but do have distinctive patterns, based on the relative magnitudes of the x, y, and z, values, while the four other activities (Figure 2a-d), which involve repetitive motions, do exhibit periodic behavior. Note that for most activities the y values have the largest accelerations. This is a consequence of Earth's gravitational pull, which causes the accelerometer to measure a value of 9.8 m/s^2 in the direction of the Earth's center. For all activities except sitting this direction corresponds to the y axis (see Figure 1).

The periodic patterns for walking, jogging, ascending stairs, and descending stairs (Figure 2a-d) can be described in terms of the time between peaks and by the relative magnitudes of the acceleration values. The plot for walking, shown in Figure 2a, demonstrates a series of high peaks for the y-axis, spaced out at approximately ½ second intervals. The peaks for the z-axis acceleration data echo these peaks but with a lower magnitude. The distance between the peaks of the z-axis and y-axis data represent the time of one stride. The x-axis values (side to side) have an even lower magnitude but nonetheless mimic the peaks associated with the other axes. For jogging, similar trends are seen for the z-axis and y-axis data, but the time between peaks is less (~¼ second), as one would expect. As one might expect, the range of y-axis acceleration values for jogging is greater than for walking, although the shift is more noticeable in the negative direction.

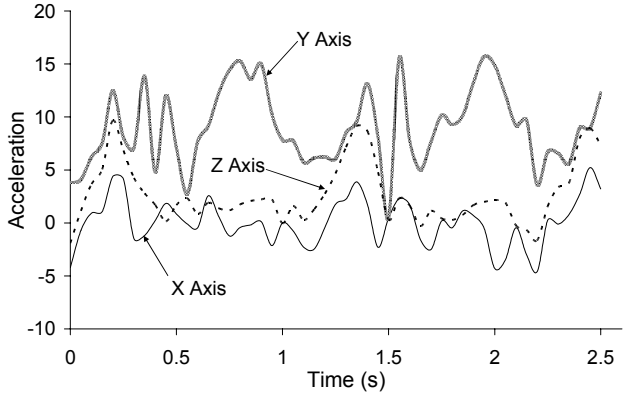
For descending stairs, one observes a series of small peaks for y-axis acceleration that take place every ~½ second. Each small peak represents movement down a single stair. The z-axis values show a similar trend with negative acceleration, reflecting the regular movement down each stair. The x-axis data shows a series of semi-regular small peaks, with acceleration vacillating again between positive and negative values. For ascending stairs, there are a series of regular peaks for the z-axis data and y-axis data as well; these are spaced approximately ~¾ seconds apart, reflecting the longer time it takes to climb up stairs.



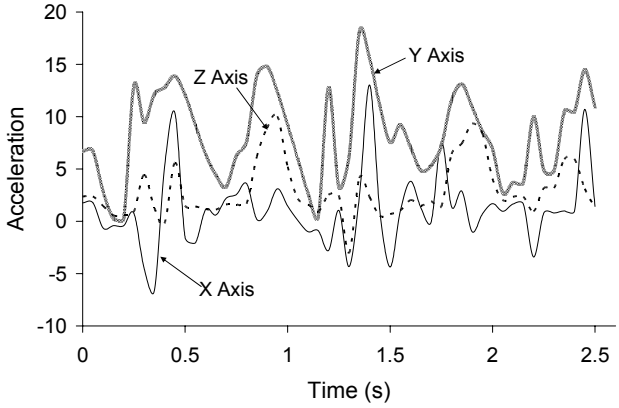
(a) Walking



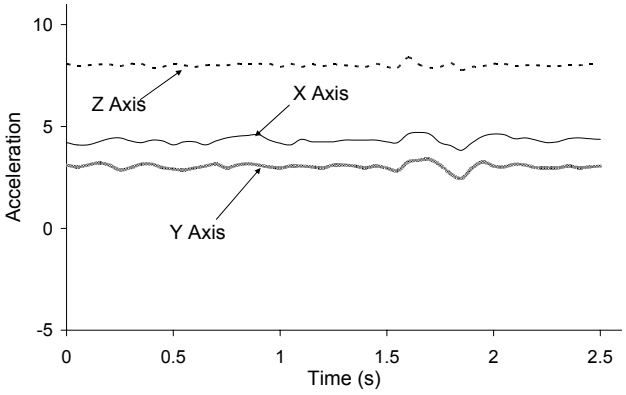
(b) Jogging



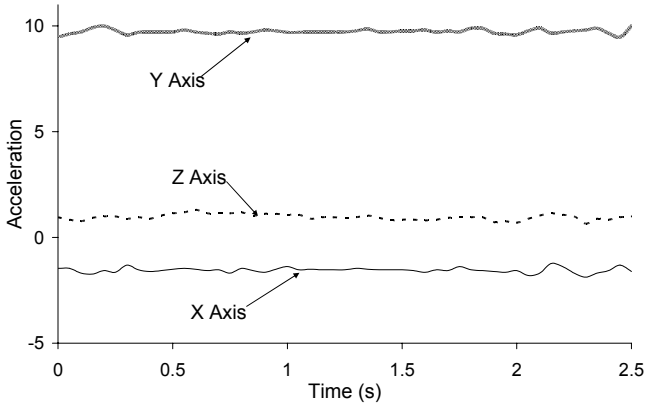
(c) Ascending Stairs



(d) Descending Stairs



(e) Sitting



(f) Standing

Figure 2: Acceleration Plots for the Six Activities (a-f)

As one would expect, sitting and standing do not exhibit any regular periodic behavior and all of the acceleration values are relatively constant. As mentioned earlier, the primary differences between these activities is the relative magnitudes of values for each axis, due to the different orientations of the device with respect to the Earth when the user is sitting and standing Thus it appears easy to differentiate between sitting and standing, even though neither involves much movement. Note that because the accelerometers are themselves able to determine orientation with respect to the Earth's gravitational field, it would be relatively straightforward to compensate/correct for any changes in the cell phone's orientation due to the phone shifting position in a user's pocket. We plan to implement this correction in future work.

3. EXPERIMENTS

In this section we describe our experiments and then present and discuss our results for the activity recognition task.

3.1 Description of Experiments

Our experiments first require us to collect the labeled raw accelerometer data and then transform that data into examples. This process was described in Section 2. The resulting examples contain 43 features and cover twenty-nine users. This forms the data set, described in Table 1, which is subsequently used for training and testing. The last row in Table 1 shows the percentage of the total examples associated with each activity.

Table 1: Number of Examples per User and Activity

ID	Walk	Jog	Up	Down	Sit	Stand	Total
1	74	15	13	25	17	7	**151**
2	48	15	30	20	0	0	**113**
3	62	58	25	23	13	9	**190**
4	65	57	25	22	6	8	**183**
5	65	54	25	25	77	27	**273**
6	62	54	16	19	11	8	**170**
7	61	55	13	11	9	4	**153**
8	57	54	12	13	0	0	**136**
9	31	59	27	23	13	10	**163**
10	62	52	20	12	16	9	**171**
11	64	55	13	12	8	9	**161**
12	36	63	0	0	8	6	**113**
13	60	62	24	15	0	0	**161**
14	62	0	7	8	15	10	**102**
15	61	32	18	18	9	8	**146**
16	65	61	24	20	0	8	**178**
17	70	0	15	15	7	7	**114**
18	66	59	20	20	0	0	**165**
19	69	66	41	15	0	0	**191**
20	31	62	16	15	4	3	**131**
21	54	62	15	16	12	9	**168**
22	33	61	25	10	0	0	**129**
23	30	5	8	10	7	0	**60**
24	62	0	23	21	8	15	**129**
25	67	64	21	16	8	7	**183**
26	85	52	0	0	14	17	**168**
27	84	70	24	21	11	13	**223**
28	32	19	26	22	8	15	**122**
29	65	55	19	18	8	14	**179**
Sum	**1683**	**1321**	**545**	**465**	**289**	**223**	**4526**
%	**37.2**	**29.2**	**12.0**	**10.2**	**6.4**	**5.0**	**100**

Note that certain activities contain fewer examples than others, mainly because the users were not asked to perform strenuous activities (e.g., jogging, climbing stairs) for very long and because we thought that the patterns in other activities (e.g., standing) would become apparent quickly so that there would be no need to waste the users time literally "standing around." Furthermore, certain activities, like standing and sitting, were only added after the study began, so we have no data for these activities for some users.

Once the data set was prepared, we used three classification techniques from the WEKA data mining suite [20] to induce models for predicting the user activities: decision trees (J48), logistic regression and multilayer neural networks. In each case we used the default settings. We used ten-fold cross validation for all experiments and all results are based on these ten runs.

3.2 Results

The summary results for our activity recognition experiments are presented in Table 2. This table specifies the predictive accuracy associated with each of the activities, for each of the three learning algorithms and for a simple "straw man" strategy. The straw man strategy always predicts the specified activity (i.e., walking for the first row in Table 2 and jogging for the second row of Table 2) or, when assessing the overall performance of the classifier (i.e., the last row of Table 2), always predicts the most frequently occurring activity, which happens to be walking. The baseline straw man strategy allows us to consider the degree of class imbalance when evaluating the performance of the activity recognition system.

Table 2: Accuracies of Activity Recognition

	% of Records Correctly Predicted			
	J48	**Logistic Regression**	**Multilayer Perceptron**	**Straw Man**
Walking	89.9	**93.6**	91.7	37.2
Jogging	96.5	98.0	**98.3**	29.2
Upstairs	59.3	27.5	**61.5**	12.2
Downstairs	**55.5**	12.3	44.3	10.0
Sitting	**95.7**	92.2	95.0	6.4
Standing	**93.3**	87.0	91.9	5.0
Overall	85.1	78.1	**91.7**	37.2

Table 2 demonstrates that in most cases we can achieve high levels of accuracy. For the two most common activities, walking and jogging, we generally achieve accuracies above 90%. Jogging appears easier to identify than walking, which seems to make sense, since jogging involves more extreme changes in acceleration. It appears much more difficult to identify the two stair climbing activities, but as we shall see shortly, that is because those two similar activities are often confused with one another. Note that although there are very few examples of sitting and standing, we can still identify these activities quite well, because, as noted earlier, the two activities cause the device to change orientation and this is easily detected from the accelerometer data. Our results indicate that none of the three learning algorithms consistently performs best, but the multilayer perceptron does perform best overall. More detailed results are presented in Tables

3-5, which show the confusion matrices associated with each of the three learning algorithms.

Table 3: Confusion Matrix for J48

		Predicted Class					
		Walk	Jog	Up	Down	Sit	Stand
Actual Class	Walk	**1513**	14	72	82	2	0
	Jog	16	**1275**	16	12	1	1
	Up	88	23	**323**	107	2	2
	Down	99	13	92	**258**	1	2
	Sit	4	0	2	3	**270**	3
	Stand	4	1	2	7	1	**208**

Table 4: Confusion Matrix for Logistic Regression

		Predicted Class					
		Walk	Jog	Up	Down	Sit	Stand
Actual Class	Walk	**1575**	14	53	36	2	3
	Jog	15	**1294**	6	6	0	0
	Up	277	36	**150**	77	1	4
	Down	259	6	136	**57**	3	4
	Sit	1	0	4	11	**260**	6
	Stand	3	1	7	3	15	**194**

Table 5: Confusion Matrix for Multilayer Perceptron

		Predicted Class					
		Walk	Jog	Up	Down	Sit	Stand
Actual Class	Walk	**1543**	5	73	60	1	1
	Jog	3	**1299**	16	3	0	0
	Up	84	24	**335**	98	2	2
	Down	108	10	136	**206**	2	3
	Sit	0	2	4	1	**268**	7
	Stand	1	0	5	4	8	**205**

The most important activities to analyze are the climbing-up and climbing-down stair activities, since these were the only activities that that were difficult to recognize. The confusion matrices indicate that many of the prediction errors are due to confusion between these two activities. If we focus on the results for the J48 decision tree model in Table 3, we see that when we are climbing up stairs the most common incorrect classification occurs when we predict "downstairs," which occurs 107 times and accounts for a decrease in accuracy of 19.6% (107 errors out of 545). When the actual activity is climbing downstairs, walking slightly outpaces "upstairs" in terms of the total number of errors (99 vs. 92), but this is only because walking occurs more than three times as often as climbing upstairs in our dataset. If we look at Figures 2a, 2c, and 2d, we see that the patterns in acceleration data between "walking", "ascending stairs" and "descending stairs" are somewhat similar. To limit the confusion between the ascending and descending stair activities, we ran another set of experiments where we combine ascending stairs and descending stairs into one activity. The resulting confusion matrix for the J48 algorithm is shown in Table 6 (in the interest of space we do not show them for the other two algorithms). We see that the results are substantially improved, although stair climbing is still the hardest activity to recognize.

Table 6: Confusion Matrix for J48 Model (Stairs Combined)

		Predicted Class					Accur. (%)
		Walk	Jog	Stairs	Sit	Stand	
Actual Class	Walk	**1524**	7	148	2	2	90.6
	Jog	10	**1280**	31	0	0	96.9
	Stairs	185	33	**784**	4	4	77.6
	Sit	4	0	2	**272**	4	96.5
	Stand	3	1	10	0	**209**	93.7

4. RELATED WORK

Activity recognition has recently gained attention as a research topic because of the increasing availability of accelerometers in consumer products, like cell phones, and because of the many potential applications. Some of the earliest work in accelerometer-based activity recognition focused on the use of multiple accelerometers placed on several parts of the user's body. In one of the earliest studies of this topic, Bao & Intille [3] used five biaxial accelerometers worn on the user's right hip, dominant wrist, non-dominant upper arm, dominant ankle, and non-dominant thigh in order to collect data from 20 users. Using decision tables, instance-based learning, C4.5 and Naïve Bayes classifiers, they created models to recognize twenty daily activities. Their results indicated that the accelerometer placed on the thigh was most powerful for distinguishing between activities. This finding supports our decision to have our test subjects carry the phone in the most convenient location—their pants pocket.

Other researchers have, like Bao & Intille, used multiple accelerometers for activity recognition. Krishnan et. al. [9] collected data from three users using two accelerometers to recognize five activities—walking, sitting, standing, running, and lying down. This paper claimed that data from a thigh accelerometer was insufficient for classifying activities such as sitting, lying down, walking, and running, and thus multiple accelerometers were necessary (a claim our research contradicts). In another paper, Krishnan et. al. [10] examined seven lower body activities using data collected from ten subjects wearing three accelerometers. This method was tested in supervised and semi-naturalistic settings. Tapia et. al. [16] collected data from five accelerometers placed on various body locations for twenty-one users and used this data to implement a real-time system to recognize thirty gymnasium activities. A slight increase in performance was made by incorporating data from a heart monitor in addition to the accelerometer data. Mannini and Sabitini [23] used five tri-axial accelerometers attached to the hip, wrist, arm, ankle, and thigh in order to recognize twenty activities from thirteen users. Various learning methods were used to recognize three "postures" (lying, sitting, and standing) and five "movements" (walking, stair climbing, running, and cycling). Foerster and Fahrenberg [28] used data from five accelerometers in one set of experiments and from two of those accelerometers in another for activity recognition. Thirty-one male subjects participated in the study and a hierarchical classification model was built in order to distinguish between postures such as sitting and lying at specific angles, and motions such as walking and climbing stairs at different speeds.

Researchers have used a combination of accelerometers and other sensors to achieve activity recognition. Parkka et. al. [27] created

a system using twenty different types of sensors (including an accelerometer worn on the chest and one worn on the wrist) in order to recognize activities such as lying, standing, walking, running, football, swinging, croquet, playing ball, and using the toilet in specific locations. Lee and Mase [25] created a system to recognize a user's location and activities, including sitting, standing, walking on level ground, walking upstairs, and walking downstairs using a sensor module that consisted of a biaxial accelerometer and an angular velocity sensor worn in the pocket combined with a digital compass worn at the user's waist. Subramayana et. al. [26] addressed similar activities by building a model using data from a tri-axial accelerometer, two microphones, phototransistors, temperature and barometric pressure sensors, and GPS to distinguish between a stationary state, walking, jogging, driving a vehicle, and climbing up and down stairs.

While these systems using multiple accelerometers or a combination of accelerometers and other sensors were capable of identifying a wide range of activities, they are not very practical because they involve the user wearing multiple sensors distributed across their body. This could work for some short term, small scale, highly specialized applications (e.g., in a hospital setting) but would certainly not work for the applications that we envision.

Some studies have also focused on combining multiple types of sensors in addition to accelerometers for activity recognition. Maurer et al. [13] used "eWatch" devices placed on the belt, shirt pocket, trouser pocket, backpack, and neck to recognize the same six activities that we consider in our study. Each "eWatch" consisted of a biaxial accelerometer and a light sensor. Decision trees, k-Nearest Neighbor, Naïve Bayes, and Bayes Net classifiers with five-fold cross validation were used for learning. Choudhury et. al [6] used a multimodal sensor device consisting of seven different types of sensors (tri-axial accelerometer, microphone, visible light phototransitor, barometer, visible+IR light sensor, humidity/temperature reader, and compass) to recognize activities such as walking, sitting, standing, ascending stairs, descending stairs, elevator moving up and down, and brushing one's teeth. Cho et. al. [5] used a single tri-axial accelerometer, along with an embedded image sensor worn at the user's waist, to identify nine activities. Although these multi-sensor approaches do indicate the great potential of mobile sensor data as more types of sensors are being incorporated into devices, our approach shows that only one type of sensor—an accelerometer—is needed to recognize most daily activities. Thus our method offers a straightforward and easily-implementable approach to accomplish this task.

Other studies, like our own, have focused on the use of a single accelerometer for activity recognition. Long, Yin, and Aarts [22] collected accelerometer data from twenty-four users using a tri-axial accelerometer worn without regard for orientation at the user's waist. Data was collected naturalistically, and decision trees as well as a Bayes classifier combined with a Parzen window estimator were used to recognize walking, jogging, running, cycling, and sports. Lee et. al. [24] used a single accelerometer attached to the left waists of five users. Standing, sitting, walking, lying, and running were all recognized with high accuracies using fuzzy c-means classification. However unlike these studies, which use devices specifically made for research purposes, our method utilizes commercial devices that are widely-available without any additional specialized equipment. This approach enables make practical real-world applications for our models.

Several researchers have considered the use of widely-available mobile devices such as cell phones to address the activity recognition problem. However the earlier approaches did not take advantage of the sensors incorporated into the mobile devices themselves. For example, Gyorbiro et. al. [7] used "MotionBands" attached to the dominant wrist, hip, and ankle of each subject to distinguished between six different motion patterns. Each MotionBand contained a tri-axial accelerometer, magnetometer, and gyroscope. As the MotionBand collected data, the data was then transmitted to a smart phone carried by the user to be stored. Ravi et. al. [15] collected data from two users wearing a single accelerometer-based device and then transmitted this data to the HP iPAQ mobile device carried by the user. Using this data for activity recognition, researchers compared the performance of eighteen different classifiers. Lester et. al. [11] used accelerometer data, along with audio and barometric sensor data, to recognize eight daily activities from a small set of users. While these studies could have used a cell phone to generate the accelerometer data, they did not do this. Instead, the data was generated using distinct accelerometer-based devices worn by the user and then sent to a cell phone for storage.

A few studies, like ours, did use an actual commercial mobile device to collect data for activity recognition. Such systems offer an advantage over other accelerometer-based systems because they are unobtrusive and do not require any additional equipment for data collection and accurate recognition. Miluzzo et. al. [14] explored the use of various sensors (such as a microphone, accelerometer, GPS, and camera) available on commercial smart phones for activity recognition and mobile social networking applications. In order to address the activity recognition task, they collected accelerometer data from ten users to build an activity recognition model for walking, running, sitting, and standing using J48. This model had particular difficulty distinguishing between the sitting and standing activities, a task that our models easily achieve. Yang [21] developed an activity recognition system using the Nokia N95 phone to distinguish between sitting, standing, walking, running, driving, and bicycling. This work also explored the use of an activity recognition model to construct physical activity diaries for the users. Although the study achieved relatively high accuracies of prediction, stair climbing was not considered and the system was trained and tested using data from only four users. Brezmes et. al. [4] also used the Nokia N95 phone to develop a real-time system for recognizing six user activities. In their system, an activity recognition model is trained for each user, meaning that there is no universal model that can be applied to new users, for whom no training data exists. Our models do not have this limitation.

5. CONCLUSIONS AND FUTURE WORK

In this paper we described how a smart phone can be used to perform activity recognition, simply by keeping it in ones pocket. We further showed that activity recognition can be highly accurate, with most activities being recognized correctly over 90% of the time. In addition, these activities can be recognized quickly, since each example is generated from only 10 seconds worth of data. We have several interesting applications in mind for activity recognition and plan to implement some of these applications in the near future.

Our work would not have been possible without establishing our WISDM Android-based data collection platform, and we view

this software and hardware architecture, where data is transmitted by the phone to our Internet-based server, as a key resource produced as a consequence of this work. By having this in place we will be able to mine other mobile sensor data much more quickly. This platform, as well as the data that we collected, will ultimately be made public.

We plan to improve our activity recognition in several ways. The straightforward improvements involve: 1) learning to recognize additional activities, such as bicycling and car-riding, 2) obtaining training data from more users with the expectation that this will improve our results, 3) generating additional and more sophisticated features when aggregating the raw time-series data, and 4) evaluating the impact of carrying the cell phone in different locations, such as on a belt loop. In addition, in the near future we plan to significantly enhance our WISDM platform so that we can generate results in real-time, whereas currently our results are generated off-line and are not reported back to the mobile phone and the user. We plan to provide real-time results in two ways. The first way minimizes the intelligence required on the phone by having the phone transmit the data to the Internet-based sever over the cellular connection, as usual, with the server applying the activity recognition model and transmitting the results back to the phone. In one variant, the phone will send the raw accelerometer data and in a second variant the phone will perform the data transformation step and only transmit the data when an example is generated. The second method involves implementing the activity recognition model directly on the cell phone. Given the computational power of these devices, this is certainly a feasible option. One key advantage of this method is that it removes the need for a server, which makes the solution perfectly scalable, and ensures the user's privacy, since the sensor data is kept locally on the device.

The work described in this paper is part of a larger effort to mine sensor data from wireless devices. We plan to continue our WISDM project, applying the accelerometer data to other tasks besides activity recognition and collecting and mining other sensor data, especially GPS data. We believe that mobile sensor data provides tremendous opportunities for data mining and we intend to leverage our Android-based data collection/data mining platform to the fullest extent possible.

6. REFERENCES

[1] Anderson, I., Maitland, J., Sherwood, S., Barkhuus, L., Chalmers, M., Hall, M., Brown, B., and Muller, H. 2007. Shakra: Tracking and sharing daily activity levels with unaugmented mobile phones. In *Mobile Networks and Applications*. 12(2-3).

[2] Apple iPhone and Apple iPod Touch. 2009. Apple Inc. www.apple.com.

[3] Bao, L. and Intille, S. 2004. Activity Recognition from User-Annotated Acceleration Data. *Lecture Notes Computer Science 3001*, 1-17.

[4] Brezmes, T., Gorricho, J.L., and Cotrina, J. 2009. Activity Recognition from accelerometer data on mobile phones. In *IWANN '09: Proceedings of the 10th International Work-Conference on Artificial Neural Networks*, 796-799.

[5] Cho, Y., Nam, Y., Choi, Y-J., and Cho, W-D. 2008. SmartBuckle: human activity recognition using a 3-axis accelerometer and a wearable camera. In *HealthNet.*

[6] Choudhury, T., Consolvo, S., Harrison, B., LaMarca, A., LeGrand, L., Rahimi, A., Rea, A., Borriello, G., Hemingway, B., Klasnja, P., Koscher, K., Landay, J., Lester, J., Wyatt, D., and Haehnel, D. 2008. The mobile sensing platform: An embedded activity recognition system. In *IEEE Pervasive Computing*, 7(2), 32-41.

[7] Gyorbiro, N., Fabian, A., and Homanyi, G. 2008. An activity recognition system for mobile phones. In *Mobile Networks and Applications*, 14(1), 82-91.

[8] Inooka, H., Ohtaki, Y. Hayasaka, H. Suzuki, A., and Nagatomi, R. 2006. Development of advanced portable device for daily physical assessment. In *SICE-ICASE, International Joint Conference*, 5878-5881.

[9] Krishnan, N., Colbry, D., Juillard, C., and Panchanathan, S. 2008. Real time human activity recognition using tri-Axial accelerometers. In *Sensors, Signals and Information Processing Workshop*.

[10] Krishnan, N. and Panchanathan, S. 2008. Analysis of Low Resolution Accelerometer Data for Continuous Human Activity Recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing, (ICASSP 2008).* Pages 3337-3340.

[11] Lester, J., Choudhury, T. and Borriello, G. 2006. A practical approach to recognizing physical activities. *Lecture Notes in Computer Science: Pervasive Computing*, 1–16.

[12] Mathie, M., Celler B., Lovell N., and Coster A. 2004. Classification of basic daily movements using a triaxial accelerometer. In *Medical & Biological Engineering and Computing,* 42.

[13] Maurer, U., Smailagic, A., Siewiorek, D., & Deisher, M. 2006. Activity recognition and monitoring using multiple sensors on different body positions. In *IEEE Proceedings on the International Workshop on Wearable and Implantable Sensor Networks*, 3(5).

[14] Miluzzo, E., Lane, N., Fodor, K., Peterson, R., Lu, H., Musolesi, M., Eisenman, S., Zheng, X. and Campbell, A. 2008. Sensing meets mobile social networks: The design, implementation and evaluation of the CenceMe application. In *The 6th ACM Conference on Embedded Networked Sensor Systems*, 337-350.

[15] Ravi, N., Dandekar, N. 2005. Activity recognition from accelerometer data. In *Proceedings of the Seventeenth Conference on Innovative Applications of Artificial Intelligence*.

[16] Tapia, E.M., Intille, S.S. et al. 2007. Real-Time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor. In *Proceedings of the 2007 11th IEEE International Symposium on Wearable Computers*, 1-4.

[17] Unwired View.com. 2009. Google wants to make your Android phone much smarter with accelerometer and other sensors. Stasys Bielinis.http://www.unwiredview.com/2009/05/21/google-wants-to-make-your-android-phone-much-smarter-with-accelerometer-and-other-sensors/

[18] Weiss, G. M., and Hirsh, H. 1998. Learning to predict rare events in event sequences, In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, CA, 359-363.

[19] WISDM (Wireless Sensor Data Mining) Project. Fordham University, Department of Computer and Information Science, http://storm.cis.fordham.edu/~gweiss/ wisdm/

[20] Witten, I. H. and Frank, E. Data Mining: Practical Machine Learning Tools and Techniques, 2nd ed. Morgan Kaufmann, June 2005.

[21] Yang, J. 2009. Toward physical activity diary: Motion recognition using simple acceleration features with mobile phones, In *First International Workshop on Interactive Multimedia for Consumer Electronics* at ACM Multimedia.

[22] Long, X., Yin, B., and Aarts, R.M. 2009. Single accelerometer-based daily physical activity classification. *In 31st Annual International Conference of the IEEE EMBS*, 6107-6110.

[23] Mannini, A. and Sabatini A.M. 2010. Machine learning methods for classifying human physical activity from on-body accelerometers. In *Sensors 2010*, 10, 1154-1175.

[24] Lee, M., Kim, J., Kim, K., Lee, I., Jee, S.H., and Yoo, S.K. 2009. Physical activity recognition using a single tri-axis accelerometer. In *Proceedings of the World Congress on Engineering and Computer Science 2009*, 1.

[25] Lee, S.-W. and Mase, K. 2002. Activity and location recognition using wearable sensors. In *IEEE Pervasive Computing*, 1(3):24–32.

[26] Subramanya, A., Raj, A., Bilmes, J., and Fox, D. 2006. Recognizing activities and spatial context using wearable sensors. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*.

[27] Parkka, J., Ermes, M., Korpipaa, P., Mantyjarvi, J., Peltola, J., and Korhonen, I. 2006. Activity classification using realistic data from wearable sensors. In *IEEE Transactions on Information Technology in Biomedicine*, 10(1), 119-128.

[28] Foerster F. and Fahrenberg J. 2000. Motion pattern and posture: correctly assessed by calibrated accelerometers. In *Behavior Research Methods, Instruments, & Computers*, 32(3), 450–7.

A New Algorithm Based On Sequential Pattern Mining For Person Identification In Ubiquitous Environments

Belkacem Chikhaoui
Prospectus Laboratory
University of Sherbrooke
Canada
Belkacem.Chikhaoui@USherbrooke.ca

Shengrui Wang
Prospectus Laboratory
University of Sherbrooke
Canada
Shengrui.Wang@USherbrooke.ca

Hélène Pigot
Domus Laboratory
University of Sherbrooke
Canada
Helene.Pigot@USherbrooke.ca

ABSTRACT

This paper presents an approach to person identification in ubiquitous environments. Our approach uses the sequential pattern mining principle to extract frequent patterns in data collected from the different sensors disseminated in the ubiquitous environment. In contrast with existing, intrusive, person identification algorithms that have been proposed in the literature, where the data is basically composed of audiovisual or image files recorded during experiments, our approach is fully non-intrusive and is based on event sequences collected from heterogeneous sensors. Our approach is divided into three main phases: (1) frequent pattern mining, (2) assignment of weights to extracted patterns, and (3) classification. Experiments using data collected in the Domus and Testbed smart homes demonstrate that our approach accurately identifies persons and improves classification results, outperforming two of the approaches reported in the literature.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*; I.2.6 [**Artificial Intelligence**]: Learning—*Knowledge acquisition*

General Terms

Algorithms, Design, Human Factors

Keywords

Smart homes, ubiquitous computing, sequential patterns, event sequence, episode discovery, person identification.

1. INTRODUCTION

Ubiquitous environments constitute a technological challenge for contemporary research. These environments are characterized by an increasing number of sensors, actuators, displays, devices, and computational elements embedded in everyday objects, and connected through a network [29]. The recent emergence of ubiquitous environments such as smart homes has allowed the provision of housekeeping, assistance and monitoring for chronically ill patients, and enabled persons with special needs and the elderly to receive services in their own home environments [7, 24]. Using such technology can help to reduce costs considerably, and relieve pressure on healthcare systems. However, this technology poses many challenges, such as person identification, activity recognition, assistance, monitoring, and adaptation.

Person identification in ubiquitous environments is a subject of great interest, particularly in smart homes, where specific individuals must be monitored and assisted according to their needs. To deal with this issue, several research projects have been conducted using video and image processing [31, 6, 4]; some other systems are also discussed in [3]. However, these approaches are intrusive and do not preserve personal privacy. By "intrusive", we mean that individuals are monitored by cameras which invade their privacy, or by other body-worn sensors which diminish their sense of autonomy. Little work has been done on using non-intrusive systems which utilize devices and sensors (for motion, pressure, RFID, etc.) disseminated in the environment [24, 20]. These sensors capture events about the state of the environment and any changes that occur in it. These events constitute a form of sequence. Each sequence of events is associated with a particular activity performed within the environment. The same activity can be performed by the same person in different ways, which means that the same activity can correspond to different sequences of events. This change in the person's behavior leads to the generation of a set of frequent patterns (episodes)[1] that characterize this person.

Mining event sequences is an important task for the ubiquitous computing community, and for the KDD community as well. In fact, by discovering frequent patterns, the underlying association rules, temporal constraints and progress, changes over time and expected utilities, it becomes possible to characterize the behavior of persons (and objects, such as those most purchased in a supermarket) and automate tasks such as service adaptation, activity monitoring and assistance, and many other complex activities and applications [22].

The general approaches to person identification in ubiq-

[1]We use the terms "pattern" and "episode" interchangeably

SensorKDD'10, July 25, 2010, Washington, DC, USA.

uitous environments are either intrusive, or resource- and time-consuming. In this paper we present a new algorithm for person identification, based on sequential pattern mining. Our algorithm consists of three main steps. In the first step, the frequent patterns (episodes) are extracted from the sequential dataset using the *Apriori* algorithm [2], one of the efficient pattern mining algorithms in the literature [9]. The second step evaluates the extracted episodes. A new scoring function for weighting the extracted episodes is proposed, based on the work of Chang-Rong Lin [16], and adapted to sequential patterns. This step is one of the innovations made by the method proposed in this paper. The weight assigned to an episode indicates the importance of the episode in the event sequence, and these weights improve the classification accuracy in the third step of our algorithm.

The strengths of our approach are, first, that it is based on a fully non-intrusive technology, which broadly preserves personal privacy; and second, that it takes events collected from sensors in the form of sequences, which means that our approach is generic and can be applied to any sequential data. In addition, our approach is infrastructure-independent: this means that it is applicable to evolving and scalable environments because the focus is on the generated sequences, not on the environment itself. We take advantage of the sequential form of events to extract frequent patterns that can be used to characterize each person and distinguish him or her from others living in the same environment. To our knowledge, no other paper in the literature addresses the problem of person identification by using sequential pattern mining with weights assigned to episodes. There is a single report of a study that assigned weights to sequential patterns to classify sequences [5]. However, this work used a very simple scoring function based on the pattern length, and did not take into account patterns of length 1. This motivates our effort to create a non-intrusive approach for automatic person identification, based on sequential pattern mining. We will show, through experiments, how the use of frequent pattern mining in our approach can significantly improve the quality of identification, allowing it to achieve higher classification accuracy than the existing methods.

This paper is organized as follows. Section 2 gives an overview of related work. In Section 3 we introduce our approach, presenting the overall architecture and the proposed algorithm. The experiment is described in Section 4 and the experimental results are presented in Section 5. Section 6 is devoted to a discussion of the results, followed by a conclusion in Section 7.

Contribution of this paper

This paper proposes a new algorithm for person identification in ubiquitous environments. While the problem of person identification using non-intrusive technology has recently been studied in a limited way in the literature [20, 33], this is the first comprehensive study which proposes a fully non-intrusive approach based on frequent pattern mining. In addition to the efficient algorithm proposed, our approach is based on the generation of long episodes which better characterize human behavior. The innovation of the proposed algorithm is its assignment of weights for frequent episodes, using a new scoring function that evaluates episodes with respect to the sequence as well as the class to which they belong. The episode weights allow us to find significant episodes in the event sequence, which can be used to characterize the person and distinguish him or her from other people, improving classification accuracy. Moreover, the assignment of weights reduces the dimensionality of the space by allowing consideration of significant episodes only, which is helpful in developing real-time applications. Finally, our approach is validated on real-life data which will be available on the Web very soon. This will make our experiments repeatable.

2. RELATED WORK

Several research studies reported in the literature have been interested in person identification in ubiquitous environments. J. Suiutala et al. [24] introduced methods for footstep-based person identification using a large pressure-sensitive floor with a sensory system. This approach is related to the biometric identification domain [14, 13], which includes physiological (iris, fingerprints, hand shapes, etc.) and behavioral (handwriting, speech) characteristics. The behavioral characteristics of a walking person are used to model the person's identity [24]. In this approach, the person identification system is based on sensor measurements derived from a pressure-sensitive floor. The authors used what they call ElectroMechanical Film (EMFi) [21], which senses pressure changes affecting its surface and provides footstep profiles of the walking person as an input to the identification system.

Much work on person identification has been done using cameras and machine vision methods. Pfinder [31] is a real-time system for tracking people and interpreting their behavior. The Pfinder system uses a multiclass statistical model of color and shape to obtain a 2D representation of head and hands in a wide range of viewing conditions. In [6], the authors propose a real-time face recognition system for consumer/embedded applications. The system is embedded in an interconnected home environment and allows intelligent servicing via automatic identification of users. This system is based on four principal steps: face detection, model-based facial feature extraction, face normalization, and face recognition by discrimination analysis. Bernardin Keni et al. [4] presented a system for audiovisual multi-person tracking and identification of persons in smart environments. Information from several fixed cameras is fused in a particle filter framework to simultaneously track multiple occupants. In [17], the authors introduced a new model for recognizing people by their gait. This model is based on motion shape, which varies with the type of moving figure and the type of motion. The identification process in this system is based on modeling the walking stride sequence, using consecutive frames from a side-view camera. Different features are calculated from the posture and limb positions of the person and from the frequency and phase presentation of walking [17]. The main drawback of the audiovisual-based approaches remains privacy issues. Indeed, these approaches are intrusive and do not respect personal privacy.

Little work has been done on non-intrusive systems which use different types of devices and sensors (motion, pressure, RFID, etc.) disseminated in the environment. In [20], the authors used a graph-based data mining system to identify inhabitants of an intelligent environment. The activity patterns for individual inhabitants are represented as graphs, which can be used to identify persons. This system is used to identify inhabitants based on observed interactions with the home. Each event in a smart home corresponds to in-

teraction with a device, such as turning the light on or off or opening or closing the door. The event is considered as a device whose state is being changed at a particular time. The authors use the SubdueCLM [20] tool to find concepts describing each inhabitant's activity pattern. These concepts can then be used to classify new activities. In this way, inhabitants can be identified according to the activity they perform. S. Zhang et al. [33] present a probabilistic learning approach to characterize behavioral patterns for multi-inhabitant smart homes. The authors propose a snowflake schema model to store the smart home activity data. Then, a supervised learning algorithm is proposed to learn the behavioral patterns of these activities. The authors aim to distinguish inhabitants and provide personalized services. However, in this study users are asked to indicate their name, the activity they plan to do and when they will begin it. This makes the recognition process less credible and sidesteps many crucial issues in inhabitant and activity recognition, such as event triggers.

Many of the aforementioned solutions suffer from computational difficulties. For instance, the audiovisual multi-person tracking approach needs very intensive computing to process the audiovisual data, in addition to the increasingly high costs of the equipment employed, such as the cameras needed for recording high-quality videos. The efficiency of the graph-based approach proposed by [20] depends basically on the size of the datasets to be processed. The larger the datasets, the more difficult and time-consuming the graph processing becomes. The use of sequential pattern mining appears to be a promising solution to overcome these drawbacks. Indeed, extracting frequent patterns from sequences can be achieved in a shorter time, as mentioned in [2, 15], which can significantly aid the development of real-time applications for identifying persons and activities.

3. OUR APPROACH

In this section we present our approach for person identification using frequent pattern mining. The overall architecture of our approach is presented in Figure 1. The different phases of our approach are detailed in the next sections.

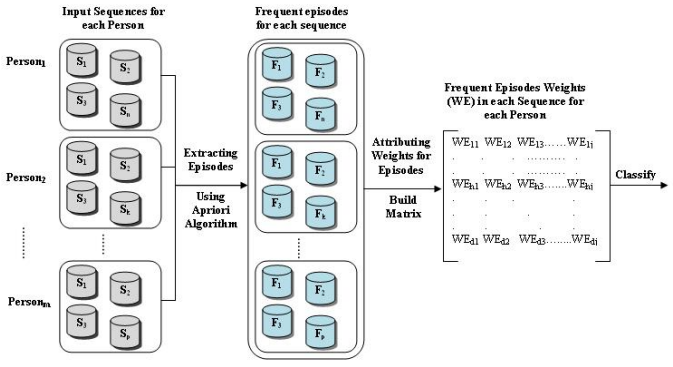


Figure 1: Our approach

3.1 Frequent Pattern Mining in Sequential Data Sets

Pattern mining is applied to sequences of various types, including protein, weblog, trace, customer purchase history and event sequences. The sequences most often studied are the event sequences generated by individuals or objects. The goal is to explain the behavior of the individuals or objects in these event sequences and determine how to deal with them. In our case, events correspond to sensor states. For each event, some additional information may be available, such as the sensor name, the sensor state/value, and temporal constraints indicating the event occurrence time. The events contained in a sequence are listed in timestamp ascending order. Table 1 shows an example of events collected in a smart home. Frequent pattern mining in sequential

Table 1: Example of events collected in a smart home

Date	Time	Sensor Name	State / value
2009-02-02	12:18:44	MotionSensor16	ON
2009-02-02	12:18:46	MotionSensor17	OFF
2009-02-02	12:28:50	DoorSensor12	OPEN
2009-02-02	12:29:55	ItemSensor03	PRESENT
2009-02-05	08:05:52	HotWaterSensor-B	0.0448835
2009-02-05	12:21:51	DoorSensor09	CLOSE
2009-02-10	17:03:57	ItemSensor03	ABSENT

datasets has been the subject of intensive research efforts in the past decade, and several algorithms have been proposed. The first was put forward by Agrawal and Srikant [1], who also developed a generalized and refined algorithm called GSP (Generalized Sequential Patterns) [23], based on the *Apriori* property [2]. Since then, several sequential pattern mining algorithms have also been proposed for performance improvements [32, 8, 27]. Before describing them, we will introduce some additional notation and some definitions of frequent pattern mining.

3.2 Problem definition

In [28], the problem of frequent pattern mining is defined as follows:

DEFINITION 1 (SEQUENTIAL PATTERN MINING). *Let $\mathcal{I} = \{i_1, i_2, ..., i_n\}$ be a set of items. A sequence $\mathcal{S}$ is an ordered list of events, denoted by $\langle e_1, e_2, ..., e_m \rangle$, where e_i is an item, that is $e_i \in \mathcal{I}$ for $1 \leq i \leq m$. A sequence can also be written as $e_1, e_2, ..., e_m$.*

An event e_i corresponds to a sensor state. For example, $e_i = MotionSensorON$, or $e_i = MotionSensorOFF$. From the definition, an item can occur multiple times in different events of a sequence. An event is associated with a timestamp which indicates the occurrence time of the event. For example, $(e_1, 10)$ means that the event e_1 occurs at time $t = 10$.

The length of a sequence is determined by the number of events composing it. A sequence of length l is called an $l-$sequence. For example, $(e_1, 10)(e_2, 20)(e_3, 30)(e_2, 40)(e_4, 50)$ is a 5-sequence.

A sequence $S_a = a_1, a_2, ..., a_n$ is contained in another sequence $S_b = b_1, b_2, ..., b_m$ if there exist integers $1 \leq i_1 \leq i_2 \leq ... \leq i_n \leq m$ such that $a_1 = b_{i1}$, $a_2 = b_{i2}$, ..., $a_n = b_{in}$. If S_a is contained in S_b, then S_a is called a **subsequence** of S_b and S_b a **supersequence** of S_a, denoted by $S_a \sqsubseteq S_b$.

An input sequence database $\mathcal{D}$ is a set of tuples $(sid, \mathcal{S})$, where sid is a sequence identifier and $\mathcal{S}$ an input sequence. The number of tuples in $\mathcal{D}$ is called the base size of $\mathcal{D}$, denoted by $|\mathcal{D}|$. A tuple $(sid, \mathcal{S})$ is said to contain a sequence $\mathcal{S}_\alpha$ if $\mathcal{S}$ is a *supersequence* of $\mathcal{S}_\alpha$.

The **absolute support** of a sequence $\mathcal{S}_\alpha$ in $\mathcal{D}$ is the number of tuples that contain S_α, denoted by $sup^{\mathcal{D}}(\mathcal{S}_\alpha)$.

Given a specified minimum support called (*min-sup*), a sequence $\mathcal{S}_\alpha$ is a **frequent** sequence on $\mathcal{D}$ if $sup^{\mathcal{D}}(\mathcal{S}_\alpha) \geq$ *min-sup*.

Sequential pattern mining is the process that allows the discovery of all patterns with a particular significance. In our case, these patterns are called episodes. An episode is a collection of events that occur relatively close to each other in a given partial order [18]. The concept of episode discovery was first introduced by H. Mannila [18]. Formally, the episode concept is defined as follows:

DEFINITION 2 (EPISODE). *An episode α is defined by a triple, $(\mathcal{V}_\alpha, \leq_\alpha, g_\alpha)$, where $\mathcal{V}_\alpha$ is a collection of nodes, $\leq_\alpha$ is a partial order on $\mathcal{V}_\alpha$ and $g_\alpha : \mathcal{V}_\alpha \rightarrow \mathcal{E}$ is a map that associates each node with an event type from a finite set of events $\mathcal{E}$.*

The episode α is parallel if the partial order relation $\leq$ is trivial (or empty). The episode α is serial if the partial order relation $\leq$ is a total order.

The measure of how often an episode occurs in an event sequence is called the episode frequency. Several methods for defining the episode frequency exist in the literature [19, 18]. An episode is considered interesting if it occurs sufficiently often in the event sequence. In our paper, the episode frequency is defined as the number of occurrences of the episode. Figure 2 shows an example of a frequent episode in an event sequence, where e_i is an event and t_i is the occurrence time of the event.

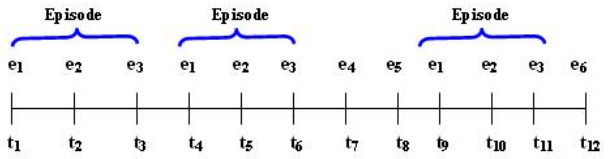


Figure 2: Frequent episode "$e_1e_2e_3$"

Our goal is to identify the person among the others living in the same environment and to distinguish him or her from the others. For this purpose, our approach uses frequent patterns as a key solution to characterize the person, first, and then to distinguish him or her from the others.

3.3 Algorithm

This section presents our proposed algorithm for person identification. Our algorithm is composed of three main phases. The first phase is episode generation. The second is the attribution of weights to the frequent episodes generated in the first phase and the construction of the frequent-episode weight matrix (**FEWM**). Finally, in the third phase, FEWM is provided to a classification algorithm.

Algorithm 1 requires the following inputs: the sequence database $\mathcal{D}$, the minimum support threshold *min-sup* defined by the user, and the length N of the episodes to be generated (for example, N=5 for episodes of length 5). The output of the algorithm is the FEWM matrix.

The steps of the algorithm 1 are described in detail below.

Step 1:

In this step, we parse all sequences in the sequence database using the **Apriori** algorithm, in order to extract frequent episodes. The extracted frequent episodes are used as input for the next step. As mentioned previously, an episode's frequency indicates how often the episode occurs in the sequence database. There are many ways to define the episode frequency. For example, in [19], the authors defined the episode frequency as the number of fixed-width sliding windows over the time where each contains an occurrence of the episode. [15] proposed a frequency measure based on non-overlapped occurrences. The standard approach used for frequent episode discovery is to use an Apriori-style level-wise procedure. Starting with frequent episodes of size 1 (events), frequent episodes of greater size are then obtained until no more frequent episodes can be found. Two steps are involved at each level: a candidate generation step and a frequency counting step.

Algorithm 1 Person identification algorithm

Inputs:
- Sequence database $\mathcal{D} = \{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_m\}$;
- Minimum support threshold ***min-sup***;
- Length **N** of episodes to be generated;

Outputs:
- **FEWM** matrix;

Steps:
1: Parse the sequence database to extract frequent episodes using **Apriori** algorithm;
2: Extract the frequencies of the generated episodes for each sequence in the database, and build the frequent episodes frequency matrix (**FEFM**);
3: Update the **FEFM** matrix by attributing weights to frequent episodes, and construct the **FEWM** matrix;
4: Return **FEWM**;
5: Classify (**FEWM**)

For simplicity in the implementation part of this step, we used the TDMiner tool[2], which is a temporal data mining tool developed in java, based on a frequent episode framework. We used the TDMiner tool to count the episode frequency, based on the Apriori-style level-wise procedure, using the fast non-overlapped count algorithm mentioned earlier. The TDMiner tool returns frequent episodes of length $1, 2, ..., N$ (N is specified by the user), with the corresponding frequencies. These results are stored in separate files.

The following example shows how episodes are extracted from an event sequence. Consider the following event sequence:

- $\{(e_1, 10), (e_2, 20), (e_1, 32), (e_3, 35), (e_1, 50), (e_2, 70)\}$

Let *min-sup* $= 2$ be the minimum support specified by the user. The episodes of length 1 that will be extracted are e_1 and e_2, given their respective frequencies, $3 \geq 2$, and $2 \geq 2$. In the same way, there are a total of four occurrences of the episode e_1e_2 of length 2. We list them here:

1. $\{(e_1, 10)(e_2, 20)\}$
2. $\{(e_1, 10)(e_2, 70)\}$
3. $\{(e_1, 32)(e_2, 70)\}$
4. $\{(e_1, 50)(e_2, 70)\}$

The remaining episodes of length 2 and those of greater size are extracted in the same way. The frequency of an episode is computed in each sequence in the database.

[2]http://neural-code.cs.vt.edu/index.html

Step 2:

This step uses the episodes extracted in step 1 to build the **FEFM** matrix. We implemented an application that parses the generated episodes, extracts their frequencies, and builds the **FEFM** matrix. The columns of this matrix are the episodes generated in step 1, and the rows correspond to the sequences in the sequence database. The greater the episode size, the better the explanation provided by the episode as to the person's behavior, and the more helpful it is in discriminating the person's behavior from that of others. Therefore, in our approach we are interested in long episodes, in order to perfectly characterize the person's behavior. In generating episodes, the Apriori algorithm takes into account the different possibilities for the order relation between events. The reason for considering all these possibilities is that the same task performed by a user in the ubiquitous environment can take several forms (episodes). For example, several sequences of events correspond to the action "add sugar to a cup of coffee". It could be (take spoon, take sugar, pour the sugar into the cup of coffee), or (take sugar, take spoon, pour the sugar into the cup of coffee). This allows us to study the different behaviors of users when performing tasks.

Step 3:

This step constitutes the core of our algorithm. The episode weight computation as such is performed in this step. After episode extraction, we have a set of episodes with their corresponding frequencies in each event sequence. Episodes associated with a high frequency are not necessarily more significant than episodes with a low frequency in the event sequence. It is thus important to identify the significance of each frequent episode extracted in step 1.

Given the variability of human behavior, the significance of an episode is based on its significance with respect to the event sequence as well as the class to which it belongs. The significance of an episode with respect to the event sequence, denoted by $SES(E,S)$, can be obtained by dividing the frequency of the episode, denoted by $F_{E,S}$, by the total sum of frequencies of all episodes contained in the event sequence. In the same way, the significance of an episode with respect to the class, denoted by $SEC(E,C)$, is obtained by summing up the $SES(E,S)$ in every event sequence. The following formulas show the $SES(E,S)$ and $SEC(E,C)$ computations.

$$SES(E,S) = \frac{F_{E,S}}{\sum_{\forall FE \in S} F_{Es,S}} \quad (1)$$

where FE denotes a frequent episode, and $F_{Es,S}$ denotes the frequency of the episode Es contained in the event sequence S.

$$SEC(E,C) = \sum_{\forall S \in C} SES(E,S) \quad (2)$$

In our case the class C corresponds to a person. Given the increased number of sensors disseminated in the environment, and the difference in the person's behavior, a frequent episode with a high SEC in one class is not necessarily more important than an episode with a low SEC in another class. Therefore, we should normalize the significance of an episode with respect to a class in order to obtain the normalized SEC in a class, denoted by $NSEC(E,C)$. We use the following formula to compute the $NSEC$, where MaxSEC(C) and MinSEC(C) correspond respectively to the maximum and minimum of the SEC of the frequent episodes in the class C. We add 1 to the numerator and denominator to avoid the case where $MaxSEC(C) - MinSEC(C) = 0$.

$$NSEC(E,C) = \frac{SEC(E,C) - MinSEC(C) + 1}{MaxSEC(C) - MinSEC(C) + 1} \quad (3)$$

Once $NSEC$ is calculated for each frequent episode, we update the frequency of each episode by adding the $NSEC$ of each episode to its frequency. The new frequency obtained is called the weight of the frequent episode, denoted by $WFE(E)$. The weight of each frequent episode is calculated by the following formula, which balances the episode frequency over the class to which the episode belongs:

$$WFE(E) = F_{E,S} + NSEC(E,C) \quad (4)$$

The pseudocode for the implementation of the new scoring function employed to attribute weights to episodes is given in Algorithm 2.

Algorithm 2 Pseudocode of the employed scoring function

Initialization: Frequent episode frequency matrix
for $i = 1, ..., N_c$ (for each class) {
  for $j = 1, ..., N_s$ (for each sequence in the class) {
    for $k = 1, ..., N_e$ (for each frequent episode) {
    - Compute the NSEC (E, C) (compute the significance of the episode with respect to the class C) using formula 3.
    - Compute the weight using formula 4.
    - Update the **FEFM** Matrix.
    - Return the **FEWM** Matrix.
    end
  end
end

The result of this step is the completed FEWM matrix, which will be provided to a classification algorithm in step 5. As noted earlier, there is only one study reported in the literature that assigns weights for frequent patterns to classify sequences [5]. However, this study employs a very simple scoring function inspired by the work of [26], that computes the weight by dividing the pattern length minus 1 by the number of patterns that describe the corresponding class. This scoring function does not take into account patterns of length 1. Moreover, it cannot determine the significant patterns in a given sequence. In contrast to this approach, our approach can be applied to all episodes, can determine the significant episodes, and gives good classification results.

An example of the content of the FEWM matrix is shown in Table 2.

Table 2: Example of the FEWM matrix

Ep 1	Ep 2	Ep 3	Ep 4	Ep 5	Ep 6	User
17.6665	21.0001	1.0004	1.0004	2.3336	0.0001	U1
10.0004	12.6665	1.2225	5.0666	3.0333	3.3333	U2
9.0001	12.1666	4.3190	5.0238	5.6654	5.0048	U3

Step 5:

In this step we apply a classification algorithm such as the decision tree procedure to the **FEWM** matrix. Note that other supervised learning algorithms can be applied at this stage to perform the classification.

4. EXPERIMENTS

In this section, we present the datasets obtained from experiments conducted at the Domus laboratory [11] and the Testbed smart home [12] to validate our approach.

4.1 Description of the Domus smart home

The Domus smart home is a one-bedroom apartment on the University of Sherbrooke campus. It includes one bedroom, one bathroom, a kitchen, a dining room, and a living room. (See Figure 3.)

In this study, we considered the sensors which are already mounted in the DOMUS apartment. Six zones are defined to cover the different apartment areas, as shown in Figure 3. The number of installed sensors varies depending on the zone of interest. The Domus smart home apartment is equipped with the following sensor categories:

- Infrared (IR) movement detectors: These cover a zone or a part of a zone; for example, in the dining room and living room there is only one IR detector that covers the entire zone, whereas three others are installed in the kitchen, covering the oven, the sink, and the toaster. These sensors provide the user's location in a zone.
- Pressure detector in the form of tactile carpeting: This sensor, placed in the entrance hall, detects the user moving between the bedroom and living room. There are two paths connecting these two zones: through the kitchen or the entrance hall.
- Light switches: These sensors send an event every time the occupant turns the lights on or off.
- Door contacts: These sensors are placed on the doors, and send an event related to the door state (open or closed).
- Closet contacts: The same as door contacts, these are placed on the cupboards and fridge. They provide an event when their state (open or closed) is changed.
- Flow meters: These provide the states of the taps and the flush toilet. Two are mounted on the cold and hot water taps of the kitchen sink, one on the washbasin cold water tap and another in the flush toilet. They send an event when the tap is opened or closed or the flush toilet is used.

4.2 Experiments Scenario

Six adults (Master's and Ph.D. students at the University of Sherbrooke) participated in these experiments, which evaluated early morning routines (grooming, breakfast), corresponding to basic tasks of everyday living. Two series of experiments were performed in the Domus smart home apartment. In the first experiments (series 1), the user was asked to perform the early morning routine as he would normally do it at home. In the second set (series 2) he was asked to repeat the same routine, with the introduction of a constraint. The constraint involved learning a tea recipe which takes at most 10 minutes. In series 1, the user came 10 times to the laboratory over two consecutive weeks. After 2 weeks' break, the user was asked to come for 5 days in one week to perform series 2. In both series, the user was free to use any equipment available in the apartment, and to determine the order of the activities composing his or her routine. Each experiment lasted about 45 minutes. Data were recorded for each user in XML format, using our system recorder framework developed in the Domus laboratory.

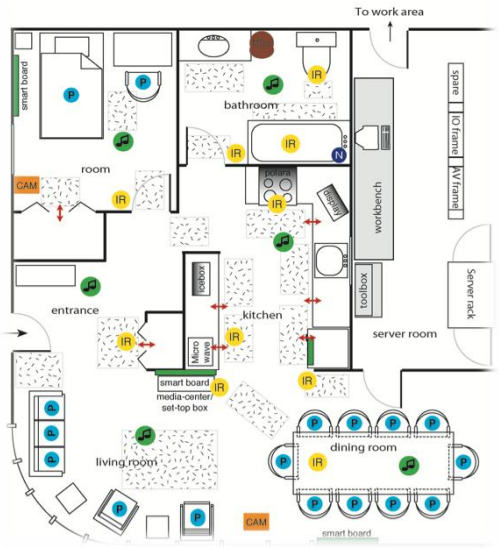


Figure 3: Domus smart home plan

5. EXPERIMENTAL RESULTS

We conducted our experiments under the Weka framework (version 3.7.0) [30]. The purpose of this experiment was to observe the general performance of our algorithm and its accuracy in practical environments. The dataset is composed of 55 sequences corresponding to series 1 (5 sequences were damaged and therefore excluded from the experiments), and 30 sequences corresponding to series 2. The length of sequences varies between 100 and 470 events for series 1 and between 210 and 680 events for series 2. The number of sensors installed in the Domus smart home apartment is 36, corresponding to 72 sensor states (ON, OFF, OPEN, CLOSED, ..., etc), in addition to one sensor (pressure detector) with a single state, for a total of 73 sensor states. We used decision trees as our classification algorithm. In fact, the decision tree procedure is one of the most widely used techniques in event sequence classification [25], and its classification time is very low. We used different episode lengths (3, 4, and 5) in order to ensure the efficiency of our algorithm and a reliable evaluation of our approach. The number of episodes used in our experiments depends on the events contained in each sequence. Table 3 shows the minimum and maximum number of episodes extracted in each dataset. In order to enrich our evaluation, we also used another dataset obtained from the Testbed smart home at Washington State University [12]. The dataset corresponds to data gathered from the Testbed smart home, where two participants performed daily living activities such as those involved in breakfast and grooming. More than 160 sensor states are involved when performing these activities. The dataset is composed of 62 sequences gathered during the breakfast activity, and 84 sequences during the grooming activity. The lengths of the sequences vary between 25 and 491 and between 36 and

Table 3: Minimum and maximum number of episodes extracted in each dataset (k = Kilo)

DataSet	Episode Length		
	N=3	N=4	N=5
Domus Series 1	[0.18 k, 16.533 k]	[0.264 k, 69.501 k]	[0.337 k, 4.646 k]
Domus Series 2	[1.647 k, 83. 874 k]	[0.04 k, 5.035 k]	[0.307 k, 26.929 k]
Testbed Breakfast	[0.04 k, 1.985 k]	[0.036 k, 9.679 k]	[0.119 k, 57.031 k]
Testbed Grooming	[0.014 k, 1.332 k]	[0.023 k, 3.034]	[0.031 k, 55.24 k]

432 events per sequence for breakfast and grooming, respectively.

Additional tests using the frequency-based (FB) approach [10] and the SubDue framework [20] were performed for comparison purposes, using the same datasets. In the FB approach, the frequency of an episode corresponds to the number of times the episode occurs in the event sequence. There is no scoring function applied to the episode frequency in the FB approach. In our experiments, we used a 10-fold cross-validation strategy to perform the classification.

5.1 SubDue framework

For the SubDue framework, we implemented a program that generates graphs from the event sequences. The resulting graphs are the input of the SubDue framework. For example, for the following event sequence composed of three sensors,

- 2009-05-06 08:40:10 S_1 ON
- 2009-05-06 08:42:43 S_2 OFF
- 2009-05-06 08:45:33 S_3 OPEN

the following graph representation is generated:

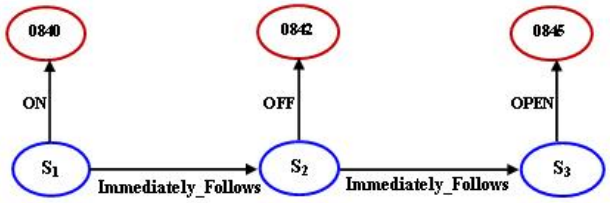


Figure 4: Example of graph representation of event sequence data

In Figure 4, the blue vertices represent sensors (S_1, S_2, S_3,..., etc.), and the red vertices represent the time when the sensor is activated (08:40, 08:42, 08:45,..., etc.). Edges between blue and red vertices represent the state of the sensor being activated, whereas edges between blue vertices represent consecutive events. More details about the graph representation in the SubDue framework can be found in [20]. We employed the one-against-all technique for training and classification, using 50% of the data for training and 50% for the test. We applied the SubDue framework on the Domus dataset for six persons, and on the Testbed dataset for two persons. The results of the classification using the three approaches on these datasets are shown in Table 4.

Figures 6(a) and 6(b) show the classification accuracy of our approach and the FB approach, when episode length is varied. Figure 5 shows the classification accuracy obtained in the SubDue framework.

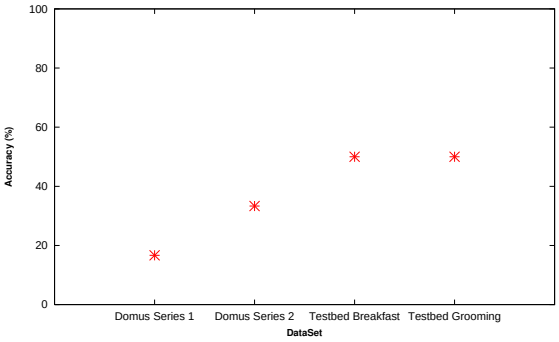


Figure 5: Classification accuracy in SubDue framework

As we can see from figures 6(a), 6(b) and 5, the identification accuracy of our approach is generally between 93 and 100, which is a high accuracy score compared to the FB approach and the SubDue framework. This is further confirmation for the effectiveness of our approach.

A comparison of the classification results obtained for each dataset using the three approaches is shown in figures 7(a), 7(b), 7(c) and 7(d). The SubDue framework performs poorly in a multi-class problem (Domus dataset with 6 persons), and relatively well in a bi-class problem (Testbed dataset with 2 persons).

From figures 7(a), 7(b), 7(c) and 7(d), we observe that episodes of greater size yield relatively higher accuracy than those of smaller size. In practice, the larger the episode, the better it reflects the task being performed, and the more understanding is provided. The results obtained in our experiments confirm this observation.

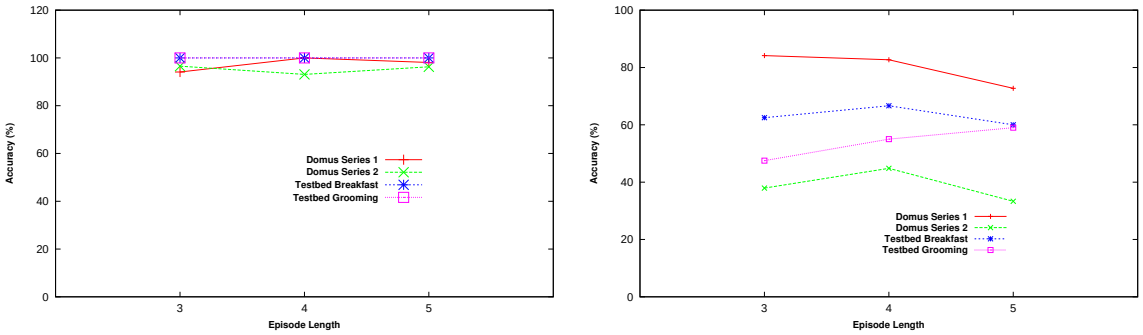
6. DISCUSSION

Our approach is aimed at identifying persons in a ubiquitous environment. Given the variability of human behavior, characterizing a person from sensor states contained in a sequence is a challenging and intricate task. By using the frequent episode principle and exploiting its inherent benefits, namely episode discovery and temporal constraints, identification can be accomplished by considering the frequent episodes extracted from event sequences. However, using frequent episodes alone, without deep analysis and study, does not necessarily guarantee good identification, as shown in figure 6(b). In this context, therefore, additional processing and analysis of frequent episodes are necessary and important.

In our approach, the frequent episodes extracted are supported by an additional piece of information: the weight of the episode. If an episode is less frequent in a given sequence and more frequent in other sequences of the same

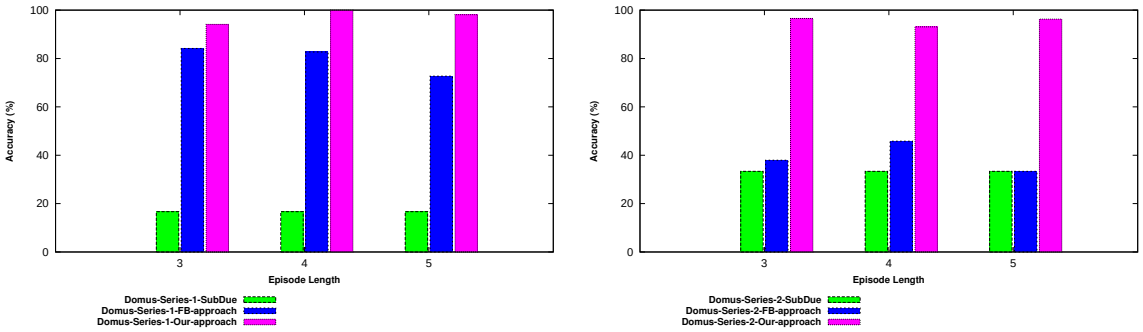
Table 4: Classification accuracy using our approach, FB approach and SubDue framework

Dataset	Accuracy (%)						
	Our approach			**Frequency Based approach**			**SubDue framework**
	Episode Length			Episode Length			
	N=3	N=4	N=5	N=3	N=4	N=5	
Domus Series 1	**94.11**	**100**	**98.11**	84.16	82.69	72.72	16.66
Domus Series 2	**96.55**	**93.10**	**96.29**	37.93	44.82	33.33	33.33
Testbed Breakfast	**100**	**100**	**100**	62.50	66.66	60.00	50.00
Testbed Grooming	**100**	**100**	**100**	47.50	55.00	59.00	50.00

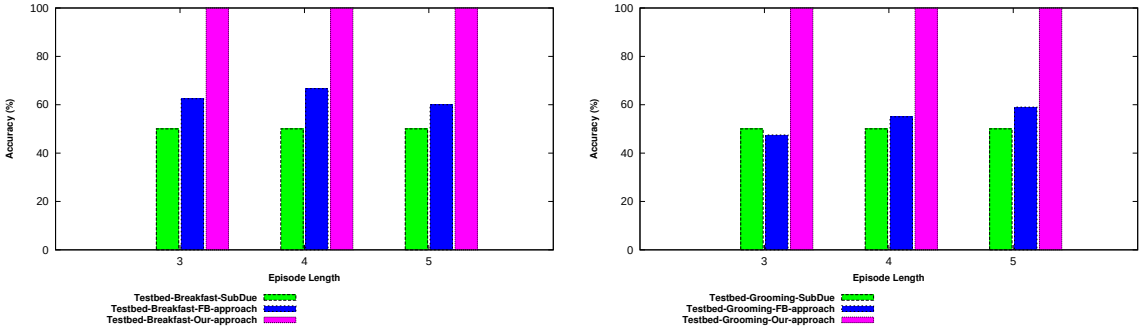


(a) Our approach (b) FB approach

Figure 6: Classification accuracy in our approach and FB approach



(a) Domus Series 1 (b) Domus Series 2



(c) Testbed Breakfast (d) Testbed Grooming

Figure 7: Comparison of the classification accuracy in our approach, FB approach and SubDue framework

class, the assignment of weights allows us to balance the episode frequency by increasing the lowest frequency, making this episode more important. The more important the episode, the higher its weight, and the more discriminative it will be. As shown in Figure 6(a), the assignment of weights significantly improves the classification results.

The other benefit of weighting is the detection of interesting (significant) episodes. By significant episodes, we mean those that can best characterize the person and have the most discriminative power. This direction, which was not exploited in the present study, is the subject of our ongoing work. Indeed, the assignment of weights to episodes poses two challenges: detecting significant episodes, and reducing the dimensionality of the space using only these significant episodes. The development of real-time applications with a low dimensionality space thus constitutes a major contribution to the ubiquitous environment and KDD communities.

Our approach has broad social, economic and academic impacts. It can be used in ubiquitous environments such as smart homes and smart hospitals to characterize the inhabitants and adapt a variety of tasks accordingly. Our approach can also be used to characterize persons and products in supermarkets and study the purchase trends for each product. Finally, our approach contributes to the literature by introducing new algorithms and methods that can be used by the scientific community for teaching and/or research purposes.

7. CONCLUSION

In this paper, we presented a new efficient algorithm for person identification in ubiquitous environments, based on frequent pattern mining. Specifically, we analyzed data gathered from two ubiquitous environments: the Domus and Testbed smart homes. In order to identify persons, we first extracted frequent episodes from the datasets. Next, we assigned weights to these episodes. Finally, we applied a classification algorithm to classify the frequent episodes. We concluded that episode weighting is more appropriate than the episode frequency-based approach, and achieves high classification accuracy.

We illustrated the effectiveness and suitability of our approach on multiple datasets extracted from two smart homes, Domus and Testbed. The experimental results show that our approach is able to identify persons with a significantly higher accuracy than the frequency-based approach or the SubDue framework. Moreover, we tested our approach on different episode lengths, and observed that episodes of greater size yield relatively better accuracy than those of smaller size. In addition, our algorithm is based on sequential pattern mining, which can be used in various applications involving large amounts of sequential data, such as protein classification, temporal sequences, and financial sequences. We used our approach to detect significant episodes in the event sequences. Indeed, the weights assigned to frequent episodes are used to distinguish significant episodes from non-significant ones. Episodes with greater weights are more significant than those with lower weights. Therefore, a person's behavior can be characterized using significant episodes that can distinguish him or her from others.

8. KNOWLEDGEMENT

The authors wish to thank Virginie Charette, a professor at the Department of Mathematics of the University of Sherbrooke for her help.

9. REFERENCES

[1] R. Agrawal and R. Srikant. Mining sequential patterns. In *proceedings of the International Conference on Data Engineering,*, pages 3–14, 1995.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, 1994.

[3] M. Kellner B. Ivanov, H. Ruser. Presence detection and person identification in smart homes. In *proceedings of the International Conference of Sensors and Systems*, pages 80–85, 2002.

[4] Keni Bernardin and Rainer Stiefelhagen. Audio-visual multi-person tracking and identification for smart environments. In *Proceedings of the 15th international conference on Multimedia*, pages 661–670, 2007.

[5] Themis P. Exarchos, Markos G. Tsipouras, Costas Papaloukas, and Dimitrios I. Fotiadis. A two-stage methodology for sequence classification based on sequential pattern mining and optimization. *Data Knowl. Eng.*, 66(3):467–487, 2008.

[6] Peter H. N. de With Fei Zuo. Real-time face recognition for smart home applications. In *proceedings of the International Conference on Consumer Electronics*, pages 183–190, 2005.

[7] Y Guang-Zhong. *Body Sensor Networks*. Springer-Verlag, London, 2006.

[8] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Freespan: frequent pattern-projected sequential pattern mining. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 355–359, 2000.

[9] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Min. Knowl. Discov.*, 8(1):53–87, 2004.

[10] Edwin O. Heierman and Diane J. Cook. Improving home automation by discovering regularly occurring device usage patterns. In *Proceedings of the Third IEEE International Conference on Data Mining*, page 537, 2003.

[11] Bauchet J., Giroux S., Pigot H., Lussier-Desrochers, and D. Lachapelle. Pervasive assistance for people with intellectual disabilities in smart homes : A case study on meal preparation. *International Journal of Assistive Robotics and Mechatronics*, 9(4), 2008.

[12] Cook D J and Schmitter-Edgecombe M. Assessing the quality of activities in a smart environment. *Methods of Information in Medicine*, 48(5), 2009.

[13] A.K. Jain, Lin Hong, S. Pankanti, and R. Bolle. An identity-authentication system using fingerprints. *Proceedings of the IEEE*, 85(9):1365–1388, Sep 1997.

[14] Anil Jain, Lin Hong, and Sharath Pankanti. Biometric identification. *Commun. ACM*, 43(2):90–98, 2000.

[15] Srivatsan Laxman, P. S. Sastry, and K. P. Unnikrishnan. A fast algorithm for finding frequent episodes in event streams. In *Proceedings of the KDD*, pages 410–419, 2007.

[16] Chang-Rong Lin, Ning-Han Liu, Yi-Hung Wu, and Arbee L. P. Chen. Music classification using significant repeating patterns. In *Proceedings of DASFAA*, pages 506–518, 2004.

[17] James J. Little and Jeffrey E. Boyd. Recognizing people by their gait: The shape of motion. *Videre: Journal of Computer Vision Research*, 1:1–33, 1998.

[18] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovering frequent episodes in sequences. In *KDD*, pages 210–215, 1995.

[19] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Min. Knowl. Discov.*, 1(3):259–289, 1997.

[20] Ritesh Mehta, Diane J. Cook, and Lawrence B. Holder. Identifying inhabitants of an intelligent environment using a graph-based data mining system. In *Proceedings od FLAIRS Conference*, pages 314–318, 2003.

[21] Mika Paajanen, Jukka Lekkala, and Kari Kirjavainen. Electromechanical film (emfi) – a new multipurpose electret material. *Sensors and Actuators A: Physical*, 84(1-2):95 – 102, 2000.

[22] Parisa Rashidi and Diane J. Cook. An adaptive sensor mining framework for pervasive computing applications. In *Proceedings of the International Workshop on Knowledge Discovery from Sensor Data(Sensor-KDD)*, pages 41–49, 2008.

[23] Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In Peter M. G. Apers, Mokrane Bouzeghoub, and Georges Gardarin, editors, *Proceedings of the 5th Int. Conf. Extending Database Technology*, volume 1057, pages 3–17. Springer-Verlag, 1996.

[24] Jaakko Suutala and Juha Röning. Methods for person identification on a pressure-sensitive floor: Experiments with multiple classifiers and reject option. *Information Fusion*, 9(1):21–40, 2008.

[25] Vincent S. Tseng and Chao-Hui Lee. Effective temporal data classification by integrating sequential pattern mining and probabilistic induction. *Expert Syst. Appl.*, 36(5):9524–9532, 2009.

[26] Vincent Shin-Mu Tseng and Chao-Hui Lee. Cbs: A new classification method by using sequential patterns. In *Proceedings of SDM*, 2005.

[27] J. Wang and J. Han. Bide: efficient mining of frequent closed sequences. In *Proceedings of the 20th International Conference on Data Engineering*, pages 79–90, 2004.

[28] Jianyong Wang, Jiawei Han, and Chun Li. Frequent closed sequence mining without candidate maintenance. *IEEE Trans. Knowl. Data Eng.*, 19(8):1042–1056, 2007.

[29] Mark Weiser. The computer for the 21st century. *SIGMOBILE Mobile Computing and Communications Review*, 3(3):3–11, 1999.

[30] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementation*. Morgan Kaufmann, 2000.

[31] Christopher Richard Wren, Ali Azarbayejani, Trevor Darrell, and Alex Paul Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

[32] Mohammed J. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1/2):31–60, 2001.

[33] Shuai Zhang, Sally I. McClean, Bryan W. Scotney, Xin Hong, Chris D. Nugent, and Maurice D. Mulvenna. Decision support for alzheimer's patients in smart homes. In *proceedings of the IEEE international Symposium on Computer-Based Medical Systems*, pages 236–241, 2008.

Activity Recognition using Actigraph Sensor

Raghavendiran Srinivasan
Washington State University
Pullman, WA 99164
USA
rsriniva@eecs.wsu.edu

Chao Chen
Washington State University
Pullman, WA 99164
USA
cchen@eecs.wsu.edu

Diane J. Cook
Washington State University
Pullman, WA 99164
USA
cook@eecs.wsu.edu

ABSTRACT
Numerous accelerometers are being extensively used in the recognition of simple ambulatory activities. Using wearable sensors for activity recognition is the latest topic of interest in smart home research. We use an Actigraph watch with an embedded accelerometer sensor to recognize real-life activities done in a home. Real-life activities include the set of Activities of Daily Living (ADL). ADLs are the crucial activities we perform everyday in our homes. Actigraph watches have been profusely used in sleep studies to determine the sleep/wake cycles and also the quality of sleep. In this paper, we investigate the possibility of using Actigraph watches to recognize activities. The data collected from an Actigraph watch was analyzed to predict ADLs (Activities of Daily Living). We apply machine learning algorithms to the Actigraph data to predict the ADLs. Also, a comparative study of activity prediction accuracy obtained from four machine learning algorithms is discussed.

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

General Terms
Algorithms, Performance, Experimentation, Human Factors.

Keywords
ADL, Activity, Actigraph, Wearable Sensor, Smart Home.

1. INTRODUCTION
As the world's population ages [1], those suffering from diseases associated with dementia will increase. Smart homes can assist their residents by acting as a cognitive prosthesis, by handling various appliances/objects and also by facilitating emergency communication. Being able to automate activity recognition from human motion patterns using unobtrusive wearable sensors can be useful in monitoring older adults in their homes and keeping track of their Activities of Daily Living (ADL) and behavioral changes [2]. This could lead to a better understanding of numerous medical conditions and treatments. Wireless sensors provide continuous monitoring capability that traditional methodology lacks. Also in addition, the assessments performed in the clinical setting may not give the actual representation of the patient's behavior. Real life assessments can provide a better understanding of the patient than assessments performed in a clinical setting. Health care effectiveness in several situations has improved significantly using current wireless communication technologies. Traditionally, the sensors for these instruments are attached to the patient by wires. This sequentially makes the patient become bed-bound. In addition, whenever a patient needs to be moved, all monitoring devices have to be disconnected and then reconnected later. In current technological advancements, all of these time-consuming tasks are terminated and patients could be liberated from instrumentation and bed by wireless technology. The future aims at the integration of the existing smart home sensors with pervasive, wireless networks. They will co-exist with the installed infrastructure, augmenting data collection and ultimately providing real-time responses. Also, integrated wireless devices could communicate with a gateway that connects to the medical center's network and transmits data to health data stores for monitoring, control, or evaluating in real time.

Unobtrusive wearable sensors will allow vast amounts of data to be collected and mined for next-generation clinical trials. By using wearable sensors, data will be collected and reported automatically, reducing the cost and inconvenience of regular visits to the physician. With the growing technological advancements, we aim to provide telecare which allows elderly people to remain living in their own home. Furthermore, researchers are also aiming at integrating the knowledge gained from wearable (wireless) sensors with the wired smart home sensors. This will give a big picture of the physical behavior of the person. Also, by integrating this knowledge with information collected by object sensors we can get a picture of the person's physical interaction with the surroundings. By collecting this data for a longer duration, we can predict physical interactions and provide necessary prompting when the person forgets to do an activity. Also, wearable sensors could be used to provide accurate measures of motor abilities in the home and community settings. Their use could tremendously facilitate the implementation of tele-rehabilitation protocols. The wearable devices must also be

SensorKDD'10, July 25, 2010, Washington, DC, USA.

lightweight enough to be worn without much inconvenience. With demographic changes of the aging population and an increasing number of people living alone, smart homes are set to play an important role in maintaining the independence and improving the quality of life for elderly persons at a lower cost.

The present development of the demography of elderly people in the U.S as well as the other parts of the world will generate a shortage of caretakers for elderly people in the near future. The new concept of health monitoring is advanced by which health parameters are automatically monitored at home without disturbing daily activities. The recording of sensor data over extended periods of time is necessary to design and implement an efficient activity recognition algorithm.

In Section 2, we analyze previous works done in the field of wireless wearable sensors and activity recognition. Section 3 describes the Actigraph watch which was used in our experimentation. Section 4 explains the process of feature extraction from the datasets. Section 5 gives a brief description of the machine learning algorithms we considered for testing our dataset. In Section 6, we present the results of our experiments and compare the accuracy and time performance of activity prediction by different algorithms.

2. RELATED WORKS

Actigraph watches are used as effective data collection tools in the study and clinical assessment of sleep disorders [3]. Apart from Actigraph, various other accelerometer-based sensors have been used to detect activity behaviors. The eWatch system [4] proposes a wearable sensor and computing platform for context aware research for activity recognition and a related health monitoring system. Assessing sleep disruption is the most significant contribution that wrist Actigraphy has made. Previously, Actigraph sensors have been used primarily for sleep studies. The Actigraph device has also been used in finding circadian activity rhythms in healthy individuals as well as people with primary and co-morbid insomnia. Explicit focus on identifying which location the wearable has to be placed was concentrated by many researchers. By placing wearable sensors at different locations, researchers have tried to evaluate the accuracy of activity recognition [5][6]. Also, predicting the activity based on the user-annotated data has been carried using wearable Biaxial accelerometers [7]. Further, the prediction of complex activities performed inside a woodshop was recognized by attaching wearable accelerometers and a microphone on the human body [8].

Today most wearable systems are based on conventional notebook architectures integrated into some sort of belt or a device tied around the body that will make the participant to be easily spotted among the common public. Embedding sensors into garments is an idea that was first pursued by a research team at Georgia Tech. Research work by this team eventually led to a wearable system referred to as the *Smart Shirt* [9]. This approach appears to be ideal for very long-term monitoring (i.e. months to years) of individuals in the home and community settings. Alarmnet [10] being developed at University of Virginia is also targeting remote healthcare monitoring [11]. Predicting activities using tri-axial accelerometers is one of the most commonly used techniques. To identify activities using tri-axial accelerometers is done by formulating the activity recognition as a classification problem [12]. This approach also compares the performance of base-level and meta-level classifiers. Also, predicting the activity level from the energy expenditure is an extension of the application of accelerometers. The activity levels have been predicted using the energy expenditure data collected from tri-axial accelerometers [13].

Recognizing activities with wrist-worn watches is the latest area of focus due to its portability and ease of use. Further these devices do not draw unwanted attention to the people who wear them. Wearable health monitoring systems integrated into a telemedicine system represent novel information technology that will be able to support early detection of abnormal conditions and prevention of its serious consequences.

3. ACTIGRAPH SENSOR DESCRIPTION

Actigraphs are wristwatch-like devices with an accelerometer inside. They are small, portable devices that detect physical motion, generate an internal signal each time they are moved, and store that information. Typically, it is used for measuring general or random motor restlessness in order to evaluate the rest-activity cycle. Actigraphy is a method in which the user places accelerometers on different parts of the body to measure physical acceleration at each location and to estimate the level of human activity. Actigraphy is a relatively non-invasive method of monitoring human rest/activity cycles. Actigraphy has been shown to best estimate sleep duration. In sleep research applications the periods of low activity are considered as sleep. Actigraphs are generally worn on the wrist of the non-dominant arm. They may be worn for weeks at a time. The Actigraph watch used in our experiments is the Motionlogger Actigraph. This sensor can collect data for 21 days. Once the data is collected, it is downloaded to a computer. Actigraph devices are very practical, because of their low power consumption and the small size of the electronic components. The accelerometer records movement that will indicate when someone is active and quiet. It should be worn for 24-hours regardless of the activity. It is waterproof and can be worn in the shower, hot tub or when washing dishes. It can also be worn while swimming which involves partial submersion.

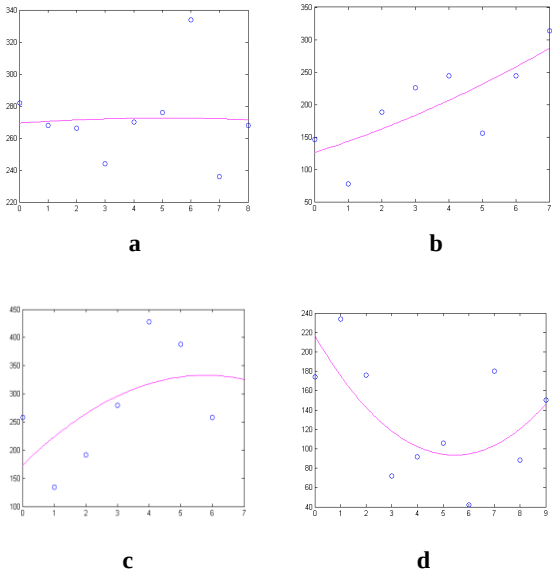
Figure 1. Actigraph wrist watch sensor.

The Motionlogger Actigraph monitor shown in Figure 1, contains a piezoelectric bimorph-cantilevered beam, which generates a voltage for each movement made. The voltage generated is passed on to the Analog circuitry, second essential element of the Motionlogger circuit. Here, the signal received is modulated and filtered through a 2-3 Hz bandpass filter. This conditioned analog signal can be processed in two different ways based on the mode of operation i.e., either Zero Crossing (ZC) Mode or the Proportional Integral Mode (PIM). Zero Crossing

mode logs the number of times that the acceleration value exceeds a threshold amount, while PIM mode keeps track of more basic movement values. The derived information obtained based on the mode of operation is accumulated over the epoch and is stored in the memory of the device. Our Motionlogger Actigraph utilizes an epoch which is of 1 minute duration. The Actigraph watch should be set in the ZC or PIM mode well before the participant wears it in the wrist. The mode once set cannot be changed till the data collection stops. After setting the sensor in one of the two modes of operation, the time we want to start the data collection can also be defined with the help of the Actigraph software [14]. After selecting the start time for the watch, it is given to the participant. The sensor will be collecting data for next 21 days from the prescribed time of start. When the memory gets full, the data collection stops. The Actigraph watch is connected to a data collection system and the data stored in the memory of the device is extracted using the Actigraph software. The memory of the device should be refreshed for further usage. The Actigraph software also has the capability to generate the sleep and wake cycles just from the raw data. The software was hard-coded to predict sleep/wake cycles to a better extent and reduce the false positives generated by the software. The data finally got from the software is used for manual annotation.

4. FEATURE EXTRACTION

We plotted Actigraph values for various activities used in our experiments on a time scale to observe the patterns that are generated by the different activities. In order to generate these graphs, we apply curve fitting to Actigraph data using least squares method [15]. With curve fitting, we try to find the best fit to a series of data points. It can serve as an aid for data visualization, to approximate the values when no data are available, and to express the relationships between different data points. The line generated will be the pattern that corresponds to the particular ADL.



a **b**

c **d**

Figure 2. Curve fitting of ADLs for ZC values. (X-axis: time in minutes; Y-axis: ZC value; a. Cooking; b. Eating; c. Hygiene; d. Watching Movie)

From Figure 2, we can observe that each ADL follows a specific time-varying pattern. As a result, we hypothesize that we can use selected machine learning algorithms to recognize each pattern. The raw dataset was not given as input to the machine learning algorithms. The raw data extracted from the Actigraph watch contains only numerical information of the participant's motion in the form of ZC values. This value cannot be easily used by themselves in their raw form to predict activities. We have two more parameters in the dataset which we can integrate with the ZC value to extract features. The three parameters in the raw dataset are thus: Date, Time and ZC value. In order to obtain even more information from the existing dataset we consider that fact that the clock time of an activity can vary each time it is being performed in a real-life scenario. Thus, we extract additional specific information such as the amount of time spent for an activity, which hour of a day was the activity performed, what were the previous & next activities, and so forth. We extract features [16] from the raw dataset to get the processed data. The features extracted from the dataset were:

Min ZC value – The minimum zero-crossing value of an activity. This value is calculated for every instant of the activity.

Max ZC value – The maximum zero-crossing value of an activity for every instant of the activity.

Sleep status – The sleep/awake status of the participant. This value is generated through the Actigraph software after extracting the raw data.

Time length – The amount of time taken for the completion of an activity. This value is calculated for every instant of the activity.

Begin hour – The time of day is split into a 24 parts with each part denoting an hour. The begin hour of every activity is calculated for each occurrence of the activity.

Number of events – The total number of actigraph events per activity calculated for every instance.

Bin – We discretized the raw ZC value data into five interval sizes by equal width binning [17].

Total ZC value – The total ZC value obtained for each activity for every instant.

Pre-Activity –We note every activity's previous activity.

Post-Activity – The following activity of every activity is taken as the post activity.

The processed dataset with the value of all the extracted features was used for our testing. By using the data mining and machine learning concepts, we can perform pattern recognition to predict the ADLs.

5. PREDICTION ALGORITHMS

Machine learning algorithms have been used exclusively to learn and recognize complex patterns and classify objects based on sensor data. Hence, we use these techniques to identify the appropriate ADLs. The following parts give a brief description of four machine learning algorithms used in our experiments.

5.1 BayesNet

Bayesian belief networks are based on the work of the mathematician and theologian Thomas Bayes. Bayesian belief networks [18] were introduced by Judea Pearl in 1985. BayesNet

belongs to the family of probabilistic graphical models which represent a set of conditional independence assumptions by a directed acyclic graph. Each of the variables in the Bayesian belief network is represented by a node in the graph and the edges represent direct dependence among the variables. Bayesian belief networks offer consistent semantics for representing uncertainty and an intuitive graphical representation of the interactions between various causes and effects, thus they are a very effective method of predicting uncertain situations that depend on cause and effect. ADLs of an individual have a great extent of uncertainty. Hence, we use the BayesNet for predicting them.

5.2 Artificial Neural Network

Artificial neural networks (ANNs) [19] are composed of interconnecting artificial neurons. They are abstract computational models based on the organizational structure of the human brain. ANNs provide a general and robust method to learn a target function from input examples. The multilayer perceptron (MLP), a type of ANN, is a feedforward artificial neural network that maps the input to appropriate output. In our experiments we choose, MLP because it is one of the commonly used algorithms for any supervised-learning pattern recognition process.

5.3 Sequential Minimal Optimization

In a traditional Support Vector Machine (SVM), the quadratic programming problem involves a matrix, whose elements are equal to the number of training examples. If the training set is large, the SVM algorithm will use a lot of memory. To solve such a problem, Sequential Minimal Optimization (SMO) [20] decomposes the overall quadratic programming problem into a series of smaller quadratic programming problems. During the training process, SMO picks a pair of Lagrange multipliers for every iteration and solves each small quadratic programming problem, then repeats the same process until it converges to a solution to the overall quadratic programming problem. SMO significantly improves scaling of training set size and computation time for SVMs.

5.4 LogitBoost Ensemble

Boosting is another most commonly used supervised learning algorithm. Boosting algorithms have become very successful in machine learning. The idea of boosting is to combine many "weak" classifiers to create a "strong" classifier. LogitBoost [21] combines the aspect of AdaBoost (Adaptive Boosting) and Logistic regression. Considering AdaBoost as a generalized additive model, if we apply the cost functionalities of Logistic Regression then we arrive at the LogitBoost algorithm.

6. EXPERIMENTS & RESULTS

In our experiment, we have used the Actigraph watch in the ZC mode. Previously the ZC mode has been used for detecting sleep studies. Here we explore its use for activity recognition. The watch was worn for 21 days by one participant. The participant noted down the activities of only 17 of the days. So, we considered only those days of the data. The ADLs performed by the participant was noted down manually. The participant performed a well-defined set of activities regularly. After the data collection stops, the data was extracted from the Actigraph watch and the dataset was manually annotated. The participant performed the following set of activities for 17 continuous days: Hygiene, Cooking, WebCam Chat on Laptop, Working on Laptop, Watching movies on Laptop, Sleeping and Eating. Recognition results are calculated using 3-fold cross-validation to generalize the results. We analyzed recognition results using four Machine Learning algorithms: 1) LogitBoost, 2) BayesNet, 3) NeuralNet, and 4) SMO. We use graphs to analyze the performance of the algorithms and also discuss the overall accuracy obtained in recognizing all the ADLs using the four algorithms. We also discuss the accuracy obtained for individual ADLs using the chosen machine learning techniques. Further, we plot the accuracy of an algorithm with its execution time to find the algorithm which runs in optimal time when compared to others. All the three plots are used to support our conclusion as to which algorithm outperforms the other three.

From Figure 3, we observe that LogitBoost performs better than the other three algorithms. This is because of the fact that the LogicBoost classifier estimates the accuracy by combining the performance of many “weak” classifiers. We also find that the next best algorithm is SMO and the accuracy rate is the same for NeuralNet and BayesNet. We cannot conclude that both of the algorithms have poorer accuracy than the others. This makes it very difficult to judge which one is better than other on the basis of accuracy performance alone.

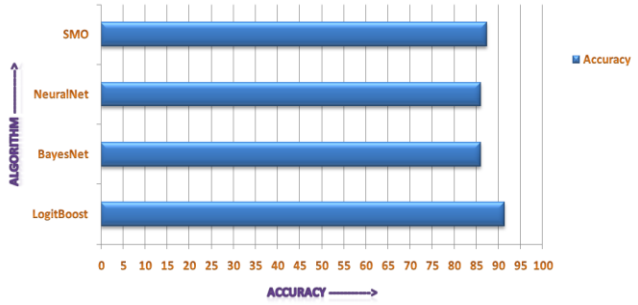


Figure 3. Overall Accuracy predicted by different algorithms.

Figure 4 shows the accuracy of each individual activity along the lines of every classifier used. SMO is not considered to be the best classifier for this task because it could not adequately differentiate three activities Watching Movie, Using WebCam chat and Working on Laptop. The activities Watching Movie and Using WebCam chat were also classified under working in Laptop. This prevented the algorithm from becoming the overall best algorithm. Also, we find that all the algorithms were able to recognize three activities more efficiently than rest: 1) Cooking, 2) Sleeping, and 3) Eating. The activity Working on Laptop had a few instances misclassified either as Using WebCam chat or as Watching Movie in Laptop. This led to the lowered performance of the Working on Laptop. The ADL which was poorly recognized by all the algorithms is Watching Movie on a Laptop. This may be due to varied reasons like the lowered activity level when the participant watches a slow movie and the raised activity level when the participant watches a thriller or horror movie.

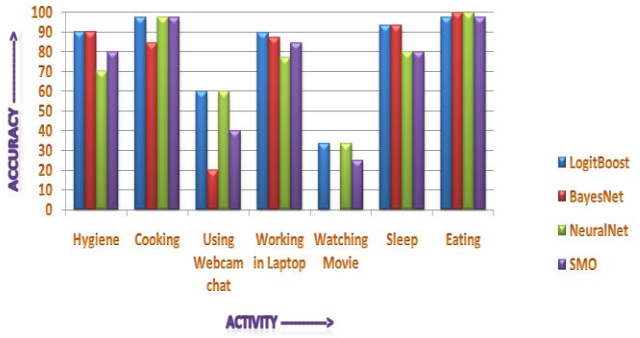


Figure 4. Individual Accuracy of ADLs predicted by different algorithms.

In Figure 5, we observe that the BayesNet performs the fastest in terms of the running time of the algorithm. BayesNet was not the overall best because it was not able to predict even one instance of the activity of WatchingMovie. The activity of watching movie didn't have a noticable uncertainty in its behavior and also the algorithm was not able to differentiate between the activities Watching Movie and Using WebCam Chat.

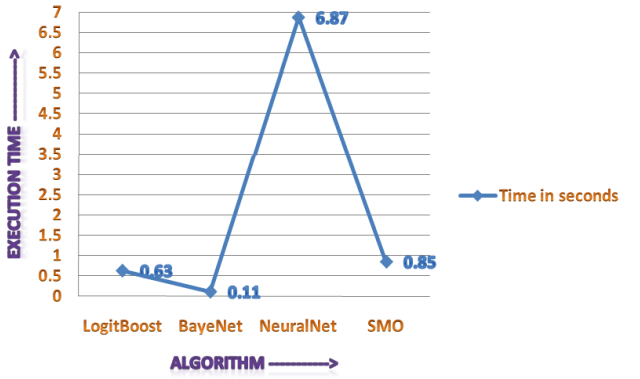


Figure 5. Execution Time of different algorithms.

Also, NeuralNet is not ranked as our overall best algorithm as it failed to meet the requirement of smaller execution time. The left out algorithms are LogitBoost and SMO. However, SMO takes a running time of 0.23 seconds more than LogitBoost. In this criterion also, LogitBoost outperforms all other algorithms. Hence LoogitBoost becomes the overall efficient algorithm for our activity prediction.

7. FUTURE WORK & CONCLUSION

Constant monitoring using wearable sensors will allow patients to engage in normal ADLs, rather than relying upon specialized medical services outside their homes. From the viewpoint of the industrial sector, the addition of constant monitored data into the medical databases will allow an integrated analysis of all data to optimize individualized care and provide knowledge discovery through various data mining approaches. Typically, accelerometers are used to consider only simple ambulatory movements like sitting, standing and walking. In this paper we have considered complex ambulatory movements. The ambulatory movements considered are functional Activities of Daily Living (ADL). We were able to achieve an accuracy of 91.39% using the LogitBoost algorithm. Previous works had concentrated on using multiple sensors in different body locations to recognize activities. The main advantage over the previous works is we have used only one sensor placed in one location of the body (non-dominant wrist). We will try to use few more sensors like pulse rate detectors and energy sensors to predict activities. Continuous monitoring has the capability of pro diagnosis of disease at early stage of occurrence and it likely has the potential to provide patients with an increased level of confidence, which in turn may improve quality of life We are also planning to integrate wearable sensors with wired sensors used in the CASAS Smart home to predict activities. In the CASAS Smart Home, we also aim at using object sensor in a smart home along with the wearable sensors to predict the physical behavior of a person in that environment of living and try to use it to predict the behavior of the same person in another environment.

REFERENCES

[1] http://www.census.gov/population/www/projections/

[2] Cook, D.J., 2006. *Health monitoring and assistance to support aging in place.* Journal of Universal Computer Science. **12**(1): p. 15–29.

[3] de Souza, L., et al., 2003. *Further validation of actigraphy for sleep studies.* Sleep. **26**(1): p. 81–5.

[4] Maurer, U., et al., 2006. *eWatch: a wearable sensor and notification platform*, in *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on*. p. 4.

[5] Maurer, U., et al., 2006. *Activity recognition and monitoring using multiple sensors on different body positions*, in *Wearable and Implantable Body Sensor Networks. BSN 2006. International Workshop on*. p. 4.

[6] Lee, S.W. and K. Mase, 2002. *Activity and location recognition using wearable sensors.* IEEE Pervasive Computing: p. 24–32.

[7] L. Bao and S. S. Intille. 2004. *Activity Recognition from User-Annotated Acceleration Data*. In A. Ferscha and F. Mattern, editors, *Pervasive*, volume 3001 of *Lecture Notes in Computer Science*, pages 1–17. Springer.

[8] Lukowicz, P., et al., 2004. *Recognizing workshop activity using body worn microphones and accelerometers.* Pervasive Computing: p. 18–32.

[9] Park, S. and S. Jayaraman, 2003. *Enhancing the quality of life through wearable technology.* IEEE Engineering in medicine and biology magazine. **22**(3): p. 41–48.

[10] Wood, A., et al., 2006. *ALARM-NET: Wireless sensor networks for assisted-living and residential monitoring.* University of Virginia Computer Science Department Technical Report.

[11] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L. Littman, 2005. "*Activity recognition from accelerometer data*.," in Proceedings, The

Seventeenth Innovative Applications of Artificial Intelligence Conference. pp. 1541–1546.

[12] X. Long, B. Yin, and R.M. Aarts, 2009. *"Single-Accelerometer-Based Daily Physical Activity Classification*". *31st Annual International IEEE EMBS Conference*, Minneapolis, MN.

[13] Istepanian, R.S.H., E. Jovanov, and Y.T. Zhang, 2004. *Guest editorial introduction to the special section on M-health: beyond seamless mobility and global wireless health-care connectivity*. IEEE Transactions on Information Technology in Biomedicine. **8**(4): p. 405–414.

[14] Jean-Louis, G., et al., 1997. *The actigraph data analysis software: I. A novel approach to scoring and interpreting sleep-wake activity*. Perceptual and motor skills. **85**(1): p. 207.

[15] Sage, A.P. and G.W. Masters, 1967. *Least-Squares Curve Fitting and Discrete Optimum Fitting*. IEEE Transactions on Education. **10**(1): p. 29–36.

[16] Pirttikangas, S., K. Fujinami, and T. Nakajima, 2006. *Feature selection and activity recognition from wearable sensors.* Ubiquitous Computing Systems: p. 516–527.

[17] Liu, H., et al., 2002. Discretization: An enabling technique. Data Mining and Knowledge Discovery. 6(4): p. 393–423.

[18] Uusitalo, L., 2007. *Advantages and challenges of Bayesian networks in environmental modelling*. Ecological modelling. **203**(3-4): p. 312–318.

[19] Zornetzer, S.F., 1995. *An introduction to neural and electronic networks*: Morgan Kaufmann.

[20] Platt, J., 1998. *Sequential minimal optimization: A fast algorithm for training support vector machines.*.

[21] Schapire, R.E., 1990. *The strength of weak learnability*. Machine learning. **5**(2): p. 197–227.

Network Comprehension by Clustering Streaming Sensors

Pedro Pereira Rodrigues
LIAAD - INESC Porto L.A.
SBIM-FMUP & DCC-FCUP
University of Porto, Portugal
pprodrigues@fc.up.pt

João Gama
LIAAD - INESC Porto L.A.
Faculty of Economics
University of Porto, Portugal
jgama@fep.up.pt

João Araújo
DCC - Faculty of Sciences
University of Porto, Portugal
joao.araujo@dcc.fc.up.pt

Luís Lopes
CRACS - INESC Porto L.A.
DCC - Faculty of Sciences
University of Porto, Portugal
lblopes@dcc.fc.up.pt

ABSTRACT

Sensor network comprehension tries to extract information about global interaction between sensors by looking at the data they produce. When no other information is available to extract, usual knowledge discovery approaches are based on unsupervised techniques. However, if these techniques require data to be gathered centrally, communication and storage requirements are often unbounded. The goal of this paper is to discuss sensor network comprehension techniques, presenting a local algorithm to compute clustering of sensors at each node, using only neighbors' centroids, as an approximation of the global clustering of streaming sensors computed by a centralized process. The clustering algorithm is based on the moving average of each node's data over time: the moving average of each node is approximated using memoryless fading average; clustering is based on the furthest point algorithm applied to the centroids computed by the node's direct neighbors. The algorithm was evaluated on a state-of-the-art sensor network simulator, measuring the agreement bewteen local and global clustering. Results show a high level of agreement between each node's clustering definitions and the global clustering definition, with special emphasis on separability agreement. Overall, local approaches are able to keep a good approximation of the global clustering, improving the ability to keep global network comprehension at each sensor node, with increased privacy, and decreased communication and computation load in the network.

Categories and Subject Descriptors

I.5.3 [**Pattern Recognition**]: Clustering—*Algorithms*; C.2.4 [**Distributed Systems**]: Distributed Applications

SensorKDD'10, July 25, 2010, Washington, DC, USA.

General Terms

Algorithms

1. RATIONALE

Sensor networks can include a variable number of small sensors, with dynamic network topologies and evolvable concepts producing data. In real-world applications, data flows at huge rates, with information being usually forwarded throughout the network into a common sink node, being afterwards available for analysis [7]. However, common applications usually inspect behaviors of single sensors, looking for threshold-breaking values or failures. To increase the ability to understand the inner dynamics of the entire network, deeper knowledge should be extracted.

Knowledge discovery is a wide area of research where machine learning, data mining and data warehousing techniques converge to the common goal of describing and understanding the world. Sensor network comprehension tries to extract information about global interaction between sensors by looking at the data they produce [32]. When no other information is available, usual knowledge discovery approaches are based on unsupervised techniques (e.g. clustering). However, two different clustering problems exist: *clustering data streams* and *clustering streaming sensors*. This paper addresses clustering of streaming sensors, but the former problem is nonetheless relevant to discuss in terms of sensor network comprehension.

In this work we explore different characteristics of sensor networks which define new requirements for knowledge discovery, with the common goal of extracting some kind of comprehension about sensor data and sensor networks. This network comprehension ability is related with both sensor data clustering and clustering of streaming sensors. A wide range of techniques already exists to assess these interactions in centralized scenarios, but the seizable processing abilities of sensors in distributed algorithms present several benefits that shall be considered in future designs [31].

Clustering is probably the most frequently used data mining algorithm [19], used in exploratory data analysis. It consists on the process of partitioning data into groups, where elements in the same group are more similar than elements in different groups. The main problem in applying clustering to data streams is that systems should consider data evolution, being able to compress old information and adapt

to new concepts. The range of clustering algorithms that operate online over data streams is wide, including partitional [6, 27] or hierarchical [1, 41], density-based [14] and grid-based methods [30, 36]. A common connecting feature is the definition of unit cells or representative points, from which clustering can be obtained with less computational costs [1].

Clustering data streams is the task of clustering data flowing from a continuous stream, based on data points similarity [1], aiming to discover structures in data over time [4]. Algorithms usually search for dense regions of the data space, identifying hot-spots where streaming data sources tend to produce data [32]. For example, in a sensor network, sensor 1 is at high values when sensor 2 is in mid-range values, and this happens more often than any other combination. Clustering streaming sensors is the task of clustering different sources of data streams, based on the data series similarity [33]. Algorithms aim to find groups of sensors that behave similarly through time. For example, in the same sensor network, sensors 1 and 2 are highly correlated in the sense that when one's values are increasing the other's are also increasing. This is highly related with clustering time series. Although a lot of research has been done on clustering subsequences of time series (which raised some controversy in the data mining community [21,24]), clustering streaming data sources approaches whole clustering instead, so most of the existent techniques can be successfully applied, but only if incremental versions are possible.

From the previous two procedures additional knowledge can be exploited. Consider mobile sensor networks where each sensor produces a stream with its current GPS location. Clustering the examples would give an indication of usual dispersion patterns, while clustering the sensors could give indication of physical binding between sensors, forcing them to move with similar paths. Another application could rise from temperature/pressure sensors placed around geographical sites such as volcanoes or seismic faults. Furthermore, the evolution of these clustering definitions is also relevant. If each sensor's stream consists of IDs of the sensors for which this sensor is forwarding messages, changes in the clustering structure would indicate changes in the physical topology of the network, as dynamic routing strategies are commonly encountered in current sensor network applications. Overall, the main goal of sensor network comprehension is to apply automatic unsupervised procedures in order to discover interactions between sensors, trying to exploit dynamism and robustness of the network being deployed in the objective site.

2. CLUSTERING FOR COMPREHENSION

Clustering streaming data from sensors has been far more addressed in literature than clustering streaming sensors. This way, we shall just point out some approaches to the former task, giving more emphasis to the later. In the following, we address scenarios where each sensor produces one stream of data, being afterwards combined in such way with the remaining network streams to achieve a global clustering definition. This process tries to extract knowledge about the similarity between data series produced by different sensors.

2.1 Comprehension by Clustering Data

This section of sensor network comprehension tries to extract knowledge in order to define dense regions of the sensor data space. Clustering streaming examples is widely spread in the data mining community as a technique used to discover structures in data over time [1]. This task also requires high-speed processing of examples and compact representation of clusters, yielding adaptivity issues. In this topic we study the problem of continuously maintain a cluster structure over the data points generated by the entire network. A wide range of techniques already exists to assess this characteristic in centralized scenarios, but distributed algorithms seem more adaptable and reliable as data and processing is distributed across sensors in the network. Usual techniques operate by forwarding and concentrating the entire data in a central server, processing it as a multivariate stream. The seizable processing abilities of sensors present several benefits that must be considered in the design of clustering algorithms. If data streams are being produced separately (each variable in each sensor) in distributed sites, and although each site should process its own univariate stream locally before any clustering procedure, a coordinator site must execute some kind of processing (actually it should execute the clustering procedure) on the whole gathered data, possibly feeding the remote sites with the current clustering model.

Since current applications generate many pervasive distributed computing environments, data mining systems must nowadays be designed to work not as a monolithic centralized application but as a distributed collaborative process. The centralization of information yields problems not only with resources such as communication and memory, but also with privacy of sensitive information. Instead of centralizing relevant data in a single server and afterwards perform the data mining operations, the entire process should be distributed and, therefore, paralleled throughout the entire network of processing units. A good example of collaborative work that can be done to achieve clustering on sensor networks was developed by Kargupta et. al. , who presented a collective principal component analysis, and its application to distributed cluster analysis [23]. Other approaches include Klusch et. al. proposal, a kernel density based clustering method over homogeneous distributed data [25] and a distributed majority vote algorithm which can be seen as a primitive to monitor a k-means clustering over peer-to-peer networks [11]. These techniques present a good feature as they perform only a few rounds of data transmission through the network. Cormode et al. [9] proposed different strategies to achieve the same goal, with local and global computations, in order to balance the communication costs. They considered techniques based on the *furthest point* algorithm [18], which gives an approximation for the radius and diameter of clusters with guaranteed cost of two times the cost of the optimal clustering. They also present the *parallel guessing* strategy, which gives a slightly worse approximation but requires only a single pass over the data. They conclude that, in actual distributed settings, it is frequently preferable to track each site locally and combine the results at the coordinator site. Moreover, the recent development of global frameworks that are capable of mining data on distributed sources is rising. Taking into account the lack of resources usually encountered on sensor networks, Resource-Aware Clustering [15] was proposed as a stream clustering algorithm that can adapt to the changing availability of different resources. The system is integrated in a generic framework that enables resource-awareness in streaming computation, monitoring main resources like memory, battery and

CPU usage, in order to achieve scalability to distributed sensor networks, by means of adaptation of algorithm's parameters. Data arrival rate, sampling and number of clusters are examples of parameters that are controlled by this monitoring process.

Clustering examples in sensor networks can be used to search for hot-spots where sensors tend to produce data. In this settings, grid-based clustering represents a major asset as regions can be, strictly or loosely, defined by both the user and the adaptive process. The application of clustering to grid cells enhances the abstraction of cells as interval regions which are better interpreted by humans. Moreover, comparing intervals or grids is usually easier than comparing exact points, as an external scale is not required: intervals have intrinsic scaling. For example, when querying for the top hot-spot of a given sensor network, instead of achieving results such as "usually, sensor 1 is around 100.2 when sensor 2 is around 10.5", we would get "usually, sensor 1 is between 95 and 105 when sensor 2 is within 10.4 and 10.6". The comprehension of how sensors are interacting in the network is greatly improved by using grid-based clustering techniques for the data examples produced by sensors. This strategy was used in DGClust, a recent approach which combines both grid-based approximations and point-based clustering [30].

2.2 Comprehension by Clustering Sensors

Sensor network comprehension is also highly related with the interaction between sensors, in terms of similar behaviors or readings. Clustering streaming sensors is the task of clustering streaming data series produced by sensors on a wide sensor network. This process tries to extract knowledge about the similarity between data produced by different sensors. Most works on clustering analysis for sensor networks actually concentrate on clustering the sensors by their geographical position [7] and connectivity, mainly for power management [39] and network routing purposes [20]. However, in this topic, we are interested in clustering techniques for data produced by the sensors, instead. Considering the dynamic behavior usually enclosed in streaming data, clustering streaming sensors should be addressed as an online and incremental procedure, in order to enable faster adaptation to new concepts and produce better models through time. However, centralized clustering strategies tend to be inapplicable as usual techniques have quadratic complexity on the number of sensors, and sensor networks grow unbounded [31].

Clustering streaming data sources is an emerging area of research which has been already studied in various fields of real world applications [5, 10, 33]. However, algorithms previously proposed tend to deal with data as a centralized multivariate stream [32]. They are designed as a single process of analysis, without taking into account the locality of data produced by sources on a distributed scenario, the transmission and processing resources of the network, and the breach in the transmitted data quality. In fact, many motivating domains could benefit from (and some of them even require) a distributed approach, given their objective application or specialized setting [31]. Most works on clustering analysis for distributed sources (e.g. sensor networks) actually concentrate on clustering the sources by their geographical position [7] and connectivity, mainly for power management [39] and network routing purposes [20]. However, in this topic, we are interested in clustering techniques using the data produced by the sources, instead. Many algorithms have been developed for distributed clustering in peer-to-peer environments and sensor networks settings. However, they do not target our problem, as they might operate with a central server [9, 23], are directed towards data clustering (and not data sources) [11,15], address homogeneous distributed clustering [3,25] (each node has a sample of the same data), or if they target clustering of streaming data sources, they take into account the network infrastructure in the process of finding a clustering definition [38].

The basic requirements usually presented for clustering data streams are that the system must possess a compact representation of clusters, must process data in a fast and incremental way and should clearly identify changes in the clustering structure [4]. Nevertheless, there are some conceptual differences when addressing multiple streaming sources. This way, systems that aim to cluster streaming data sources should [31]: process with constant update time and memory, enable an anytime compact representation, include techniques for structural drift detection, enable the incorporation of new relevant sources, and operate with adaptable configuration.

This section focus on clustering of the streams produced by sensors, which tries to extract knowledge about the similarity between data series produced by different sensors. This task relates with sensor network comprehension as clustering sensors finds groups of sensors that behave similarly through time. The distributed setup proposed in this section enables a transient user to query a local node for its position in the overall clustering structure of sensors, without asking the centralized server. For example, a query for a given sensor could be answered by "this sensor and sensors 2 and 3 are highly correlated", in the sense that when one's values are increasing the others' are also increasing, or "the answering sensor is included in a group of sensors that has the following profile or prototype of behavior". Hence, the comprehension of how sensors are related in the network is also greatly improved by using distributed sensor clustering techniques.

The aim of a distributed clustering algorithm should be to be able to answer queries for the global clustering definition of the entire system. If data sources are distributed, with local sites being accessible from transient devices, queries could be issued at each local site, enabling fast answers to be sent to the querying device. However, the setup of forwarding data to a central server, forces not only the data but also the queries to be transmitted across the network into a sink. Several issues emerge in the development of new techniques to efficiently and effectively perform clustering of distributed streaming data sources. In previous work [31], the requirements of clustering distributed streaming data sources have been discussed and enumerated: a) the requirements for clustering streaming data sources must be considered, with more emphasis on the adaptability of the whole system; b) processing must be distributed and synchronized on local neighborhoods or querying nodes; c) The main focus should be on finding similar data sources irrespectively to their physical location; d) processes should minimize different resources (mainly energy) consumption in order to achieve high uptime; and e) operation should consider a compact representation of both the data and the generated models, enabling fast and efficient transmission

and access from mobile and embedded devices. The final goal is to infer a global clustering structure of all relevant data sources. Hence, approximate algorithms should be considered to prevent global data transmission.

Advantages of distributed procedures for clustering streaming data sources are multidimensional, having different impacts on specific domains of applications. For example, preventing dimensionality burden in clustering electrical load profiles [29], or having the hability to operate on ad-hoc sensor networks created for natural phenomena monitoring [35]. Also embedding GPS mobile devices with context information [40] or identifying similar signals in patients [34] while keeping sensitive data decentralized, and in sensor network management. This last topic appears as extremely relevant as distributed clustering of streaming sensors enables the improvement of sensor deployment, might reduce message forwarding, preserves privacy of observed data, and improves network comprehension, in the sense that each sensor is able to tell where in the sensor data domain it is located [32].

3. LOCAL-TO-GLOBAL CLUSTERING

The basic idea behind clustering streaming data sources is to find groups of sources that behave similarly through time, which is usually measured in terms of the distance between the data series or the data distribution. Let X be a sensor node producing observations x_i at each timestep i. The goal of an incremental clustering system for streaming data sources is to find (and make available at any time i) a partition $C(i)$ of d sources, where data sources in the same cluster tend to be more alike than data sources in different clusters [28].

A local algorithm is proposed to perform clustering of sensors on ubiquitous sensor networks, based on the moving average of each node's data over time. There are two main characteristics. On one hand, each sensor node keeps a sketch of its own data. On the other hand, communication is limited to direct neighbors, so clustering is computed at each node. The moving average of each node is approximated using memoryless fading average, while clustering is based on the furthest point algorithm applied to the centroids computed by the node's direct neighbors. This way, each sensor acts as data stream source but also as a processing node, keeping a sketch of its own data, and a definition of the clustering structure of the entire network of data sources. In this work we search for a definition of k clusters of sensor nodes, with k previously known by the system. Although this simple example lacks some of the common characteristics of real-world scenarios (e.g. unknown number or clusters or unbalanced data), its extension is straightforward. If the number of clusters to find is unknown, each node could search for a clustering with different number of clusters. As only centroids are transmitted, and used as single points (as if operating with emsembles of clusters), there's no need to know how many points come from each node; all centroids that are received are included in the clustering as single points. For unbalanced data (in terms of the assignment of nodes to clusters) we believe that the convergence would take longer, but deeper analysis is required in future work.

3.1 Fading-Average Sketching of Sensor Data

We want to define a clustering structure for the sensors, where sensors producing streams which are alike are clustered together. Overall, we should consider techniques that project each sensor's data stream into a reduced set of dimensions which suffice to extract similarity with other sensors. These estimates can be seen as the sensor's current overview of its own data, giving an indication of where in the data-space this sensor is included [32]. One way to summarize a data stream x is by computing its sample mean $\hat{\mu}_x$ and standard deviation $\hat{\sigma}_x$. More complex strategies could include distribution distances based on the histograms of each sensor's data (e.g. relative entropy [26]), where each sensor would have to transmit the frequency of each data interval to its neighbors, or using approximations of the original data [5]. Our approach is to keep track of the moving average of each sensor, as the sample mean of most recent data. We assume the dissimilarity between two sensors to be the absolute distance between their sample means. Although in several real-world scenarios this is not true, we should assume the sample mean of each sensor to be uncorrelated with its physical location and connectivity (see Fig. 2), as the matching between data clusters and physical clusters is a promising strategy for sensor network comprehension, so we should not bias the clustering solution.

Each sensor produces data continuously. Given this, each sensor s is responsible of keeping its own estimate of the sample mean ($\hat{\mu}_s$) in an online fashion. Moving averages are usually easy to compute, if we can keep a small buffer of data points [16]. However, in such resource-demanding scenarios, this is seldom the case. Sum-based statistics computed on sliding windows can be approximated by weighting the sums using fading factors [17]. The *fading sum* $S_{x,\alpha}(i)$ of observations from a stream x is computed at time i, as:

$$S_\alpha(i) = x_i + \alpha \times S_\alpha(i-1)$$

where $S_\alpha(1) = x_1$ and α ($0 \ll \alpha < 1$) is a constant determining the forgetting factor of the sum. Each value of α, which should be close to 1 (e.g. 0.999), will converge to sliding windows of different sizes. This way, the *fading average* at observation i is then computed as:

$$M_\alpha(i) = \frac{S_\alpha(i)}{N_\alpha(i)}$$

where $N_\alpha(i) = 1 + \alpha \times N_\alpha(i-1)$ is the corresponding *fading increment*, with $N_\alpha(1) = 1$. An important feature of the fading increment is that:

$$\lim_{i \to +\infty} N_{\alpha<1}(i) = \frac{1}{1-\alpha}$$

This way, at each observation i, $N_\alpha(i)$ gives an approximated value for the weight given to recent observations used in the fading sum. Due to space restrictions, we do not present the entire theoretical proof for fading averages, but Figure 1 presents an illustrative comparison between moving averages and fading averages for a data stream with concept drift, empirically showing the applicability of such approximations. Moreover, the fading statistics are memoryless, an important property in streaming scenarios.

Given the exposed benefits of using memoryless summaries, we propose to use the *fading average* $M_{x,\alpha}(i)$ as sketching structure for each sensor node x. This way, each sensor is responsible to keep a unique value: the fading average computed so far. Sensors produce readings asynchronously, so sketch update needs to be triggered with the arrival of a new data point, irrespectively of the time elapsed since the pre-

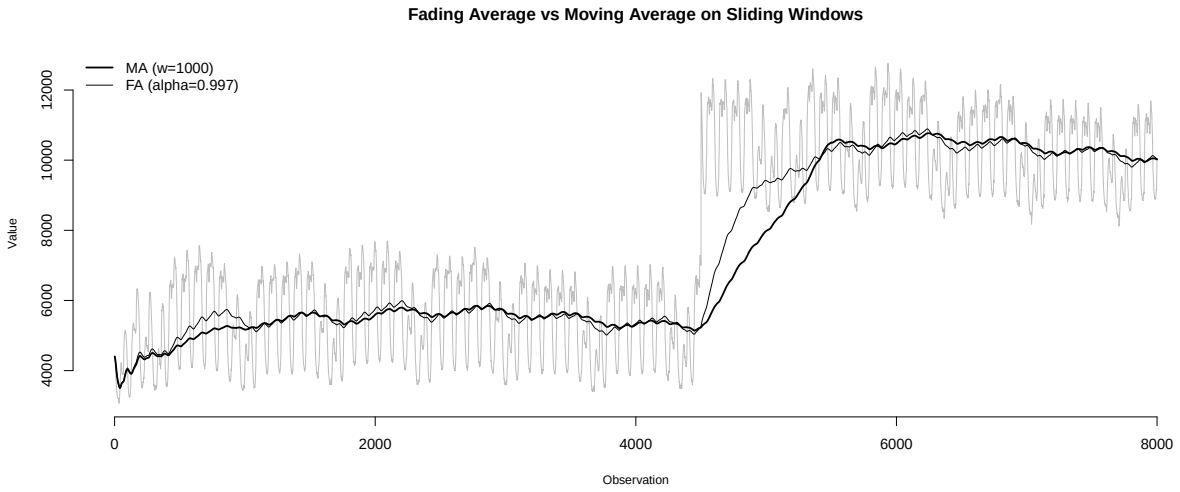


Figure 1: Comparison of the evolution of the moving average (thick black line, window size $w = 1000$) and the fading average (thin black line, forgetting factor $\alpha = 0.997$) for a drifting data stream (thin grey line).

vious point. Future developments should take time into account, if more complex sketches are to be computed. At each new observation x_i, the node performs a simple update of its sketch with $M_{x,\alpha}(i)$ being an approximation of the mean of the most recent observations of x, i.e. $\hat{\mu}_x = M_{x,\alpha}(i)$.

3.2 Approximations of the Global Clustering

The goal is to have at each local site a global clustering structure of the entire sensor network. Each sensor should include incremental clustering techniques which operate with distance metrics developed for the dimensionally-reduced sketches of the data streams. Given the simple sketch definition, the distance between two sensors x and y can be defined as $d(x, y) = |\hat{\mu}_x - \hat{\mu}_y|$. As each sensor x is able to sketch its own data in a dimensionally-reduced definition (the fading average $M_{x,\alpha}$), it is also able to interact with its neighboring nodes η_x, in order to assess a local clustering of sensors. The main characteristic of our approach is that, at each new observation i produced by sensor x, instead of sendind its own sketch $M_{x,\alpha}$ to its neighbors η_x, the node sends its own estimate of the global clustering $C_x(i)$, i.e. the centroids gathered by clustering its neighbors' centroids. Note that with this approach, each sensor keeps an approximate estimate of the global cluster centers. This estimate can be seen as the sensor's current view of the entire network which, together with its own sketch, gives an indication of where in the entire network data-space this sensor is included.

At first, each sensor node x has only access to its own sketch $M_{x,\alpha}(i)$. While producing data, node x also receives estimates of the global cluster centers computed by their neighbors η_x. Let $P_x(i)$ be the complete set of clustering definitions $\{C_j(i) \mid j \in \eta_x\}$ received by node x before observation x_i. The idea behind this step is to aggregate all the locally defined centers and apply a clustering procedure on these centers, considering them as points for the clustering. Therefore, $C_x(i)$ is computed by clustering the set of points $\{M_{x,\alpha}(i)\} \cup C_x(i-1) \cup P_x(i)$. This way, next time this sensor uses or transmits its estimate $C_x(i)$ of the global clustering structure, it is already updated with its most recent sketch and neighbors' information. In the general task of finding k centers given m points, there are two major objectives: minimize the *radius* (maximum distance between a point and its closest cluster center) or minimize the *diameter* (maximum distance between two points assigned to the same cluster) [9]. The *Furthest Point* clustering algorithm [18] is applied to the set of points $\{M_{x,\alpha}(i)\} \cup C_x(i-1) \cup P_x(i)$, which gives a guaranteed 2-approximation for both the *radius* and *diameter* measures. It begins by picking an arbitrary point as the first center, c_1, then finding the remaining centers c_i iteratively as the point that maximizes its distance from the previously chosen centers $\{c_1, ..., c_{i-1}\}$. After k iterations, one can show that the chosen centers $\{c_1, c_2, ..., c_k\}$ represent a factor 2 approximation to the optimal clustering [18]. See [9] for a proof. This strategy gives a good initialization of the cluster centers, computed by finding the center k_i of each cluster after attracting remaining points to the closest center c_i. Since we are applying clustering to cluster centroids, we are in fact merging clustering definitions, a known technique which as been argued to give good results [9].

To prevent clustering using unstable fading averages, and to prevent excessive communication, nodes only send their estimate C_x to direct neighbors from time to time. Specifically, in our setup, nodes only perform clustering and transmission after $\frac{1}{NC(1-\alpha)}$, where NC is the number of clustering processes executed within the time frame of a recent fading window (e.g. for $\alpha = .999$, the considered recent fading window is $w = 1000$; for $NC = 10$, clustering is performed every 100 examples).

4. EVALUATION STRATEGY

The global aim of this work is to assess the feasibility of computing local approximations of the global clustering structure of a ubiquitous network of streaming data sources. A first evaluation was designed, simulating sensor networks, in order to validate that possibility. UC Berkeley's project Ptolemy [13] produced an open-source, Java-based, software framework called Ptolemy II, with tools for the

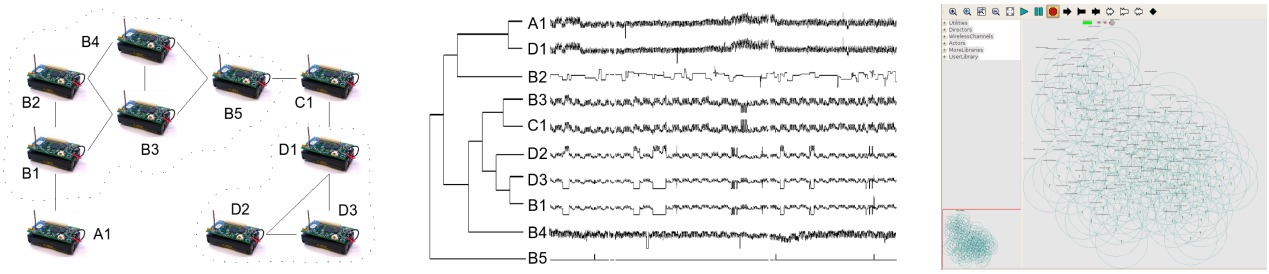


Figure 2: Example of a mote sensor network (left), with links of possible transmission represented by straight lines and physical subnetworks (represented by IDs) separated by dashed lines, and a possible clustering definition of the series produced by each sensor (center plot). This illustrative example shows the orthogonality that is expected to exist between network topology and the sensors' data clustering structure. Right plot presents a VisualSense simulation environment with 128 sensors; circles denote the range of transmission, hence defining links between nodes.

modeling, simulation and design of concurrent, real-time, embedded systems. The main underlying software abstraction is the actor, software components that execute concurrently and communicate through messages sent via interconnected ports. Application specific models can then be represented as hierarchical interconnections of actors coordinated by special components called directors. VisualSense [2] was developed under this common framework specifically to allow the modeling of wireless sensor networks. In particular, each sensor can be implemented as one or more actors that communicate with each other. The tool allows very sophisticated modeling of features like communication channels, hardware sensor devices, networking protocols, MAC protocols and energy consumption in sensor nodes. Right plot of Figure 2 presents a screen shot of a simulation for one network with 128 sensors. All experiments in this paper were implemented using the VisualSense sensor network simulation environment.

Although in real world data is seldomly random or normally distributed, a first validity check was designed with a synthetic sample of such data. In this experiment we follow the scenario design used in [12], where data is generated in the unit hypercube. Each scenario was produced according to three parameters: the number of clusters K, the number of sensor nodes D, and the standard deviation σ used to generate random data. The cluster centers μ_k are generated one at a time, by sampling uniformly from the $[2\sigma, 1-2\sigma]$ interval. This ensures that most data points lie within the unit hypercube. Any μ_k that was less than σ/K away from a previously generated center was rejected and regenerated to avoid too close centers which are unlikely to be separated by clustering procedures. Each sensor x produces a stream X of $100K$ random data points with distribution $X \sim N(\mu_x, \sigma)$. To determine the μ_x parameter, each sensor is uniformly assigned to one of the clusters, let's say μ_{c_x}. Each sensor's mean μ_x is then randomly sampled from $N(\mu_{c_x}, \sigma)$. Each data scenario is applied to several network configurations. These differ on network size and network deployment. Each network is generated by a cascade procedure within the Visual Sense 1000x1000 pixel square: first, a random point is selected for the first sensor node; then, each sensor node is placed at a time in the network geographical space by uniformly sampling a previous sensor node and randomly choosing a point within the predefined range of that chosen sensor node (in our experiments, $range = 300$). Sensors which are placed closer than 150 from any previously place node are relocated. For each network size, 3 different configurations are generated. To determine the sensitivity of our approach to the random effects produced by the evaluation setting, the analysis is done in three dimensions: network size $d = \{8, 16, 32, 64, 128\}$, number of clusters $K = \{2, 3, 4, 5, 6, 7\}$, standard deviation $\sigma = \{0.01, 0.05, 0.1\}$. For each data scenario, network configuration, and parameters choice, 10 different experiments were run. Fading averages are computed with $\alpha = .999$, representing a fading window of 1000 examples, while NC was fixed to 10.

Our goal is to assess the feasibility of computing local approximations of the global clustering structure. This way, we will compare the clustering definitions of each sensor node C_x with the global clustering definition C_g that a centralized server would compute having access to all data being generated in the network. Since the focus of analysis is the locality of computations, the global clustering C_g is computed exactly as the local clusterings C_x, except for the fact that C_g uses all the sensors sketches $M_{x,\alpha}$ directly in the clustering step. When dealing with clustering algorithms, several validity measures exist that can fulfill any researcher's desire. See [19] for a comprehensive study on this topic. More than computing a loss function or a validity index for each sensor's clustering definition, the goal of our work is to achieve a local clustering definition at each sensor that would *agree* with the global clustering definition, if queried to assign each pair of sensors in the network to the same or to different cluster centers. Several external validity indices are based on the agreement of clustering assignments (e.g. the Jaccard coefficient [22]), but they are biased towards a strict comparison. In this work, we propose to use the agreement theory directly, as different agreement proportions give different insights of clustering comparisons. To compute agreement between clustering definitions C_x and C_g, we need to compute four quantities: n_{xg}, number of sensor pairs clustered together by both C_x and C_g; $n_{x\bar{g}}$, number of sensor pairs clustered together by C_x but not by C_g; $n_{\bar{x}g}$, number of sensor pairs clustered together by C_g but not by C_x; $n_{\bar{x}\bar{g}}$, number of sensor pairs clustered separatedly by both C_x and C_g. Note that $n_{xg} + n_{x\bar{g}} + n_{\bar{x}g} + n_{\bar{x}\bar{g}} = N = d(d-1)/2$, where d is the number of sensors in the network.

The statistic $\hat{\kappa} = (P(A) - P(e)) / (1 - P(e))$ gives a simple **sanity** check, where $P(A) = (n_{xg} + n_{\bar{x}\bar{g}}) / N$ is the ob-

served proportion of agreement, and $P(e)$ is the expected proportion of agreement that would be observed if clusterings agreed only by chance [8][1]. For values higher than zero, C_x and C_g agree more than just by chance ($\hat{\kappa} = 1$ represents total agreement). However, comparing clustering definitions according to improvement from agreeing only by chance is rather poor and is only used as sanity check. More interesting than $\hat{\kappa}$ are the agreement proportions. The *global agreement proportion* $P(A)$ gives a clear assessment of the agreement of C_x and C_g, hence serving as **validity** index for C_x. The goal is to have $P(A) = 1$. However, when total agreement is not achieved, there are two directions where clustering definitions might agree: *positive* and *negative* agreement. In our problem, a test is considered positive when the pair of sensors are assigned to the same cluster, and negative otherwise. Positive agreement is then defined as $P(A^+) = 2n_{xg}/(2n_{xg} + n_{x\bar{g}} + n_{\bar{x}g})$, being interpreted as the conditional probability of agreement considering that one of the clustering definitions has already stated that the pair of sensors should be clustered together. From our point of view, this clearly relates to the proportion of agreement on **compactness** of the clustering structure, focusing on the pair of points that both clustering definitions state should be together. On the other hand, negative agreement is defined as $P(A^-) = 2n_{\bar{x}\bar{g}}/(2n_{\bar{x}\bar{g}} + n_{x\bar{g}} + n_{\bar{x}g})$, being interpreted as the conditional probability of agreement considering that one of the clustering definitions has already stated that the pair of sensors should be separated. From our point of view, this clearly relates to the proportion of agreement on **separability** of the clustering structure, focusing on the pair of points that both clustering definitions state should be separated.

Each network's quality is assessed using the mean (over all sensors) of the indices: $\hat{\kappa}$ statistic, a *better-than-chance sanity check*; $P(A)$, the global proportion of agreement, interpreted as *validity*; $P(A^+)$, the positive proportion of agreement, as *compactness*; and $P(A^-)$, the negative proportion of agreement, as *separability*. The percentage of sensors with $P(A) = 1$ is also evaluated (interpreted as *perfectness*).

5. RESULTS

Each experimental run results in a learning curve for each index. Upper plot of Fig. 3 presents the evolution of the average $P(A)$ index (over all sensors), for one network with $\{k = 7, \sigma = .01, d = 128\}$. Following the strategy proposed in [17], the curve should be smoothed by applying the *fading average* to the computed value (bottom plot of Fig. 3). The main point to stress is that the network converges, besides some small oscillation due to randomness of data being produced. Since for each data scenario (k, σ and d) 10 runs are executed for each of the 3 configurations of the sensor network, results are presented as mean values (and the corresponding 95% confidence interval) over the 30 runs. Given the convergence empirical observation (different scenarios converge at different number of node interactions), we compare the means (over all runs) for the fading average of each measure at the last time point of each run. Table 1 presents the complete set of results.

The first result to extract is that all scenarios passed the sanity check, as $\hat{\kappa}$ statistic is always positive (in fact, higher than .58). After confirming the sanity of our approach, we can stress that a high level of average agreement ($P(A) > 90\%$) is achieved for most of the scenarios. Moreover, for the theoretically hardest scenario presented here ($\{k = 7, \sigma = .10, d = 128\}$) the lower limit of the confidence interval is $P(A) = 0.86$, meaning that, on average, a sensor node would agree with the global clustering on over 86% of the total pairs of sensors. Regarding the direction of agreement, we should stress that our approach clearly gives more relevance to separability. In all scenarios where the average proportion of positive agreement (compactness) is different (under the 95% confidence level) from the average proportion of negative agreement (separability), the separability index is always higher than the compactness index. Moreover, even for scenarios with lower average proportion of agreement, separability tends to stay high, meaning that in our approach, each single node will have a high level of agreement on answer queries for which pairs of nodes should be clustered separately than on queries for which pairs of nodes should be clustered together. Finally, it became clear that, at least for harder scenarios, it is very difficult to achieve networks where all nodes agree with central clustering on 100% of pairs of nodes. From our observation, this might result from different node connectivity (nodes placed on the outer ribbon of the network could have more difficult to converge to global clustering). Nevertheless, results put the agreement level at a high level, stating that local clustering gives a good approximation of the global clustering. Due to space restrictions, we postpone the presentation of a more detailed analysis on sensitivity to network configuration. On the other hand, we can argue that agreement levels are ro-

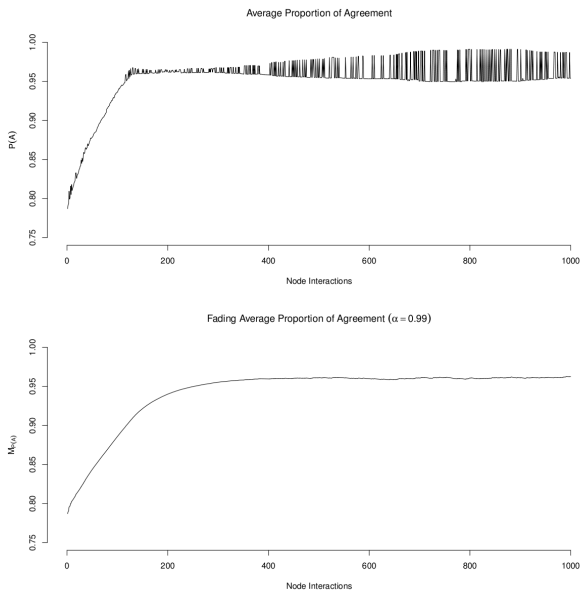


Figure 3: Evolution of the average proportion of agreement between sensors and the global clustering definition (one experimental run with $k = 7$, $d = 128$, $\sigma = .01$). Top plot presents the actual average proportion of agreement (over all sensors). Bottom plot presents the fading average (over time) of the average proportion of agreement.

[1]This statistic has been shown to be equivalent to the Adjusted Rand Index [37].

Table 1: Clustering validity results, in terms of $\hat{\kappa}$ statistic (sanity), global agreement (assessment), positive agreement (compactness), negative agreement (separability) and the percentage of nodes presenting total agreement (perfectness): mean and the 95% confidence interval are presented for each combination of $\{k, \sigma, d\}$ parameters, averaging results over 10 random datasets for each of the 3 different network configurations (30 runs).

			$\hat{\kappa}$		$P(A)$		$P(A^+)$		$P(A^-)$		$P(A)=1$	
			Mean	*(95%CI)*	*Mean*	*(95%CI)*	*Mean*	*(95%CI)*	*Mean*	*(95%CI)*	*Mean*	*(95%CI)*
$k=2$	$\sigma=.01$	$d=8$	0.99	(0.98;1.00)	0.99	(0.99;1.00)	0.99	(0.99;1.00)	1.00	(0.99;1.00)	0.98	(0.96;1.00)
		$d=16$	0.99	(0.97;1.00)	0.99	(0.99;1.00)	0.99	(0.99;1.00)	0.99	(0.99;1.00)	0.98	(0.95;1.00)
		$d=32$	0.99	(0.98;1.00)	1.00	(0.99;1.00)	1.00	(0.99;1.00)	0.99	(0.99;1.00)	0.99	(0.98;1.00)
		$d=64$	1.00	(1.00;1.00)	1.00	(1.00;1.00)	1.00	(1.00;1.00)	1.00	(1.00;1.00)	1.00	(1.00;1.00)
		$d=128$	0.86	(0.78;0.94)	0.93	(0.89;0.97)	0.94	(0.90;0.97)	0.92	(0.88;0.97)	0.71	(0.55;0.87)
	$\sigma=.05$	$d=8$	0.94	(0.89;0.99)	0.97	(0.94;0.99)	0.97	(0.94;0.99)	0.97	(0.95;1.00)	0.92	(0.85;0.98)
		$d=16$	0.94	(0.89;0.99)	0.97	(0.95;0.99)	0.97	(0.95;0.99)	0.97	(0.95;0.99)	0.88	(0.79;0.97)
		$d=32$	0.88	(0.80;0.96)	0.94	(0.91;0.98)	0.95	(0.92;0.98)	0.91	(0.86;0.97)	0.80	(0.68;0.91)
		$d=64$	0.88	(0.81;0.94)	0.94	(0.91;0.97)	0.94	(0.91;0.97)	0.94	(0.90;0.97)	0.64	(0.49;0.79)
		$d=128$	0.71	(0.64;0.79)	0.86	(0.82;0.89)	0.86	(0.83;0.90)	0.85	(0.80;0.89)	0.32	(0.16;0.48)
	$\sigma=.10$	$d=8$	0.81	(0.73;0.88)	0.91	(0.87;0.94)	0.91	(0.88;0.95)	0.89	(0.85;0.93)	0.73	(0.63;0.83)
		$d=16$	0.86	(0.82;0.91)	0.93	(0.91;0.95)	0.93	(0.91;0.95)	0.93	(0.91;0.95)	0.63	(0.51;0.74)
		$d=32$	0.73	(0.66;0.80)	0.87	(0.83;0.90)	0.88	(0.84;0.91)	0.85	(0.81;0.90)	0.35	(0.25;0.46)
		$d=64$	0.67	(0.59;0.75)	0.84	(0.80;0.88)	0.85	(0.82;0.88)	0.82	(0.77;0.87)	0.20	(0.11;0.29)
		$d=128$	0.55	(0.49;0.60)	0.77	(0.75;0.80)	0.79	(0.77;0.81)	0.75	(0.72;0.78)	0.02	(0.02;0.03)
$k=3$	$\sigma=.01$	$d=8$	0.91	(0.85;0.97)	0.96	(0.93;0.99)	0.94	(0.90;0.98)	0.97	(0.95;0.99)	0.82	(0.70;0.95)
		$d=16$	0.97	(0.95;1.00)	0.99	(0.98;1.00)	0.98	(0.97;1.00)	0.99	(0.98;1.00)	0.92	(0.84;1.00)
		$d=32$	0.95	(0.91;0.99)	0.98	(0.96;1.00)	0.97	(0.95;0.99)	0.98	(0.96;1.00)	0.82	(0.70;0.93)
		$d=64$	1.00	(1.00;1.00)	1.00	(1.00;1.00)	1.00	(1.00;1.00)	1.00	(1.00;1.00)	1.00	(1.00;1.00)
		$d=128$	0.90	(0.84;0.95)	0.95	(0.92;0.98)	0.94	(0.90;0.97)	0.96	(0.93;0.98)	0.71	(0.54;0.87)
	$\sigma=.05$	$d=8$	0.95	(0.91;0.99)	0.98	(0.96;1.00)	0.97	(0.94;0.99)	0.98	(0.97;1.00)	0.93	(0.87;0.98)
		$d=16$	0.86	(0.81;0.91)	0.94	(0.91;0.96)	0.91	(0.87;0.94)	0.95	(0.93;0.97)	0.59	(0.47;0.70)
		$d=32$	0.75	(0.71;0.78)	0.88	(0.86;0.90)	0.84	(0.82;0.87)	0.90	(0.88;0.92)	0.21	(0.10;0.32)
		$d=64$	0.80	(0.76;0.85)	0.91	(0.89;0.93)	0.88	(0.85;0.90)	0.93	(0.91;0.94)	0.27	(0.15;0.38)
		$d=128$	0.76	(0.70;0.82)	0.89	(0.86;0.91)	0.85	(0.82;0.89)	0.91	(0.88;0.93)	0.09	(0.05;0.13)
	$\sigma=.10$	$d=8$	0.95	(0.93;0.97)	0.98	(0.97;0.99)	0.97	(0.95;0.98)	0.99	(0.98;0.99)	0.90	(0.86;0.94)
		$d=16$	0.74	(0.69;0.79)	0.88	(0.86;0.90)	0.83	(0.80;0.87)	0.91	(0.89;0.92)	0.46	(0.38;0.54)
		$d=32$	0.63	(0.58;0.68)	0.83	(0.80;0.85)	0.77	(0.74;0.80)	0.86	(0.84;0.88)	0.09	(0.04;0.14)
		$d=64$	0.59	(0.57;0.62)	0.81	(0.79;0.82)	0.75	(0.73;0.76)	0.84	(0.83;0.86)	0.04	(0.02;0.06)
		$d=128$	0.56	(0.54;0.58)	0.79	(0.78;0.80)	0.74	(0.73;0.75)	0.82	(0.81;0.83)	0.01	(0.00;0.01)
$k=4$	$\sigma=.01$	$d=8$	0.95	(0.91;0.99)	0.98	(0.97;1.00)	0.96	(0.93;0.99)	0.99	(0.98;1.00)	0.89	(0.80;0.97)
		$d=16$	0.97	(0.95;0.99)	0.99	(0.98;1.00)	0.98	(0.96;0.99)	0.99	(0.99;1.00)	0.86	(0.77;0.96)
		$d=32$	0.94	(0.91;0.98)	0.98	(0.96;0.99)	0.96	(0.94;0.99)	0.98	(0.97;0.99)	0.79	(0.66;0.92)
		$d=64$	0.95	(0.91;0.98)	0.98	(0.96;0.99)	0.96	(0.94;0.99)	0.98	(0.97;0.99)	0.71	(0.55;0.87)
		$d=128$	0.91	(0.87;0.96)	0.96	(0.94;0.98)	0.94	(0.90;0.97)	0.97	(0.96;0.99)	0.54	(0.39;0.69)
	$\sigma=.05$	$d=8$	0.88	(0.84;0.93)	0.96	(0.94;0.98)	0.91	(0.87;0.95)	0.98	(0.96;0.99)	0.71	(0.59;0.84)
		$d=16$	0.82	(0.78;0.86)	0.93	(0.92;0.95)	0.86	(0.83;0.90)	0.96	(0.95;0.97)	0.47	(0.36;0.59)
		$d=32$	0.78	(0.73;0.82)	0.91	(0.89;0.92)	0.84	(0.81;0.87)	0.93	(0.92;0.94)	0.18	(0.09;0.28)
		$d=64$	0.73	(0.69;0.76)	0.89	(0.87;0.90)	0.81	(0.78;0.83)	0.92	(0.91;0.93)	0.04	(0.02;0.06)
		$d=128$	0.66	(0.64;0.68)	0.86	(0.85;0.87)	0.76	(0.75;0.78)	0.90	(0.89;0.90)	0.00	(0.00;0.00)
	$\sigma=.10$	$d=8$	0.93	(0.88;0.98)	0.98	(0.96;0.99)	0.94	(0.90;0.98)	0.99	(0.98;1.00)	0.85	(0.76;0.94)
		$d=16$	0.81	(0.77;0.85)	0.93	(0.91;0.94)	0.85	(0.82;0.89)	0.95	(0.94;0.96)	0.40	(0.28;0.52)
		$d=32$	0.69	(0.65;0.73)	0.87	(0.85;0.89)	0.78	(0.75;0.81)	0.91	(0.89;0.92)	0.06	(0.04;0.08)
		$d=64$	0.67	(0.63;0.72)	0.86	(0.85;0.88)	0.77	(0.74;0.80)	0.90	(0.89;0.91)	0.01	(0.01;0.02)
		$d=128$	0.59	(0.56;0.61)	0.82	(0.81;0.83)	0.72	(0.70;0.74)	0.87	(0.86;0.88)	0.00	(0.00;0.00)
$k=5$	$\sigma=.01$	$d=8$	0.94	(0.90;0.98)	0.99	(0.98;1.00)	0.95	(0.91;0.98)	0.99	(0.99;1.00)	0.85	(0.75;0.95)
		$d=16$	0.95	(0.92;0.97)	0.98	(0.97;0.99)	0.96	(0.94;0.98)	0.99	(0.98;0.99)	0.83	(0.73;0.92)
		$d=32$	0.95	(0.93;0.97)	0.98	(0.98;0.99)	0.96	(0.94;0.98)	0.99	(0.98;0.99)	0.66	(0.52;0.80)
		$d=64$	0.93	(0.91;0.96)	0.98	(0.97;0.98)	0.95	(0.93;0.97)	0.98	(0.98;0.99)	0.44	(0.29;0.60)
		$d=128$	0.87	(0.83;0.91)	0.95	(0.94;0.97)	0.90	(0.87;0.93)	0.97	(0.96;0.98)	0.29	(0.15;0.43)
	$\sigma=.05$	$d=8$	0.86	(0.81;0.90)	0.97	(0.95;0.98)	0.88	(0.83;0.92)	0.98	(0.97;0.99)	0.72	(0.63;0.81)
		$d=16$	0.84	(0.79;0.88)	0.94	(0.92;0.96)	0.87	(0.84;0.91)	0.96	(0.95;0.97)	0.46	(0.33;0.59)
		$d=32$	0.76	(0.73;0.79)	0.92	(0.91;0.93)	0.82	(0.80;0.84)	0.95	(0.94;0.95)	0.10	(0.04;0.15)
		$d=64$	0.70	(0.67;0.73)	0.89	(0.88;0.90)	0.77	(0.75;0.80)	0.93	(0.92;0.93)	0.01	(0.00;0.02)
		$d=128$	0.67	(0.65;0.69)	0.88	(0.87;0.89)	0.75	(0.74;0.77)	0.92	(0.91;0.92)	0.00	(0.00;0.00)
	$\sigma=.10$	$d=8$	0.91	(0.86;0.95)	0.98	(0.97;0.99)	0.92	(0.88;0.96)	0.99	(0.98;0.99)	0.74	(0.60;0.88)
		$d=16$	0.83	(0.81;0.86)	0.94	(0.93;0.95)	0.87	(0.85;0.89)	0.96	(0.96;0.97)	0.35	(0.25;0.46)
		$d=32$	0.69	(0.65;0.73)	0.89	(0.87;0.90)	0.76	(0.73;0.79)	0.93	(0.92;0.94)	0.12	(0.05;0.20)
		$d=64$	0.67	(0.64;0.70)	0.87	(0.86;0.88)	0.76	(0.74;0.78)	0.91	(0.90;0.92)	0.01	(0.00;0.02)
		$d=128$	0.59	(0.58;0.61)	0.84	(0.84;0.85)	0.70	(0.69;0.71)	0.89	(0.89;0.90)	0.00	(0.00;0.00)
$k=6$	$\sigma=.01$	$d=8$	0.97	(0.94;1.00)	1.00	(0.99;1.00)	0.97	(0.95;1.00)	1.00	(1.00;1.00)	0.95	(0.89;1.00)
		$d=16$	0.95	(0.92;0.98)	0.99	(0.98;0.99)	0.96	(0.93;0.98)	0.99	(0.99;1.00)	0.78	(0.66;0.90)
		$d=32$	0.92	(0.89;0.95)	0.97	(0.96;0.98)	0.93	(0.91;0.96)	0.98	(0.98;0.99)	0.54	(0.37;0.70)
		$d=64$	0.88	(0.84;0.92)	0.96	(0.95;0.98)	0.90	(0.87;0.94)	0.98	(0.97;0.99)	0.40	(0.26;0.54)
		$d=128$	0.88	(0.86;0.90)	0.96	(0.96;0.97)	0.91	(0.89;0.92)	0.98	(0.97;0.98)	0.16	(0.04;0.28)
	$\sigma=.05$	$d=8$	0.90	(0.84;0.96)	0.99	(0.98;1.00)	0.91	(0.85;0.97)	0.99	(0.99;1.00)	0.83	(0.73;0.94)
		$d=16$	0.84	(0.80;0.87)	0.95	(0.94;0.96)	0.86	(0.83;0.89)	0.97	(0.97;0.98)	0.45	(0.36;0.55)
		$d=32$	0.81	(0.76;0.85)	0.94	(0.93;0.96)	0.84	(0.81;0.88)	0.97	(0.96;0.97)	0.24	(0.13;0.35)
		$d=64$	0.72	(0.69;0.75)	0.90	(0.89;0.92)	0.78	(0.76;0.81)	0.94	(0.93;0.95)	0.02	(0.01;0.02)
		$d=128$	0.63	(0.62;0.65)	0.88	(0.87;0.89)	0.71	(0.70;0.72)	0.93	(0.92;0.93)	0.00	(0.00;0.00)
	$\sigma=.10$	$d=8$	0.92	(0.87;0.97)	0.99	(0.98;0.99)	0.92	(0.88;0.97)	0.99	(0.99;1.00)	0.86	(0.77;0.94)
		$d=16$	0.76	(0.73;0.80)	0.94	(0.92;0.95)	0.80	(0.77;0.83)	0.96	(0.95;0.97)	0.20	(0.13;0.27)
		$d=32$	0.72	(0.69;0.76)	0.91	(0.90;0.92)	0.78	(0.75;0.81)	0.95	(0.94;0.95)	0.11	(0.06;0.16)
		$d=64$	0.65	(0.63;0.68)	0.89	(0.88;0.90)	0.73	(0.71;0.75)	0.93	(0.92;0.93)	0.01	(0.01;0.02)
		$d=128$	0.61	(0.60;0.63)	0.86	(0.86;0.87)	0.70	(0.69;0.71)	0.91	(0.91;0.92)	0.00	(0.00;0.00)
$k=7$	$\sigma=.01$	$d=8$	0.83	(0.71;0.94)	0.99	(0.98;0.99)	0.83	(0.72;0.95)	0.99	(0.99;1.00)	0.70	(0.53;0.87)
		$d=16$	0.96	(0.94;0.98)	0.99	(0.99;1.00)	0.96	(0.94;0.98)	0.99	(0.99;1.00)	0.80	(0.70;0.91)
		$d=32$	0.92	(0.89;0.95)	0.98	(0.97;0.99)	0.93	(0.90;0.96)	0.99	(0.98;0.99)	0.59	(0.44;0.74)
		$d=64$	0.87	(0.85;0.90)	0.97	(0.96;0.97)	0.89	(0.87;0.91)	0.98	(0.98;0.98)	0.17	(0.07;0.27)
		$d=128$	0.85	(0.82;0.87)	0.96	(0.95;0.96)	0.87	(0.85;0.89)	0.98	(0.97;0.98)	0.07	(0.01;0.12)
	$\sigma=.05$	$d=8$	1.00	(1.00;1.00)	1.00	(1.00;1.00)	1.00	(1.00;1.00)	1.00	(1.00;1.00)	1.00	(0.99;1.00)
		$d=16$	0.85	(0.82;0.89)	0.97	(0.96;0.97)	0.87	(0.84;0.90)	0.98	(0.98;0.98)	0.43	(0.30;0.55)
		$d=32$	0.77	(0.74;0.80)	0.94	(0.92;0.95)	0.81	(0.78;0.84)	0.96	(0.95;0.97)	0.09	(0.04;0.14)
		$d=64$	0.70	(0.67;0.72)	0.91	(0.90;0.92)	0.75	(0.73;0.77)	0.95	(0.94;0.95)	0.00	(0.00;0.00)
		$d=128$	0.66	(0.64;0.67)	0.89	(0.89;0.90)	0.72	(0.71;0.74)	0.93	(0.93;0.94)	0.00	(0.00;0.00)
	$\sigma=.10$	$d=8$	0.93	(0.88;0.97)	0.99	(0.99;1.00)	0.93	(0.89;0.97)	1.00	(0.99;1.00)	0.84	(0.72;0.96)
		$d=16$	0.83	(0.80;0.87)	0.96	(0.96;0.97)	0.85	(0.82;0.88)	0.98	(0.97;0.98)	0.41	(0.31;0.51)
		$d=32$	0.74	(0.71;0.77)	0.93	(0.93;0.94)	0.78	(0.76;0.81)	0.96	(0.96;0.97)	0.05	(0.03;0.08)
		$d=64$	0.65	(0.63;0.67)	0.90	(0.89;0.90)	0.72	(0.70;0.73)	0.94	(0.93;0.94)	0.00	(0.00;0.01)
		$d=128$	0.59	(0.58;0.60)	0.87	(0.86;0.87)	0.67	(0.67;0.68)	0.92	(0.91;0.92)	0.00	(0.00;0.00)

bust to an increase on the number of clusters, being, however, a bit more sensitive with respect to network size ($P(A)$ decreases when d increases) and cluster overlapping ($P(A)$ decreases when σ increases). This effect is observed also for the other indices. Given the comparison between $P(A^+)$ and $P(A^-)$, we can conclude that it is the extra sensitivity of the compactness agreement that decreases the quality of the overall agreement, enhancing its sensitivity to the afore mentioned parameters.

6. CONCLUDING REMARKS

In this work a local algorithm was proposed to perform clustering of sensors on ubiquitous sensor networks, based on the moving average of each node's data over time. We can argue that performing local clustering at each node, without a centralized server, is a valuable approximation of the global clustering, hence increasing the level of comprehension that each sensor node can have about the entire network. Local algorithms present an extremely high level of agreement with the global clustering, especially in terms of separability agreement, so each node is able to tell to which cluster it is assigned, and especially when queried with other node's data, tell if they should be separated or not.

Future work is focused on communication, as this is one of the most resource-consuming procedures of sensor networks [7]. If the concept of the data being produced in the network is stable, then the clustering estimates will converge, and transmissions will become redundant. We should include mechanisms to allow each sensor to decide to which neighbors it is still valuable to send information. However, the world is not static. It is possible that, with time, the sketches of each sensor will change, adapting to new concepts of data. On a long run, the communication management strategy could prevent the system from adapting to new data. Methods should include change detection [16] mechanisms that would trigger if the data changes, either univariatedly at each sensor, or in the global interaction of sensor data. From another point of view, since each node clusters its neighbors' centroids as single points, the node could fit a clustering definition with different number of clusters. Moreover, it could several clustering definitions and test which of them is better, using, for example, the Davies-Bouldin index [19] which has the positive characteristic of not depending on the number of clusters, hence enabling the comparison of clusterings with different k. The application to real data is also being performed, using data from real load demand sensor networks used in electricity distribution networks [29].

Sensor network comprehension is a wider concept than the two clustering task that were inspected in this paper. Other tasks may yield additional elements for a global sensor network comprehension: the extraction of rules for certain network events, which may reveal breaches of security in the current network topology; inspection of predictive errors across the network, which may reveal interactions between sensors not observed in unsupervised results; or the definition of a ranking of sensor activity, which may reveal unused or overloaded sensors in the network. The main focus of any sensor network comprehension process should be on using distributed processing of data and queries, and distributed data mining procedures, enabling fast answers and access from transient mobile devices.

7. ACKNOWLEDGMENTS

The work of P.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 CRACS. This work was done under the joint scope of FCT projects KDUDS (PTDC/EIA-EIA/98355/2008) and CALLAS (PTDC/EIA/71462 /2006). The first author also thanks the help of Cristina Santos on fruitful discussions on agreement theory.

8. REFERENCES

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *VLDB 2003, Proceedings of 29th International Conference on Very Large Data Bases*, pages 81–92. Morgan Kaufmann, September 2003.

[2] P. Baldwin, S. Kohli, E. A. Lee, X. Liu, and Y. Zhao. Modelling of Sensor Nets in Ptolemy II. In *Proceedings of the Third International Symposium on Information Processing in Sensor Networks (IPSN'04)*, pages 359–368. ACM Press, 2004.

[3] S. Bandyopadhyay, C. Giannella, U. Maulik, H. Kargupta, K. Liu, and S. Datta. Clustering distributed data streams in peer-to-peer environments. *Information Sciences*, 176(14):1952–1985, 2006.

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

[5] J. Beringer and E. Hüllermeier. Online clustering of parallel data streams. *Data and Knowledge Engineering*, 58(2):180–204, August 2006.

[6] P. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 9–15. AAAI Press, 1998.

[7] H. Chan, M. Luk, and A. Perrig. Using clustering information for sensor network localization. In *First IEEE International Conference on Distributed Computing in Sensor Systems*, pages 109–125, 2005.

[8] J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20:37–46, 1960.

[9] G. Cormode, S. Muthukrishnan, and W. Zhuang. Conquering the divide: Continuous clustering of distributed data streams. In *Proceedings of the 23rd International Conference on Data Engineering (ICDE 2007)*, pages 1036–1045, 2007.

[10] B.-R. Dai, J.-W. Huang, M.-Y. Yeh, and M.-S. Chen. Adaptive clustering for multiple evolving streams. *IEEE Transactions on Knowledge and Data Engineering*, 18(9):1166–1180, September 2006.

[11] S. Datta, K. Bhaduri, C. Giannella, R. Wolff, and H. Kargupta. Distributed data mining in peer-to-peer networks. *IEEE Internet Computing*, 10(4):18–26, 2006.

[12] P. Domingos and G. Hulten. A general method for scaling up machine learning algorithms and its application to clustering. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pages 106–113, 2001.

[13] J. Eker, J. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Sachs, and Y. Xiong. Taming heterogeneity - the Ptolemy approach. *Proceedings of the IEEE*, 91(1):127–144, 2003.

[14] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Second Int Conf on Knowledge Discovery and Data Mining*, pages 226–231, Portland, Oregon, 1996. AAAI Press.

[15] M. M. Gaber and P. S. Yu. A framework for resource-aware knowledge discovery in data streams: a holistic approach with its application to clustering. In *Proceedings of the ACM Symposium on Applied Computing*, pages 649–656, 2006.

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

[17] J. Gama, R. Sebastião, and P. P. Rodrigues. Issues in evaluation of stream learning windows. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2009)*, pages 329–337, Paris, France, 2009. ACM Press.

[18] T. F. Gonzalez. Clustering to minimize the maximum inter-cluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[19] M. Halkidi, Y. Batistakis, and M. Varzirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145, 2001.

[20] J. Ibriq and I. Mahgoub. Cluster-based routing in wireless sensor networks: Issues and challenges. In *International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, pages 759–766, 2004.

[21] T. Idé. Why does subsequence time-series clustering produce sine waves. In *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery from Databases (PKDD 2006)*, volume 4213 of *LNAI*, pages 211–222, Berlin, Germany, September 2006. Springer Verlag.

[22] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[23] H. Kargupta, W. Huang, K. Sivakumar, and E. L. Johnson. Distributed clustering using collective principal component analysis. *Knowledge and Information Systems*, 3(4):422–448, 2001.

[24] E. J. Keogh, J. Lin, and W. Truppel. Clustering of time series subsequences is meaningless: Implications for previous and future research. In *Proceedings of the IEEE International Conference on Data Mining*, pages 115–122. IEEE Computer Society Press, 2003.

[25] M. Klusch, S. Lodi, and G. Moro. Distributed clustering based on sampling local density estimates. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 485–490, 2003.

[26] S. Muthukrishnan. *Data Streams: Algorithms and Applications*. Now Publishers Inc, New York, NY, 2005.

[27] L. O'Callaghan, A. Meyerson, R. Motwani, N. Mishra, and S. Guha. Streaming-data algorithms for high-quality clustering. In *Proceedings of the Eighteenth Annual IEEE International Conference on Data Engineering*, pages 685–696. IEEE Computer Society, 2002.

[28] P. P. Rodrigues and J. Gama. Clustering techniques in sensor networks. In *Learning from Data Streams*, chapter 9, pages 125–142. Springer Verlag, 2007.

[29] P. P. Rodrigues and J. Gama. A system for analysis and prediction of electricity load streams. *Intelligent Data Analysis*, 13(3):477–496, June 2009.

[30] P. P. Rodrigues, J. Gama, and L. Lopes. Clustering distributed sensor data streams. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECMLPKDD 2008)*, volume 5212 of *Lecture Notes in Artificial Intelligence*, pages 282–297, Antwerpen, Belgium, September 2008. Springer Verlag.

[31] P. P. Rodrigues, J. Gama, and L. Lopes. Requirements for clustering streaming sensors. In *Knowledge Discovery from Sensor Data*, chapter 4, pages 33–51. CRC Press, 2008.

[32] P. P. Rodrigues, J. Gama, and L. Lopes. Knowledge discovery for sensor network comprehension. In *Intelligent Techniques for Warehousing and Mining Sensor Network Data*, chapter 6, pages 118–135. IGI Global, 2010.

[33] P. P. Rodrigues, J. Gama, and J. P. Pedroso. Hierarchical clustering of time-series data streams. *IEEE Transactions on Knowledge and Data Engineering*, 20(5):615–627, May 2008.

[34] D. M. Sherrill, M. L. Moy, J. J. Reilly, and P. Bonato. Using hierarchical clustering methods to classify motor activities of copd patients from wearable sensor data. *Journal of Neuroengineering and Rehabilitation*, 2(16), 2005.

[35] J.-Z. Sun and J. Sauvola. Towards advanced modeling techniques for wireless sensor networks. In *Procs of 1st Int Symp on Pervasive Computing and Applications*, pages 133–138. IEEE Computer Society, 2006.

[36] W. Wang, J. Yang, and R. R. Muntz. STING: A statistical information grid approach to spatial data mining. In *23rd Int Conf on Very Large Data Bases*, pages 186–195, Greece, 1997. Morgan Kaufmann.

[37] M. J. Warrens. On the equivalence of cohen's kappa and the hubert-arabie adjusted rand index. *Journal of Classification*, 25(2):177–183, November 2008.

[38] J. Yin and M. M. Gaber. Clustering distributed time series in sensor networks. *Icdm 2008: Eighth Ieee International Conference On Data Mining, Proceedings*, pages 678–687, 2008.

[39] O. Younis and S. Fahmy. HEED: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. *IEEE Transactions on Mobile Computing*, 3(4):366–379, 2004.

[40] K. Zhang, K. Torkkola, H. Li, C. Schreiner, H. Zhang, M. Gardner, and Z. Zhao. A context aware automatic traffic notification system for cell phones. In *27th International Conference on Distributed Computing Systems Workshops (ICDCSW '07)*, pages 48–50. IEEE Computer Society, 2007.

[41] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2):141–182, 1997.

Energy Prediction Based on Resident's Activity

Chao Chen
Washington State University
Pullman, WA 99164
USA
cchen@eecs.wsu.edu

Barnan Das
Washington State University
Pullman, WA 99164
USA
barnandas@wsu.edu

Diane J. Cook
Washington State University
Pullman, WA 99164
USA
cook@eecs.wsu.edu

ABSTRACT
In smart home environment research, little attention has been given to monitoring, analyzing, and predicting energy usage, despite the fact that electricity consumption in homes has grown dramatically in the last few decades. We envision that a potential application of this smart environment technology is predicting the energy would be used to support specific daily activities. The purpose of this paper is thus to validate our hypothesis that energy usage can be predicted based on sensor data that can be collected and generated by the residents in a smart home environment, including recognized activities, resident movement in the space, and frequency of classes of sensor. In this paper, we extract useful features from sensor data collected in a smart home environment and utilize several machine learning algorithms to predict energy usage given these features. To validate these algorithms, we use real sensor data collected in our CASAS smart apartment testbed. We also compare the performance between different learning algorithms and analyze the prediction results for two different experiments performed in the smart home.

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

General Terms
Algorithms, Performance, Experimentation, Human Factors.

Keywords
Energy Prediction, Smart Environments, Machine Learning.

1. INTRODUCTION
Recently, smart home environments have become a very popular topic, due to a convergence of technologies in machine learning and data mining as well as the development of robust sensors and actuators. In this research, attention has been directed toward the area of health monitoring and activity recognition. Georgia Tech Aware Home [2] identifies people based on the pressure sensors embedded into the smart floor in strategic locations. This sensor system can be used for tracking inhabitant and identifying user's location. The Neural Network House [3] designs an ACHE system, which provides an Adaptive Control of Home Environment, in which the home is proactive to program itself with the lifestyle and desires of the inhabitant. The smart hospital project [4] develops a robust approach for recognizing user's activities and estimating hospital-staff activities using a hidden Markov model with contextual information in the smart hospital environment. MIT researchers [5] recognize user' s activities by using a set of small and simple state-change sensors, which are easy and quick to install in the home environment. Unlike one resident system, this system is employed in multiple inhabitant environments and can be used to recognize Activities of Daily Living (ADL). CASAS Smart Home Project [6] builds probabilistic models of activities and used them to recognize activities in complex situations where multiple residents are performing activities in parallel in the same environment.

Based on a recent report [7], buildings are responsible for at least 40% of energy use in most countries. As an important part of buildings, household consumption of electricity has been growing dramatically. Thus, the need to develop technologies that improve energy efficiency and monitor the energy usage of the devices in household is emerging as a critical research area. The BeAware project [8] makes use of an iPhone application to give users alerts and to provide information on the energy consumption of the entire house. This mobile application can detect the electricity consumption of different devices and notify the user if the devices use more energy than expected. The PowerLine Positioning (PLP) indoor location system [9] is able to localize to sub-room level precision by using fingerprinting of the amplitude of tones produced by two modules installed in extreme locations of the home. Later work of this system [10] records and analyzes electrical noise on the power line caused by the switching of significant electrical loads by a single, plug-in module, which can connect to a personal computer, then uses machine learning techniques to identify unique occurrences of switching events by tracking the patterns of electrical noise. The MITes platform [11] monitors the changes of various appliances in current electricity flow for the appliance, such as a switch from on to off by installing the current sensors for each appliance. Other similar work [12] also proposes several approaches to recognize the energy usage of electrical devices by the analysis of power line current. It can detect whether the appliance is used and how it is used.

In our study, we extend smart home research to consider the resident's energy usage. We envision three applications of smart environments technologies for environmental energy efficiency:

SensorKDD'10, July 25, 2010, Washington, DC, USA.

1) analyzing electricity usage to identify trends and anomalies, 2) predicting the energy that will be used to support specific daily activities, and 3) automating activity support in a more energy-efficient manner. In this paper, we focus on the second task. The purpose of this paper is thus to validate our hypothesis that energy usage can be predicted based on sensor data that can be collected and generated by the residents in a smart home environment, including automatically-recognized activities, resident movement in the space, and frequency of classes of sensor events. The results of this work can be used to give residents feedback on energy consumption as it relates to various activities. Ultimately this information can also be used to suggest or automate activities in a more energy-efficient way.

In section 2, we introduce our CASAS smart environment architecture and describe our data collection and annotation modules. Section 3 presents the relationship between the energy data and the activities and describes machine learning methods to predict energy usage. Section 4 summarizes the results of our experiments and compares the performance between different learning methods and different experimental parameters.

2. CASAS SMART ENVIRONMENT

The smart home environment testbed that we are using to predict energy usage is a three bedroom apartment located on the Washington State University campus.

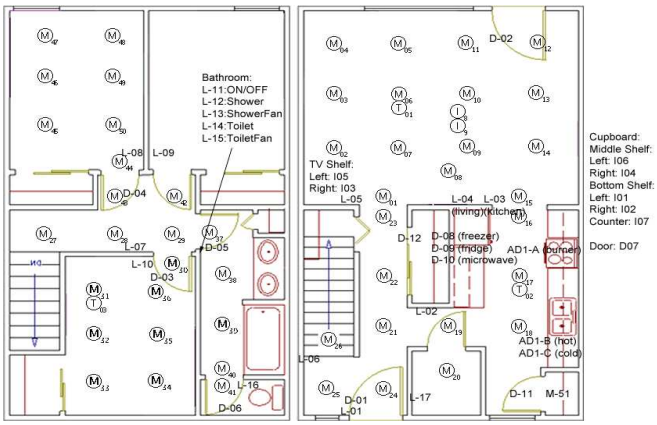


Figure 1. Three-bedroom smart apartment used for our data collection (motion (M), temperature (T), water (W), burner (B), telephone (P), and item (I)).

As shown in Figure 1, the smart home apartment testbed consists of three bedrooms, one bathroom, a kitchen, and a living/dining room. To track people's mobility, we use motion sensors placed on the ceilings. The circles in the figure stand for the positions of motion sensors. They facilitate tracking the residents who are moving through the space. In addition, the testbed also includes temperature sensors as well as custom-built analog sensors to provide temperature readings and hot water, cold water and stove burner use. A power meter records the amount of instantaneous power usage and the total amount of power which is used. An in-house sensor network captures all sensor events and stores them in a SQL database. The sensor data gathered for our study is expressed by several features, summarized in Table 1. These four fields (Date, Time, Sensor, ID and Message) are generated by the CASAS data collection system automatically.

Table 1. Sample of sensor events used for our study. The first two events correspond to motion sensor ON and OFF messages. The third event is an ambient temperature reading, and the last two events represent current electricity usage.

Date	Time	Sensor ID	Message
2009-02-06	17:17:36	M45	ON
2009-02-06	17:17:40	M45	OFF
2009-02-06	11:13:26	T004	21.5
2009-02-05	11:18:37	P001	747W
2009-02-09	21:15:28	P001	1.929kWh

To provide real training data for our machine learning algorithms, we collect data while two students in good health were living in the smart apartment. Our training data was gathered over a period of several months and more than 100,000 sensor events were generated for our dataset. Each student had a separate bedroom and shared the downstairs living areas in the smart apartment. All of our experimental data are produced by these two students' normal lives, which guarantee that the results of this analysis are real and useful.

After collecting data from the CASAS smart apartment, we annotated the sensor events with the corresponding activities that were being performed while the sensor events were generated. Because the annotated data is used to train the machine learning algorithms, the quality of the annotated data is very important for the performance of the learning algorithms. As a large number of sensor data events are generated in a smart home environment, it becomes difficult for researchers and users to interpret raw data into residents' activities [13] without the use of visualization tools.

To improve the quality of the annotated data, we built an open source Python Visualizer, called PyViz, to visualize the sensor events. Figure 2 shows the user interface of PyViz for the CASAS project. PyViz can display events in real-time or in playback mode from a captured file of sensor event readings. Furthermore, we also built an Annotation Visualizer to visualize the resident's activities as shown in Figure 3.

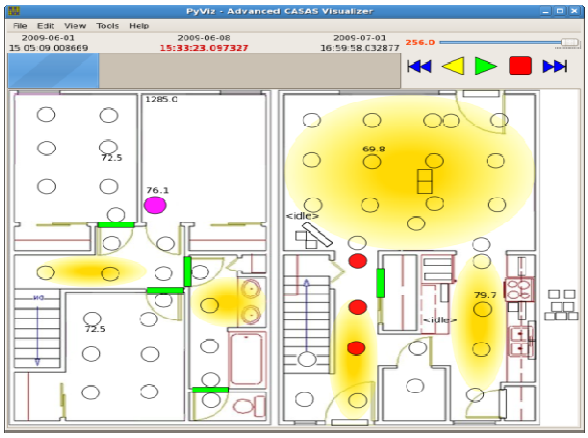


Figure 2. PyViz visualizer.

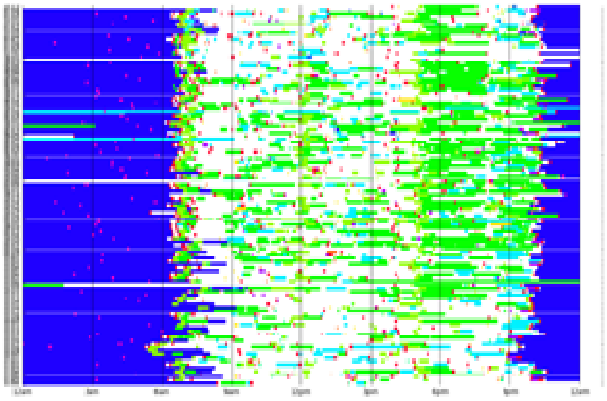


Figure 3. Visualizing activities in a smart home environment.

With the help of PyViz, activity labels are optionally added to each sensor event, providing a label for the current activity. For our experiment, we selected six activities that the two volunteer participants regularly perform in the smart apartment to predict energy use. These activities are as follows:

1. Work at computer
2. Sleep
3. Cook
4. Watch TV
5. Shower
6. Groom

All of the activities that the participants perform have some relationship with measurable features such as the time of day, the participants' movement patterns throughout the space, and the on/off status of various electrical appliances. These activities are either directly or indirectly associated with a number of electrical appliances and thus have a unique pattern of power consumption. Table 2 gives a list of appliances associated with each activity. It should be noted that, there are some appliances which are in "always on" mode, such as the heater (in winter), refrigerator, phone charger, etc. Thus, we postulate that the activities will have a measurable relationship with the energy usage of these appliances as well.

Table 2.Electricical appliances associated with each activity.

Activity	Appliances Directly Associated	Appliances Indirectly Associated
Work at computer	Computer, printer	Localized lights
Sleep	None	None
Cook	Microwave, oven, stove	Kitchen lights
Watch TV	TV, DVD player	Localized lights
Shower	Water heater	Localized lights
Groom	Blow drier	Localized lights

3. ENERGY ANALYSIS

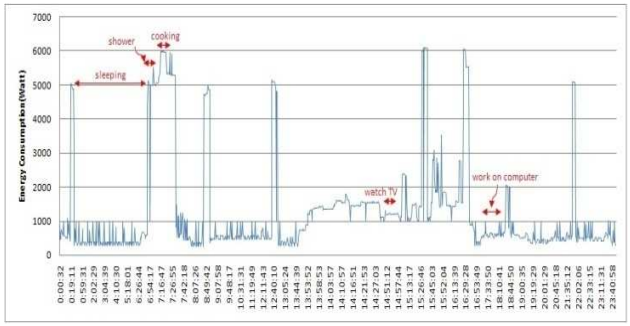


Figure 4. Energy usage for a single day.

Figure 4 shows the energy fluctuation that occurred during a single day on June 2nd, 2009. The activities have been represented by red arrows. The length of the arrows indicates the duration of time (not to scale) for different activities. Note that there are a number of peaks in the graph even though these peaks do not always directly correspond to a known activity. These peaks are due to the water heater, which has the highest energy consumption among all appliances, even though it was not used directly. The water heater starts heating by itself whenever the temperature of water falls below a certain threshold.

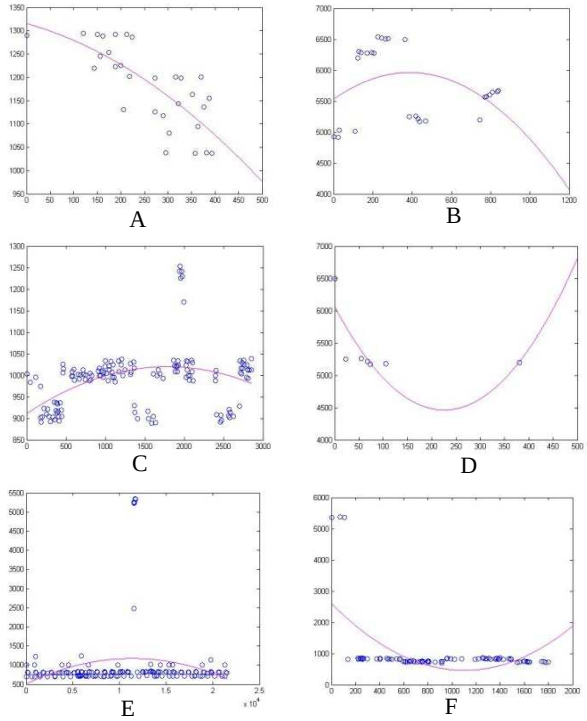


Figure 5. Energy data curve fitting for each activity.

(X-axis: wattage; Y-axis: second; A: Shower; B: Cook; C: Work on computer; D: Groom; E: Sleep; F: Watch TV)

Figure 5 plots typical energy data for each activity together with the result of applying curve fitting to the data. Curve fitting [14] is the process of building a mathematical function model that can

best fit to a series of data points. It serves as an aid for data visualization, to approximate the values when no data are available, and to express the relationships between different data points. From the figure, we see that each resident's activity generates different energy patterns. The "cook" activity consumes the highest energy because the participants may open the refrigerator and use the stove or microwave oven, which need a relatively high power. Meantime, when the participants were sleeping, the energy consumption was the lowest because most appliances were idle.

3.1 Feature Extraction

Data mining and machine learning techniques use enormous volumes of data to make appropriate predictions. Before making use of these learning algorithms, another important step is to extract useful features or attributes from the raw annotated data. We have considered some features that would be helpful in energy prediction. These features have been generated from the raw sensor data by our feature extraction module. The following is a listing of the resulting features that we used in our energy prediction experiments.

1. Activity label
2. Activity length (in seconds)
3. Previous activity
4. Next activity
5. Number of kinds of motion sensors involved
6. Total number of times of motion sensor events triggered
7. Motion sensor M1…M51 (On/Off)

Target Feature: Total energy consumption range for an activity (in watts)

Activity label gives the name of the activity performed. Activity length is the duration of time a particular activity takes from beginning to the end. Features 3 and 4 represent the preceding and the succeeding activities to the current activity. Feature 5 takes into account the total number of different unique sensors used. Features 6 keeps a record of total number of sensor events associated with an activity. Feature 7 is not just one feature, but a collection of 51 features each representing a single motion sensor. Each of these sensor data records the total number of times a motion sensor was fired.

The input to the learning algorithm is a list of these seven features as computed for a particular activity that was performed. The output of the learning algorithm is the amount of electricity that is predicted to be consumed while performing the activity. To address the goal of predicting energy usage, we discretize the energy readings using equal width binning. Equal width binning [15] is also widely used in data exploration, preparation, and mining. Both of these binning techniques have been used to preprocess continuous-valued attributes by creating a specified number of bins, or numeric ranges. These benchmarks can be used to evaluate other machine learning classifiers we use in our experiments. In this paper, we discretize the target average energy data into several interval sizes (two classes, three classes, four classes and five classes, six classes) to assess the performance of our experiments.

3.2 Feature Selection

During feature extraction, our algorithm generates a large number of features to describe a particular situation. However, some of these features are redundant or irrelevant, resulting in a drastic raise of computational complexity and classification errors [16]. Features are selected by a method called attribute subset selection which finds a minimum set of attributes such that the resulting probability distribution of the data classes is as close as possible to the original distribution obtained using all attributes. In this paper, we have used information gain [17] to create a classification model, which can measure how well a given attribute separate the training examples according to their target classification. The performance of each attribute is measured in terms of a parameter known as information gain. It is a measure based on entropy, a parameter used in information theory to characterize the purity of an arbitrary collection of examples. It is measured as:

$$Entropy(S) \equiv -p_+ \log_2 p_+ - p_- \log_2 p_-$$

where, S is the set of data points, P^+ is number of data points that belong to one class (the positive class) and P^- is the number of data points that belong to the negative class. We adapt this measure to handle more than two classes for our experiments.

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where, *Values (A)* is the set of all possible values for attribute A. *Gain(S, A)* measures how well a given attribute separates the training examples according to their target classification. By using information gain, we can determine which features are comparatively more important than others for the task of target classification.

3.3 Energy Prediction

Machine learning [18] algorithms are capable to learn and recognize complex patterns and classify objects based on sensor data. In our study, we make use of four popular machine learning methods to represent and predict energy usage based on the features we selected: a Naïve Bayes Classifier, a Bayes Net Classifier, a Neural Network Classifier, and a Support Vector Machine. We test these four algorithms on the data collected in the CASAS smart home apartment testbed.

3.3.1 Naïve Bayes Classifier

A naïve Bayes Classifier [19] is a simple probabilistic classifier that assumes the presence of a particular feature of a class is unrelated to any other features. It applies Bayes' theorem to learn a mapping from the features to a classification label.

$$\text{argmax}_{e_i \in E} P(e_i|F) = \frac{P(F|e_i)P(e_i)}{P(F)}$$

In this equation, E represents the energy class label and F stands for the features values we describe above. $P(e_i)$ is estimated by counting the frequency with which each target value e_i occurs in the training data. Based on the simplifying assumption that feature values are independent given the target values, the probabilities of observing the features is the product of the probabilities for the individual features:

$$P(F|e_j) = \prod_i P(f_i|e_j)$$

Despite its naïve design and over-simplified assumptions, the naïve Bayes classifier often works more effectively in many complex real world situations than other classifiers. It only requires a small amount of training data to estimate the parameters needed for classification.

3.3.2 Bayes Net

Bayes belief networks [20] belong to the family of probabilistic graphical models. They represent a set of conditional independence assumptions by a directed acyclic graph, whose nodes represent random variables and edges represent direct dependence among the variables and are drawn by arrows by the variable name. Unlike the naïve Bayes classifier, which assumes that the values of all the attributes are conditionally independent given the target value, Bayesian belief networks apply conditional independence assumptions only to the subset of the variables. They can be suitable for small and incomplete data sets and they incorporate knowledge from different sources. After the model is built, they can also provide fast responses to queries.

3.3.3 Artificial Neural Network

Artificial Neural Networks (ANNs) [21] are abstract computational models based on the organizational structure of the human brain. ANNs provide a general and robust method to learn a target function from input examples. The most common learning method for ANNs, called Backpropagation, which performs a gradient descent within the solution's vector space to attempt to minimize the squared error between the network output values and the target values for these outputs. Although there is no guarantee that an ANN will find the global minimum and the learning procedure may be quite slow, ANNs can be applied to problems where the relationships are dynamic or non- linear and capture many kinds of relationships that may be difficult to model by other machine learning methods. In our experiment, we choose the Multilayer-Perceptron algorithm with Backpropagation to predict electricity usage.

3.3.4 Support Vector Machine

Super Vector Machines (SVMs) were first introduced in 1992 [22]. This is a training algorithm for data classification, which maximizes the margin between the training examples and the class boundary. The SVM learns a hyperplane which separates instances from multiple activity classes with maximum margin. Each training data instance should contain one class label and several features. The goal of a SVM is to generate a hyperplane which provides a class label for each data point described by a set of feature values.

4. EXPERIMENT RESULTS

We performed two series of experiments. The first experiment uses the sensor data collected during two summer months in the testbed. In the second experiment, we collected data of three winter months in the testbed. The biggest difference between these two groups of data is that some high energy consuming devices like room heaters were only used during the winter, which are not directly controlled by the residents and are therefore difficult to monitor and predict. The test tool we use, called Weka [23], provides an implementation of learning algorithms that we can easily apply to our own dataset. Using Weka, we assessed the classification accuracy of our four selected machine learning algorithms using 3-fold cross validation.

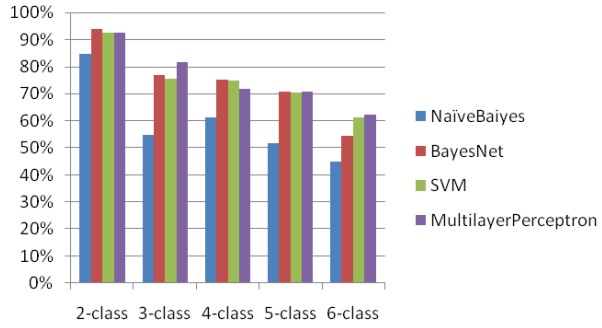


Figure 6. Comparison of the accuracy for summer dataset.

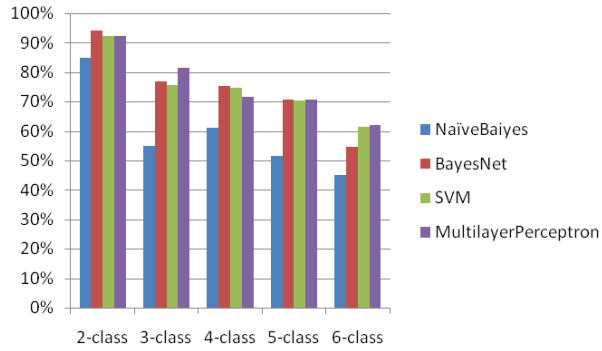


Figure 7. Comparison of the accuracy for winter dataset.

Figures 6 and 7 plot the accuracies of the two different group experiments, respectively. As shown in these two figures, the highest accuracy is around 90% for both datasets to predict the two-class energy usage and the lowest accuracy is around 60% for the six-class case in both datasets. These results also show that the higher accuracy will be found when the precision was lower because the accuracy of all four methods will drop from about 90% to around 60% with an increase in the number of energy class labels.

From the figures we see that the Naïve Bayes Classifier performs worse than the other three classifiers. This is because it is based on the simplified assumption that the feature values are conditionally independent given the target value. On the contrary, the features that we use, are not conditionally independent. For example, the motion sensors associated with an activity is used to find the total number of times motion sensor events were triggered and also the kinds of motion sensors involved.

To analyze the effectiveness of decision tree feature selection, we apply the ANN algorithm to both datasets with and without feature selection. From Figure 8, we can see the time efficiency has been improved greatly using feature selection. The time for building the training model drops from around 13 seconds to 4 seconds after selecting the features with high information gain. However, as seen in Figure 9, the classification accuracy is almost the same or a slight better than the performance without feature selection. The use of feature selection can improve the time

performance without reducing the accuracy performance in the original data set.

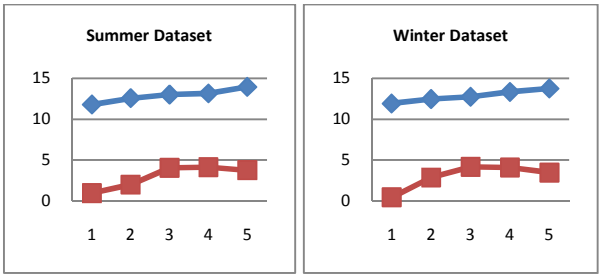


Figure 8. Comparison of time efficiency.

(1:2-class; 2:3-class; 3:4-class; 4:5-class; 5:6-class; Y-axis: second; Red: with feature selection; Blue: without feature selection). Time is plotted in seconds.

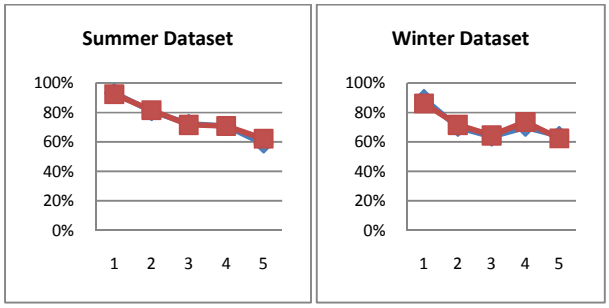


Figure 9. Comparison of prediction accuracy.

(1:2-class; 2:3-class; 3:4-class; 4:5-class; 5:6-class; Red: with feature selection; Blue: without feature selection).

Figure 10 compares the performance of the ANN applied to the winter and summer data sets. From the graph, we see that the performance for the summer data set is shade better than the performance for the winter dataset. This is likely due to the fact that the room and floor heater appliances are used during winter, which consumes a large amount of energy and are less predictable than the control of other electrical devices in the apartment.

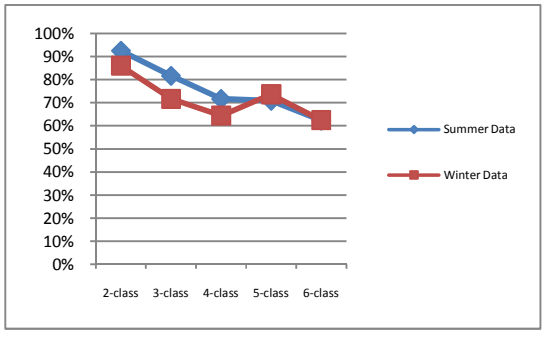


Figure 10 Comparison of the accuracy between two datasets.

5. DISCUSSIONS

Analyzing these results, we see that machine learning methods can be used as a tool to predict energy usage in smart home environments based on the human's activity and mobility. However, the accuracy of these methods is not as high as we anticipated when the energy data is divided into more than three classes. There are several reasons that lead to low performance of these algorithms. One reason is that some of the major devices are difficult to monitor and predict, such as the floor heater, which may rely on the outdoor temperature of the house. Another reason is that there is no obvious cycle of people's activities. An additional factor we can't ignore is that there is some noise and perturbation motion when the sensors record data and transfer them into the database. Finally, the sensor data we collect is not enough to predict energy precisely. As a result, we intend to collect more kinds of sensor data to improve the prediction performance.

6. CONCLUSIONS

In this work, we introduced a method of predicting energy usage using an integrated system of collecting sensor data and applied machine learning in a smart home environment. To predict energy precisely, we extracted features from real sensor data in a smart home environment and selected the most important features based on information gain, then used an equal width binning method to discretize the value of the features. To assess the performance of the four machine learning methods, we performed two group experiments during two different periods, analyzed the results of the experiments and provided the explanation of those results.

In our ongoing work, we plan to further investigate new and pertinent features to predict the energy more accurately. To improve the accuracy of energy prediction, we intend to install more sensitive sensors to capture more useful information in the smart home environment. We are also planning to apply different machine learning methods to different environments in which different residents perform similar activities. This will allow us to analyze whether the same pattern exists across residents and environments. In our next step we will analyze the energy usage data itself to find trends and cycles in the data viewed as a time series. The results of our work can be used to give residents feedback on energy consumption as it relates to various activities and be also treated as a reference to research human's life style in their homes. In addition, predicted electricity use can form the basis for automating the activities in a manner that consumes fewer resources including electricity.

7. REFERENCES

[1] Brumitt, B., et al. 2000. Multi-Camera Multi-Person Tracking for EasyLiving. In *Conf. Proc. 3rd IEEE Intl. Workshop on Visual Surveillance*.

[2] Orr, R. J. and Abowd, G. D. 2000. The smart floor: A mechanism for natural user identification and tracking. In *Conference on Human Factors in Computing Systems.* 275–276.

[3] Mozer, M. C. 1998. The Neural Network House: An Environment hat Adapts to its Inhabitants. In *Proc. AAAI Spring Symp. Intelligent Environments.*

[4] Sánchez, D., Tentori, M. and Favela, J. 2008. Activity recognition for the smart hospital. *IEEE Intelligent Systems.* 23, 2 , 50–57.

[5] Tapia, E. M., Intille, S. S. and Larson, K. 2004. Activity recognition in the home using simple and ubiquitous sensors. *Pervasive Computing*. 158–175.

[6] Singla, G., Cook, D. J. and Schmitter-Edgecombe, M. 2010. Recognizing independent and joint activities among multiple residents in smart environments. *Journal of Ambient Intelligence and Humanized Computing*. 1–7.

[7] Energy Efficiency in Buildings. 2009. DOI= www.wbcsd.org.

[8] BeAware. 2010. DOI= www.energyawareness.eu/beaware.

[9] Patel, S.N., Truong, K.N. and Abowd, G. D. 2006. PowerLine Positioning: A Practical Sub-Room-Level Indoor Location System for Domestic Use. In *proceedings of UbiComp 2006: 8th international conference*. Springer Berlin / Heidelberg, 441-458.

[10] Patel, S.N., et al. 2007. At the flick of a switch: Detecting and classifying unique electrical events on the residential power line, In *proceedings of UbiComp 2007: 9th international conference*. Innsbruck, Austria, 271.

[11] Tapia, E., et al. 2006. The design of a portable kit of wireless sensors for naturalistic data collection. *Pervasive Computing*. 117–134.

[12] Bauer, G., Stockinger, K. and Lukowicz, P. 2009. Recognizing the Use-Mode of Kitchen Appliances from Their Current Consumption. *Smart Sensing and Context*. 163–176.

[13] Szewcyzk, S., et al. 2009. Annotating smart environment sensor data for activity learning. *Technol. Health Care*. 17, 3, 161-169.

[14] Coope, I. D. 1993. Circle fitting by linear and nonlinear least squares. *Journal of Optimization Theory and Applications*. 76, 2, 381–388.

[15] Liu, H., et al. 2002. Discretization: An enabling technique. *Data Mining and Knowledge Discovery*. 6, 4, 393–423.

[16] Bellman, R. E. 1961. *Adaptive control processes - A guided tour*. *Princeton*, New Jersey, U.S.A.: Princeton University Press. 255.

[17] Quinlan, J. R. 1986. Induction of Decision Trees. *Mach. Learn.*, 1, 1, 81-106.

[18] Mitchell, T. 1997. *Machine Learning*, New York，AMcGraw Hill.

[19] Rish, I. 2001. An empirical study of the naive Bayes classifier. In *IJCAI-01 workshop on "Empirical Methods in AI"*.

[20] Pearl, J. 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference*, Morgan Kaufmann.

[21] Zornetzer, S. F. 1955. *An introduction to neural and electronic networks*, Morgan Kaufmann.

[22] Boser, B. E., Guyon, I. M. and Vapnik, V. N. 1992. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, Pittsburgh, Pennsylvania, United States, 144-152.

[23] Witten, I. H. and Frank, E. 1999. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations* (The Morgan Kaufmann Series in Data Management Systems), Morgan Kaufmann.

SHORT RESEARCH PAPERS

Multi Home Transfer Learning for Resident Activity Discovery and Recognition

Parisa Rashidi
Washington State University
Pullman, Washington
prashidi@eecs.wsu.edu

Diane J. Cook
Washington State University
Pullman, Washington
cook@eecs.wsu.edu

ABSTRACT

Activity discovery and recognition can provide unprecedented opportunities for health monitoring, automation, energy efficiency and security. Despite all the potential benefits, in practice we are faced with the main challenge of collecting huge amounts of data for each new physical space in order to carry out the conventional activity discovery algorithms. This results in a prolonged installation in the real world. More importantly, if we ignore what has been learned in previous spaces, we face redundant computational effort and time investment and we miss the insights gained from past experience that can improve the recognition accuracy. To overcome this problem, we propose a method of transferring the knowledge of learned activities from multiple source physical spaces, e.g. home A and B, to a target physical space, e.g. home C. Our method called Multi Home Transfer Learning (MHTL) is based on a location mining method for target activity discovery, a semi-Em framework for activity mapping, and an ensemble method for label assignment. In this paper we introduce the MHTL methodology. To validate our algorithms, we use the data collected in several smart apartments with different physical layouts.

Categories and Subject Descriptors

H.2.8 [**Information Systems**]: DATABASE MANAGEMENT—*Data mining*; I.2.6 [**Computing Methodologies**]: ARTIFICIAL INTELLIGENCE—*Learning*

General Terms

Activity Discovery, Transfer Learning, Smart Environments

1. INTRODUCTION

With remarkable recent progress in computing power, networking, and sensor technology, we are steadily moving into the world of ubiquitous computing where technology recedes into the background of our lives. Using sensor technology combined with the power of data mining and machine learning, many researchers are now working on smart environments which can discover and recognize residents' activities and respond to the needs of the residents in a context aware way [5]. Some of these efforts have been demonstrated in actual physical testbeds such as the CASAS project [21], the MavHome project [6], the Gator Tech Smart House [12], the iDorm [9], and the Georgia Tech Aware Home [1]. A smart environment typically contains many sensors such as motion sensors that provide us with unprecedented opportunities for health monitoring, automation, energy efficiency and security via activity discovery and recognition [26]. For example researchers are recognizing that smart environments can assist with valuable functions in the area of remote health monitoring and intervention by monitoring the daily activities of elderly adults with memory deficiencies and helping them via timely prompts [23].

In response to this recognized need, researchers have designed a variety of approaches for discovering, modeling and recognizing activities. Those methods exploit naive Bayes [2], decision trees [16], Markov models [15], dynamic Bayes networks [13], conditional random fields [18], frequent sequence mining [10] and mixed frequent-periodic sequence mining [20]. The problem with all those approaches is that they do not exploit the knowledge learned in previous spaces in order to discover and recognize activities in a new space. Therefore, for each new space a huge amount of data needs to be collected in order to carry out the conventional unsupervised activity discovery methods such as frequent or periodic data mining methods. Even if supervised methods are used, a greater burden is placed on the user of the smart environment, who must annotate sufficient data in order to train the recognition algorithms. Our testbeds have required at least one hour of an expert's time to annotate a single day's worth of sensor data. This particularly becomes problematic if we are targeting a deployment in the home of an older adult. Also, learning the model of each environment separately and ignoring what has been learned in other physical settings leads to redundant computational effort, excessive time investment, and loss of beneficial information that can improve the recognition accuracy. Therefore it is beneficial to develop models that can exploit the knowledge of learned activities by employing them in new spaces. This transfer concept results in reducing the need for data collection, reducing or eliminating the need for data annotation and besides achieving an accelerated learning pace. Using multiple sources and fusing their data together can leverage this learning process even more by using a more diverse set of

SensorKDD'10 July 25, 2010, Washington, DC, USA.

activity models that can help in discovering and recognizing the target activities.

The process of exploiting the knowledge gained in one problem and applying the learned knowledge to a different but related problem is called transfer learning [4][19]. It is a hallmark of human intelligence, and has been vastly studied in the literature [17], but it has been applied to activity discovery and recognition in very few cases.

Our goal is to transfer the knowledge of learned activities from multiple physical source spaces, e.g. home A and B, to a target physical space, e.g. home C. Previously we have shown a method for transferring learned activities from one resident to another [22]. Zhang et al. [27] have developed a model for mapping different types of activities to each other (e.g. sweeping to cleaning) by learning a similarity function via a Web search. Kasteren et. al [25] describe a simple method for transferring the transition probabilities of Markov models for two different spaces. They only transfer the transition probabilities, and most other activity features such as the activity's structure and related temporal features is ignored, as they assume the structure of HMMs is given and pre-defined.

In our approach, the activity model includes much more information based on using structural, temporal and spatial features of the activities. Also, unlike the approach of Kasteren et al.[25], we do not manually map the sensor networks. Instead, we learn sensor mappings based on the available data and activity models. It should be noted that in order to exploit the knowledge learned in different spaces, we transfer the activities from multiple physical source spaces to a target physical space. First we use a location based data mining method to find target activities in the target data. Then the activities from both source and target spaces are represented in a canonical form called an "activity template" in order to allow for a more efficient mapping process. Next we use a semi-EM framework to map source activities from each single source to the target activities. Finally by using an ensemble learning method based on a weighted majority voting [8] and fusing multiple data sources we assign activity labels to the target activities.

The remainder of the paper is organized as follows. First we describe our approach in more detail, including its three main stages. The first stage discovers activities by mining data and extracting activity models, while the second stage maps source activities to the target activities, and the third stage assigns labels to the target activities. We then show the results of our experiments on data obtained from five different smart apartments. Finally we end the paper with our conclusions and discussion of future work.

2. MODEL DESCRIPTION

Our objective is to develop a method that can transfer learned activities across different physical spaces. We assume that labeled activity data is available in the source space $\mathbb{S}$ consisting of N individual sources $S_1, ..S_N$, and limited unlabeled data is available in the target space T. Our goal is to use the source space knowledge to learn the activity labels in the target space where the physical aspects of the space and the sensors may be different. We assume that the nature of the problem is "inductive transfer learning" or "self taught" [17], i.e. we have labeled data in the source domain, and none or few data labels are available in the target domain. This allows us to reduce several weeks or months of data collection and annotation in the target space to only a few days' worth of data collection. We also assume that the number of available sources (N) is limited and computationally manageable, as reducing the number of sources and source selection is outside the scope of this paper. Our ultimate objective is to be able to correctly recognize the activities in the target space. By using our method, labeled target activity data becomes available that can be consumed by conventional learning algorithms to perform activity recognition, or it can be used as a baseline for other techniques such as active learning techniques. In the remainder of this section we describe our notations and also we will provide a high level description of the algorithm.

Timestamp (ts)	**Sensor ID** (s)	**Label** (l)
7/17/2009 09:52:25	$M004$	Personal Hygiene
7/17/2009 09:56:55	$M030$	Personal Hygiene
7/17/2009 14:12:20	$M015$	None

Table 1: Example sensor data. Here $M004$, $M030$ and $M015$ denote sensor IDs.

The input data is a sequence of sensor events e in the form $e = \langle ts, s, l\rangle$ where ts denotes a timestamp, s denotes a sensor ID, and l is the activity label, if available. An example showing several sensor events can be seen in Table 1. As depicted in Table 1, each sensor event can be part of a labeled activity such as the first and second sensor events, or it can have no activity labels such as the third sensor event. Each sensor ID is associated with its room name (e.g. kitchen) which we will refer to as a location tag L. A standard set of location tags is used across all different sources. We define an activity as $a = \langle \mathcal{E}, l, t, d, \mathcal{L}\rangle$ where $\mathcal{E}$ is a sequence of n sensor events $\langle e_1, e_2, ..e_n\rangle$, l is its label (if available), t and d are the start time and duration of the activity, and $\mathcal{L}$ represents the set of location tags where a has occurred.

We denote the set of activities in each individual source space S_k as $\mathcal{A}_{S_k}$. The set of all source activities is denoted by $\mathcal{A}_{\mathbb{S}}$ which is the union of activities from all individual sources, i.e. $\mathcal{A}_{\mathbb{S}} = \bigcup_k \mathcal{A}_{S_k}$. The set of target activities is denoted by $\mathcal{A}_T$. The set of source sensors and the set of target sensors is denoted by $\mathcal{S}_{\mathbb{S}}$ and $\mathcal{S}_T$. In order to be able to map activities from the source space to a target space, we need to find a way to map the source sensor network to the target sensor network i.e. we're looking for the mapping $\mathcal{F}'(\mathcal{S}_{\mathbb{S}}) = \mathcal{S}_T$, as the source sensors will have different locations and properties than the target sensors. Based on using activity features and also the sensor mappings $\mathcal{F}'$, we will find the activity mapping function $\mathcal{F}(\mathcal{A}_{\mathbb{S}}) = \mathcal{A}_T$. Note that for the individual mappings from S_k to T the above mapping functions are written as $\mathcal{F}_k(A_{S_k})$, and $\mathcal{F}'_k(S_{S_k})$.

The extent to which an activity $a_i \in \mathcal{A}_{S_k}$ maps to activity $a_j \in \mathcal{A}_T$ is reflected in matrix M_k, where $M_k[i,j] \in [0..1]$ shows the probability that activity a_i and a_j have the same label. Similarly, a second matrix $m_k[p,q] \in [0..1]$ shows the probability that sensor $s_p \in \mathcal{S}_{S_k}$ maps to sensor $s_q \in \mathcal{S}_T$ based on their location and their role in activity models. Note that the mappings need not to be one to one, due to the differences in the number of sensors and number of activities in the source and target spaces.

Our multi home transfer learning algorithm (MHTL) per-

forms activity discovery and transfer in several stages (see Figure 1). The first step involves processing labeled data from the source space and mining available unlabeled data from the target space in order to extract the activity models in each space. In the source space, for each individual source S_k we extract the activities $\mathcal{A}_{S_k}$ by converting each contiguous sequence of sensor events with the same label to an activity. To reduce the number of activities and to find a canonical mapping, similar activities in $\mathcal{A}_{S_k}$ are consolidated together to represent an "activity template". To avoid mapping irrelevant sensors, a filter feature selection method based on mutual information [11] is used to remove the irrelevant sensors for each activity template. In the target space the data is mined to find unlabeled activity patterns based on using location closure. Target activities are then consolidated using an incremental clustering method [3]. If any labeled data is available in the target space, it can be used to refine the target activity models.

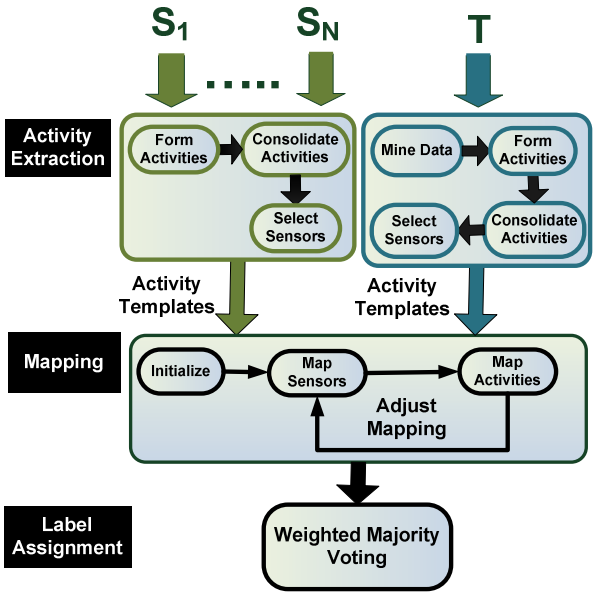


Figure 1: Main components of MHTL for transferring activities from multiple source spaces to a target space.

Next, source activity templates are mapped to the target activity templates. First the activity templates' initial mapping probabilities are computed based on structural, temporal and spatial similarities. The sensors' initial mapping probabilities are assigned based on a spatial similarity measure. After initialization, the algorithm starts an Expectation Maximization like framework [7] called semi-EM in an iterative manner. First, the sensor mapping probabilities are adjusted based on the activity mapping probabilities, next the activity mapping probabilities are adjusted based on the updated sensor mapping probabilities. This continues until no more changes are perceived or until a user defined number of iterations is reached.

Finally, we assign an activity label to each target activity a_j based on the obtained activity mapping probabilities M. To map activity labels we use an ensemble method based on a weighted majority voting. For each space S_k a vote for the label of the target activity a_j is casted. The voted label is selected the same as the label of a source activity a_i that maximizes the mapping probability M_k for a_j. Each vote is weighted by the overall similarity between the source space S_k and the target space T, as will be described later. At the end, the label with the maximum weighted votes is considered as the label of the activity a_j. Note that in this method all the sources contribute to the label mapping process in order to generate a final activity label for each target activity. We provide a more detailed description of these three steps in the following discussion.

2.1 Activity Extraction

The first step of the multi-stage MHTL algorithm is to extract the activity models from input data in both source and target spaces. For each single source space S_k we convert each contiguous sequence of sensor events with the same label to an activity a. This results in finding the set of activities $\mathcal{A}_{S_k}$ for each one of the individual source spaces S_k. The start time of the activity is the timestamp of its first sensor event, while its duration is the difference between its last and first timestamps. Due to the prohibitively large number of extracted activities and possible similarity among them, we combine similar activities together as an "activity template". Representing a set of similar activities as an activity template allows for a more efficient canonical mapping from source to target, as only a few activity templates will be mapped from source to target instead of mapping a large number of similar activities with only minor differences. The activity template for a set of activities is itself an activity, formed by merging activities' sensors, durations, and start times where the merged start times and durations form a mixture normal distribution. The temporal mixture model allows us to capture and model variations of the same activity that occur at different times. For example consider the "eating" activity which usually happens three times a day, once in the morning as breakfast, once at noon as lunch, and once at night as dinner. Using a mixture model for the start time we are able to capture all the three variations by using a single activity model. During activity consolidation, all the source activities that have the same label will be merged into one single activity template. Note that as each activity template is itself an activity, we use the terms activity and activity template interchangeably.

The next step after similar activities are consolidated is to perform sensor selection for each activity template a by preserving only relevant sensors. Performing sensor selection on each activity template allows for even a more compact representation and a more accurate mapping, as it allows us to map only the relevant sensors and to avoid mapping the irrelevant sensors as noise. Our sensor selection method is a filter feature selection method based on mutual information [11]. For each activity template a and each sensor s we define their mutual information $MI(s, a)$ as in Equation 1. This value measures their mutual dependence and shows how relevant sensor s is in predicting the activity's label. Here $P(s, a)$ is the joint probability distribution of s and a, while $P(s)$ and $P(a)$ are the marginal probability distributions, all computed from the sensor and activity occurrences in the data. A high mutual information value indicates the sensor is relevant for the activity template. We simply consider sensors with a mutual information above the midpoint (0.5) as relevant, otherwise they will be discarded.

$$MI(s,a) = P(s,a) * \log \frac{P(s,a)}{P(s)P(a)} \quad (1)$$

To find activity patterns in unlabeled target data, we perform a data mining step on the input data. First we partition the input data into activities. A sensor event $e_1 = \langle ts_1, s_1, l_1 \rangle$ and a successor sensor event $e_2 = \langle ts_2, s_2, l_2 \rangle$ are part of the same activity if $L_{s_1} = L_{s_2}$, i.e. if both sensors are in the same location. Such a local partitioning allows us to have a baseline for finding individual activities. This approach is based on the intuition that occurrences of the same activity usually happen within the same location (such as preparing meal in the kitchen, grooming in the bathroom, etc), and more complex activities occurring in different locations can be composed of those basic activities. Notice that as we only have access to limited input data (perhaps a few days or even a few hours), we cannot use conventional activity discovery methods such as frequent or periodic sequence mining methods [20] to find activity patterns in the data. Therefore exploiting the spatial closure can be a way to overcome this problem. After partitioning data into the initial activities, we consolidate those activities by grouping together similar activities into an activity template. To combine activities together, we use an incremental clustering method [3], such that each activity is assigned to the most similar centroid if their similarity is above threshold ς, and then the centroid is recomputed. Otherwise the activity forms a separate cluster. The centroid is itself represented as an activity template. At the end all the activities in one cluster are consolidated together and the sensor selection is carried out. For two activities a_i and a_j, their similarity $\Upsilon(i,j)$ is defined as in Equation 2.

$$\Upsilon(i,j) = \Upsilon_t[i,j] + \Upsilon_d[i,j] + \Upsilon_{\mathcal{L}}[i,j] + \Upsilon_S[i,j] \quad (2)$$

In above equation, Υ_t refers to start time mapping (if the two activities happen at similar times, e.g. both around noon), Υ_d refers to duration mapping (if the two activities have similar durations), $\Upsilon_{\mathcal{L}}$ refers to location mapping (if the two activities happen in similar locations, e.g. both in the kitchen), and Υ_S refers to structure mapping (if the two activities have similar structure in terms of sensors). We normalize $\Upsilon(i,j)$ to fall within the range [0..1]. For simplicity, we have chosen the mappings to have equal effects, however it's possible to define $\Upsilon(i,j)$ as a weighted average.

As mentioned, the start times are in form of a mixture normal distribution with means $\Theta = \langle \theta_1 .. \theta_r \rangle$. We represent start time θ in an angular form Φ measured in radians instead of a linear representation. This allows for time differences to be represented correctly (2:00 am will be closer to 12:00 pm than to 5:00 am). The similarity between the two start time distributions is thus calculated using Equation 3.

$$\Upsilon_t[i,j] = \max_{\substack{\theta_1 \in \Theta_i \\ \theta_2 \in \Theta_j}} (1 - \frac{|\Phi_{\theta_2} - \Phi_{\theta_1}|}{2\pi}) \quad (3)$$

Duration mapping is calculated as in Equation 4 where durations are in form of a mixture normal distribution with means $\Gamma = \langle \gamma_1 .. \gamma_r \rangle$.

$$\Upsilon_d[i,j] = \max_{\substack{\gamma_1 \in \Gamma_i \\ \gamma_2 \in \Gamma_j}} (1 - \frac{|\gamma_2 - \gamma_1|}{\max(\gamma_2, \gamma_1)}) \quad (4)$$

To compute $\Upsilon_{\mathcal{L}}$ we use Equation 5 which is the Jaccard similarity coefficient [24] for the sets of locations of the two activities. A similar Jaccard similarity coefficient based on similar sensors is defined for the structure mapping Υ_S in Equation 6.

$$\Upsilon_{\mathcal{L}}[i,j] = \frac{|\ \mathcal{L}_i \bigcap \mathcal{L}_j\ |}{|\ \mathcal{L}_i \bigcup \mathcal{L}_j\ |} \quad (5)$$

$$\Upsilon_S[i,j] = \frac{|\ \mathcal{E}_i \bigcap \mathcal{E}_j\ |}{|\ \mathcal{E}_i \bigcup \mathcal{E}_j\ |} \quad (6)$$

2.2 Mapping Activities

The next step after the activity models for the source and target space have been identified is to map the source activity templates to the target activity template. First we initialize the sensor and activity mapping matrixes, m_k and M_k for each pair of source S_k and target T. The initial values of the sensor mapping matrix $m_k[p,q]$ for two sensors $s_p \in S_k$ and $s_q \in T$ is defined as 1.0 if they have the same location tag, and as 0 if they have different location tags. The initial value of $M_k[i,j]$ for two activities $a_i \in \mathcal{A}_{S_k}$ and $a_j \in \mathcal{A}_T$ is obtained based on exploiting related spatial and temporal information and also prior activity label information (if available), as in Equation 7. Note that in Equation 7 the first case applies to the few labeled target activities, while for the majority of the target activities the second case is applied.

$$M_k[i,j] = \begin{cases} 1.0 & if \quad l_i = l_j \\ \Upsilon(i,j) & otherwise \end{cases} \quad (7)$$

For computing subsequent mapping probabilities, we use an Expectation Maximization (EM) like framework [7] by estimating the mapping probabilities in an iterative manner. First, the sensor mapping probabilities are computed; and in the next step the activity mapping probabilities are maximized based on the sensor probabilities. Though this model doesn't exactly reflect an EM algorithm, however due to its iterative manner and likelihood estimation in two steps, we refer to it as a semi-EM framework.

To compute sensor mapping probabilities $m_k[p,q]$ for sensors $s_p \in \mathcal{S}_{S_k}$ and $s_q \in \mathcal{S}_T$, we rely on activities in which s_p and s_q appear in, as in Equation 8. The learning rate α refers to how fast we want to converge on the new values, while $m_k^n[p,q]$ and $m_k^{n+1}[p,q]$ refer to the current and updated values of $m_k[p,q]$ in iteration n and $n+1$, respectively.

$$m_k^{n+1}[p,q] = m_k^n[p,q] - \alpha * \Delta m_k[p,q] \quad (8)$$

$$\Delta m_k[p,q] = m_k^n[p,q] - \frac{1}{|X_p||Y_q|} \sum_{a_i \in X_p} \sum_{a_j \in Y_q} M_k[i,j] \quad (9)$$

$$\begin{aligned} X_p &= \{a_i \in \mathcal{A}_{S_k} | s_p \in \mathcal{E}_i\} \\ Y_q &= \{a_j \in \mathcal{A}_T | s_q \in \mathcal{E}_j\} \end{aligned} \quad (10)$$

In Equation 9, X_p and Y_q give us all the activities in which sensors p and q appear. This means that those activities which do not include a given sensor will not contribute to that sensor's mapping probability.

In the next step, to adjust the mapping probability between each two activities, we use Equation 11 to account for the updated sensor mappings. Here $M_k^n[i,j]$ and $M_k^{n+1}[i,j]$ refer to the current and updated values of $M_k[i,j]$ in iteration n and $n+1$, respectively.

$$M_k^{n+1}[i,j] = M_k^n[i,j] - \alpha * \Delta M_k[i,j] \tag{11}$$

$$\Delta M_k[i,j] = M_k^n[i,j] - \frac{1}{|\mathcal{E}_i|} \sum_{s_p \in \mathcal{E}_i} \max_{\substack{q \\ s_q \in \mathcal{E}_j}} m_k[p,q] \tag{12}$$

The above procedure for computing sensor mapping and activity mapping probabilities is repeated until no more changes are perceived or until a pre-defined number of iterations is reached. Next, the labels are assigned to the target activities based on the obtained probability mapping matrices.

2.3 Mapping Labels

In order to assign labels to the target activities, we use a voting ensemble method [8] based on the activity models $\mathcal{A}_{S_k}$ for each space S_k. Combining data from different sources to improve the accuracy and to have access to complimentary information is known as data fusion or as a form of ensemble learning [14]. Ensemble learning is a strategic way to combine multiple models, such as different classifiers or hypotheses to solve a computational problem. In our problem, using multiple sources allows us to fuse data from different sources and to form different activity models, therefore being able to map target activities based on a more diverse set of source activities. In order to be able to successfully apply the ensemble learning technique, an ensemble system needs classifiers whose decision boundaries are adequately different from each other. The most popular method is to use different training datasets to train individual classifiers. The diversity condition of ensemble learning in our problem is achieved by using different training sets from N different physical source spaces, resulting in N different hypotheses. We build a classifier based on each individual hypothesis h_k and then by combining the predicted labels of all classifiers for a certain target activity we are able to make a decision about the activity's final label.

Each hypothesis h_k is constructed based on using the activity templates $\mathcal{A}_{S_k}$ for space S_k plus the activity and sensor mapping probabilities M_k and m_k. We represent each hypothesis as $h_k = \{\mathcal{F}_k, \mathcal{F}'_k\}$ where $\mathcal{F}$ and $\mathcal{F}'$ denote the activity and sensor mapping functions. For a single space S_k, Equations 13, 14 and 15 provide us with the activity mapping function $\mathcal{F}$, sensor mapping function $\mathcal{F}'$ and the assigned label l_{a_j} for each activity $a_j \in T$. As can be seen in Equation 15, the target activity's label is selected to be the same as the label of a source activity $a_i \in S_k$ that maximizes the mapping probability M_k for a_j.

$$\mathcal{F}_k(a_i) = \max_{a_j}(M_k[i,j]) \tag{13}$$

$$\mathcal{F}'_k(s_p) = \max_{s_q}(m_k[p,q]) \tag{14}$$

$$l_{a_j} = l_{a_i} \quad s.t. \quad M_k[i,j] = \max_z(M_k[z,j]) \tag{15}$$

In order to combine the assigned labels for each a_j using different hypotheses, we use the weighted majority voting algorithm as in Figure 2.3. The input of this algorithm is the source activities $\mathcal{A}_{\mathbb{S}}$, the activity mapping probabilities M, the sensor mapping probabilities m, and activity a_j. The output of the algorithm is the label of a_j as l_{a_j}. For each source space S_k we find the label of a_j by using Equation 15. Each predicted label l is associated with a weight $W[l]$, which is the total similarity between the source S_k and T. The total similarity between S_k and T is calculated as in Equation 16 by summing over the best mapping from S_k to T for each $a_i \in S_k$. Obviously a label can be voted for by different hypotheses and its weight will be increased as a result.

Data: $\mathcal{A}_{\mathbb{S}}$, M, m, a_j
Result: l_{a_j}
// The voted labels and their weights
Λ, W
foreach $S_k \in \mathbb{S}$ **do**
  $l = l_{a_i} \quad s.t. \quad M_k[i,j] = \max_z(M_k[z,j])$
  add l to Λ as $\Lambda[l]$
  $\mathrm{Sim}(S_k, T) = \sum_{a_x \in \mathcal{A}_{S_k}} M_k[x, \mathcal{F}_k(a_x)]$
  $W[l] = W[l] + \frac{Sim(S_k,T)}{|\mathcal{A}_{S_k}|}$
end
$l_{a_j} = \max_l W[l]$
return a_j

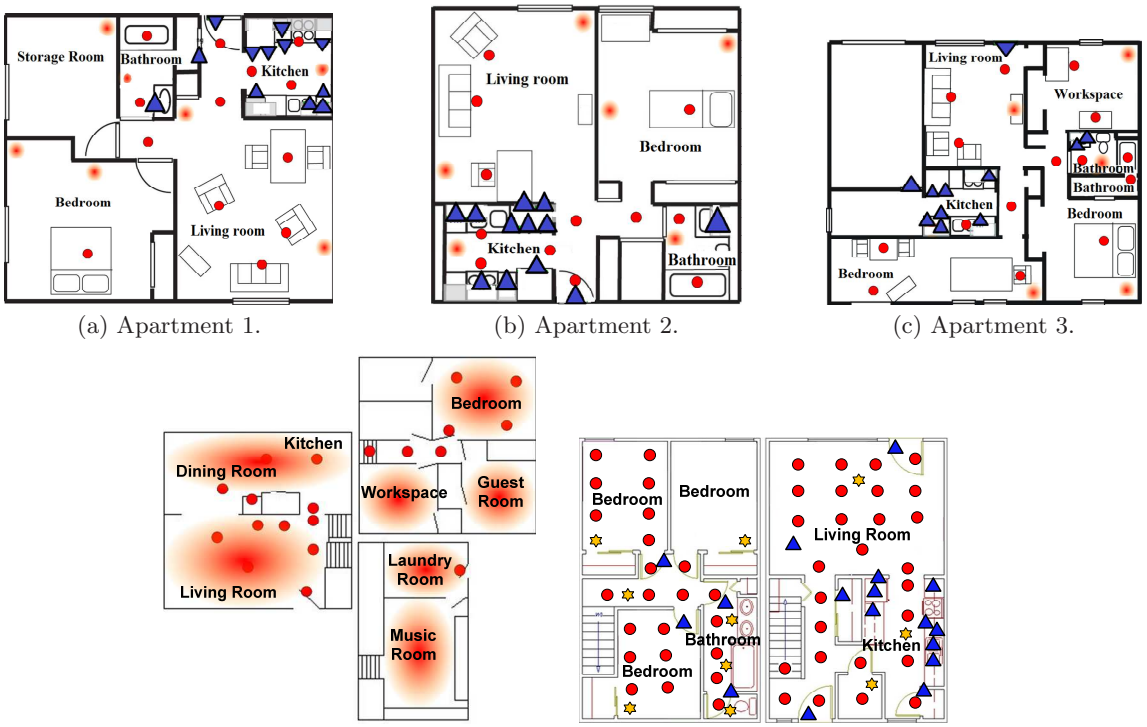
Figure 2: The weighted majority voting schema for label assignment.

$$Sim(S_k, T) = \sum_{a_x \in \mathcal{A}_{S_k}} M_k[x, \mathcal{F}_k(a_x)] \tag{16}$$

At the end, the label with the greatest number of weighted votes is considered as the label of the activity a_j. After obtaining the labels of all target activities, we can use the obtained labels to train a conventional activity recognition algorithm. We also can use the labels in conjunction with other techniques such as active learning in order to further improve the results.

3. EXPERIMENTS

We evaluated the performance of our MHTL algorithm using the data collected from five different smart apartments. The layout of the apartments including sensor placement and location tags are shown in Figure 3. We will refer to apartments in Figures 3(a) through 3(e) as apartments 1 to 5. The data was collected during a three month period for apartments 1, 2, and 3, and during a two month period for apartments 4 and 5. Each apartment is equipped with motion sensors, and most of the apartments are also equipped with contact sensors which monitor the open/closed status of doors and cabinets. Apartment 5 is also equipped with light sensors and some item sensors to sense the presence of key items. As can be seen in Figure 3 the apartments have different layouts. For example, apartments 3 and 4 have two bedrooms, while apartments 1 and 2 have one bedroom. In addition, some functional spaces might not be available in



(a) Apartment 1. (b) Apartment 2. (c) Apartment 3.

(d) Apartment 4. (e) Apartment 5.

Figure 3: Figures (a-e) show sensor map and location tags for each apartment. On the map, circles show motion sensors while triangles show switch contact sensors. The hollow-shaped motion sensors (as in Figure d) are the area motion sensor. The stars in Figure (e) show the light sensors.

all five apartments, such as the workspace, laundry room or the music room.

The residents also have quite different schedules, as can be seen from the activity distribution diagrams shown in Figure 4. For example, in the first apartment housekeeping is performed each Friday, while in the second apartment this is performed once a month, and in the third apartment the housekeeping activity is replaced by a work activity. Also the activity level in each apartment is different, as can be seen clearly by comparing activity distribution diagrams for apartment 4 versus other apartments. The activity level is dependent on the activity level of the residents as well as the number of sensors that monitor the activities. The three first apartments were single resident apartments, while for the fourth apartment the residents included a man, a woman, and a cat. The fifth apartment included two undergraduate student residents. All the data was collected while residents were performing their normal daily activities during a 2-3 month period.

Each of the datasets was annotated with activities of interest for the corresponding resident and apartment. A total of 11 activities were noted for apartments 1, 2 and 3. Those activities included bathing, bed-toilet transition, eating, enter home, housekeeping (for the third apartment this is replaced by "work"), leave home, meal preparation, personal hygiene, sleeping in bed, sleeping not in bed (relaxing) and taking medicine. For the fourth apartment, 7 activities were noted including bed-toilet transition, taking medicine, eating, leaving home, laundry, sleeping in bed and working. The fifth apartment included 7 activities as working, sleeping in bed, bed-toilet transition, personal hygiene, meal preparation, housekeeping, sleeping not in bed (relaxing).

We ran our algorithm for each one of the apartments as the target space, resulting in five different transfer learning problems. In each setting, all the apartments except for the target apartment were used as the source apartments. In each setting, we used all the available source labeled data, 1 to 7 days of target unlabeled data, and 0 to 1 days of target labeled data.

The first step, activity extraction, resulted in a considerable reduction in the number of source activities. In particular 3384, 2602, 1032, 428, and 492 activity instances from the first, second, third, fourth and fifth apartments were represented by as few as 11, 10, 9, 7, and 7 activity templates. The reason that we have obtained less templates than the 11 predefined activities in the second and third apartment is that the "eating" activity was done rather in an erratic way and in different locations, therefore our sensor selection algorithm didn't choose any specific sensor for that activity, and as a result the activity was eliminated. The same applied for "taking medicines" in third apartment. This shows how our algorithm can avoid mapping very irregular activities. It also shows how the algorithm condensed the activity instances into a compressed representation, as we approximately obtained the 11 predefined activities for the first three apartments and exactly 7 activities for the last two apartments. During activity extraction, also the number of sensors for each activity template was reduced from an average of 69.32 sensors to 3.73 sensors, as the algorithm removed the irrelevant sensors and preserved only the relevant sensors. This shows that for each activity a few key sensors can be used to identify the activity, e.g. taking medicine can be identified by the cabinet sensor where the medicines are kept.

In the target space, data was mined to extract the activity templates. For example, using three days of unlabeled target data and no labeled target data, we discovered 8, 7, 7, 5 and 5 activity templates for apartments 1 through 5. The similarity threshold ς in those experiments was set to the midpoint 0.5. The reason that fewer activity templates are discovered compared to the predefined activities, is because some similar activities might be merged into one activity, such as relaxing and eating which happen at similar times and similar places. In addition, some activities cannot be easily discovered based only on a few days of data. One example is the housekeeping activity which happens quite rarely compared to other activities; and even if it happens to be in the data, because of its erratic nature and occurring all over the home, it is not very easy to discover

In the next step the source activities were mapped to the target activities. In order to be able to evaluate the mapping accuracy of our algorithm, we embedded the actual labels of target activities in data. This label is not used during training, rather it's only used at the end to verify the correctness of the results. Mapping accuracy is defined as the number of activities in the target space whose transferred label matches the correct expert-supplied label. Figure 6 shows the mapping accuracy for different amounts of unlabeled target data and no labeled target data, in several different settings. Figure 6 also shows a comparison between mapping accuracy based on using multiple sources vs. average mapping accuracy using a single source, based on using 3 days of unlabeled target data. The mapping accuracies vary from space to space, depending on the consistency of activities in target space, as well as the similarity between the source and target spaces. It should be noted that some activities might not be present in all spaces, such as working or housekeeping. The same applies for lack of certain spaces in different apartments, such as laundry room or workspace. We noted that transfer between apartments that have a more similar layout and functional structure is more satisfactory.

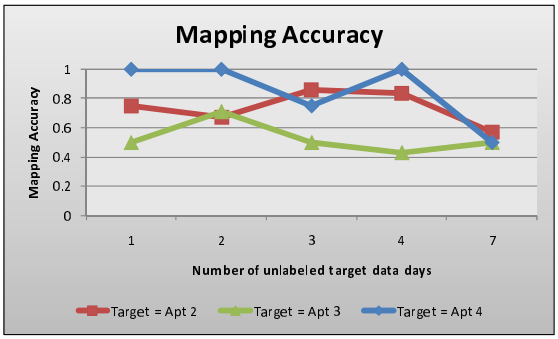
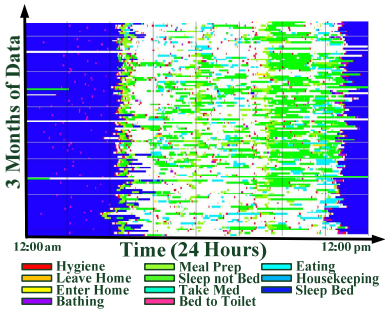
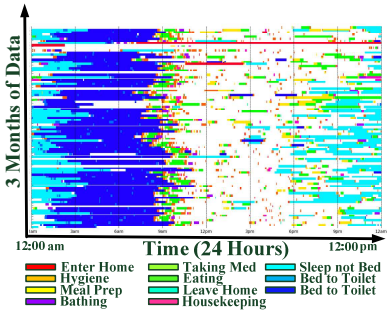


Figure 5: Mapping accuracy in several different settings.

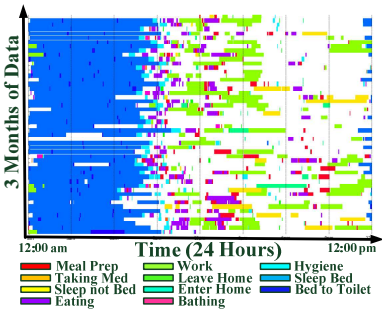
We tested two of our own activity recognition algorithms on the transferred labeled data. The first algorithm is a nearest neighborhood (1NN) algorithm based on the similarity measure in Equation 2. The second algorithm represents activities and sensor events with a hidden Markov model and learns the activities using the Viterbi algorithm. The models performed almost the same with the nearest neigh-



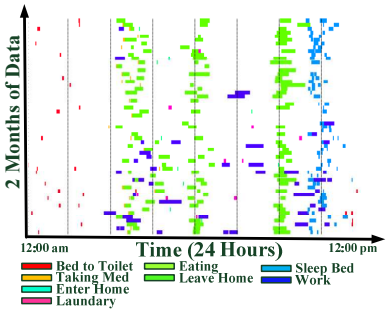
(a) Apartment 1.



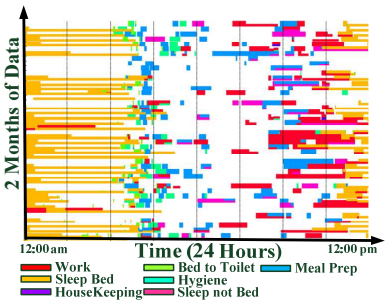
(b) Apartment 2.



(c) Apartment 3.



(d) Apartment 4.



(e) Apartment 5.

Figure 4: Figures (a-e) show residents' activity distribution per 24 hour (horizontal axis) for 2-3 of month data (vertical axis).

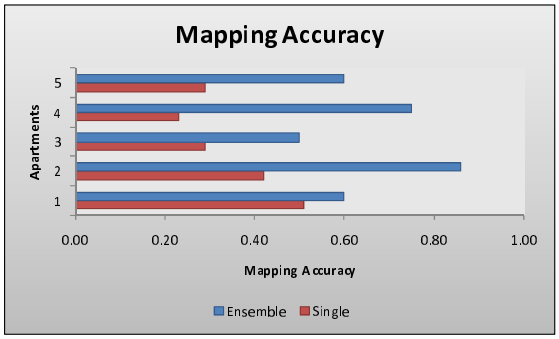


Figure 6: Mapping accuracy in several different settings.

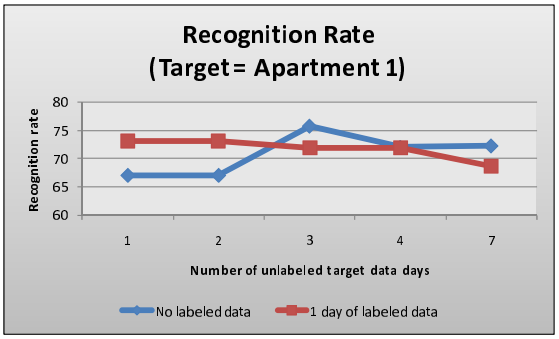


Figure 7: Nearest neighborhood's recognition rate based on mapping to apartment 1 using 0 and 1 day of labeled target data.

borhood algorithm sometimes slightly outperforming HMM due to its use of temporal and spatial features. Using the embedded labels we define the recognition rate as the percentage of sensor events predicted with the correct label. Figure 7 shows 1NN's recognition rate for apartment 1 as the target apartment using 0 and 1 day of labeled target data. Figure 8 shows both 1NN and HMM recognition rate for apartment 1 as the target apartment. Our results show that despite using little to no labeled target data, and having different layouts, schedules and different activities, both algorithms still perform recognition well in a target space using data from a source space.

4. CONCLUSIONS AND FUTURE WORK

This paper introduces a method of transferring learned activities from several different physical spaces to a target physical space. Transferring activities to a target space allows us to avoid the time consuming task of data annotation for each new physical space and to achieve a more accelerated deployment process. Using data from multiple source spaces allows us to be able to map from a more diverse set of activities to a target space by using an ensemble method. Our experiment results show that it is possible to recognize activities using no labeled data from the target space, and despite the fact that the apartment layouts and residents schedules are different. In the future, we intend to combine this method with adaptive and active learning methods in

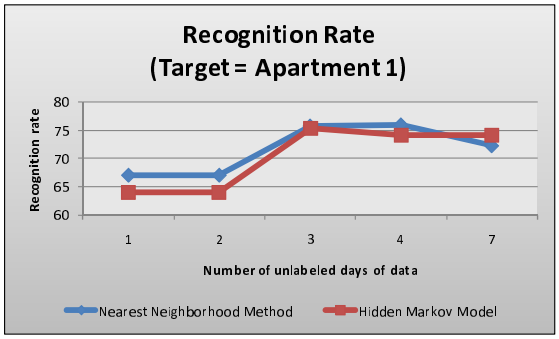


Figure 8: Recognition rate based on mapping to apartment 1 for nearest neighborhood and HMM.

order to be able to enhance the results over time. We also want to develop algorithms that can map activities from environments with totally different functionalities, such as from a workplace to a residential space. We also intend to find methods for selecting the best subset of physical source spaces among a large set of source spaces, in order to improve the system's efficiency.

5. ACKNOWLEDGEMENT

The authors would like to thank Brian Thomas for developing the visualizer software and making available the activity distribution diagrams.

6. REFERENCES

[1] G. Abowd and E. Mynatt. *Smart Environments: Technology, Protocols, and Applications*, chapter Designing for the human experience in smart environments., pages 153–174. Wiley, 2004.

[2] O. Brdiczka, J. Maisonnasse, and P. Reignier. Automatic detection of interaction groups. In *Proceedings of the 7th international conference on Multimodal interfaces*, pages 32–36, 2005.

[3] F. Can. Incremental clustering for dynamic information processing. *ACM Transactions on Information Systems*, 11(2):143–164, 1993.

[4] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[5] D. Cook and S. Das. *Smart Environments: Technology, Protocols and Applications*. Wiley Series on Parallel and Distributed Computing. Wiley-Interscience, 2004.

[6] D. Cook, M. Youngblood, I. Heierman, E.O., K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja. Mavhome: an agent-based smart home. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, pages 521–524, March 2003.

[7] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *The Royal Statistical Society*, 39(1):1–38, 1977.

[8] T. G. Dietterich. Ensemble methods in machine learning. In *MCS '00: Proceedings of the First International Workshop on Multiple Classifier Systems*, pages 1–15, 2000.

[9] F. Doctor, H. Hagras, and V. Callaghan. A fuzzy embedded agent-based approach for realizing ambient intelligence in intelligent inhabited environments. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 35(1):55–65, Jan. 2005.

[10] T. Gu, Z. Wu, X. Tao, H. Pung, , and J. Lu. epsicar: An emerging patterns based approach to sequential, interleaved and concurrent activity recognition. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communication*, 2009.

[11] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Machine Learning Research*, 3:1157–1182, 2003.

[12] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen. The gator tech smart house: A programmable pervasive space. *Computer*, 38(3):50–60, 2005.

[13] T. Inomata, F. Naya, N. Kuwahara, F. Hattori, and K. Kogure. Activity recognition from interactions with objects using dynamic bayesian network. In *Casemans '09: Proceedings of the 3rd ACM International Workshop on Context-Awareness for Self-Managing Systems*, pages 39–42, 2009.

[14] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[15] L. Liao, D. Fox, and H. Kautz. Location-based activity recognition using relational markov networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 773–778, 2005.

[16] U. Maurer, A. Smailagic, D. P. Siewiorek, and M. Deisher. Activity recognition and monitoring using multiple sensors on different body positions. In *BSN '06: Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks*, pages 113–116, 2006.

[17] S. J. Pan and Q. Yang. A survey on transfer learning. Technical Report HKUST-CS08-08, Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China, November 2008.

[18] M. Philipose, K. Fishkin, M. Perkowitz, D. Patterson, D. Fox, H. Kautz, and D. Hahnel. Inferring activities from interactions with objects. *IEEE Pervasive Computing*, 3(4):50–57, Oct.-Dec. 2004.

[19] R. Raina, A. Y. Ng, and D. Koller. Constructing informative priors using transfer learning. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 713–720, 2006.

[20] P. Rashidi and D. J. Cook. An adaptive sensor mining framework for pervasive computing applications. In *International Workshop on Knowledge Discovery from Sensor Data(Sensor-KDD 2008)*, pages 41–49, 2008.

[21] P. Rashidi and D. J. Cook. the resident in the loop: Adapting the smart home to the user. *IEEE Transactions on Systems, Man, and Cybernetics journal, Part A*, 39(5):949–959, September 2009.

[22] P. Rashidi and D. J. Cook. Transferring learned activities in smart environments. In *5th International Conference on Intelligent Environments*, volume 2 of

Ambient Intelligence and Smart Environments, pages 185–192, 2009.

[23] V. Rialle, C. Ollivet, C. Guigui, and C. Hervé. What do family caregivers of alzheimer's disease patients desire in smart home technologies? contrasted results of a wide survey. *Methods of Information in Medicine*, 47:63–9, 2008.

[24] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Adison Wesley, 2005.

[25] T. van Kasteren, G. Englebienne, and B. Krose. Recognizing activities in multiple contexts using transfer learning. In *AAAI AI in Eldercare Symposium*, 2008.

[26] C. Wren and E. Munguia-Tapia. Toward scalable activity recognition for sensor networks. In *Proceedings of the Workshop on Location and Context-Awareness*, pages 218–235, 2006.

[27] V. W. Zheng, D. H. Hu, and Q. Yang. Cross-domain activity recognition. In *Ubicomp '09: Proceedings of the 11th international conference on Ubiquitous computing*, pages 61–70, 2009.

Using Semantic Annotation for Knowledge Extraction from Geographically Distributed and Heterogeneous Sensor Data

Alexandra Moraru, Carolina Fortuna, Dunja Mladenić
Jozef Stefan Institute
39 Jamova, 1000 Ljubljana
Slovenia
+38614773144
firstname.lastname@ijs.si

ABSTRACT
Using semantic technologies for enriching sensor data description in scalable and heterogeneous sensor network are intended as a solution for better interoperability and easier maintainability. Through semantic annotations it is possible to provide context for sensor networks, which will improve knowledge extractions from sensor data streams and will facilitate reasoning capabilities. We propose an architecture for a system able to automatically annotate sensors descriptions, as provided by the publishers, with semantic concepts. The annotated sensor data become more meaningful and machine understandable, enabling better analysis and processing from heterogeneous streams of data. Based on the system proposed, we provide illustrative examples for demonstrating the improvements that semantic context brings and we discuss a real-world scenario of Participatory Sensing.

Categories and Subject Descriptors
I.2.4 **[Artificial Inteligence]**: Knowledge Representation Formalisms and Methods – *Semantic networks*

H.3.4 **[Information and Storage Retrieval]**: Systems and Software - *Question-answering (fact retrieval) systems, User profiles and alert services*

General Terms
Algorithms, Experimentation, Human Factors.

Keywords
Sensor Web, semantic annotations, real-world data, Participatory Sensing, Semantic Sensor Web, Knowledge Extraction, reasoning.

1. INTRODUCTION
The development of the Internet towards a network of interconnected objects, ranging from cars and transportation cargos to electrical appliances to any type of sensing devices, is leading to the Internet of Things. This development will provide new services and will enable new kinds of communications (i.e. "things-to-persons" or "thing-to-thing"). Furthermore, Internet of Things relies on scalable networks, mobility of wirelessly connected objects and offering interoperability for heterogeneous and complex networks [1].

Sensor Webs (or Networks) play a major role in the development of Internet of Things. In the Open Geospatial Consortium (OGC) acceptance, a Sensor Web represents a "web accessible sensor networks and archived sensor data that can be discovered and accessed using standard protocols and application program interfaces" [2], while a sensor network interconnects only sensor devices in a computer accessible network with the intention of monitoring and recording conditions at diverse location. One of the important characteristics of the Sensor Web is that its components share and use the information gathered [3].

Derived from Sensor Webs, the concept of Participatory Sensing as defined by the authors of [4], is "data collection and interpretation" and it includes mainly mobile devices that can be used to build a sensor network for capturing and sharing local data. The range of applications for this field can vary from urban planning, to public health or to natural resource management. Although the common devices that are considered to be used in Participatory Sensing are the mobile phones, the field is open to other types of sensing objects. For instance, Pachube[1], one of the existing platforms for storing and sharing real-time sensor data, is enabling people to interact with sensors, from physical or virtual environments, which are connected to the internet. It registers data for over 3700[2] sensor nodes, with over 9400 data streams, varying from temperature, to air quality monitoring, to power consumption or to users' Skype status. Similar projects are SensorMap[3] and Sesorpedia[4] also aiming at providing a "social-network" for sensors.

The principle of Participatory Sensing brings the advantage of providing access to various types of data which can be used in monitoring, studying and analyzing large scale natural and artificial systems. However, it relies on the involvement of participants or community groups for documenting the data they send. Due to the large and diverse communities that are participating at building such networks, problems may appear in

SensorKDD'10, July 25, 2010, Washington, DC, USA.

[1] http://www.pachube.com/
[2] reported on 30 April 2010
[3] http://atom.research.microsoft.com/sensewebv3/sensormap/
[4] http://www.sensorpedia.com/

searching and finding sensors published by different participants. These problems are caused by using different vocabularies in describing the sensors and by the large and increasing number of heterogeneous sensors. Therefore, extracting knowledge from sensor description for understanding the data that it sends can be difficult, but it is also very important for maximizing the power of participatory sensing.

The documentation provided in a natural language by the participants who publish their data is not enough for a machine to understand it. A solution for improving knowledge extraction from sensor data streams is to provide semantic context. Enriching sensor description with semantic concepts leads to the development of Semantic Sensor Web (SSW), increasing interoperability and enabling complex reasoning with the contextual knowledge resulted from the semantic concepts [5].

In this paper, we propose a system architecture for semantically annotating sensor descriptions with concepts from an ontology, in order to offer a common vocabulary and a representation model which will enable better sensor discovery and will provide reasoning capabilities. Afterwards, we demonstrate with illustrative examples how the semantic context can help in complex searching of sensors and how reasoning can be applied for inferring new knowledge from the sensor descriptions. We also show the results of annotating descriptions of sensors from a real-world sensor web with semantic concepts and we discuss what improvements are required on the semantic level, without changing the design of the sensor web. To the best of our knowledge this is the first effort of annotating such a large amount of sensor data streams.

The rest of the paper is organized as follows. In Section 2 we present related work. Section 3 discusses the technologies used in semantic annotation of sensor webs and describes the system architecture that we suggest for building the SSW. Section 4 outlines the case study of participatory sensing on the Pachube platform, while conclusions and future work are included in Section 5.

2. RELATED WORK

Previous works in the Sensor Web domain have proposed and discussed different possibilities of combining Sensor Web and semantic technologies [6][7][8][9]. Through illustrative examples they explained the advantages that semantics would bring, how the resulted SSW would ease the path on using sensor data in different studies and how it will enable better communication between parties involved in building and maintaining a heterogeneous Sensor Web. In this paper we try to apply similar scenarios on a real Sensor Web, from a collaborative environment of over 3700 sensor nodes.

In [6], the authors discuss the design of the SSW, suggesting existing data on the semantic web to be used for annotations. They present some illustrative examples for using Linked Data for annotating sensor data, exploiting the data already published. The assumption is that one is annotating sensor data directly with semantic concepts, which implies that all publishers will have to adopt a common ontology and annotate their sensor data with concepts from that ontology. An example of reasoning on semantically annotated sensor data is given, using DBpedia geographical data and then formulating a complex query in SPARQL. Our work differs in the sense that we assume that the publisher can describe the sensor data using simple tags or natural language and, afterwards, we automatically annotate those descriptions with semantic concepts.

The problem of geographical information retrieval is approached in [7]. The authors propose the use of semantic rules for adding additional processing capabilities for ontologies represented in Web Ontology Language (OWL). These rules will overtake the lack of mathematical calculus of OWL and will enable context-aware geographical information retrieval. To demonstrate the applicability of semantic rules in solving the problem, the authors developed an application ontology for the scenario of finding surf spots with respect to the users preferences. One of their achievements is that of integrating numerical data from Sensor Web with nominal data for Semantic Web, while in our work we propose a system that already incorporates this type of integration.

3. SEMANTIC ANNOTATION FOR SENSOR DATA

Extending sensor webs with semantics implies finding a suitable representation of the the afferent knowledge in such a way as to enable interoperability and reasoning mechanisms. One of the advantages that semantic technologies bring in knowledge representation are better scalability and interoperability, since adding or changing new information to a set of programs that use the same model resumes at changing the external model, while the design of those programs can remain the same, without the need of human involvement [10].

The complexity of SSW technology is derived both from the semantic and the sensor network point of view. Ontologies used in knowledge representation play a key role in usefulness of combing semantics with sensor networks. Depending on how general is the knowledge represented by an ontology they can be categorized in domain ontologies and upper ontologies. The first category represents models of specific domains (e.g. sensors) and the particular meaning of concepts related to that domain, while the second category is used to model general concepts applicable on a large set of domain ontologies. The authors of [7] are mentioning about an even more specified type of ontologies, referred to as application ontologies which "specify the conceptualization that underlie specific applications".

Three of the existing sensor network ontologies developed are presented in [11][12][13] and have a set of common concepts related to the taxonomy of different types of sensors, physical properties of sensor devices, data acquisition and sensed domain. However, the features of the sensed domain may vary depending on the application where the sensor network is used and further development of these set of concepts is required. A detailed survey of semantic specification of sensor networks is provided in [14], where eleven sensor network ontologies are analyzed.

Such ontologies can be used for semantic annotation of sensor descriptions. Figure 1 presents the system architecture that we propose for building the SSW. We start form a sensor web composed of heterogeneous sensors which are described by their publishers. The sensor descriptions provide information about data streams, such as the type of measurements that the sensor performs (e.g. temperature, humidity, power consumption, etc.) or its physical location.

Further, there are two assumptions on which we base the rest of the system:

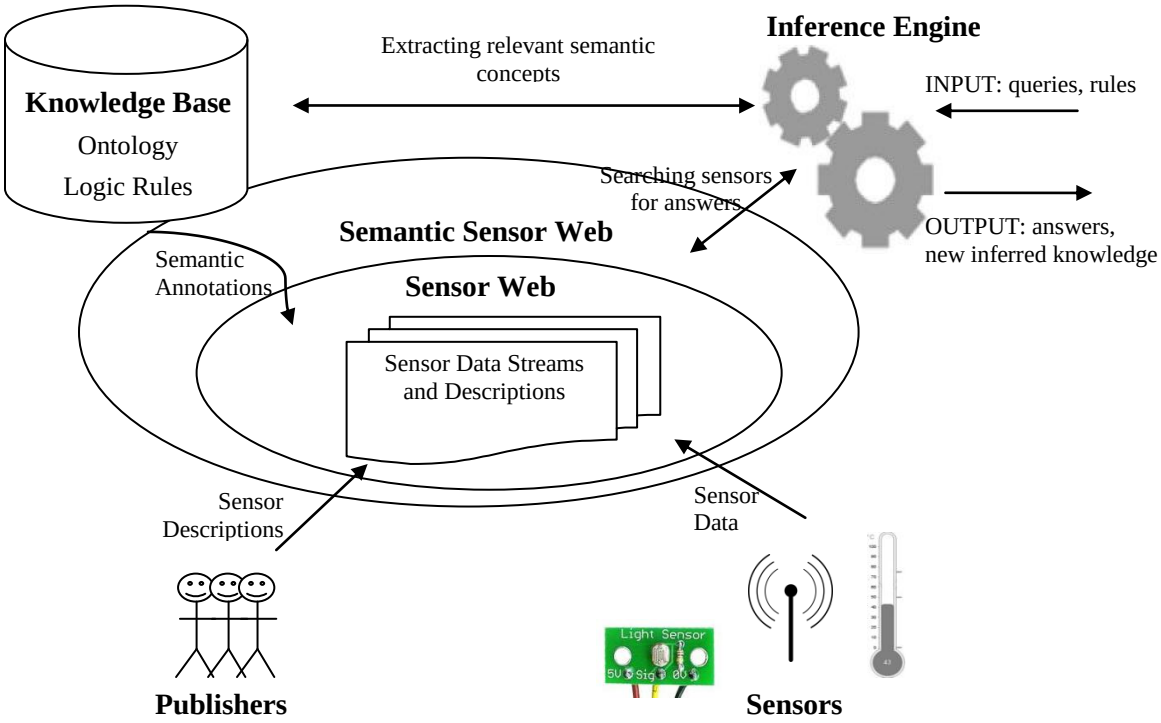


Figure 1. Suggested System Architecture

- the publishers provide at least a tag word for describing sensor measurements and/or location;
- the ontology concepts used for annotation are provided with a description, more exactly a string term is associated.

The next component is represented by an ontology that contains the sensor web concepts needed for annotation. These concepts are used to automatically annotate the sensor descriptions, based on the string terms associated. For demonstration purposes we utilize the Cyc [15] technology for this component and we describe it in detail in Section 3.1. Apart from the ontology, there are also the logic rules that are applied on the ontology relationships and concepts, and together with the ontology they form a knowledge base. Applying the semantic concepts on the sensor web leads to the SSW which will provide more meaningful descriptions of sensors.

The last component that we mention for our system is the inference engine, which plays an important role. The annotated sensor descriptions can be understood by such inference engines and used for solving complex queries and deriving new information. A query formulated by a user will be solved by the inference engine, which will first look into the knowledge base for the information needed for processing the query and then it will be able to search the sensors or data streams that are requested by that query, based on their annotations.

3.1 Cyc

Cyc [15] is an artificial intelligence project that aims at building a general ontology and a knowledge base for representing common sense knowledge. The Cyc technology components that present interest in this work are the knowledge base, the representation language (CycL) and the inference engine. The Cyc knowledge base is a formalized representation of fundamental human knowledge: facts, rules, and heuristics for reasoning about the objects and events of everyday life. Cyc's knowledge is represented in CycL, while its inference engine performs general logical deduction. One of the advantages that Cyc is bringing is the very broad knowledge base covering common sense knowledge, as well as domain specific knowledge for a number of domains, which can support description of the domain of sensing for various sensor networks and also provide context for different applications. Another advantage is that of the specialized inference engine which performs modular search in the proof space enabling reasoning at large scale.

The Cyc knowledge base is organized into "microtheories", which are focused on providing context for particular domains at different level of details or different time intervals. The microtheory structure allows Cyc to independently maintain knowledge which can be contradictory for particular domains, enabling also a better performance of the system, by giving the possibility of controlling the inference process. In our work, we used the Cyc ontology without bringing any major modifications, except for introducing some simple predicates meant for illustration purposes. However, Cyc knowledge base can be modified and extended to meet the requirements of very specific domains, like the sensor web. In future work we take into consideration creating a specific microtheory that will provide the semantic context needed for building SSW.

In the rest of the paper we use CycL for formulating rules and queries, as it is an intuitive language that can be easily understood. A detailed description of CycL is beyond the scope of this paper, and it can be found in [16]. Note mentioning that Cyc concepts are represented with the #$ symbols as prefix.

3.2 Reasoning with Sensor Data

One of the most important arguments for semantically annotating sensor data is that of providing a support for performing reasoning on top of it. This will enable the possibility of applying logic rules through which new information can be inferred from the data available.

```
(implies
 (and
    (isa ?SENSOR Sensor)
    (sensorMeasurmentsInterval ?SENSOR ?INT)
    (temporalBoundsContain ?SEASON ?INT)
    (isa ?SEASON SummerSeason)
    (hasRegionLocation ?SENSOR ?REGION)
    (hasClimateType ?REGION MediterraneanClimateCycle)
    (hasExposure ?SENSOR Outdoor)
    (hasDataStream ?SENSOR ?DS)
    (measures ?DS Temperature)
    (valueOf ?DS (DegreeCelsius ?C))
    (lessThan ?C 10))
 (anomalousMeasurments ?SENSOR ?DS))
```

Figure 2 CycL rule for detection of anomalous measurements

An example of such new information is regarding detection of anomalous data measurements. For instance, let's assume the following scenario:

- a large number of data stream measuring temperature in a Mediterranean region are available for summer time;
- for proper analysis of data we want to eliminate any anomalous measurements, which could have been caused by devices malfunctions or transmission errors.

Considering that there is summer season, any temperature measurements below a minimal value (e.g. 10 ℃ for our illustrative example) are considered anomalous for an outdoor exposure of the sensing device. The representation of such a rule in CycL can be represented as depicted in Figure 2.

A rule in CycL, begins with *#$implies* and has two parts, called its antecedent and consequent, or left-hand side and right-hand side. In our example the antecedent part is represented by all the conditions that make a data stream measurement into an anomaly. The consequent part is the assertion of a predicated which identifies the anomaly. The representation of the logic formula in natural language is:

- For every sensor S, which is performing measurements in the temporal bounds of summer season and is located in region with a Mediterranean climate, with an outdoor exposure and a data stream measuring temperature, if the measured temperature is less than 10 ℃ then, the sensor S is sending anomalous measurements.

When executing such a rule, inference is used also for detecting the geographical region of the sensor location based on the geographical coordinates. Similar rules can be applied for other type of measurements, such us humidity, light. Detecting an anomaly is it important not only for eliminating corrupted data before any further analysis, but also for detecting alarms, depending on the context of the problem.

4. CASE STUDY: PARTICIPATORY SENSING

An example scenario covered by the proposed system architecture is that of Participatory Sensing, which relies on publishers' descriptions of sensors, so that the data shared can be used by others. An example of a sensor web that applies the principles of Participatory Sensing is Pachube.

```
<environment updated="2010-05-01T15:16:55"id="6777" >
  <title> SunSPOT</title>
  <feed>http://www.pachube.com/api/feeds/6777.xml</feed>
  <status>live</status>
  <location domain="physical" exposure="indoor">
     <name>WSN SensorLab</name>
     <lat>46.0425085163033</lat>
     <lon>14.4882792234421</lon>
  </location>
  <data id="0">
   <tag>Temperature</tag>
   <value minValue="18.5" maxValue="38.75"> 33.25
            </value>
   <unit type="basicSI" symbol="°C">Celsius</unit>
  </data>
  <data id="1">
   <tag>Light</tag>
   <value minValue="0.0" maxValue="727.0">723</value>
   <unit type="basicSI" symbol="%">Percent</unit>
  </data>
</environment>
```

Figure 3. XML description of a sensor node on Pachube

Pachube is a platform that supports storing and sharing sensor data with the aim of contributing to the building of Internet of Things. It offers support for remote environments interactions, as well as structured metadata describing the sensor data streams.

The publisher can send their data from a sensor node for storage, making it available to other users. Each sensor node has a unique id and can send more data streams from different sensor devices. When registering a sensor node on Pachube the publisher can also provide a description of the data sensed which can include information about the location of the sensor (latitude and longitude), exposure of the sensor node (indoor or outdoor), tags and units of measurements for a data stream. The descriptions are in XML format and besides the publisher data they also contain automatically generated data, such as timestamp of the last update, the current status of the sensor node (live or frozen) or minimum and maximum values for a specific stream. An example o a sensor description is presented in Figure 3.

The sensor node described in Figure 3 has the title " SunSpot" and it has an URL address through with it can be accessed. Further, at the time of the last update (stated in the "environment" node, the "updated" attribute) the status of the sensor node was "live" meaning that it was sending data. The location node gives details about sensor location. From the description we can understand that the node is located indoor, at specified latitude and longitude coordinates. The description about the data streams are given in

the "data" node. Since a sensor node can send more data streams, each one has an id, a tag describing its measurements, last value sensed, minimum and maximum values and unit of measurements.

Table 1. Frequent tags for data streams descriptions in Pachube

Domain	Tag	Number of occurrences
Temperature related tags	temperature	336
	Temperature	42
	temp	32
	celsius	293
Power consumption related tags	electricity	389
	power	34
	watts	34
No description	null	1437
	Distinct tags	**Data streams**
Total	2238	9466

A particular aspect of these descriptions that captured our attention is the tags used for describing a data stream. These tags are introduced by the sensor publisher and they play a major role in sensor discovery, as one would use the tags when searching for a specific type of sensors. We have analyzed over 3700 sensor description which provide over 9400 data streams. The most frequent tags are presented in Table 1. It can be observed that for a single domain, like temperature monitoring, there can be several different tags that describe the data streams. This can bring difficulties in sensor discovery since a simple search by tag will not reveal all the sensors that one may be interested in.

4.1 Using Semantics for Pachube Sensor Descriptions

An important aspect of Participatory Sensing is the description that the user provides for the data sent. In general, when tagging the data sent there is no common vocabulary that the participants use. Therefore, processing these tags for extracting knowledge is required. In our approach we used Cyc ontology for finding corresponding concepts for sensor tags.

Table 2 Cyc Concepts for Sensor tags

Tag	Cyc Concept	Cyc Related Strings
temperature	Fever	Temperature
temp	TemporaryWorker	Employee
celsius	DegreeCelsius	Celsius
electricity	Electricity	Electrical power
power	powerRating	Power ratings
watts	Watt	W, Watt

The advantage of the Cyc ontology is that it provides string terms for concepts. For instance, for the concept of #$Temperature one of the strings associated, using the termStrings predicate is "temperature" (termStrings Temperature "temperature"). This predicate can be used to find the associated concept for a string term. However, due to the very large number of concepts from very different domains that Cyc includes, problems may appear when trying to find a concept for less explicit strings.

To illustrate this we searched into to the Cyc knowledge base for concepts associate to a set of sensor tags. The results are represented in Table 2, showing that a too broad ontology can introduce noise when associating concepts to strings. For instance, for the "temp" tag, which is used to represent a temperature sensor, the concept found in Cyc (#$TemporaryWorker) is not even related to what we were looking for. Furthermore, even for an explicit tag, like "temperature" the first concept returned for associating the string with is #$Fever and the #$Temperature concept is found only in the related concepts. However, for other tags, such as "celsius" or "electricity", it is possible to correctly annotate them with ontology concepts.

The results that we obtained for annotating sensor tags with semantic concepts require improvements for real-world scenarios. One way to achieve these improvements could be obtained by introducing context when searching the concept related to those tags. The context can be created with ontologies specialized for sensor networks or by using microtheories in Cyc. Providing a context for concept searching will reduce the number of irrelevant concepts and will provide a better categorization of sensor types of measurements.

Individual: IJSSensor
isa: Sensor
hasDataStream: IJSSensor-Data1
hasDomain: Physical
hasExposure: Indoor
latitude: (Degree-UnitOfAngularMeasure 46.0425085163033)
longitude: (Degree-UnitOfAngularMeasure 14.4882792234421)

Individual: IJSSensor-Data1
isa: DataStream
hasUnitOfMeasurement: DegreeCelsius
measures: Temperature

Figure 4 Representation of a Sensor Description in Cyc

The tags used in describing data streams are important in determining what type of measurements a sensor sends (e.g temperature, light intensity, power consumption). Nevertheless, other important features of a sensor can be found in the XML description. For instance, the geographical location of the sensor, if it is in an indoor or outdoor environment or if it sends data from a physical or virtual environment, are important features that one should take into consideration when interested in using these data streams. An illustrative example of representing a sensor node in Cyc ontology is depicted in Figure 4, which corresponds to the XML description from Figure 3. The sensor node is represented as an individual of the #$Sensor collection and it sends a data stream with temperature measurements. Further, the geographical location in terms of latitude and longitude are represented, along with few other characteristics. We would like to mention that most of the concepts already exist in the Cyc ontology (such as #$Sensor, #$DataStream, #$latitude, #$longitude, #$Temperature), while we also introduced some basic predicates

for illustrating other features that are relevant (#$hasDataStream, #$hasDomain, #$hasExposure).

4.2 Searching for Sensors

Having sensor descriptions annotated with semantic concepts can improve search capabilities. Assuming that context is provided for semantic concepts search, all sensors tagged with "temperature", "temp" or "celsius" will be annotated with the #$Temperature concept. From here, retrieving all the sensors that measure temperature is a trivial task. However, more complex queries could be stated when having semantically annotated sensors.

QUERY: (What are the sensors that measure temperature in Ljubljana?)

```
(and
  (isa ?X Sensor)
  (hasDataStream ?X ?DS)
  (measures ?DS Temperature)
   (distanceBetween ?X CityOfLjubljanaSlovenia
      (Kilometer ?DIST))
   (lessThan ?DIST 10))
```

ANSWERS:

?X	?DS	?DIST
CityCenterSensor	CityCenterSensor-Data1	3.2271206217234605
IJSSensor	IJSSensor-Data1	2.4810075343716043

Figure 5 Example of a query and its answers in Cyc

An example of a complex query derives from the possibility of inferring geographical regions from sensors' location. For instance, the distance between a sensor node and a specific location can be calculated based on geographical coordinates. This means that new type of queries can be solved, such as: "Which are the sensors that measure temperature in Ljubljana?". Such a question can be solved in Cyc by using the #$distanceBetween predicate, which can calculate the distance between two locations based on their latitude and longitude coordinates. The advantage of having access to a large knowledge base that incorporates common sense knowledge is highlighted in this example, where we already have represented the concepts of cities, also with details about their location. Then, we can assume that by "sensor located in Ljubljana" it is meant a fixed distance between city coordinates and sensor coordinates. For instance, we can consider that an area of 10 kilometers from the city coordinates is in the city region. An illustration of this query and the answer provided is represented in Figure 5.

After solving a query, Cyc also provides explanation of the inference performed. For the second answer of our query example, we can observe the answer bindings for each of the query variables. These bindings are resulted from simple default assertions (such as: #$isa #$IJSSensor #$Sensor) and from complex logic rules (such as the rule for computing the distance between two locations). An illustration of the summary of answer computed bindings is represented in Figure 6 or proving the validity of the inference. We show simplified explanation of the results, because the total number of steps that the Cyc inference engine performed for solving this query is 78, and a detailed explanation is out of the scope of this paper.

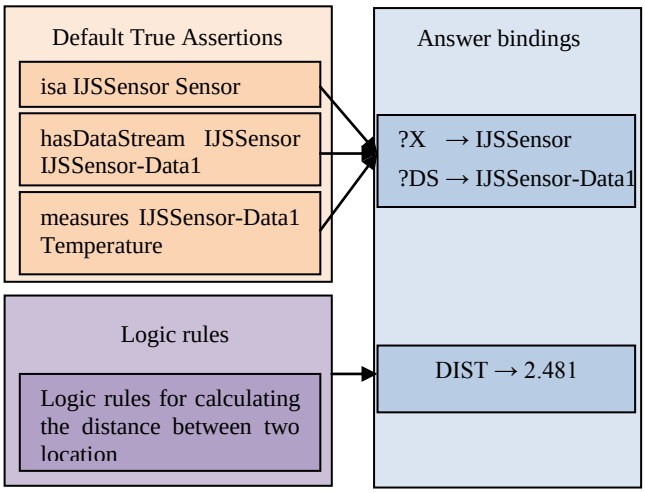


Figure 6 Answer bindings

With the current[5] setting of the Pachube platform, these types of queries are impossible to solve automatically. The only search for sensors is possible by tags or title, which present the disadvantage of not returning all the answers. For location matter they provide a world-map with the location of each sensor, but the user has to manually navigate through that map.

5. CONCLUSIONS AND FUTURE WORK

A solution for enriching sensor data streams for providing machine understandable meaning is represented by the semantic technologies. Semantic annotations can provide context for the sensor measurements and observations, transforming data streams from simple binary models into meaningful data, which can be used in further analysis. In this paper we proposed and discussed a system architecture that is able to automatically annotate, with semantic concepts, sensor description provided by publishers. We demonstrated through illustrative examples the advantages of applying reasoning mechanisms on semantically enriched sensor descriptions.

We also presented the results of applying our system in a real-world scenario, that of Participatory Sensing. One of the main conclusions after analyzing these results is that a too general ontology will not be able to successfully annotate all the descriptions with relevant concepts. This problem can be solved by using a domain specific ontology or creating context in the more general ones. However, the advantages of having access to common sense knowledge have been illustrated in the examples provided in this paper, namely extended knowledge can help in inferring geographical regions or creating more complex rules.

In our future work we plan to provide more specific context for the concepts used in sensor annotation. This will enable more accurate annotation of the sensor description and will perform better in real-world scenarios. In addition, we are considering semantic solutions for sensor composition. The virtual sensor created through composition of sensors will introduce a level of abstractness that can enable better communication between people and sensor networks or between sensor networks themselves. Furthermore, the future work will be conducted using some OWL-

[5] Reported on 30 April 2010

based ontologies in comparison with the Cyc ontology, as well as on considering different sensor description representations, including relational databases and more standardized descriptions.

6. ACKNOWLEDGMENTS

This work was supported by the Slovenian Research Agency and the IST Programme of the European Community under ENVISION - ENVIronmental Services Infrastructures with ONtologies (ICT-2009-249120) and PASCAL2 Network of Excellence (ICT-NoE-2008- 216886).

7. REFERENCES

[1] Commission of the European Communities. 2009. Internet of Things — An action plan for Europe. Communication from the Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the Regions (Brussels, June 18, 2009).

[2] Botts, M., Percivall, G., Reed, C., Davidson, J. 2007. OGC White Paper OGC® Sensor Web Enablement: Overview And High Level Architecture. White Paper. OpenGIS.

[3] Delin,K.A., Jackson S. 2001. The Sensor Web: A New Instrument Concept. Presented at SPIE's Symposium on Integrated Optics (San Jose, CA, January 20-26 2001).

[4] Goldman, J. Shilton, K., Burke, J., Estrin D., Hansen M., Ramanathan N., Reddy S., Samanta, V., Srivastava M. B., West, R. 2009. Participatoy Sensing: A Citizen-powered Approach to Illuminating the Patterns That Shape our World. Foresight & Governance Project, White Paper.

[5] Seth, A., Henson, C., Sahoo, S.S. 2008. Semantic Senor Web. Internet Computing, IEEE, pp 78-83.

[6] Wei, W., Barnaghi, P. 2009. Semantic Annotation and Reasoning for Sensor Data. 4th European Conference on Smart Sensing and Context, EuroSSC 2009, volume 5741 of Lecture Notes in Computer Science, pp. 67-76. Berlin, 2009. Springer.

[7] Keβler, C., Raubal, M., Wosniol, C. Semantic Rules for Context-Aware Geographical Information Retrieval. . 4th European Conference on Smart Sensing and Context, EuroSSC 2009, volume 5741 of Lecture Notes in Computer Science, pp. 77-92. Berlin, 2009. Springer.

[8] Barnaghi, P., Meissner, S., Presser, M., Moessner, K. 2009. Sense and Sens'ability: Semantic Data Modelling for Sensor Networks. In Proceedings of ICT-Mobile Summit 2009.

[9] Janowicz, K., Schade, S., Bröring, A., Keßler, C., Maué, P., Stasch, C. 2010. Semantic Enablement for Spatial Data Infrastructures. Transactions in GIS (TGIS) volume 14, issue 2, 2010, pp. 111–129.

[10] Berners-Lee, T., Hendler, J., Lassila, O. 2001. The semantic web a new form of web content that is meaningful to computers will unleash a revolution of new possibilities. Scientific American, May 2001.

[11] Neuhaus, H., Compton, M. 2009. The Semantic Sensor Network Ontology: A Generic Language to Describe Sensor Assets. In AGILE Workshop: Challenges in Geospatial Data harmonization.

[12] Russomanno, D. et all. 2005. Building a Sensor Ontology: A Practical Approach Leveraging ISO and OGC Models. In Procceding of the International Conference on Artificial Intelligence (Las Vegas, Nevada, 2005).

[13] Clader, M., Morris, R., Peri, F. 2009. Machine reasoning about anomalous sensor data. Ecological Informatics.

[14] Compton, M., Henson, C., Lefort, L., Neuhaus, H., Sheth, A. 2009. A Survey of the Semantic Specification of Sensors, In 2nd International Workshop on Semantic Sensor Networks, at 8th International Semantic Web Conference (October 2009).

[15] Lenat, D.B. 1995. Cyc: A Large-Scale Investment in Knowledge Infrastructure. Comm. of the ACM 38:11.

[16] Cycorp. The Syntax of CycL. Cycorp 1996-2002.

Random Kernel Perceptron on ATTiny2313 Microcontroller

Nemanja Djuric
Department of Computer and Information Sciences, Temple University
Philadelphia, PA 19122, USA
nemanja.djuric@temple.edu

Slobodan Vucetic
Department of Computer and Information Sciences, Temple University
Philadelphia, PA 19122, USA
vucetic@ist.temple.edu

ABSTRACT
Kernel Perceptron is very simple and efficient online classification algorithm. However, it requires increasingly large computational resources with data stream size and is not applicable on large-scale problems or on resource-limited computational devices. In this paper we describe implementation of Kernel Perceptron on ATTiny2313 microcontroller, one of the most primitive computational devices with only 128B of data memory and 2kB of program memory. ATTyny2313 is a representative of devices that are popular in embedded systems and sensor networks due to their low cost and low power consumption. Implementation on microcontrollers is possible thanks to two properties of Kernel Perceptrons: (1) availability of budgeted Kernel Perceptron algorithms that bound the model size, and (2) relatively simple calculations required to perform online learning and provide predictions. Since ATTiny2313 is the fixed-point controller that supports floating-point operations through software which introduces significant computational overhead, we considered implementation of basic Kernel Perceptron operations through fixed-point arithmetic. In this paper, we present a new approach to approximate one of the most used kernel functions, the RBF kernel, on fixed-point microcontrollers. We conducted simulations of the resulting budgeted Kernel Perceptron on several datasets and the results show that accurate Kernel Perceptrons can be trained using ATTiny2313. The success of our implementation opens the doors for implementing powerful online learning algorithms on the most resource-constrained computational devices.

Categories and Subject Descriptors
C.3 [**Computer Systems Organization**]: Special purpose and application-based systems – *Real-time and embedded systems*

General Terms
Algorithms, Performance, Experimentation.

Keywords
Kernel Perceptron, Budget, RBF kernel, Microcontroller.

SensorKDD'10, July 25th, 2010, Washington, DC, USA.

1. INTRODUCTION
Kernel Perceptron is a powerful class of machine learning algorithms which can solve many real-world classification problems. Initially proposed in [6], it was proven to be both accurate and very easy to implement. Kernel Perceptron learns a mapping $f: \mathrm{X} \rightarrow \mathrm{R}$ from a stream of training examples $S = \{(\mathbf{x}_i, y_i), i = 1 \ldots N\}$, where $\mathbf{x}_i \in \mathrm{X}$ is an M-dimensional input vector, called the data point, and $y_i \in \{-1, +1\}$ is a binary variable, called the label. The resulting Kernel Perceptron can be represented as

$$f(\mathbf{x}) = sign(\sum_{i=1}^{N} \alpha_i K(\mathbf{x}, \mathbf{x}_i)), \quad (1)$$

where α_i are weights associated with training examples, and K is the kernel function. The RBF kernel is probably the most popular choice for Kernel Perceptron because it is intuitive and often results in very accurate classifiers. It is defined as

$$K(\mathbf{x}, \mathbf{x}_i) = \exp(-\frac{\| \mathbf{x} - \mathbf{x}_i \|^2}{A}), \quad (2)$$

where $\| \cdot \|$ is the Euclidean distance, and A is the positive number defining the width of the kernel.

Training of the Kernel Perceptron is simple: starting from the zero function $f(\mathbf{x}) = 0$ at time $t = 0$, data points are observed sequentially, and $f(\mathbf{x})$ is updated as $f(\mathbf{x}) \leftarrow f(\mathbf{x}) + \alpha_i \cdot K(\mathbf{x}_i, \mathbf{x})$, where $\alpha_i = y_i$, if point $\mathbf{x}_i$ is misclassified ($y_i \cdot f(\mathbf{x}_i) \leq 0$) and $\alpha_i = 0$ otherwise. Examples for which $\alpha_i = y_i$ have to be stored and they are named the support vectors. Despite the simplicity of this training algorithm, Kernel Perceptrons often achieve impressive classification accuracies on highly non-linear problems. On the other hand, number of the support vectors grows linearly with the number of training examples on noisy classification problems. Therefore, these algorithms require $\mathcal{O}(N)$ memory and $\mathcal{O}(N^2)$ time to learn in a single pass from N examples. Because number of examples N can be extremely high, Kernel Perceptrons can be infeasible on noisy real-world classification problems. This is the reason why budgeted versions of Kernel Perceptron have been proposed with the objective of retaining high accuracy while removing the unlimited memory requirement.

Budget Kernel Perceptron [4] has been proposed to address the problem of the unbounded growth in resource consumption with training data size. The idea is to maintain a constant number of support vectors during the training by removing a single support vector every time the budget is exceeded upon addition of a new

support vector. Given a budget of T support vectors, Budget Kernel Perceptrons achieve constant space scaling $\mathcal{O}(T)$, linear time to train $\mathcal{O}(TN)$ and constant time to predict $\mathcal{O}(T)$. There are several ways one may choose to maintain the budget. The most popular approach in literature is removal of an existing support vector to accommodate the new one. For example Forgetron [5] removes the oldest support vector, Random [3] the randomly selected one, Budget [4] the one farthest from the margin, and Tightest [13] the one that impacts the accuracy the least. There are more advanced and computationally expensive approaches such as merging of two support vectors [12] and projecting a support vector to the remaining ones [9].

Although Budget Kernel Perceptrons are very frugal, requiring constant memory and linear space to train, they are still not applicable to one of the simplest computational devices, microcontrollers. These devices have extremely small memory and computational power. In this paper we consider microcontroller ATTiny2313, which has only 128B of memory available to store the model, maximum processor speed of 8MHz and whose hardware does not support floating-point operations. We propose a modification of Budget Kernel Perceptron that uses only integers and makes it very suitable for use on resource-limited microcontrollers. Our method approximates floating-point Budget Kernel Perceptron operations using fixed-point arithmetic that provides nearly the same accuracy, while allowing much faster execution time and requiring less memory. We use *Random Removal* as budget maintenance method [3, 11]. When the budget is full and new support vector is to be included to the support vector set, one existing support vector is randomly chosen for deletion. Although this update rule is extremely simple, the resulting Perceptron can achieve high accuracy when the allowed budget is sufficiently large. The pseudocode of Budget Kernel Perceptron is given as Algorithm 1.

It should be noted that a significant body of research exists on the topic of efficient hardware implementation of various machine learning and signal processing algorithms. In Compressed Kernel Perceptron [11] the authors consider efficient memory utilization through data quantization, but assume floating-point processor. Several solutions have been proposed for implementation of kernel machines on fixed-point processors as well. In [1], authors propose special hardware suitable for budgeted Kernel Perceptron. Similarly, in [2] authors consider implementation of Support Vector Machines [10] on fixed-point hardware. The two proposed algorithms, however, assume that the classifiers are already trained. Additionally, use of the problem-specific hardware is limited to a single problem. We, on the other hand, implement our method on general-purpose hardware, which makes the implementation much easier and more cost-efficient. Moreover, we combine the two proposed approaches, quantization of data and the use of fixed-point hardware, to obtain an accurate classifier that can be used on the simplest existing computational devices.

The paper is organized as follows. In Section 2 the proposed method is explained in detail. In Section 3 the results are presented and discussed. Finally, Section 4 concludes the paper.

Algorithm 1 - *Budget Kernel Perceptron*

Inputs : *data sequence* $((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N))$, *budget* T
Output : *support vector set* $SV = \{SV_i, i = 1 \ldots I\}$

$I \leftarrow 0$; $i \leftarrow 1$
$SV = \emptyset$
for $i = 1 : N$
{
  if $\left(\left(y_i \cdot \sum_{j=1}^{I} y_j \cdot K(\mathbf{x}_i, \mathbf{x}_j) \right) \le 0 \right)$
  {
    if $(I == T)$
      $new = random(I)$
    else
    {
      $I \leftarrow I + 1$
      $new \leftarrow I$
    }
    $SV_{new} = (x_i, y_i)$
  }
}

2. METHODOLOGY

In this paper we focus on implementation of Kernel Perceptron on the specific microcontroller. We first describe its properties and then explain the proposed algorithm.

2.1 Microcontroller

The microcontroller ATTiny2313 [14] has been chosen as the target platform because its specifications make it extremely challenging to implement a data mining algorithm. It has very limited speed (0-8 MHz, depending on voltage that ranges from 1.8 to 5.5V), low power requirements (when in 1.8V and 1MHz mode, it requires only 300μA and 540μW power), and very limited memory (2kB of program memory that is used to store executable code and constants; 128B of data memory that is used to store model and program variables). Because of its limitations it is also very cheap, with the price of around 1$ [15]. It supports 16-bit integers and supports fixed-point operations. Multiplication and division of integers is very fast, but at the expense of potential overflow problems with multiplication and round-off error with division. ATTiny2313 does not directly support floating-point numbers, but can cope with them using certain software solutions. Floating-point calculations can also be supported through C library, but only after incurring significant overhead in both memory and execution time.

2.2 Attribute quantization

In order to use the limited memory efficiently the data are first normalized and then quantized using B bits, as proposed in [11]. Using quantized data, instead of (1), the predictor is

$$f(\mathbf{x}) = sign(\textstyle\sum_i \alpha_i K(q(\mathbf{x}), q(\mathbf{x}_i))), \quad (3)$$

where $q(\mathbf{x})$ is the quantized data point $\mathbf{x}$. Quantization of data introduces quantization error, which can have significant impact on the classification, especially for the data points that are located near the separating boundary. The quantization loss suffered by the prediction model due to quantization is discussed in much more detail in [11]. Objective of this paper is to approximate (3) using fixed-point arithmetic.

2.3 Fixed-point method for prediction

Our algorithm should be implemented on microcontroller with extremely low memory. Although quantization can save valuable memory space, there are additional memory-related issues that must be considered. First, number of program variables must be minimal. Prudent use of variables allows more space to store the model and leads to increased accuracy. On the computational side, all unnecessary operations, such as excess multiplications and exponentials should be avoided. If this is not taken into consideration, the program can take a lot of memory and a lot of time to execute. Furthermore, due to inability of ATTiny2313 to deal with floating-point numbers in hardware, use of floating-point library results in even larger program size.

Instead of RBF kernel (2) our algorithm uses its modified version, as proposed in [2],

$$K(\mathbf{x},\mathbf{x}_i) = e^{-\frac{|\mathbf{x}-\mathbf{x}_i|}{2^A}}, \quad (4)$$

where several differences from RBF kernel in (2) can be noticed. First, instead of Euclidean distance, Manhattan distance is used. In [2], kernel function with Manhattan distance is used because multiplications can be completely omitted from their algorithm. For our application, we use the Manhattan distance to limit the range of distances between the quantized data, which will lead to improved performance of our algorithm. Furthermore, without the loss of generality, we represent the kernel width as 2^A, where A is any number.

Although the simplifications in equation (4) have been introduced, the algorithm cannot be implemented without the penalty in the speed of computation. This is because very simple devices are not designed to compute exponents or other operations involving floating-point arithmetic efficiently. This results in slow run-time and excessive memory usage. Therefore, an alternative way needs to be devised to perform kernel calculation. The idea of our method is to use only integers, which are handled easily by even the simplest devices, and still manage to use the RBF kernel for predicting the label of new point.

The normalized data point $\mathbf{x}$ is quantized using B bits and its integer representation $I(\mathbf{x})$ is

$$I(\mathbf{x}) = round(\mathbf{x} \cdot 2^B). \quad (5)$$

The value of $I(\mathbf{x})$ is in the range $[0,\ 2^B - 1]$. Since $\mathbf{x}$ is now approximated by $q(\mathbf{x}) = I(\mathbf{x})\, 2^{-B}$, we can define kernel function which takes quantized data

$$K_q(\mathbf{x},\mathbf{x}_i) \overset{\Delta}{=} K(q(\mathbf{x}), q(\mathbf{x}_i)) = e^{-\frac{|I(\mathbf{x})-I(\mathbf{x}_i)|}{2^{A+B}}}. \quad (6)$$

If we represent $|I(\mathbf{x}) - I(\mathbf{x}_i)|$ as d_i, and introduce for the simplicity of notation

$$g = e^{-\frac{1}{2^{A+B}}}, \quad (7)$$

we get

$$K_q(\mathbf{x},\mathbf{x}_i) = g^{d_i}. \quad (8)$$

We need to calculate $sign(\sum_{i=1}^{T} y_i \cdot K_q(\mathbf{x},\mathbf{x}_i))$, where T is the budget.

Since $sign(\sum_{i=1}^{T} y_i K_q(\mathbf{x},\mathbf{x}_i)) = sign(\sum_{i=1}^{T} y_i c K_q(\mathbf{x},\mathbf{x}_i))$ for any $c > 0$, we can replace g^{d_i} from (8) with $Cg^{d_i - d_{nn}}$, where d_{nn} is the distance between newly observed data point $\mathbf{x}$ and the nearest support vector and C is a positive integer. Then, to allow integer representation, we again replace it with integers

$$w(d_i - d_{nn}) \overset{\Delta}{=} round(C \cdot g^{d_i - d_{nn}}), \quad (9)$$

and approximate $K_q(\mathbf{x},\mathbf{x}_i) \approx w(d_i - d_{nn})$. It is clear that if $d_i = d_{nn}$ then $w = C$, if $d_i - d_{nn} = 1$ then $w = round(C\, g)$, and if $d_i - d_{nn}$ is sufficiently large then $w = 0$. We can notice that if the i-th support vector is close to data point $\mathbf{x}$ its weight w and its influence on classification will be large. Our final classifier is approximated as

$$f(\mathbf{x}) = sign(\sum_{i=1}^{T} y_i \cdot w(d_i - d_{nn})), \quad (10)$$

where y_i is the label of i-th support vector, and w is its weight which depends on the distance d_i from data point $\mathbf{x}$. The drawback of the proposed method is that the support vector set needs to be scanned every time a new data point $\mathbf{x}$ arrives in order to find the nearest support vector distance. In addition, there is a computational problem related to calculation of weights. These issues will be addressed in Section 2.4.

2.3.1 Error of weight approximation

By using rounding operation (9) we are inevitably introducing additional error, on top of the error made due to quantization of data point using B bits [11]. In order to understand the effect of the approximation error, let us define the relative error R as

$$R = \frac{|true\_value - approximation|}{true\_value}, \quad (11)$$

where *true_value* is the actual value of the number, and *approximation* is value of the number after rounding. It can be shown [7] that the relative error R we are making by the rounding (9) depends on the parameter C as $R \sim 1/C$. This shows that large C leads to small relative error. However, because the microcontroller supports only 16-bit integers, too large C will render our method useless on the chosen platform.

2.4 Weight calculation

To be able to use predictor (10) there is a need to calculate w_i for every support vector. For given parameters (A, B, C) we can calculate the weight for every possible distance $d = d_i - d_{nn}$ as illustrated in Table 1. In the naïve approach, we can precalculate all the weights and save them as array in microcontroller memory. If the dataset is M-dimensional, then d is in range $[0,\ M(2^B - 1)]$, and storing each weight could become impossible.

Table 1 Distances and corresponding weights

Distance (d)	0	1	...	d	…
Weight ($w(d)$)	Cg^0	Cg^1	...	Cg^d	…

Therefore, we must find a way to represent entire array of weights with only a subset of weights in order to save memory. Two

approaches are discussed next, both requiring very limited memory.

2.4.1 Sequential method

The most obvious way is to store weights $w(0)$ and $w(1)$, for distances 0 and 1, respectively. By storing these two weights we can calculate other weights in Table 1. Every other weight can be iteratively calculated, in integer calculus, as

$$w(d) = \frac{w(d-1)^2}{w(d-2)}. \quad (12)$$

The memory requirement of this approach is minimal, but there are certain problems. In order to calculate particular weight we need to calculate all the weights up to that one by repeatedly applying (12), which can take substantial time. In addition, the division of two integers results in rounding error that would propagate and increase for large d.

2.4.2 Log method

Different idea is to save only the weight of distance 0, which equals C, together with the weights at distances that are the power of 2, namely weights of distances 1, 2, 4, 8, 16 and so on. In this way the number of weights we need to save is logarithm of total number of distances, which is approximately $log(M\,2^B)$. Using the saved weights we calculate weight for any distance using Algorithm 2.

Algorithm 2 - *Find Weight*

Inputs : *array of weights W and corresponding distances D*
distance d for which the weight is being calculated
Output : *weight w*

i ← *Index of the nearest smaller distance to d in array D*
w ← *W*[*i*]
d ← *d* – *D*[*i*]
while (*d* > 0)
{
    i ← *Index of the nearest smaller distance to d in array D*
    w ← *w* · *W*[*i*] / *W*[0]
    d ← *d* – *D*[*i*]
}

As can be seen in Algorithm 2, if we are looking for the weight of the distance d that is the power of 2, the weight will immediately be found in the array W. If not, we are looking for the distance in array D that is the closest smaller value than d, and start the calculation from there. We repeat the process until convergence. In this way, we will always try to make the smallest error while calculating the weight, and the algorithm runs using $\mathcal{O}(log(M\,2^B))$ space and $\mathcal{O}(log(M\,2^B))$ time, which is efficient and fast even for highly-dimensional data and large bit-lengths.

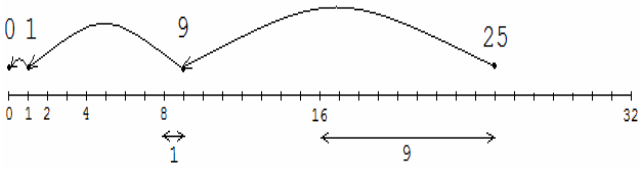


Figure 1 Illustration of *log* method

Log method is illustrated in Figure 1. Assume that we are looking for the weight of distance 25, and we saved weights at distances 0, 1, 2, 4, 8, 16 and 32. It is clear that 25 can be represented as 16 + 8 + 1. In the first step we will use $w(16)$, then $w(8)$, because 8 is the closest to 25−16, and finally $w(1)$. The sought-after weight will be found both very quickly and with minimal error.

3. EXPERIMENTAL RESULTS

We evaluated our algorithm on several benchmark datasets from UCI ML Repository whose properties are summarized in Table 2. The digit dataset *Pendigits*, which was originally multi-class, was converted to binary dataset in the following way: classes representing digits 1, 2, 4, 5, 7 (non-round digits) were converted to negative class, and those representing digits 3, 6, 8, 9, 0 (round digits) to the positive class. *Banana* and *Checkerboard* were originally binary class datasets. *Checkerboard* dataset is shown in Figure 8.

Table 2 Dataset summaries

Dataset	Training set size	Testing set size	Number of attributes
Banana	4800	500	2
Checkerboard	3500	500	2
Pendigits	2998	500	16

In all reported results, A is the kernel width parameter, B is bit-length, and C is positive integer, all defined in Section 2.3.

In Figure 2 the absolute difference between the true weights and the calculated ones using *sequential* and *log* methods for weight calculation described in Sections 2.4.1 and 2.4.2 is shown. It can be seen that *log* method makes very small error with minimal memory overhead over the whole range of distances. We use the *log* method as our default for weight calculation.

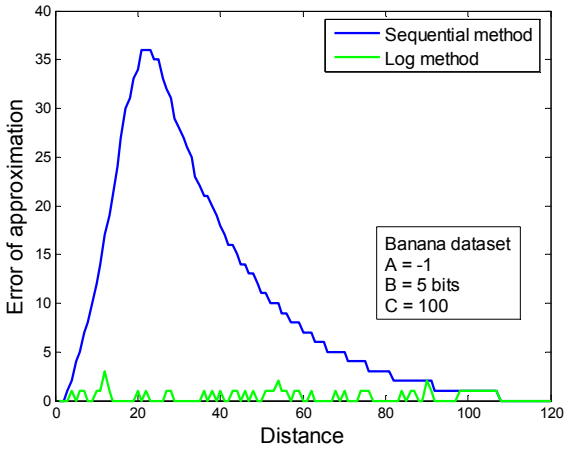


Figure 2 Error of two weight calculation methods

In the next set of experiments the quality of approximation was evaluated. First, the Random Kernel Perceptron was trained using equation (6) that requires floating-point calculations. Then, our fixed-point method was executed, and the predictions of the two algorithms were compared. The approximation accuracy is defined as

$$accuracy = \frac{\sum_{i=1}^{N} I(y_i^{fixed}, y_i^{floating})}{N}, \quad (13)$$

where I is the indicator function that is equal to 1 when the predictions are the same and 0 otherwise, y_i is the classifier prediction, and N is the size of the test set.

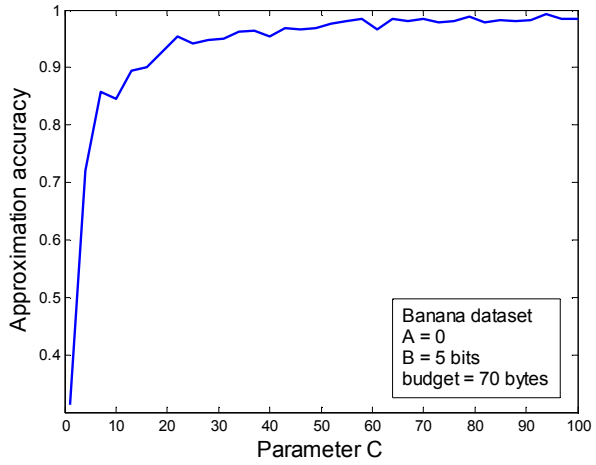


Figure 3 Approximation accuracy as the function of *C*

Figure 3 shows that the approximation accuracy is relatively large even for small values of C and that it quickly increases with C, as expected from discussion in Section 2.3.1. Since ATTiny2313 microcontroller supports 16-bit integers, the maximum value of a number is 65535. However, as can be seen from Algorithm 2, in order to calculate weights multiplications are required, and at no point the product should exceed this maximum value. Therefore, the best choice for C is the square root of 65535, which is 255. In the following experiments C was set to 255.

In Figure 4 the approximation accuracy is given for 3 datasets. Total budget was fixed to only 70 bytes. Each dataset was shuffled before each experiment and the reported results are averages after 10 experiments. The value of parameter A was changed in order to estimate its impact on the performance. High negative values of A correspond to the algorithm that acts like nearest neighbour. High positive values of parameter A correspond to the majority vote algorithm. It can be seen that over large range of A values the approximation accuracy was around 99%, and that is was lower at the extreme values of A. This behavior is due to the numerical limitations of double-precision format used when calculating equation (6). The exponentials of large negative numbers are too close to 0 and are rounded to 0, which results in much smaller accuracy. On the other hand, it can be seen that for large values of A corresponding to the algorithm that acts like majority vote the approximation accuracy starts to decrease. This was expected, because in this case all the weights, as calculated by our method, are practically the same since the weights decrease extremely slowly and very small differences between actual weights are lost when they are calculated using the *log* method. However, it should be noted that in this problem setting kernel width depends on A as 2^A, and the values where prediction accuracy starts to fall are practically never used. Experiments were conducted for different values of parameter B as well. However, the results for other bit-lengths were very similar, and are thus not shown.

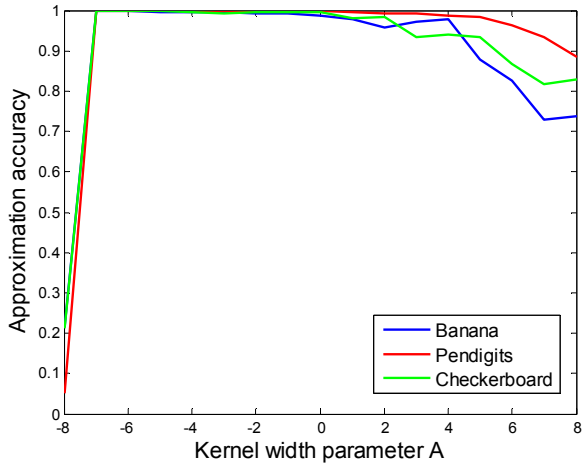


Figure 4 Approximation accuracy (*B* = 5 bits)

The next set of experiments was conducted to evaluate prediction accuracy of the proposed Random Kernel Perceptron implementation. The values of parameters A and B were changed in order to estimate their influence on the accuracy of methods. The results are the averages over 10 experiments and are reported in Figures 5, 6 and 7.

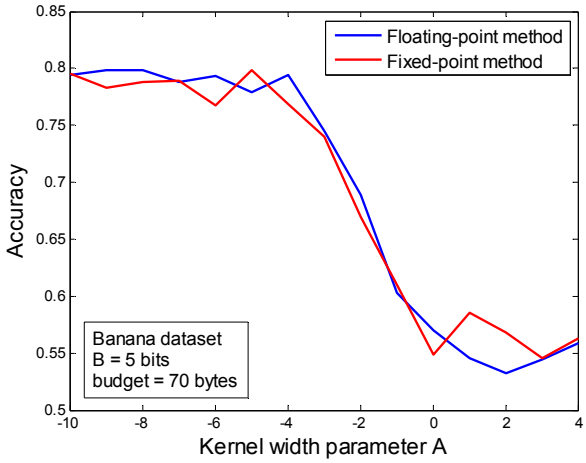


Figure 5 Accuracy of two methods

It can be seen that both floating- and fixed-point implementations achieve essentially the same accuracy, as was expected since the approximation has been proven to be very good. In some cases for very small values of A the accuracy of floating-point method drops sharply, as in Figure 6. This happens because of the numerical problems, where the predictions are so close to 0 that when calculated in double-precision they are rounded to 0. Our method does not have this problem. It is also worth noticing that for large A the accuracy drops sharply for both algorithms, which supports the claim that large kernel widths are usually not useful. It is exactly for these large values of A that the approximation accuracy drops as well. Therefore, as can be seen in Figures 5, 6 and 7, these values are not of practical relevance. Only results for bit-length of 5 bits were shown, since for the other bit-lengths the results were nearly the same.

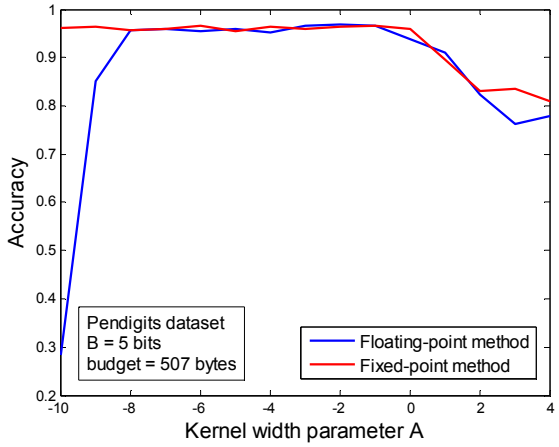


Figure 6 Accuracy of two methods

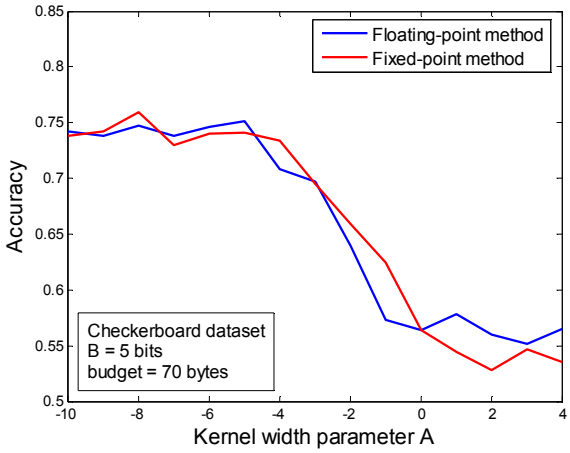


Figure 7 Accuracy of two methods

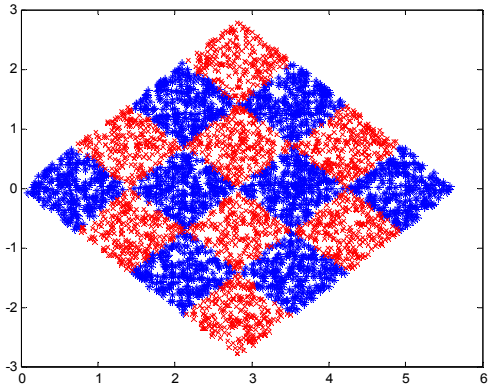


Figure 8 *Checkerboard* dataset

3.1 Hardware implementation

The algorithm was tested on microcontroller ATTiny2313. The program was written in C programming language using free IDE *AVR Studio 4* available for download from Atmel's website.

Different methods were used in order to assess the complexity of implementation on microcontroller. The details are given in Tables 3, 4 and 5. *Program* and *Data* represent the required program and data memory in bytes. *Time*, in milliseconds, is given after 100 observed examples, and kernel width parameter A is set to -6. Accuracy is calculated after all data points were observed. Numbers in the brackets represent the total size of memory block of the microcontroller. The microcontroller speed is set to 4 MHz. Number of support vectors was chosen so that the entire data memory of microcontroller is utilized in the case of fixed-point method. This number of support vectors was then imposed on both methods. Therefore, as the bit-length is increased, the number of support vector decreased.

For Tables 3 and 4 the target microcontroller was ATTiny2313. For Table 5 microcontroller with twice bigger memory was used because of the high dimensionality of *Pendigits* dataset. ATTiny48 was chosen which has 4kB of program memory and 256B of data memory.

It is clear that proposed fixed-point algorithm takes much less memory and is faster than the floating-point algorithm. Both time and space costs of fixed-point algorithm are nearly 3 times smaller. In fact, in order to run simulations using floating-point numbers, microcontroller ATTiny84 with four times larger memory was used since the memory requirement was too big for both ATTiny2313 and a more powerful ATTiny48, colored red in the tables. The accuracy of these two methods is practically the same, and the difference in computational cost is therefore even more striking. In addition, it can be concluded that accuracy depends greatly on the number of support vectors. The higher number of support vectors usually means higher accuracy. However, in our experiments, higher number of support vectors is achieved at the expense of smaller bit-lengths, so the results in the tables can be misleading. In some experiments larger support vector set does not necessarily lead to better performance because the quantization error is too big and this affects predictions considerably. It should be noted that the support vector set size/bit-length trade-off is not the topic of this work and will be addressed in our future study.

4. CONCLUSION

In this paper we proposed approximation of Kernel Perceptron that is well-suitable for implementation on resource-limited devices. The new algorithm uses only integers, without any need for floating-point numbers, which makes it perfect for simple devices such as microcontrollers.

The presented results show that the approach considered in this paper can be successfully implemented. The described algorithm approximates the kernel function defined in (6) with high accuracy, and yields good results on several different datasets. The results are, as expected, not perfect, which is the consequence of the highly limited computation capabilities of ATTiny2313 that required several approximations in calculation of the prediction. While the quantization and approximation error limit the performance, the degradation is relatively moderate considering very limited computational resources. The simulations on ATTiny microcontrollers proved that the algorithm is very frugal and fast, while being able to match the performance of its unconstrained counterpart.

Although the kernel function defined in (6) is used, which is a slightly modified RBF kernel, the fixed-point method can also be used to approximate the original RBF kernel (2). The proposed idea could probably be extended to some other kernel functions

Table 3 Microcontroller implementation costs

Banana	*2 bits*		*4 bits*		*6 bits*		*8 bits*	
ATTiny2313	***Fixed***	***Float***	***Fixed***	***Float***	***Fixed***	***Float***	***Fixed***	***Float***
Program [B] (2048B)	1744	6036	1720	6012	1720	6012	1748	6040
Data [B] (128B)	128	379	128	379	128	379	128	381
Time [ms]	1192	6604	1985	7505	1883	7610	1739	7496
Accuracy [%]	**67.32**	**65.12**	**81.08**	**81.00**	**79.36**	**79.60**	**78.00**	**77.80**
# of SVs	112		62		43		32	

Table 4 Microcontroller implementation costs

Checkerboard	*2 bits*		*4 bits*		*6 bits*		*8 bits*	
ATTiny2313	***Fixed***	***Float***	***Fixed***	***Float***	***Fixed***	***Float***	***Fixed***	***Float***
Program [B] (2048B)	1744	6036	1720	6012	1720	6012	1748	6040
Data [B] (128B)	128	379	128	379	128	379	128	381
Time [ms]	1568	7626	2226	6725	2410	7366	2232	7703
Accuracy [%]	**52.20**	**51.20**	**72.60**	**72.64**	**78.20**	**78.60**	**74.04**	**73.80**
# of SVs	112		62		43		32	

Table 5 Microcontroller implementation costs

Pendigits	*2 bits*		*4 bits*		*6 bits*		*8 bits*	
ATTiny48	***Fixed***	***Float***	***Fixed***	***Float***	***Fixed***	***Float***	***Fixed***	***Float***
Program [B] (4096B)	1836	6078	1810	6058	1812	6056	1816	5912
Data [B] (256B)	256	513	256	511	256	507	256	503
Time [ms]	3280	14186	3572	15352	4881	17102	5063	17373
Accuracy [%]	**93.80**	**94.20**	**92.76**	**93.92**	**86.44**	**86.32**	**80.72**	**81.68**
# of SVs	46		23		15		11	

that require operations with floating-point numbers, such as polynomial kernel. This would be of great practical importance for problems that do not work well with the RBF kernel. This issue will be pursued in our future work.

The proposed algorithm could be improved by some more advanced budget maintenance method, such as the one described in [8]. By using support vector merging, the performance of predictor would certainly improve, but it is to be seen if the limited device can support such an approach. Also, in this paper we assumed that parameters A and B, kernel width and bit-length, respectively, are given. Although this is reasonable if we want to use this method for applications where the classification problem is well understood, it would be interesting to implement on a resource-limited device an algorithm that can automatically find the optimal parameter values.

5. ACKNOWLEDGMENTS

This work is funded in part by NSF grant IIS-0546155.

6. REFERENCES

[1] Anguita, D., Boni, A., Ridella, S., Digital Kernel Perceptron, *Electronics letters*, 38: 10, pp. 445-446, 2002.

[2] Anguita, D., Pischiutta, S., Ridella, S., Sterpi, D., Feed–Forward Support Vector Machine without multipliers, *IEEE Transactions on Neural Networks*, 17, pp. 1328-1331, 2006.

[3] Cesa-Bianchi, N., Gentile, C., Tracking the best hyperplane with a simple budget Perceptron, *Proc. of the Nineteenth Annual Conference on Computational Learning Theory*, pp. 483-498, 2006.

[4] Crammer, K., Kandola, J., Singer, Y., Online Classification on a Budget, *Advances in Neural Information Processing Systems*, 2003.

[5] Dekel, O., Shalev-Shwartz, S., Singer, Y., The Forgetron: A kernel-based Perceptron on a fixed budget, *Advances in Neural Information Processing Systems*, pp. 259-266, 2005.

[6] Freund, Y., Schapire, D., Large Margin Classification Using the Perceptron Algorithm, *Machine Learning*, pp. 277-296, 1998.

[7] Hildebrand, F. B., Introduction to Numerical Analysis, 2nd edition, Dover, 1987.

[8] Nguyen, D., Ho, T., An efficient method for simplifying support vector machines, *Proceedings of ICML*, pp. 617–624, 2005.

[9] Orabona, F., Keshet, J., Caputo, B., The Projectron: a bounded kernel-based Perceptron, *Proceedings of ICML*, 2008.

[10] Vapnik, V., The nature of statistical learning theory, Springer, 1995.

[11] Vucetic, S., Coric, V., Wang, Z., Compressed Kernel Perceptrons, *Data Compression Conference*, pp. 153-162, 2009.

[12] Wang, Z., Crammer, K., Vucetic, S., Multi-Class Pegasos on a Budget, *Proceedings of ICML*, 2010.

[13] Wang, Z., Vucetic, S., Tighter Perceptron with Improved Dual Use of Cached Data for Model Representation and Validation, *Proceedings of IJCNN*, 2009.

[14] www.atmel.com/dyn/resources/prod_documents/doc2543.pdf ATTiny2313 Datasheet

[15] www.digikey.com

Anomalous Thermal Behavior Detection in Data Centers using Hierarchical PCA

Manish Marwah
HP Labs
Palo Alto, CA, USA
manish.marwah@hp.com

Ratnesh Sharma
HP Labs
Palo Alto, CA, USA
ratnesh.sharma@hp.com

Wilfredo Lugo
HP Enterprise Business
Aguadilla, Puerto Rico
wilfredo.lugo@hp.com

Lola Bautista
CISE, University of Puerto Rico
Mayagüez, Puerto Rico
Lola.Bautista@ece.uprm.edu

ABSTRACT

In recent years, there has been a significant growth in number, size and power densities of data centers. A significant part of a data center power consumption is attributed to the cooling infrastructure, such as air handling units and chillers. For energy efficient operation and management of the cooling infrastructure, data centers are beginning to be extensively instrumented with temperature sensors. However, it is virtually impossible to manually inspect and analyze the large volumes of dynamic data generated by these sensors for presence of anomalous behavior. Furthermore, threshold-based methods are useful but limited in the kind of anomalies that can be detected. Thus, in order to improve energy efficiency of data centers, there is a need for real-time detection of thermal anomalies such that corrective measures can be promptly taken to remove the inefficiencies and save power.

In this paper, we propose a hierarchical principal component analysis (PCA) based methodology for detection of anomalous thermal behavior, and demonstrate it on a large temperature sensor network in a production data center. Specifically, the technique is applied to thermal anomalies that result from inefficiencies in the airflow pattern in a part of a data center and normally go undetected since no thresholds are violated. The hierarchical analysis performed on the temperature sensor data streams also identifies the location and scope of such anomalous behavior. A prototype of this technique has been implemented and applied to a temperature sensor network spanning 75 racks with 10 sensors each for over a period of 30 days. The results – accuracy: 98.0%, sensitivity: 87.1%, and specificity: 98.8% – demonstrate the effectiveness of our methodology in real-time detection of anomalous thermal behavior in data centers.

SensorKDD'10, July 25, 2010, Washington, DC, USA.

Categories and Subject Descriptors

H.2.8 [**Information Systems**]: Database Management:Database Applications - Data Mining; K.6.2 [**Management of Computing and Information Systems**]: Installation Management - Computing Equipment Management

Keywords

Data centers; sensor data; PCA; thermal anomalies; anomaly detection

1. INTRODUCTION

In recent years, the demand for data centers has seen tremendous growth. Many of the largest data centers in the US are experiencing a growth of 20% per year and over 40% of enterprises are refurbishing or building new data centers to support ongoing business operations and future demand [11]. However, energy consumption of data centers is a concern. The Environmental Protection Agency (EPA) calculates that in 2006, 61 billion kilowatt-hour (kWh) was consumed by data centers in the US. This amount accounts for 1.5% of the total electricity consumed, costing $4.5 billion [1]. Moreover, the cooling infrastructure can be responsible for up to 50% of that consumption [7]. It is estimated that data center power consumption will increase 4% to 8% annually and is expected to reach 100 billion kWh by 2011 [12].

Given these trends, monitoring thermal conditions in data centers and responding rapidly to anomalies assumes great significance and can help save energy and operational costs. Until recently data centers were a black box with minimal instrumentation in the way of thermal sensing. After their initial design (where cooling infrastructure was typically overprovisioned, thus leading to higher capital and operational energy costs), there was not much monitoring, only scheduled maintenance or repair after a failure occurred. However, the state-of-the-art data centers today are extensively instrumented and closely monitored. Indeed, a large data center can easily contain tens of thousands of sensors which produce a continuous stream of data. Although these sensors produce a wealth of information on the state of a data center, using this information effectively is a challenge. To detect an anomaly, an administrator must correlate observed mea-

surements to anomalous behavior based on past experience. In addition to very specific domain knowledge required, just the volume of data can be prohibitive to examine manually. The current industry trend is towards a lights out data center that is managed remotely with no manual intervention required.

The monitoring techniques currently deployed in data centers are typically threshold based, that is, they alarm when an administrator configured threshold is crossed. These, however, do not always work well and important anomalies are missed since many do not manifest as threshold violations. Also, early detection of anomalies, which allow preemptive measures to be taken, is difficult using only threshold techniques.

Furthermore, when an anomalous sensor reading is observed, current monitoring systems raise alarms requiring investigation by an administrator. It is nontrivial to determine if the cause of the anomaly is local or related to a larger, facility wide outage. For example, a high temperature sensor reading could be caused by any of the following: 1) a faulty sensor; 2) a rack level anomaly e.g. obstruction of a cool air vent near a rack; or, 3) a failed computer room air-conditioning (CRAC) unit affecting a significant portion of a data center. Automated mechanisms to determine which of the above has occurred is challenging.

The observations made above necessitate automated, timely and specific anomaly detection using the available sensor data streams. In this paper, we propose a hierarchical, principal component analysis (PCA) based technique for automated monitoring of correlations between sensor measurements within a data center. Any change in the correlations signals anomalous behavior. Correlations across several hierarchical groupings are analyzed to determine the extent of an anomaly. Furthermore, the sensors most likely responsible for the anomalous behavior are identified.

We conducted performance evaluation of our methodology at a large, heterogeneous, state-of-the-art production data center. For efficient monitoring and control, this facility has an extensive infrastructure of sensors. The results show that we can detect anomalies at rack, zone and data center region levels. For rack level analysis, the results show an accuracy, sensitivity and specificity of 97.96%, 87.1% and 98.76%, respectively. Threshold based methods are unable to detect most of these anomalies.

Specifically, in this paper, we make three key contributions.

1. We present a scalable, hierarchical PCA-based data mining methodology that can be applied to a large data center sensor network.

2. We introduce a mechanism that allows PCA hidden variables associated with short-lived and insignificant trends to be ignored.

3. We demonstrate the effectiveness of our technique by analyzing sensor data from around 375 temperature sensors for a period of 30 days in a real life production data center.

The rest of the paper is organized as follows. In the next section, we discuss related work. In section 3, we discuss the hierarchical anomaly detection methodology. The layout and architecture of the data center where we demonstrate our techniques is described in section 4. The results are presented in section 5. Finally, we conclude in section 6.

2. RELATED WORK

Considering the huge potential for cost and energy savings, mining of streams of environmental data in data centers has recently received attention. Additionally, local temperature sensing within a data center for better thermal management is becoming important [19, 6]. In the past, exploratory data analysis techniques have been used to evaluate data center environmental data [18]. While statistical and Fourier analysis of air temperature data from rack inlet sensors was performed, the study did not detect events or anomalies within a data center.

SPIRIT [16] performs on-line PCA on n data streams by incrementally updating the principal components as each data point arrives. As long as correlations between these streams continue to hold, the number of hidden variables remain constant. Change in the number of hidden variables indicates anomalous behavior. While our methodology is based on SPIRIT, we make it scalable by using hierarchical groupings, and add a mechanism to filter out hidden variables associated with short-lived trends.

InteMon [13] provides a prototype for monitoring data center information through use of SPIRIT [16]. It analyzes correlations in real-time and alarms on detecting an anomaly. While our work is related to InteMon, there are clear differences. InteMon uses only four temperature sensors, while we analyze a large sensor network consisting of 375 temperature sensors. Using a hierarchical approach makes our technique inherently more scalable. Furthermore, it is only through rich instrumentation that anomalies that we are interested in surface.

In recent years, data streams have been the focus of extensive research. The availability of continuous, real time, dynamic data in systems, such as, sensor networks and web servers, and the need for real-time monitoring and analysis have been the prime motivations. Traditional database management systems (DBMS) [10] are not suited to store or process such high volume data streams due to performance and storage limitations. However, data stream management systems (DSMS) [5] have emerged to address this need. They aim to provide DBMS like functionalities for data streams [5, 8, 15].

In addition to DSMS, the other major area of research in data streams – and the one that is the focus of this paper – is mining of data streams for discovering patterns and correlations. Numerous research efforts have focused on clustering and classifying data streams into groups including CluStream [2] and HPStream [3]. Each data stream is passed through an evaluation function – typically based on distance measures – which determines the membership of the stream. Data mining of time series data has been investigated in many research projects including SPIRIT [16] and StatStream [21]. StatStream uses discrete Fourier transform for computing statistical measures over time series data streams.

3. STREAM MINING OF SENSOR DATA

Mining of sensor data in a data center can provide important information for its management including control, optimization and fault-tolerance of various devices and pro-

cesses. Early detection of abnormal events such as failure of a computer room air conditioning (CRAC) unit can be used to redeploy resources and minimize any potential user impact. Cooling resources from other CRAC units can be provided to the affected region. Additionally, if server virtualization techniques are in use, workload can be preemptively migrated to other racks not affected by the failure. Similarly, identification of an errand temperature sensor by observing the correlations between the sensors in the same rack provides valuable information to the data center management system, allowing it to ignore measurements from such sensors instead of taking remedial actions.

3.1 Hierarchical Methodology

The goal of our methodology is to analyze sensor data streams to detect anomalous behavior in a data center. Anomalies that manifest as broken correlations between sensors are detected. In addition to detecting an anomaly, the level or scope of the anomaly is also inferred. In the data center context, this refers to whether the anomalous behavior occurred at a sensor level, a rack level, or in an entire zone of a data center. An advantage of our approach is that no prior learning or training is required. Furthermore, by virtue of being hierarchical, it is very scalable.

The core component of the technique consists of analyzing sensor measurements organized in hierarchical groupings. The analysis comprises performing streaming principal component analysis (PCA) on each grouping and at each level. Considering hierarchical groupings provides a deeper insight into the location and nature of an anomaly. The groupings exploit correlations that exist between sensors during normal operation. In this paper, a simple mechanism of grouping the sensors, based on their physical locality, is used since the expectation is that closely located temperature sensors receive similar air flow and hence are likely to show correlated behavior. We verified this by using historic data to compute correlation coefficients between pairs of sensors in the same group. Note that in the absence of any domain knowledge, these correlation coefficients computed over historic data could be used to group the sensors. Further, our technique is generic and does not depend on the criterion used for the grouping.

We consider three groupings: 1) Rack level, 2) Zone level, and 3) Region level. As shown in Figure 1, PCA is conducted in a bottom up fashion starting at the rack level. At each level, trends in sensor measurements are evaluated to identify anomalous behavior of the entities comprising that level. Analysis at a particular level requires trends from the level below. For example, zone level analysis requires rack trends and allows rack anomalies to be discovered. Trends (hidden variables) identified as anomalous at a particular level are removed from analysis at higher levels. The three hierarchical levels considered are described below.

Rack Level. This is the lowest level consisting of sensors located in a rack. The objective of rack level analysis is to identify anomalous behavior at the scale of a sensor. Incremental PCA is separately performed on groups of sensor data streams from each rack. The expectation is that during normal operation sensors in the same rack are correlated. Ceasing of this correlation indicates an anomaly. In PCA, this is reflected by a change in the number of hidden variables. Furthermore, the sensor(s) associated with the anomalous behavior is (are) also identified as discussed in the next section.

Zone Level. A zone consists of a group of racks. Zones are demarcated based on commonality of an attribute related to the sensor measurements. For example, for temperature sensors, racks in a row are considered one zone. The objective of zone level analysis is to identify entire racks within a zone that show anomalous behavior. Analysis at this level utilizes the results of the rack level analysis. The trends (hidden variables) – discovered in the rack level analysis – of each rack in a zone are analyzed together to determine if the number of trends remain preserved. An additional trend indicates anomalous behavior. An example of a rack level anomaly is an obstruction blocking a cool air vent next to a rack. This causes the rack sensors to exhibit deviant behavior. Note that rack level analysis is unlikely to uncover this problem, since the blocked vent affects all sensors in the rack, which are likely to remain correlated.

Region Level. This level consists of a number of related zones. For example, all zones under the influence of a particular CRAC unit can be considered together. The objective of analysis at this level is to discovery aberrant zones. The main trends from each zone – computed in the zone level analysis – are used to perform incremental PCA. The emergence of additional trends in the results indicates anomalous behavior.

Although, in this paper, we only use temperature sensors, sensors measuring other characteristics such as humidity can also be simultaneously analyzed to detect related anomalies [20]. Furthermore, even with one kind of sensor, different criteria for association of sensors can be used. In addition to physical locality, locality based on data center cooling infrastructure or workload/power consumption of servers can be exploited.

3.1.1 Pre-processing

Before being passed through the PCA algorithm, a data stream is pre-processed to make it more amenable to PCA analysis. This consists of two main components. First, high frequency noise is removed through use of a moving average filter. Then, the data is normalized such that it has zero mean and a standard deviation of one, that is,

$$T' = (T - \mu)/\sigma \tag{1}$$

Although this is trivial to do for historical data, efficiently computing mean and standard deviation over a sliding window for streaming data is challenging. While several research efforts [21, 9, 4] have focused on computing statistics for streaming data without having to store an entire window's worth of data, we use a simple solution.

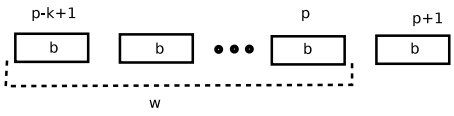


Figure 2: Computing sliding window mean for streaming data.

The basic idea is to divide the sliding window into blocks and statistics related to these blocks are preserved and used to update statistics over the sliding window. Assume a window size of w is divided into k blocks of size b, as shown in

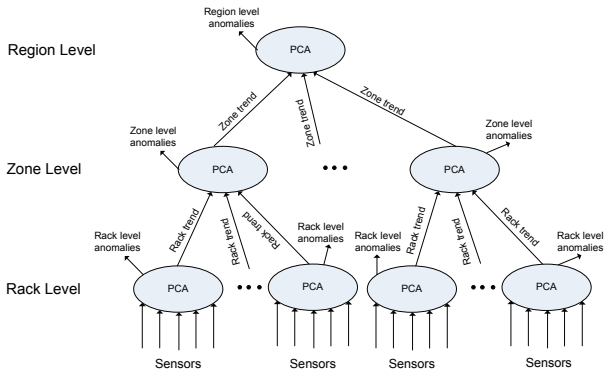


Figure 1: Our hierarchical PCA methodology applied to data centers with groupings of temperature sensors at three levels, namely, rack, zone and region.

Figure 2. At the beginning/end of each block, the statistics can be accurately computed. The sliding window mean at the end of block $p+1$ is given by

$$\mu_{w,p+1} = \mu_{w,p} - \frac{\mu_{b,p-k+1} \cdot b}{w} + \frac{\mu_{b,p+1} \cdot b}{w} \tag{2}$$

where $\mu_{w,p}$ is the sliding window mean at the end of block p. While in a block, the mean can be estimated by assuming that the data point moving out of the sliding window is equal to the average of its block and updating the mean with the newly arrived point. Standard deviation of streaming data can be similarly computed. It requires that sum of squares for each block be also saved.

Since temperatures depend on server workloads, the appropriate choice of the window size is governed by the workload patterns. In our analysis, based on the observed diurnal workload pattern in the data center, we use a window size of 24 hours with block size of 15 minutes.

3.1.2 PCA of streaming data

Our methodology to discover trends and anomalous behavior in data center sensor streams involves using principal component analysis (PCA) [14]. PCA is a generic technique to reduce the dimensionality of correlated variables by introducing a new orthogonal basis. These are called the principal components (PCs). Each PC is successively chosen such that it captures the maximum variance remaining in the data. Thus, usually the first few PCs are sufficient for reconstructing the data to a good approximation. Since the PC directions are orthogonal, they are uncorrelated. Note that PCA only considers linear dependence; non-linear interdependence between variables is not captured by PCA. Another assumption is that the data has a normal distribution, a property satisfied by the temperature sensor data considered in this paper.

At each time tick, a data point (vector containing a measurement from each sensor) is received and transformed from the original n-dimensional space to the new m-dimensional space by taking its projection onto the PCs.

$$\mathbf{y}_{m\times 1} = \mathbf{W}_{m\times n} \cdot \mathbf{x}_{n\times 1} \tag{3}$$

where $\mathbf{x}$ is the input vector; $\mathbf{y}$ is the output vector in the PC space (the components of $\mathbf{y}$ are also called hidden variables); $\mathbf{W}$ is the projection matrix with its ith row containing a unit vector along the ith PC. A row vector of $\mathbf{W}$ is also called the participation weight vector since its elements determine the contribution of an input (x_i) to a hidden variable (y_i). This is very useful information since it can be used to rank the contributions of input variables to a particular hidden variable. The original variables can be reconstructed as follows:

$$\tilde{\mathbf{x}}_{n\times 1} = \mathbf{W}^T_{n\times m} \cdot \mathbf{y}_{m\times 1} \tag{4}$$

The reconstruction error is given by $||\mathbf{x} - \tilde{\mathbf{x}}||^2$.

The basic assumption in using PCA for anomalous behavior detection is that during normal operation the number of PCs remains constant. An increase or decrease in the number of hidden variables indicates an underlying change in the number of correlations of the original data and hence considered anomalous behavior. While our application of PCA to streaming data is based on SPIRIT [16], we improve scalability by hierarchical processing. We also make one other enhancement (described further in the next section): the criterion for determining the number of hidden variables is modified such that short-lived and insignificant trends are ignored. The algorithm incrementally updates the PCs (matrix W) as each data point arrives. It is efficient with $O(mn)$ complexity in both time and space and is independent of the total number of data points seen. In our analysis each sensor measurement is considered a separate dimension. Thus, n is equal to the number of sensors being analyzed.

3.1.3 Number of hidden variables

The number of hidden variables depends on the degree of reconstruction accuracy desired. A common technique to estimate this number is energy thresholding [16, 14]. Energy of a time series variable (y_i) is defined as the average sum of squares of all its past values.

$$E(i) = 1/t \sum_{i=1}^{t} {y_i}^2 \qquad i \in [1, t] \tag{5}$$

Energy thresholding operates as follows. The energies of the reconstructed and original variables are compared. As long as this ratio is within threshold limits (e.g. 0.95 and

0.98), the number of hidden variables is kept constant. If it falls below the lower threshold (indicating unacceptably high reconstruction error), the number of hidden variables is increased. On the other hand, if it rises above the upper threshold (indicating unacceptably low reconstruction error), the number is decreased.

An issue with energy thresholding is that small changes in the value of the energy ratio around the threshold values increases or decreases the number of hidden variables, signaling anomalous behavior. However, these new trends created may be short lived and insignificant, likely related to a transient phenomenon in the original data. In order to filter out such trends, energy thresholding is enhanced to also consider the energy contribution of a new hidden variable in conjunction with the thresholds. A new hidden variable, i, is considered viable only if it has made a significant contribution to the total energy since it appeared, i.e.,

$$E(i)_a \geq \alpha \cdot E_a \tag{6}$$

continues to hold for b time ticks. Here, $E(i)_a$ is the contribution of the ith hidden variable since time a; E_a is the total energy since time a; and, α is the contribution threshold. The parameters α and b can be adjusted based on the degree of sensitivity desired. For the results described in section 5, the values of α and b were set at 0.4 and 6, respectively. These values worked well for the temperature data analyzed.

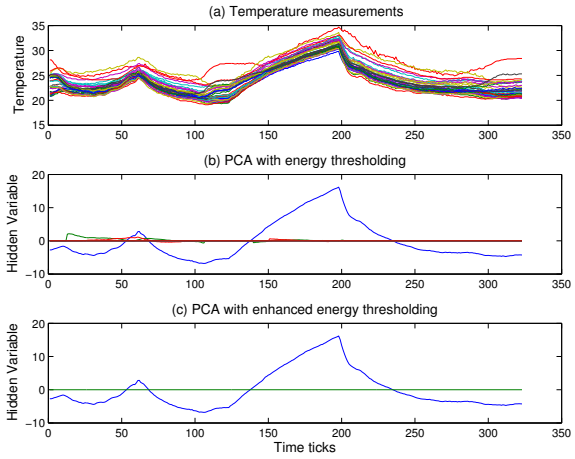


Figure 3: Use of enhanced energy thresholding with PCA analysis removes insignificant trends.

Figure 3 (a) shows temperature measurements from seven racks, that is, 35 sensors in all. The hidden variables that result from conducting incremental PCA are shown in Figure 3 (b). In addition to the main trend, four short-lived trends (appearing at time 12, 23, 139 and 150) are also seen. These are caused by transitory disturbances and are not significant trends. Although uninteresting, these events are not distinguished from cases where a major new trend appears since both are signaled by the appearance of a hidden variable. However, using the mechanism described above, these insignificant trends are filtered out (shown in Figure 3 (c)). In all the results described in section 5, this hidden variable filtering algorithm was used.

4. EXPERIMENTAL TEST BED

We apply our analysis and data stream mining methodology to a real life, state-of-the-art data center. In this study, temperature sensor data from a production data center is considered. Power consumption on a per rack basis in this data center ranges from 5 to 20kW. Racks comprise of off-the-shelf standard or blade servers, storage arrays and network switches. Note that our methodology is generic and not limited to the data center architecture presented here.

4.1 Data Center Infrastructure

These data centers are air-cooled with a raised floor plenum to distribute cool air, power and networking. Figure 4 depicts a typical state-of-the-art data center air-conditioning environment with under-floor cool air distribution [17]. Computer room air conditioning (CRAC) units cool the exhaust hot air from the computer racks. Energy consumption in data center cooling comprises work done to distribute the cool air to the racks and to extract heat from the hot exhaust air. The air movers in the CRAC units pressurize the plenum with cool air which enters the data center through vented tiles located on the raised floor close to the inlet of the racks. Typically the racks are laid out in rows separated by hot and cold aisles as shown in Figure 4. This separation is done for thermal efficiency considerations. Air inlets for all racks face cold aisles while hot air is expelled to hot aisles.

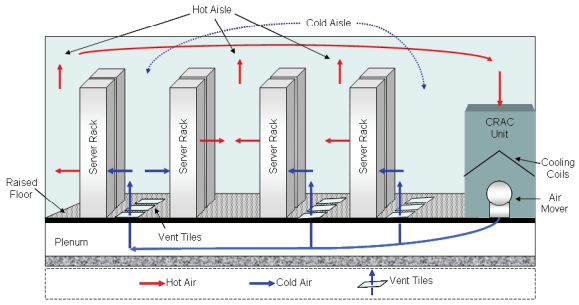


Figure 4: A typical raised-floor data center.

4.2 Sensor Network and Data Aggregation

Temperature data is collected from sensors, mounted at the inlet and outlet of racks (see Figure 5). A data center wide distribution of such temperature sensor networks are deployed on rack-to-rack basis. The placement density of the sensors is based on the power dissipated per unit area of a data center. The temperature sensors are mounted on racks as shown in the figure and provide temperature data at both air inlet and outlet of the racks. The digital output from each sensor is accurate to 0.5 C in the range of interest. Since the sensor is primarily a transistor with compensation for leakage, no calibration is needed. Ten temperature sensors are attached to each rack, with five at the inlet and the other five at the outlet. Each rack also contains a base station to which all sensors on a rack are connected. The base station has an Ethernet interface and multicasts the temperature data collected on the data center LAN. In addition

to temperature sensors, data is collected from CRAC units, Variable fan drive (VFD) units and power distribution units (PDUs). However, in this paper, only rack inlet temperature data is considered since it is more critical (as compared to outlet temperature) in determining the thermal well-being of the entities in a data center.

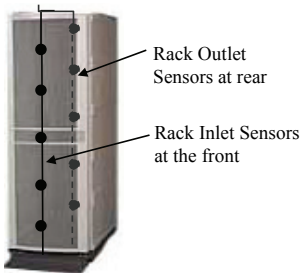


Figure 5: Sensors mounted at the rack inlet and exhaust.

An underlying assumption in the use of PCA on a data set is that it is normally distributed. Figure 6 shows the cumulative frequency distribution (CFD) of typical instances of the temperature sensor data taken from the test bed data center. The standard normal curve is also shown for comparison. In the anomaly-free case, the close agreement between the normal CFD and the temperature data CFD indicates that the temperature data is normally distributed with random variations. The anomalous data deviates slightly from the normal curve due to the systemic variation in the temperature values because of the anomaly.

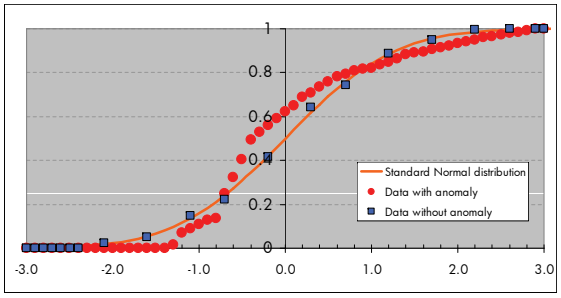


Figure 6: Distribution of temperature sensor data.

5. RESULTS AND DISCUSSION

As a proof of concept, we have implemented a prototype of our methodology and applied it to sensor data streams obtained from a real-life production data center located in Palo Alto, CA. The servers in this 3000 sq. ft. facility dissipate 350 KW of power. Its architecture is similar to that described in Section 4. There are 75 racks of computing equipment, each with 10 temperature sensors, arranged in rows as shown in Figure 7. Each rack can contain up to 64 blade servers or 42 1U servers. Six CRAC units provide cooling. Temperature data streams from five sensors located at the air inlet of each rack, resulting in 375 data streams in all, are analyzed.

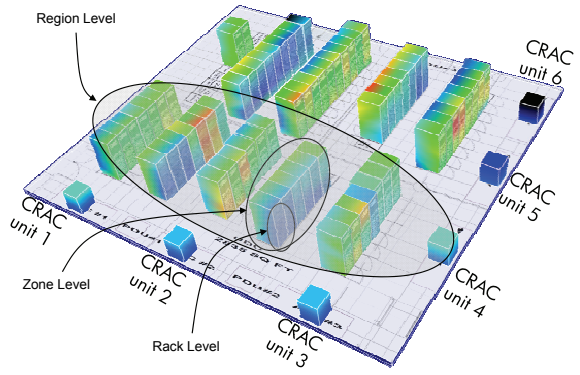


Figure 7: Layout of the test bed data center.

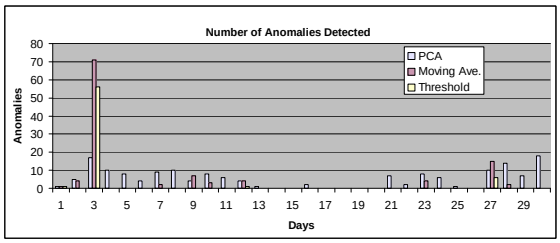


Figure 8: Number of anomalies detected over a period of 30 days.

Figure 8 shows the number of rack anomalies detected on each day for a 30 day period, from January 1, 2008, to January 30, 2008. In addition to our methodology (labeled PCA), also shown are the number of rack anomalies detected through (1) a threshold method, where an anomaly is flagged if any temperature sensor in a rack exceeds 30°C, and (2) a moving average method, where an anomaly is flagged if a rack temperature is greater than 5°C from the moving average of the previous 24 hours. During this period a major failure occurred on day 3 when all CRAC units failed due to unavailability of chilled water, and a similar minor failure occurred on day 27. These are captured well by the moving average and threshold methods. The PCA method does not appear to do well for such large scale anomalies where temperature of sensors remain correlated while increasing. However, many anomalies manifest with no violation of temperature thresholds and are thus particularly hard to detect. Several of these can be detected through the PCA method since they result in uncorrelated behavior between sensors. These anomalies indicate inefficient airflow in the data center and result in higher power consumption. The cause of airflow inefficiencies could be related to misconfiguration of equipment, or increased recirculation of hot air. However, automatic determination of the cause of a particular anomaly is beyond the scope of the current work. Anomaly detection allows an operator to investigate further and if required take corrective measures to fix airflow inefficiencies, thus, saving power. Note that the threshold-based and PCA-based methods compliment each other.

In order to validate the performance of the PCA method,

a thermal sciences expert visually inspected the daily rack temperature plots for the 30 days and identified racks that seemed to exhibit abnormal behavior. Each of the 75 racks, for each of the 30 days, were marked as anomalous or normal. These labeled samples were then compared with the results obtained using PCA. The resulting confusion matrix is shown in Table 1. In all, there are 2250 day-long samples (75 racks over 30 days). In the table, *Positive* indicates presence of an anomaly while *Negative* indicates its absence. 135 anomalous and 2069 normal samples are correctly classified. There are 26 false positive samples while 20 are false negatives. There are 155 anomalies in all (about 7%). Since the anomaly rate is relatively low, the total *accuracy*, that is, proportion of correctly classified samples, of 97.96%, although high, is not very significant. The *sensitivity*, which measures the true positive rate, and the *specificity*, which measures the true negative rate, are better indicators of the performance. As shown in Table 2, these are 87.1% and 98.76%, respectively. The precision of the PCA method, that is, the proportion of true positives out of the total number of positives, is 83.85%.

		PCA Method		
		Positive	Negative	Total
Actual	Positive	135	20	155
	Negative	26	2069	2095
	Total	161	2089	2250

Table 1: Results from the PCA method as compared to the actual positive (anomalous) and negative (normal) results, as provided by the domain (thermal sciences) expert.

Measure	Value(%)
Accuracy	97.96
Sensitivity	87.1
Specificity	98.76
Precision	83.85

Table 2: Summary of the performance of the PCA method.

In the following sections, we present some qualitative results from the 30 day run and show how the hierarchical analysis allows the source and scope of an anomaly to be identified.

5.1 Rack Level Analysis

Figure 9 (a) shows the temperature measurements from five sensors located on the same rack (A1). Each time tick corresponds to about 1 minute. The key point to note is that although the temperature varies in the rack, the five sensors follow the same trend. This trend is captured by a single hidden variable obtained by conducting PCA (shown in Figure 9(b)). This also shows the usefulness of hidden variables in summarizing trends.

Five temperature measurements from a different rack (Bext4), during the same period of time, are shown in Figure 10(a). After conducting PCA, we discover two trends (Figure 10(b)). The larger trend is similar to the one seen for the previous rack (A1); however, an additional trend starting at time

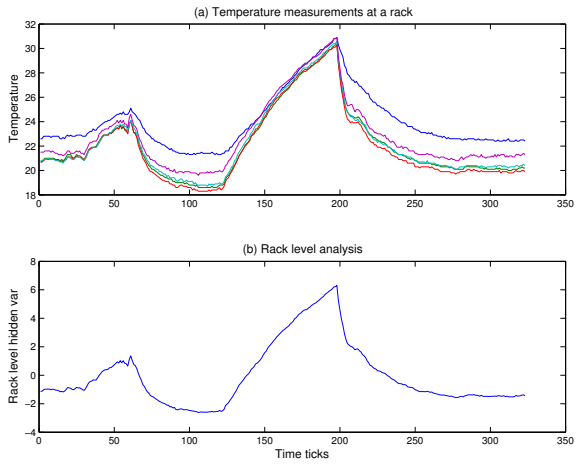


Figure 9: (a) Rack temperature data; (b) One hidden variable is able to capture all five temperature sensors.

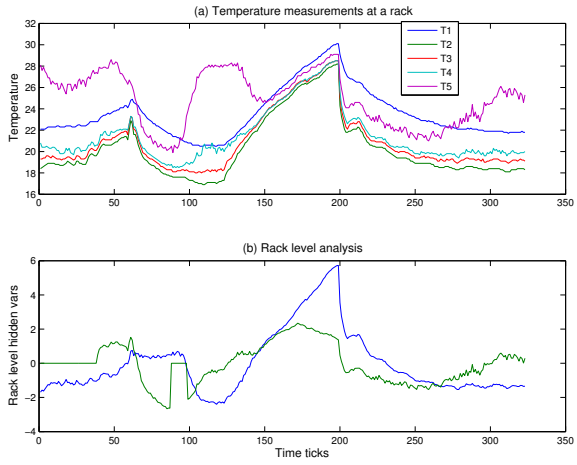


Figure 10: Analysis of Uncorrelated Rack Temperature Data.

tick 38 is also seen. The largest contributor to this new trend – as determined from the weight vector – is sensor 5 (T5). Although the fact – that T5 shows deviant behavior – is quite apparent from the temperature plot, the ability to identify this behavior and the particular sensor involved autonomously in a data center with thousands to tens of thousands of sensors is a big advantage. Furthermore, in this case, the new trend is detected before (at time tick 38) it becomes visually obvious from the temperature plot (between time ticks 50 and 100). This is an extremely useful input to a data center monitoring and control system which can perform further analysis to determine the root cause, or take other preemptive actions. Note that since the deviant sensor shows temperatures that are within the normal operation range, a threshold based mechanism will be unable to detect this behavior.

5.2 Zone Level Analysis

At this level, a group of racks, organized as a zone, is an-

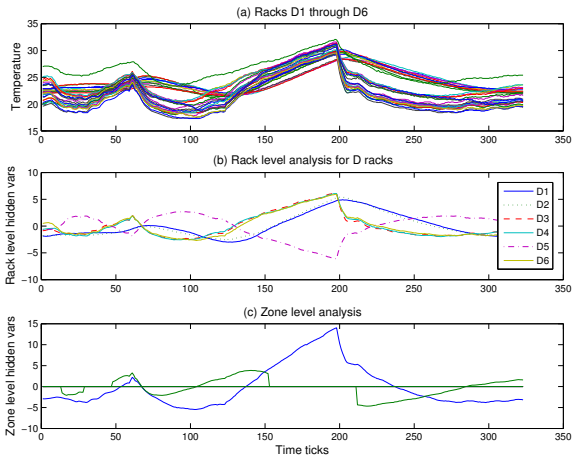


Figure 11: Zonal Analysis of Rack Temperature Data.

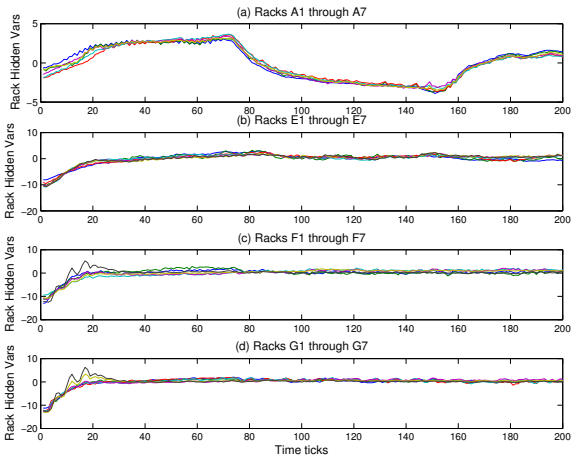


Figure 12: Hidden Variable plots from rack level analysis for zones A, E, F and G.

alyzed with the objective of detecting anomalous behavior in an entire rack. Figure 11(a) shows the raw temperatures of six racks (D1 through D6). These racks, comprising a zone, are located in the same aisle. The main trends (hidden variables) for the six racks, computed during rack level analysis, is shown in Figure 11(b). These six variables, each representing one rack, are passed through another round of PCA. The results, shown in Figure 11(c), indicate two hidden variables. The smaller trend can be traced to racks D1 and D2. The larger one represents the other racks. Note that in Figure 11(b) trend D5 is essentially the same as D3, D4 and D6 (inverting D5 will result in a close approximation of the others). The results indicate that racks D1 and D2 show anomalous behavior as compared to the other racks in the zone. Another observation (from Figure 11 (c)) is that the anomalous behavior is intermittent as on two occasions the second hidden variable disappears. Although deviant behavior can be identified, the cause of the deviance cannot be inferred though this analysis.

5.3 Region Level Analysis

At the region level, trends from multiple zones are analyzed together to detect the existence of zone-wide anomalous behavior. Note that an anomaly impacting an entire zone may not be detected at the zone level analysis, since the zone may continue to show correlated behavior. However, conducting PCA on multiple zones, that show correlated behavior during normal operation, can facilitate identification of entire zones that exhibit anomalous behavior.

Figure 12 shows the hidden variables of racks, obtained from rack level analysis, for four different zones (Zones A, E, F and G). Each zone consists of seven racks in a single aisle and each rack is summarized by one hidden variable. Zone level trends for the four zones are plotted in Figure 13 (a). Note that each zone can be represented by one hidden variable implying that within each of these zones the temperature behavior is highly correlated.

PCA is performed on the four zone level hidden variables and the results are plotted in Figure 13 (b). Two distinct trends can be seen. Trend T1 strongly corresponds to Zone A (as determined from the participation weight vector) while trend T2 is associated with the remaining zones, namely, E, F and G. The implication is that while the behavior of zones E, F and G remains correlated, zone A shows anomalous behavior. Note that this is obvious from the rack level hidden variables (Figure 12) where racks A1 through A7 show markedly different behavior than the other racks. The key advantage is that this distinction can be autonomously deduced without human involvement. From knowledge of the data center during this time period, it is known that the settings at a CRAC unit next to zone A racks were being manually changed. Due to their location, Racks E, F and G were not impacted by this event. The region level analysis is aimed at detection of such larger scale anomalies.

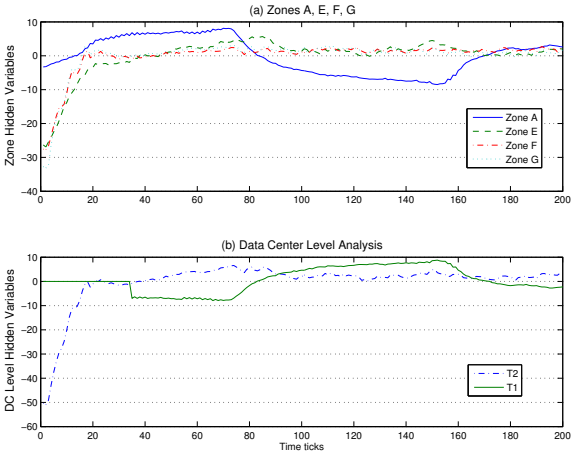


Figure 13: Zonal and Region level Analysis.

6. CONCLUSIONS AND FUTURE WORK

Timely and specific discovery of anomalous behavior is vital for efficient and cost-effective management of state-of-the-art data centers hosting tens of thousands of servers. Considering the large volumes of sensor data continuously being produced, automated mechanisms for discovering anoma-

lies and trends are indispensable. In this paper, we used incremental PCA on data streams generated by a large temperature sensor network in a data center. This allowed hard-to-detect anomalous behavior resulting from airflow inefficiencies in a data center to be detected and then potentially fixed to save energy. A hierarchical methodology, that is inherently scalable and allows the scope of an anomaly to be determined, was proposed. Furthermore, an enhanced mechanism to detect new hidden variables — that filters short-lived and insignificant trends — was presented. Our methodology was deployed in a production data center and we presented results from 30 continuous days of operation involving tens of racks and hundreds of sensors. The results are encouraging and validate the performance of our methodology.

While anomalous events where correlations break can be detected, the severity of the events or their root cause cannot be determined by PCA analysis alone. Bayesian networks could be used to model the temperature sensors to achieve that; similarly, classifiers could also be integrated to identify specific anomalies. Other future directions include mining of correlations between different kinds of sensors, for example, temperature and humidity sensors; and IT systems data, such as, OS logs, server CPU utilizations, application response times, etc.

7. REFERENCES

[1] U.S. Environmental Protection Agency. Report to congress on server and data center energy efficiency public law. pages 109–431, 2007.

[2] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for clustering evolving data streams. In *vldb'2003: Proceedings of the 29th international conference on Very large data bases*, pages 81–92. VLDB Endowment, 2003.

[3] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for projected clustering of high dimensional data streams. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pages 852–863. VLDB Endowment, 2004.

[4] Arvind Arasu and Gurmeet Singh Manku. Approximate counts and quantiles over sliding windows. In *PODS '04: Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 286–296, New York, NY, USA, 2004. ACM.

[5] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2002.

[6] Cullen Bash, Chandrakant Patel, and Ratnesh Sharma. Dynamic thermal management of air-cooled datacenter. In *ITherm2006, Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*, 2006.

[7] C. L. Belady. In the data center, power and cooling costs more than the it equipment it supports. *Electronics Cooling*, 13(1), Feb 2007.

[8] Sirish Chandrasekaran, Owen Cooper, Amol Deshpande, Michael J. Franklin, Joseph M. Hellerstein, Wei Hong, Sailesh Krishnamurthy, Samuel R. Madden, Fred Reiss, and Mehul A. Shah. Telegraphcq: continuous dataflow processing. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 2003.

[9] Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows: (extended abstract). In *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 635–644, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.

[10] Lukasz Golab and M. Tamer Özsu. Issues in data stream management. *SIGMOD Rec.*, 32(2):5–14, 2003.

[11] Carl Greiner. Considerations for a 'green' energy-efficient data center. *Ovum*, 2008.

[12] The Climate Group. Smart 2020: Enabling the low carbon economy in the information age, 2008. http://www.theclimategroup.org.

[13] Evan Hoke, Jimeng Sun, John D. Strunk, Gregory R. Ganger, and Christos Faloutsos. Intemon: continuous mining of sensor data in large-scale self-infrastructures. *SIGOPS Oper. Syst. Rev.*, 40(3):38–44, 2006.

[14] I. T. Jolliffe. *Principal Component Analysis*. Springer, 2002.

[15] Alberto Lerner and Dennis Shasha. Aquery: query language for ordered data, optimization techniques, and experiments. In *vldb'2003: Proceedings of the 29th international conference on Very large data bases*, pages 345–356. VLDB Endowment, 2003.

[16] Spiros Papadimitriou, Jimeng Sun, and Christos Faloutsos. Streaming pattern discovery in multiple time-series. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 697–708. VLDB Endowment, 2005.

[17] R. Schmidt. Computer and telecommunications equipment room cooling: A review of literature. In *Proceedings of Eighth Intersociety Conference on Thermal and Thermomecanical Phenomena in Electronic Systems*, May 2002.

[18] Ratnesh Sharma, Rocky Shih, Chandrakant Patel, and John Sontag. Application of exploratory data analysis to temperature data in conventional data centers. In *Proc, IPACK*, 2007.

[19] Ratnesh K. Sharma, Cullen Bash, Chandrakant D. Patel, Richard J. Friedrich, and Jeffrey S. Chase. Balance of power: Dynamic thermal management for internet data centers. *IEEE Internet Computing*, 9(1):42–49, 2005.

[20] Jimeng Sun, Dacheng Tao, and Christos Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 374–383, New York, NY, USA, 2006. ACM.

[21] Yunyue Zhu and Dennis Shasha. Statstream: statistical monitoring of thousands of data streams in real time. In *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*, pages 358–369, 2002.

Self-Organizing Energy Aware Clustering of Nodes in Sensor Networks Using Relevant Attributes

Marwan Hassani [•] Emmanuel Müller [•] Pascal Spaus [•] Adriola Faqolli
Themis Palpanas [○] Thomas Seidl [•]

[•]Data Management and Data Exploration Group
RWTH Aachen University, Germany
{hassani, mueller, seidl}@cs.rwth-aachen.de

[○]Department of Computer Science
University of Trento, Italy
themis@disi.unitn.eu

ABSTRACT

Physical clustering of nodes in sensor networks aims at grouping together sensor nodes according to some similarity criteria like neighborhood. Out of each group, one selected node will be the group representative for forwarding the data collected by its group. This considerably reduces the total energy consumption, as only representatives need to communicate with distant data sink. In data mining, one is interested in constructing these physical clusters according to similar measurements of sensor nodes. Previous data mining approaches for physical clustering concentrated on the similarity over all dimensions of measurements.

We propose ECLUN, an energy aware method for physical clustering of sensor nodes based on both spatial and measurements similarities. Our approach uses a novel method for constructing physical clusters according to similarities over some dimensions of the measured data. In an unsupervised way, our method maintains physical clusters and detects outliers. Through extensive experiments on synthetic and real world data sets, we show that our approach outperforms a competing state-of-the-art technique in both the amount of consumed energy and the effectiveness of detecting changes in the sensor network. Thus, we achieve an overall significantly better life times of sensor networks, while still following changes of observed phenomena.

Categories and Subject Descriptors

H.2.8 [**Database management**]: Database applications—*Data mining*

General Terms

Algorithms, Management

Keywords

Physical Clustering, Relevant Attributes, Subspace Clustering, Sensor networks, Energy Efficiency, Change detection

SensorKDD'10, July 25, 2010, Washington, DC, USA.

1. INTRODUCTION

The communication process is the dominating energy consumer in sensor networks, particularly when this is happening over long distances. Sensor nodes need to use their full sending power to forward their sensed data to distant sink, while they can use less power when communicating locally with each other. Considering the energy limited resources in sensor networks, this motivated a lot of research on the physical clustering of sensor nodes. The idea is to divide sensor nodes into groups according to some criteria, and then selecting one node from each of these group to serve a group representative. The main task of the group representative is forwarding the readings of sensor from its group to this distant sink. As nodes need to communicate within the group (the cluster) using less energy, this considerably reduces the total consumed energy in the whole network.

Data mining approaches contributed to this problem mainly in two parts: the criteria used for clustering and the process of selecting representatives. The similarity of sensed measurements and spatial characteristics were used as a grouping measure. Thus, inside each cluster, the node with the most similar readings to the measurements of all nodes inside that cluster is selected as a cluster representative.

In both cases, the selection methodology is based on the similarity between all attributes of clustered nodes. Todays sensor nodes are collecting increasingly many number of dimensions for each sensor node. The similarity measures should cope with the increasing dimensionality of sensed data. In such data, distances grow more and more alike. The full data space is thus sparse and each nodes will be alone in its physical cluster as no global similarity between the measurements of different nodes can be observed. We are tackling this point in this work, by introducing a novel method for performing physical clustering based on the similarity over some of the sensed attributes using subspace clustering. We show that this method produces improvements in energy consumption even for low dimensional data.

In addition to the importance of saving energy, we designed our method to cope with the change detection. Detecting novelty in input stream is an important feature that has to be considered when designing any data knowledge technique in sensor networks. For example, it is an essential point in evaluating learning algorithms in drifting environments [9].

1.1 Our Contribution

The following aspects are our main contributions we included in this work:

- **Reducing the communication burden**
  In our approach, nodes do not continuously communicate with the representative. Communication is established only when a state change is detected in the monitored phenomena. By the careful construction of clusters, this communication is further reduced by using the similarity to representative readings.
- **Subspace physical clustering**
  Our novel method for building clusters according to the relevant attributes results in more consistent clusters, and helps for maintaining the clusters with less effort.
- **Outlier-aware change detection**
  We present a simple but effective method for detecting outliers in the input stream performed by each node, and another one performed by the representative to detect deviating nodes in its cluster. We show that our approach by applying this method, is still capable of detecting changes in input stream.
- **Uniform utilization of energy resources in sensor network**
  We suggest further optimization methods to our approach to uniformly distribute the usage of energy between the nodes. We cope with the cases of single-node clusters, and changing representatives according to residual energy. This results in a longer lifetime of the whole sensor network as nodes die close to each other.

The remainder of this paper is organized as follows. Section 2 mainly reviews the literature related to the physical clustering problem. Section 3 introduces some formulations and definitions used in our approach. Section 4 describes in detail our algorithm. Section 5 presents the experimental results. We conclude the paper and suggest future work in Section 6.

2. RELATED WORK

In this section we list briefly the related work to our physical clustering problem.

Traditional offline clustering algorithms e.g. [8], [4],[25] can not cope with the streaming and distributed nature of sensor nodes.

Although some **distributed versions of clustering algorithms** were established like SDBDC [11], DFEKM [12], they are still dealing with offline data and can not simply adapted to perform online distributed clustering.

Many algorithms were developed to deal with the **online distributed** clustering of data. EDISKCO [10] is an energy efficient approach for online approximative clustering of sensor data. The Distributed Grid Clustering algorithm [22] is an example of an online 2-layer distributed clustering of sensor data. ELink [16] and the Distributed Single-Pass Incremental algorithm DSIC [24] are two examples on time series clustering of sensor nodes. None of these algorithms considered the possibility of having clusters hidden in subsets of the attributes.

Subspace clustering has been proposed, for today's applications with incising number of given dimensions. Subspace clustering detects clusters in arbitrary projections by automatically determining a set of relevant dimensions for each cluster [20, 15]. Thus, one is able to detect objects as part of various clusters in different subspaces. Recent research has seen a number of approaches using different definitions of what constitutes a subspace cluster [3, 13]. As summarized in a recent evaluation study [19], their common problem is that the output generated is typically huge. In recent subspace clustering algorithms we have focused on tackling redundancy [5, 6, 18]. In contrast, projected clustering assigns each object to a single projection [2, 17]. This strict partitioning of the data into projected clusters can be regarded as extreme redundancy elimination. Projected clustering results in a manageable number of clusters, but is not able to detect overlapping clusters. Both subspace clustering and projected clustering have its focus on offline data outside sensor networks. In contrast, we aim at combining clustering in subspace projections [5, 2] with physical clustering for sensor networks [10].

SERENE [7] is a framework for SElecting REpresentatives in a sensor NEtwork. It uses clustering techniques to select the subset of nodes that can best represent the rest of sensors in the network. In order to reduce communication, rather than directly querying all network nodes, only the representative sensors are queried. In this way the overall energy consumption in sensor network is reduced and consequently sensor network lifetime is extended.
To select an appropriate set of representative sensors, SERENE performs the analysis of historical readings of sensor nodes, in order to find out the correlations both in space and time dimensions among sensors and sensor readings. Sensors may be physically correlated. Sensor readings may be correlated in time. Physically correlated sensors with correlated readings are assigned to the same cluster. Then each cluster performs further analysis in order to select the sensors with the highest representation quality. The last two steps of this process are the same with the steps of clustering process in our algorithm.

Similar to our algorithm, this technique uses density-based clustering algorithm, DBSCAN [8]. Nevertheless, different from our algorithm, in SERENE approach the first stage of clustering process is analysis of historical data for detecting correlations among nodes and sensor readings. Due to restrictions of energy, computational and memory capacity in sensor nodes, this analysis can not be performed by the nodes themselves.

Continuous storing of historical data for all nodes that are spatially correlated, in order to analyze correlation of their readings, requires more memory capacity than a sensor node

possesses. Processing of all the analyses over measurements of sensors to find out correlations, needs high computation resources as well. Moreover, this process requires exchange of attribute measurements between all the nodes that are spatially correlated. This is followed by a high energy consumption in nodes, due to frequent communication and data exchange with more than one node in their clusters. Due to all these restrictions, in this approach sensor nodes can not be self organized into clusters. As a result, this technique is suitable only for those scenarios where nodes operate in a supervised way.

Another difficult part of this technique is related with maintenance of SERENE platform. With passing of time, the readings of sensor nodes change, consequently the same set of sensors may not be anymore correlated with each other, or a new correlation may appear among some other nodes. This change requires a reorganization of nodes in clusters. Reclustering process is followed by additional communication among nodes for updating historical data. This will increase the communication burden and the size of transmitted data will be significantly high. More analyses should be performed over data, meaning more resources will be consumed for computation purposes.
All the above mentioned reasons make this approach expensive in terms of energy and not easy to maintain in cases of continuous clustering applications.

In [23] a Data-Driven Processing Technique in Sensor Networks was suggested. The goal of this technique is to provide continuous data without continuous reporting, but with checks against the actual data. To achieve this goal, this approach introduces temporal and spatio-temporal suppression schemes, which use the in-network monitoring to reduce the communication rate to the central server. Based on these schemes, data is routed over a chain architecture. At the end of this chain, the nodes that are most near to central server send the aggregate change of the data to it.

Snapshot Queries [14] is another approach that introduces a platform for energy efficient data collection in sensor networks. By selecting a small set of representative nodes, this approach provides responses to user queries and reduces the energy consumption in the network. In order to select its representative, each sensor node in this approach builds a data model for capturing the distribution of measurement values of its neighbors for each attribute.

After a node decides which of its neighbors it can represent, it broadcasts its list of candidate cluster members to all its neighbors. Each node selects as its representative that neighbor that can represent it and that additionally has the longest list of candidate cluster members. This is again expensive as all messages are broadcasted and not directed to specific nodes, which might result in repeated broadcasting in case of message loss. Maintaining this model is very expensive in terms of energy, as all nodes needs to exchange all historical readings among each other. In our algorithm, each sensor node maintains a small cache of past measurements of itself for each attribute. And the control messages exchanged among nodes during the initialization phase are directed to specified nodes. As the closest state-of-the-art to our approach, we evaluate our algorithm by comparing it to Snapshot Queries [14].

3. PROBLEM FORMULATION

In this section we formally define the related problems to our algorithm.

3.1 The Representatives Selection Problem

Given a set SN of n sensor nodes $SN = \{sn_1,\ sn_2, ..., sn_n\}$ each measuring a set of attributes $\{a_1, a_2, ..., a_k\}$, an Euclidean distance function $d(sn_a, sn_b) \geq 0$; $\{a, b\} \subset \{1, 2, ..., n\}$ and a real number $\varepsilon > 0$. The problem of selecting representative nodes in SN is to find a subset $R = \{r_1, r_2, ..., r_m\} \subseteq SN$; $m \leq n$ each $r_i \in R$ is representing a set of nodes $D_i = \{sn_{i1}, sn_{i2}, ..., sn_{il}\}$, $D_i \subseteq SN$ and $\forall sn \in D_i$: $d(r_i, sn) \leq \varepsilon$ such that the measurements sensed by all members of D_i are best represented by the measurements of r_i.

Definition 1. (Physical cluster of nodes)
A physical cluster C of sensor nodes is a set D_i with a maximum number of $MaxNds > 0$ nodes represented by the representative r_i such that $\forall sn \in D_i$:

$$\sqrt{(x_{r_i} - x_{sn})^2 + (y_{r_i} - y_{sn})^2 + (z_{r_i} - z_{sn})^2} \leq \varepsilon$$

where ε is the radius of C.

Definition 2. (Spatial and non-spatial attributes)
Each node $sn \in SN$ is defined in each time stamp t by a set of attributes $\{a_{1t}, a_{2t}, ..., a_{kt}, x_{sn}, y_{sn}, z_{sn}\}$ where $\{a_{1t}, a_{2t}, ..., a_{kt}\}$ is a set of non-spatial attributes which represent the measurements of sn at time stamp t and $\{x_{sn}, y_{sn}, z_{sn}\}$ are the spatial attributes of sn.

Definition 3. (Relevant attributes)
Let $\{\mu_1(t), \mu_2(t), \cdots \mu_k(t)\}$ and $\{\sigma_1(t), \sigma_2(t), \cdots \sigma_k(t)\}$ be respectively the mean values and the standard deviations of the non-spatial attributes of l readings of sensor node sn_a at time stamp t, a non-spatial attribute a_m, where $1 \leq m \leq k$, is called a relevant attribute between two nodes sn_a and sn_b at the time t if $X_{mt}(sn_b) \in [\mu_m(t) - 2\sigma_m(t), \mu_m(t) + 2\sigma_m(t)]$ where $X_{mt}(sn_b)$ is the sensor sn_b reading of attribute m at time stamp t.

3.2 The Problem of the ECLUN Algorithm

Given a set SN of n sensor nodes with a set of attributes deployed in an environment for monitoring physical phenomena and a base station to collect these measurements, the general problem of ECLUN algorithm is to decrease the total amount of consumed energy in SN by grouping the nodes of SN into physical clusters C_i where $1 \leq i \leq n$ each represented by a representative r_i with some relevant attributes a_j where $1 \leq j \leq k$ and then sending to the base station either the readings of the relevant attributes of r_i to represent the readings of all members of C_i or the summary of the readings of all members of C_i. The target is to continuously update the base station by all important changes in the sensed phenomena.

4. ECLUN ALGORITHM

In this section we describe in details our approach. We differentiate between two phases of the algorithm. The initialization phase where physical clusters are constructed in an unsupervised way, and the running phase when these clusters are maintained and updates are sent to the representative and the server.

4.1 The Initialization Phase

Algorithm 1 gives an overview of this phase. We will next describe each of these steps in details.

Algorithm 1 Initialization Phase of ECLUN

1. *Caching of initial data*
2. *Detection of geographical neighbors*
3. *Setting relevant attributes*
4. *Estimation of representation quality for each node*
5. *Selection of local representatives*
6. *Load balancing among representatives*

4.1.1 Caching of Initial Data

Each node senses the first l measurements for each attribute and stores them in the cache of data history. These l measurements will be exchanged between nodes, and thus will decide the initial physical clustering of nodes. Therefore, outlier readings must completely be excluded in this phase. We assume that attribute measurements for each sensor are normally distributed. Therefor, nodes during this phase continuously calculate the mean and standard deviation values of their measurements for each attribute. If any new reading falls out the corresponding confidence interval $[\mu - 2\sigma, \mu + 2\sigma]$, it is suspected to be an outlier, and stored in the suspected list with a maximum length of s. If the suspected list was filled within the previous s time stamps, then its readings are considered in the main list, otherwise it is excluded completely from both lists.

4.1.2 Detection of Geographical Neighbors

Every node detects its *geographical neighbors* GN by running spatial queries with radius ε. This is done by broadcasting its ID and spatial attributes and mean values of non-spatial attributes $< ID_n, \mu_1, \mu_2, \cdots, \mu_k, x_{sn}, y_{sn}, z_{sn} >$ within a radius ε, where $\mu_1, \mu_2, \cdots, \mu_k$ are the mean values of the initial l readings of sn for each non-spatial attribute. Thus, every node becomes aware of the geographical coordinates as well as the initial readings of its neighbors, these data are stored in a list GN in each node.

4.1.3 Setting Relevant Attributes

In this step, each node decides the relevant attributes between it and each node in its GN list. According to Definition 3, the node uses the non-spatial readings received in the previous step and the statistics of its own previous l measurements to decide the relevant attributes. The threshold $Min\_Rel\_Attr \leq k$; k is the number of non-spatial attributes, decides the minimum number of relevant attributes between two nodes when one wants to represent the other. Each node excludes from the list of GN, all neighboring nodes with less than Min_Rel_Attr relevant attributes to it. The rest nodes are stored in the *candidate cluster member* CCM list.

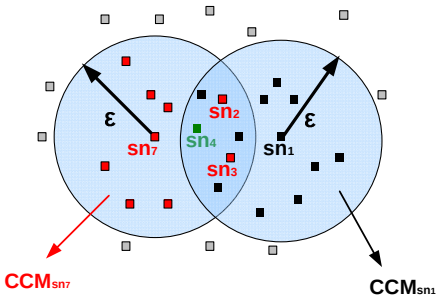


Figure 1: Candidate Cluster Members

In Figure 1, although nodes sn_2 and sn_3 are part of GN of node sn_1, they do not belong to its *candidate cluster members* CCM. Apparently, there are less than Min_Rel_Attr relative attributes between node sn_1 and each of sn_2 and sn_3, so they are both not in the CCM of sn_1.

4.1.4 Estimation of Representation Quality for Each Node

In this step of algorithm, each node analyzes how effective it is in representing its CCM nodes in the network.

Definition 4. (Representation quality)
The representation quality $RepQ$ of node sn when representing its CCM nodes is defined as:

$$RepQ(sn) = (1-\alpha)\frac{\sum_{sn_i \in CCM(sn)}(\varepsilon - d(sn, sn_i))}{\varepsilon \times \mid CCM(sn) \mid} + \alpha\frac{RE_{sn}}{IE_{sn}}$$

Where ε is the maximum radius of the possible cluster that might be represented by sn, $d(sn, sn_i)$ is the distance between sn and any of its CCM, α is a coefficient for weighting the energy, IE_{sn} and $RE_{sn}$are the initial and the residual energy of sn respectively.

According to Definition 4, $RepQ$ is greater when the members of CCM are forming a compact cluster around sn. Closer nodes means less consumed energy and much more similar measurements. Additionally, the residual energy is an important factor, as the possibility will be less later that sn gets soon out of energy. The bigger the value of α, the more the importance of the residual energy factor when selecting representatives.

4.1.5 Selection of Local Representatives

Each node decides whether it will be itself a local representative or it will be represented by any other similar node in its neighborhood. To take this decision, nodes refer to the representation quality parameter.

Each node sn_i broadcasts its $RepQ$ value to every node sn_j that belongs to its CCM. Every node $sn \in SN$ stores the list of *candidate local representatives* CLR, together with the $RepQ$ values received by them, and includes also itself

in this list. The list is ranked in a decreasing order according to *RepQ*. One of the following will happen:

1. If the current node has its own *RepQ* value in the top of this list, it announces itself as a representative.
2. If two nodes have the same *RepQ* value, then the closer node is selected as a representative for the current node
3. Otherwise, node sn is represented by the node which is having the *RepQ* in its CLR

After this step, every node either has chosen only one node as its representative, or is a representative itself. Since representatives announce themselves, each node collects the IDs of representative nodes in its neighborhood, it stores them in an internal list called *neighbor local representatives* NLR.

As we saw when building CCM, we had $Min\_Rel\_Attr \leq k$; k is the number of non-spatial attributes, we adopt this idea from the subspace clustering area [5, 2]. For many given attributes, one can hardly find two sensor nodes that can have similar measurements in all attributes, this will result in a huge number of single-node clusters. To avoid this we relax the representation criteria in such a way that the representative needs only to have some relevant attributes with its represented nodes. Algorithm 2 gives a description of the process of selecting the representative according to the relevant attributes and updating the server with relevant and non-relevant attributes by each node.

Algorithm 2 Selecting representatives per attributes

```
1.  if this_attributeis a relevant_attribute then
2.     Let it be represented by the local representative
       rep_a which is sharing the highest number of
       relevant attributes
3.  else if (other representative rep_b can represent
          this_attribute) then
4.     Some attributes are represented by rep_a others
       by rep_b
5.  else
6.     Let this_attribute be forwarded to server by rep_a
7.  end if
```

4.1.6 Load Balancing Among Representatives

To provide a uniform utilization of energy resources in sensor network, we set a threshold $MaxNds$ for the *maximum number of nodes* that can be represented by one representative. According to that, representatives decide to exclude from its cluster the most distant cluster members. The excluded node then tries to join the nearest representative in its NLR list.

At the end of the initialization phase, physical clusters C are established.

4.2 The Running Phase

The algorithm initiates the communication process only when a state change is detected. Nodes communicate with their representatives only when they detect a state change in the attribute measurements of the event they are monitoring. Similarly, representatives send data to the server only if they detect a state change in the statistics of the measurements collected from all the nodes of their clusters. We have then two possible communication paths: node-representative and representative-server.

4.2.1 Node-Local Representative Communication

Each node sn compares the current measurement values $X_{jt_i}(sn)$ on non-spatial attribute $j = (1 \cdots m)$ sensed at t_i with the mean value $\mu_j(t_{i-1})$ of the l previous measurements values of the corresponding attribute. If $|X_{jt_i}(sn) - \mu_j(t_{i-1})| \leq \delta_j$ where δ_j; $j = (1 \cdots m)$ are the measurements thresholds for attribute j, then a change in the measurements is detected and an update of X_{jt_i} should be sent to the corresponding representative. Otherwise no data is sent to the representative and old measurements sent previously to the representative by sn are used.

4.2.2 Local Representative-Server Communication

During each time stamp in the running phase, the representative executes Algorithm 3. After this, and at the same time stamp, the representative maintains C. It checks whether $X_{jt_i}(sn)$ that it has received from sn falls inside the confidence interval $[\mu_{jC}(t_{i-1}) - 2\sigma_{jC}(t_{i-1}), \mu_{jC}(t_{i-1}) + 2\sigma_{jC}(t_{i-1})]$ for each relevant attribute in C or not. If this was not the case, then sn is temporarily excluded from the t_i statistics. Its readings are saved in a list with a maximum length s. If s was filled within the previous s time stamps with readings of sn, then the representative requests sn to join another physical cluster, and forwards its s readings together with the ID of sn to the server. And sn in turn, searches for a neighboring representative in its NLR list and continues from step 5 in Algorithm 1. The only exception here will be that Min_Rel_Attr threshold does not apply, as nodes try to minimize the number of nodes representing it.

Algorithm 3 Representative running phase

```
1.  while updates are received from nodes at time t_i do
2.     if any attribute is missed then
3.        use t_{i-1} values
4.     for each relevant attribute j do
5.        μ_jC(t_i) = (1/|C|) Σ_{sn∈C}
6.        if |mu_jC(t_i) − μ_jC(t_{i-1})| ≤ ψ_j then
7.           update the server with μ_jC(t_i) and σ_jC(t_i)
8.     end for
9.  end while
```

4.3 Energy Aware Optimizations

We suggest further optimizations for the sake of energy efficiency in our algorithm.

4.3.1 Delegation of Representative Authority

The energy of local representatives decreases rapidly much more than the energy of other nodes in the network.

If a representative runs out of energy, all of its cluster nodes should recluster.This is considerably energy consuming. Furthermore, losing the representative node will cause a big lack of information about the monitored phenomena delivered by the complete cluster. We suggest a uniform utilization of

energy sources in the network, by applying a technique of delegation representative authority.

Each local representative is aware of its residual energy. At the time it notices that its energy capacity is decreased under a certain threshold (for instance: 50% of its initial energy as it started to represent this cluster), local representative requests the residual energy values of nodes in C. The authority of representing the cluster is delegated to the node with the highest residual energy including current representative. If none of the cluster nodes has more residual energy than current representative, then it continues being the representative of its cluster and performs later the same check again.

4.3.2 Optimization in Case of Single Node Cluster

In such a scenario, sensor node has to communicate with distant server for updating only its measurements. To avoid this, each node that is alone in its cluster sends lazily *'join requests'* to its neighbor representatives. Each neighbor representative then checks whether the attribute measurements are relevant to its cluster. Accordingly it might join that cluster or keep representing itself. In case of more acknowledgments, it selects the nearest neighbor representatives. Receiving no-acknowledgment means that the node is selecting different data than its neighbors and will keep representing itself. This might mean that either this node is corrupted or measuring some local event.

5. EXPERIMENTAL EVALUATION

To evaluate the performance of ECLUN, we performed a set of experiments to test the effectivity of each feature of ECLUN, and to compare the performance of ECLUN with the state-of-the-art competing algorithm, Snapshot Queries [14]. We start by describing our real and synthetic datasets in 5.1, then our evaluation methodology for each set of experiments in 5.2, in 5.3 we describe the settings of our experiments and then we conclude this section by discussing the experimental results in 5.4.

5.1 Datasets

We have used three real datasets in addition to one synthetic dataset for evaluating ECLUN. We give a description of each with some of the parameter settings applied with them on both ECLUN and Snapshot Queries. Unless otherwise stated, these parameter settings applies to all experiments.

5.1.1 Real Datasets

Intel Berkeley Research Lab: Intel Lab 1
Out of the 54 nodes readings collected in [1]. Three nodes had a huge number of missing readings, therefore we used the clean readings of 51 nodes each contains 4-parameters readings taken every 31 seconds. The clean processed dataset contained 15730 readings. We have mapped these time stamps into 5 days, 15 hours, 27 min and 10 sec period of time. The network topology was selected to be as close as possible to original nodes topology. When applying this dataset on any algorithm we set the initial energy IE of each node to 295 Joules. We call this real dataset *Intel Lab 1* in our next experiments.

Table 1: Generated events in the synthetic dataset

	Event 1	**Event 2**
Values per dimension	D1{Low} D2{High} D3{Low}	D1{High} D2{Low} D3{High}
Time stamps [From, to]	[0,200], [1000,1100], [10000,11000]	[300,350], [1000,1100], [2000,2500]
Most affected node	Node 2, coordinates: (2,0)	Node 47, coordinates: (4,6)

Intel Berkeley Research Lab: Intel Lab 2
To get more readings, we have excluded 5 nodes from the original Intel lab dataset. This resulted in 23077 healthy readings. For small missed values in between, we have always inserted the last received value instead of later missed readings. Again we set the topology of the network in both evaluated algorithms to be as close as possible to the original topology. When applying this dataset on any algorithm we set the initial energy IE of each node to 10000 Joules. We call this dataset *Intel Lab 2*.

I9 Sensor Dataset
Explanation about this one-dimensional dataset with 40589 readings can be found in [10]. The 16 nodes were randomly inserted to the algorithms without mapping the coordinates of network topology. When applying this dataset on any algorithm we set the initial energy IE of each node to 1100 Joules.

5.1.2 Synthetic Dataset

The synthetic dataset was generated mainly for evaluating the response of each of the competing algorithms to some inserted changes in the monitored phenomena.We generated readings for 49 sensor nodes distributed in one 7x7 grid with 12000 3-dimensional readings for each node. The normally distributed random readings were mainly simulating the humidity, light and temperature attributes sensed by TelosB nodes [21]. The total range of each attribute was divided into three subranges: Low, normal and High. Inserted events are any combination of three ranges, each taken from an attribute. We have generated 2 different in different parts of the network, details about these events are depicted in Table 1.

5.2 Evaluation Methodology

We evaluated ECLUN from three different perspectives:

1. **Evaluating each Feature of ECLUN:** We have tested the effect of each feature of ECLUN by evaluating for every feature two versions of ECLUN, one containing this feature and the other not. The measure was the total number of dead nodes in the whole network with the progress of time.

2. **Energy Consumption:** Two measures were performed to evaluate the energy consumption of ECLUN with that of Snapshot Queries, the total number of dead

nodes in the network, and the total amount of consumed energy in Joules.

3. **Detection of Changes in Input Stream:** We wanted to see the values of readings delivered to the server by the representatives in each algorithm on time stamps where we synthetically inserted events as in Table 1.

5.3 Setup of the Experiments

For evaluating the energy consumption, in all experiments we used the energy model described in [10]. For all experiments of ECLUN, we had the following settings on all datasets: the radius of covered nodes by the range of each node: $\varepsilon = 2$, the maximum number of nodes in one physical cluster: $MaxNds = 4$ and the delegation authority threshold: 50% of initial energy. For **Intel Lab 1** and **Intel Lab 2** datasets, we have selected the number of initial readings $l = 10$, the threshold of relevant attributes for representing $Min\_Rel\_Attr = 2$, the node-representative update thresholds: δ_j; $j = (1 \cdots 4)$ as $(0.2, 0.2, 0.2, 0.2)$ and the representative-server update thresholds: ψ_j; $j = (1 \cdots 4)$ as $(0.2, 0.2, 0.2, 0.2)$. For **I9** dataset: $l = 10$, $Min\_Rel\_Attr = 1$, $\delta_1 = 0.2$ and $\psi_1 = 0.2$. To have fair results, the parameter settings of Snapshot Queries were always identical to that of ECLUN whenever they apply. We set the error threshold T_j; $j = (1 \cdots 4)$ to $(5, 5, 5, 5)$. According to [14], these values deliver the best results in terms of number of participating nodes in each cluster on the one hand, and an accepted representation error on the other hand.

5.4 Experimental Results

5.4.1 Results of Features Evaluation

In each of the following selected two experiments, we test a feature in ECLUN, by comparing the energy consumption of two versions of ECLUN that differ only in including this feature or not.

Node - Representative and Representative - Server Communications

This feature enables the update of the representative or the

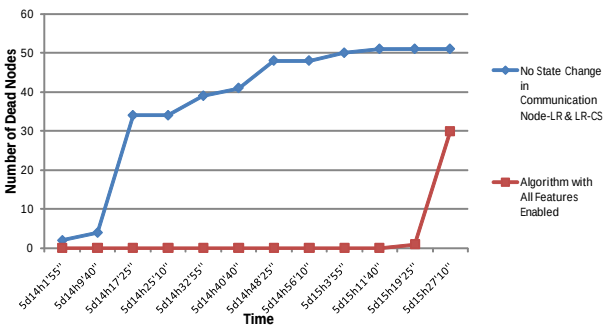


Figure 2: Testing the ECLUN feature of performing the update only when a change is detected using the Intel Lab 1 dataset

server to occur only when a change is detected. Excluding it means that the nodes always communicate with the representative whenever they have a new reading, and the representative in turn always communicates with the server.

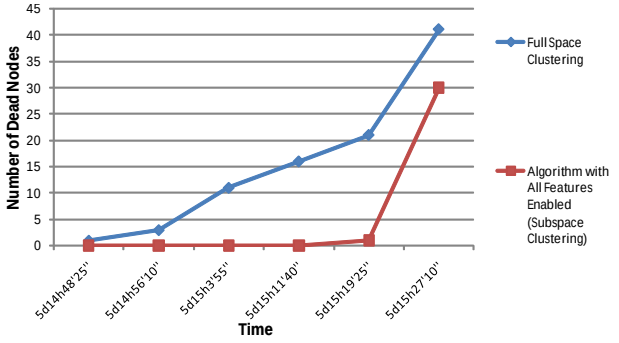


Figure 3: Testing the ECLUN feature of subspace clustering using the Intel Lab 1 dataset

As expected, the results in Figure 2 shows that this feature extends the whole network life time. Without using this features nodes start to die in the network after 5 days, 14 hours, 1 minute and 55 seconds, while by using the first node dies around 1 hour and 20 minutes after that. Additionally, by the end of dataset 21 nodes are still alive when enabling this feature, while all nodes die when disabling it. As we will see in the change detection results, this feature is not delaying important changes.

Subspace Clustering:(Clustering per Relevant Attributes)

Disabling this feature means that a node can only be represented by nodes that are relevant to it in all spaces (attributes). The possibility for nodes to find such a representative in its neighbors will be very low. Which ends with a self representation by the node. As depicted in Figure 3, using this feature delays the death of first node around 31 minutes and increases the number of the still-alive nodes by 11 with the end of the simulation. The impact of the subspace clustering is even stronger with higher dimensions.

5.4.2 Results of Energy Consumption Evaluation

We evaluate here the energy consumption of ECLUN and Snapshot Queries algorithm [14]. Figure 4 depicts the residual energy in Joules of each sensor node after the initialization phase. As shown, the initialization phase of Snapshot Queries consumes more energy than that of ECLUN. This is because of the extensive messages of big sizes that are exchanged between nodes in Snapshot Queries during this phase. Although this initialization phase happens not so often in ECLUN through reclustering, our experiments showed that it happens more likely in Snapshot Queries. This is due to the subspace nature that ECLUN uses. It can be seen also in Figure 4, that the energy consumption in ECLUN is balanced between all the nodes after this phase, in contrast to Snapshot Queries, where selected representatives consumes more energy than others even during this phase.

Figure 5 presents a comparison between two versions of each algorithm. For ECLUN, we used the all-features version and another without the previous features tested in 5.4.1 and without the delegation of authority optimization. For Snapshot Queries, we applied the two forms of changing the representative with the decrease of energy suggested in [14].

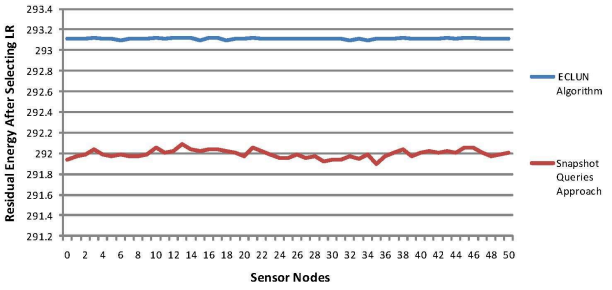


Figure 4: The residual energy in each of the 51 nodes after the process of selecting representatives in ECLUN and Snapshot Queries using Intel Lab 1

Table 2: Total energy consumption of ECLUN and Snapshot queries in Joules

Dataset	Number of Nodes	Number of Readings per Node	**ECLUN** Energy Consumption [Joules]	**Snapshot** Energy Consumption [Joules]
Intel Lab 2	49	23077	**22552.5**	25546.9
I9	16	40589	**13837.1**	14094.9

The first one is similar to ECLUN, where nodes are invited to send their residual energy and the one with the highest residual energy is selected. The other approach randomly selects the next representative, we called this (Snapshot Queries with randomized representatives). The two versions of ECLUN extend considerably the network life time much more than both of the versions of Snapshot Queries. The better version of Snapshot Queries starts to lose nodes around 7 hours and 15 minutes earlier than the normal ECLUN. When the dataset ends, ECLUN has still 21 alive nodes, while the two versions of Snapshot Queries almost lose all of their nodes. Another important feature is that the nodes in ECLUN die close to each other, which yields a better usage of the network resources and more data about observed phenomena. Figure 6 and Table 2 show the efficiency of ECLUN over Snapshot Queries for different sizes of data with different dimensionality.

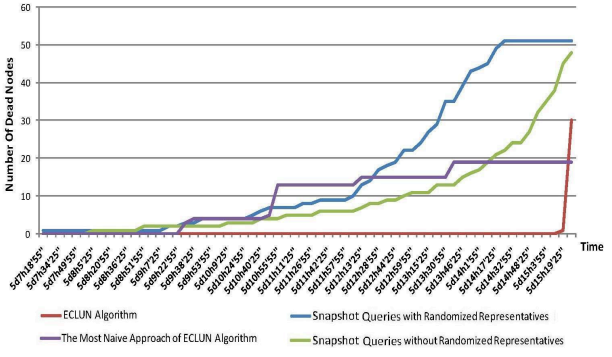


Figure 5: Number of dead nodes in different versions of ECLUN and Snapshot Queries using Intel Lab 1

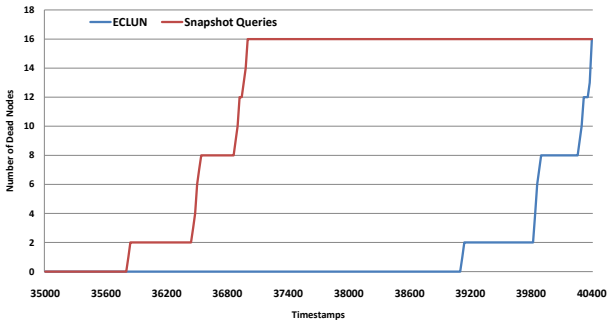


Figure 6: Number of dead nodes using the I9 dataset

5.4.3 Results of Change Detection Evaluation

For evaluating this measure, we used the synthetic dataset. Figure 7 depicts the input events affecting some parts of the sensor network, and the corresponding output sent by ECLUN and Snapshot Queries to the server at the same time stamp. Figures 7(a) and 7(b) show the input and ECLUN output Event 1 at time stamp:11. Snapshot Queries detected this event at time stamp:12 Figure 7(c). Obviously, ECLUN was not only able to detect this event exactly when it appeared, it could also deliver the involved nodes in this event with few false positives. Snapshot Queries detected the change event with a delay of one time stamp, then delivered the data of only one node out of the six involved in Event 1. Figures 7(d), 7(e) and 7(f) describe the input, ECLUN output and Snapshot Queries output at time stamp:1000. ECLUN detected the event at the same time stamp but was less accurate than at time stamp:11. This is due to the fact that at time stamp:11, ECLUN has clustered the nodes according to the first $l = 10$ readings. Event 1 was existing during that interval, and thus detecting it was much more accurate. Snapshot Queries could not detect this event at all. Figures 7(g), 7(h) and 7(i) depict the same sequence for Event 2 at time stamp: 1050.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a novel algorithm for an energy aware physical clustering of sensor nodes. Our algorithm considers both spatial and data similarities when building these physical clusters. Nodes in our suggested approach make use of established data mining techniques like subspace clustering for joining physical clusters according to relevant attributes, and outlier detection for online exclusion of outlying readings. We further suggested a powerful method for the maintenance of the constructed clusters. This enables the network to adapt with different changes of observed phenomena in an unsupervised way, while consuming less energy. We proved the efficiency and effectiveness of our approach through comprehensive experiments.

In the future, we would like to combine our sensor stream data clustering approach (EDISKCO [10]) with this node clustering approach. This can further save energy, and might improve the correctness of approximative solutions applied on stream sensor data. With the huge exchange of data in physical clustering of sensor nodes, security looks like an important issue. We aim at tackling this point by extending our outlier ranking techniques. As an additional require-

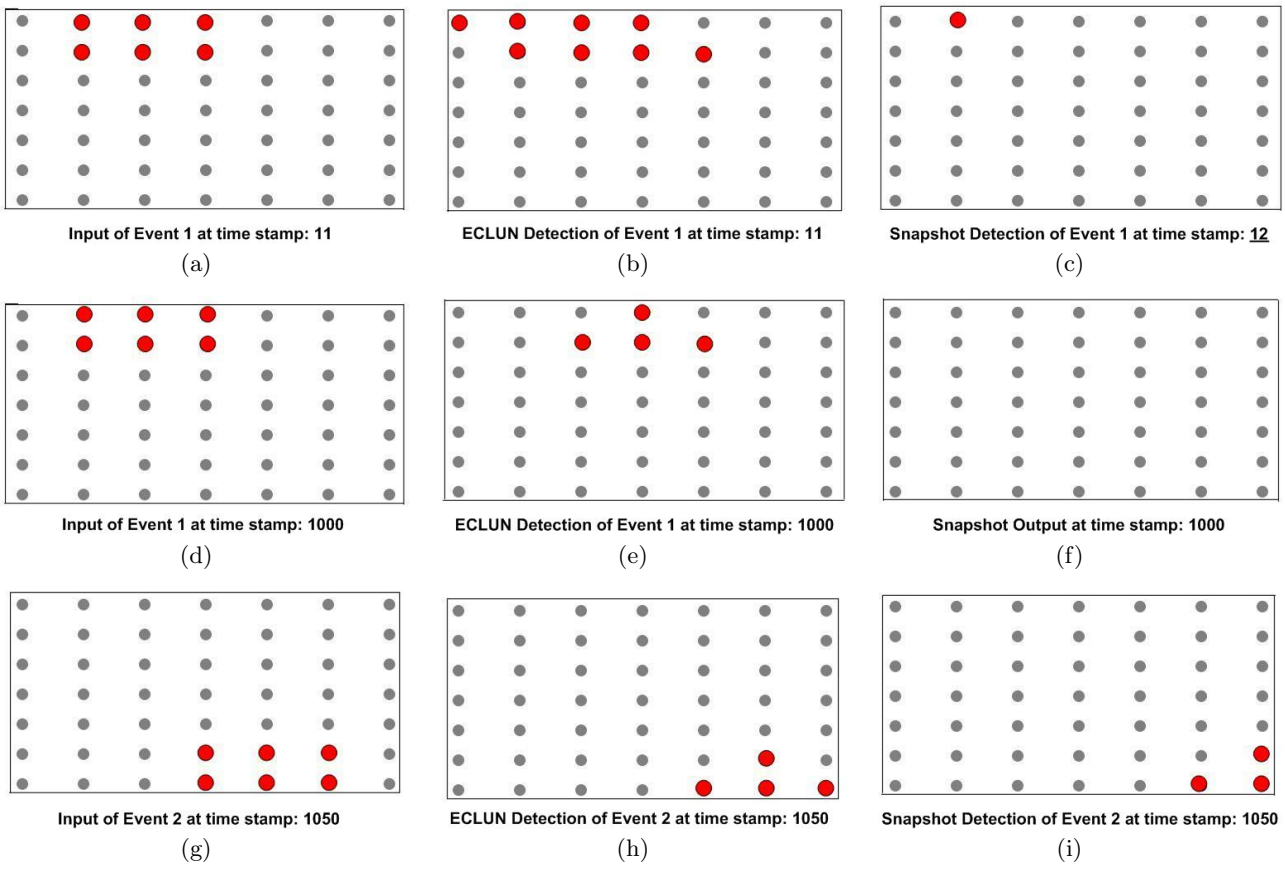


Figure 7: Change detection evaluation using the Synthetic Dataset with 2 inserted events

ment, we would like to extend our approach additionally to consider subspace physical clustering of mobile sensor nodes.

Acknowledgments

This research was funded in part by the cluster of excellence on Ultra-high speed Mobile Information and Communication (UMIC) of the DFG (German Research Foundation grant EXC 89).

7. REFERENCES

[1] Dataset of intel berkeley research lab. In *Online under http://db.csail.mit.edu/labdata/labdata.html*, 2004.

[2] C. Aggarwal, J. Wolf, P. Yu, C. Procopiuc, and J. Park. Fast algorithms for projected clustering. In *SIGMOD*, pages 61–72, 1999.

[3] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD*, pages 94–105, 1998.

[4] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure. In *Proceedings ACM SIGMOD' 99 Int Conf. on Management of Data*, 1999.

[5] I. Assent, R. Krieger, E. Müller, and T. Seidl. DUSC: Dimensionality unbiased subspace clustering. In *ICDM*, pages 409–414, 2007.

[6] I. Assent, R. Krieger, E. Müller, and T. Seidl. INSCY: Indexing subspace clusters with in-process-removal of redundancy. In *ICDM*, pages 719–724, 2008.

[7] E. Baralis and T. Cerquitelli. Selecting representatives in a sensor network. In *In Proceedings of the SEBD*, pages 351–360, 2006.

[8] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[9] A. R. Ganguly, J. Gama, O. A. Omitaomu, M. M. Gaber, and R. R. Vatsavai. *Knowledge Discovery from Sensor Data*. 2008.

[10] M. Hassani, E. Müller, and T. Seidl. EDISKCO: Energy efficient distributed in-sensor-network k-center clustering with outliers. In *In Proceedings of the Third International Workshop on Knowledge Discovery from Sensor Data SensorKDD*, pages 39–48, 2009.

[11] E. Januzaj, H.-P. Kriegel, and M. Pfeifle. Scalable density-based distributed clustering. In *Proceedings 8th European Conference on Principles and Practice of Knowledge Discovery in Databases PKDD*, 2004.

[12] R. Jin, A. Goswami, and G. Agrawal. Fast and exact out-of-core and distributed k-means clustering. *Knowledge and Information Systems*, 10(1):17–40, 2006.

[13] K. Kailing, H.-P. Kriegel, and P. Kröger. Density-connected subspace clustering for

high-dimensional data. In *SDM*, pages 246–257, 2004.

[14] Y. Kotidis. Snapshot queries: Towards data-centric sensor networks. In *Proceeding of the 21st International Conference on Data Engineering, ICDE*, 2005.

[15] H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *TKDD*, 3(1), 2009.

[16] A. Meka and A. K. Singh. Distributed special clustering in sensor networks. In *EDBT 2006, LNCS 3896*, pages 980–1000, 2006.

[17] G. Moise, J. Sander, and M. Ester. P3C: A robust projected clustering algorithm. In *ICDM*, pages 414–425, 2006.

[18] E. Müller, I. Assent, S. Günnemann, R. Krieger, and T. Seidl. Relevant subspace clustering: Mining the most interesting non-redundant concepts in high dimensional data. In *ICDM*, pages 377–386, 2009.

[19] E. Müller, S. Günnemann, I. Assent, and T. Seidl. Evaluating clustering in subspace projections of high dimensional data. *PVLDB*, 2(1):1270–1281, 2009.

[20] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: a review. *SIGKDD Explorations*, 6(1):90–105, 2004.

[21] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks, IPSN*, 2005.

[22] P. P. Rodrigues, J. Gama, and L. Lopes. Clustering distributed sensor data streams. In *ECML PKDD 2008, LNAI. Springer-Verlag*, 2008.

[23] A. Silberstein, R. Braynard, G. Filpus, G. Puggioni, A. Gelfand, K. Munagala, and J. Yang. Data-driven processing in sensor networks. In *Proceedings of the 3rd Biennal Conference on Innovative Data Systems Research (CIDR)*, 2007.

[24] J. Yin and M. M. Gaber. Clustering distributed time series in sensor networks. In *In Proceedings of the Eighth IEEE Conference on Data Mining, ICDM*, 2008.

[25] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. In *Proceedings ACM SIGMOD' 96 Int Conf. on Management of Data*, 1996.

Anomaly Localization by Joint Sparse PCA in Wireless Sensor Networks

Ruoyi Jiang, Hongliang Fei, Jun Huan
Department of Electrical Engineering and Computer Science
University of Kansas
Lawrence, KS 66047-7621, USA
{jiangruoyi, hfei, jhuan}@ittc.ku.edu

ABSTRACT

Principal Component Analysis based anomaly detection approaches have been extensively studied recently. However, none of these approaches address the problem of anomaly localization. In this paper, we proposed a novel approach based on PCA to perform anomaly detection and localization in sensor networks simultaneously. By enforcing the joint sparsity across the Principal Components in the abnormal subspace, we can accurately localize the abnormal sensor nodes from normal nodes. We demonstrate the localization performance in the experimental study on two real world data sets.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation

Keywords

Anomaly Localization, Regularization, Joint Sparsity

1. INTRODUCTION

Anomaly localization in wireless sensor network is an emerging research topic in sensor data analysis. As an important step after anomaly detection, anomaly localization is to identify the specific sensor(s) that contribute to the observed anomaly. There is a wide range of applications of anomaly localization in sensor networks. For instance we need to localize the exact position of damage when we use sensors to monitor the state of buildings such as bridges [17]. In designing protocols against car theft [16], the knowledge of which car is in danger is far more useful than the knowledge that whether a theft happened in a parking lot. Anomaly localization is also a critical step in nature disaster monitoring including flooding and forest fire monitoring [1, 4]. A accurate anomaly localization technique helps us quickly localize the anomaly and recover the abnormal situation.

PCA is a widely investigated unsupervised anomaly detection technique. In recent years, an extensive research has been done to demonstrate the utility of PCA in anomaly detection [1, 6, 10, 11]. However, a fundamental problem of PCA is that the current PCA technology can not be used for anomaly localization, as claimed in [15]. The root cause of the problem is that each principal component is a linear combination of features from the whole network. This property makes PCA difficult to identify the particular sensor(s) that contribute significantly to the abnormal subspace spanned by a few principal components.

In this paper, we propose a novel PCA based anomaly localization method, named as joint sparsity PCA (JSPCA), to fill in the perceived technology gap. The key observation of JSPCA is that PCA may be formulated as a regularized smoothing technique through which we identify a low dimensional approximation of a high dimensional data, as originally studied in [19]. Utilizing recently developed sparse learning techniques such as L_1 regularization, we could obtain a sparse representation of principle components, each of which is a linear combination of data from just a few sensors and hence achieve "localization" and "detection" simultaneously. We have provided theoretical insights about the power of our method. Extensive experimental study also shows that JSPCA is able to single out those sensor nodes that contribute most to the abnormal data. To the best of our knowledge, this is the first work using joint sparse PCA to detect and localize anomaly simultaneously.

The remainder of the paper is organized as follows. In Section 2 we present the related work of anomaly localization in data analysis domain and introduce some basic knowledge of PCA when applying to anomaly detection. In section 3, we introduce the formulation of JSPCA and illustrate why JSPCA can localize the anomaly. Our experiments in section 4 demonstrate the effectiveness of the approach on on two real world data set, followed by conclusion in section 5.

2. RELATED WORK

2.1 Anomaly Localization

A few anomaly localization techniques based on data mining techniques have been proposed recently [5, 7, 8]. Ide et al. proposed a method [8] called stochastic nearest neighborhood. The assumption of this method is that the neighborhood graph of each node is almost invariant under normal

SensorKDD'10, July 25, 2010, Washington, DC, USA

conditions. The abnormal score of each node is measured by how much its neighborhood network changes. A higher abnormal score indicates a higher probability that it was an abnormal node. In [7], Ide et al. extended their previous work of neighborhood graph to localize anomalous node, while focusing more on the advanced method for constructing the neighborhood network for each node.

In [5], Hirose et al. also used the principle of neighborhood preservation to measure the node that deviated most compared with other nodes. They formulated the detection and localization tasks as a compression of eigen equation. By conducting eigen equation compression, the whole network is divided into three clusters from the view point of each node: the node itself, the nodes with higher correlation, and the nodes with lower correlation. The correlation between clusters can be used as a metric for cluster structure. By tracking how significantly the cluster structures changed, the abnormal score was computed for each node. The abnormal node was the one with the highest abnormal score.

The disadvantage of these two methods is that both of them have a neighborhood/cluster parameter and the parameter is difficult to tune. In our method, we present a totally different formalization and show that we can directly compute the abnormal score to localize the abnormal node(s) without constructing a neighborhood for each node.

2.2 PCA in Anomaly Detection

As a widely investigated unsupervised technique for anomaly detection, PCA was first proposed to detect network volume anomaly by Lakhina et al. [10]. After that, a lot of research proposed extensions to the initial method. Huang et al. [6] develop a distributed PCA anomaly detector by installing a local filter in each node. Recently, Brauckhof et al. [1] considers both the temporal and spatial correlation of the data by extending PCA to Karhunen-Loeve Expansion (KLE) and solve the sensitivity problem of PCA proposed by Ringberg et al. [15].

PCA detects anomaly of the whole network by the construction of normal and abnormal subspace. The normal subspace is extended by the top few principal components while the abnormal subspace is extended by the last few. The normal subspace can capture the behaviors common among all the nodes, at the same time the abnormal subspace can be used to represent the noise and abnormal behaviors which happen to a small part of the nodes. The basic idea of PCA is that it makes use of the correlation of the data.

3. METHODOLOGY

In this section, we design the framework of joint sparse PCA for anomaly localization. The approach is divided into two phases: abnormal data projection and anomaly localization via joint sparsity construction. At the first phase, we construct normal and abnormal subspace the same as previous research on PCA based anomaly detection. After that, we project original data into the abnormal subspace and perform joint sparse PCA on the projected data in the abnormal subspace to localize abnormal nodes. Our key observation is that the projection of normal data onto the abnormal subspace will have very limited even no contribution to the nonzero entries of the Principal Components (PCs). Therefore, by enforcing joint sparsity for the entries across PCs corresponding to each sensor nodes, we can effectively and accurately locate the anomalies. Before formally presenting the approach, we provide the notation for this paper bellow.

3.1 Notation

In this paper, we use capital letters, such as X to represent a matrix and bold lowercase letters such as $\mathbf{x}$ to represent a vector. The columns of a matrix $X \in R^{n\times p}$ are $\mathbf{x_1}$ through $\mathbf{x_p}$, while the rows are given (as vectors) by $\tilde{\mathbf{x}}_\mathbf{1}^T$ through $\tilde{\mathbf{x}}_\mathbf{n}^T$. A Greek letter such as λ is a scalar to represent a Lagrangian multiplier. $< A, B >$ represents matrix inner product where $< A, B >= Tr(A^T B)$. We use $||.||_F$ to denote Frobenius norm of a matrix, where $||X||_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{p} x_{ij}^2}$. $||\mathbf{x}||_2 = \sqrt{\sum_{i=1}^{p} x_i^2}$ denotes the L_2 norm of $\mathbf{x} \in R^p$. Unless state otherwise, all vectors are column vectors.

3.2 Anomaly Localization with PCA

PCA is often introduced as a variance maximization technique. In PCA, we aim to identify a set of vectors sequentially to maximize the variance of the data when the data are projected to those vectors. These identified vectors are known as principle components. Such explanation has been widely used when applying PCA to anomaly detection [1, 11, 6]. This definition leads to a sequential formulations that compute principal components one at a time.

Another approach to interpret PCA is that PCA finds some bases, such that the projections on these bases represent a low dimension approximation of data which minimizes the reconstruction error (i.e. the squared distance between the original data and its "estimate"). Following this, we could estimate the first k principle components simultaneously without ranking the corresponding eigen values of the covariance matrix of X.

Given the data matrix $X \in R^{n\times p}$, where n is the number of observations during a period of time and p is the number of nodes. X can be represented as $X = [\tilde{\mathbf{x}}_\mathbf{1}^T, \tilde{\mathbf{x}}_\mathbf{2}^T, \cdots, \tilde{\mathbf{x}}_\mathbf{n}^T]^T$, where $\tilde{\mathbf{x}}_\mathbf{i} \in R^p$ is a vector from all nodes at a given time stamp i. Assuming data matrix X has been centered along n observations, PCA can be formalized by the second interpretation as:

$$\begin{aligned} &\text{argmin}_{\|\mathbf{v}\|_2=1} \frac{1}{n}\sum_{i=1}^{n} \|\tilde{\mathbf{x}}_\mathbf{i} - \sum_{j=1}^{k} \mathbf{v_j}\mathbf{v_j}^T \tilde{\mathbf{x}}_\mathbf{i}\|_2^2 \\ &\textstyle\sum_{j=1}^{k} \mathbf{v_j}\mathbf{v_j}^T \tilde{\mathbf{x}}_\mathbf{i} = \hat{\tilde{\mathbf{x}}}_\mathbf{i}, k \le p \end{aligned}$$

Here $\hat{\tilde{\mathbf{x}}}_\mathbf{i}$ is an approximated data of $\tilde{\mathbf{x}}_\mathbf{i}$ with a lower dimension k. $\mathbf{v_i}$ is the ith principal component with the orthogonal property and unit norm. When $k = p$, $\tilde{\mathbf{x}}_\mathbf{i} = \hat{\tilde{\mathbf{x}}}_\mathbf{i}$.

The solution to find the principal components is equivalent to computing the Singular Value Decomposition (SVD) of X:

$$X = UDV^T \tag{1}$$

The ith column of V is the ith principal component $\mathbf{v_i}$, $V = [\mathbf{v_1}, \mathbf{v_2}, \cdots \mathbf{v_p}]$. D is a diagonal matrix with the ith singular value equal to the ith diagonal element λ_i. Since V is also a unitary matrix with the property $V^T V = I$, (1) can be

represented by multiplying V on both sides as:

$$XV = UD$$

Let $\mathbf{u_i}$ the ith column of matrix U, for each principal component $\mathbf{v_i}$, the projection of data matrix X onto V can be represented as:

$$X\mathbf{v_i} = \sum_{j=1}^{p} \mathbf{x_j} v_{ij} = \mathbf{u_i}\lambda_i \tag{2}$$

where $\mathbf{x_j}$ is the jth column of X.

From (2), we can see that the projection is a summation of $\mathbf{x_j}$, which is the data from the jth node weighted by v_{ij}. If v_{ij} is zero, $\mathbf{x_j}$ has no projection onto $\mathbf{v_i}$. Therefore, we can build a connection between the original data stream and the entries in the principal components.

By the previous work on standard PCA based anomaly detection [1, 6, 11], the last $p-k$ principal components $\{\mathbf{v_i}\}_{i=k+1}^{p}$ are used to represent the abnormal subspace [10]. If the jth entry of $\mathbf{v_{k+1}}$ is zero, the data stream from the jth node has no projection (zeros) along the direction $\mathbf{v_{k+1}}$. If the jth entries from $\mathbf{v_{k+1}}$ to $\mathbf{v_p}$ are all zero, the data from jth node can be projected to the normal subspace $[\mathbf{v_1} \dots \mathbf{v_k}]$ completely without any loss. In such a situation, we can determine that the jth node is a normal node.

Our key insight of the anomaly localization is that once the anomaly is detected, we check the ith entries across the principal components in abnormal subspace: if all of them are (close to) zero, the corresponding node i is an innocent node which is not responsible for the abnormal event.

However, in most situation, we cannot observe a joint zero (sparse) across the abnormal subspace $[\mathbf{v_{k+1}}, \cdots, \mathbf{v_p}]$, if it is directly generated from PCA. For the most cases, the jth entry in one PC is close to zero, while in another PC is a large absolute value. Furthermore, the noise is expressed in the abnormal subspace as well, which causes the simultaneous zero entries even more difficult to achieve. Therefore, using PCA directly in anomaly localization is not practical in most situations.

In order to overcome the challenge to get a joint sparsity in abnormal subspace, we propose joint sparse PCA (JSPCA). JSPCA is an extension of PCA with the regularization to constrain the entries in the same position of principal components to share the sparsity pattern. Sparsity can enforces the unimportant entries to be zero or close to zero, which releases the influence of noise. Thus, the abnormal node will be located by a series of greater value across the abnormal principal components. Therefore, we can efficiently localize the anomalous node(s).

3.3 Joint Sparse PCA

Motivated by the formalization of sparse PCA in [19], we propose a new regularization framework with joint sparsity on bases for each node across the last $p-k$ principal components. The new formalization has connection to multi-task feature learning with $L_{1,2}$ [12] and $L_{1,\infty}$ [2] regularization, in which the regularization imposes joint sparsity for the weights of each individual feature across all the tasks. The difference is that our framework is totally unsupervised and the sparsity is imposed on a part of the bases ($\{\mathbf{v_i}\}_{i=k+1}^{p}$).

As indicated before, we suppose the principal components are $V = [\mathbf{v_1}, \mathbf{v_2}, \cdots, \mathbf{v_p}]$ computed from standard PCA on data $X_{n\times p}$, we select top k principal components as normal subspace denoted by $Vn_{p\times k} = [\mathbf{v_1}, \mathbf{v_2}, \cdots, \mathbf{v_k}]$. Following the method in [6, 11], we calculate abnormal subspace as $I - VnVn^T$ and then project original data X to the abnormal subspace:

$$X' = X(I - VnVn^T) \tag{3}$$

so that the projected data keeps the abnormal information only. For the projected data, we perform PCA with joint sparsity to localize the anomalous nodes in the abnormal subspace.

Similar to the formalization of sparse PCA in [19] (equation 3.12) with a different regularization scheme rather than elastic net penalty [18], we consider the following optimization problem:

$$\begin{array}{ll} \min_{A,B} & \sum_{i=1}^{n} ||\tilde{\mathbf{x}}_\mathbf{i}' - AB^T\tilde{\mathbf{x}}_\mathbf{i}'||_2^2 + \lambda \sum_{j=1}^{p} ||\tilde{\mathbf{b}}_\mathbf{j}||_2 \\ \text{s.t.} & A^TA = I_{(p-k)\times(p-k)} \end{array} \tag{4}$$

where $\tilde{\mathbf{x}}_\mathbf{i}'$ is the ith row of X', λ is a scalar controlling the sparse penalty, $A, B \in R^{p\times(p-k)}$ and A is the basis and B is the loading matrix. The regularization part $B_{1,2} = \sum_{j=1}^{p} ||\tilde{\mathbf{b}}_\mathbf{j}||_2$ is a $L_{1,2}$ penalty which enforces joint sparsity for each node across the principal components. If $A = B$, then the minimizer under the orthogonal constrain is exactly the principal components of ordinary PCA. Based on the theorem 3 in [19], the minimizer B^* of (4) is proportional principal component of X'.

3.4 Optimization algorithms

We propose an algorithm to solve (4) based on the work of [2, 12, 19]. The algorithm solves A, B iteratively and alternatively.

A **given** B: If B is fixed, we directly use the result from [19] because we can ignore the regularization part in 4. Now 4 degenerates to

$$\begin{array}{ll} \min_{A} & \sum_{i=1}^{n} ||\tilde{\mathbf{x}}_\mathbf{i}' - AB^T\tilde{\mathbf{x}}_\mathbf{i}'||_2^2 = ||X' - X'BA^T||_F^2 \\ \text{s.t.} & A^TA = I_{(p-k)\times(p-k)} \end{array} \tag{5}$$

The solution is obtained by a reduced rank form of Procrustes rotation. We compute the SVD of $(X'^TX')B$ to obtain the solution:

$$\begin{array}{rcl} (X'^TX')B & = & UDV^T \\ \hat{A} & = & UV^T \end{array} \tag{6}$$

See [19] for more information.

B **given** A: On the other hand, if A is fixed and let $f(B) = ||X' - X'AB^T||_F^2$, it is easy to verify that f is a convex and smooth function over $B \in R^{p\times(p-k)}$. (4) can be reformulated as:

$$\min_B f(B) + \lambda \sum_{j=1}^{p} ||\tilde{\mathbf{b}}_\mathbf{j}||_2 \tag{7}$$

We adopt the framework in [2, 12] to solve (7). Since $f(B)$ is a smooth and convex function but the regularization part is nonsmooth (non-differentiable), we apply Nestrerov first order (gradient) method [13] with $O(1/t^2)$ convergence rate where t is the number of iterations, and perform Euclidean Projection onto the $L_{1,2}$ ball for each gradient update step.

First we define the generalized gradient update step given B_t at iteration $t+1$ as following:

$$\begin{aligned} Q_L(B, B_t) &= f(B_t) + < B - B_t, \nabla f(B_t) > \\ &\quad + L/2||B - B_t||_F^2 + \lambda||B||_{1,2} \\ q_L(B_t) &= \text{argmin}_B \, Q_L(B, B_t) \end{aligned} \tag{8}$$

Where L is the lipschitz Constant. We follow the framework proposed in [2] and simplify (8) to

$$\begin{aligned} q_L(B_t) &= \text{argmin}_B(\tfrac{1}{2}||B - C||_F^2 + \tilde{\lambda}||B||_{1,2}) \\ &= \text{argmin}_{\tilde{\mathbf{b}}_1, \cdots, \tilde{\mathbf{b}}_\mathbf{p}} \textstyle\sum_{i=1}^{p} (\tfrac{1}{2}||\tilde{\mathbf{b}}_\mathbf{i} - \tilde{\mathbf{c}}_\mathbf{i}||_2^2 + \tilde{\lambda}||\tilde{\mathbf{b}}_\mathbf{i}||_2) \end{aligned} \tag{9}$$

where $C = B_t - \frac{1}{L}\nabla f(B_t)$ and $\tilde{\lambda} = \lambda/L$.

By the additivity of (9), we decompose (9) into p subproblems. For each subproblem:

$$\text{min}_\mathbf{b} \, \frac{1}{2}||\mathbf{b} - \mathbf{c}||_2^2 + \tilde{\lambda}||\mathbf{b}||_2 \tag{10}$$

By forming the Lagrangian dual form, the analytical solution of (10) is

$$\mathbf{b}^* = \begin{cases} (1 - \frac{\tilde{\lambda}}{||\mathbf{c}||_2})\mathbf{c} & ||\mathbf{c}||_2 > \tilde{\lambda} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

With (10), the problem of Euclidean projection defined in (8) can be solved efficiently. Therefore, we compute the gradient of f at current estimate B_t and perform Projected subgradient to find the next update B_{t+1}. We repeat this iteratively until converges. Refer to [2, 3, 12] therein for more information about Euclidean projection and Nesterov algorithm.

We summarize what is briefly discussed previously in the algorithm called JSPCA. Given the data $X \in R^{n \times p}$, number of Principal Components of normal subspace k and regularization parameters λ, we first project X into abnormal subspace, then perform joint sparse PCA in the abnormal subspace on the projected data.

Algorithm 1 JSPCA(X, λ, k)

1: Perform PCA on X and obtain Principal Components $V = [\mathbf{v_1}, \mathbf{v_2}, \cdots, \mathbf{v_p}]$;
2: $V_n := [\mathbf{v_1}, \cdots, \mathbf{v_k}]$; $X' := X(I - V_n V_n^T)$;
3: $A := [\mathbf{v_{k+1}}, \cdots, \mathbf{v_p}]$; $B := 0$;
4: **while** not converge **do**
5: Update B given A defined in (7) using (8), (9), (10), (11);
6: Update A given B in (5) using (6);
7: **end while**
8: return B;

4. EXPERIMENT

In this section, we evaluate our proposed approach joint sparse PCA (PCA) on two real world data sets. One is the Sun SPOTs data set collected from a short haul trial and the other one is a benchmark data set. We demonstrate that our method can simultaneously detect anomalies and localize the anomalous nodes with a high accuracy. To perform the comparison to the state of art, we implement two localization algorithms: stochastic nearest neighbor (SNN) [8] and eigen equation compression (EEC) [5]. All the experiments are conducted in Matlab on a desktop with 6 GB memory and Intel core i7 2.66 GHz CPU.

4.1 Data sets

4.1.1 *Sun Spot Sensor Data Set*

In the experiment, we use wireless Sun Small Programmable Object Technologies (SPOTs) [14] to collect data for transport chain security validation. A Sun SPOT contains a 3-axis accelerometer, a light sensor and a temperature sensor as well as a 180MHz 32-bit ARM920T core processor with 512K RAM and 4M Flash memory, a 2.4GHz radio with an integrated antenna on the board, and a 3.7V rechargeable, 750 mAh lithium-ion battery.

The sensor data was collected during a car trial along the campus of University of Kansas under a noisy environment. Seven Sun SPOTs were fixed in separated boxes and loaded on the back seat of a car. During the trial, each sensor recorded the magnitude of accelerations along x,y,z axis, temperature and luminance with a sample rate 3.33Hz. To collect the data for the trip, the SPOTs were programmed to continuously read and aggregate the sensor value for each sensor. We used the overall acceleration $(x^2 + y^2 + z^2)^{\frac{1}{2}}$ as the feature to detect the designed anomalous events with our anomaly detection and localization algorithm.

The following plot shows the collected data of acceleration magnitude readings for each of the seven SPOTs. The time of the events are marked in the plot as well.

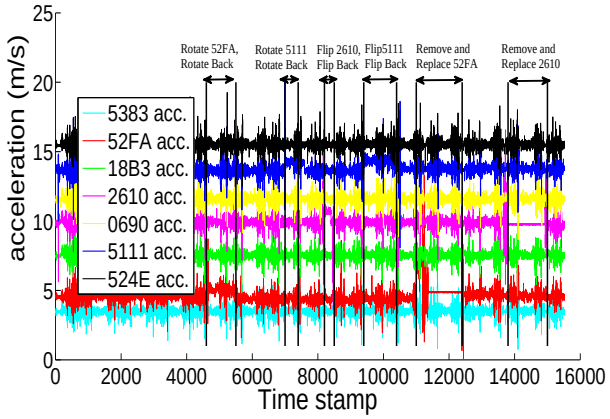


Figure 1: Sun Spot Sensor Data for Entire Trip

The plot shows around 16,000 samples we collected for almost the entire trip, including all the events we simulated. In the plot, each curve shows the overall accelerations for each of the seven Sun SPOTs, each in a different color and scaled and shifted up so that each could be seen along. The black vertical lines show the recorded times of key events. During the whole trip, we simulated box removal and replacement, box rotation and flipping. Each event was repeated twice, in around 5 minutes interval. The whole experiment last about 1 hour.

4.1.2 *Motor Current Data Set*

The Motor Current Data is the current observation generated by the state space simulations, which is available at UCR Time Series Archive [9]. The anomalies are the simulated machinery failure in different components of a machine. The current value was observed from 21 different motor operating conditions,including one healthy operating mode, 10 broken bars and 10 broken end-ring connectors.

For each motor operating condition, there are 20 time series, each with a length of 1,500 samples with the sample rate 33.3 kHz. Therefore, there are 20 normal timer series and 400 different abnormal time series altogether.

In our evaluation, we constructed a data matrix with normal and abnormal data to detect anomalies happened to the Motor Current data set. This data matrix contains 20 time series, each of which has 1500 samples: $\{x_i(t)\}_{i=1:20}^{t=1:1500}$. We single out all the 20 time series data under normal operation, and then replaced time series $x_6 \sim x_{10}$ and $x_{16} \sim x_{20}$ at $1000 \leq t \leq 1500$ by 10 anomalous time series randomly sampled from 400 time series in the 20 abnormal operations. Figure2 shows parts of normal and abnormal current readings. Our object is to detect the anomalies and localize the anomalous currents(here they are No. 6 to No. 10 and No. 16 to No. 20) simultaneously.

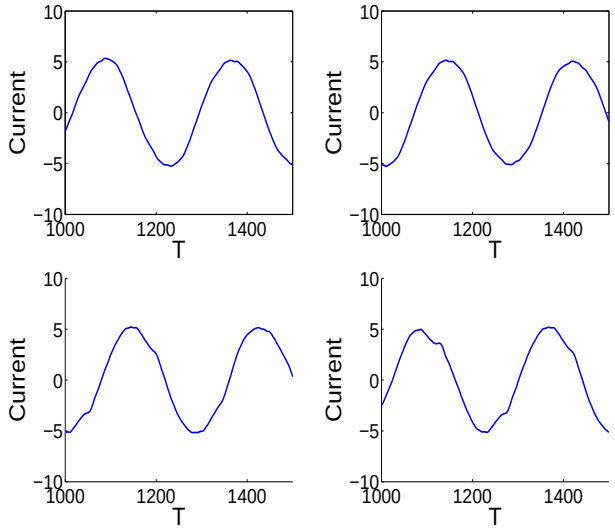


Figure 2: Time Series Motor Current Data. Top Row: Normal Time Series; Lower Row: Abnormal Time Series

4.2 Experiment Protocol

In this subsection, we provide the experimental protocol regarding the parameter selection and performance evaluation. For the EEC method, there is only one parameter: number of clusters c. For SSN, the number of neighbors k is the parameter. In our experiment, we use $c = 3$ and $k = 2$ as provided in their papers.

For our own method, we have a regularization parameter λ for localization. For each data set, we single out a period of time series containing one event to select λ which gives the best $effectivenes$ of localization defined as:

$$eff = \min\{s_i\}_{i\in A} - \max\{s_i\}_{i\in N} \qquad (12)$$

Where s is the vector of normalized scores for all the nodes, A is the set of anomalous nodes and N is the set of normal nodes. The key insight within eff is that the node with a small score among all the abnormal nodes is the most likely to be misclassified as a normal node. Conversely, the normal node with a large score is prone to be judged as abnormal. Therefore, we use Equation (12) to measure the effectiveness of localization on the whole network. The larger of eff, the more effective of localization method is. Within training data, we tune the best λ arranging from $2^{-10}, 2^{-9}, \cdots, 2^{10}$ by criteria in (12). We use the tuned parameter λ for testing on the rest of the data after the training event.

For the Sun SPOTs data set, we choose the first event (rotation of Sensor #2) as the training data. Since the abnormal events of our collected sensor data are very limited, we cannot perform cross validation within training data, while just perform the procedure described above one time. For Motor Current data set, both training and testing data were generated by the same step aforementioned in Sun SPOTs data. We perform 5 fold cross validation within training data to get the best λ.

Since anomaly detection of our method is directly inherited from standard PCA based anomaly detection algorithm [10], we only focus on localization. It is nontrivial to compare the localization performance because different methods use different abnormal score criteria. In our experiment, we normalize the score for each sensor between 0 and 1 for each method. A significantly high score indicates an abnormal state of a sensor.

4.3 Experimental Results

4.3.1 Anomaly localization with Sun SPOTs Data

In this subsection, we give the localization performance on Sun SPOTs data. We use the first 3 Principal components(PCs) to represent the normal subspace, and the rest 4 PCs as the abnormal subspace. In Figure 3, we plot the abnormal subspace composed of the four principal components with joint sparsity learned from the projected data on the rest 4 PCs. Each column along the x-axis is one PC in the abnormal subspace and each row corresponds to one sensor. The lighter color represents larger values of the entries in these abnormal PCs. Each abnormal PC is normalized to make the largest entry as one. The components with black color indicates a larger value corresponding to the node contributing more to the anomaly, while the whiter ones indicate the entries equal or close to zero with less contribution to the abnormal subspace. In Figure 3, it is obvious that the sixth row corresponding to the sensor #6, has a significantly higher value compared with all the other sensors, which are almost all 0s. We can see that joint sparsity on the principal components of the projected data on the abnormal subspace is capable of localizing the abnormal node.

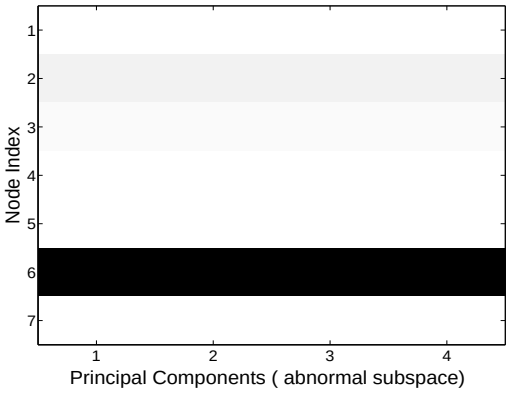


Figure 3: Abnormal Subspace

We also plot the the abnormal scores for the other events in the upper row of Figure 4, for all the seven sensors. Each figure corresponds to a different event. As shown, the abnormal score of the anomalous node is significantly higher than that of the other normal nodes.

For comparison, we show the localization performance of EEC [5] and SNN [8] in the middle and lower low in Figure 4. In the upper row of Figure 4, we show the abnormal score computed by SNN with the neighborhood graph size $k = 2$. We compared the difference of two neighborhood graph for each node between normal and abnormal time stamps and computed the abnormal score in [8]. The result shows that the abnormal score cannot reflect abnormal event. For example, in the rotation event of node 6, there is nearly no difference between normal (especially # 3) and abnormal node. In our experiment, we found that the score is quiet sensitive to the choice of neighbor size k, while our method is pretty stable and we will demonstrate that in section 4.3.3.

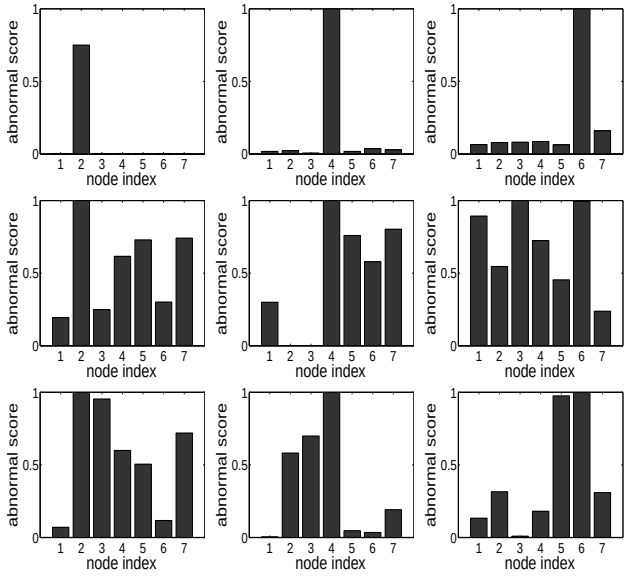


Figure 4: Upper Row: JSPCA. Middle Row: Stochastic Nearest Neighbor. Lower Row: Eigen Equation Compression. From left to right: Abnormal Score for Removal and Replacement Event of Node #2. Flipping Event of Node #4. Rotation Event of Node #6.

In the lower row of Figure 4, we show the localization performance of EEC. Following [5], we clustered the whole network into 3 groups for each sensor. The abnormal score was measured as the change of such group information for each time stamp. We plotted the average abnormal score within each event on the second row of Figure 4. We can observe that EC has the same problem as in SNN. The difference between the scores of normal and abnormal nodes is too small to distinguish the abnormal nodes.

4.3.2 Anomaly localization with Motor Current Data

In this subsection, we evaluate the localization performance of JSPCA on the Motor Current Data Set. We use 5 PCs to represent normal subspace, while the left 15 to represent the abnormal subspace. In the first two rows of Figure 5, we show four situations by randomly selecting four abnormal current patterns. We can observe that JSPCA effectively localizes all of the abnormal patterns. As shown in Figure 5, the abnormal scores from the 6th to the 10th, and from the 16th to the 20th nodes are much higher than the normal nodes. We can easily visualize and separate the normal and abnormal nodes although there exist some fluctuations for different situations.

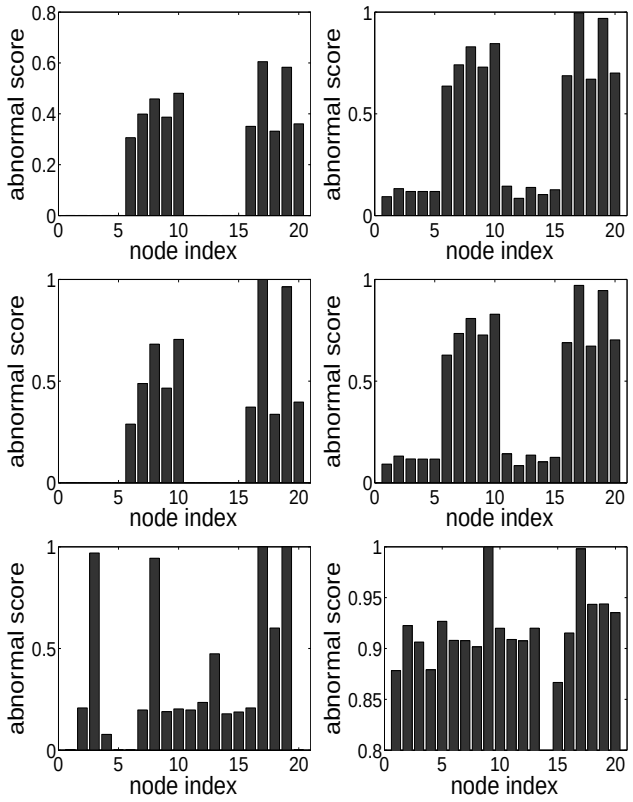


Figure 5: First Two Rows: JSPCA. Last Row: Left: Stochastic Nearest Neighborhood. Right: Eigen Equation Compression with 3 Clusters

Figure 5 also shows the performance of the two compared method on Motor Current data. We computed the average abnormal score for each node during the interval $t > 1000$. From the figure, it is difficult to distinguish the normal nodes($1 \sim 5, 11 \sim 15$) from the abnormal nodes($6 \sim 10, 16 \sim 20$). The score values of abnormal nodes $17 \sim 20$ are a little higher but still close to the score of normal nodes. For the other abnormal nodes, we cannot determine whether they are anomalous nodes based on the abnormal score. Compared with these methods, our algorithm is more effective to localize abnormal nodes.

4.3.3 Robustness on Parameters

Since we have a parameter λ in the joint sparse PCA formalization, we evaluate the robustness of our method by changing different parameter values on Sun SPOTs data, although this parameter can be tuned from training data. In Figure 6, we show the effectiveness of localizing the removal and replacement event of node 2 by varying λ from 2^{-5} to 2^5. From Figure 6, we can observe that the eff is always above 0, which demonstrates that our method can effectively distinguish abnormal nodes from normal ones over a wide range of λ.

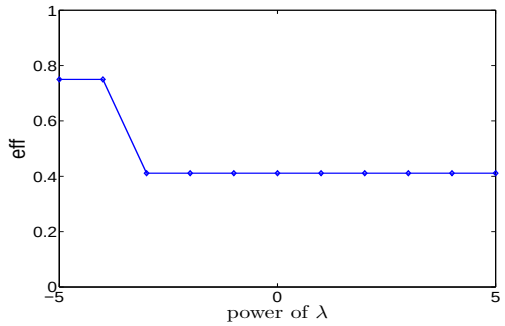


Figure 6: Sensitivity of λ for the Removal and Replacement Event of Node 2

4.3.4 *Importance of joint sparse regularization*

To further study the role of joint sparse regularization in localization, we analyze the performance of standard PCA, in which $\lambda = 0$. The detailed abnormal score for the event that removal and replacement of Node No.2 is shown in Figure 7. We can see that eff is a negative value for standard PCA, which claims that the maximum score is not from the abnormal node. For example, the abnormal score of node 5 is a little higher than node 2 and the abnormal score of node 4 is very close to node 2, all of which undermine the performance of localizing the abnormal node 2. Compared with Figure 7, the first figure in Figure 4, where the abnormal score was computed with joint sparsity, is much clear to localize the abnormal node.

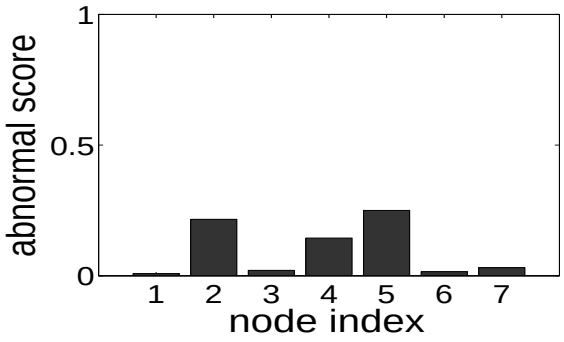


Figure 7: Standard PCA without Regularization($\lambda = 0$)

5. CONCLUSIONS AND FUTURE WORK

Previous work on PCA based anomaly detection claimed that PCA cannot be used for anomaly localization, but in our paper, we proposed a novel approach: joint sparse PCA (JSPCA) in abnormal space to localize anomaly in sensor network. By bridging the sensor node and the corresponding entries in Principal Components, we enforce joint sparseness on PCs to realize anomaly localization. Our experiment study on two real world data sets demonstrates the effectiveness of our approach. Future works focus on two directions: (1) how to select the number of principal components which best interpreting the normal subspace. (2) How to integrate the network topology information of sensor network into JSPCA to improve the localization performance.

Acknowledgments

This work has been partially supported by an Office of Naval Research award N00014-07-1-1042 and an NSF grant IIS 0845951.

6. REFERENCES

[1] D. Brauckhoff, K. Salamatian, and M. May. Applying pca for traffic anomaly detection: Problems and solutions. In *INFOCOM*, pages 2866–2870. IEEE, 2009.

[2] X. Chen, W. Pan, J. T. Kwok, and J. G. Carbonell. Accelerated gradient method for multi-task sparse learning problem. In *ICDM*, pages 746–751, 2009.

[3] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the $_1$-ball for learning in high dimensions. In *ICML*, pages 272–279, 2008.

[4] C. Franke and M. Gertz. Orden: outlier region detection and exploration in sensor networks. In *SIGMOD Conference*, pages 1075–1078, 2009.

[5] S. Hirose, K. Yamanishi, T. Nakata, and R. Fujimaki. Network anomaly detection based on eigen equation compression. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1185–1194, New York, NY, USA, 2009. ACM.

[6] L. Huang, M. I. Jordan, A. Joseph, M. Garofalakis, and N. Taft. In-network pca and anomaly detection. In *In NIPS*, pages 617–624, 2006.

[7] T. Idé, A. C. Lozano, N. Abe, and Y. Liu. Proximity-based anomaly detection using sparse structure learning. In *SDM*, pages 97–108, 2009.

[8] T. Idé, S. Papadimitriou, and M. Vlachos. Computing correlation anomaly scores using stochastic nearest neighbors. In *ICDM '07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, pages 523–528, Washington, DC, USA, 2007. IEEE Computer Society.

[9] E. Keogh and T. Folias. The ucr time series data mining archive. Website, 2002. `http://www.cs.ucr.edu/eamonn/TSDMA/index.html`.

[10] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *In ACM SIGCOMM*, pages 219–230, 2004.

[11] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. In *In ACM SIGCOMM*, pages 217–228, 2005.

[12] J. Liu, S. Ji, and J. Ye. Multi-task feature learning via efficient l2,1-norm minimization. In *Conference on Uncertainty in Artificial Intelligence (UAI) 2009*, 2009.

[13] Y. Nesterov. Gradient methods for minimizing composite objective function. *CORE Discussion Paper*, 76:265–286, 2007.

[14] B. Quanz and C. Tsatsoulis. Determining object safety using a multiagent, collaborative system. In *Environment-Mediated Coordination in Self-Organizing and Self-Adaptive Systems (ECOSOA 2008) Workshop*, Venice, Italy, October 2008.

[15] H. Ringberg, A. Soule, J. Rexford, and C. Diot. Sensitivity of pca for traffic anomaly detection. In *SIGMETRICS '07: Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 109–120, New York, NY, USA, 2007. ACM.

[16] H. Song, S. Zhu, and G. Cao. Svats: A sensor-network-based vehicle anti-theft system. In *INFOCOM*, pages 2128–2136, 2008.

[17] N. Xu, S. Rangwala, and et al. A wireless sensor network for structural monitoring. In *IN SENSYS*, pages 13–24, 2004.

[18] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society B*, 67:301–320, 2005.

[19] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286, 1996.