Workshop on Ubiquitous Data Mining

in conjunction with the
20th European Conference on Artificial Intelligence
(ECAI 2012)
Montpellier, France
August 27 - 31, 2012

Edited by

João Gama, Manuel F. Santos, Nuno Marques,
Paulo Cortez and Pedro P. Rodrigues

Preface

Ubiquitous Data Mining is a recent research topic that uses data mining techniques to extract useful knowledge from data continuously generated from devices with limited computational resources that move in time and space. The goal of this workshop is to convene researchers (from both academia and industry) who deal with machine learning and data mining techniques for Ubiquitous Data Mining. We strongly welcome papers describing real-world applications of Ubiquitous Data Mining. This is the second workshop in the topic held in conjunction with ECAI. We received 15 submissions that were evaluated by 2 members of the Program Committee. The PC recommended accepting 9 papers. We have a diverse set of papers focusing from applications in medical domains, activity recognition, predicting taxis demand, to more theoretical aspects of computing statistics in data streams. All papers deal with different aspects of evolving data and/or distributed data.

We would like to thank all people that make this event possible. First of all, we thank authors that submit their work and the Program Committee for the work in reviewing the papers, and proposing suggestions to improve the works. A final Thanks to the ECAI Workshop Chairs for all the support.

João Gama, Manuel F. Santos, Nuno Marques,
Paulo Cortez and Pedro P. Rodrigues
Program Chairs

Organization

Program Chairs:
João Gama, Manuel F. Santos, Nuno Marques,
Paulo Cortez and Pedro P. Rodrigues

Publicity Chair:
Carlos A. Ferreira

Organized in the context of the project Knowledge Discovery from Ubiquitous Data Streams (PTDC/EIA-EIA/098355/2008)

Program Committee:

- Albert Bifet, Technical University of Catalonia, Spain
- Alfredo Cuzzocrea, University of Calabria, Italy
- André Carvalho, University of São Paulo (USP), Brazil
- Antoine Cornuéjols, LRI, France
- Carlos A. Ferreira, Institute of Engineering of Porto, Portugal
- Eduardo Spinosa, University of São Paulo (USP), Brazil
- Elaine Faria, University of Uberlândia (UFU), Brazil
- Elaine Sousa, University of São Paulo, Brazil
- Elena Ikonomovska, University St. Cyril & Methodius, Macedonia
- Ernestina Menasalvas, Technical University of Madrid, Spain
- Florent Masseglia, INRIA, France
- Geoffrey Holmes, University of Waikato, New Zealand
- Hadi Tork, LIAAD-INESC TEC, Portugal
- Jesús Aguilar, University of Pablo Olavide, Spain
- Jiong Yang, Case Western Reserve University, USA
- João Gama, University of Porto, Portugal
- José Ávila, University of Malaga, Spain
- Josep Roure, CMU, USA
- Manuel F. Santos, University of Minho, Portugal

- Márcia Oliveira, University of Porto, Portugal
- Mark Last, Ben Gurion University, Israel
- Matjaž Gams, Jozef Stefan Institute, Slovenia
- Michael May, Fraunhofer Bonn, Germany
- Min Wang, IBM, USA
- Miroslav Kubat, University of Miami, USA
- Mohamed Gaber, University of Plymouth, UK
- Myra Spiliopoulou, University of Magdeburg, Germany
- Nuno Marques, University of Nova Lisboa, Portugal
- Paulo Cortez, University of Minho, Portugal
- Pedro P. Rodrigues, University of Porto, Portugal
- Petr Kosina, Masaryk University, Czech Republic
- Philip Yu, IBM Watson, USA
- Raquel Sebastião, University of Porto, Portugal
- Rasmus Pederson, Copenhagen Business School, Denmark
- Ricard Gavaldà, University of Barcelona, Spain
- Rosane Maffei, University of São Paulo (USP), Brazil
- Shonali Krishnaswamy, Monash University, Australia
- Xiaoyang S. Wang, University of Vermont, USA
- Ying Yang, Monash University, Australia

Table of Contents

Invited Talks

Papers

Posters

LIFT - Local Inference in Massively Distributed Systems

Michael May

Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS), Sankt Augustin, Germany
michael.may@iais.fraunhofer.de

Abstract

As the scale of todays networked techno-social systems continues to increase, the analysis of their global phenomena becomes increasingly difficult, due to the continuous production of streams of data scattered among distributed, possibly resource-constrained nodes, and requiring reliable resolution in (near) real-time. We will present work from an on-going European funded research project: LIFT - Local Inference in Massively Distributed Systems. On the theoretical side, the project investigates novel approaches for realising sophisticated, large-scale distributed data-stream analysis systems, relying on processing local data in situ. A key insight is that, for a wide range of distributed data analysis tasks, we can employ novel geometric techniques for intelligently decomposing the monitoring of complex holistic conditions and functions into safe, local constraints that can be tracked independently at each node (without communication), while guaranteeing correctness for the global-monitoring operation. An application area where this leads to very interesting applications is the real-time analysis of human mobility and traffic phenomena. In this case, privacy concerns add another dimension to the problem. We present a number of case studies how the LIFT-approach can be used for efficient, privacy-aware analysis of human mobility.

An Efficient Spatio-Temporal Mining Approach to Really Know Who Travels with Whom!

Pascal Poncelet

LGI2P Research Center, Nimes, France

Pascal.Poncelet@ema.fr

Abstract

Recent improvements in positioning technology has led to a much wider availability of massive moving object data. A crucial task is to find the moving objects that travel together. Usually, they are called spatio-temporal patterns. Due to the emergence of many different kinds of spatio-temporal patterns in recent years, different approaches have been proposed to extract them. However, each approach only focuses on mining a specific kind of pattern. In addition to the fact that it is a painstaking task due to the large number of algorithms used to mine and manage patterns, it is also time consuming. Additionally, we have to execute these algorithms again whenever new data are added to the existing database. To address these issues, in this talk we first redefine spatio-temporal patterns in the itemset context. Secondly, we propose a unifying approach, named GeT Move, using a frequent closed itemset-based spatio-temporal pattern-mining algorithm to mine and manage different spatio-temporal patterns. GeT Move is proposed in two versions which are GeT Move and Incremental GeT Move. Experiments performed on real and synthetic datasets and the experimental results will be also presented to show that our approaches are very effective and outperform existing algorithms in terms of efficiency. Finally we will present some future work.

Cooperative Kernel-Based Forecasting in Decentralized Multi-Agent Systems for Urban Traffic Networks

Jelena Fiosina and **Maksims Fiosins**[1]

Abstract. The distributed and often decentralised nature of complex stochastic traffic systems having a large amount of distributed data can be represented well by multi-agent architecture. Traditional centralized data mining methods are often very expensive or not feasible because of transmission limitations that lead to the need of the development of distributed or even decentralized data mining methods including distributed parameter estimation and forecasting. We consider a system, where drivers are modelled as autonomous agents. We assume that agents possess an intellectual module for data processing and decision making. All such devices use a local kernel-based regression model for travel time estimation and prediction. In this framework, the vehicles in a traffic network collaborate to optimally fit a prediction function to each of their local measurements. Rather than transmitting all data to one another or a central node in a centralized approach, the vehicles carry out a part of the computations locally by transmitting only limited amount of data. This limited transmission helps the agent that experiences difficulties with its current predictions. We demonstrate the efficiency of our approach with case studies with the analysis of real data from the urban traffic domain.

1 INTRODUCTION

Multi-agent systems (MAS) often deal with complex applications, such as sensors, traffic, or logistics networks, and they offer a suitable architecture for distributed problem solving. In such applications, the individual and collective behaviours of the agents depend on the observed data from distributed sources. In a typical distributed environment, analysing distributed data is a non-trivial problem because of several constraints such as limited bandwidth (in wireless networks), privacy-sensitive data, distributed computing nodes, etc. The distributed data processing and mining (DDPM) field deals with these challenges by analysing distributed data and offers many algorithmic solutions for data analysis, processing, and mining using different tools in a fundamentally distributed manner that pays careful attention to the resource constraints [3].

Traditional centralized data processing and mining typically requires central aggregation of distributed data, which may not always be feasible because of the limited network bandwidth, security concerns, scalability problems, and other practical issues. DDPM carries out communication and computation by analyzing data in a distributed fashion [10]. The DDPM technology offers more efficient solutions in such applications.

In this study we focus on the urban traffic domain, where many traffic characteristics such as travel time, travel speed, congestion probability, etc. can be evaluated by autonomous agents-vehicles in a distributed manner. In this paper, we present a general distributed regression forecasting algorithm and illustrate its efficiency in forecasting travel time.

Numerous data processing and mining techniques were suggested for forecasting travel time in a centralized and distributed manner. Statistical methods, such as regression and time series, and artificial intelligence methods, such as neural networks, are successfully implemented for similar problems. However, travel time is affected by a range of different factors. Thus, accurate prediction of travel time is difficult and needs considerable amount of traffic data. Understanding the factors affecting travel time is essential for improving prediction accuracy [13].

We focus on non-parametric, computationally intensive estimation, i.e. Kernel-based estimation, which is a promising technique for solving many statistical problems, including parameter estimation. In our paper, we suggest a general distributed kernel-based algorithm and use it for forecasting travel time using real-world data from southern Hanover. We assume that each agent autonomously estimates its kernel-based regression function, whose additive nature fits very well with streaming real-time data. When an agent is not able to estimate the travel time because of lack of data, i.e., when it has no data near the point of interest (because the kernel-based estimation uses approximations), it cooperates with other agents. An algorithm for multi-agent cooperative learning based on based on transmission of the required data as a reply to the request of the agent experiencing difficulties was suggested. After obtaining the necessary data from other agents, the agent can forecast travel-time autonomously.

The travel-time, estimated in the DDPM stage, can serve as an input for the next stage of distributed decision making of the intelligent agents [5].

This study contributes in the following ways: It suggests (a) a decentralized kernel-based regression forecasting approach, (b) a regression model with a structure that facilitates travel-time forecasting; and it improves the application efficiency of the proposed approach for the current real-world urban traffic data.

The remainder of this paper is organized as follows. Section 2 describes the related previous work in the DDPM field for MAS, kernel density estimation, and travel-time prediction. Section 3 describes the current problem more formally. Section 4 presents the multivariate kernel-based regression model adopted for streaming data. Section 5 describes the suggested cooperative learning algorithm for optimal prediction in a distributed MAS architecture. Section 6 presents a case study and the final section contains the conclusions.

[1] Institure of Informatics, Clausthal University of Technology, Germany
email: {jelena.fiosina, maksims.fiosins}@gmail.com

2 RELATED PREVIOUS WORK

2.1 Distributed Data Mining in Multi-agent Systems

A strong motivation for implementing DDPM for MAS is given by Da Silva et al. in [3], where authors argue that DDPM in MAS deals with pro-active and autonomous agents that perceive their environment, dynamically reason out actions on the basis of the environment, and interact with each other. In many complex domains, the knowledge of agents is a result of the outcome of empirical data analysis in addition to the pre-existing domain knowledge. DDPM of agents often involves detecting hidden patterns, constructing predictive and clustering models, identifying outliers, etc. In MAS, this knowledge is usually collective. This collective 'intelligence' of MAS must be developed by distributed domain knowledge and analysis of the distributed data observed by different agents. Such distributed data analysis may be a non-trivial problem when the underlying task is not completely decomposable and computational resources are constrained by several factors such as limited power supply, poor bandwidth connection, privacy-sensitive multi-party data.

Klusch at al. [11] concludes that autonomous data mining agents, as a special type of information agents, may perform various kinds of mining operations on behalf of their user(s) or in collaboration with other agents. Systems of cooperative information agents for data mining in distributed, heterogeneous, or homogeneous, and massive data environments appear to be quite a natural progression for the current systems to be realized in the near future.

A common feature of all approaches is that they aim at integrating the knowledge that is extracted from data at different geographically distributed network sites with minimum network communication and maximum local computations. Local computation is carried out on each site, and either a central site communicates with each distributed site to compute the global models or a peer-to-peer architecture is used. In the case of the peer-to-peer architecture, individual nodes might communicate with a resource-rich centralized node, but they perform most tasks by communicating with neighbouring nodes through message passing over an asynchronous network [3].

A distributed system should have the following features for the efficient implementation of DDPM: the system consists of multiple independent data sources, which communicate only through message passing; communication between peers is expensive; peers have resource constraints (e. g. battery power) and privacy concerns [3].

Typically, communication involves bottlenecks. Since communication is assumed to be carried out exclusively by message passing, the primary goal of several DDPM methods, as mentioned in the literature, is to minimize the number of messages sent. Building a monolithic database in order to perform non-distributed data processing and mining may be infeasible or simply impossible in many applications. The costs of transferring large blocks of data may be very expensive and result in very inefficient implementations [10].

Moreover, sensors must process continuous (possibly fast) streams of data. The resource-constrained distributed environments of sensor networks and the need for a collaborative approach to solve many problems in this domain make MAS architecture an ideal candidate for application development.

In our study we deal with homogeneous data. However, a promising approach to agent-based parameter estimation for partially heterogeneous data in sensor networks was suggested in [7]. Another decentralized approach for homogeneous data was suggested in [18] to estimate the parameters of a wireless network by using a parametric linear model and stochastic approximations.

2.2 Travel Time Prediction Models

Continuous traffic jams indicate that the maximum capacity of a road network is met or even exceeded. In such a situation, the modelling and forecasting of traffic flow is one of the important techniques that need to be developed [1]. Nowadays, knowledge about travel time plays an important role in transportation and logistics, and it can be applied in various fields and purposes. From travellers' viewpoints, the knowledge about travel time helps to reduce the travel time and improves reliability through better selection of travel routes. In logistics, accurate travel time estimation could help reduce transport delivery costs and increase the service quality of commercial delivery by delivering goods within the required time window by avoiding congested sections. For traffic managers, travel time is an important index for traffic system operation efficiency [13].

There are several studies in which a centralized approach is used to predict the travel time. The approach was used in various intelligent transport systems, such as in-vehicle route guidance and advanced traffic management systems. A good overview is given in [13]. To make the approach effective, agents should cooperate with each other to achieve their common goal via the so called gossiping scenarios. The estimation of the actual travelling time using vehicle-to-vehicle communication without MAS architecture was described in [14].

On the other hand, a multi-agent architecture is better suited for distributed traffic networks, which are complex stochastic systems. Further, by using centralized approaches the system cannot adapt quickly to situations in real time, and it is very difficult to transmit a large amount of information over the network. In centralized approaches, it is difficult or simply physically impossible to store and process large data sets in one location. In addition, it is known from practice that the most drivers rely mostly on their own experience; they use their historical data to forecast the travel time [5]. Thus, decentralized multi-agent systems are are fundamentally important for the representation of these networks [1]. We model our system with autonomous agents to allow vehicles to make decisions autonomously using not only the centrally processed available information, but also their historical data.

Traffic information generally goes through the following three stages: data collection and cleansing, data fusion and integration, and data distribution [12]. The system presented in [12] consists of three components, namely a Smart Traffic Agent, the Real-time Traffic Information Exchange Protocol and a centralized Traffic Information Centre that acts as the backend. A similar architecture is used in this study, but the prediction module, incorporated into Start Traffic Agent (vehicle agent), is different. In our study we do not focus on the transmission protocol describing only the information, which should be sent from one node to another, without the descriptions of protocol packets. The centralized Traffic Information Centre in our model is used only for storing system information.

The decentralized MAS approach for urban traffic network was considered in [2] also, where the authors forecast the traversal time for each link of the network separately. Two types of agents were used for vehicles and links, and a neural network was used as the prediction model.

For travel time forecasting different regression models can be applied. Linear multivariate regression model for decentralized urban traffic network was proposed in [4]. This regression model is well-studied, is parametric and allows estimation of each variables contribution. However is not sufficiently effective in the case of non-linear systems. Alternatively non-parametric kernel-based regression models can be applied. These models can be effectively used for any types

of systems, however are relatively new and not well-studied yet. The efficiency of non-parametric kernel-based regression approach for traffic flow forecasting in comparison to parametric approach was made in [17]. In this study we apply non-parametric kernel regression for the similar traffic system as in [4] in order to increase the prediction quality.

2.3 Kernel Density Estimation

Kernel density estimation is a non-parametric approach for estimating the probability density function of a random variable. Kernel density estimation is a fundamental data-smoothing technique where inferences about the population are made on the basis of a finite data sample. A kernel is a weighting function used in non-parametric estimation techniques.

Let $X_1, X_2, \ldots, X_n$ be an iid sample drawn from some distribution with an unknown density f. We attempt to estimate the shape of f, whose kernel density estimator is

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right), \tag{1}$$

where kernel $K(\bullet)$ is a non-negative real-valued integrable function satisfying the following two requirements: $\int_{-\infty}^{\infty} K(u)du = 1$ and $K(-u) = K(u)$ for all values of u; $h > 0$ is a smoothing parameter called bandwidth. The first requirement to $K(\bullet)$ ensures that the kernel density estimator is a probability density function. The second requirement to $K(\bullet)$ ensures that the average of the corresponding distribution is equal to that of the sample used [8]. Different kernel functions are available: Uniform, Epanechnikov, Gausian, etc. They differ in the manner in which they take into account the vicinity observations to estimate the function from the given variables.

A very suitable property of the kernel function is its additive nature. This property makes the kernel function easy to use for streaming and distributed data [8], [3], [7]. In [11], the distributed kernel-based clustering algorithm was suggested on the basis of the same property. In this study, kernel density is used for kernel regression to estimate the conditional expectation of a random variable.

3 PROBLEM FORMULATION

We consider a traffic network with several vehicles, represented as autonomous agents, which predict their travel time on the basis of their current observations and history. Each agent estimates locally the parameters of the same traffic network. In order to make a forecast, each agent constructs a regression model, which explains the manner in which different explanatory variables (factors) influence the travel time. A detailed overview of such factors is provided in [13]. The following information is important for predicting the travel time [15]: average speed before the current segment, number of stops, number of left turns, number of traffic light, average travel time estimated by Traffic Management Centres (TMC). We should also take into account the possibility of an accident, network overload (rush hour) and the weather conditions.

Let us consider a vehicle, whose goal is to drive through the definite road segment under specific environment conditions (day, time, city district, weather, etc.). Let us suppose that it has no or little experience of driving in such conditions. For accurate travel time estimation, it contacts other traffic participants, which share their experience in the requested point.

In this step, the agent that experiences difficulties with a forecast sends its requested data point to other traffic participants in the transmission radius. The other agents try to make a forecast themselves. In the case of a successful forecast, the agents share their experience by sending their observations that are nearest to the requested point. After receiving the data from the other agents, the agent combines the obtained results, increases its experience, and makes a forecast autonomously.

In short, decentralized travel-time prediction consists of three steps: 1) local prediction; 2) in the case of unsuccessful prediction: selection of agents for experience sharing and sending them the requested data point; 3) aggregation of the answers and prediction.

4 LOCAL PARAMETER ESTIMATION

The non-parametric approach to estimating a regression curve has four main purposes. First, it provides a versatile method for exploring a general relationship between two variables. Second, it can predict observations yet to be made without reference to a fixed parametric model. Third, it provides a tool for finding spurious observations by studying the influence of isolated points. Fourth, it constitutes the flexible method of substitution or interpolating between adjacent **X**-values for missing values [8].

Let us consider a non-parametric regression model [9] with a dependent variable Y and a vector of d regressors X

$$Y = m(\mathbf{x}) + \epsilon, \tag{2}$$

where ϵ is the disturbance term such that $E(\epsilon|\mathbf{X} = \mathbf{x}) = 0$ and $Var(\epsilon|\mathbf{X} = \mathbf{x}) = \sigma^2(\mathbf{x})$, and $m(\mathbf{x}) = E(Y|\mathbf{X} = \mathbf{x})$. Further, let $(\mathbf{X}_i, Y_i)_{i=1}^n$ be the observations sampled from the distribution of $(\mathbf{X}, Y)$. Then the Nadaraya-Watson kernel estimator is

$$m_n(\mathbf{x}) = \frac{\sum_{i=1}^{n} K\left(\frac{\mathbf{x}-\mathbf{X}_i}{h}\right) Y_i}{\sum_{i=1}^{n} K\left(\frac{\mathbf{x}-\mathbf{X}_i}{h}\right)} = \frac{p_n(\mathbf{x})}{q_n(\mathbf{x})}, \tag{3}$$

where $K(\bullet)$ is the kernel function of R^d and h is the bandwidth. Kernel functions satisfy the restrictions from (1). In our case we have a multi-dimensional kernel function $K(u) = K(u_1, u_2, \ldots, u_d)$, that can be easily presented with univariate kernel functions as: $K(u) = K(u_1) \cdot K(u_2) \cdot \ldots \cdot K(u_d)$. We used the Gaussian kernel in our experiments.

Network packets are streaming data. Standard statistical and data mining methods deal with a fixed dataset. There is a fixed size n for dataset and algorithms are chosen as a function of n. In streaming data there is no n: data are continually captured and must be processed as they arrive. It is important to develop algorithms that work with non-stationary datasets, handle the streaming data directly, and update their models on the fly [6].

This requires recursive windowing. The kernel density estimator has a simple recursive windowing method that allows the recursive estimation using the kernel density estimator:

$$m_n(\mathbf{x}) = \frac{p_n(\mathbf{x})}{q_n(\mathbf{x})} = \frac{p_{n-1}(\mathbf{x}) + K\left(\frac{\mathbf{x}-\mathbf{X}_n}{h}\right) Y_n}{q_{n-1}(\mathbf{x}) + K\left(\frac{\mathbf{x}-\mathbf{X}_n}{h}\right)}. \tag{4}$$

5 DECENTRALISED MULTI-AGENT COOPERATIVE LEARNING ALGORITHM

In this section, we describe the cooperation for sharing the prediction experience among the agents in a network. While working with

streaming data, one should take into account two main facts. The nodes should coordinate their prediction experience over some previous sampling period and adapt quickly to the changes in the streaming data, without waiting for the next coordination action.

Let us first discuss the cooperation technique. We introduce the following definitions.

Let $\mathbf{L} = \{L^j \mid 1 \leq j \leq p\}$ be a group of p agents. Each agent $L^j \in \mathbf{L}$ has a local dataset $D^j = \{(\mathbf{X}_c^j, Y_c^j) | c = 1 \ldots, N^j\}$, where $\mathbf{X}_c^j$ is a d-dimensional vector. In order to underline the dependence of the prediction function (3) from the local dataset of agent L^j, we denote the prediction function by $m[D^j](\mathbf{x})$.

Consider a case when some agent L^i is not able to forecast for some d-dimensional future data point $\mathbf{X}_{new}^i$ because it does not have sufficient data in the neighbourhood of $\mathbf{X}_{new}^i$. In this case, it sends a request to other traffic participants in its transmission radius by sending the data point $\mathbf{X}_{new}^i$ to them. Each agent L^j that has received the request tries to predict $m[D^j](\mathbf{X}_{new}^i)$. If it is successful, it replies to agent L^i by sending its best data representatives $\hat{D}^{(j,i)}$ from the neighbourhood of the requested point $\mathbf{X}_{new}^i$. Let us define $G^i \subset L$, a group of agents, which are able to reply to agent L^i by sending the requested data.

To select the best data representatives, each agent L^j makes a ranking among its dataset D^j. It can be seen from (3) that each Y_c^j is taken with the weight w_c^j with respect to $\mathbf{X}_{new}^i$, where

$$w_c^j = \frac{K\left(\frac{\mathbf{X}_{new}^i - \mathbf{X}_c^j}{h}\right)}{\sum_{l=1}^{n} K\left(\frac{\mathbf{X}_{new}^i - \mathbf{X}_l^j}{h}\right)}.$$

The observations with maximum weights w_c^j are the best candidates for sharing the experience.

All the data $\hat{D}^{(j,i)}$, $L^j \in G^i$ received by agent L^i should be verified, and duplicated data should be removed. We denote the new dataset of agent L^i as $D_{new}^i = \bigcup_{L^j \in G^i} \hat{D}^{(j,i)}$. Then, the kernel function of agent L^i is updated taking into account the additive nature of this function:

$$m[D_{new}^i](\mathbf{x}) = \sum_{L^j \in G^i} m[\hat{D}^{(j,i)}](\mathbf{x}) + m[D^i](\mathbf{x}). \qquad (5)$$

Finally, agent L^i can autonomously make its forecast as $m[D_{new}^i](\mathbf{X}_{new}^i)$ for $\mathbf{X}_{new}^i$.

6 CASE STUDIES

We simulated a traffic network in the southern part of Hanover (Germany). The network contains three parallel and five perpendicular streets, creating fifteen intersections with a flow of approximately 5000 vehicles per hour.

The vehicles solve a travel time prediction problem. They receive information about the centrally estimated system variables (such as average speed, number of stops, congestion level, etc.) for this city district from TMC, combine it with their historical information, and make adjustments according to the information of other participants using the presented cooperative-learning algorithm. In this case, the regression analysis is an essential part of the local time prediction process. We consider the local kernel based regression model (3) and implement the cooperative learning algorithm (5). The factors are listed in Table 1. The selection and significance of these variables was considered in [4].

We simulated 20 agents having their initial experience represented by a dataset of size 20 till each agent made 100 predictions, thus making their common experience equal to 2400. We assumed the maximal number of transmitted observations from one agent equals 2.

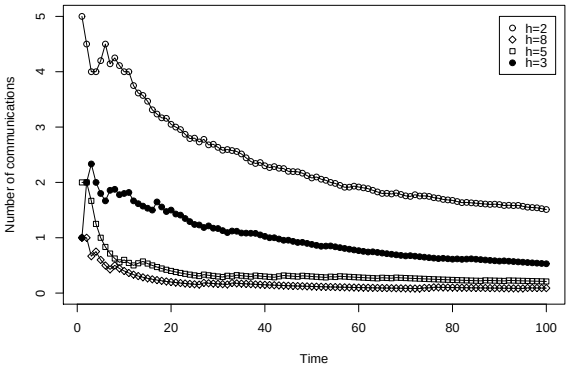


Figure 1. Average number of communications over time for different h

Table 1. System factors influencing the travel time

Variable	Description
Y	travel time (min);
X_1	length of the route (km);
X_2	average speed in the system (km/h);
X_3	average number of stops in the system (units/min);
X_4	congestion level of the flow (Veh/h);
X_5	number of traffic lights in the route (units);
X_6	number of left turns in the route (units).

During the simulation, to predict more accurately, the agents used the presented cooperative learning algorithm that supported the communication between agents with the objective of improving the prediction quality. The necessary number of communications depends on the value of the smoothing parameter h. The average number of necessary communications is given in Figure 1. We can see the manner in which the number of communications decreased with the learning time. We varied h and obtained the relation between the communication numbers and h as a curve. The prediction ability of one of the agents is presented at Figure 2. Here, we can also see the relative prediction error, which decreases with time. The predictions that used communication between agents are denoted by solid triangles, and the number of such predictions also decreases with the time.

The goodness of fit of the system was estimated using a cross-validation technique. We assume that each agent has its own training set, but it uses all observations of other agents as a test set, so we use 20-fold cross-validation. To estimate the goodness of fit, we used analysis of variance and generalized coefficient of determination R^2 that provides an accurate measure of the effectiveness of the prediction of future outcomes by using the non-parametric model [16]. The calculated R^2 values and the corresponding number of the observations that were not predicted (because cooperation during testing was not allowed) depending on h are listed in Table 2. From Figure 3 we can also see how R^2 is distributed among the system agents. The results suggest that we should find some trade-off between system

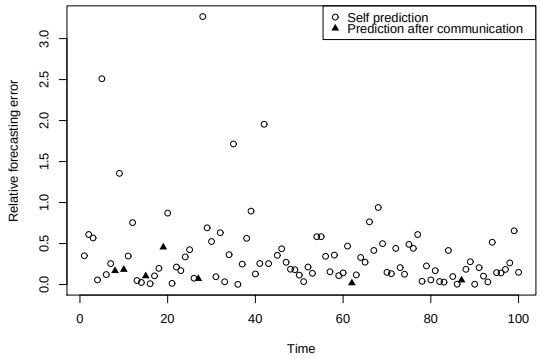


Figure 2. Relative prediction error and communication frequency of a single agent over time

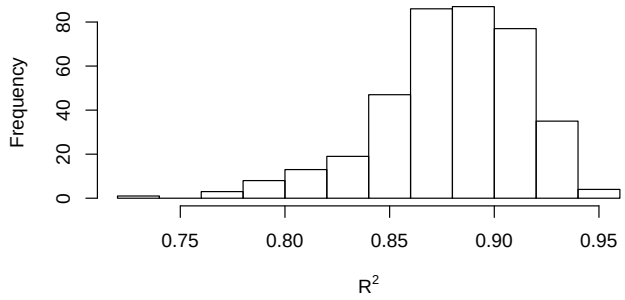


Figure 3. Distribution of goodness of fit measure R^2 using cross-validation for the whole system, $h = 2$

accuracy (presented by R^2) and the number of necessary communications (presented by the percentage of not predicted observations), which depend on h. The point of trade-off should depend on the communication and accuracy costs.

Table 2. R^2 goodness of fit measure using cross-validation for the whole system for different h

Characteristic of System	h=2	h=3	h=5	h=8
System average R^2	0.86	0.84	0.83	0.82
Average % of not predicted observations	3	1	0.5	0.1

A linear regression model [4] applied to the same data gives lower average goodness of fit R^2=0.77, however predictions can be calculated for all data points.

7 CONCLUSIONS

In this study, the problem of travel-time prediction was considered. Multi-agent architecture with autonomous agents was implemented for this purpose. Distributed parameter estimation and cooperative learning algorithms were presented, using the non-parametric kernel-based regression model. We demonstrated the efficiency of the suggested approach through simulation with real data from southern Hanover. The experimental results show the high efficiency of the proposed approach. In future we are going to develop a combined approach that allows agent to choose between parametric and non-parametric estimator for more accurate prediction.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No. PIEF-GA-2010-274881. We thank Prof. J. P. Müller for useful discussions during this paper preparation.

REFERENCES

[1] A.L Bazzan, J. Wahle, and F. Kluegl, 'Agents in traffic modelling - from reactive to social behaviour', *Advances in Artificial Intelligence (LNAI)*, **1701**, 509–523, (1999).
[2] R. Claes and T. Holvoet, 'Ad hoc link traversal time prediction', in *Proc. of the 14th Int. IEEE Conf. on Intelligent Transportation Systems*, (October 2011).
[3] J.C. da Silva, C. Giannella, R. Bhargava, H. Kargupta, and M. Klusch, 'Distributed data mining and agents', *Eng. Appl. Artif. Intell.*, **18**(7), 791–801, (2005).
[4] J. Fiosina, 'Decentralised regression model for intelligent forecasting in multi-agent traffic networks', *Distributed Computing and Artificial Intelligence (Advances in Intelligent and Soft Computing)*, **151**, 255–263, (2012).
[5] M. Fiosins, J. Fiosina, J. Müller, and J. Görmer, 'Agent-based integrated decision making for autonomous vehicles in urban traffic', *Advances on Practical Applications of Agents and Multiagent Systems (Advances in Intelligent and Soft Computing)*, **88**, 173–178, (2011).
[6] *Handbook of Computational Statistics: Concepts and Methods*, eds., J. E. Gentle, W. Härdle, and Y. Mori, Springer, Berlin/Heidelberg, 2004.
[7] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden, 'Distributed regression: an efficient framework for modeling sensor network data', in *Proc. of the 3rd Int. Sym. on Information Processing in Sensor Networks*, Berkeley, (2004).
[8] W. Härdle, *Applied Nonparametric Regression*, Cambridge University Press, Cambridge, 2002.
[9] W. Härdle, M. Müller, S. Sperlich, and A. Werwatz, *Nonparametric and Semiparametric Models*, Springer, Berlin/Heidelberg, 2004.
[10] *Advances in Distributed and Parallel Knowledge Discovery*, eds., H. Kargupta and P. Chan, AAAI Press The MIT PressAcademic Press, Melno Park and Cambridge and London, 2000.
[11] M. Klusch, S. Lodi, and G. Moro, 'Agent-based distributed data mining: The kdec scheme', in *Proc. of Int. Conf. on Intelligent Information Agents - The AgentLink Perspective*, volume 2586 of *LNCS*, pp. 104–122. Springer, (2003).
[12] W. Lee, S. Tseng, and W. Shieh, 'Collaborative real-time traffic information generation and sharing framework for the intelligent transportation system', *Information Sciense*, **180**, 62–70, (2010).
[13] H. Lin, R. Zito, and M.A.P. Taylor, 'A review of travel-time prediction in transport and logistics', in *Proc. of the Eastern Asia Society for Transportation Studies*, volume 5, pp. 1433 – 1448, Hamburg, (2005).
[14] G. Malnati, C. Barberis, and C.M. Cuva, 'Gossip: Estimating actual travelling time using vehicle to vehicle communication', in *Proc. of the 4-th Int. Workshop on Intelligent Transportation*, Hamburg, (2007).
[15] C.E. McKnight, H.S. Levinson, C. Kamga, and R.E. Paaswell, 'Impact of traffic congestion on bus travel time in northern new jersey', *Transportation Research Record Journal*, **1884**, 27–35, (2004).
[16] J.S. Racine, 'Consistent significance testing for nonparametric regression', *Journal of Business and Economic Statistics*, **15**, 369379, (1997).
[17] B.L. Smith, B.M. Williams, and R.K. Oswald, 'Comparison of parametric and nonparametric models for traffic flow forecasting', *Transportation Research Part C*, **10**, 303–321, (2002).
[18] S.S. Stankovic, M.S. Stankovic, and D.M. Stipanovic, 'Decentralized parameter estimation by consensus based stochastic approximation', *IEEE Trans. Automatic Controll*, **56**, (2009).

Event and anomaly detection using Tucker3 decomposition

Hadi Fanaee Tork[1], **Márcia Oliveira**[2], **João Gama**[3], **Simon Malinowski**[4] and **Ricardo Morla**[5]

Abstract. Failure detection in telecommunication networks is a vital task. So far, several supervised and unsupervised solutions have been provided for discovering failures in such networks. Among them unsupervised approaches has attracted more attention since no label data is required [1]. Often, network devices are not able to provide information about the type of failure. In such cases, unsupervised setting is more appropriate for diagnosis. Among unsupervised approaches, Principal Component Analysis (PCA) has been widely used for anomaly detection literature and can be applied to matrix data (e.g. Users-Features). However, one of the important properties of network data is their temporal sequential nature. So considering the interaction of dimensions over a third dimension, such as time, may provide us better insights into the nature of network failures. In this paper we demonstrate the power of three-way analysis to detect events and anomalies in time-evolving network data.

1 INTRODUCTION

Event detection can be briefly described as the task of discovering unusual behavior of a system during a specific period of the time. On the other hand, anomaly detection concentrates on the detection of abnormal points. So clearly it is different from event detection since it just considers the points rather than a group of points. Our work takes into account both issues using multi-way data analysis. Our methodology comprises the following steps: 1) Anomaly detection: detection of individual abnormal users 2) Generating user trajectories (i.e. behavior of users over time), 3) Clustering users' trajectories to discover abnormal trajectories and 4) Detection of events: group of users who show abnormal behavior during specific time periods. Although there is a rich body of research on the two mentioned issues, to the best of our knowledge we are the first ones applying multi-way analysis to the anomaly and event detection problem. In the remainder of this section we explain some basic and related concepts and works. Afterwards, we define the problem, and then discuss three-way analysis methods. Hereafter, we introduce the dataset and experiments. Finally, we discuss the results and point out possible future directions.

1.1 ANOMALY DETECTION

Anomaly is as a pattern in the data that does not conform to the expected behavior [1]. Anomaly detection has a wide range of application in computer network intrusion detection, medical informatics, and credit card fraud detection. A significant amount of research has been devoted to solve this problem. However our focus is on unsupervised methods. Anomaly detection techniques can be classified into five groups [1]: classification-based, clustering-based, nearest neighbor based, statistical methods, information theory-based methods and spectral methods. Based on this classification, our method is placed in the group of spectral methods. These approaches first decompose the high-dimensional data into a lower dimension space and then assume that normal and abnormal data points appear significantly different from together. This some benefits: 1) they can be employed in both unsupervised and supervised settings 2) they can detect anomalies in high dimensional data, and 3) unlike clustering techniques, they do not require complicated manual parameter estimation. So far, most of the work related to spectral anomaly detection was based on Principal Component Analysis (PCA) and Singular Value Decomposition (SVD). Two of the most important applications of PCA during recent years has been in the domain of intrusion detection [2] [3] and traffic anomaly detection [4] [5].

1.2 EVENT DETECTION

Due to huge amount of sequential data being generated by sensors, event detection has become an emerging issue with several real-world applications. Event is a significant occurrence or pattern that is unusual comparing to the normal patterns of the behavior of a system [6]. This can be natural phenomena or manual system interaction. Some examples of events can be an attack on the network, bioterrorist activities, epidemic disease, damage in an aircraft, pipe-breaks, forest fires, etc. A real system behaves normally most of the time, until an anomaly occurs that may cause damages to the system. Since the effects of an event in the system are not known a priori, detecting and characterizing abnormal events is challenging. This is the reason why most of the time we cannot evaluate different algorithms. One solution might be injection of artificial event into the normal data. However, construction of a realistic event pattern is not trivial [7].

1.3 HIDDEN MARKOV MODELS

Hidden Markov Models (HMMs) have been used at least for the last three decades in signal processing, especially in domain of speech recognition. They have also been applied in many other domains as bioinformatics (e.g. biological sequence analysis), environmental studies (e.g. earthquake and wind detection), and finance (financial time series). HMMs became popular for its simplicity and general mathematical tractability [8].

[1] LIAAD-INESC TEC, FEUP- University of Porto, hadi.fanaee@fe.up.pt
[2] LIAAD-INESC TEC, FEP- University of Porto, marcia@liaad.up.pt
[3] LIAAD-INESC TEC, FEP- University of Porto, jgama@fep.up.pt
[4] INESC TEC, FEUP- University of Porto, sjfm@inescporto.pt
[5] INESC TEC, FEUP- University of Porto, ricardo.morla@inescporto.pt

HMMs are widely used to describe complex probability distributions in time series and are well adapted to model time dependencies in such series. HMMs assume that observations distribution does not follow a normal distribution and are generated by different processes. Each process is dependent on the state of an underlying and unobserved Markov process [7]. Markov process denotes that the value of a process X_t only depends on the previous value of X. Using notations of [9] let:

T = Length of the Observation sequence
N = Number of states
Q = $\{q_0, q1, \ldots, q_{(N-1)}\}$ = distinct states of Markov process
A = State transition probabilities
B = set of N observation probability distributions
π = Initial State Distribution
O = $(O_0, O1, \ldots, O_{(T-1)})$ = observation sequence

A HMM Model is defined with the triple of $\lambda = (A, B, \pi)$. It assumes that Observations are drawn using the observation probability distribution associated to the current state. The transition probabilities between states are given in matrix A.

The three main problems related with HMMs are the following. The first problem consists in computing the probability P(O) that a given observation sequence O is generated by a given HMM λ. The second problem consists in finding the most probable sequence of hidden states given an observation sequence O and λ and the third problem is related to parameter inference. It consists in estimating the parameters of the HMM λ that best fits a given observation sequence O. The mainly used algorithms to solve these problems are given in the last column of Table 1. More details about these algorithms can be found in [10]. In this paper, we deal with the third problem to estimate the HMM parameters that best describe time series, as it will be explained in Section 2.

Table1. Three HMM Problems

Problem	Input	Output	Solution
Problem 1	λ, O	P(O)	Forward Backward algorithm
Problem 2	λ, O	Best Q	Viterbi algorithm
Problem 3	O	λ	Baum-Welch algorithm

1.4 THREE-WAY DATA ANALYSIS

Traditional data analysis techniques such as PCA, clustering, regression, etc. are only able to model two dimensional data and they do not consider the interaction between more than two dimensions. However, in several real-world phenomena, there is a mutual relationship between more than two dimensions (e.g. a 3D tensor (Users×Features×Time)) and thus, they should be analyzed through a three-way perspective. Three-way analysis considers all mutual dependencies between the different dimensions and provides a compact representation of the original tensor in lower-dimensional spaces. The most common three-way analysis models are Tucker2, Tucker3, and PARAFAC [10] which are generalized versions of two-mode principal component model or, more specifically, SVD. Following, we briefly introduce Tucker3 model as the best-known method for analysis of three-way data.

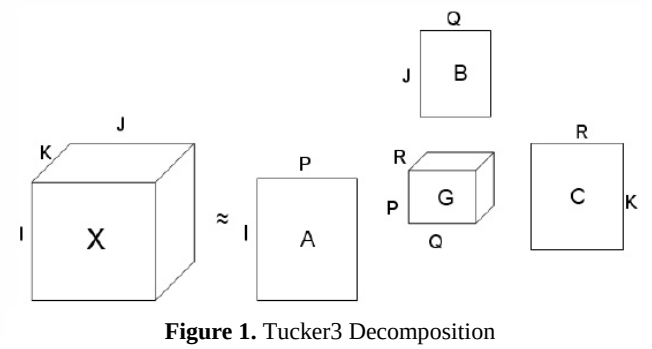


Figure 1. Tucker3 Decomposition

1.4.1 Tucker3 Model

The Tucker3 model decomposes a three-mode tensor $\mathcal{X}$ into set of component matrices **A**, **B**, **C** and a small core tensor $\mathcal{G}$. The following mathematical equation reveals the decomposition:

$$\mathcal{X}_{ijk} \approx \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} \mathcal{G}_{pqr} \times a_{ip} \times b_{jq} \times c_{kr} \quad (1)$$

Where P, Q and R are parameters of the Tucker3 model and represent the number of components retained in the first, the second and the third mode of the tensor, respectively. This decomposition is illustrated in Figure 1.

2 PROBLEM DEFINITION

One of significant issues in telecommunication systems, such as IP/TV, is to detect the anomalies at both network and user level. In order to study this, target users are usually equipped with a facility in their modem which sends an automatic notification message to the central server when the connection of a client in the network is lost or reestablished. These modems are ubiquitous and geographically dispersed.

The modeling of such behavior is not straightforward because the number of notification messages is not equal for each user during the time period under analysis. For instance, one user may face 40 connection problems in an hour, hence generating 40 messages, while others may face 5 or even no problems at all. In standard event detection problems, for each time point there is a measurement via one or multiple sensors. In the context of our application, such measurements do not take place at regular time points, since user modems (or sensors) only send messages to the server when something unexpected occurs. Figure 2 illustrates two sample users. Each circle represents the time stamp at which a notification relative to the given user is received, while ΔT represents the inter-arrival time between two consecutive messages. As it can be seen, 2 messages were related to user 1 in that period, while 4 were related to user 2 during the same period. Also, the ΔT between messages is larger for user 1 than for user 2. This means that user 2 sent messages more frequently than user 1. As in many other event detection problems, we could easily use the number of events per hour (measurement) at different users (sensors) to detect the events but this way we would lose the information content provided by the ΔT's.

As the number of ΔT is not the same for each user, this feature cannot be directly integrated in our model. Hence, this would cause

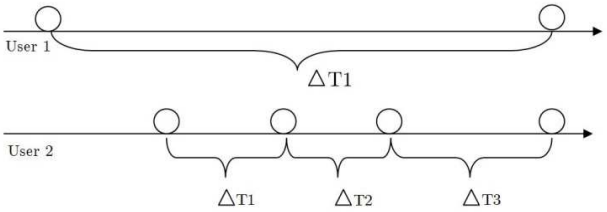


Figure 2. Two sample users with different number of messages and different intervals

some vectors to have different lengths, which is not supported by the Tucker3 analysis. To solve this, every time-series of ΔT relative to a given user is modeled by a 2-state HMM obtained by the Baum-Welch algorithms [11]. 6 parameters are extracted from the HMM and are used to describe the time-series of ΔT of the users. Using this approach we obtain the same number of features for each user and, then, include this information in our feature vectors.

Table2. Datasets in tensor format

Data	**1st mode (I)** Users	**2nd mode (J)** Features	**3rd mode (K)** Hours
X102	102	10	720
X909	909	10	720

3 DATASET

Dataset is extracted from the usage log of a European IP/TV service provider. The raw dataset includes the notification messages of users in each line including their occurrence time. As previously mentioned, it is not possible to use this data directly in our modeling approach, so some pre-processing steps were performed. In addition to the obtained HMM parameters for each hour and for each user, we included another features, such as mean, variance, entropy and number of messages per hour, to our feature vector. We generated two separated datasets, each one spanning a time period of one month, which is equivalent to 720 hours. In one set we selected 102 users and in another we selected 909 users. The latter dataset is an extended version of the former. We then transformed both datasets to the tensor format. These datasets are shown in a format of Tucker3 input tensor (figure 1) in Table 2 where I, J, K represent users, features and hours modes, respectively.

4 EXPERIMENTS

This section is divided into three subsections, according to the steps mentioned in the Introduction section. In subsection 1, we explain how we detect the abnormal users. In the next subsection we describe how we generate user trajectories And in the last subsection we explain how we cluster the trajectories using hierarchical clustering and detect events using user trajectories.

4.1 Abnormal Users

We applied Tucker3 model to both datasets X102 and X909 by employing a MATLAB package called *Three-mode component analysis (Tucker3)* [10]. Before that, we performed ANOVA test [10] to see the significance of three-way and two-way interaction in the data. The results of this test are presented in Table 3. *ANOVA Max 2D* represents the maximum value obtained via different combinations of two-way modeling (e.g. I-J, J-K, I-K). As it can be seen, bigger numbers are obtained for three-dimension interaction (*ANOVA 3D*), which reveals that there is a mutual interaction between the three dimensions in both datasets that can be explained better with three-way modeling like Tucker3, than with two-way modeling like PCA.

Table3. ANOVA test and selected model parameters P-Q-R

Data	**ANOVA** max 2D	**ANOVA** 3D	**Selected Model P-Q-R**	**fit**
X102	26.18%	38.90%	3-2-2	42.00
X909	17.02%	78.04%	40-2-4	51.01

The next step is to estimate the best parameters P, Q, R of Equation 1. P-Q-R is similar to what we have in PCA. In PCA we just determine the number of PCs for one dimension but here we need to determine the number of principal components for each one of the three modes. P, Q and R can assume values that fall within the interval $[1, max]$, where max denotes the maximum number of entities in the corresponding mode. For example, in terms of X102 the P-Q-R can go from 1-1-1 to 102-10-720. These parameters are chosen based on a trade-off between model parsimony, or complexity, and goodness of fit. For instance, regarding the mentioned dataset, 1-1-1 gives about 28% fit (less complete and less complex) and model 102-10-720 gives 100% fit (most complete and most complex). If we try parameters 3-2-2 the model has a 42% fit. So it can be more reasonable choice because it finds a good compromise between complexity and fit. In [10] the *scree test* method is proposed as a guideline to choose these parameters. We used this test to determine the best model for both datasets. The selected model parameters and their corresponding fits are presented in Table 3. This means that, for example, for dataset X102 if we choose Tucker3 model with 3, 2 and 2 components to summarize the users, features and hours modes, respectively, the model is able to explain 42% of the total variance contained in raw data. After the estimation of model parameters, we used the selected model to decompose the raw data into a lower dimensional subspace, as illustrated in Figure 1 and Equation 1. After the decomposition we obtained matrices **A**, **B** and **C**, a small core tensor $\mathcal{G}$ and a tensor of residual errors.

In order to detect the abnormal users we simply projected the users on the component space yielded by matrix **A**. This projection is presented in Figure 3, for dataset X102. The three-dimensional subspace is given by the three obtained components by the model for the 1st mode (users). As mentioned earlier, this number of components is one of the parameters of the model, namely $P = 3$, which corresponds to the first mode.

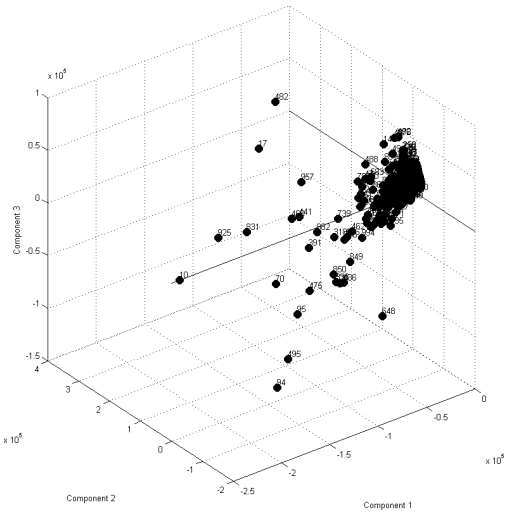


Figure 3. Projection of Users on Matrix **A** for dataset X102

In order to evaluate the reliability of the model we used the same procedure and applied a Tucker3 model to dataset X909, which includes all users of X102. Our idea was to see how this model can identify abnormal users from both datasets. For this purpose, we computed the Euclidean distance between each user in the projection space (see Figure 3) and the corresponding center (0, 0, 0), for both datasets X102 and X909. Then we normalized the distances for each dataset and computed the Pearson correlation for the common users of these two datasets, according to their distance to the center of the subspace. We obtained a correlation of 68.44%. Although, for X909 we just took 3 out of 40 main components to and model fit was different for both datasets (42% for X102 and 51.01% for X909), abnormal or normal users in X102 approximately appeared as the same way in X909 with 68.44% confidence. This denotes that Tucker3 is a robust model to detect the abnormal users.

4.2 User Trajectories

Visualization methods like the one we presented in Figure 3 are not able to show the evolving behavior of users over time. We need another solution to enable us understanding the behavior of users over time. One solution is to project the users on a decomposed feature space (matrix **B** of Figure 1) for each time point. Since both of our selected parameters have Q equal to 2 it means that after projecting Users on feature space we must have a coordinate of (x, y) for each timepoint and for each user. The process of generating this coordinates is presented in Figure 4. $B_{:,1}$ and $B_{:,2}$ represent the two components that summarize the original entities of the features mode and $\mathcal{X}$ represent the three-order tensor (see Figure 1). The rows of the front matrix are the users, the columns correspond to the features and the third mode (z-axis) represents the hours. If we compute the dot product between each tensor's rows with the columns of the component matrix **B**, yielded by the Tucker3 model we obtain the coordinate(x, y) for a given timepoint. If we repeat this procedure for all time points (e.g. hours), we are able to generate the coordinates of each user for the 720 hours. The user trajectories are

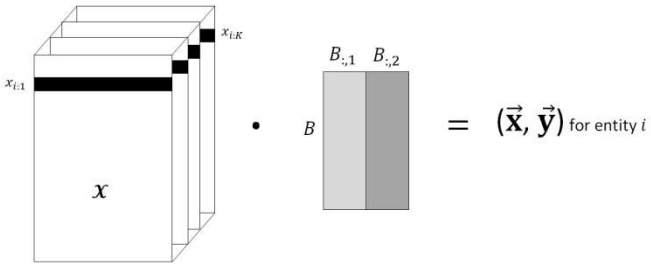


Figure 4. Generation process of user trajectories

obtained by sequentially connecting these coordinates. Formally we define user trajectories as:

Definition 1 (User Trajectory) : A sequence of time-stamped points, $Trj = p_0 \rightarrow p_1 \rightarrow \ldots \rightarrow p_i \rightarrow \cdots \rightarrow p_k$, where p_i (x, y, t) $(i = 0,1, \ldots k)$, and t is a time point.

Figure 5 shows two abnormal users appearing in the top-10 users ranked based on abnormality. These abnormal users were ranked based on decreasing values of distance to the center, as explained in subsection 4.1, user 10 (right) is ranked 2nd and user 95 (left) is ranked 4th. However, as it is clear from the figure, their behavior over time is completely different. User 95 just shows two abnormal behaviors that correspond to two time points, while user 10 shows this abnormal behavior almost in all time points. This means that user 10 is dealing with a stable problem while user 95 only has problems in specific points in time. This type of interpretation was not possible based only on the ranking list of abnormal users, obtained in subsection 4.1. Using user trajectories provides us richer insights into different kind of problems a user can experience. For instance, what made user 95 be identified as abnormal could be something that suddenly happened in the network and then was quickly solved, while for user 10, some problems occurred but they weren't solved until the end of the time period under analysis.

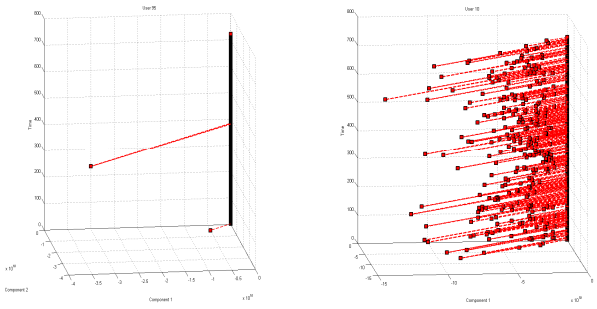


Figure 5. Two sample users trajectories in X909, Left) 4th ranked abnormal user Right) 2nd ranked abnormal user

4.3 Event Detection from user trajectories

Even though user trajectories can be useful, when the number of users is too large, the individual analysis of each trajectory can become a cumbersome task. If we notice that some group of users trajectories behave similarly, this can be understood as something abnormal happens in their network level. Then some prevention or surveillance operations can be conducted more quickly.

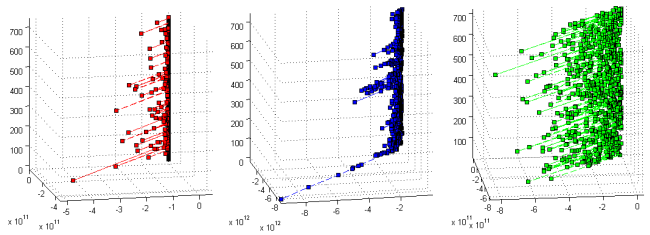


Figure 6. Center Trajectory of clusters, Left: 1 user, Center : 866 users and Right: 22 users.

To explore this goal, we employed Agglomerative Hierarchical Clustering toolbox from MATLAB to cluster user trajectories. We defined Euclidean distance between each point in trajectories as our distance function and Ward's criterion as the linkage criterion. We tested different values of cut-off from 0.6 to 1.2 to examine the clustering structure. The most suited clustering structure was obtained for a dendrogram distance of 1, which cuts the tree to level that, corresponds to three clusters. The average trajectory of these clusters is shown in Figure 6. Cluster red has1 user (0.1%), cluster blue comprises 866 users (97.4%) and cluster green includes 22 users (2.5%). As it can be seen, no specific pattern can be recognized from the green and the red cluster. The users in these two clusters show an abnormal behavior almost in all time points. Such event can be due to a stable specific problem such as a problem in the user device. Regarding the blue cluster, it is possible to detect three events. First significant event occurs between hours 350 to 400. Second and third events also occur between 450 to 480 and 520 to 560, respectively. However, the occurrence of the second and the third events should be assessed with hypothesis testing since they can be due to an accidental change.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we present a study on using the Tucker3 decomposition to discover abnormal users in an IP/TV network. Our results indicate that Tucker3 is a robust method for detecting abnormal users in situations where interactions between the three dimensions are present. From the tensor decomposition, we can define user trajectories. The trajectories allow us to observe the behavior of these users over time. We were able to identify two kinds of abnormal users: those who show frequent abnormal behavior over the whole time period and those who are associated to one or few severe abnormal behaviors over the time period. Without resorting to the analysis of user temporal trajectories it would have been harder to uncover such facts. Furthermore, from the clusters of the users' trajectories, we have identified three events that occurred during three time points in the network. The result of this work can be used in a real network surveillance system to identify failures in the quickest possible time. In this work, we did not consider the spatial relation of users. Taking into account spatial relationships between network nodes could lead to a better clustering of users. Since some users might show similar behavior, with some delays, other distance measures for clustering should be tested. Currently we are employing another distance function using dynamic time warping, which assigns two users with same behavior but with a time shift in the same cluster. The solution we presented for detection of events was based on clustering of trajectories. We are going to apply sliding window on trajectories to find time periods that have the most compact trajectories, which would lead to the discovery of events in a more accurate and reliable way

ACKNOWLEDGEMENTS

This work was supported by the Institute for Systems and Computer Engineering of Porto (INESC TEC) under projects TNET with reference BI/120033/PEst/LIAAD and project KDUS with reference PTDC/EIA-EIA/098355/2008. The authors are grateful for the financial support. This work is also financed by the ERDF European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project CMU-PT/RNQ/0029/2009. We'd like to acknowledge the support of J.A. at the European IP/TV service provider.

REFERENCES

[1] Banerjee A., Kumar V. Chandola V., "Anomaly detection: a survey," *2009*, pp. 1–58, 2009.

[2] W. Wang and R. Battiti, "Identifying intrusions in computer networks with principal component analysis," in *International Conference on Availability, Reliability and Security*, Vienna, Austria, 2006, pp. 270 - 279.

[3] X. Guan, and X. Zhang W. Wang, "A novel intrusion detection method based on principal component analysis," in *IEEE Symposium on Neural Networks*, 2004, p. 2004.

[4] M.-L., Chen, S.-C., Sarinnapakorn, K., and Chang Shyu, "A novel anomaly detection scheme-based on principal component classifier," in *3rd IEEE International Conference on Data Mining*, Melbourne, Florida, 2003, pp. 353--365.

[5] C. Callegari, "A novel PCA-based Network Anomaly Detection," in *IEEE International Conference on Communications (ICC)*, Pisa, Italy , 2011, pp. 1 - 5.

[6] M. C. et al. Kerman, "Event Detection Challenges, Methods, and Applications in Natural and Artificial Systems," in *14th International Command and Control Research and Technology Symposium*, Washington, DC, 2009,

[7] G. Shmueli, "Wavelet-based Monitoring in Modern Biosurveillance," University of Maryland, College Park, Report Working Paper RHS-06-002, 2005.

[8] M. I. Zucchini W., *Hidden Markov Models for Time Series*. USA/FL: Chapman & Hall/CRC, 2009.

[9] M. Stamp. (2004, Jan) Department of Computer Science, San Jose State University. [Online]. http://www.cs.sjsu.edu/faculty/stamp/RUA/HMM.pdf

[10] H.A.L., & van Mechelen, I. Kiers, "Three-way component analysis: Principles and illustrative application," *Psychological Methods*, vol. 6, pp. 84-110, 2001.

[11] Lawrence R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257-286, Feb 1989.

Bus Bunching Detection: A Sequence Mining Approach

Luis Moreira-Matias[1,2], Carlos Ferreira[2,3], João Gama[2,4], João Mendes-Moreira[1,2], Jorge Freire de Sousa[5]

Abstract. Mining public transportation networks is a growing and explosive challenge due to the increasing number of information available. In highly populated urban zones, the vehicles can often fail the schedule. Such fails cause headway deviations (HD) between high-frequency bus pairs. In this paper, we propose to identify systematic HD which usually provokes the phenomenon known as Bus Bunching (BB). We use the PrefixSpan algorithm to accurately mine sequences of bus stops where multiple HD frequently emerges, forcing two or more buses to clump. Our results are promising: 1) we demonstrated that the BB origin can be modeled like a sequence mining problem where 2) the discovered patterns can easily identify the route schedule points to adjust in order to mitigate such events.

1. INTRODUCTION

In highly populated urban zones, it is well known that there is some schedule instability, especially in highly frequent routes (10 minutes or less) [1-5]. In this kind of routes it is more important the headway (time separation between vehicle arrivals or departures) regularity than the fulfillment of the arrival time at the bus stops [4]. Due to this high frequency, this kind of situations may force a bus platoon running over the same route. In fact, a small delay of a bus provokes the raising of the number of passengers in the next stop. This number increases the dwell time (time period where the bus is stopped at a bus stop) and obviously also increases the bus's delay. On the other hand, the next bus will have fewer passengers, shorter dwell times with no delays. This will continue as a snow ball effect and, at a further point of that route, the two buses will meet at a bus stop, forming a platoon as it is illustrated in Fig. 1. This phenomenon has several denominations: the Bangkok effect [6], Bus Platooning [7], Vehicle Pairing [8], Headway Instability [1], Bus Clumping or Bus Bunching (BB) [2]. From now on, we will use the last one.

The occurrence of BB forces the controllers to take actions in order to avoid this headway instability, forcing the adherence to the schedule. BB situations can cause several problems like: further buses delays, full buses, decreased comfort in the buses, larger waiting times at the bus stops, growing number of passengers waiting, greater resources demand and a decrease of schedule reliability. All this can cause the loss of passengers to other transportation means and/or companies.

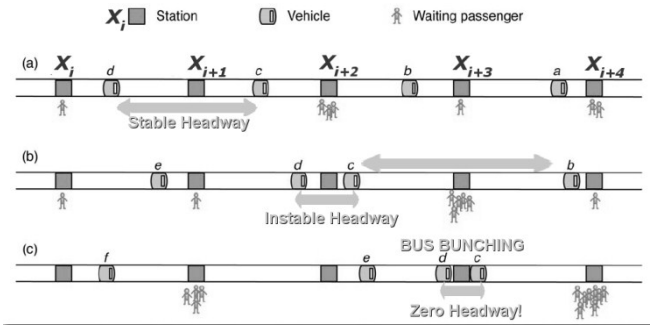


Figure 1. Bus Bunching problem illustration. Figure from [1].

Our goal is to identify the causes of BB occurrences using AVL (Automatic Vehicle Location) historical data. The BB phenomenon always starts by a headway deviation (HD) at a bus stop [9]. We intend to find frequent and systematic HD event sequences in the trips of a given route: bus stops where the bus activities - like the passenger boarding - will propagate the headway irregularities further and further. These bus stops sequences highlights problematic route regions: from now on we will refer to it as Bunching Black Spots (BBS - bus stops sequences where a HD will, with a high probability, start a BB in one of the following bus stops of the trip).

We use the PrefixSpan algorithm (presented in Section 3) to mine frequent sequences in the HD sequences extracted from this dataset. We apply this methodology to data from two urban lines of a public transport operator of Porto. It proved to be efficient in the detection of HD patterns in the bus stops of the studied routes.

The results from this framework can be highly useful to the public transport planners. One of the most known ways to mitigate the bus bunching is to adjust the slack time introduced in each schedule point (bus stops selected along the route for which the arrival time is defined) [10]. By using this framework, the planners can use the information about the BBS along the routes to select which schedule points should be changed (increasing or decreasing the slack time) to mitigate BB effectively.

The main results are: the observation that the BB phenomenon starts at the initial bus stops; and the existence of high correlation between HD that occurs at a given bus stop and the HD detected in the next ones.

This paper is structured as follows. Section 2 states a brief description of the problem we want to solve, the related work, our motivation and a clear definition of our approach. Section 3 presents the methodology proposed. Section 4 presents summarily the dataset used, its main characteristics and some statistics about it. Section 5 presents the results obtained through the application of

[1] DEI, FEUP, University of Porto, Rua Dr. Roberto Frias, s/n 4200-465 Porto – Portugal, email: [luis.matias, jmoreira]@fe.up.pt
[2] LIAAD – INESC TEC, Rua de Ceuta, 118, 6º; 4050-190 Porto – Portugal
[3] DEI, ISEP, Rua Dr. António Bernardino de Almeida, 431, 4200-072 Porto - Portugal, email: cgf@isep.ipp.pt
[4] FEP, University of Porto, Rua Dr. Roberto Frias, s/n 4200-465 Porto – Portugal, email: jgama@fep.up.pt
[5] DEIG, FEUP, University of Porto, Rua Dr. Roberto Frias, s/n 4200-465 Porto – Portugal, email: jfsousa@fe.up.pt

the PrefixSpan algorithm to our dataset and a discussion about those results. Section 6 concludes and describes the future work we intend to carry on.

2. PROBLEM OVERVIEW

Nowadays, the road public transportation (PT) companies face a huge competition of other companies or even of other transportation means like the trains, the light trams or the private ones. The service reliability is a fundamental metric to win this *race* [11]: if a passenger *knows* that a bus of a selected company will arrive *certainly* on the schedule on his bus stop, he will probably pick it often. The reverse effect is also demonstrated and a BB event forming a visual bus pair is a strong bad reliability signal to the passengers' perception of the service quality, which can lead to important profit losses [12, 13]. This tendency to form platoons is usual for urban vehicles (specially the PT ones) and arises for the specific and complex characteristics of transit service perturbations. Those are mainly related with changes in three key factors [8]: the dwell time and the loading time (highly correlated) and the non-casual passenger arriving (passengers that, for an unexpected reason – like a soccer match or a local holiday - try to board in a specific bus stop distinct from the usual one). However, the study of these changes impact on the service reliability is not in our current scope. Our goal is to find persistent and frequent headway irregularities which will *probably* provoke, in a short time horizon, a BB event.

There are two distinct approaches found in the literature to handle the BB events: the first one defines the bunching problem as a secondary effect of a traffic system malfunction like a traffic/logistic problem (signal priority handling, adaptation of bus stops/hubs logistics to the needs, adjustments of the bus routes to the passengers demand, etc.). The second one defines the BB problem like a main one that must be treated and solved *per se* (adjust the timetables and the schedule plans to improve schedules' reliability or set live actions to the irregular bus pairs, for instance). In this work, we are just focused on the second approach which related work, motivation and scope we present along this section.

2.1. Related Work

Gershenson *et. al.* presented a model adapted from a metro-like system and implemented a multi-agent simulation [1]. To achieve stability, they implemented adaptive strategies where the parameters are decided by the system itself, depending on the passenger density. As a result, the system puts a restriction to the vehicle holding time (it sets a maximum dwell time), negotiating this value for each bus stop with the other vehicles.

The introduction of AVL systems changed the research point-of-view on bus bunching, in the last ten years, from planning to control. There are several techniques in PT to improve the schedule plans on time tables based on AVL data.. C. Daganzo presents a dynamic holding time formula based on real time AVL data in order to adaptively compensate the headway instability introduced in the system [2].

The relations between the irregularities in the headway sequences and the BB events have been recently explored: in [8] is presented a study identifying the headway distributions representing service perturbations based on probability density functions (p.d.f.). Despite their useful conclusions, their model had two main disadvantages: 1) is not based in real AVL data and 2) it does not present a probability density function to represent the pattern of consecutive headways irregularities. We do believe that this specific issue can be rather addressed mining frequent sequences on real AVL data, as we present here.

2.2. Motivation and Scope

We can define the headway irregularities as events that occur in a bus stop of a given trip. Those events consist in a large variation (1 for positive or -1 for negative) on the headway: Headway Deviation events (HD).

These are usually correlated in a snowball effect that may occur (or not) in a given (straight or spaced) sequence of bus stops. Despite the analysis of the state-of-art work on the mitigation of BB events, the authors found no work on systematizing real HD patterns that seem to be in the genesis of a BB event.

An unreliable timetable is one of the main causes of many HD events. Usually, a timetable is defined using schedule points: stops for which there is an arriving or departing time defined. One of the most well-known PT planning ways to mitigate HD events is to add/reduce slack time in these defined timestamps to increase schedule plan overall reliability. However, only a small percentage of the bus stops served by a given timetable are used as schedule points. This is exemplified in the upper part of Fig. 2 (the reader can obtain further details on schedule plan building in chapter 1 from [14]).Usually, PT planners easily identify which lines present more HD and BB events. However, three questions still remain open:

- Which should be the schedule points affected?
- Which action (increase/decrease slack time) should be applied to these schedule points in order to reduce the occurrence probability of BB events?
- Which day periods should have the timestamps in these schedule points changed?

In this work, we address the first and third questions by mining frequent HD event sequences in the trips of a given route: bus stops that systematically propagate the headway irregularities further and further. The second issue is out of our scope but it is well addressed in the literature [10].

Our intention is to point out a route region where an HD event fast and systematically propagates itself along the route, forming a Bunching Black Spot (BBS). The BBS can be specific of a period of the day or continuous along the day. In the bottom part of Fig. 2 we present an example of a BBS. In the next section we present our methodology to mine BBS.

3. METHODOLOGY

Our methodology consists in finding consistent patterns of frequent HD events occurring in the same bus stops whenever a BB occurs – BBS. To do so we compare, at each bus stop, the round-trip times of every consecutive bus pairs. With the HD series thus obtained, we mine frequent sequence patterns. Firstly, we introduce the algorithm we used and finally we describe how we use it to create and mine our HD series for a given route.

3.1. Mining Time Series Sequences

There is a wide range of algorithms that can explore sequential data efficiently. To the best of our knowledge, Agrawal and Srikant introduced the sequential data mining problem in [15]. Let $\mathbf{I} = \{\mathbf{i}_1, \mathbf{i}_2, .., \mathbf{i}_n\}$ be a set of items and **e** an event such that $\mathbf{e} \subseteq \mathbf{I}$. A sequence is an ordered list of events $\mathbf{e}_1\mathbf{e}_2\ldots\mathbf{e}_m$ where each $\mathbf{e}_i \subseteq \mathbf{I}$.

Given two sequences $\boldsymbol{\alpha}=\mathbf{a_1a}_2\ldots\mathbf{a}_r$ and $\boldsymbol{\beta}=\mathbf{b}_1\,\mathbf{b}_2\,\ldots.\,\mathbf{b}_s$, sequence $\boldsymbol{\alpha}$ is called a subsequence of $\boldsymbol{\beta}$ if there exists integers $1 \leq \mathbf{j}_1 < \mathbf{j}_2 < \ldots < \mathbf{j}_r \leq s$ such that $\mathbf{a}_1 \subseteq \mathbf{b}_{j1}$, $\mathbf{a}_2 \subseteq \mathbf{b}_{j2}$, … ,$\mathbf{a}_r \subseteq \mathbf{b}_{jr}$. A sequence database is a set of tuples (**sid**, $\boldsymbol{\alpha}$) where **sid** is the sequence identification and $\boldsymbol{\alpha}$ is a sequence. The count of a sequence $\boldsymbol{\alpha}$ in **D**, denoted **count**($\boldsymbol{\alpha}$, **D**), is the number of sequences in **D** containing the $\boldsymbol{\alpha}$ subsequence.

The support of a sequence $\boldsymbol{\alpha}$ is the ratio between **count**($\boldsymbol{\alpha}$, **D**) and the number of sequences in **D.** We denote sequence support as **support**($\boldsymbol{\alpha}$, **D**). Given a sequence database **D** and a minimum support value $\boldsymbol{\lambda}$**,** the problem of sequence mining is to find all subsequences in **D** having a support value equal or higher than the $\boldsymbol{\lambda}$ value. Each one of the obtained sequences is also known as a frequent sequence.

One of the most interesting approaches to solve this kind of problems is PrefixSpan algorithm [16]. This algorithm makes use of pattern-growth strategies to efficiently find the complete set of frequent sequences. The algorithm starts by finding all frequent items (length one sequences). Then, for each one of these frequent items (the prefix) PrefixSpan partitions the current database into *prefix projections*. Each projection database contains all the sequences with the given prefix. This procedure runs recursively until all frequent sequences are found.

The PrefixSpan algorithm was chose to solve this problem due to its popularity and efficiency.

3.2. Method

Firstly we constructed headway sequences based in the AVL historic data for every bus pairs in a given route. Then we identified the headway profiles where BB events occurred based on the bus service reliability metrics presented in [17] and we extracted HD sequences from them.

Let $X = x_1x_2\ldots x_n$ be a headway sequence measured between a bus pair in a given route through n bus stops running with a frequency f $(f = 1/x_1)$. We identify a BB if there exists a x_i satisfying the inequality $x_i \leq (0.25 * 1/f)$ for at least one $i \in \{1, \ldots, n\}$. Based on this headway profiles, we formed a HD sequence as follows. Let $H = h_1h_2\ldots h_n$ be the HD sequences based on X. We compute the value of each h_i (the headway between a bus pair in the bus stop x_i), for each $i \in \{2, \ldots, n\}$, using the expression 1.

$$\boldsymbol{h_i} = \begin{cases} \mathbf{0} & \boldsymbol{if\ |x_i - x_{i-1}| < \left(\frac{1}{f}\right) * ht} \\ \mathbf{1} & \boldsymbol{if\ x_i - x_{i-1} \geq \left(\frac{1}{f}\right) * ht} \\ \mathbf{-1} & \boldsymbol{if\ x_i - x_{i-1} \leq -\left(\frac{1}{f}\right) * ht} \end{cases} \quad (1)$$

where ht is a threshold parameter given by the user for the HD definition. For the first bus stop is considered an HD of 0. Basically, a -1 event corresponds to a negative HD (delay) in a bus stop (i.e.: the two buses become closer), the 1 event is a positive HD (ahead of schedule) and the 0 occurs when the headway remains stable.

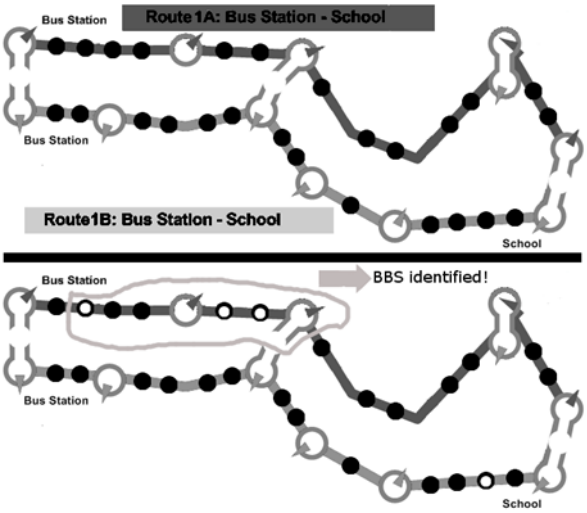


Figure 2. Example of Schedule Points and BBS. The two schemas exemplify two routes of a line running between an arbitrary school and a main bus station. In top part, route 1A has 19 bus stops represented by 13 small black circles and 6 big grey circles (the single one's are just bus stops, the double are hubs/interfaces). The last ones are the schedule points in the route's timetables. In the bottom part, the stops belonging to frequent HD sequences are identified (even if the BB itself occurs later in the route) with a small white circle inside them. The highlighted stops form a route region (Bunching Black Spot) where the schedule points need to be time-adjusted.

The xn represents a headway deviation in a bus stop n. The HD sequences are ordered according to the bus stop order defined for a given route. Our goal is to find sequences of bus stops with frequent HD by exploring a set of trips, in a given route, where BB occurrences were identified.

To do so, we collected the HD sequences of trips in work days where a BB event occurred and we mined them using the PrefixSpan algorithm by setting a (user-defined) minimum support value in order to identify HD patterns in the bus stops. The Fig.3 illustrates our methodology. We applied this methodology to four routes in a given period. This data is summarily described in Section 4.

4. DATASET

The source of this data was STCP, the Public Transport Operator of Porto, Portugal. The dataset was obtained through a bus dispatch system that integrates an Automatic Vehicle Location (AVL) system. The data captured through this system contains data of the trips from two lines (A and B) in the working days for the first ten months of 2010. Each line has two routes – one for each way {A1, A2, B1, B2}. Line B is a common urban line between Viso (an important neighborhood in Porto) passing by 26 bus stops (BS1_B1 to BS26_B1 and BS1_B2 to BS26_B2, respectively), and ending at Sá da Bandeira, a downtown bus hub. Line A is also an urban line between another downtown bus hub (Cordoaria) and Hospital São João - an important bus/light train interface in the city – using 22 bus stops (same schema than line B).

This dataset has one entry for each stop made by a bus running in the route during that period. It has associated a timestamp and a day type (1 for work days, 2-6 for other day types i.e.: holidays and weekends). Table 1 presents some statistics about the set of trips per route considered and the BB events identified. The Nr. of Trips

is the total number of trips considered in the given route, TT is the round-trip time, expressed in minutes, and DT is the number of daily trips occurred. Finally, trips with BB are the trips where at least one BB situation occurs and HD events are the positive or negative events ($h_i = 1$ or $h_i = -1$, respectively) measured in every bus stops along every trip for a given line.

5. RESULTS

We did our experiments only for the trips occurred during the peak periods (08:00 to 11:00 and 16:00 to 19:00). We did so because BB mainly occurred – as expected – during those periods. The routes A1 and A2 *suffer* more BB events and they are time-dispersed along the day. This happens because this line is an urban one between two important bus/metro interfaces (the downtown and the University Campus) with regular high frequencies during the entire day. So, they are highly frequent routes with many passengers during the entire day, which are well known factors to provoke BB occurrences. We mined sequences just in the bunching partition (trips with BB events). Moreover, we use the two partitions to compute the confidence of each sequence to be specific on the BB one. Our goal was to find patterns (i.e. frequent HD sequences) describing the headway irregular behavior of a typical BB trip.

We did two different experiments: the first one mined sequences in both peak hours simultaneously; the second one mined each peak hour considered individually (the morning and the evening ones). We did so to mine BBS peak-dependent (just occur in one of the two peaks), discovering whether the schedule points should be adjusted for the entire day or just in a specific period.

The results presented in Table 2 are for frequent subsequences of the HD sequences. We set PrefixSpan minimum support to 40% (sequences of length=1) and 20% (sequences with a length greater than 1) in the selected data partition, and a *ht*=0.15. We did so because the significance of the second case is higher than the first one. The second case demonstrates high correlations between distinct HD events in distinct bus stops that explain better the origin of the BB events.

5.1. Discussion

Firstly, we want to highlight that **only frequent HD subsequences (BBS) with events of type -1 (headway reductions) were detected**. All the sequences presents high confidence, demonstrating their specific validity in the bunching partition. In route B1 two BBS were identified: BS2_B1 and the pair BS3_B1 and BS4_B1. Both are located at the beginning of the route: the gap verified in these points may become larger in successive stops. The pair is deeply analyzed in Table 3: the isolated events in BS3_B1 and BS4_B1 have the same support than the events occurred in both bus stops. We can also set an association rule like BS3_B1= -1 -> BS4_B1= -1 (with a confidence of 97%) identifying a solid BBS in those two bus stops and an expected BB behavior.

In line A, BS2_A1 and BS2_A2 were identified as BBS. Additionally, they are - as well as the BBS identified in line B – located in the beginning of the route. The causes for this behavior are, probably, the large affluence of passengers in peak hours but the authors cannot sustain this with the available data.

Summarily, just BBS for the first bus stops were found. Based on this, we can conclude that **the BB in those routes were largely provoked by successive bus delays in the first bus stops** (the HD -1 events are mainly caused by bus delays [8]) although we cannot sustain whether they are failing the schedule.

Table 1. Descriptive statistics for each route considered. These times are in minutes. TT means round-trip times. DT means daily trips. Based in our HD event definition, the maximum number of events for a time period is given as $Nr.of\ Bus\ Stops * Nr.Of\ Trips$.

	B1	B2	A1	A2
Nr. of Trips	9391	10675	13802	12753
Nr. of Bus Stops	26	26	22	22
Minimum TT	11	11	11	11
Maximum TT	78	82	70	65
Minimum of DT	39	39	33	36
Maximum of DT	74	74	89	88
Median TT	29	21	21	38
Nr. of Bus Stops	26	26	22	22
Nr. of Trips w/ BB	332	378	559	630
Nr. of HD events detected	26905	29911	42803	43525

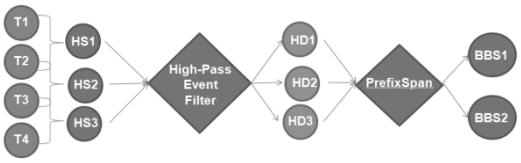


Figure 3. Bunching Black Spot Detection Methodology illustration. *Tn* is the time series measured in each bus stop of a given trip. *HS* are the corresponding Headway Sequences and *HD* the Headway Deviation event subsequences.

Table 2. The values presented are the Support of the sequences (number of trips where those events occur / total number of BB trips considered) as well as the confidence between the occurrences of those in the trips with BB and the total trips occurred in the period.

Route	Peaks Considered	Sequence (possible BBS)	Support	Confidence
B1	**Both**	**BS3_B1 = -1 BS4_B1=-1**	**0,2619**	**0,75**
B1	Both	BS2_B1 = -1	0,4206	0,80
A1	Both	BS2_A1 = -1	0,5095	0,72
A2	Both	BS2_A2 = -1	0,5706	0,61
B1	8h to 11h	BS5_B1 = -1	0,4000	0,91
B1	8h to 11h	BS2_B1 = -1	0,4308	0,85
A1	8h to 11h	BS6_A1 = -1	0,4064	0,88
A1	8h to 11h	BS3_A1 = -1	0,4225	0,87
A1	8h to 11h	BS2_A1 = -1	0,5669	0,72
A2	8h to 11h	BS2_A2 = -1	0,6237	0,74
B1	16h to 19h	BS2_B1 = -1	0,4099	0,82
A1	16h to 19h	BS2_A1 = -1	0,4500	0,81
A2	16h to 19h	BS2_A2 = -1	0,6237	0,78

Table 3. Detailed analysis of the mined sequence BS3_B1 = -1, BS4_B1=-1. The support of the highlighted sequences 01a and 01b are the same of the sequence 01: this can demonstrate an implication between the bus delays in the BS3_B1 and BS4_B1, an usual BB behavior. The confidence for a possible association rule BS3_B1 = -1 -> BS4_B1=-1 is 97%.

ID	Route	Peaks Considered	Sequence (possible BBS)	Support
01	**B1**	**Both**	**BS3_B1 = -1 BS4_B1=-1**	**0,2619**
01a	**B1**	**Both**	**BS3_B1 = -1**	**0,2619**
01b	**B1**	**Both**	**BS4_B1 = -1**	**0,2619**

In the second study, we analyzed whether the BBS identified were coherent in both peak hours. In route B1, the BS2_B1 is a BBS for both peak hours. BS2_A1 and BS2_A2 are also persistent BBS in both peaks. Those two bus stops correspond to an important bus interface (*Sá da Bandeira*) in the city and to a University Campus (*Asprela*), respectively. This happens because both routes maintain a high frequency and a large number of passengers during the day, being always busy.

In our opinion, the short lengths of the frequent subsequences mined (1 and 2) are not relevant compared with the relevance of the identified patterns. Those lengths will always depend on the routes analyzed, so they can be larger when applied to other datasets. The achieved patterns demonstrate that the BB patterns can be modeled like a frequent sequence mining problem. The results achieved demonstrate the utility of our framework to identify the exact schedule points to change in the timetables.

6. CONCLUSIONS AND FUTURE WORK

In public transportation planning, it is crucial to maintain the passengers' satisfaction as high as possible. A good way to do so is to prevent the phenomenon known as Bus Bunching.

There are two main approaches to handle this problem: the PT planning one, anticipating and identifying the origin of the problem, and a real time one, which tries to reduce the problem online (during the network function).

Our approach is a contribution to solve the PT planning problem: this framework can help to identify patterns of bus events from historical data to discover the schedule points to be adjusted in the timetables.

In this paper, we presented a methodology to identify BB events that use headway deviations from AVL trips data. We ran a sequence mining algorithm, the PrefixSpan, to explore such data. The results are promising. We clearly demonstrated the existence of relevant patterns in the HD events of the travels with bunching.

There were some bus stops sequences along the routes identified as BBS - Bunching Black Spots, forming regions within the schedule points that should be adjusted. We want to highlight the following findings:

- The high correlation between HD in distinct bus stops – one event in a given bus stop provoke an event on another one with a regularity sustained by a reasonable support and confidence;
- The detection of BBS in the beginning of the routes demonstrated that HD that occurs in the beginning of the trips can have a higher impact into the occurrence of BB compared with events occurred in bus stops further.

The main contributions of this work are: 1) to model the BB trip usual pattern like a frequent sequence mining problem; 2) to provide the operator the possibility to mitigate the BB in a given line by adjusting the timetables, instead of suggesting forced actions that can decrease schedule reliability and, consequently, reduce passengers' satisfaction.

The identified patterns are no more than alerts that suggest a systematic cause for the BB in the studied routes. This information can be used to improve the schedule. The goal is not to eliminate those events but just to mitigate them. Our future work consists in forecasting BB in a data stream environment based on AVL data. By using this approach, the BSS will be identified online as the data arrive in a continuous manner. This possibility will allow the use of control actions to avoid BB events that can occur even when the timetables are well adjusted, in order to prevent the majority of the potential BB occurrences.

ACKNOLEDGMENTS

We would like to thank STCP (Sociedade de Transportes Colectivos do Porto, S.A.) for the AVL historical data supplied to this work. We would also like to thank the support of the project Knowledge Discovery from Ubiquitous Data Streams (PTDC /EIA-EIA/098355/2008).

REFERENCES

[1] C. Gershenson, Pineda, L., "Why Does Public Transport Not Arrive on Time? The Pervasiveness of Equal Headway Instability," *PLoS ONE,* vol. 4, 2009.

[2] C. Daganzo, "A Headway-Based approach to eliminate Bus Bunching.," *Transportation Research Part B,* vol. 43, pp. 913-921, 2009.

[3] J. Pilachowski, "An approach to reducing bus bunching.," PhD, Univ. of California, Berkeley, California, 2009.

[4] J. Lin, Ruan, M., "Probability-based bus headway regularity measure," *IET intelligent transport systems,* vol. 3, pp. 400-408, 2009.

[5] L. Matias, J. Gama, J. Mendes-Moreira, and J. F. Sousa, "Validation of both number and coverage of bus Schedules using AVL data. ," presented at the ITSC'2010, Funchal, Portugal, 2010.

[6] P. Newman, "Transit-Oriented Development: An Australian Overview.," *Transit Oriented Development – Making it Happen,* 2005.

[7] J. Strathman, Kimpel, T., Callas, S., "Headway Deviation Effects on Bus Passenger Loads:Analysis of Tri-Met's Archived AVL-APC Data," 2003.

[8] G. Bellei, Gkoumas, K., "Transit vehicles' headway distribution and service irregularity," *Public Transport,* vol. 2, pp. 269-289, 2010.

[9] G. Newell, Potts, R., "Maintaining a bus schedule," in *2nd Australian Road Research Board*, 1964, pp. 388-393.

[10] J. Zhao, Dessouky, M., Bukkapatnam, S., "Optimal Slack Time for Schedule-Based Transit Operations," *Transportation Science,* vol. 40, pp. 529-539, 2006.

[11] J. Strathman, Kimpel, T., Dueker, K., "Automated bus dispatching, operations control and service reliability.," *Transportation Resesearch Record,* vol. 1666, pp. 28–36, 1999.

[12] R. Mishalani, "Passenger Wait Time Perceptions at Bus Stops: Empirical Results and Impact on Evaluating Real-Time Bus Arrival Information," *Journal of Public Transportation,* vol. 2, 2006.

[13] F. Wang, "Toward Intelligent Transportation Systems for the 2008 Olympics," *IEEE Intelligent Systems,* vol. 18, pp. 8-11, 2003.

[14] V. Vuchic, "Transit Systems, Operations and Networks.," in *Urban Transit*, ed New York: Wiley, 2005.

[15] R. Agrawal, Srikant, R., "Mining Sequential Patterns," presented at the Eleventh International Conference on Data Engineering, Taipei, Taiwan, 1995.

[16] P. Jian, Han, J., Mortazavi-asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M., "PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth.," presented at the 17th International Conference on Data Engineering, Heidelberg, Germany, 2001.

[17] TRB, "Transit Capacity and Quality of Service Manual. Transit Cooperative Research Program Web Document No. 6," presented at the Transportation Research Board - National Research Council, Washington, D.C., 1999.

Holistic distributed stream clustering for smart grids

Pedro Pereira Rodrigues[1] and **João Gama**[2]

Abstract. Smart grids consist of millions of automated electronic meters that will be installed in electricity distribution networks and connected to servers that will manage grid supervision, billing and customer services. World sustainability regarding energy management will definitely rely on such grids, so smart grids need also to be sustainable themselves. This sustainability depends on several research problems that emerge from this new setting (from power balance to energy markets) requiring new approaches for knowledge discovery and decision support. This paper presents a holistic distributed stream clustering view of possible solutions for those problems, supported by previous research in related domains. The approach is based on two orthogonal clustering algorithms, combined for a holistic clustering of the grid. Experimental results are included to illustrate the benefits of each algorithm, while the proposal is discussed in terms of application to smart grid problems. This holistic approach could be used to help solving some of the smart grid intelligent layer research problems, thus improving global sustainability.

1 INTRODUCTION

The Smart Grid (SG), regarded as the next generation power grid, is an electric system that uses two-way digital information, cyber-secure communication technologies, and computational intelligence in an integrated fashion across heterogeneous and distributed electricity generation, transmission, distribution and consumption to achieve energy efficiency. It is a loose integration of complementary components, subsystems, functions, and services under the pervasive control of highly intelligent management-and-control systems [4].

A key and novel characteristic of smart grids is the *intelligent layer* that analyses the data produced by these meters allowing companies to develop powerful new capabilities in terms of grid management, planning and customer services for energy efficiency. The development of the market with a growing share of load management incentives and the increasing number of local generators will bring new difficulties to grid management and exploitation.

1.1 Research problems

Power and current balance is major goal of all electricity distribution networks, given its impact on the need to produce, buy or sell energy. Moreover, due to the fluctuating power from renewable energy sources and loads, supply-demand balancing of power system becomes problematic [17]. Several intelligent techniques have been proposed in the past that make use of the amounts of streaming data that is available. As an example, Pasdar and Mahne (2011) proposed to use ant collony optimization on smart meters data to improve the current balancing on low-voltage distribution network. Further research could even take more advantages from smart grids if consumption patterns could be extracted [14].

The energy market is changing to meet the global challenge of power consumption awareness even at the lower household level [3]. New energy distribution concepts and the advent of smart grids has changed the way energy is priced, negotiated and billed. We are now in a world of hourly real-time pricing [1] which make use of smart meters to overcome the need for demand prediction precision and, more important, demand prediction reliability [13]. Furthermore, with the advent of micro-generation at household level, the market expanded into multiplicity of energy buyers and energy sellers. In this setting, new techniques to efficiently auction in the market are required in order to make the smart grid smarter. Ramachandran et al. (2011) developed a profit-maximizing adaptive bidding strategy based on hybrid-immune-system-based particle swarm optimization.

1.2 Components and features

Smart grids are built on different sub-systems and present special features that need to be attended. The sources of energy are heterogeneous (power plants, wind, sun, sea, etc) and might be intermittent. A key characteristic of a SG is that it supports two-way flow of electricity and information: a user might generate electricity and put it back into the grid; electric vehicles may be used as mobile batteries, sending power back to the grid when demand is high, etc. This backward flow is relevant, mainly in *microgrids*, where parts of the system that might be *islanded* due to power failures. Following [4], the three major systems in SG are:

- Smart infrastructure system that supports advanced and heterogeneous electricity generation, delivery and consumption. Is responsible for metering information and monitoring, and information transmission among of systems, devices and sensors.
- Management systems providing advanced management and monitoring, grid topology and control services. The objectives are energy efficiency improvement, supply and demand balance, emission control, operation cost reduction, and utility maximization.
- Protection system providing grid reliability analysis, failure protection, security and privacy protection services.

1.3 Advantages and challenges

Some of the anticipated benefits of a SG include [4]:

- improving power reliability and quality;
- optimizing facility utilization and averting construction of back-up (peak load) power plants;

[1] LIAAD - INESC TEC & Faculty of Medicine of the University of Porto, Portugal, email: pprodrigues@med.up.pt
[2] LIAAD - INESC TEC & Faculty of Economics of the University of Porto, Portugal, email: jgama@fep.up.pt

- enhancing capacity and efficiency of existing electric power networks, hence improving resilience to disruption;
- enabling predictive maintenance and self-healing responses to system disturbances;
- facilitating expanded deployment of renewable energy sources;
- accommodating distributed power sources, while automating maintenance and operation;
- reducing greenhouse gas emissions by enabling electric vehicles and new power sources, thus reducing oil consumption by reducing the need for inefficient generation during peak usage periods;
- presenting opportunities to improve grid security;
- enabling transition to plug-in electric vehicles and new energy storage options;
- increasing consumer choice, new products, services, and markets.

All these jointly lead to massive research problems that might be tackled by artificial intelligence techniques. Some challenges where machine learning can play a relevant role, include:

- The reliability of the system supports itself on millions of meters and other devices that require online monitoring and global asset management [2].
- Real-time simulation and contingency analysis of the entire grid have to be possible. However, not all operations models currently make use of real-time data [8].
- Interoperability issues that arise from the integration of distributed generation and alternate energy sources [17].
- The heterogeneity and volatility of smart grids require mechanisms to allow islanding [9] and self-healing [2].
- Finer granularity in management leads to strong demand response requirements [7] and dynamic pricing strategies [1].

2 THE DATA MINING POINT OF VIEW

Present SG monitoring systems suffer from the lack of machine learning technologies that can adapt the behavior of monitoring systems on the basis of the sequence patterns arriving over time. From a data mining point of view, a smart grid is a network (eventually decomposable) of distributed sources of high-speed data streams.

Smart meters produce streams of data continuously in real-time. A data stream is an ordered sequence of instances that can be read only once or a small number of times [6, 10], using limited computing and storage capabilities. These sources of data are characterized by being open-ended, flowing at high-speed, and generated by non stationary distributions.In smart grids the dynamics of data are unknown; the topology of network changes over time, the number of meters tends to increase and the context where the meter acts evolves over time.

In smart grids, several knowledge discovery tasks are involved: prediction, cluster (profiling) analysis, event and anomaly detection, correlation analysis, etc. However, different types of devices present different levels of resources and care should be taken in data mining methods that aim to extract knowledge from such restricted scenarios. All these characteristics constitute real challenges and opportunities for applied research in ubiquitous data mining. Generally, the main features inherent to ubiquitous learning algorithms are that the system should be capable of process data incrementally, evolving over time, while monitoring the evolution of its own learning process and *self-diagnosis* this process. However, learning algorithms differ in the extent of self-awareness they offer in this diagnosis. .

One of the most popular knowledge discovery techniques is *clustering*, the process of finding groups in data such that data objects clustered in the same group are more alike than objects assigned to different groups [6]. There are two different clustering problems in ubiquitous and streaming settings: *clustering sensor streams* and *clustering streaming sensors*. The former problem searches for dense regions of the data space, identifying hot-spots where sensors tend to produce data, while the latter finds groups of sensors that behave similarly through time [15]. We identify two different settings for clustering problems in smart grids. In the first setting a cluster is defined to be a set of sensors (meters, households, generators, etc.). In the second setting, a cluster is defined to be a set of data points (demand, supply, prices, etc.) generated by multiple sources.

2.1 Research on clustering electrical networks

Several real-world applications use machine learning methods to extract knowledge from sensor networks. The case of electricity load demand analysis is a paradigmatic one that has been (and continues to be) studied. Sensors distributed all around electrical-power distribution networks produce streams of data at high-speed. Three major questions rise: a) can we define consumption profiles based on similar sensors? b) can we find global patterns in network consumption? and c) can we manage the uncertainty in sensor data?

To efficiently find consumption profiles, clustering techniques were applied to the streams produced by each sensor, either hierarchically at a central server [16] or distributed in the network [15]. Although the problem is still very hard to model, given the dimensionality of the networks at stake, the incremental systems evolved and adapt to changes in the data, bridging the gap to future paths of research. Regarding global network patterns, related research has resulted in a system that distributes the clustering process into local and central tasks, based on single sensor data discretization and centralized clustering of frequent states [5]. But data and models are both uncertain. For example, if a sensor reads 100, most of times it could be 99 or 101. This uncertainty has been tackled by reliability estimators and improved predictions using those estimates [13], but reliability for clustering definitions is still uncharted territory.

2.2 Clustering as a smart grid problem solver

In this work we argue that major smart grids problems previously enunciated can and should be addressed as unsupervised machine learning problems.

Power balance Power balance is the most basic-level problem that smart grids need to solve before anything else. The strongest requirement is that energy is available in the entire network. Hence, clustering the data and sources together to find hot-spots can detect specific points of danger in the network.

Multiple alternate sources In smart grids, supply and demand must be leveled across multiple alternate sources. Hence, combining clustering definitions for power demand and power supply should give indications on how to better level the sources.

Contingency analysis Contingency analysis tries to produce detection and reaction mechanisms to specific unexpected problems. Hence, monitoring the evolution of clusters of nodes, should help on detecting drifting sources of demand or supply.

Islanding Islanding is a concept that is directly connected with clustering, in the sense that it searches for subnetworks where demand and supply are leveled. Hence, local distributed clustering of sources and data should produce the expected definitions.

Self-healing Self-healing relates to the ability to rearrange and adapt the network on-the-fly to meet unexpected changes. Hence,

ad-hoc distributed clustering of sources, independently from a centralized server, should produce procedures for self-healing.

Online monitoring and asset management These features are strongly connected with incremental learning and adaptation of learned models. Hence, incremental models for sources and data clustering, and their evolution, should provide basic information.

Dynamic energy pricing Energy pricing largely depends on supply and demand balance. Hence, clustering power demand and supply together with buy and sell prices, should give insights on prospective energy pricing.

3 HOLISTIC DISTRIBUTED CLUSTERING

The smart grid produces different types of data, on each source (node or subnetwork), which must be taken into account: power demand, power supply, energy sell price, energy buy price. As previously stated, two clustering problems exist: clustering data and clustering data sources. This way, each node might be assigned to a cluster on (at least) eight different clustering definitions. For all problems, there is a common requirement: each node (meter) should process locally their own data. Only aggregated data should be shared between the different nodes in the grid.

From the previous section it became clear that a holistic approach to clustering in smart grids is needed and should produce benefits to energy sustainability. In this section we present such a proposal, based on two existing works on stream clustering (L2GClust and DGClust) and their prospective integration in a multi-dimensional clustering system. Next sections present the original clustering algorithms, their application to electricity demand sensor data streams, and how they could be merged into a holistic clustering system.

3.1 L2GClust: Distributed clustering of grid nodes

Clustering streaming data sources has been recently tackled in research, but usual clustering algorithms need the data streams to be fed to a central server [15]. Considering the number of sensors possibly included in a smart grid, this requirement could be a bottleneck. 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 [15]. *L2GClust* has 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.

Global evaluation of the L2GClust algorithm on synthetic data revealed high agreement with the centralized, yet streaming, counterpart, being especially robust in terms of cluster separability. Also, for stable concepts, empirical evidence of convergence was found. On the other hand, sensitivity analysis exposed the robusteness of the local algorithm approach. Figure 1 shows that agreement levels are robust to an increase on the number of clusters, being, however, a bit more sensitive with respect to network size and cluster overlapping. Nonetheless, the robusteness to network communication problems is exposed, as the proportion of agreement is harmed only for high levels of communication incompleteness.

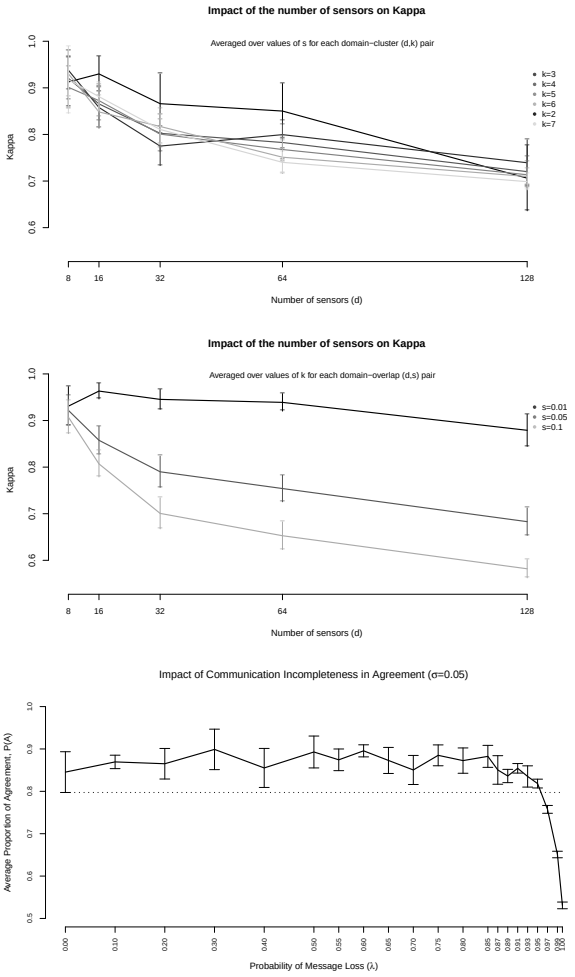


Figure 1. L2GClust: sensitivity of $\hat{\kappa}$ statistic to the number of sensors (d), for different number (k) and overlap (s) of clusters. Bottom plot presents the impact of communication incompleteness on average proportion of agreement for 5 clusters in a 128 sensor network.

One important task in electrical networks is to define profiles of consumers, to better predict their behavior in the near future. L2GClust was applied to a sample of an electrical network to try to find such profiles. From the raw data received at each sub-station, observations were aggregated on a hourly basis over more than two and a half years [14]. The log of electricity demand data from active power sensors was used to check whether consumer profiles would rise. The log has hourly data from a subsample (780 sensors) of the entire data set ($\sim$4000 sensors). Since no information existed on the actual electricity distribution network, the simulator used this dataset as input data to a random network and monitored the resulting clustering structures. Unfortunately, real data is never clean, and half of the sensors have more than 27% missing values, which naturally hindered the analysis. Given this, and the dynamic nature of the data, no convergence was possible in the clustering structures. However, we could stress that, as more data is being fed to the system, better agreement can be achieved with the centralized approach, as exposed in Figure 2. Hence, not only does the agreement tend to increase with more observations, but also changes on the clustering structure are apparently possible to detect. L2GClust presented good characteris-

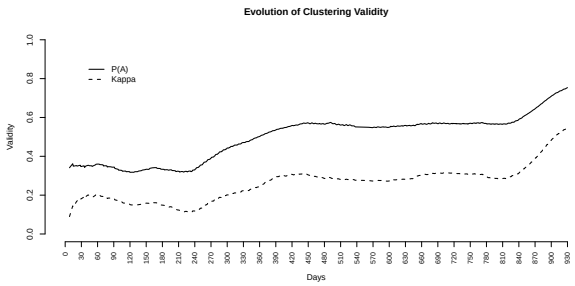


Figure 2. L2GClust evolution of clustering agreement (probability of agreement and $\hat{\kappa}$ statistic) for a real active power sensor data log.

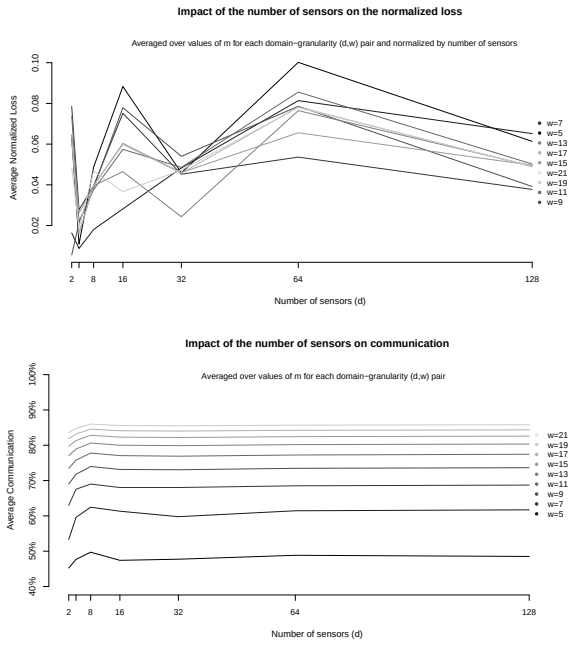


Figure 3. DGClust: impact of the number of sensors on loss to real centroids (top) and communication reduction (bottom) [5].

tics to find clusters of sensors in wide networks such as smart grids.

3.2 DGClust: Grid clustering of grid data streams

Clustering data points is probably the most common unsupervised learning process in knowledge discovery. In ubiquitous settings, however, there aren't many tailored solutions to try to extract knowledge in order to define dense regions of the sensor data space. Clustering examples in sensor networks can be used to search for hotspots 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 [5]. 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. 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.

The *Distributed Grid Clustering* (DGClust) algorithm was proposed for clustering data points produced on wide sensor networks [5]. The rationale is to use: a) online discretization of each single sensor data, tracking changes of data intervals (states) instead of raw data (to reduce communication to central server); b) frequent state monitoring at the central server, preventing processing all possible state combinations (to cut computation); and c) online clustering of frequent states (to keep high validity and adaptivity). Each local sensor receives data from a given source, producing a univariate data stream, which is potentially infinite. Therefore, each sensor's data is processed locally, being incrementally discretized into a univariate adaptive grid. Each new data point triggers a cell in this grid, reflecting the current state of the data stream at the local site. Whenever a local site changes its state, that is, the triggered cell changes, the new state is communicated to a central site. Furthermore, the central site keeps the global state of the entire network where each local site's state is the cell number of each local site's grid. Nowadays, sensor networks may include thousands of sensors. This scenario yields an exponential number of cell combinations to be monitored by the central site. However, it is expected that only a small number of this combinations are frequently triggered by the whole network, so, parallel to the aggregation, the central site keeps a small list of counters of the most frequent global states. Finally, the current clustering definition is defined and maintained by an adaptive partitional clustering algorithm applied on the frequent states central points.

To evaluate the sensitivity of the system to the number of sensors, synthetic data was used and the average result for a given value of granularity (w), averaged over all values of number of frequent states to monitor (m, as loss seemed to be only lightly dependent on this factor) was analyzed. In figure 3 we note no clear trend, strengthening the evidence of robusteness to wide sensor networks. Regarding communication reduction when compared with centralized clustering, figure 3 also shows that the amount of communication reduction does not depend on the number of sensors. This way, the benefits of reduced transmission rates are extensible to wide sensor networks.

3.3 HDClust: Holistic Distributed Clustering

The two algorithms previously exposed are designed for streaming data, and work with reduced computational costs in terms of memory and communications bandwidth. They present strong characteristics that could be even improved if used together. In L2GClust, each sensor node each node has an approximation of the global clustering. In DGClust, a centralized site maintains the global cluster structure of the entire network at reduced communication costs. The main idea of the *Holistic Distributed Clustering* (HDClust) is to integrate the local distributed approach of L2GClust, with the grid data clustering approach of DGClust, in order to achieve the holistic clustering of data and sources on sensor networks such as smart grids. Specifically, for each measured dimension:

- each local node (meter) keeps a sketch of its own data streams (as in L2GClust) and a local discretization grid (as in DGClust);
- communication is restricted to the neighborhood (as in L2GClust);
- at regular intervals, each local node receives from its neighbors the estimates of the clusters centroids (as in L2GClust) and the current data discretized grid cell (as in DGClust);
- each node keeps an estimate of the global clustering of nodes by clustering neighbors' centroids (as in L2GClust);
- each node keeps a frequent state list and maintains a clustering of the most frequent states (as in DGClust) from the neighbors;

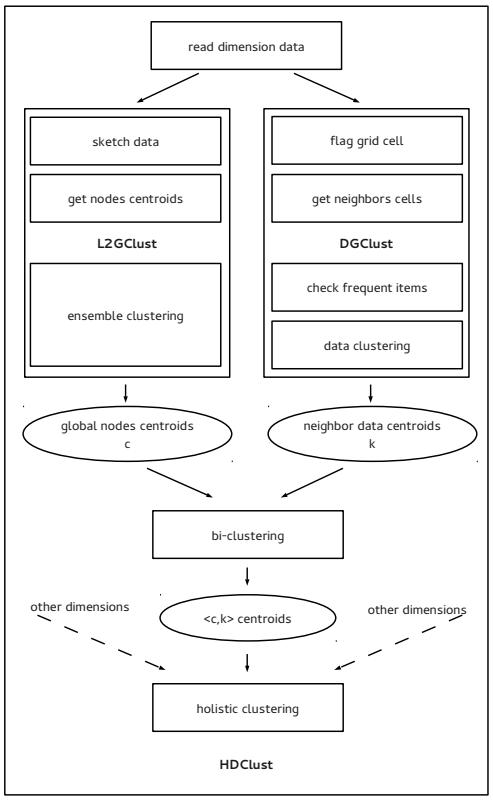


Figure 4. HDClust schema to be applied at each node, for each included dimension. Left branch applies L2GClust while right branch applies DGClust using data from the neighbors, each node acting also as central clustering agent. Both clustering definitions are then combined and integrated with other measured dimensions.

- to link clustering of sources with clustering of data, each node also receives from the neighbors their self assignment to a cluster.

In the resulting cluster structure, each sensor maintains **C** clusters of data sources, and **K** clusters of data points.

In a smart grid context, and taking advantage of the decomposable property of the grid network (microgrids), L2GClust and DGClust can work together. Assume a microgrid of **D** sensors, and 4 dimensions or quantities of interest: power demand, power supply, energy sell price and energy buy price. The resulting HDClust, the network is summarized by **C** clusters of data sources, and **K** clusters of data points, for each quantity of interest. In real-time and at each moment, each sensor is in a state $\langle c_i, k_i \rangle$ in each dimension. Figure 4 presents the global schema for a holistic approach to clustering, to be applied at each node of a smart grid. The combination of the characteristics both algorithms seems not only possible, but extremely relevant as complementary knowledge discovery in a holistic view of the grid.

4 REMARKS AND FUTURE PATHS

Smart grids are a paradigmatic example of ubiquitous streaming data sources. Data is produced at high speed, from a dynamic (time-changing) environment. Meters are geographically distributed, forming a network. On top of clustering algorithms, several tasks can be computed: profiling, anomaly and event detection, outliers detections, trends, deviations, etc. In this paper, we have discussed distributed clustering algorithm for data streams produced on wide sensor networks like smart grids. Furthermore, we have shown how smart grid problems can be addressed as clustering problems, and proposed a holistic approach to better extract knowledge from the grid. We believe that this holistic approach could be used to help solving some of the smart grid intelligent layer research problems. Current research focus on the integration of both algorithms into the schema and its evaluation on real-world electrical networks data.

ACKNOWLEDGEMENTS This work is funded by the ERDF through Programme COMPETE and by the Portuguese Government through FCT, projects PEst-C/SAU/UI0753/2011 and PTDC/EIA/098355/2008. The authors acknowledge the help of Luís Lopes and João Araújo.

REFERENCES

[1] H. Allcott, 'Rethinking real-time electricity pricing', *Resource and Energy Economics*, **33**(4), 820–842, (2011).
[2] S.M. Amin, 'Smart grid: Overview, issues and opportunities. advances and challenges in sensing, modeling, simulation, optimization and control', *European Journal of Control*, **17**(5-6), 547–567, (2011).
[3] F. Benzi, N. Anglani, E. Bassi, and L. Frosini, 'Electricity smart meters interfacing the households', *IEEE Transactions on Industrial Electronics*, **58**(10), 4487–4494, (2011).
[4] Xi Fang, Satyajayant Misra, Guoliang Xue, and Dejun Yang, 'Smart grid – the new and improved power grid: a survey', *IEEE Communications Surveys & Tutorials*, (2012). (to appear).
[5] João Gama, Pedro Pereira Rodrigues, and Luís Lopes, 'Clustering distributed sensor data streams using local processing and reduced communication', *Intelligent Data Analysis*, **15**(1), 3–28, (January 2011).
[6] Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan, 'Clustering data streams: Theory and practice', *IEEE Transactions on Knowledge and Data Engineering*, **15**(3), 515–528, (2003).
[7] A. Iwayemi, P. Yi, X. Dong, and C. Zhou, 'Knowing when to act: An optimal stopping method for smart grid demand response', *IEEE Network*, **25**(5), 44–49, (2011).
[8] J.A. Kavicky, 'Impacts of smart grid data on parallel path and contingency analysis efforts', in *IEEE PES General Meeting*, (2010).
[9] R.H. Lasseter, 'Smart distribution: Coupled microgrids', *Proceedings of the IEEE*, **99**(6), 1074–1082, (2011).
[10] S. Muthukrishnan, *Data Streams: Algorithms and Applications*, Now Publishers Inc, New York, NY, 2005.
[11] A. Pasdar and H.H. Mehne, 'Intelligent three-phase current balancing technique for single-phase load based on smart metering', *International Journal of Electrical Power and Energy Systems*, **33**(3), 693–698, (2011).
[12] B. Ramachandran, S.K. Srivastava, C.S. Edrington, and D.A. Cartes, 'An intelligent auction scheme for smart grid market using a hybrid immune algorithm', *IEEE Transactions on Industrial Electronics*, **58**(10), 4603–4612, (2011).
[13] Pedro Pereira Rodrigues, Zoran Bosnić, João Gama, and Igor Kononenko, 'Estimating reliability for assessing and correcting individual streaming predictions', in *Reliable Knowledge Discovery*, 267–287, Springer Verlag, (2012).
[14] Pedro Pereira Rodrigues and João Gama, 'A system for analysis and prediction of electricity load streams', *Intelligent Data Analysis*, **13**(3), 477–496, (June 2009).
[15] Pedro Pereira Rodrigues, João Gama, João Araújo, and Luís Lopes, 'L2GClust: Local-to-global clustering of stream sources', in *Proceedings of ACM SAC 2011*, pp. 1011–1016, (March 2011).
[16] Pedro Pereira Rodrigues, João Gama, and João Pedro Pedroso, 'Hierarchical clustering of time-series data streams', *IEEE Transactions on Knowledge and Data Engineering*, **20**(5), 615–627, (May 2008).
[17] K. Tanaka, A. Yoza, K. Ogimi, A. Yona, T. Senjyu, T. Funabashi, and C.-H. Kim, 'Optimal operation of dc smart house system by controllable loads based on smart grid topology', *Renewable Energy*, **39**(1), 132–139, (2012).

Efficient Mobility Pattern Stream Matching on Mobile Devices

Simona-Claudia Florescu[1] and **Michael Mock**[1] and **Christine Körner**[1] and **Michael May**[1]

Abstract. The increasing amount of mobile phones that are equipped with localization technology offers a great opportunity for the collection of mobility data. This data can be used for detecting mobility patterns. Matching mobility patterns in streams of spatio-temporal events implies a trade-off between efficiency and pattern complexity. Existing work deals either with low expressive patterns, which can be evaluated efficiently, or with very complex patterns on powerful machines. We propose an approach which solves the trade-off and is able to match flexible and sufficiently complex patterns while delivering a good performance on a resource-constrained mobile device. The supported patterns include full regular expressions as well as relative and absolute time constraints. We present the definition of our pattern language and the implementation and performance evaluation of the pattern matching on a mobile device, using a hierarchy of filters which continuously process the GPS input stream.

1 INTRODUCTION

The analysis of mobility behavior based on GPS-tracks has become a popular field of research [15, 7, 4, 16, 18]. In the context of the European LIFT [10] project, we aim at the on-line monitoring of global non-linear phenomena from massively distributed streams of data. In the mobility domain such global phenomena are, for example, mass events or changes in traffic flows. The basic approach of LIFT technology for the reduction of communication overhead is to build local mobility models on each device and to communicate only significant changes to a central coordinator, which is computing the global model. This paper presents an approach for building the local mobility model efficiently on a mobile device.

Mobility patterns such as used in [4] and [7] are an appropriate way of modeling spatio-temporal mobility behavior. Powerful specialized database systems such as [16] allow to retrieve patterns from spatio-temporal data using complex pattern queries, in which spatial and temporal conditions can be freely combined. Providing this flexibility for pattern definitions for building local mobility models on a mobile device would surely exceed the computational power of such devices. Patterns expressed by regular expressions only, but not supporting queries over travel times (as in [4]) might have a better chance of being efficiently implemented on a mobile device. The same holds for the work of [7], which allows queries over travel times but supports sequential patterns only. Our approach of building mobility models is based on the notion of *visits* as being formally introduced in [12, 11]. Patterns are build as regular expression over *visits*, and time constraints are applied to complete patterns. For achieving an efficient implementation on the mobile device, we spread the task of pattern matching over a filter hierarchy that is fed with the stream of GPS input data: Firstly, a *VisitEventFilter* detects whether a certain location is being visited and, if so, forwards a *visit event* to a *PatternFilter*, which can handle arbitrary regular expressions (including Kleene closure) over visit events. Lastly, a *TimeConstraintFilter* is used to check any expression over the travel time for the complete pattern. By this approach, we can use standard deterministic automatons for implementing matching of regular expressions and can perform efficient time constraint checking in constant time. The remainder of this paper is structured as follows: in the next section, we present our approach, *mobility pattern matching over streams* containing the pattern definition language and details of the implementation of the pattern matching algorithm. Section 3 contains the performance and scalability evaluation and Section 4 discusses related work. The last section, conclusions, provides a short summary, improvement suggestions and future work.

[1] Fraunhofer Institute for Intelligent Analysis and Information Systems, Germany, email:simona.florescu@gmail.com, firstname.lastname@iais.fraunhofer.de

2 MOBILITY PATTERN MATCHING

Figure 1 describes our general approach for building local and global mobility models. As described in [11], our mobility model is based on counts of occurrences of events, whereby an event represents the occurrence of a specific predefined spatio-temporal behavior in the observed GPS track. The *local mobility model* represents the behavior of a specific user and is locally computed on the device itself, whereas the *global model* is build by aggregating all local models on a single node (global coordinator). LIFT technology is used to reduce the amount of communication needed for maintaining the global model correct over time. The basic approach thereby is to define a so-called *SafeZone*, in which the local model can safely vary without notifying the global coordinator [17]. In this paper, we focus on the question whether the input for generating the local model can be computed efficiently on a mobile device, i.e., the gray shaded part in Figure 1. Being able to compute a model locally is a prerequisite for applying LIFT technology for communication reduction. The local mobility model is computed by processing the stream of GPS updates as provided by an *Android Location Provider* through a hierarchy of filters (see [8] for the details of filter interface definitions). At the first level, the *VisitEventFilter* detects whether the device stays for a pre-defined minimum time in one of the pre-defined locations, which are stored in the local location database. If so, a *visit event* is generated, which will be processed at the next layer in the hierarchy, the *PatternFilter*. This Filter takes list of predefined patterns (regular expressions over visits) as input and matches the incoming *visit events* against these patterns. In case of a match, a *pattern event*

is forwarded to the next filter. At the last filter level, the *TimeConstraintFilter*, the time constraints for the matched pattern are validated. If they are fulfilled, the respective pattern frequency count is increased. The input for our implementation consists therefore of: (1) an infinite stream of GPS-sensed location updates, (2) a given set of interesting locations to be monitored, (3) a set of patterns with the set of interesting locations as domain, as depicted in the figure below, Figure 1.

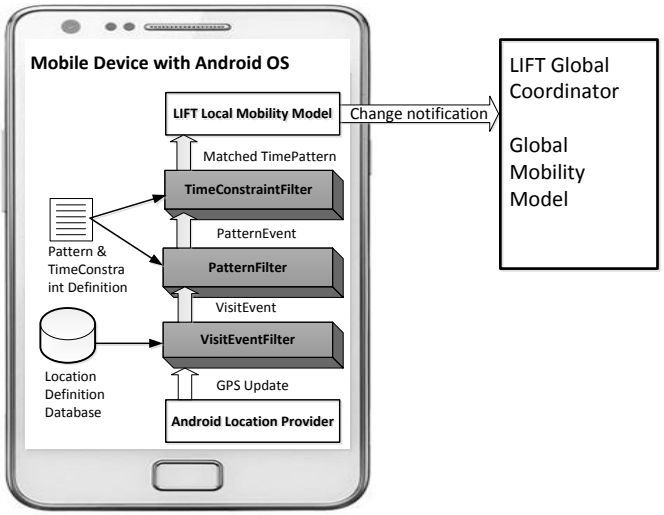


Figure 1. Filter hierarchy and data flow of the approach.

2.1 Pattern Matching Language

We propose a pattern language based on regular expressions. We define the language for the three main levels of our approach: ***VisitEventFilter***, ***PatternFilter*** and ***TimeConstraintFilter***.

Firstly, we define a ***location***, of which the input data for the *VisitEventFilter* consists. A location is defined by an *id*, a *type*, a *spatial extend* and the minimum *stay time* at the location. Note that this definition allows for overlapping locations (for example: a location for a specific attraction inside the location "Amusement Park") as well as for monitoring complete regions by dividing a region in a spatial grid of locations. In our implementation we consider rectangular spatial shapes, therefore we define the four coordinates of the bounding boxes. The id is a unique identification for the location and the type of location (e.g. cinema, fast-food, school) is coded for shortness purposes with two digits. The minimum stay time defines the time period that an encounter with a location must last in order to become a visit.[2] A location is defined as:

$$location := id, type, x_{min}, x_{max}, y_{min}, y_{max}, minStay \quad (1)$$

We represent a *visit event*, generated by the *VisitEventFilter* with the following attributes: location identification, location type, entry and exit time (in milliseconds):

$$visitEvent := (id, type, entryTime, exitTime) \quad (2)$$

We define a *visitExpression* as being the concatenation of the *id*, *type* and the *stayTime* using the within separator ",":

$$visitExpression := concat(id, type, stayTime, sep = ",") \quad (3)$$

The stay time is the difference between the exit and entry time. Similarly to [4] it is expressed as a sequence of repeated time units t so that it can be matched by regular expressions. This enables pattern queries like "a stay time of at least 5 and at most 20 minutes". The duration of a time unit depends on the required accuracy and can be set to e.g. one minute. An example of a visit expression is: ***1,01,tttt*** which represents a visit event with location 1, of type 01 (here code for cinema) and a stay time of 4 time units.

A ***pattern*** consists of (1) a regular expression of one or several *visitExpressions* and (2) a *timeConstraint* - containing absolute and relative constraints:

$$pattern := (regex(visitExpression+), timeConstraint) \quad (4)$$

$$timeConstraint := ([f_{exit}], [op_f], [l_{entry}], [op_l], [lc], [rc]) \quad (5)$$

The regular expression is defined according to the regular language specified in [14] on the alphabet of *visitExpressions* (Definition 3). Several expressions are hereby separated by a semicolon. In a digital format we represent a pattern in XML (Extended-Markup-Language). Figure 2 shows an example of a mobility pattern. The pattern's XML representation is shown in the code snippet below Figure 2. The part of the pattern containing the regular expression is

```
1,01,t{0,4};@;2,02,t{4,8};
```

It denotes a visit to location 1 (of type 01) for up to 4 time units followed by an arbitrary number of visits to unspecified locations (expressed by @[3]), followed by a visit to location 2 (of type 02) for a stay time between 4 and 8 time units.

The ***time constraint*** of the pattern definition (Definition 5) is checked by the ***TimeConstraintFilter*** and contains absolute and relative constraints. The **absolute time constraints** are: (1) f_{exit} - the constraint on the first event exit time in milliseconds; (2) op_f - the operator applied to f_{exit} with the following possible values: 0 for less, 1 for equals, 2 for greater than, - for none; (3) l_{entry} - the constraint on the last event entry time in milliseconds and (4) op_l - the operator applied to l_{entry} with the same values as op_f. In the example given below no absolute time constraints are specified but, for example, the values: $f_{exit} = 1,000,000$ and $op_1 = 2$ would impose on the first visit event that its *exitTime* must be greater than 1,000,000. The **relative time constraints** are: (5) *lc* - the left constraint for the pattern duration in milliseconds and (6) *rc* - the right constraint for the whole pattern in milliseconds e.g. $lc < l_{entry} - f_{exit} < rc$. For the given example the relative time constraint is: $0 < visitEvent_{id=2}.entryTime - visitEvent_{id=1}.exitTime < 7,200,000$.

Below, the XML for the pattern depicted in Figure 2 is shown containing the pattern id, the regular expression of the sequence of visits as well as the time constraints.

[2] Both the bounding box radius and minimum stay time are defined application-depend, depending on the location type (e.g. for bigger locations we set the minimum stay time higher) in order to distinguish between *passing by* and *visiting*.

[3] In [14] the JAVA library *automaton* specifies a regular language implementation where the symbol '@' represents any sequence of characters

Figure 2. Mobility pattern example: a visit to location 1 followed by an arbitrary number of visits to intermediate locations followed by a visit to location 2 with a maximum time period of 2 hours (7,200,000 milliseconds) between the first and last visit

```
<?xml version='1.0'?>
<PatternList>
      <Pattern>
        <id>1</id>
        <regex>1,01,t{0,4};@;2,02,t{4,8};</regex>
        <tc>
              <f_exit>-</f_exit>
              <op_f>-</op_f>
              <l_entry>-</l_entry>
              <op_l>-</op_l>
              <lc>0</lc>
              <rc>7200000</rc>
        </tc>
      </Pattern>
</PatternList>
```

2.2 Pattern Matching Algorithm

Our approach consists of several filters implemented in an embedded Android application: a *VisitEventFilter*, a *PatternFilter* and the *TimeConstraintFilter* which return a pattern distribution from a GPS stream input, see Figure 1 and Section 1.

The first filter, the ***VisitEventFilter***, receives as input the stream of GPS coordinates and a set of locations stored in a local SQLite database and generates visit events. In order to do so it checks whether there is a spatial match between the coordinates and the input of locations, described in Section 2.1 *Pattern Matching Language*. The input database contains the tables, which are joined by their id:

$Locations1(id, x_{min}, y_{min}, y_{max}, y_{max})$
$Locations2(id, latitude, longitude, name, type, min\_stay)$

The main steps of the visit filtering approach are: firstly, the database is queried to retrieve the ids of the locations in which the current position is in. As we are restricting the current implementation to rectangular locations, this can be achieved by a first query such as:

SELECT Id FROM $Locations1$ *WHERE* $x \geq x_{min}$ *and* $x \leq x_{max}$ *and* $y \geq y_{min}$ *and* $y \leq y_{max}$

followed by another query on the *Locations2* table for retrieving the rest of the location information (Definition 1).

Secondly, we are maintaining a list of *entered* locations. Whenever we detect that the current GPS point is no longer in a specific one of those entered locations, we check whether we have been staying at least a time of *minStay* within that location and, if so, generate a visit event for that location. In any case, the location is removed from the list of *entered* locations. The complete algorithm can be found in [6].

In the ***PatternFilter*** we model the patterns using *deterministic finite automata* [9]. We instantiate an automaton for each of the parsed patterns from the XML input. In the *PatternMatcher* class we create and model automata using the JAVA library *automaton* [14] to match the regular expressions specified in the first part of Definition 4.

The class structure of the pattern matching algorithm consists of:

- A *PatternFilter* class, which instantiates in its constructor a list of *PatternMatcher* objects by calling the *PatternReader* class. The *PatternFilter* maintains the list of all patterns. It receives visit events in its *update* function and generates and forwards pattern events to the next filter, the *TimeConstraintFilter*.
- A *PatternMatcher* class where an automaton is modeled. In the constructor the variables needed for saving the *automaton* data structure are initialized as well as a pattern event object for storing the information of the matched pattern.
- The *PatternReader* reads and converts a pattern from XML to an object of type *PatternMatcher*.
- The filter class *TimeConstraintFilter* checks if the time constraints are fulfilled for a received pattern event. In its constructor, it reads and parses the time constraints for each pattern into a *TimeConstraint* object.

In the *PatternMatcher* constructor, provided in the pseudocode of Class 1, the field `automaton` represents an object of type *RunAutomaton*, defined in the JAVA library *automaton*. The fields for defining the state of the automaton are `actualState` and `isInitial`. The `patternEvent` is an object of type `Event` which stores the properties of the generated pattern event for each match. The logic of the *PatternMatcher* is contained in the functions *processVisit* and *reset* shown in the pseudocode of Class 1. The `processVisit` function receives a `visitEvent` as a parameter, generated by the previous filter (the *VisitEventFilter*), and returns a boolean value of true if the visit event completes the matching of a given pattern and false if not. The *processVisit* function generates the `visitExpression` which has the structure given in Definition 3. The `stepThrough` function runs through the automaton with each character from the `visitExpression` as transition and returns the step obtained after the run. A value for `step` of -1 means that the matching failed. Any other value means that the automaton advances in another state, changing variable `actualState`. In this case and if the `isInitial` was `true`, the `patternEvent` stores the `exitTime` of the visit event. If the `visitExpression` could not be matched, the function `reset` is called. There, the `patternEvent` attributes are set to null and the state of the automaton is set to initial. Another check in the function is whether the `automaton` has reached the final state. In this case the `patternEvent` stores the `entryTime` of the visit event since this is its last visit event matched in the pattern.

The pseudocode of the *PatternFilter* is shown in Class 2. In the constructor a list of *PatternMatcher* objects is generated, one for each given pattern. Further, the *PatternReader* class, which reads and parses the XML input patterns, is called. For each pattern string the *PatternReader* instantiates a *PatternMatcher* by calling its constructor as shown in Class 1. In the `update` function of the *PatternFilter*, for each new visit event update, all the existing *PatternMatcher* objects from the `patternMatcherList` are traversed and called to execute the `processVisit` function, shown in Class 1. If the returned value from `processVisit` is true, then the matched pattern event, `patternEvent` is forwarded to the next filter.

Finally, in the ***TimeConstraintFilter*** the matched pattern time constraints are checked, if any are provided. The time constraints are de-

Class 1 PATTERNMATCHER

Fields:

$automaton$	- deterministic automaton for the pattern
$actualState$	- the actual state of the automaton
$patternEvent$	- an *Event* object for a matched pattern
$isInitial$	- boolean value for initial state of automaton

Constructor: $PatternMatcher(id, regex)$

id	- pattern id
$regex$	- regular expression

```
1: this.patternEvent.id ← id
2: this.automaton ← new RunAutomaton(regex)
   // using DFA java library from [14]
3: this.reset()
```

reset()

```
4: this.patternEvent.entryTime ← null
5: this.patternEvent.exitTime ← null
6: this.actualState ← (this.automaton.getInitialState())
7: this.isInitial ← true
```

boolean processVisit(visitEvent)

```
8: visitExpression ← makeVisitExpression(visitEvent)
9: step ← automaton.stepThrough(actualState, visitExpression)
   // stepThrough returns the state reached by inputting all characters of the visitExpression to the automaton starting with actualState
10: if step ≠ −1 then // step is -1 for a mismatch
11:     if this.automaton.isInitial then
12:         this.patternEvent.entryTime ← visitEvent.exitTime
13:         this.isInitial ← false
14:     end if
15:     actualState ← step
16: else // the visitExpression could not be entirely matched
17:     this.reset()
18: end if
    // check whether the automaton has reached the end state
19: if automaton.isFinal() then
20:     this.patternEvent.exitTime ← visitEvent.entryTime
21:     this.reset()
        // automaton is set on the initial state and all variables are reinitialized
22:     return true
23: else
24:     return false
25: end if
```

Class 2 PATTERNFILTER

Fields:

$patternMatcherList$	- a list of objects of type *PatternMatcher*

Constructor: $PatternFilter()$

```
1: reader = newPatternReader()
2: this.patternMatcherList ← reader.parseAutomaton(input_file)
   // instantiate list of PatternMatcher
```

update(visitEvent)

```
3: for PatternMatcher ∈ patternMatcherList do
4:     if PatternMatcher.processVisit(visitEvent) then
5:         forward(PatternMatcher.patternEvent)
6:     end if
7: end for
```

fined in Section 2.1 (*Pattern Matching Language*) and relate to the first and last event in the matched pattern, respectively. The *TimeConstraintFilter* constructor instantiates a *TimeConstraint* object for each pattern. When invoked with a pattern id of the incoming pattern event, it checks the existing time constraints for the respective pattern id. The generated event at this level is a matched time pattern.

3 PERFORMANCE EVALUATION

Our performance evaluation for checking the potential of the application in practice is based on synthetic data. The tests were run on a Samsung Galaxy SII GT-I9100, operating Android Gingerbread, version 2.3.3. The GPS stream data consists of synthetically generated coordinates. In addition, we retrieved 800,000 points of interest (POI) from the geo-service OSM [13] for the location set. The POIs were obtained for Germany and are of 15 different types. The generated patterns are formulated similarly to the example pattern in Section 2.1 and Figure 2, i.e. they specify the first and last location and allow an arbitrary number of visit events in between. In addition, all patterns possess time constraints. Our artificial GPS data is generated such that 40% of all points lead to a match with the location database and generate a *visitEvent*. For performance evaluations of the *PatternFilter* 2,6% of the input *visitEvents* complete a pattern match. Finally, each pattern match is checked in the *TimeConstraintFilter*.

#Locations	10	100	1K	10K	100K	500K	800K
Run-time (ms)	0.72	0.69	0.79	0.82	0.75	0.73	0.75
Stddev run	0.90	0.71	0.71	0.81	0.78	0.66	0.79
DBQuery (ms)	0.35	0.33	0.39	0.42	0.35	0.37	0.36
Stddev DBQuery	0.60	0.46	0.54	0.67	0.38	0.54	0.55

Table 1. The run-time values for the *VisitEventFilter*.

#Patterns	100	1K	10K	100K	300K
Run-time (ms)	1.2	17.1	157.0	1498.4	4397.3
Stddev.	0.08	6.5	7.1	24.0	129.6
Start-up (ms)	266.3	275.1	239.2	499.9	1341.8
Stddev.	201.6	225.1	48.2	9.3	15.3
Heap size (MB)	2.67	2.68	2.75	2.76	2.76
Memory (MB)	5	10	10	12	24

Table 2. The run-time values for the *PatternFilter*

Firstly, we measured the run-time for the ***VisitEventFilter***. We varied the size of the location set from 10 to 800,000. Table 1 shows the obtained performance results. We distinguish between the database query (DBQuery) and the entire run-time (Run-time) measured before forwarding a visit event, therefore the DBQuery run-time is included in the run-time value. Table 1 shows that the run-time of the *VisitEventFilter* is nearly constant when varying the number of locations in the underlying database. We ascribe this behavior to caching effects in the SQLite database, taking into account that the database is opened only once and that all queries are read-only. Ongoing experiments and code analysis showed that the time needed for the database part in the *VisitEventFilter* depends on the number of overlapping locations in which a given GPS point resides. In the synthetic GPS tracks used in the evaluation, the GPS points match to one (non-overlapping) location only. All visit events are detected in a time well below one second, which is the time difference between two GPS location updates in worst case. In addition, the number of monitored locations will be geographically limited in practice. For example, when restricting the collected POI (points-of-interest) to the city of Cologne, we obtained a set of about 20,000 locations.

Secondly, we evaluated the run-time of the ***PatternFilter*** under an increasing number of checked patterns with an average matching percentage of 2,6%. The results are shown in Table 2. The table also contains the reading and parsing time for all patterns (Start-up), which takes place in the *PatternFilter* constructor. The run-time for the pattern matching increases linearly with the number of tested patterns (Run-time), which corresponds to the main loop executed in the *PatternFilter.update* method. Furthermore, for each of the instances of the *PatternFilter* we show the heap size and the allocated memory. The values were captured for about three runs as delivered by the Android debugger class. The results for the heap size are relatively constant and show for the memory allocations fair results, since the high number of 300,000 patterns use about 24MB out of 64MB.

The ***TimeConstraintFilter*** has a constant run-time, since it only retrieves the time constraints for the respective pattern id from a *HashMap*, and the time constraint validation is trivial. The run-time

for the *TimeConstraintFilter* is approximately 10 milliseconds per pattern event.

4 RELATED WORK

Table 3 gives an overview on related work in the field of spatio-temporal mobility analysis. The criteria on which we compare related work are: (1) stay time (Stay) - patterns with conditions on minimum and maximum stay time, (2) travel time (TT) - the time span between two locations in the pattern, (3) the time constraints (TC) - time constraints on the full pattern, i.e. on first and last location in the pattern, (4) full regular expressions (FullRE) - supporting all the expression options from regular expressions i.e. Kleene closure (+,*), negation, conjunction, disjunction, and (5) predicates - additional complex conditions on the pattern (e.g. an attribute should have an incrementing value) and (6) Stream - whether the approach is applied on-line (on stream data) or later, off-line. Although our matched patterns are not the most complex, our approach is the first one successfully tested on a resource-constrained device.

Approach	Stay	TT	TC	FullRE	Pred.	Stream
T-patterns [7]	-	✓	-	-	-	-
Mob. patterns [4]	✓	-	-	✓	-	-
SASE [18]	✓	-	✓	✓	✓	✓
SASE$^+$ [1]	✓	-	-	✓	✓	✓
Cayuga [2]	-	-	✓	✓	✓	✓
STPQ [16]	✓	✓	✓	✓	✓	-
Our Approach	✓	-	✓	✓	-	✓

Table 3. Comparison with related work

The function `addProximityAlert` provided by the Android `LocationManager` [3] performs a similar task as our *VisitEventFilter*. Comparative experiments between both classes showed that the Android function can register proximity alerts for only less than 30,000 locations, compared to our approach, which has been tested for up to 800,000 locations.

5 CONCLUSION

In this paper we have shown that the detection of state-of-the art complex mobility patterns can be implemented on a resource-constrained environment such as a mobile device. Our experiments show that the pattern matching can process the matching of 800,000 locations and up to 10,000 complex patterns in much less than one second. For handling more locations or more patterns, measures can be taken to reduce the number of GPS position updates by configuring the Android Location Provider appropriately or by adding intermediate GPS smoothing filters. For example, for a frequency of 5 seconds per position update request, our application can efficiently scale up to at least 800,000 locations and over 300,000 complex patterns.

We consider a few improvements for future work. We have initial measurements of battery consumption which are promising, but need to be investigated in detail. The *VisitEventFilter* performs very fast (see Table 1). This results from using rectangular locations only, which allows to search for locations with the simple query shown in Section 2.2 efficiently. Specific applications may, however, require more complex location geometries. For the *PatternFilter*, run-times mainly depend of the loop executed over the set of patterns. Here, we can explore parallelism of the underlying multi-core hardware and we can apply optimizations from the area of *complex event processing*, see [5]. Furthermore, the handling of travel times (in addition to pattern time constraints) will be investigated by the repeated hierarchical composition of our *PatternFilter* and *TimeConstraint* filters. Lastly, we will evaluate the performance of our approach on real-world data collected from 70 users in the city of Cologne, Germany.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 255951 (LIFT Project).

REFERENCES

[1] J. Agrawal, Y. Diao, D. Gyllstrom, and N. Immerman, 'Efficient pattern matching over event streams', in *Proc. of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pp. 147–160, New York, NY, USA, (2008). ACM.
[2] A. Demers, J. Gehrke, B. Panda, M. Riedewald, V. Sharma, and W. White, 'Cayuga : A General Purpose Event Monitoring System', *Publish*, 412–422, (2007).
[3] Android developer website. www.developer.android.com, Last accessed: April 2012.
[4] C. du Mouza and P. Rigaux, 'Mobility patterns', *GeoInformatica*, **9**(4), 297–319, (2005).
[5] O. Etzion and P. Niblett, *Event Processing in Action*, Manning Publications Company, 2010.
[6] S.-C. Florescu, 'Efficient retrieval of mobility patterns on mobile devices', *RWTH Aachen, Germany*, (August 2012). To be submitted.
[7] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi, 'Trajectory pattern mining', in *Proc. of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pp. 330–339, New York, NY, USA, (2007). ACM.
[8] M. Hoffmann, 'A simulation environment for distributed stream analysis', *Master Thesis, Univ. of Appl. Sc. Bonn-Rhein-Sieg*, (2011).
[9] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, volume 2, Addison-Wesley, 1979.
[10] LIFT (Using Local Inference in Massively Distributed Systems). http://www.lift-eu.org.
[11] C. Körner, *Modeling Visit Potential of Geographic Locations Based on Mobility Data*, Phd thesis, University of Bonn, 2012.
[12] C. Körner, D. Hecker, M. May, and S. Wrobel, 'Visit potential: A common vocabulary for the analysis of entity-location interactions in mobility applications', in *Geospatial Thinking*, Lecture Notes in Geoinformation and Cartography, 79–95, Springer, (2010).
[13] Open Street Maps. www.osm.org, Last accessed: December 2011.
[14] A. Møller. dk.brics.automaton – finite-state automata and regular expressions for Java, 2010. `http://www.brics.dk/automaton/`.
[15] D. J. Patterson, L. Liao, K. Gajos, M. Collier, N. Livic, K. Olson, S. Wang, D. Fox, and H. Kautz, 'Opportunity knocks: a system to provide cognitive assistance with transportation services', in *Ubicomp*, pp. 433–450, (2004).
[16] M. A. Sakr and R. Hartmut Güting, 'Spatiotemporal pattern queries', *GeoInformatica*, **15**(3), 497–540, (2011).
[17] I. Sharfman, A. Schuster, and D. Keren, 'A geometric approach to monitoring threshold functions over distributed data streams', in *Proc. of the 2006 ACM SIGMOD international conference on Management of data*, SIGMOD '06, pp. 301–312, New York, NY, USA, (2006). ACM.
[18] E. Wu, 'High-performance complex event processing over streams', in *Proc. of the 2006 ACM SIGMOD international conference on Management of data*, SIGMOD '06, pp. 407–418, (2006).

Predicting Ramp Events with a Stream-based HMM framework

Carlos A. Ferreira[1] and **João Gama**[2] and **Vítor S. Costa**[3] and **Vladimiro Miranda**[4] and **Audun Botterud**[5]

Abstract. The motivation for this work is the study and prediction of wind ramp events occurring in a large-scale wind farm located in the US Midwest. In this paper we introduce the SHREA framework, a stream-based model that continuously learns a discrete HMM model from wind power and wind speed measurements. We use a supervised learning algorithm to learn HMM parameters from discretized data, where ramp events are HMM states and discretized wind speed data are HMM observations. The discretization of the historical data is obtained by running the SAX algorithm over the first order variations in the original signal. SHREA updates the HMM using the most recent historical data and includes a forgetting mechanism to model natural time dependence in wind patterns. To forecast ramp events we use recent wind speed forecasts and the Viterbi algorithm, that incrementally finds the most probable ramp event to occur.

We compare SHREA framework against Persistence baseline in predicting ramp events occurring in very short-time horizons.

1 Introduction

Ramping is one notable characteristic in a time series associated with a drastic change in value in a set of consecutive time steps. Two properties of a ramping event i.e. slope and phase error, are important from the point of view of the System Operator (SO), with important implications in the decisions associated with unit commitment or generation scheduling. Unit commitment decisions must prepare the generation schedule in order to smoothly accommodate forecasted drastic changes in wind power availability [2]. In this paper we present SHREA a novel stream-based framework that predicts ramping events in short term wind power forecasting.

The development of the SHREA framework is the answer to the three main issues available in ramp event forecasting. How can we describe and get insights on the wind power, and wind speed, time-dependent dynamic and use this description to predict short-time ahead ramp events? How can we combine real valued historical wind power and speed measurements and Numerical Weather Predictions (NWP), specially wind speed predictions, to output reliable real-time predictions? How can we continuously adapt SHREA to accommodate different natural weather regimes yet producing reliable predictions?

To answer these questions we designed a stream-based framework that continuously learns a discrete Hidden Markov Model (HMM) and uses it to generate predictions. To learn and update the HMM the SHREA framework uses a supervised strategy whereas the HMM parameters are estimated from historical data, the state transitions probabilities are estimated from wind power measurements and the emission probabilities, at each state, are estimated from wind speed observations. To estimate the state probability transitions, first, we combine a ramp filter, a derivative alike filter, and a user-defined threshold to translate the real-valued wind power time series into a labeled time-series, coding three different types of ramp events: ramp-up, no-ramp and ramp-down. Then, the transitions occurring in this labeled time series are used to estimate the transitions of the Markov process hidden in the HMM, i.e., to model the transitions between the three states associated with the three types of ramp events. To learn the HMM emission probabilities, first we combine a ramp filter and the SAX algorithm [9] to translate the wind speed measurements signal into a string. Next we use both the wind power labeled time series and the wind speed string to estimate the emission probabilities at each state. The estimative is obtained by counting the string symbols, coding wind speed variations, associated with a given state/ramp event.

When we analyze wind power historical data we observe both seasonal weather regimes and short-time ahead dependence of the recent past wind power/speed measurements. Thus, to accommodate these issues, in SHREA we included a strategy that forgets old weather regimes and continuously updates the HMM with the most recent measurements, both wind power measurements and wind speed measurements.

To generate ramp event predictions occurring in short-time ahead window we use the wind speed forecast, obtained from a major NWP provider, and the current HMM. First, we run a filter over the wind speed forecast signal to obtain a signal of wind speed variations. Next, we run the SAX algorithm to translate the resulting real-valued time series into a string. Then, we run the Viterbi algorithm [12] to obtain the most likely sequence of ramp events. We could use the Forward-Backward algorithm [12] usually used to estimate the posterior probability but we would be using long time ahead, thus unreliable, wind speed forecasts to predict current ramp events.

It is important to observe that wind speed measurements and forecasts, mainly short time horizon predictions, are approximately equally distributed over time. Moreover, the wind power output of each turbine is related to wind speed measurements.

In this work we run the SHREA framework to describe and predict very short-time ahead ramp events occurring in a large-scale wind farm located in the US Midwest. We present a comparison against the Persistence model that is known to be hard to beat in short-time forecasts [10].

Despite the difficulty of the ramp forecasting problem, in this work we make the following contributions: Develop a stream-based framework that predicts ramp events and generates both descriptive and

[1] LIAAD-INESC TEC and ISEP - Polytechnic Institute of Porto, Portugal
[2] LIAAD-INESC TEC and FEP - University of Porto, Portugal
[3] CRACS-INESC TEC and FC - University of Porto, Portugal
[4] INESC TEC and FE - University of Porto, Portugal
[5] Argonne National Laboratory, Argonne, IL, USA

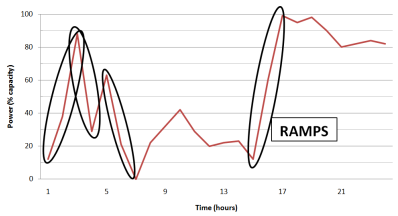


Figure 1: Illustration of ramp events, defined as a change of at least 50% in power in an interval of 4 hours

cost-effective models; Introduce a forgetting mechanism so that we can learn a HMM using only the most recent weather regimes; Use wind speed forecasts as observations of a discrete HMM to predict short-time ahead ramp events.

In the next Section we introduce the ramp event forecast problem. In Section 3 we present a detailed description of our framework. In Section 4 we present and discuss the obtained results. Last, we present some conclusions and present future research directions.

2 Ramp Event Definition and Related Work

One of the main problems in ramp forecasting is how to define a ramp. In fact, there is no standard definition [7, 3, 8] and almost all existing literature report different definitions, depending, for instance, on the location or on the farm's size.

The authors in [5] and [11] define several relevant characteristics for ramp definition, characterization and identification: to define a ramp event, we have to determine values for its three key characteristics: direction, duration and magnitude (see Figure 1). With respect to direction there are two basic types of ramps: the upward ones (or ramp-ups), and the downward ones (or ramp-downs). The former, characterized by an increase of wind power, result from a rapid raise of wind speeds, which might (not necessarily) be due to low-pressure systems, low-level jets, thunderstorms, wind gusts, or other similar weather phenomena. Downward ramps are due to a decrease in wind power, which may occur because of a sudden depletion of the pressure gradient, or due to very high wind speeds, that lead wind turbines to reach cut-out limits (typically 22-25m/s) and shut down, in order to prevent the wind turbine from damage [4]. In order to consider a ramp event, the minimum duration is assumed to be 1 hour in [11], although in [7] these events lie in intervals of 5 to 60 minutes. The magnitude of a ramp is typically represented by the percentage of the wind farm's nominal power - nameplate.

In [7] the authors studied the sensitivity of two ramp definitions to each one of the two parameters introduced above: ramp amplitude ranging from 150 to 600MW and ramp duration values varying between 5 and 60 minutes. The definition that we present and use in this work is similar to the one described in [7]. It is more appropriate to use in real operations since it does not considers a time-ahead point to identify a ramp event.

Definition 1 *A ramp event is considered to occur at time point t, the end of an interval, if the magnitude of the increase or decrease in the power signal is greater than the threshold value, the P_{ref}:*

$$|P(t) - P(t - \Delta t)| > P_{ref}$$

The parameter Δt is related to the ramp duration and defines the size of the time interval considered to identify a ramp. In [11] some results are presented that relate this parameter to the type and magnitude of identified ramps. The P_{ref} parameter is usually defined according to the specific features of the wind farm site and, usually, is defined as a percentage of the nominal wind power capacity or a specified amount of megawatts.

A comprehensive analysis of ramp modeling and prediction may be found in [2].

Algorithm 1: SHREA: a stream-based ramp predictor

input : Three time series: $\mathbb{P}_T$, wind power measurements; $\mathbb{O}_T$, wind speed measurements; and $\mathbb{J}_T$, wind Speed forecasts; a, the forecast horizon; P_{ref}, threshold to identify ramp events; Δt, the ramp definition parameter; W, the PAA parameter that specifies the amount of signal aggregation; σ, a forgetting factor

output: A sequence of predictions $\mathbb{Q}^d_r \ldots \mathbb{Q}^d_{r+a}$ for each period/window $d = 1, \ldots$

$countTimePeriods \leftarrow 0$; $flag \leftarrow 0$; $A^{count} \leftarrow 0$; $B^{count} \leftarrow 0$;
for *each period/window d* **do**
  $countTimePeriods++$
1 **Preprocessing**
  $\mathbb{P}^d_s \leftarrow$ fitSpline($\mathbb{P}^d$), $\mathbb{O}^d_s \leftarrow$ fitSpline($\mathbb{O}^d$), $\mathbb{J}^d_s \leftarrow$ fitSpline($\mathbb{J}^d$)
  $\mathbb{P}^d_f \leftarrow$ rampDef($\mathbb{P}^d_s, \Delta t$), $\mathbb{O}^d_f \leftarrow$ rampDef($\mathbb{O}^d_s, \Delta t$), $\mathbb{J}^d_f \leftarrow$ rampDef($\mathbb{J}^d_s, \Delta t$)
  $L^d \leftarrow$ label($\mathbb{P}^d_f, P_{ref}$); // *Label Data*
  $\mathbb{O}^d_n \leftarrow$ znorm($\mathbb{O}^d_f$), $\mathbb{J}^d_n \leftarrow$ znorm($\mathbb{J}^d_f$)
  $\mathbb{O}^d_{str} \leftarrow$ SAX(PAA($\mathbb{O}^d_n$)), $\mathbb{OF}^d_{str} \leftarrow$ SAX(PAA($\mathbb{J}^d_n$))
2 **Learn Supervised HMM**
  $\pi \leftarrow$
  $(\delta(L^d(r) = rampDown), \delta(L^d(r) = noramp), \delta(L^d(r) = rampUp))$
  $\lambda^d(A, B, \pi) \leftarrow$ LearnHMM($O^d_{str}(1, \ldots, r), L^d(1, \ldots, r), A_{count}, B_{count}$)
3 **Predict Ramp Events using the learned HMM**
  $\mathbb{Q}^d_r \ldots \mathbb{Q}^d_{r+a} \leftarrow Viterbi(\lambda, \mathbb{OF}^d_{str}(r+1, \ldots, r+a))$
  $\lambda^d(A, B, \pi) \leftarrow$ updateHMM($O^d_{str}(r+1, \ldots), L^d(r+1, \ldots)$)
4 **Forgetting mechanism**
  if *(countTimePeriods==σ)* **then**
    $A^{aux}_{count} \leftarrow A_{count}$; $B^{aux}_{count} \leftarrow B_{count}$; $flag \leftarrow 1$
  if *(countTimePeriods mod $\sigma == 0$ & flag==1)* **then**
    $A_{count} \leftarrow A_{count} - A^{aux}_{count}$; $B_{count} \leftarrow B_{count} - B^{aux}_{count}$

3 Methodology developed to Forecast Ramps

In this section we present SHREA framework, a stream-based framework that uses a supervised learning strategy to obtain a HMM. SHREA continuously learns a discrete HMM on a fixed size non-overlapping moving window and, at each time period, uses the updated HMM to predict ramp events. We introduce a forgetting mechanism to forget old wind regimes and to accommodate weather global changes. The SHREA architecture has three main steps (see algorithm pseudo-code in Algorithm 1): preprocessing phase, where a ramp filter and the SAX algorithm are used to translate real valued signals into events/strings; learning phase, where a supervised strategy is used to learn a HMM; and prediction phase, where the Viterbi algorithm is used to forecast ramp events. In the following lines we describe each one of these phases.

3.1 Preprocessing In the preprocessing phase we translate the real-valued points occurring in a given time period d, i.e. occurring inside a non-overlapping fixed size window, into a discrete time-series suitable to be used at HMM learning and prediction time. First, we fit a spline to both the wind power and wind speed measurements time series obtaining, respectively, two new signals, $\mathbb{P}^d_s$ and $\mathbb{O}^d_s$. We run the same procedure over $\mathbb{J}$ time series, a wind speed forecast, and obtain $\mathbb{J}^d_s$. We fit splines to the original data to remove high frequencies that can be considered noisy data. Second, we run ramp definition one, presented above in Section 2, to filter the three smoothed signals and obtain three new signals: $\mathbb{P}^d_f, \mathbb{O}^d_f$ and $\mathbb{J}^d_f$. These signals are wind power and speed variations, derivative alike signals, suitable to identify ramp events. Third, we use a user-defined power variation threshold, the input parameter P_{ref} value, to translate the wind power signal $\mathbb{P}^d_f$ into a labeled time series $L^d(1, \ldots, r+a)$, where 1 is the first point of the time window, r is the forecast launch time and a is the time horizon. We map each wind power variation into one of

three labels/ramp events: ramp-up, ramp-down and no-ramp. These three labels will be the three states of our HMM and the transitions will be estimated using the points of the L^d time series.

At this point we already have the data needed to estimate the transitions of the Markov process hidden in the HMM process. Now we need to transform wind speed data into a format suitable to estimate emission probabilities of the discrete HMM that we are learning. We combine Piecewise Aggregate Approximation (PAA) and SAX algorithms [9] to translate the wind speed variations into symbolic time series, more precisely. Thus, we normalize the two wind speed signals and obtain $\mathbb{O}_n^d$ and $\mathbb{J}_n^d$ signals. $\mathbb{O}_n^d$ will be used to estimate the HMM emission probabilities and the $\mathbb{J}_n^d$ will be used as the ahead observations that will be used to predict ramp events. Next, we run the PAA algorithm in each one of these signals to reduce complexity and, again, obtain smoothed signals. The degree of signal compression is the W PAA parameter that is a user-defined parameter of SHREA. This parameter is related with time point aggregation. Next, we run the SAX algorithm to map each PAA signal into string symbols. This way we obtain two discrete signals $\mathbb{O}_{str}^d$ and $\mathbb{J}_{str}^d$. After the preprocessing phase we have two discrete time series, L^d and $\mathbb{O}_{str}^d$ that will be used to learn the HMM state transitions and emissions probabilities, respectively.

3.2 Learn a Discrete HMM Here we explain how do we learn the HMM in the time period d, and then how we update it in time.

In the HMM that we learn, compactly written $\lambda(A, B, \pi)$, the state transitions, the A parameter, are associated with wind power measurements and the emissions probabilities, the B parameter, are associated with wind speed measurements. In Figure 2 we show a HMM learned by SHREA at the end of the 2010 winter. To estimate these two parameters we use the ramp labels, $L^d(1, \ldots, r)$, and the wind speed mesurements signals, $\mathbb{O}_{str}^d(1, \ldots, r)$, and run the well-known and straightforward supervised learning algorithm described in [12]. To estimate the transition probabilities between states, the three-way matrix A, we count the transitions between symbols observed in $L^d(1, \ldots, r)$ and compute the marginals to estimate the probabilities. To estimate the emission probabilities for each state, the matrix B, we count, for each state, the observed frequency of each symbol and then use state marginals to compute the probabilities. This way, we obtain the maximum likelihood estimate of both the transitions and the emission probability matrices.

We now explain how to update the model in the time. We design our framework to improve over the time with the arriving of new data. At each time period d SHREA is fed with new data and the HMM parameters are updated to include the most recent historical data. At each time period d we update the HMM parameters by counting the state transitions and state emissions coded in the current vectors $O_{str}^d(1, \ldots, r)$ and $L^d(1, \ldots, r)$, obtaining the number of state transitions and emissions at each HMM state, the A_{count} and B_{count}. Then, we compute the marginal probabilities of each matrix and obtain the updated HMM, the model $\lambda^d(A^d, B^d, \pi^d)$ that will be used to predict ramp events. The learned HMM, λ^d, will be used to predict ramp events occurring between r and $r + a$. In the next time period (i.e. the next fixed sized time window) we will update the λ^d HMM, using this same strategy but including also the transitions and emissions of the time period d that were not used to estimate λ^d, i.e., we update A_{count} and B_{count} with the wind measurements of the time period d occurring after d's launch time and before $d+1$ period launch time, the r point. By using this strategy we continuously update the HMM to include both the most recent data and all old data. By using this strategy, and with the course of time, the HMM can become less sensitive to new weather regimes. Thus we introduce a forgetting strategy to update the HMM using only the most recent measurements and forgetting the old data. This strategy relies on a threshold that specifies the number of time periods to include in the HMM estimation. This forgetting parameter, σ, is a user-defined value that can be set by experienced wind power technicians. Considering that at time period d we have read σ time periods and that we backup the current counts into A_{count}^{aux} and B_{count}^{aux} temporary matrices. After reading 2σ time periods we will use the following forgetting mechanism: $A_{count}^{2\sigma} = A_{count}^{2\sigma} - A_{count}^{aux}$ and $B_{count}^{2\sigma} = B_{count}^{2\sigma} - B_{count}^{aux}$. Then, we reset A_{count}^{aux} and B_{count}^{aux} equal to the updated $A_{count}^{2\sigma}$ and $B_{count}^{2\sigma}$ matrices, respectively. Next, to predict ramp events occurring in the time periods following 2σ, we will update and use the HMM parameters obtained from the $A_{count}^{2\sigma}$ and $B_{count}^{2\sigma}$ to forecast ramp events. Every time we read a number of time periods that equals a multiple of σ we apply this forgetting mechanism using the updated auxiliary matrices.

3.3 Predict Ramp Events using the learned HMM In this step we use the HMM learned in time period d, the λ^d, and the string $\mathbb{J}_{str}^d$, obtained from wind speed forecasts, to predict ramp events for the time points ranging from r to $r + a$. Remember that r is the prediction launch time and a is the forecast horizon.

To obtain the ramp event predictions we run the Viterbi algorithm [12]. We feed this algorithm with $\mathbb{J}_{str}^d$ and λ^d and get the state predictions (the ramp events) $\mathbb{Q}_{r+1}^d, \ldots, \mathbb{Q}_{r+a}^d$ for the time points $r + 1, \ldots, r + a$ of time period d. Saying it in other way we obtain predictions for the points occurring in a non overlapping time window starting at r and with length equal to a. We will obtain the most likely sequence of states that best explains the observations, i.e., we will obtain a sequence of states $\mathbb{Q}_{r+1}^d, \ldots, \mathbb{Q}_{r+a}^d$ that maximizes the probability $P(\mathbb{Q}_{r+1}^d, \ldots, \mathbb{Q}_{r+a}^d | \mathbb{J}_{r+1}^d, \ldots, \mathbb{J}_{r+a}^d, \lambda^d)$.

Regarding the π parameter, we introduce a non classical approach to estimate this parameter. We defined this strategy after observing that it is almost impossible to beat a ramp event forecaster that predicts the ramp event occurring one step ahead to be the current observed ramp event. Thus, we set π to be a distribution having zero probability for all events except the event observed at launch time, the r time point. In the pseudo code we write $\pi \leftarrow (\delta(L^d(r) == rampDown), \delta(L^d(r) == noramp), \delta(L^d(r) == rampUp),)$ where δ is a Dirac delta function defined by $\delta(x) = 1$, if x is $TRUE$ and $\delta(x) = 0$, if x is $FALSE$.

4 Experimental Evaluation

In this section we describe the configurations, the metrics and the results that we obtain in our experimental evaluation.

Table 1: Misclassification Costs

Predicted \ Observed	down	no	up
down	0	10	80
no	20	0	10
up	100	30	0

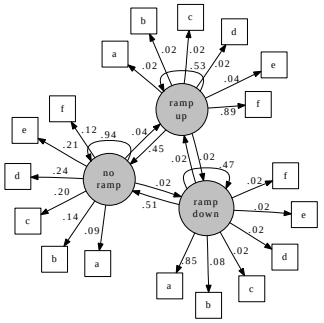


Figure 2: Winter HMM

Table 2: KSS, SS and Expected Cost Mean and standard deviation for the last 100 days of the evaluation period

Time ahead	Metric	SHREA									Persistence		
		Δt=1			Δt=2			Δt=3			Δt=1	Δt=2	Δt=3
		phE=0	phE=1	phE=2	phE=0	phE=1	phE=2	phE=0	phE=1	phE=2			
30 min	KSS	0.144(0.002)	–	–	0.332(0.001)	–	–	0.446(0.002)	–	–	0.144(0.002)	0.332(0.001)	0.446(0.002)
	SS	0(0)	–	–	0(0)	–	–	0(0)	–	–	–	–	–
	ECost	3.129(0.016)	–	–	4.176(0.027)	–	–	4.04(0.019)	–	–	3.129(0.02)	4.176(0.03)	4.041(0.02)
60 min	KSS	0.152(0.001)	0.202(0.002)	–	0.278(0.001)	0.314(0.204)	–	0.369(0.001)	0.417(0.001)	–	0.127(0.009)	0.203(0.001)	0.343(0.001)
	SS	0.028(0.001)	0.085(0.002)	–	0.094(0.00)	0.139(0.001)	–	0.038(0.001)	0.113(0.001)	–			
	ECost	2.312(0.18)	2.107(0.014)	–	3.860(0.39)	3.719(0.39)	–	4.374(0.61)	4.108(0.61)	–	8.731(0.99)	14.687(1.50)	16.104(1.63)
90 min	KSS	0.123(0.000)	0.185(0.001)	0.231(0.002)	0.193(0.001)	0.240(0.001)	0.296(0.002)	0.271(0.001)	0.316(0.001)	0.345(0.001)	0.101(0.001)	0.163(0.002)	0.258(0.002)
	SS	0.0244(0.002)	0.093(0.001)	0.145(0.001)	0.035(0.001)	0.091(0.002)	0.159(0.001)	0.018(0.001)	0.079(0.001)	0.118(0.001)	–	–	–
	ECost	2.089(0.013)	1.938(0.012)	1.807(0.010)	4.252(0.03)	4.028(0.025)	3.728(0.024)	5.165(0.025)	4.893(0.023)	4.677(0.025)	3.204 (0.030)	6.112(0.042)	6.783(0.050)

4.1 Experimental Configuration Our goal is to predict ramp events in a large-scale wind farm located in the US Midwest. To evaluate our system we collected historical data and, to make predictions, use wind speed power predictions (NWP) for the time period ranging between 3rd of June 2009 and 16th of February 2010. Each turbine in the wind farm has a Supervisory Control and Data Acquisition System (SCADA) that registers several parameters, including the wind power generated by each turbine and the measured wind speed at the turbine, the latter are 10 minute spaced point measurements. In this work we consider a subset of turbines and compute, for each time point, the subset mean wind power output and the subset mean wind speed, obtaining two time series of measurements. The wind speed power prediction for the wind farm location was obtained from a major provider. Every day we get a wind speed forecast with launch time at 6 am and having 24 hours horizon. The predictions are 10 minute spaced point forecasts. In this work we run SHREA to forecast ramp events occurring 30, 60 and 90 minutes ahead, the a parameter. We start by learning a HMM using five days of data and then use the learned, and updated, HMM to generate predictions for each fixed size non overlapping time window. Moreover, we split the day in four periods and run SHREA to learn four independent HMM models: dawn, period ranging between zero and six hours; morning, period ranging between six to twelve hours; afternoon, period ranging between twelve and eighteen hours; nigh, period ranging between eighteen and midnight. The last four models were only used to give some insight on the ramp dynamics and were not used to make predictions. We define a ramp event to be a change in wind power production higher than 20% of the nominal capacity, i.e., we set the P_{ref} threshold equal to 20% of the nominal capacity. Moreover, we run a set of experiments by setting Δt parameter equal to 1, 2 and 3 time points, i.e., equal to 30, 60 and 90 minutes. We run SHREA using thirty minute signal aggregation, thus each time point represents thirty minutes of data. In these experiments we also consider phase error corrections. Phase errors are errors in forecasting ramp timing [5]. We identify events that occur in a timestamp, t, not predicted at that time, but predicted instead to occur in one, or two, time periods immediately before or after t.

Furthermore, as SHREA is continuously updating the HMM, we set the forgetting parameter $\sigma = 30$, i.e., each time the system reads a new period of 30 days of data, the system forgets 30 days of old data. The amount of forgetting used in this work results from a careful study of the wind patterns.

For this configuration we compute and present the Hanssen & Kuippers Skill Score (KSS) and the Skill Score (SS) [1, 6]. Moreover, we compute the expected misclassification costs (EC) using the formula presented in [13]. The cost matrix presented in Table 1 defines the misclassification costs. We compare SHREA against a Persistence baseline algorithm. Despite its simplicity, the predictions of this model are the same as the last observation, this model is known to be hard to beat in short-time ahead predictions [10].

4.2 Results This work is twofold and here we present and analyze both the descriptive and predictive performance of the SHREA framework.

In Figure 2 we present an example of HMM generated by SHREA in February. This model was learned when running SHREA to predict 90 minutes ahead events and setting $\Delta t = 2$. This HMM has three states, each state is associated with one ramp type, and each state emits six symbols, each representing a discrete bin of the observed wind speed. The lower level of wind speed is associated with the a character and the higher level of wind speed is associated with the f character. The labels in the edges show the state emissions and the state transition probabilities.

The HMM models that we obtained in our experiments uncover interesting ramp behaviors. If we consider all the data used in these experiments, when we set $\Delta t = 1$ we found that there were detected 7% more ramp-up events than ramp-down events. When we set $\Delta t = 3$ we get the inverse behavior, we get 4% more ramp-downs than ramp-ups. This behavior is easily explained by the wind natural dynamics that causes steepest ramp-up events and smooth ramp-down events. If we analyze independently the four periods of the day we can say that we have a small number of ramp events, both ramp-ups and ramp-downs, in the afternoon. If we compute the mean number of ramps, for all Δt parameters we get approximately $30\%(15\%)$ more ramp-up(ramp-down) events at night than in the afternoon. Overall, we can say that we get more ramp events at night and, in second place, at the dawn period. Moreover, we can say that in the summer we get, both for ramp-up and ramp-down events, wind speed distributions with higher entropy, we get approximately 85% of the probability concentrated in two observed symbols. Different from this behavior, in the winter we have less entropy in the wind speed distribution associated with both types of ramp events. In the winter we have approximately 91% of the probability distribution concentrated in the one symbol. The emission probability distribution of the ramp-down state is concentrated in symbol a and the emission probability distribution in the ramp-down state is concentrated in symbol f. These two findings are consistent with our empirical visual analysis and other findings [4]: Large wind ramps tend to occur in the winter and usually there is a rapid wind speed increase followed by a more gradual wind speed decrease. These findings are also related with the average high temperature in the summer and with the stable temperatures registered during the afternoons. Considering the Δt parameter, we can say that the number of ramps, both ramp-ups and ramp-downs, increase with the Δt parameter. In general, we observe large ramps only when we compare time points that are 20 to 30 minutes apart.

As is illustrated in Figure 2 we identified a large portion of self-loops, especially ramp-up to ramp-up transitions in the winter nights. The percentage of self-loops range between 12%, when we run SHREA with $\Delta t = 1$, and 55% when we set $\Delta t = 3$. This self-loop transition shows that we have a high percentage of ramp events hav-

ing a magnitude of at least 40% of the nameplate, two times the P_{ref} threshold. Furthermore, in the winter we get a higher proportion of ramp-up to ramp-down and ramp-down to ramp-up transitions than in the summer. This is especially clear at the dawn and night periods. This phenomena can be related with the difference in the average temperatures registered in these time periods.

Before presenting the forecast performance, it must be said that the quality of ramp forecasting depends a great deal on the quality of meteorological forecasts. Moreover, as the HMMs represent probability distributions it is expected that SHREA will be biased to predict no-ramp events. Typically SHREA over predicts no-ramp events but makes less severe errors. This biased behavior of SHREA is an acceptable feature since it is better to forecast a no-ramp event when we observe a ramp-down(ramp-up) event than predicting a ramp-up(ramp-down) event. In real wind power operations (see Table 1) the cost of the later error is several times larger than the former errors.

In Table 2 we present the mean (inside brackets we present the associated standard deviation) KSS, SS and Expected Cost metrics that we obtained when running SHREA, and the reference model, to predict ramp events occurring in the last hundred days of the evaluation period.

Before presenting a detailed discussion of the obtained results, we must say that, if we consider the same Δt parameter, in all experiments we obtained better, or equal, results than the baseline algorithm, the Persistence algorithm. Moreover, we must say that when we generate predictions for the 30 minute horizon (one time point ahead, since we use 30 minutes aggregation) we get the same results as the Persistence model. This phenomena is related with the strategy that we used to define the HMM initial state distribution. Remember that we set the HMM π parameter equal to the last state observed.

As expected, the KSS results worsen with the increase of the time horizon. It is well known that the forecast reliability/fit worsens as the distance from the forecast launch time increases. Moreover we can say that we obtained better KSS values for the morning period than in the other three periods of the day. For lack of space we do not present a detailed description of the results that we obtain when we run SHREA to predict ramp events occurring in each one of the four periods of the day. This can be related with the wind speed forecasts launch time. The wind speed forecast that we use in this work is updated every day at 6 am.

The analysis of the Δt parameter shows that the mean KSS values increase with the increase in the Δt value. Again, this can be explained by the wind patterns, typically the wind speed increases smoothly during more than 30 minutes. In Table 2 we can see clearly that SHREA performance improves with the increase in Δt parameter. We observe the same behavior when inspecting the results that we obtained by running the Persistence algorithm. Concerning the SS, we can see that we obtain improvements over the Persistence forecast that ranges between 0% and 16%.

Concerning the phase error technique, we get important improvements for the two phase error parameter values considered in this study. The amount of improvement that we obtained by considering the phase error can be valuable in real time operations. The technicians can prepare the wind farm to deal with a nearby ramp event. In Table 2 we present the results without considering the phase error technique, $phE = 0$, and considering one time point (30 minutes), $phE = 1$, and two time points (60 minutes), $phE = 2$, phase errors corrections.

We also introduce a misclassification cost analysis framework that can be used to quantify the management decisions. We define a misclassification cost scenario (see Table 2) and show that SHREA produces valuable predictions. In this real scenario, SHREA generates significant lower operational costs and better operational performance than the baseline model.

5 Conclusions and Future Work

In this work we obtained some insights on the intricate mechanisms hidden in the ramp event dynamics and obtain valuable forecasts for very short-time horizons. For instance, we can now say that steepest and large wind ramps tend to occur more often in the winter. Moreover, typically there is a rapid wind speed increase followed by a more gradual wind speed decrease. Overall, with the obtained HMM models we both obtained insights on the wind ramp dynamics and generate accurate predictions that prove to be cost beneficial when compared against a Persistence forecast method.

The performance of SHREA is heavily dependent on the wind speed forecasts quality. Thus, in a near future we hope to get special purpose NWP suitable to detect ramp events and having more frequent daily updates. Moreover, we will study multi-variate HMM emissions to include other NWP parameters like wind direction and temperature.

Acknowledgments: This manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory ("Argonne"). Argonne, a U.S. Dep. of Energy Office of Science laboratory, is operated under Contract No. DE AC02-06CH11357. The authors also acknowledge EDP Renewables, North America, LLC. This work was is also funded by the ERDF - through the COMPETE programme and by National Funds through the FCT Project KDUS.

REFERENCES

[1] K.T. Bradford, R.L. Carpenter, and B. Shaw, 'Forecasting southern plains wind ramp events using the wrf model at 3-km', in *AMS Student Conference*, (2010).

[2] C. Ferreira, J. Gama, V. Miranda, and A. Botterud, 'A survey on wind power ramp forecasting', in *Report ANL/DIS 10-13*, Argonne National Laboratory, (2010).

[3] U. Focken and M. Lange, 'Wind power forecasting pilot project in alberta', Oldenburg, Germany: energy & meteo systems GmbH, (2008).

[4] J. Freedman, M. Markus, and R. Penc, 'Analysis of west texas wind plant ramp-up and ramp-down events', in *AWS Truewind, LLC*, Albany, NY, (2008).

[5] B. Greaves, J. Collins, J. Parkes, and A. Tindal, 'Temporal forecast uncertainty for ramp events', *Wind Engineering*, **33(11)**, 309–319, (2009).

[6] A.W. Hanssen and W.J.A. Kuipers, 'On the relationship between the frequency of rain and various meteorological parameters', *Mededelingen van de Verhandlungen*, **81**, (1965).

[7] C. Kamath, 'Understanding wind ramp events through analysis of historical data', in *IEEE PES Transmission and Distribution Conference and Expo*, New Orleans, LA, United States, (2010).

[8] A. Kusiak and H. Zheng, 'Prediction of wind farm power ramp rates: A data-mining approach', *J. of Solar Energy Engineering*, **131**, (2009).

[9] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, 'A symbolic representation of time series, with implications for streaming algorithms', in *8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, San Diego, CA, (2003).

[10] C. Monteiro, R. Bessa, V. Miranda, A. Botterud, J. Wang, and G. Conzelmann, 'Wind power forecasting: State-of-the-art 2009', in *Report ANL/DIS 10-1*, Argonne National Laboratory, (209).

[11] C. W. Potter, E. Grimit, and B. Nijssen, 'Potential benefits of a dedicated probabilistic rapid ramp event forecast tool', *IEEE*, (2009).

[12] L.R. Rabiner, 'A tutorial on hidden markov models and selected applications in speech recognition', *Proceedings of the IEEE*, **77(2)**, (1989).

[13] A. Srinivasan, 'Note on the location of optimal classifiers in n-dimensional roc space', in *Oxford University Technical Report PRG-TR-2-99*, Oxford, England, (1999).

People Identification Based on Sitting Patterns

Nguyen Gia [1] and **Takuya Takimoto** [2] and **Nguyen Doan Minh Giang** [3] and **Jin Nakazawa** [4] and **Kazunori Takashio** [5] and **Hideyuki Tokuda** [6]

Abstract. This paper proposes a people identification method based on the sitting patterns. This method uses weak evidences from pressure sensors, accelerometer sensors, and light sensors placed on a chair to recognize who is sitting on the chair without any psychological and physical burden on users. We discuss how we have implemented the system using softmax regression model, gradient descent algorithm and nearest neighbor search algorithm. Our experimental result shows that this method can be used in places which has private properties such as a home or small a office.

1 Introduction

Nowadays, there are several biometric people identification methods such as fingerprint based[1], iris based[2] or by the using of vein[3]. These biometric identifiers are strong and suitable for applications that request high accuracy such as security applications. However these identifiers annoy user with the requests for specific actions. For example, in the case of fingerprint, users have to properly touch a fingerprint scanner or in the case of retina, users have to look at retina scanner for a while, which might cause a psychological and physical burden on users. These methods also need delicate and expensive devices such as fingerprint scanner or retina scanner. In such situations as inside of a house or a small office with a small number of users, we do not need high accuracy as those available with strong identifiers. For example, in a small office, which employees come from some different country, an employee comes to the office, sits on a public chair and turns on a public computer. And then, a greeting sound of his/her country comes out of the speaker and that computer's language will be automatically change to his/her native language. Is it interesting? For the other scenario, an office has a meeting but the boss is in a business trip so he/she uses a robot for teleconference. The robot stands in the middle of meeting room and when the boss wants to talk with one of his/her employees, he/she only has to let the robot knows the employee's name instead of rotating robot by hand. Both of these scenarios can be realized with one of the above people identification methods, but using biometric identifiers for this scene is wasteful and unnecessary. Any mistake of people recognition in these scenes is not a big problem, users can easily overcome it by simple actions. So, it is acceptable to inference to the user who is sitting beforehand based on weaker evidences. Collecting weak evidences also can be implemented without any psychological and physical burden of users.

This paper proposes an easy deployment and inexpensive people identification method that uses weak evidences from pressure sensors, accelerometer sensors and light sensor placed on a chair. The reason of using pressure sensor is the difference of weight among users. Also, we think that the sitting patterns are different between users so we use accelerometer sensors to recognize the movement of the chair when user sit in it. The light sensor is used to measure the coverage of user in the chair. We have used softmax regression model, a supervised learning algorithm and gradient descent algorithm, an algorithm to solve optimization problem to inference who is sitting in the chair

Remainder of this paper is organized as follows. Section 2 describes the design and implementation of system. The softmax regression model, gradient descent algorithm, nearest neighbor search algorithm and how they are used are discussed in Section 3. Section 4 shows the result of our experiment while Section 5 is about related work. Conclusions and future work are described in Section 6.

2 Design and Implementation

2.1 Hardware

We use SunSPOT[8] for accelerometer sensor and light sensor. SunSPOT (Sun Small Programmable Object Technology) as shown in Figure 1(a) is a wireless sensor network mote which developed by Sun Microsystems. One SunSPOT device has three types of sensor including an accelerometer sensor, a light sensor and a temperature sensor. In this research, we only use one SunSPOT device to sense accelerometer and light data. These data can be sent to a computer for processing through a base station as shown in Figure 1(b).

(a) SunSPOT device (b) SunSPOT base station

Figure 1. SunSPOT

We also attach to the chair four pressure sensors. We use FSR406

[1] FPT Software Co. Ltd, Vietnam, email: gia@ht.sfc.keio.ac.jp
[2] Graduate School of Media and Governance, Keio University, Japan, email: tacky@ht.sfc.keio.ac.jp
[3] Faculty of Environment and Information Studies, Keio University, Japan, email: spider@ht.sfc.keio.ac.jp
[4] Graduate School of Media and Governance, Keio University, Japan, email: jin@ht.sfc.keio.ac.jp
[5] Faculty of Environment and Information Studies, Keio University, Japan, email: kaz@ht.sfc.keio.ac.jp
[6] Faculty of Environment and Information Studies, Keio University, Japan, email: hxt@ht.sfc.keio.ac.jp

for a pressure sensor and Figure 2 shows how it can be viewed in fact. We want to use as least as possible sensors to reduce the cost of the system and we think that four is a good number. It is enough for people recognition issue based on weak evidences.

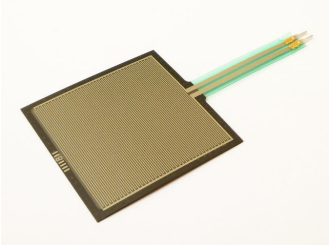


Figure 2. FSR406 Pressure Sensor

Figure 3 shows how sensors are placed in the chair. The light sensor is used to measure the coverage of user in the chair so it should be placed in the side of the chair.

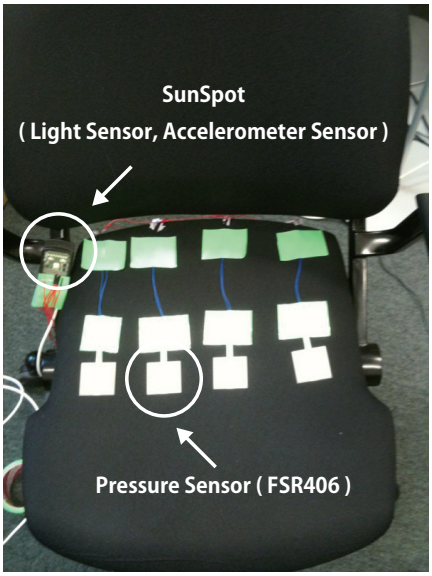


Figure 3. Sensors placed on a chair

2.2 Software

The software diagram of this system is shown in Figure 4 . The data receiver module receives data from sensors and forwards it to the data processing module. In here, the data is normalized to be used in the learning module. In the learning module, the system uses softmax regression model and gradient descent algorithm to inference to the user who are sitting beforehand, output result and get confirmation from user through the user interaction module.

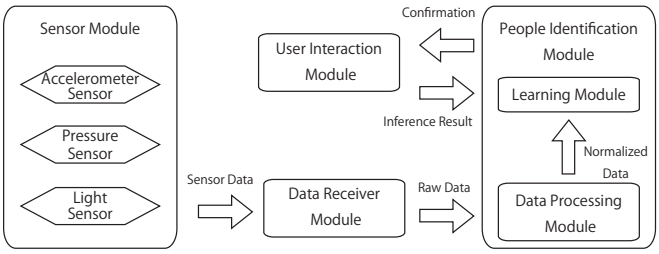


Figure 4. The Software Diagram

3 Approach

We consider the people identification problem with a small number of users as a classification problem. The system classifies the data from sensors into groups. A group represents a user so the number of groups equal to the number of users exists in the data training set. When a user sit down on the chair, a set of data will be created. The system will determine which group that this data set belongs to. By this way, the system can recognize the user who is sitting beforehand. We used Nearest Neighbor Search Algorithm to resolve this classification problem. The "weight" used in the nearest neighbor search algorithm are determined by Softmax Regression Model and Gradient Descent Algorithm. The softmax regression model is discussed in subsection below.

3.1 Softmax Regression Model

Softmax Regression Model[4] is a supervised learning algorithm used for multi-class classification. In Softmax Regression Model, we have a training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(n)}, y^{(n)})\}$ of n labeled examples, where the input features are m dimensional vector $x^{(i)} \in \Re^m$ and the label y can take on k different values, $y^{(i)} \in \{1, 2, \ldots, k\}$. Given a test input x, we want our hypothesis to estimate the probability that $P(y = j|x)$ for each value of $j = 1, 2, \ldots, k$. I.e., we want to estimate the probability of the class label taking on each of the k different possible values. Thus, our hypothesis will output a k dimensional vector (whose elements sum to 1) giving us our k estimated probabilities. Concretely, our hypothesis $h_\theta(x)$ takes the form:

$$h_\theta(x^{(i)}) = \begin{bmatrix} P(y^{(i)} = 1|x^{(i)}; \theta) \\ P(y^{(i)} = 2|x^{(i)}; \theta) \\ \vdots \\ P(y^{(i)} = k|x^{(i)}; \theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_2^T x^{(i)}} \\ \vdots \\ e^{\theta_k^T x^{(i)}} \end{bmatrix}$$

Here, m dimensional vectors $\theta_1, \theta_2, \ldots, \theta_k \in \Re^m$ are the parameters of this model and θ_i^T is the transpose vector of θ_i. Notice that the term $\frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}}$ normalizes the distribution, so that it sums to one. For convenience, we will also write θ to denote all the parameters of our mode.

$$\theta = \begin{bmatrix} -\theta_1^T - \\ -\theta_2^T - \\ \vdots \\ -\theta_k^T - \end{bmatrix}$$

If we know the h_θ function, we can determine the class label that given input vector x belong to. That is the class label that have maximum estimated probability. But the h_θ function is expressed by the θ parameters, so we have to find all the θ parameters.
The cost function of Softmax Regression Model is shown in the equation below.

$$J(\theta) = -\frac{1}{n}\left[\sum_{i=1}^{n}\sum_{j=1}^{k} Q(i,j)\log\frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right]$$

Here, Q(i, j) is defined as follow:

$$Q(i,j) = \begin{cases} 1 & \text{if } y^{(i)} = j \\ 0 & \text{otherwise} \end{cases}$$

In Softmax Regression, we also have:

$$P(y^{(i)} = j|x^{(i)};\theta) = \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}$$

Now, the training is mean finding all θ parameters that minimize the cost function. There are several methods to do it such as gradient descent algorithm or limited-memory BFGS algorithm. In this paper, we used gradient descent algorithm that is described below.

3.2 Gradient Descent Algorithm

The gradient descent algorithm[5] is a algorithm used to choosing θ to minimize the cost function $J(\theta)$. It starts with some "initial guess" for θ, and that repeatedly change θ to make $J(\theta)$ smaller, until hopefully converge to a value of θ that minimizes $J(\theta)$. The gradient descent algorithm repeatedly performs the update:

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta)$$

This update is simultaneously performed for all value of j. Here, α is called the learning rate.

To using gradient descent algorithm to minimize the cost function of softmax regression model, we need to compute the partial derivative of cost function $J(\theta)$. It is shown by the equation below.

$$\nabla_{\theta_j}J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left[x^{(i)}(Q(i,j) - P(y^{(i)} = j|x^{(i)};\theta))\right]$$

In particular, $\nabla_{\theta_j}J(\theta)$ is itself a m dimensional vector, so that its l-th element is $\frac{\partial}{\partial\theta_{jl}}J(\theta)$, the partial derivative of $J(\theta)$ with respect to the l-th element of θ_j. So we can use it to compute the update value of all parameters in softmax regression model.

But, take a look, if we take each of our parameter vectors θ_j, and subtract fixed vector ψ from it, so that every θ_j is now replaced with $\theta_j - \psi$ (for every $j = 1, 2, \ldots, k$), we have:

$$P(y^{(i)} = j|x^{(i)};\theta) = \frac{e^{(\theta_j-\psi)^T x^{(i)}}}{\sum_{l=1}^{k} e^{(\theta_l-\psi)^T x^{(i)}}}$$
$$= \frac{e^{\theta_j^T x^{(i)}} e^{-\psi^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}} e^{-\psi^T x^{(i)}}} = \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}$$

It show that if the cost function $J(\theta)$ is minimized by some setting of parameters $(\theta_1, \theta_2, \ldots, \theta_k)$, then it is also minimized by $(\theta_1 - \psi, \theta_2 - \psi, \ldots, \theta_k - \psi)$ for any value of ψ. Thus, the minimizer of $J(\theta)$ is not unique. To fix it, the cost function $J(\theta)$ is modified by adding a weight decay term $\frac{\lambda}{2}\sum_{i=1}^{k}\sum_{j=1}^{m}\theta_{ij}^2$ which penalizes large values of the parameters. Our cost function is now

$$J(\theta) = -\frac{1}{n}\left[\sum_{i=1}^{n}\sum_{j=1}^{k} Q(i,j)\log\frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right] + \frac{\lambda}{2}\sum_{i=1}^{k}\sum_{j=1}^{m}\theta_{ij}^2$$

With this weight decay term (for any $\lambda > 0$), the cost function $J(\theta)$ is now strictly convex, and is guaranteed to have a unique minimize solution. Also, the gradient descent algorithm is guaranteed to converge to the global minimum. To apply the gradient descent algorithm, we also need the partial derivative of this new definition of $J(\theta)$. The partial derivate is shown below.

$$\nabla_{\theta_j}J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left[x^{(i)}(Q(i,j) - P(y^{(i)} = j|x^{(i)};\theta))\right] + \lambda\theta_j$$

By using gradient descent algorithm with this equation to minimize the cost function $J(\theta)$, we will have a working implementation of softmax regression model.

3.3 Nearest Neighbor Search Algorithm

Nearest neighbor search (NNS)[6][7], also known as proximity search, similarity search or closest point search, is an optimization problem for finding closest points in metric spaces. The problem is: given a set S of points in a metric space M and a query point $q \in M$, find the closest point in S to q. In many cases, M is taken to be d-dimensional Euclidean space and distance is measured by Euclidean distance. The Euclidean distance between points p and q is the length of the line segment connecting them. In Cartesian coordinates, if $p = (p_1, p_2, \ldots, p_d)$ and $q = (q_1, q_2, \ldots, q_d)$ are two points in d-dimensional Euclidean space, then the distance from p to q, or from q to p is given by:

$$d_{pq} = d_{qp} = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \ldots + (q_d - p_d)^2}$$
$$= \sqrt{\sum_{i=1}^{d}(q_i - p_i)^2}$$

Now, if $CP(q)$ is the closest point to q in set S, we have:

$$CP(q) = \{p|p \in S; d_{pq} = \min_{i=1}^{n} d_{S_i q}\}$$

3.4 Calculation Process

We have discussed about softmax regression model, gradient descent algorithm and nearest neighbor search generally in subsections above. In this subsection, we describe how we use those algorithms in fact.

We use one accelerometer sensor, one light sensor and four pressure sensors placed on a chair for people recognition, so we have eight values of sensor data.

- Ax: The X-axis accelerometer value
- Ay: The Y-axis accelerometer value
- Az: The Z-axis accelerometer value
- Light: The light sensor value value
- A1: The first pressure sensor value
- A2: The second pressure sensor value
- A3: The third pressure sensor value
- A4: The fourth pressure sensor value

When a user sitting down to the chair, an array of 15 records, each has 8 values

$$r = (Ax, Ay, Az, Light, A1, A2, A3, A4)$$

will be created. This array is called RA and it describes the information of one sitting time of a user. In our data training set, there are 10 RA for one user. So if the number of user is k, the number of RA in data training set is $n = 10k$.

$$RA = \{r_1, r_2, \ldots, r_{15}\}$$

We use nearest neighbor search with 8-dimensional Euclidean space for this classification problem. But the Euclidean distance function we use has a litter different to general function. Because these eight sensor data affect to result in different ways, we modify the Euclidean distance function like this:

$$d_{pq} = \sqrt{\sum_{i=1}^{8} \theta_i (q_i - p_i)^2}$$

The parameters $\theta_1, \theta_2, \ldots, \theta_8$ is the "weight" of each sensor data and we use softmax regression model and gradient descent algorithm to determine them. Our people identify process can be described as following:

When a user sit in the chair, a RA is created. We take the average of all records of this RA and the average of all records of all RA in data training set and use softmax regression model to compute the parameters used in nearest neighbor search algorithm.The gradient descent algorithm is implemented with learning rate $\alpha = 0.001$ and $\lambda = 0.001$ to minimize the cost function in softmax regression model. Finally, we use nearest neighbor algorithm with determined parameters to classify the new RA to one of k class labeled. The result is the user whom this class labeled stand for. There are always 10 RA for one user in our data training set, but the data training set is dynamic. When a user sit down to the chair, after the system receives the confirmation from the user, in data training set, the oldest RA of this user is replaced by the newest RA. By this way, the system can adapt with the change of user's sitting pattern.

4 Evaluation

We evaluated this system in two cases. In the first case, we evaluated with a group of five people and in the other case, we evaluated with a group of ten people. In both case, one person must sit in a chair twenty times, ten for training and ten for testing. Figure 5 shows the result of first case while Figure 6 shows the result of last one. In the case of group of five people, we achieved an accuracy as 90 percentage and 72 percentage in the case of ten people.

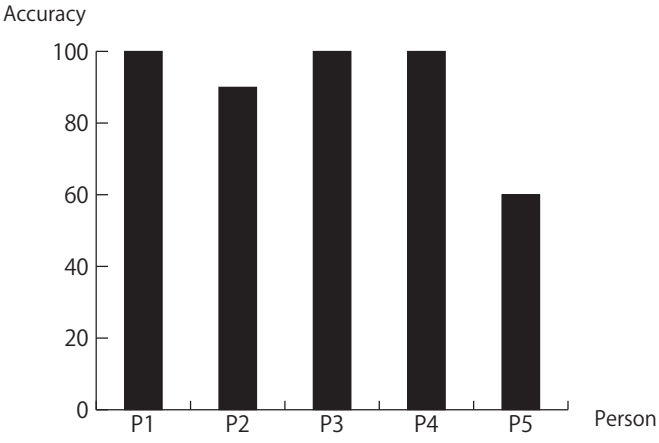


Figure 5. Accuracy for 5 people case

In the case of five people, there were three people who are identified with the accuracy as 100%. One people with the accuracy as 90%, it means that there was only one mistake.

With the achievement accuracy as 72% in the case of ten peoples, this method certainly can be used in a small office or inside a house with a small number of users.

5 Related Works

Masafumi Yamada et al.[9] have used 32 pressure sensors placed on a chair to people recognition. They have tested with a group of eight people who are required to sit 20 times, 19 for training and only one for testing. The result is shown in the Figure 7. Their system does not recognize user at the time user was sitting down but after few seconds, when the values of sensor get steady.The value of sensors were collected starting from a few seconds before the user starts sitting until the values of the sensor get steady after sitting. From the data they cut out two parts. One of them is the part during the user is sitting down, labeled as "Sitting part" . Another is the part after the sensor value gets steady, labeled as "Stable part". The classifier used is nearest neighbor method. Every testing data are classified to the nearest training data. Used features are classified into four groups to investigate how useful the information of pressure sensors is.

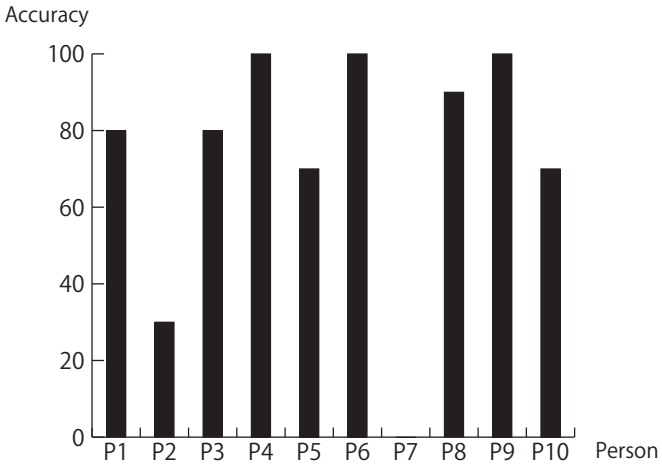


Figure 6. Accuracy for 10 people case

- Feature Set1 (FS1): 32 sensors Ω values (32)
- Feature Set2 (FS2): sum of 32 sensors Ω values (1)
- Feature Set3 (FS3): time difference of FS1 (32)
- Feature Set4 (FS4): normalized sensor values of FS3 (1)

As we can see, the achieved accuracy in steady part is about 90% but only 56% in the sitting part for average with a group of eight people.

No.	Features	Used part	Average recognition rate(%)
# 1	(FS1 + FS2)	Sitting	63
# 2	(FS1 + FS2)	Stable	90
# 3-(a)	(FS1)	Sitting	63
# 3-(b)	(FS4)	Sitting	52
# 3-(c)	(FS2)	Sitting	21
# 4-(a)	(FS1 + FS2+FS3)	Sitting	86
# 4-(b)	(FS3 + FS4)	Sitting	54

Figure 7. Average Recognition Rates by Yamada's Method

6 Conclusions and Future Works

We have proposed a people identification method based on sitting patterns of user. This method used weak evidences collected by accelerometer sensor, light sensor, and pressure sensor placed on a chair to inference who is sitting on it. We considered this problem as a classification problem and use nearest neighbor search algorithm with "weight" to resolve it. The "weight" used in nearest neighbor search algorithm is determined by softmax regression model while the cost function is minimized by gradient descent algorithm. We also presented the result of experiments which shown that this people identification has the accuracy enough to be used in places which have private properties such as inside of a house or a small office.

How to due with other evidences and what is the best way to place sensor to a chair are the things to be discussed in the future. Moreover, the evolution of performance increasing the number of people need to be studied. We also intend to implement a module to recognize the posture of user or the user's mood.

ACKNOWLEDGEMENTS

This research was partly supported by National Institute of Information and Communications Technology (NICT).

REFERENCES

[1] A.K. Jain, L. Hong, S. Pankanti and R. Bolle, 'An Identity Authentication System Using Fingerprints' , *Proc. IEEE*, vol. 85, no. 9, pp. 1,365-1,388, 1997.

[2] L. Ma , T. Tan , D. Zhang and Y. Wang, Personal identification based on iris texture analysis', *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 25, p.1519 , 2003.

[3] R. Hill, 'Retina Identification, Biometrics: Personal Identification' in *Networked Society*, A.K. Jain, R. Bolle, and S. Pankanti, eds., Kluwer Academic, pp. 123-141, 1999.

[4] C. Do and A. Ng. 'Transfer learning for text classification' . In *Proceedings of Neural Information Processing Systems (NIPS)*, 2005

[5] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, 'Learning to Rank using Gradient Descent'. In *ICML '05 Proceedings of the 22nd international conference on Machine learning*.

[6] Peter N.Yianilos, 'Data structures and algorithms for nearest neighbor search in general metric spaces'. *SODA '93 Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*

[7] Stefan Berchtold, Christian Bohm, Daniel A.Keim, Hans-Peter Kriegel, 'A cost model for nearest neighbor search in high-dimensional data space'. *PODS '97 Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*.

[8] http://www.sunspotworld.com, August, 2011

[9] M. Yamada, J. Toyama, M. Kudo, 'Person Recognition by Pressure Sensors'. *Knowledge-Based Intelligent Information and Engineering Systems, Lecture Notes in Computer Science*, Vol. 3684, Rajiv Khosla, Robert J. Howlett, Lakhmi C. Jain (Eds.), Springer, 2005, 703-708.

[10] Michael R. Bastian, Jacob H. Gunther, Todd K. Moon, 'A simplified natural gradient learning algorithm'. *Journal Advances in Artificial Neural Systems*, Vol. 2011, Article No. 3, Hinwadi Publishing Corp, Newyork, 2011

Semi-supervised learning: predicting activities in Android environment

Alexandre Lopes[1], **João Mendes-Moreira**[2,3], **João Gama**[3,4]

Abstract. Predicting activities from data gathered with sensors gained importance over the years with the objective of getting a better understanding of the human body. The purpose of this paper is to show that predicting activities on an Android phone is possible. We take into consideration different classifiers, their accuracy using different approaches (hierarchical and one step classification) and limitations of the mobile itself like battery and memory usage. A semi-supervised learning approach is taken in order to compare its results against supervised learning. The objective is to discover if the application can be adapted to the user providing a better solution for this problem. The activities predicted are the most usual in everyday life: walking, running, standing idle and sitting. An android prototype, embedding the software MOA, was developed to experimentally evaluate the ideas proposed here.

1 INTRODUCTION

Recognizing human activities with sensors next to the body has become more important over the years, aiming to create or improve systems in elder care support, health/fitness monitoring, and assisting those with cognitive disorders.

It is important to have systems that are practical for the user and that have the possibility to always be with them whilst not feeling strange or uncomfortable. Taking this into account we will attempt to use only one sensor instead of a, less practical but more accurate, system of distributed multi-sensors.

The new generation of smart phones has incorporated many powerful sensors, such as acceleration sensors (i.e. accelerometers), GPS sensors etc. They give the opportunity to create a system that can always be next to the user and work in real-time. In this work we will focus on the motion sensor of the cell phone, accelerometer, in order to predict the activity that the user is performing, as was attempted previously by Bao & Intille [1].

This problem will be treated as a classification problem using techniques of semi-supervised learning. This will be done in order to take advantage of existing examples (typically unlabeled) from the current user.

Knowledge discovery systems are constrained by three main limited resources: time, memory and sample size. In traditional applications of machine learning and statistics, sample size tends to be the dominant limitation. The problem of working with data streams is the arrival rate of the examples. When new examples arrive at a higher rate than they can be mined, the quantity of unused data grows without bounds as time progresses.

By building a new Smartphone application we attempt to solve problems consistent with previous undertakings, such as: accuracy, cost, performance among others. We explore matters like: (1) the impact of the app on the phone's battery lifetime; (2) how long should the interval to collect samples be in order to guarantee accurate classifications; (3) the time to create a model; and (4) the memory space needed.

All software used is open-source so the experiments can be continued and the application can be improved.

The aim of this work will be to create an application that adapt to each new user along time, learning his behavior and becoming more accurate.

2 RELATED WORK

Activity recognition is not new. Bao & Intille [1] created a system capable of recognizing twenty activities with bi-axial accelerometers positioned in five different locations of the user's person. This work led to an important discovery, which was possible to get accurate results predicting activities just using acceleration values gathered by a sensor placed on the thigh or dominant wrist. Despite this work uses twenty activities the most common activities used in other works [2,9,17] are walking, running, sitting, standing, up and downstairs.

Some research exists aiming to create a universal model that can be applied to any user. The idea is to use it in an Android application in order to measure the physical exercise of the user by predicting his activities [2]. This study uses three classification algorithms from WEKA (decision trees J48, logistic regression and multilayer neural networks) to induce models to predict user activities. Other studies, that also use the WEKA toolkit, implement common algorithms like Naïve Bayes, decision tables, K-nearest neighbors and SVM .

The common activities that research tries to predict are walking, running, sitting, standing, up and downstairs.

Gu et al. [3] tried to solve the activity recognition problem with techniques of semi-supervised learning using a large amount of unlabeled data, together with the labeled data, to build better classifiers. Because semi-supervised learning requires less human effort and gives higher accuracy, it is of great interest both in theory and in practice [4].

[1] Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias, s/n 4200-465 Porto – Portugal, email: alexolopes89@gmail.com
[2] Departamento de Engenharia Informática, Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias, s/n 4200-465 Porto – Portugal, email: jmoreira@fe.up.pt
[3] LIAAD – INESC TEC, Rua de Ceuta, 118, 6º; 4050-190 Porto – Portugal
[4] Faculdade de Economia, Universidade do Porto, Rua Dr. Roberto Frias, s/n 4200-464 Porto - Portugal, email: jgama@fep.up.pt

One of the most important aspects of the research, in this field, is the classifiers' accuracy and the difficulty of label new instances. Both Masud et al. [15] and Guan et al. [16] use ensemble methods to increase accuracy in partially labeled data (semi-supervised problems). A common thing in all the works is how they try to find the more accurate model, testing multiple classifiers with the same data. Authors like Kwapisz et al. [2] showed, when trying to solve this classification problem using decision trees, that the most important attribute to differentiate the activities is the acceleration they induce on the accelerometer. Domingos et al. [11] showed that decision trees like C4.5 could be outperformed by Hoeffding trees, and demonstrated their importance when dealing with streams and limited memory space. The biggest problem of decision trees is that they assume that all training examples can be stored simultaneously in main memory, and are thus severely limited in the number of examples they can learn from. Still, regarding the accuracy, the problem can be solved in a hierarchical way. Hierarchical classification splits the initial problem into simpler sub-problems. The objective is to have a tree in the end where tests are done in each node. The classes contained in different nodes from the same level of the tree should be independent [5] so there is no possible uncertainty when choosing the path. It is expected to obtain more accurate classifiers by training them in the split data. For activity recognition, this can be done by classifying firstly whether the activity is motion or motionless and, in a second step, classifying it in lying, sitting, standing (if it was classified as motionless in the first step) or walking, gentle motion and posture translation (if it was classified as motion in the first step). These experiments came to the conclusion that rule-based reasoning can improve the overall accuracy proving the lustiness of this approach [6].

The main drawbacks of using such approaches in a mobile phone are the limited battery and memory. Experiments were carried out to determine how long the data samples provided by the cell accelerometer should be in order to obtain accurate classification. Some experiments were made and it was discovered that at least they need to be captured for 6s and the interval between them can be up to 10s [7]. These results are used in our experiments as described in section 4. Another thing that has impact on the cell phone, more specifically in its memory, is how the data is saved. Not all the data needs to be saved. Using sliding windows only the most recent data needs to be available [8]. The features of the raw accelerometer data that can be retrieved are the mean, the standard deviation, the energy and the correlation [9]. The usefulness of these features has already been demonstrated [1]. It allows saving both data and memory.

In terms of mobile applications, DiaTrace [10] is a system developed to aid in sport activities. The authors do not explain how they carry out the classification. However they guarantee 95% of accuracy if the mobile phone is used in the trousers front pocket. This is an example of how the market demands this type of applications.

3 METHODS

The tests were made on Naïve Bayes and Hoeffding Trees [11]. These two algorithms were chosen because some studies showed that Naïve Bayes can predict equally as well as decision trees (Langley, Iba, & Thomas 1992; Kononenko 1990; Pazzani 1996) and Hoeffding trees can learn in a very small constant time what is of major importance since we are dealing with streams in a mobile context.

The Naive Bayes algorithm is a classification algorithm based on Bayes rule and can often outperform more sophisticated classification methods. The Naive Bayes algorithm is based on conditional probabilities; it calculates a probability by counting the frequency of values and combinations of values in the historical data. Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. It assumes that the attributes X1…Xn are all conditionally independent of one another, given the target variable Y. The value of this assumption is that it simplifies dramatically the representation of P(X|Y), and the problem of estimating it from the training data [12]. An important advantage of this algorithm is the possibility to calculate the required probabilities in one pass over the training set. Additionally, it is able to obtain good classification performance even when trained in a small amount of data. We can conclude that this classifier can be trained on an efficient way, gathering the probabilities of each attribute

Hoeffding trees [13] operate by collecting, for each leaf node, sufficient statistics of the training instances each leaf contains. Periodically, these leaves are checked to compare the relative merits of each candidate attribute for splitting. The Hoeffding bound, or similar metric, is used to determine when a candidate is better than the others. At this point the leaf is split on the best attribute, allowing the tree to grow. Typically, information gain is used to rank the merits of the split candidates, although other metrics could be used. In the case of discrete attributes, it is sufficient to collect counts of attribute labels relative to class labels to compute the information gain afforded by a split. There are some variations of Hoeffding Trees, based on VFDT (Very Fast Decision Tree learner) which is a high-performance data mining system [11]. It is effective in taking advantage of massive numbers of examples by using a very small constant time per example. Since we are working with a mobile phone the biggest advantage is that Hoeffding trees do not store any examples (or parts thereof) in main memory, requiring only a space proportional to the size of the tree and the associated sufficient statistics [11].

The novelty of our work is the creation of the Android application that records data from the accelerometer. It uses a semi-supervised learning algorithm to process data with a model previously learned. This model is used to label the unlabeled data in real-time. This new labeled data can be used to train future models that fit over the user. In the semi supervised approach we defined a threshold of 70% (value that we assumed to be a good percentage of certainty for a classification) which means that we add to the training file the instances classified with 70% or more of certainty. We can also define the number of these new instances that we need to gather in order to create a new model. The older instances are deleted in order to maintain the size of the file.

4 AN ANDROID PROTOTYPE

We have implemented an Android application that records data from the accelerometer. We use: (1) sequence-based sliding windows [8] in order to save memory; and (2) the method of duty cycles [7] in order to save battery.

In sequence-based sliding windows an amount of data is defined. The file will have only the amount of data that the sequence-based sliding window allows. If new data is added it

replaces the oldest data in order to keep the size stipulated by the window.

In the duty cycles, 6s of data is needed in order to get enough data so an accurate classification can be achieved. To proceed with the classification we have 10s before retrieving new data. It means that the data from the accelerometer does not need to be fetched all time, saving battery with less operations of the app running. To sum up, we record data for 6s. Then, an instance is created with an average of the collected values. Finally, it is classified on the next 10s. This cycle is repeated along time.

4.1 EXPERIMENTAL SETUP

Before testing the application some decisions had to be made in order to have a controlled environment so we knew which result we were expecting for each test done.

The placement of the mobile was an important issue. Without having the option of placing sensors in different parts of the human body we have chosen the trousers' front pocket [14] to conduct all experiments. So there is recorded data with the mobile in a vertical and horizontal position inside the pocket.

To create the models, data from two persons was used. This data contained the average of the values recorded by an accelerometer for several hours doing, only, activities of walking, running, standing idle and sitting, being the waking activity the one with more recorded instances. In the total approximately 27 thousand instances were used.

The unlabeled data (files from approximately 16 thousand to 30 thousand instances) was not used to create the model. It belongs to the two people that contributed with data to create the model. There is, also, data from a third person that was not used for learning the models. It was used to evaluate the semi-supervised learning approach.

We needed to choose between timestamp and sequence-based sliding windows depending whether the window length is defined according to a predefined interval or a predefined amount of data. We have chosen sequence-based sliding windows because we wanted to keep the number of instances controlled and with a time interval that is impossible because the number of data elements in the window may vary over time.

A threshold of 70% probability is used to proceed with semi-supervised learning as explained in section 3. This allows creating new models by appending to previous data the recent labeled data when classified with 70% of certainty, at least.

4.2 EXPERIMENTS AND RESULTS

Previously, labeled data from three different persons was recorded. The data contained four activities: walking, running, sitting and standing idle. Using MOA, two different approaches were taken.

Firstly, models were induced using both Naïve Bayes and Hoeffding Tree. The classifiers were tested on unlabeled data from one person (Table 1).

Table 1. Classifiers' accuracy.

	Naïve Bayes	Hoeffding Tree
Accuracy	92.00%	94.78%

Secondly, a hierarchical approach with two levels was also carried out using the same classifiers. The hierarchical approach has two classifications: (1) The first one classifies the data into Dynamic or Static whether the activities involve motion or not, respectively (Table 2); (2) Then, in the second classification, a model was built on each category so we could proceed to the classification on Walking or Running on the Dynamic category, and Sitting or Standing Idle on the Static one (Table 3).

Table 2. Classifiers' accuracy in the first level of the hierarchical approach.

Dynamic vs. Static	Naïve Bayes	Hoeffding Tree
Accuracy	82.11 %	99.85%

Table 3. Classifiers' accuracy in the second classification of the hierarchical approach.

	Naïve Bayes	Hoeffding Tree
Running, walking	76.25%	99.05%
Sitting, standing idle	99.83%	99.93%

To test the effectiveness of the classification, unlabeled data of a person, which was not used for training the classifier, was used. Here are the results for the walking activity – Table 4.

Table 4. Accuracy for the walking activity using as test set data from a person without data on the training sets

	One-step classification	Hierarchical 1st classif.	Hierarchical 2nd classif.
Naïve Bayes	86,37%	90,17%	84,27%
Hoeffding Tree	67,65%	94,04%	88,09%

These results only show that Hoeffding Tree is better than Naïve Bayes for the walking activity on a hierarchical approach. However, Naïve Bayes gives better results on the one-step approach (Table 4). Further tests were needed for the remaining activities. Additionally, a semi-supervised approach was also used, besides the supervised one described above, in order to evaluate the usefulness of using unlabeled data from the user that is being tested.

In order to adapt the model to the normal user of the cell phone a threshold of 70% was created, as described in section 3. This meant that data labeled with at least 70% of certainty would be recorded on the training file of the classifier, so a new model, more suitable to the user, could be generated. This approach is compared against the supervised approach (Figure 1). It is easier to check the better accuracy when using the semi-supervised approach.

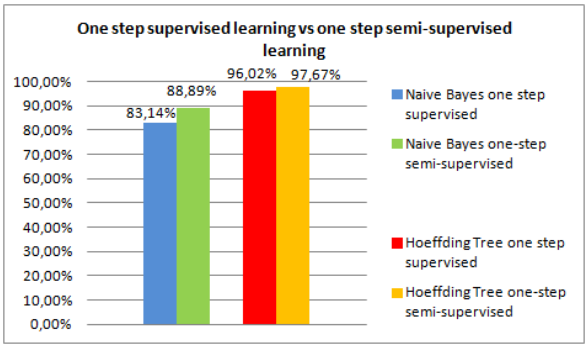


Figure 1 Accuracy of one step classification using both supervised and semi-supervised learning.

After doing the hierarchical classification (Figure 2 and 3) the labeled data was checked by visual inspection and it was easy to

observe that Hoeffding Tree tend to label data on the first classification as Dynamic (probably because the dataset is unbalanced and the Dynamic class is the majority one: there are about 15000 Dynamic instances and about 8000 Static ones). Naïve Bayes seems more balanced when labeling new data in the first classification of the hierarchical approach.

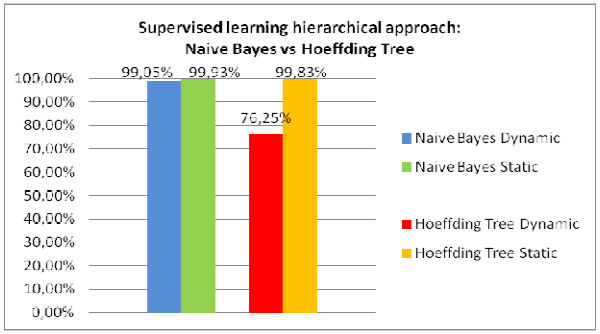


Figure 2. Classifiers' accuracy on final step of hierarchical classification with a supervised learning approach.

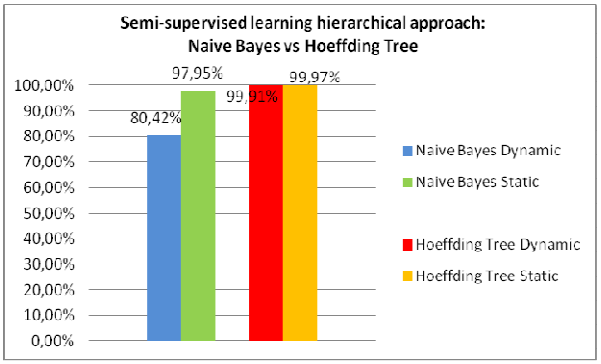


Figure 3. Classifiers' accuracy on final step of hierarchical classification with a semi-supervised learning approach.

At last we tested how using the two classifiers together would affect the classification (Figure 4).

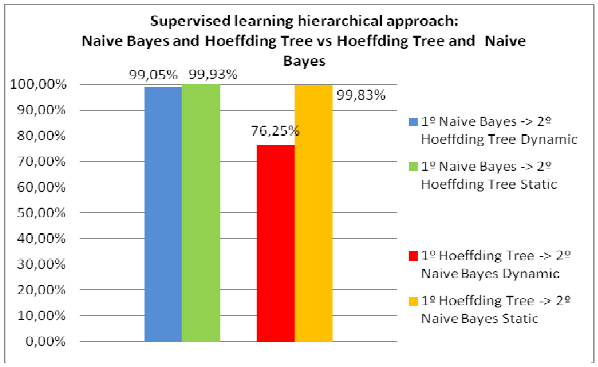


Figure 4. Classifiers' accuracy on final step of hierarchical classification with a supervised learning approach, using different classifiers for each step.

The balance characteristic of Naïve Bayes mentioned before can be verified in Figure 5, giving better results when used in the first classification. The tendency of Hoeffding Trees to classify, in the first step, the data as a Dynamic movement has influence on the second classification where Naïve Bayes has difficulties to label data because it gets lots of Static labeled data as Dynamic data from the first step. Overall better accuracy is achieved when using the Naïve Bayes classifier on the first classification (Dynamic or Static movement) and Hoeffding Tree on the second classification.

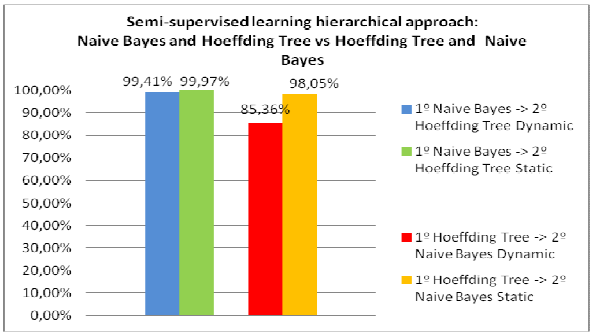


Figure 5. Classifiers' accuracy on final step of hierarchical classification with a semi-supervised learning approach, using different classifiers for each step.

The application had also concerns about both the battery and the memory usage. In order to test the battery usage, a stress situation where the app did both the hierarchical classification and the one step classification was created. In order to do it two models were created using the data of about 23.000 lines of labeled data, and doing the classification of 10 unlabeled instances. This experiment told us that the battery usage needs a maximum of 600.0mW for the CPU and between 500mW and 600mW for the LCD, which gives a total between 1100 and 1200mW on hierarchical classification. The one step classification only creates one model. The battery usage needs a maximum of 526mW for the CPU, the LCD needs the same power as the hierarchical approach, of course. Running the application five times, in a row, we got an energy usage of 120.8J for the CPU in hierarchical classification. However in one step classification we get a total of 110.3J.

Creating models and classifying about 10 instances took almost 60s which is a good time since we have only to classify 1 instance every 16s.

In terms of memory, the prototype is about 3Mb, and the files used for training the model having about 23000 lines are 1.466kB each. At most we will have the existence of three files for training (hierarchical approach). These files will grow because we defined a limit of 30000 instances for the training set (sequence-based window), which means that until we reach this limit none of the old training data will be erased and new data is added. When we reach the 30000 instances the sequence-based window will keep the size of the file. Whenever new labeled data from the user arrives (using the aforementioned 70% threshold) it will substitute the oldest data in order to have a semi-supervised learning approach.

The accuracy is not the only indicator of the classifiers' performance. Precision and recall are also important. The technique with higher accuracy might not be the one with the best balance between precision and recall. In our experiments we noticed that Hoeffding Trees have a better balance between precision and recall than Naïve Bayes.

5 CONCLUSIONS AND FUTURE WORK

The encouraging results of the experiments lead us to affirm that a step forward has been taken in the study of activity classification.

The most difficult activities to distinguish are walking and running because it is not clear where to draw the line between these two activities.

To achieve good results the techniques do not need to be too complex, like it was shown using Naïve Bayes. A fair conclusion after analyzing the figures is that hierarchical approach gives better results with Naïve Bayes doing the first classification and Hoeffding Tree dealing with the final one. With less complex techniques less power of the mobile is needed, leading to a minor impact on the classification performance. So, if Naïve Bayes does not decrease the accuracy it is better to use it in order to save memory and battery.

The battery usage confirms that the app can be used non-stop. It would be thrilling and of greater convenience to create a way that could swap classification techniques when the battery was low so it could be saved and the application did not have to stop. Changing from hierarchical classification to one step classification would have a maximum impact of 2% on the accuracy using Hoeffding tree as classifier.

The model used only has to be created when the application starts working. It is used for classifying until the app is shut down. It has only to classify one instance every 16s which is enough to do it, so the duty cycles work perfectly.

Regarding the memory usage a limit on the training files can be created, when this limit is reached the older data can be erased and new data added. This allows the adaptation of the application to new users as long as the application is being used by these new users.

The application can be improved by making possible to wear the mobile on other location, testing other classifiers or changing the way the data is processed.

New tests can be made using data from people with mobility constraints. Improving the app so it can adapt to this kind of people can be important if an accurate prediction can be made. Studies of patients with diseases that tend to degrade the ability to move can be accomplished to prevent, for example, falls or just to study how the movements change. This prevention can also be applied to elder people.

With this knowledge, people who practice sport can also benefit. For example, understanding how their body posture can be corrected in order to achieve better results.

This is just the beginning of an application that can be expanded in order to provide a better intimate experience between users and mobile phones.

ACKNOWLEDGEMENTS

This work is funded by the ERDF through the Programme COMPETE and by the Portuguese Government through FCT - Foundation for Science and Technology, project KDUS ref. PTDC/EIA-EIA/098355/2008.

REFERENCES

[1] L. Bao & S. S. Intille (2004). “Activity Recognition from User-Annotated Acceleration Data”, LNCS 3001, Springer, pp. 1-17.

[2] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, “Activity Recognition using Cell Phone Accelerometers,” *SIGKDD Explorations*, vol. 12, no. 2, pp. 74-82, 2010.

[3] Tao Gu, Zhanqing Wu, Xianping Tao, Hung Keng Pung, and Jian Lu. epSICAR: An Emerging Patterns based Approach to Sequential, Interleaved and Concurrent Activity Recognition. In Proc. of the 7th Annual IEEE International Conference on Pervasive Computing and Communications (Percom '09), Galveston, Texas, March 9–13, 2009.

[4] X. Zhu, Semi-Supervised Learning Literature Survey, Tech. report, Computer Sciences, University of Wisconsin-Madison, USA, 2005.

[5] A. C. F. Coster, “Classification of basic daily movements using a triaxial accelerometer,” *Medical & Biological Engineering*, pp. 679-687, 2000.

[6] M. Schneider, M. Velten, and J. Haupert, “The ObjectRules Framework - Providing Ad Hoc Context-Dependent Assistance in Dynamic Environments,” *2010 Sixth International Conference on Intelligent Environments*, pp. 122-127, Jul. 2010.

[7] N. S. Y. Wang, J. Lin, M. Annavaram, Q. A. Jacobson, J. Hong, B. Krishnamachari, “A Framework of Energy Efficient Mobile Sensing for automatic user state recognition”, pp. 179-191, 2009.

[8] B. Babcock and M. Datar, “Sampling from a moving window over streaming data,” *of the thirteenth annual ACM-SIAM*, 2002.

[9] N. Ravi, N. Dandekar, and P. Mysore, “Activity recognition from accelerometer data,” *Proceedings of the National*, pp. 1541-1546, 2005.

[10] G. Bieber, J. Voskamp, and B. Urban, “Activity Recognition for Everyday Life on Mobile Phones,” *Universal Access in HCI, Part II, HCII 2009, LNCS 5615*, pp. 289-296, 2009.

[11] P. Domingos & G. Hulten (2000). Mining high-speed data streams. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD’00*, 71-80. New York, New York, USA: ACM Press. doi:10.1145/347090.347107.

[12] T. Mitchell (1997). Machine Learning, McGraw Hill.

[13] Holmes, K. Richard, B. Pfahringer (2005). Tie-breaking in Hoeffding trees. In proceedings of the Second International Workshop on Knowledge Discovery from Data Streams, Porto, Portugal, 2005.

[14] S. Sprager and D. Zazula, “A cumulant-based method for gait identification using accelerometer data with principal component analysis and support vector machine,” *WSEAS Transactions on Signal Processing*, vol. 5, no. 11, pp. 369–378, 2009.

[15] M.M. Masud, J. Gao, L. Khan, J. Han and B. Thuraisingham, 2008. “A practical approach to classify evolving data streams: Training with limited amount of labeled data”. Proceedings of the 8th International Conference on Data Mining, December 15-19, 2008, Pisa, Italy, pp: 929-934.

[16] D. Guan , W. Yuan , Y. Lee , A. Gavrilov , S. Lee, “Activity Recognition Based on Semi-supervised Learning”, Proceedings of the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, p.469-475, August 21-24, 2007.

[17] T. Brezmes, J. Gorricho, J. Cotrina, “ Activity Recognition from accelerometer Data on a Mobile Phone”, Lecture Notes in Computer Science, 2009, Volume 5518, Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living, Pages 796-799

Applying Neural Networks for Concept Drift Detection in Financial Markets

Bruno Silva [1] and **Nuno Marques** [2] and **Gisele Panosso** [3]

Abstract. Traditional stock market analysis is based on the assumption of a stationary market behavior. The recent financial crisis was an example of the inappropriateness of such assumption, namely by detecting the presence of much higher variations than what would normally be expected by traditional models. Data stream methods present an alternative for modeling the vast amounts of data arriving each day to a financial analyst. This paper discusses the use of a framework based on an artificial neural network that continuously monitors itself and allows the implementation on a multivariate financial non-stationary model of market behavior. An initial study is performed over ten years of the Dow Jones Industrial Average index (DJI), and shows empirical evidence of concept drift in the multivariate financial statistics used to describe the index data stream.

1 INTRODUCTION

Data streams are generated naturally within several domains. Network monitoring, web mining, telecommunications data management, stock-market analysis and sensor data processing are applications that have vast amounts of data arriving continuously. In such applications, the process may not be strictly stationary, i.e., the target concept may change over time. *Concept drift* means that the concept about which data is being collected may shift from time to time, each time after some minimum permanence [6].

In this paper we address the detection and analysis of concept drift in financial markets by employing a methodology based on Artificial Neural Networks (ANN). ANN are a set of biologically inspired algorithms and well-established data mining methods popular for technical market analysis and price predictions. We are currently undergoing a wider research on using ANN in Ubiquitous Data Mining. This work, in essence, is a real-world application of a mechanism to detect concept drift while processing data streams. The motivation for this approach in the financial field can be easily explained. Mathematical finance has made wide use of normal distributions in stock market analysis to maximize return rates, i.e., they assume stationary distributions, which are easier to understand and work well most of the times. However, this traditional approach neglects big heavy-tails, i.e.,huge asset losses, in the distributions and their potential risk evaluation [11, 12]. This is where the detection of drifting from this normal behavior is of critical importance to reduce investment risk in the presence of non-normal distribution of market events.

[1] DSI/ESTSetúbal, Instituto Politécnico de Setúbal, Portugal, email: bruno.silva@estsetubal.ips.pt
[2] CITI and Departamento de Informática, FCT, Universidade Nova de Lisboa, Portugal, email: nmm@fct.unl.pt
[3] ISEGI, Instituto Superior de Estatística e Gestão de Informação, Universidade Nova de Lisboa, Portugal, email: m2010147@isegi.unl.pt

The main contributions of this work are: (i) a drift detection method based on the output of *Adaptive Resonance Theory* (ART) networks [7] which produce *aggregations* (or *data synopsis* in some literature) of d-dimensional data streams. These fast aggregations compress a, possibly, high-rate stream while maintaining the intrinsic relations within the data. A fixed sequence of consecutive aggregations is then analyzed to infer concept drift in the underlying distribution – Section 2 ; (ii) an application of the previous scheme to the stock market, namely to the Dow Jones Industrial index (DJI), using a stream of with a chosen set of statistical and technical indicators. The detection of concept drift is performed over an incoming stream of these observations –Section 3.

These contributions adhere to the impositions of data stream models in [8], namely: the data points can only be accessed in the order in which they arrive; random access to data is not allowed; memory is assumed to be small relatively to the number of data points, thus only allowing a limited amount information to be stored. Therefore, all of the additional indicators are computed using sliding windows, thus only needing a small subset of data kept in memory. This is also true for the number of aggregations needed to compute the concept drift.

At the end of the paper, Section 4, discussion of the results are made together with final conclusions.

2 METHODOLOGY

The presented methodology for drift detection comprises two modules. The first module uses an ART network that receives the incoming stream and produces aggregations, or data synopsis, compressing the data and retaining the intrinsic relationships within the distribution (Section 2.1). This module feeds a second module that takes a fixed set of these aggregations and through simple computations produces an output that can be used to detect concept drift.

2.1 Online Data Aggregation

One should point out that algorithms performing on data streams are expected to produce "only" approximated models [6], since the data cannot be revisited to refine the generated models. The aggregation module is responsible for the online summarization of the incoming stream and processes the stream in blocks of size S. For each S observations q representative prototypes of data are created, where $q \ll S$. This can be related to an incremental clustering process that is performed by an ART network. Each prototype is included in a tuple that stores other relevant information, such as the number of observations described by a particular prototype and the point in time that a particular prototype was last updated. These data structures were popularized in [1] and called *micro-clusters*.

Hence, we create q "weighted" prototypes of data stored in tuples $Q = \{\mathcal{M}_1, ..., \mathcal{M}_j, ..., \mathcal{M}_q\}$, each containing: a prototype of data P_j; the number of inputs patterns N_j assigned to that prototype and a *timestamp* T_j that contains the point in time that prototype was last accessed, hence $\mathcal{M}_j = \{P_j, N_j, T_j\}$. The prototype together with the number of inputs assigned to it (*weighting*) is important to preserve the input space density if one is interested in creating offline models of the distribution. The timestamp allows the creation of models from specific intervals in time.

ART [7] is a family of neural networks that develop stable recognition categories (clusters) by self-organization in response to arbitrary sequences of input patterns. Its fast commitment mechanism and capability of learning at moderate speed guarantees a high efficiency. The common algorithm used for clustering in any kind of ART network is closely related to the *k*-means algorithm. Both use single prototypes to internally represent and dynamically adapt clusters. The *k*-means algorithm clusters a given set of input patterns into *k* groups. The parameter *k* thus specifies the coarseness of the partition. In contrast, ART uses a minimum required similarity between patterns that are grouped within one cluster. The resulting number k of clusters then depends on the distances (in terms of the applied metric) between all input patterns, presented to the network during training. This similarity parameter is called *vigilance* ρ. *K*-means is a popular algorithm in clustering data streams, e.g., [4], but suffers from the problem that the initial *k* clusters have to be set either randomly or through other methods. This has a strong impact on the quality of the clustering process. On the other hand, ART networks do not suffer from this problem.

More formally, a data stream is a sequence of data items (observations) $x_1, ..., x_i, ..., x_n$ such that the items are read once in increasing order of the indexes i. If each observation contains a set of d-dimensional features, then a data stream is a sequence of $X_1^d, ..., X_i^d, ..., X_n^d$ vectors. We employ an ART2-A [3] network specially geared towards fast one-shot training, with an important modification given our goals: constrain the network on a maximum of q prototypes. It shares the basic processing of all ART networks, which is based on *competitive learning*. ART requires the same input pattern size for all patterns, i.e., the dimension d of the input space where the clusters regions shall be placed. Starting with an empty set of prototypes $P_1^d, ..., P_j^d, ..., P_q^d$ each input pattern X_i^d is compared to the j stored prototypes in a *search* stage, in a *winner-takes-all* fashion. If the degree of similarity between current input pattern and best fitting prototype W_J is at least as high as vigilance ρ, this prototype is chosen to represent the micro-cluster containing the input. Similarity between the input pattern i and a prototype j is given by Equation 1, where the distance is subtracted from one to get $S_{X_i,P_j} = 1$ if input and prototype are identical. The distance is normalized with the dimension d of an input vector. This keeps measurements of similarity independent of the number of features.

$$S_{X_i,P_j} = 1 - \sqrt{\frac{1}{d}\sum_{n=1}^{d}(X_i^n - P_j^n)^2} \tag{1}$$

The degree of similarity is limited to the range $[0, 1]$. If similarity between the input pattern and the best matching prototype does not fit into the vigilance interval $[\rho, 1]$, i.e., $S_{X_i,P_j} < \rho$, a new micro-cluster has to be created, where the current input is used as the prototype initialization. Otherwise, if one of the previously committed prototypes (micro-clusters) matches the input pattern well enough, it is adapted by shifting the prototype's values towards the values of the input by the update rule in Equation 2.

$$P_J^{(new)} = \eta \cdot X_i + (1 - \eta) \cdot P_J^{(old)} \tag{2}$$

The constant learning rate $\eta \in [0, 1]$ is chosen to prevent prototype P_J from moving too fast and therefore destabilizing the learning process. However, given our goals, i.e., to perform an adaptive vector quantization, we define η dynamically in such a way that the mean quantization error of inputs represented by a prototype is minimized. Equation 3 establishes the dynamic value of η, where N_J is the current number of assigned input patterns for prototype J. This way, it is expected that the prototypes converge to the mean of the assigned input patterns.

$$\eta = \frac{N_J}{N_J + 1} \tag{3}$$

This does not guarantee the convergence to local minimum, however, according to the adaptive vector quantization (AVQ) convergence theorem [2], AVQ can be viewed as a way to learn prototype vector patterns of real numbers; it can guarantee that average synaptic vectors converge to centroids exponentially quickly.

Another needed modification arises from the fact that ART networks, by design, form as much prototypes as needed based on the vigilance value. At the extremes, $\rho = 1$ causes each unique input to be encoded by a separate prototype, whereas $\rho = 0$ causes all inputs to be represented by a single prototype. Therefore, for decreasing values of ρ coarser prototypes are formed. However, to achieve exactly q prototypes solely on a manually tuned value of ρ is a very hard task, mainly due to the input space density, that can change over time, and is also different from application to application.

To overcome this, we make a modification to the ART2-A algorithm to impose a restriction on creating a maximum of q prototypes and dynamically adjusting the vigilance parameter. We start with $\rho = 1$ so that a new micro-cluster is assigned to each arriving input vector. After learning an input vector, a verification is made to check if $q = j + 1$, where j is the current number of stored micro-clusters. If this condition is met, then to keep only q we need to merge the *nearest pair* of micro-clusters. Let $T_{r,s} = min\{\|P_r - P_s\|^2 : r, s = 1, ..., q, r \neq s\}$ be the minimum Euclidean distance between prototypes stored in micro-clusters $\mathcal{M}_r$ and $\mathcal{M}_s$. We merge the two micro-clusters into one:

$$\mathcal{M}_{merge} = \{P_{merge}, N_r + N_s, max\{T_r, T_s\}\} \tag{4}$$

with the new prototype being a "weighted" average between the previous two:.

$$P_{merge} = \frac{N_r}{N_r + N_s}P_r + \frac{N_s}{N_r + N_s}P_s \tag{5}$$

With d-dimensional input vectors, Equation 1 defines a hypersphere around any stored prototype with radius $r = (1 - \rho) \cdot \sqrt{d}$. By solving this equation in respect to ρ, we update the vigilance parameter dynamically with Equation 6, hence $\rho^{(new)} < \rho^{(old)}$ and the radius, consequently, increases.

$$\rho^{(new)} = 1 - \frac{T_{r,s}}{\sqrt{d}} \tag{6}$$

Our experimental results show that this approach is effective in providing a summarization of the underlying distribution within the data streams. The inclusion of these results is out of the scope of this paper.

We must point out that the aggregation module produces more information that it is actually necessary for the concept drift detection,

namely the weighting of the prototypes and the timestamps. This module is an integrating part of a larger framework that also generates offline models of the incoming stream for specific points in time.

2.2 Detecting Concept Drift

Our method assumes that if the underlying distribution is stationary that the error-rate of the learning algorithm will decrease as the number of samples increases [5]. Hence, we compute the quantization error at each aggregation phase of the ART network and track the changes of these errors over time.

We use a queue $\mathcal{B}$ of b aggregation results, such that $\mathcal{B} = \{Q_l, Q_{l-1}, ..., Q_{l-b+1}\}$, where Q_l is the last aggregation obtained. For each Q_l that arrives, we compute the average Euclidean distance between each prototype P_i in Q_l and the closest one in $\mathcal{B}_{l-1} = \{Q_{l-1}, ..., Q_{l-b+1}\}$. Equation 7 formalizes this Average Quantization Error (AQE) computation for the l^{th} aggregation, where $\| \cdot \|^2$ is the Euclidean distance and q is the number of prototypes in Q_l by definition. This computes the error of the last aggregation in "quantifying" previous aggregations in a particular point in time.

$$AQE(l) = \frac{1}{q} \sum_{i=1}^{q} min(\quad \| P_i - P_j \|^2, \quad \forall P_j \in \mathcal{B}_{l-1} \quad) \quad (7)$$

By repeating this procedure over time, we obtain a series of errors that stabilizes and/or decreases when the underlying distribution is stationary and presents increases on this curve when the underlying distribution is changing, i.e., concept drift is occurring. This series of errors is the drift curve.

Larger values of b are used to detect *abrupt* changes in the underlying distribution, whereas to detect *gradual* concept drift a lower value should be adopted. We exemplify the automatic concept drift detection in this drift curve using a moving average in Section 3.2.

3 APPLICATION TO DOW JONES INDUSTRIAL

We present an application of the previous methodology to the stock market, namely to the Dow Jones Industrial index (DJI). Instead of using daily prices of several stocks that compose the DJI, our approach to this problem uses the DJI daily index values themselves and other computed statistical and technical indicators, which are explained in Section 3.1. We make extensive use of moving averages, as they reduce the short term volatility of time series and retain information from previous market events; another statistical indicator is the Hurst index [9], defined as a function to uncover changes in the direction of the trend of a set of values in time. We believe that these indicators, together with the index value, can provide a multi-variate insight to hidden and subtle changes in the normality of financial events and be used to assess the risk of investment at any point in time, thus lowering exposure to risk.

This application makes use of data gathered from the period comprised between the 1^{st} of January of 2001 to the 31^{st} of December of 2011, in a total of 2767 observations.

3.1 Variable Selection and Generated Data Stream

The data gathered was composed by a set of technical variables including different index values for one trading day like Open, Close, High and Low values. From these we chose the lowest daily price (PX LOW) because it provides better insight to the risk of a fall. Other available technical indicator was the trading Volume.

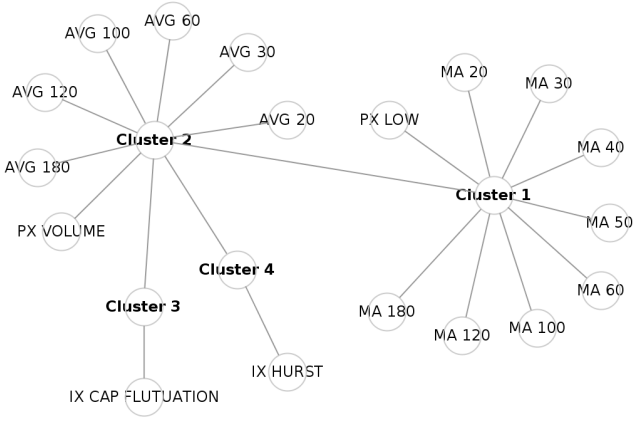


Figure 1. Hierarchical clustering of variables produced by VARCLUS.

In terms of statistical indicators, we initially considered a large number of them, like moving averages (MA) from 20 to 180 trading days, relative numbers, i.e., the DJI index value divided by moving averages (AVG), price fluctuation and Hurst index. However, it was important to reduce the number of variables because redundant variables can reduce the model efficiency. For this purpose we performed an analysis with the VARCLUS procedure (SAS/STAT).The VARCLUS procedure can be used as a variable-reduction method. The VARCLUS procedure divides a set of numeric variables into disjoint or hierarchical clusters through principal component analysis. All variables were treated as equally important. VARCLUS created an output was used by the TREE procedure to draw a tree diagram of hierarchical clusters (SAS/STAT®9.1 User's Guide p. 4797). The tree diagram is depicted in Figure 1. We can observe in the hierarchical clustering that the price variables and moving averages are correlated, so it was only chosen PX LOW of *Cluster 1*. In *Cluster 2* all variables were selected because, although they are correlated, they measure different characteristics. In the case of relative numbers different averages were selected because it is interesting to see the differences between the analysis of short, medium and long term. Finally in *Cluster 3* and *Cluster 4* just Hurst index and price fluctuation appeared, because they are not correlated with any other variable, so these variables were included in the final data set.

Hence, the complete set of features in the data stream is the following:

PX LOW: Minimum daily price;
PX VOLUME: Volume of daily business;
IX HURST: Hurst index computed for 30 days;
IX CAP FLUTUATION: PX LOW(t)/ PX LOW (t 1). This variable represents price fluctuation for one day interval;
AVG 20: PX LOW / 20-day moving average. This variable represents the relative number of current price divided by the 20-day Moving Average. This shows whether the current price is cheap, average value, expensive or really expensive. The same applies to the next indicators but within other time frames;

AVG 30: PX LOW / 30-day moving average;
AVG 60: PX LOW / 60-day moving average;
AVG 100: PX LOW / 100-day moving average;
AVG 120: PX LOW / 120-day moving average;
AVG 180: PX LOW / 180-day moving average;

The dataset is depicted in Figure 2, where the behavior of all variables can be seem. This data is our data stream. The stream comprises 10 features, e.g., a 10-dimensional stream.

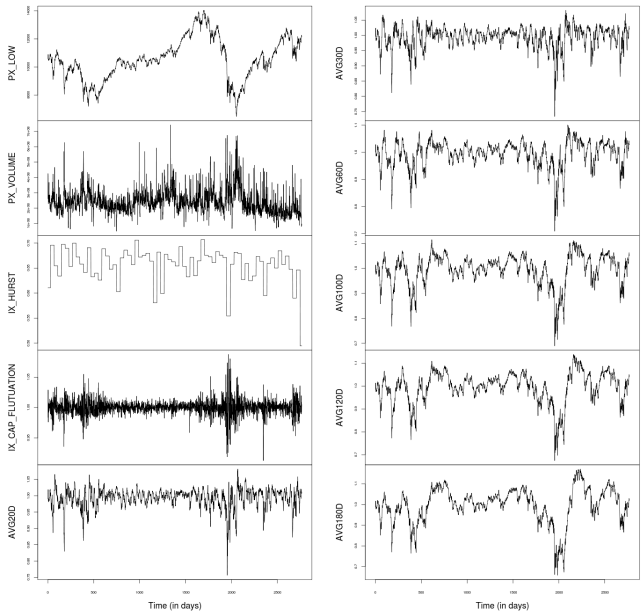


Figure 2. Variables of the data stream used in the presented application. It comprises technical and statistical indicators (description in text).

3.2 Concept Drift in the Dow Jones Industrial

The methodology presented in Section 2 was applied to the above data. It is converted into a data stream by taking the data input order as the order of the streaming. All features were previously normalized to the range $[0, 1]$ so they have equal importance in the Euclidean distances used to process them. The largest moving average indicator computed was over 180 days. Therefore, only after the 180^{th} observation can the stream be presented to the algorithm.

However, since we are dealing with financial time-series, it is important to retain the time dependency of the sequence of observations. Therefore, in this application, we use a sliding-window of 100 trading days, i.e., approximately a trimester of trading as input to each aggregation phase. Note that a year of trading has approximately 260 days. This means that the stream is processed in blocks of 100 observations that are kept in a queue. For each new observation that arrives the oldest in the queue is discarded and the new one added. The parameterization used was the following:

Block size: $S = 100$;
Number of micro-clusters: $q = 10$;
Concept drift buffer size: $b = 15$

The result of the procedure of Section 2.2 applied to the data stream is presented in Figure 3. Each point of the series corresponds to the error of the model for a particular trading day, thus providing possible indications of drifting. It can be seen an overall shape of a curve that indicates the drift over time. Since this drift is being computed for every trading day, the "noise" around the curve is considered normal since it is affected by the daily volatility of the index values.

To obtain a "clean" curve we apply a convolution filter along this drift series of the same size as b, i.e., 15 days. An alarm scheme is created through the generation of an empirical moving average of 60 days performed over the drift series. The cleaned drift curve and its moving average are depicted in Figure 4a).

We then compare the differences between the drift series and its moving average obtaining a line that oscillates around zero. We call this line the *drift trend*, shown in Figure 4b). Whenever the drift series has values lower than its moving average we are in a descending trend. This is reflected in the drift trend with values lower than zero. Whenever the moving average is crossed by the drift series it signals a shift in the trend and the drift trend crosses zero. This reasoning to detect trends is also very popular in financial technical analysis. In this context, the 60 trading days moving average reflects the intuitive notion of long-term "decreasing" or "increasing" trend of the drift.

All plots in Figure 4 are aligned in time for easy comparison. Figure 4c) shows the time series of PX LOW, i.e., the DJI index, that we compare to the detection of drift performed.

4 DISCUSSION AND CONCLUSIONS

Based on experiments we found that a tenth of prototypes relative to the number of observations are sufficient in most applications to represent them adequately, hence, $q = 10$. Usage of higher values of q did not improve the results with the additional problem of increased computational time. Additionally, since we are both interested in abrupt and gradual drift detection we used a moderate sized buffer of aggregations ($b = 15$) to compute the series of quantification errors. During our experiments we found that this value was appropriate for the established goals.

By inspecting Figure 4 and comparing the *drift trend* with the behavior of the DJI index we can make two important observations: (i)

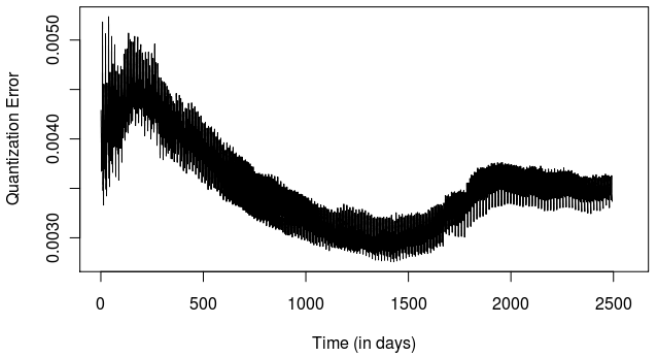


Figure 3. Concept drift series obtained through the methodology in Section 2.2 computed for each trading day.

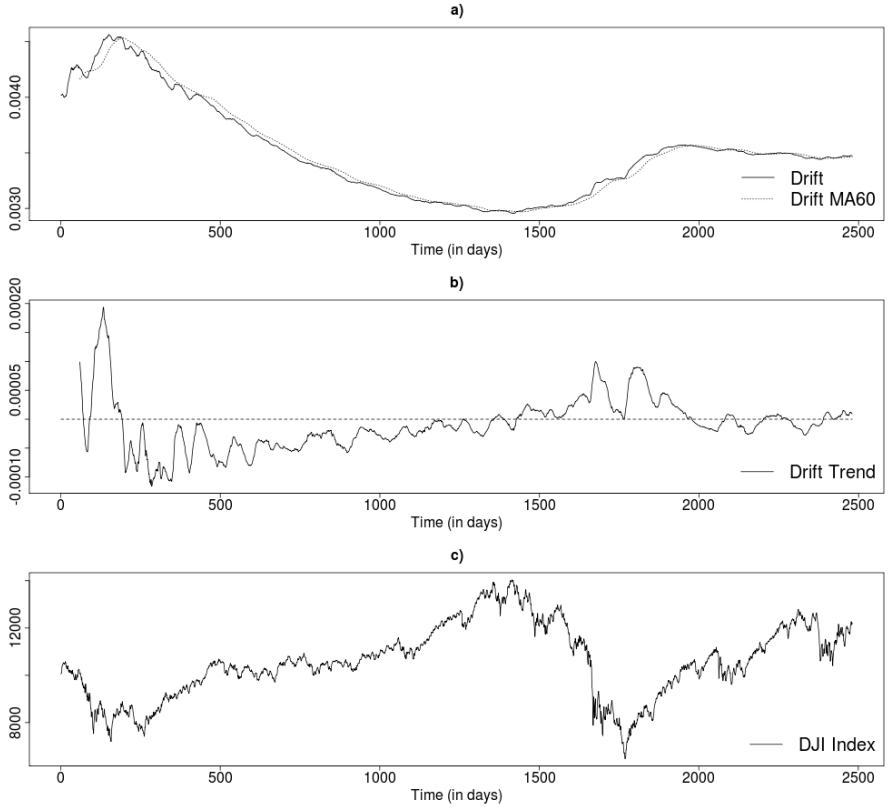


Figure 4. a) Cleaned drift curve and its moving average. b) The trend drift curve is used to automatically detect drifting. c) The DJI index time series (PX LOW variable).

the drift trend crossed zero before the market crash of 2008 (around day 1500). It appears that the concept that was being learned changed sometime before the crash occurred. (ii) it may be reasonable to assume that in periods of normality the long-term tendency of these indexes is upwards. One of such periods is after the recovery of the 2002 market crash, i.e., the dot-com bubble, until the other crash of 2008 (approximately between days 300 and 1300). During such period it is interesting to see that the drift trend was always below zero.

In the present work we have shown a methodology to detect concept drift in financial markets. We intend to apply this same methodology to intra-day trading as soon as it is possible, thus reinforcing the need to efficient processing of large volumes of data. The proposed methodology applied over a data stream comprised of carefully chosen technical and statistical indicators seems promising in detecting changes in markets events ahead of time that can reduce the exposure to risk.

The characterization of the drifts, i.e., trying to understand what is really changing in the markets through inspection of hidden changes in the indicators is reserved for future work. Work is under way in this subject and we are using Self-Organizing Maps [10] to produce different mappings of the variables for particular segments in time, namely ones where the market seems to exhibit a stable behavior and comparing with others where it does not. This segments are obtained by segmenting time with the concept drift detection. As another immediate future work we will apply this methodology to other indexes and perform the same study.

REFERENCES

[1] C.C. Aggarwal, J. Han, J. Wang, and P.S. Yu, 'A framework for clustering evolving data streams', in *Proceedings of the 29th International Conference on Very Large Databases*, volume 29, pp. 81–92. Morgan Kaufmann Publishers Inc., (2003).

[2] K. Bart, *Neural networks and fuzzy systems: A dynamical systems approach to machine intelligence*, Prentice-Hall of India, 1997.

[3] G.A. Carpenter, S. Grossberg, and D.B. Rosen, 'Art 2-a: An adaptive resonance algorithm for rapid category learning and recognition', *Neural networks*, **4**(4), 493–504, (1991).

[4] F. Farnstrom, J. Lewis, and C. Elkan, 'Scalability for clustering algorithms revisited', in *ACM SIGKDD Explorations Newsletter*, volume 2, pp. 51–57. ACM, (2000).

[5] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, 'Learning with drift detection', *Advances in Artificial Intelligence–SBIA 2004*, 66–112, (2004).

[6] Joao Gama, *Knowledge discovery from data streams*, Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, 2010.

[7] S. Grossberg, 'Adaptive pattern classification and universal recording: II. Feedback, expectation, olfaction, illusions', *Biological Cybernetics*, **23**, (1976).

[8] Monika R. Henzinger, Prabhakar Raghavan, and Sridhar Rajagopalan, 'External memory algorithms', chapter Computing on data streams, 107–118, American Mathematical Society, Boston, MA, USA, (1999).

[9] H.E. Hurst, RP Black, and YM Simaika, *Long-term storage: An experimental study*, Constable, 1965.

[10] T. Kohonen, 'Self-organized formation of topologically correct feature maps', *Biological cybernetics*, **43**(1), 59–69, (1982).

[11] B. Mandelbrot, R.L. Hudson, and E. Grunwald, 'The (mis) behaviour of markets', *The Mathematical Intelligencer*, **27**(3), 77–79, (2005).

[12] N.N. Taleb, 'Errors, robustness, and the fourth quadrant', *International Journal of Forecasting*, **25**(4), 744–759, (2009).

MLP Networks Applied to the Problem of Prediction of Runtime of SAP BW Queries

Tatiana Escovedo[1], Tarsila Tavares[2], Rubens Melo[3], Marley M.B.R. Vellasco[4]

Abstract. The SAP BW is a BI tool used daily by about 8000 employees of a big oil company in Brazil, running monthly about 150,000 queries to assist in the analysis inherent in their professional activities. A query is created to meet a need for specific business analysis and its response time is directly affected by the use of BW server by other users. The main problem today is that there is no way to estimate the execution time in advance, for the user to decide the best time to execute the query he needs for his work. This article proposes a solution to this problem by developing a prediction classification model for the performance of BW queries at certain times of the day using Multilayer Perceptron (MLP) neural network and the Weka tool [WEKA, 2011].

1 INTRODUCTION

The constant changes in the market have prompted the need for more timely and accurate integrated information from different departments in order to accelerate the process of decision making in the organizations. An analysis of historical integrated data can provide indicators of business growth or danger. This fact also prompted the emergence of data warehouse technology.

Data Warehousing is not a product but a strategy that recognizes the need to store data in separate information systems and consolidate them in order to support various professionals of a company in making decisions quickly and effectively. A data warehouse is a subject oriented, integrated, time variant and nonvolatile collection of data, which aims to support the process of decision making [Inmon, 1992].

To meet its needs for analytical information, the company uses the SAP BW since 2004. Through so-called BW queries, users can perform various analysis of business information. Data extracted from many sources is integrated and stored in the data warehouse implemented with the BW, and then accessed through queries which give useful information for the daily work of thousands of users. However, a significant problem is that many queries take considerable time to perform, because they work with a large volume of data and also because they dispute server time with other users. Currently, users have no way to estimate the runtime of their query in advance, in order to decide the best schedule for execution of the query. This article aims to investigate the application of neural networks in the problem of prediction of the performance of BW queries along the day. Section 2 presents a summary on Neural Networks, the tools Weka and SAP BW. Section 3, in turn, explains the problem in study and section 4 describes the proposed solution. The following section 5, will present the results and section 6 will evaluate them, pointing out some possible future work. Finally, section 7 concludes this work.

[1, 2, 3] Department of Electrical Engineering - Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil

[4] Department of Computer Science - Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil

e-mail: [1]tatiana@inf.puc-rio.br, [2]tarsilabello@gmail.com, [3]rubens@inf.puc-rio.br, [4]marley@ele.puc-rio.br

2 BASIC CONCEPTS

The purpose of this section is to present briefly the basic concepts related to neural networks and the tools Weka and SAP BW used in this work. These concepts will provide the theoretical background to the issues raised in this study.

2.1 Neural Networks

Artificial neural networks are used in the area of intelligent systems and computational intelligence [Jang et al, 1997] [ZADECH, 1992]. A neural network is a parallel and distributed system composed of simple processing units. Its goal is to store experimental knowledge and make it available for use. Artificial neural networks are similar to the human brain, because knowledge is acquired through a learning process and the strength of connections between neurons, or synaptic weights are used to store the acquired knowledge [Haykin, 1999]. The Multilayer Perceptron (MLP) is a type of neural network widely used for its ease of implementation and for being considered a universal approximator [Hornik et al, 1989]. MLP networks have powerful computing power due to the insertion of intermediate layers that enable the solution of non-linearly separable problems. Thus, MLP networks have at least three layers: the input layer, the intermediate or hidden layer and the output layer. A network with one hidden layer can implement any continuous function, with two intermediate layers, can approximate any mathematical function [CIBENKO, 1989].

2.2 Weka

Weka [WEKA, 2011] is a free software for data mining. Its implemented features and techniques are described in detail in [Witten & Frank, 2005], the implementers of the tool. One of the uses of the tool is the extraction of classifiers in databases. A

classifier (or classification model) is used to identify the class to which a specific observation in a database belongs from its characteristics (attributes). This tool is used in this work to classify BW queries into categories according to the scheduled time of execution.

2.3 SAP BW

SAP BW is part of the solution of SAP Business Intelligence software; whose main purpose is to store in a single location (data warehouse) separate from the transactional environment, data from different sources, facilitating the provision of analytical information for users in an integrated and uniform way. Because it is integrated with SAP ERP R/3 it does not need interface files, which provides highly reliable transfer and maintenance of the quality of the information. The tool encourages the generation of queries (queries) by the users, by means of user-friendly tools on the web and / or Excel [McDonald et al, 2006]. In this environment, information is stored in a structured manner to facilitate queries and analysis, thus supporting the decision making and management. The BW software is complementary to the R/3. While the R/3 is configured to optimally run transactions from day to day business (eg buy, sell, manufacturing) BW is optimally configured to allow analysis (eg how are my indicators going, how good are the sales of a product group to a customer, how is the stock of a particular product evolving, etc). The information can be extracted from SAP BW by creating queries using the Query Designer tool, which is also part of the suite SAP BW [PALEKAR et al, 2010]. This tool has a simple interface and can be easily used by the business users to build queries to support their analysis. Figure 2.1 shows the construction of a query with the tool query designer.

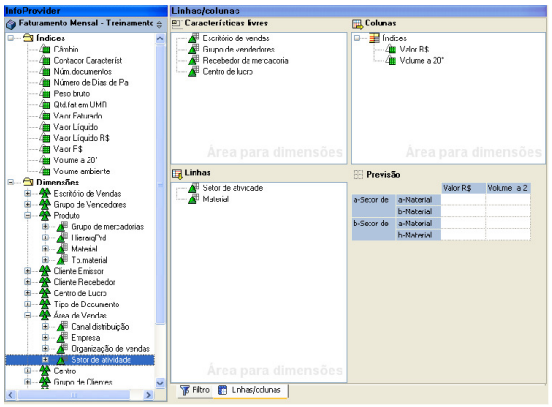


Figure 2.1. Creating a BW Query

After the creation of the queries either by business users or by IT staff of the company they are published and made available to the users so that they can extract the necessary data for their analyses. The execution of queries can be performed by the Business Explorer (BEx), a tool that is also part of the suite SAP BW. This tool is integrated with Microsoft Excel, the environment in which most business users are already familiar. Alternatively, the query may be executed through the web interface. Figure 2.2 illustrates the execution of a query using the web interface.

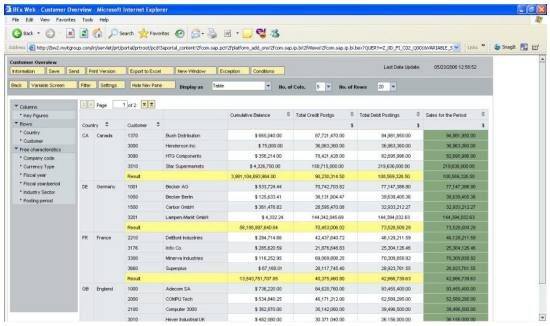


Figure 2.2. Executing a BW Query

This section presented briefly the basic concepts related to neural networks and the tools Weka and SAP BW, which are the basics necessary to follow the issues raised in this article. The next section presents the real problem addressed in this work.

3 PROBLEM DESCRITION

The SAP BW BI tool, as previously mentioned, is part of the day to day work of about 8000 users of the company, using queries to assist in the analysis inherent in their professional activities. Monthly, about 150 thousand executions of more than 8000 business user queries in the production environment are recorded.

A query is created to meet a specific need for business analysis, and can contain several input filters and several rows and columns that can be customized according to user navigation. Thus, some queries work with millions of records and take a long time to run. Besides relying on adequate filters, the query response time is also directly affected by the concurrent use of the BW server by other users. Thus, the response time of a single query using the same filters can be very different for two runs in different times.

The main problem today is that there is no way to estimate the execution time in advance, so that the user can decide whether or not to execute the query in the moment. Without this information, many users start a query and when they check that it is taking too long, they stop it. However, the query may continue to run on the server, overloading the machine unnecessarily.

If before executing the query the user had its forecast of execution time, he could decide whether to run it now or postpone it to a time that has faster response avoiding unnecessarily burden for the server with an application that will not be completed.

The execution time of all the queries from BW for each of the users is recorded by the BW Statistics BW SAP software another tool that is also part of the SAP suite. We intend to use this historical basis for predicting the execution time of queries based on previous runs. Valuable information is available from the BW Statistics, such as the user who executed the query, the total execution time, date and time of execution and number of lines returned.

There are two main benefits expected from implementing a solution to predict the execution time of queries from BW. First, users knowing in advance the predicted performance of a particular query may avoid its unproductive execution and the indefinite awaiting for the information he or she needs. Second, by possibly not executing a query classified as long for the moment, the user will relieve the server from an unproductive burden.

4 SOLUTION DESCRITPION

To execute a BW query, the user enters the desired input filters. Depending on the existing filters in the query, the same query may be specified for different periods of time. For example, for a period of one month or one complete year. Therefore, the filters play an important role in determining the size of the result set of the query. Hence, to minimize the problem of different running times of the same query caused by the difference of input filters which may imply in different sizes of the result, we have chosen for this work only queries that are always executed using the same filter. Thus, differences in execution time are caused only by the occupation of the server, not by higher or lower amount of records returned by the query. Thus, the queries chosen for this study were:

- **Query 1:** P_5_CO_P_COOM07M_00014 - Gross Administrative Expenditure
- **Query 2:** P_5_PS_P_PSCO02M_0029 – CAPEX
- **Query 3:** P_5_MM_P_MMPU14M_0002 - Total Purchasing

The following subsections will detail the process of selecting variables, handling of the historical database, setting adequate configurations of Weka and execution of simulations. The database used was extracted from BW Statistics for the years 2010 and 2011.

4.1 Variables choice and the database

The BW Statistics tool provides valuable information regarding the execution of queries. For this study, the information considered relevant in the first analysis is illustrated by Figure 4.1.

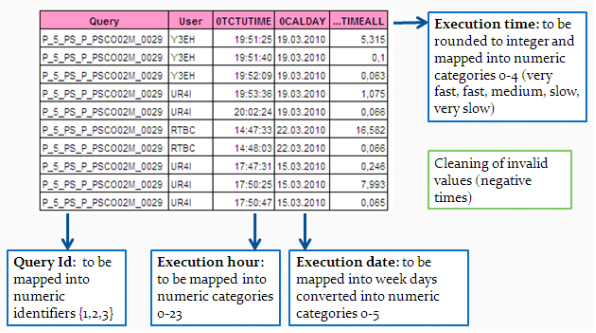


Figure 4.1. Relevant Information of BW Statistics

The column "Query" refers to the technical name of the query. As only three types of queries were used in this study, this field has been mapped to identifiers 1,2 and 3, representing the query used. The column "User" provides information on which user performed the query. As we consider that queries are executed the same way, regardless of the user, we do not consider this information relevant to this study. The information "0TCTUTIME" represents the time of execution of the query. This information is mapped into classes 0-23 representing the hour of the day when the query was performed. The column "0CALDAY" represents the date of execution of the query. As we consider the day of the week as the most relevant information instead of a specific date, this information was mapped into categories 0-5, where 0 represents Saturday or Sunday and the other values from Monday to Friday.

Finally, the column information "TIMEALL" represents the query execution time in minutes. This information is rounded to integer and then mapped into categories 0-4 numerical classification representing the query, as follows:

0: Very fast - up to 2 minutes
1: Fast - 2 to 5 minutes
2: Medium - 5 to 10 minutes
3: Slow - 10 to 20 minutes
4: Very slow - more than 20 minutes

In addition to these conversions, the database passed through a cleanup that eliminated inconsistent values, eg, negative execution time. No incomplete information was found in the database. The resulting database, consisting of 10,893 records, was divided into training set (2/3 or 7262 records) and test set (1/3 or 3631 records), both chosen randomly. Then the training and testing files for Weka were created, as detailed in the next subsection.

4.2 Weka Configurations and Execution of the Simulation

The training and testing files were generated from the database as shown in Figure 4.2. We used the Multilayer Perceptron (MLP) network topology and the BackPropagation (BP) learning algorithm.

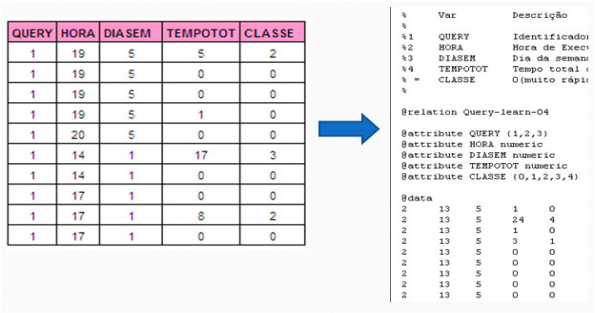


Figure 4.2. Creation of Files for Weka

Various configurations were tried in Weka in order to achieve the best classification rate in the samples. Figure 4.3 illustrates a neural network generated by Weka.

We used nine different settings, illustrated by Table 4.1:

I: Without normalization of the input attributes

II: With normalization of the input attributes and varying the neurons in the hidden layer as follows:

a) 4 neurons
b) 5 neurons
c) 6 neurons

III: With normalization of the input attributes and varying the number of training epochs as follows:

a) 1 epoch
b) 100 epochs
c) 1000 epochs

IV: With normalization of the input attributes and using a validation set. V: With normalization of input attributes and binary encoding of input attributes.

V: With normalization of input attributes and binary encoding of input attributes.

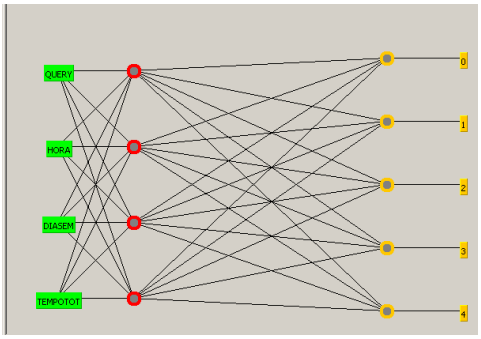


Figure 4.3. Neural Network generated by WEKA

Table 4.1. Weka Configurations

	I	II a	II b	II c	III a	III b	III c	IV	V
hiddenLayers	a	4	5	6	4				
nominalToBinary Filter	False								True
normalizeAttributes	False	True							
trainingTime	500	500	500	500	1	100	1000		
validationSetSize	0	0	0	0	0	0	0	10	0

This section described the proposed solution to the problem of predicting the execution time of BW queries. The next section presents the results.

5 RESULTS

After the execution of nine simulations as detailed in the previous section, different results were obtained. Table 5.1 summarizes these results obtained in different simulations. The results were compared using the following criteria (all in %):

Correct classification
Incorrect classification
Mean absolute error MAE
Root mean squared error RMSE
Relative absolute error RAE
Root relative squared error RRSE

As we can see in Table 5.1, the configuration that showed the best results was the configuration IIIc. This configuration used normalization of input attributes, four neurons in the hidden layer and 1000 epochs of training. It did not use the validation set and nor binary encoding of the input attributes. In contrast, the setting that showed the worst results was the setting IIIa. This configuration also used normalization of input attributes and four neurons in the hidden layer, but used only one training epoch, it also did not use neither the validation set and nor the binary encoding of the input attributes.

Table 5.1. Summary of Results

	I	II a	II b	II c	III a	III b	III c	IV	V
Epochs	500	500	500	500	1	100	1000	1000	1000
Hidden layer	a	4	5	6	4	4	4	4	4
Correct classification (%)	90,3%	90,0%	87,7%	87,9%	67,4%	81,5%	94,3%	82,6%	85,7%
Incorrect classification (%)	9,7%	10,0%	12,3%	12,1%	32,6%	18,5%	5,7%	7,4%	14,3%
Mean absolute error MAE (%)	5,5%	5,9%	6,2%	6,3%	21,0%	11,4%	4,8%	9,1%	7,4%
Root mean squared error RMSE (%)	17,0%	16,6%	19,3%	18,6%	32,1%	22,2%	13,4%	21,6%	21,4%
Relative absolute error RAE (%)	26,8%	28,8%	30,4%	30,8%	102,6%	55,7%	23,4%	44,4%	36,2%
Root relative squared error RRSE (%)	52,8%	51,7%	60,1%	57,8%	99,8%	69,1%	41,6%	67,2%	66,4%

There is a strong indication that for this particular problem and considering the data used, the number of training epochs was a decisive factor in the results, since as we reduce this number (settings IIIa and IIIb), the results become far worse. The variation in the number of processors in the hidden layer can also have some correlation, because when we start to increase this number (configurations IIb and IIb), the results begin to deteriorate, indicating that four processors in the hidden layer represent a good number for this problem.

The use of a validation set (configuration IV), which is usually a good practice, gave much worse results. In contrast, we expected a worse outcome when we do not use the normalization of the input attributes (configuration I), but the results were very similar (and even slightly better) than most of the settings that used the normalization.

6 EVALUATING THE SOLUTION

After performing the simulations and analysis of the results some suggestions of future work as well as some open questions aiming at improvements were found. These points are presented below.

6.1 Treatment of the Database

Examining the training base, after the simulations, we found that it showed some discrepancies that were not previously treated. We cite as an example, the number of occurrences of the second query (61% of training base), which is much larger than the number of occurrences of the query 1 and 3, as illustrated in Figure 6.1.

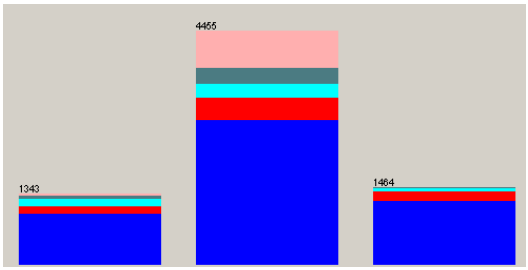


Figure 6.1. Distribution of Training Base per Query (1, 2 and 3)

We also observed by Figure 6.2 a discrepancy in the class of runtime: class 0 (very fast) represents 68% of the training base. In further work, minimization of such differences should be sought by means of homogenization techniques of the database.

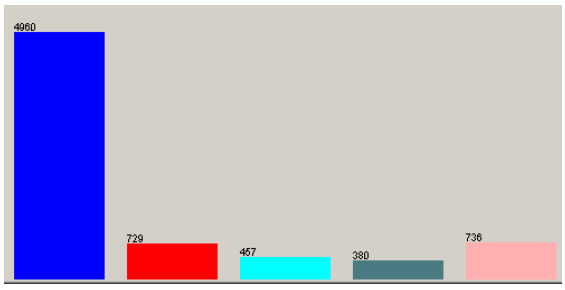


Figure 6.2. Distribution of Training Base per Class (0 to 4)

Another point observed from the analysis of the training base is the largest concentration of execution of queries between 11h30 and 0h (UTC), as illustrated in Figure 6.3. In future work, the use of hour intervals instead of the hour variable may possibly give more accurate results.

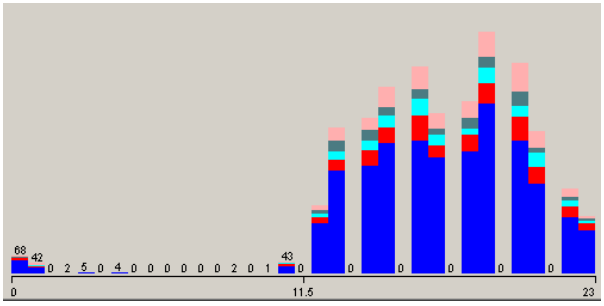


Figure 6.3. Distribution of Training Base per Execution Time (0 to 23)

6.2 Use of other queries

The aim of this paper is to investigate the application of neural networks in the problem of predicting the response time of BW queries. Only three types of queries were used in this study. New studies are required to confirm the results obtained, using a wider variety of queries. Moreover, as mentioned earlier, the parameterization of queries implies in different user input filters and, depending on the values used, the same type of query can return a very different number of records from one execution to another, influencing directly in its running time. Therefore, in future, we should not only use a larger number of queries, but also study a way to handle the customization of the filter input queries. One possible approach is to tackle the filter issue mentioned in section 4. The idea is to perform the classification of a query into different categories considering also the possibilities of filters and record for each execution, which filters were used and what were their values.

6.3 Integration of another system with Weka

Another interesting future work and extremely useful for the company would be the integration of the estimate generated by Weka with a system that the user could use to check the predicted time of execution of a query "now" versus a future time (eg, within 15 min, within 1 hour, in 4 hours). Thus, the user could decide the best time to execute the query and reduce the occupation of the server with unproductive and inconsequent queries.

6.4 Other methods

For future research, we intend to evaluate other methods for solving the problem of forecasting performance of BW queries, for example, fuzzy logic and neuro-fuzzy models. The idea is, after its implementation, to compare the results in order to raise issues not yet perceived for the problem at hand.

This section presented some suggestions for future work within the same theme explored in this article. The following section concludes this paper.

7 CONCLUSION

This paper investigated the application of neural networks in the problem of prediction of execution time of BW queries in a big organization where this represents a huge problem. Section 2 presented a summary on Neural Networks, the tool Weka and SAP BW. Section 3, in turn, detailed the problem in question and section 4 describes the proposed solution. Next, Section 5 presented the results and section 6 evaluated them, pointing out some possible future work.

The classification model utilized in this work was built using MLP networks and the BackPropagation algorithm. The setting that showed the best results used normalization of input attributes, four neurons in the hidden layer and 1000 epochs of training. It used neither validation set nor binary encoding of the input attributes. This configuration classified correctly 94.3% of the samples in the expected classes and proved to be the best configuration of this research.

Finally it must be pointed out that the implementation of such a mechanism to predict the execution time of BW queries is extremely useful to the company, not only to provide greater visibility to the users of the expected time of their queries, but also to avoid burdening the server with queries that are interrupted by the users because they are taking longer than expected. Thus, we intend to deepen this research using the suggestions for future work outlined above.

REFERENCES

[INMON, 1992] Inmon, W. H., *Building the Data Warehouse*. 1st Edition. Wiley and Sons, 1992.

[HAYKIN, 1999] Haykin, S., *Redes neurais - Princípios e prática*, 2a edição, Editora Bookman, 1999.

[JANG ET AL, 1997] Jang, J.S.R, Sun, C.T., Mizutani, E., *Neuro-fuzzy and soft computing: a computacional approach to learning and machine intelligence*. Prentice-Hall, Upper Sadler River, New Jersey, USA, 1997.

[ZADEH, 1992] Zadeh, L., *Fuzzy logic, neural networks and soft computing*. Proceedings of the 2nd International Conference on Fuzzy Logic and Neural Networks, Izuka, Japan, PP. 13-14, 1992.

[CYBENKO, 1989] Cybenko, G., *Approximations by superpositions of sigmoidal functions*. Math. Control, Signals, Systems, 2:303-314, 1989.

[HORNIK et al, 1989] Hornik, K., Stinchcombe, M., White, H., *Multilayer Feedforward Networks are Universal Approximators, Neural Networks*, Vol. 2, pp. 359-366, 1989.

[WITTEN & FRANK, 2005] WITTEN, I. H. e FRANK, E., *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann Publishers, 2nd edition, 2005.

[WEKA, 2011] Site da ferramenta Weka: http://www.cs.waikato.ac.nz/ml/Weka/. Acessado em dezembro de 2011.

[McDONALD et al, 2006] McDonald; Wilmsmeier, Dixon, Inmon, *Mastering the SAP Business Information Warehouse*, Second Edition ed. Wiley, 2006.

[PALEKAR et al, 2010] Amol Palekar, Bharat Patel, and Shreekant Shiralkar, *A Practical Guide to SAP Netweaver Business Warehouse 7.0*. SAP PRESS, 2010.

Pruning AdaBoost for Continuous Sensors Mining Applications

M. Rastgoo, G. Lemaitre, X. Rafael Palou, F. Miralles and **P. Casale** [1]

Abstract. In this work, pruning techniques for the AdaBoost classifier are evaluated specially aimed for a continuous learning framework in sensors mining applications. To assess the methods, three pruning schemes are evaluated using standard machine-learning benchmark datasets, simulated drifting datasets and real cases. Early results obtained show that pruning methodologies approach and sometimes out-perform the no-pruned version of the classifier, being at the same time more easily adaptable to the drift in the training distribution. Future works are planned in order to evaluate the approach in terms of time efficiency and extension to big-data analysis.

1 Introduction

As the number of sensors deployed every day in the real world increases, the ambition of mining these continuous data-streams becomes a crucial part in applications. In the recent years, data mining techniques started to be very popular in sensors mining tasks specially when related to learning from data streams [3] [10]. These techniques, stated upon the machine learning framework, are designed to generate a predictive model from a well sampled training dataset distribution. The model is further used to classify any future instance of data without the possibility to be updated if the value distribution of the data-stream changes. In other words, the paradigm provided by the typical machine learning setting is not suitable for continuous mining of data streams [5]. The AdaBoost learning function [2] allows a suitable framework for mining continuous streams [8]. Being an incremental ensemble of classifiers, this learning function is updated to *grow its knowledge* just adding new classifiers to the previous models. Nevertheless, when many subsequent batches of data are provided, Adaboost tends to create large ensembles that suffer of two main drawbacks: (i) increasing memory needed to store the decision model and (ii) over-fitting. *Pruning* techniques can be suited for reducing the dimension of the ensemble by selecting only specific models. The first attempt of pruning an AdaBoost classifiers was introduced by Margineantu and Dietterich [6] by mean of comparing five different methods, namely (i) *early stopping*, (ii) *KL divergence*, (iii) *Kappa statistics*, (iv) *Kappa error convex Hull* and (v) *Reduce error with back-fitting*. Hernanadez-Lobato et al. [4] used Genetic Algorithms to prune the AdaBoost ensemble, searching in the space of all possible subsets of classifiers created by AdaBoost. Zhang et al. [11] defined pruning as a quadratic integer programming problem with the aim to find a fixed size subset of k classifiers with minimum misclassification and maximum diversity. Nevertheless, those works are no suitable solutions for pruning AdaBoost in a continuous learning framework. In this paper, experiments on pruning methods for continuous data-streams mining are performed. The AdaBoost algorithm is trained on subsequent batches of incoming data followed by consecutive *pruning* steps. The advantage of this approach is twofold: (i) on the first hand, when new concepts are learned, pruning allows to maintain the ensemble in order to be the least memory consuming and (ii) on the other hand, pruning provides a first attempt to retain only the *significant information* acquired from previous knowledge. The reminder of this paper is organized as follows. In Section 1, the continuous learning framework, the AdaBoost algorithm and the used pruning methods are introduced and explained in details. In Section 3, validation protocols are described and, in Section 4, results are presented. Finally, Section 5 discusses the obtained results and concludes the paper.

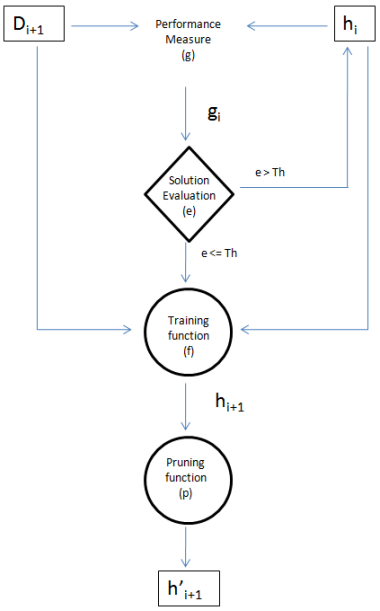


Figure 1. Continuous Learning Framework

2 Pruning AdaBoost in Continuous Learning

In a continuous learning framework, as shown in Fig. 1, new knowledge is acquired only when the current model does not fit anymore the incoming data-stream distribution [9]. This decision is performed by evaluating the current classifier using the function g as performance measure and evaluating the obtained performance e. When e is not *good enough*, the current model h_i is updated training the

[1] Barcelona Digital Technology Center, Barcelona, Spain, email: plcasale@bdigital.org

learning function f with the new incoming data D_{i+1}. Incremental learning functions should be preferred. In this way, only the new incoming data will be used for both maintaining the previous knowledge acquired, not having to store historical data. The AdaBoost algorithm represents an incremental learning function able to properly meet these requirements. Nevertheless the classifiers created by AdaBoost grows linearly as many subsequent learning steps are performed. Here, the pruning function p allows to maintain the model computationally optimal. Aim of this work is evaluating between different pruning functions p in terms of classifier performance. In the following subsection, AdaBoost and the pruning methods are presented and explained in details.

2.1 AdaBoost

AdaBoost, short for Adaptive Boosting, is an ensemble learning algorithm that allows to obtain an high performance classifier by a linear combination of weak learners. Algorithm 1 shows the pseudocode for AdaBoost. The algorithm takes as input a training set $(\mathbf{x}_i, y_i)$ where $\mathbf{x}_i$ is a N-dimensional feature vector, and y_i are the class labels. After T rounds of training, T weak classifiers h_t and T weights α_t are combined to assemble the final strong classifier. Higher weights α_t are assigned to the best weak classifiers h_t. Instantiations of AdaBoost may differ due to the choice of the

Algorithm 1 AdaBoost Algorithm

Input:
- Training set of N samples $(\mathbf{x}_i, y_i)$, with $i = 1 \dots N$, $\mathbf{x}_i \in R^N$, $y_k \in Y = \{1, +1\}$;
- Weak learning algorithm *WeakLearn* ;
- Number of learning iteration T ;

Initialize $W_1(k) = 1/N, k = 1, \dots, N$;

for $t = 1, \dots, T$ **do**

1. Train *WeakLearn* using distribution W_t and get weak hypothesis h_t ;
2. Compute classification error $\epsilon_t = Pr_{k \sim W_t}[h_t(x_k) \neq y_k]$;
3. Compute $\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$;
4. Update distribution:

$$W_{t+1}(k) = \frac{W_t(k) \exp(-\alpha_t y_k h_t(x_k))}{Z_t} ;$$

where Z_t is a normalization factor chosen so that W_{t+1} will be a proper distribution function.

end for

Output:
$H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))$;

weak learning algorithm, defined as a learner performing slightly better than random guessing ($> 50\%$ right-classification). A variety of weak learners e.g., neural networks or decision trees can be used. Decision stumps are the most common weak classifiers used in AdaBoost. Decision stumps are one-level decision trees equivalent to a threshold that best splits the data. Each stump learner is characterized by three parameters: (i) the n^{th} dimension of the features set where the classifier is applied, (ii) the decision level, i.e., the *threshold* splitting the data in the n^{th} given dimension and (iii) the decision sign (-1 or $+1$) determining the inequality direction for the thresholding. For a given batch of data with a set of features of size n, at each iteration of AdaBoost the decision stump that minimizes the error ϵ_t in an n^{th} dimension of the training distribution is selected. The information provided by the final set of decision stumps selected by AdaBoost can be used for mining which are the significant features of the data-stream and, more important, which is the best split in the data.

2.2 Pruning methods

Three different pruning methods have been used and compared, namely (i) Reduce Error, (ii) Learner Weights Analysis and (iii) Pareto Analysis. The Reduced Error algorithm was used first in [6]. Being the original implementation not suitable from a continuous learning framework, an improved version is proposed in this work in order to speed-up the process. Pruning has also been performed using Learner Weights and Pareto Analysis methodologies, both of them able to provide a set of most discriminative learners from the whole ensemble. From the far of our knowledge, no previous application of those methodologies has been done in the tasks of pruning an AdaBoost ensemble.

2.2.1 Reduce Error (RE)

In this algorithm, the first step is performed in order to initialize the pruning distribution W_t and to select the weak classifier h_t from the ensemble H which minimizes the classification error ϵ_t on W_t distribution. This classifier is added to the pruned ensemble P, a weight α_t is assigned to it and W_{t+1} distribution is also updated as in AdaBoost routine. Then, iteratively, each remaining classifier h_t is individually added to the ensemble P and the classification error ϵ_t of this new ensemble is evaluated on the pruning set using W_{t+1} distribution. In order to select the best classifier, the classifier h_t combined with P minimizing the classification error ϵ_t is definitely added to P, a weight α_t is assigned to it and W_{t+2} distribution is also updated as in AdaBoost routine. The routine stops when the number of classifiers in the sub-ensemble P reaches a ppre-specified size. The two main changes with respect the original RE algorithm are the following. In the original version, a final back-fitting approach is performed only after the selection of each weak classifier while in our approach selection is done at each step. In addition, each weak classifier is added to the pruned ensemble P only after being re-weighted. This procedure ensures better classification results than the original RE formulation.

2.2.2 Learner Weights Analysis (WA)

From the distributions of the weights α_t in the ensemble, weak learners were selected based on the following assumptions: (i) weak learners with higher ensemble weight α_t are the best weak learners of the ensemble and (ii) an ensemble is better when more diversified the classifiers forming it are. The technique works as follow. AdaBoost is applied on the batch of data to obtain an ensemble of T classifiers. Then, a matrix M is built, by grouping the ensemble weights α_t of each decision stump classifier using their dimension parameter. M is of size $n \times D$ where n is the number of element for each of the D dimensions. In order to select the *best* classifiers, M is first sorted formerly by row and subsequently by column, always in a descendant order. M is transformed into a vector V by concatenating all its columns. Finally t classifiers corresponding to the t first weights of V, with $t << T$, are selected. The value of t determines the pruning percentage.

2.2.3 Pareto Analysis (PA)

PA is based on the assumption that few key actions will produce significant overall effects. Applied to ensemble learning, this technique implies that only few key weak classifiers will have an high impact on the overall performance of the ensemble. *PA* proposed a statistical

point of view in order to select these key classifiers. This technique is used to estimate effectiveness of each feature dimension, and accordingly selects the classifiers from feature dimensions with high impact. The effectiveness could be adjusted using a threshold. First, the features are grouped based on the total number of ensemble weight which are considers as outliers in each dimension. The outliers could be found with reference to first and third quartile (Q_1, Q_3), and inter quartile range (IQR). Values above $Q_3 + 1.5 \times (IQR)$ are considered as outliers in each case. The frequency distribution of these outliers is sorted in descendant order and the cumulative distribution is computed. Then, the features dimensions are selected based on a threshold level corresponding to the number of classifiers to keep. All dimensions with lower cumulative percentage than the threshold (i.e. desired percentage of maximum cumulative value) are taken into account. From the selected feature dimensions, the maximum weights are used to highlight the learners. The technique can be perceived as a principle dimension selection, where the dimensions considered as more important are selected.

3 Validation Protocol

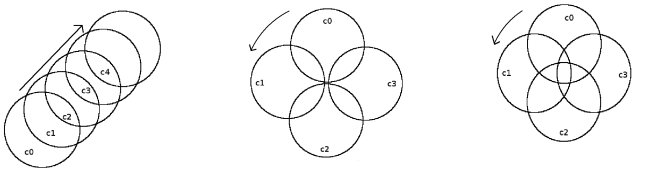
Three typologies of experiments have been performed in order to validate the effectiveness of the pruning methods on both static and drifting distributions. A cross-validation approach has been used for validating the methods. At each step of the cross-validation procees, the dataset has been randomly divided into three sub-sets, training (50%), pruning (40%) and testing(10%) sets. In the following sections the validation protocols adopted for each topology of experiment are described. Under the model described in Fig. 1, a proper threshold Th has been chosen in order to train the model always on the new incoming data.

3.1 UCI Datasets Repository

Five datasets from the UCI repository [1] have been used for evaluating the effectiveness of the pruning methods. In this validation step, the KL divergence method as originally proposed in [6], has been added in order to have a baseline comparison. The datasets considered are Australian, Breast, Diabetes, Heart and Pima. The mean number of instances in the datasets is around 700, except Heart having 270 instance. The aim of the experiment is to analyse the results by pruning at 90% an initial ensemble. The average error rate for each technique was computed using a modified version of ten fold cross-validation able to consider the pruning sets into the evaluation process, with the percentage previously outlined. AdaBoost algorithm was used to create an ensemble of hundred weak classifiers. Then, each pruning method was performed in order to create a pruned sub-ensemble containing only ten classifiers.

3.2 Simulated Drifting Datasets

The second set of experiments has been focused on testing the pruning methods in a continuous learning framework. These have been performed using three sets of simulated data-streams that include drifting. The datasets are generated using the software provided by [7]. Figure 2 shows the three different settings for each experiment. Four linear drifts have been considered for the first dataset and three circular drifts have been created for the remaining two datasets. The ensemble was incrementally grown using all the drifted distributions. The experiments performed using the simulated datasets are described in the following.



(a) Linear Drift (b) Circular Drift (c) Circular Narrow Drift

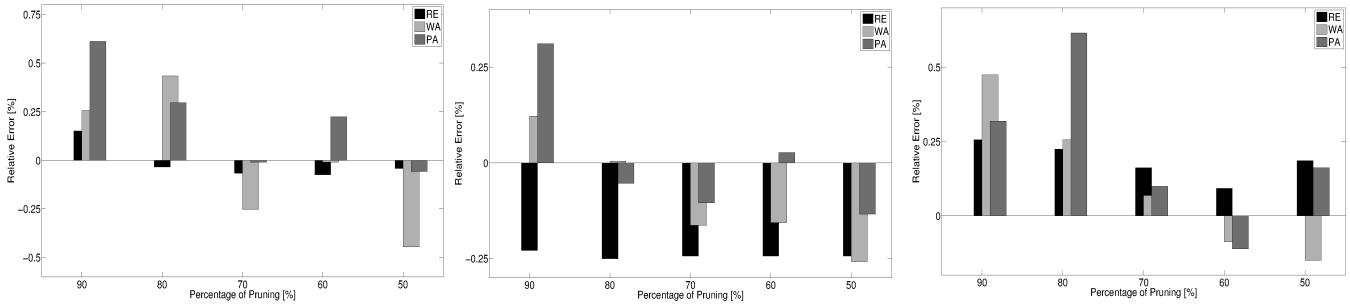
Figure 2. Artificial data with drifts

Exp. 1: In the first experiment, it is assumed that data distribution is subject to the change due to different drifts and the ensemble is incrementally grown over the drifted batches of incoming data with the main aim to classify the current batch of information. After the training, the pruning and the testing are applied on a different samplings of the same drifted batch. The experiment is repeated five times following a 5-fold cross-validation paradigm.
Exp. 2: The aim of the second experiment is to evaluate the potential of pruning in classifying both previous and current information. With the training kept as in the previous experiment, at each step i the ensemble is pruned and tested on pruning and testing sets of the joint distribution $C_0 \cup \ldots \cup C_i$. The experiment has been performed on five different runs, following a 5-fold cross-validation paradigm.

3.3 Real World Datasets

Three real-world datasets have been used in order to evaluate the proposed methodology on a real world scenario. The datasets considered are described in the following.

- The *Sensor Stream*(SS) dataset [12] contains sensors information (temperature, humidity, light and sensor voltage) collected from fifty-four sensors deployed at Intel Berkeley Research Lab. The whole stream contains consecutive information over two months (2 219 803 instances). The experiment aims to infer the illuminance state based on the measurements provided by each sensor. Illuminance higher than 200 *lux* are considered as class 1 otherwise considered as class -1. Every fifteen days, a new batch of data is collected which leads to three drifts considering the changes in the lab environment due to weather, humidity and office work. The experiment was performed using 4-fold cross-validation paradigm.
- *Power Supply*(PS) [12] is the second dataset used. The dataset contains hourly power supply consumptions of the Italian electricity company. The stream contains three year power supply records from 1995 to 1998 (29 928 instances). The experiment aims to predict the day state morning (1) - night (-1)) based on the raw consumption value. The drifting in this stream is mainly derived by some features such as the season, weather, hours of a day and the day of the week. The data were split in three batches representing one drift for each year. The experiment was performed using 3-fold cross-validation paradigm.
- *Elec 2*(E2) is the third dataset used. This dataset containing 27 549 instances is composed of seven drifts, each representing a week day. The drifts are due to changes of power consumptions over the weekdays. The experiment was performed using 7-fold cross-validation paradigm.



(a) Results obtained on Linear Drift in Exp.1 (b) Results obtained on Circular Drift in Exp.1 (c) Results obtained on Circular Narrow Drift in Exp.1

Figure 3. Results obtained on simulated drifting datasets for Exp.1

As in the Exp. 2 on simulated data, AdaBoost is trained for each drift on the training set of current data. The pruning function is applied on a pruning set which contains samples of previous and new batches of data.

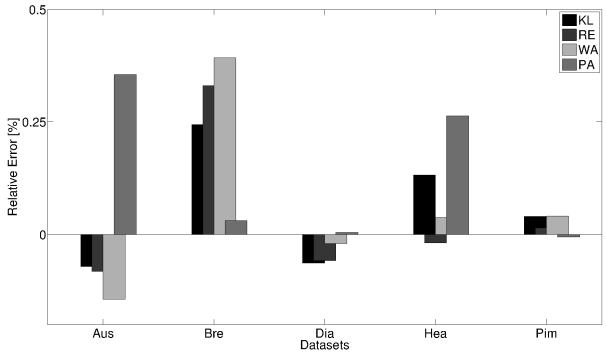


Figure 4. Performance of pruning methods on UCI datasets with %90 pruning

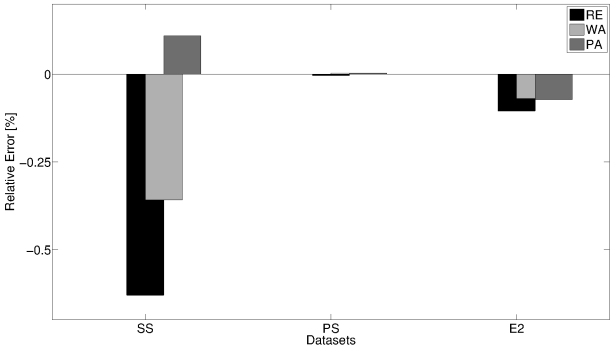


Figure 5. Performance of pruning methods on real world datasets

4 Experimental Results

In this section, results obtained on the experiments described in the previous section are reported. Misclassification error has been chosen as performance measure. In particular, the pruning methods has been evaluated using the relative error (ϵ_{rel}) with respect to the error provided by the no-pruned version of AdaBoost, computed as shown in Eq. 1. Hence, methods with negative relative errors are performing better than the reference model.

$$\epsilon_{rel} = -1 \cdot \frac{\epsilon_{no\ pruned} - \epsilon_{pruned}}{\epsilon_{no\ pruned}} \quad (1)$$

4.1 UCI Datasets Repository

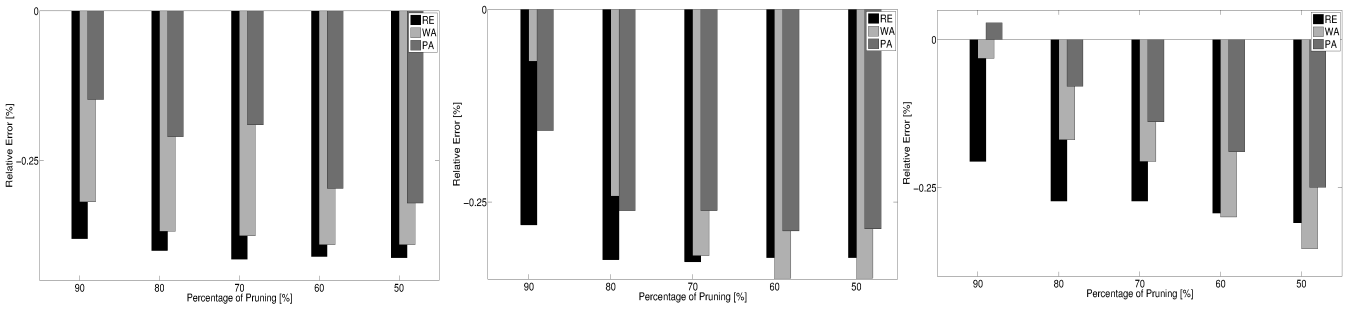
In Fig. 4 the results obtained on the five UCI datasets are reported. RE is the method performing better than the others, being better than the reference in Aus, Dia and Hea datasets, and slightly worst than the reference in Pim. Similar behavior is obtained by WA. Pruning performs always bad on Bre, where the best result is provided by PA.

4.2 Simulated Drifting Datasets

Results obtained on simulated drifting datasets with Exp. 1 are reported in Fig. 3. RE is the best pruning method for linear and circular drifting datasets, as previous experiments suggest. In both linear and circular drifting, WA performs better than PA. Non of the pruning methods work better than the no-pruned version for high percentage of pruning. Nevertheless, WA works slightly better than no-pruned Adaboost when the percentage of pruning is almost 50%. As it may be expected, the performance of the pruned ensemble generally get worse as the percentage of pruning increases. Nevertheless, RE is able to maintain its performance constant over the pruning percentage in the circular dataset and almost constant in the narrow pruning dataset. For Exp. 2, results obtained on simulated drifting datasets are reported in Fig. 6. In this setting, all the pruned ensemble behave better than their correspondent no-pruned classifiers. As all previous experiment suggest, RE is the best method, followed by WA. Also in this case, although the performance of the methods decreases as the percentage of pruning increases, RE remains almost constant regardless of the percentage. It should be also noted that the AdaBoost performance in this experiment is rather bad, reaching a global error up to 40%. The pruning methods improve this performance until reaching an error of 25%.

4.3 Real World Datasets

Results obtained on the real world dataset are shown in Fig. 5. Results obtained with the PS datasets are shown in Fig. 7. RE confirms to be the best pruning method, followed by WA. For SS and E2 datasets, WA and PA provide the same performance. It should be noted that RE performs better than the no-pruned version for all the experiments.



(a) Results obtained on Linear Drift in Exp.2 (b) Results obtained on Circular Drift in Exp.2 (c) Results obtained on Circular Narrow Drift in Exp.2

Figure 6. Results obtained on simulated drifting datasets for Exp. 2

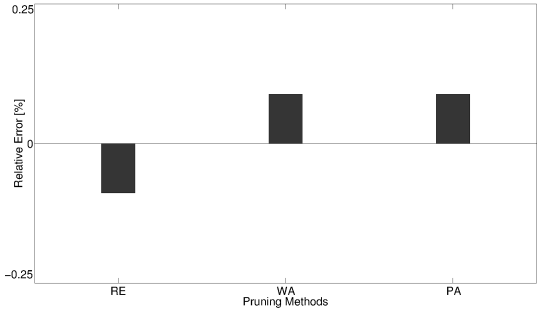


Figure 7. Performance of pruning methods on the PS dataset

5 Discussions and Conclusions

In this work, experiments have been carried out in order to evaluate the potential of different pruning methods and their performance in the framework of continuous learning. The *Reduced Error* method is the most consistent method followed by *Learner Weight Analysis*. The use of *Pareto Analysis* does not seem to be justified during the experiment. Nevertheless, one of the important characteristic of this method consists in the capability of defining automatically the number of classifiers of the pruned ensemble. PA may be automatized by thresholding the performance. Early results show that this *automatic* version performs better than the original method in most of the cases. Experiments on simulated datasets in case of $Exp\ 1$ show that pruning methods are more efficient over wider drifted distribution rather than narrow drifted distribution. Due to the nature of the narrow circular dataset, drift stages have more common area and since in this experiment, current stage has more effect for pruning, compare to previous stage, the pruning performances are slightly lower. At the same time, $Exp\ 2$ show that pruning methods perform better than the original classifier when the whole drifting distribution is presented. Based on Fig. 6, pruning ensemble through the incremental learning, definitely improves the final results. Finally, results obtained by experiments on real datasets prove that pruning through the continuous learning process provides very close or better results than AdaBoost. As future works, an evaluation of the method efficiency in terms of computational complexity will be considered since this parameter has a great importance in a continuous learning framework. For this main motivation, the reduced error method had been modified in our research in order to be conceptually capable to run following time efficiency guidelines and methods based on genetic algorithm and semi-definite programming have been not used for comparison. Finally, a study on the extension of the proposed methods towards a big-data approach is planned to be done. This research shows that pruning by selecting the weak classifiers from different pools of subsampled data may improve the final ensemble in terms of accuracy, diversity and adaptation ability to drift. The employed procedures in this work can be easily adapted for large datasets and continuous learning environment with the high quantity of incoming data.

6 Acknowledgments

This work is supported by the Information and Communication Technologies Collaborative Project action BrainAble within the Seventh Framework of the European Commission, project number ICT-2010-247447.

REFERENCES

[1] D.J. Newman A. Asuncion. UCI machine learning repository, 2007.
[2] Y. Freund and R. Schapire, 'A short introduction to boosting', *J. Japan. Soc. for Artif. Intel.*, **14**(5), 771–780, (1999).
[3] J. Gama and M. Gaber (Eds), *Learning from Data Streams – Processing techniques in Sensor Networks*, Springer, 2007.
[4] Daniel Hernández-Lobato, José Miguel Hernández-Lobato, Rubén Ruiz-Torrubiano, and Ángel Valle, 'Pruning adaptive boosting ensembles by means of a genetic algorithm', in *IDEAL*, pp. 322–329, (2006).
[5] Ludmila I. Kuncheva, 'Classifier ensembles for changing environments', in *In Multiple Classifier Systems*, pp. 1–15. Springer, (2004).
[6] Dragos D. Margineantu and Thomas G. Dietterich. Pruning adaptive boosting, 1997.
[7] Leandro L. Minku, Allan P. White, and Xin Yao, 'The impact of diversity on online ensemble learning in the presence of concept drift', *IEEE Trans. on Knowl. and Data Eng.*, **22**(5), 730–742, (May 2010).
[8] Martin Scholz and Ralf Klinkenberg, 'Boosting classifiers for drifting concepts', *Intelligent Data Analysis (IDA), Special Issue on Knowledge Discovery from Data Streams*, **2006**, 2007, (2006).
[9] Jan Tozicka, Michael Rovatsos, Michal Pechoucek, and Stepan Urban, 'Malef 58: Framework for distributed machine learning and data mining', *Int. J. Intell. Inf. Database Syst.*, **2**(1), 6–24, (February 2008).
[10] Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han, 'Mining concept-drifting data streams using ensemble classifiers', in *Proceedings of the ninth ACM SIGKDD*, KDD '03, pp. 226–235, New York, NY, USA, (2003). ACM.
[11] Yi Zhang, Samuel Burer, and W. Nick Street, 'Ensemble pruning via semi-definite programming', *Journal of Machine Learning Research*, **7**, 1315–1338, (2006).
[12] X. Zhu. Stream data mining repository, 2010.

Finding the best ranking model for spatial objects

Hadi Fanaee Tork[1]

Abstract. Top-k spatial preference queries has a wide range of applications in service recommendation and decision support systems. In this work we first introduce three state of the art algorithms and apply them on a real data set which includes geographic coordinates and quality data of over 355 hotels, 276 point of interests and 563 restaurants in Lisbon, Portugal extracted from well-known TripAdvisor[2]. This is the first time that mentioned algorithms are evaluated on a real data set. We also use some optimization tasks for the estimation of algorithms parameters. Finally we rank the hotels using the best obtained ranking model. Result reveals that influence score with a particular radius is able to rank spatial objects very near to the real rankings.

1 INTRODUCTION

There exists an wide range of location-based applications that rely on spatial preference queries. For instance, the tourist species a spatial constraint (for instance the range around a hotel) to retrieve the facilities around the hotel. Then, if the eligible facilities are rated, the result of the query might be the top-k hotels which have the best ranked facilities [3]. Top-k spatial preference query answers such kind of questions. It returns a ranked set of the k best data objects based on the non-spatial score (quality) of feature objects and spatial score (distance) in its spatial neighborhood [1,2]. Several approaches have been proposed for ranking spatial data objects based on defining the score of a spatial data object p based on the scores of feature objects that have p as their nearest neighbor. In the rest of the paper we first introduce a general framework of three algorithms entitled Range Score, Nearest neighbor (NN) and Influence Score. Then in section 3 we present the data set used in the paper. In the section 4 we explain our performed experiments. Later in section 5 we express the results. in section 6 we show how we rank hotels of Lisbon based on the best ranking model obtained and finally in section 7 we discuss the results and bring the conclusion of the paper.

2 TOP-K SPATIAL FRAMEWORK

A Spatial preference query, ranks the spatial objects based on quality of its neighbor facilities. For instance a tourist might retrieve a sorted list of hotels based on the facilities around that (e.g. restaurant, hospital , market, etc.). Assume that p is our point of interest (e.g. a hotel) and we have *m* type of facilities(e.g. restaurant means m=1 and park means m=2). Then assume that f_m^n is *n-th* facility from type *m (e.g.* Restaurant *A)*. First we retrieve a list of candidates for *P* according to Table 1. Table 1 shows how one of the methods choose the primary candidates.

Table 1 Candidate Selection Criteria

Method	
Nearest Neighbor	$\min(d(p, f_m^n))$
Range Score	$d(p, f_m^n) < R$
Influence Score	All

As we can see, Nearest Neighbor, from each type *m* retrieves *n-th* element of that (f_m^n) which has the minimum distance with p. Range score retrieves a list of items which have at least distance(*d*) of pre-defined *R* with *P*. Influence score retrieves all the items for further computation. Afterwards, We define Score of point *P* according to the following equation:

$$S_p = \sum_{1}^{m} Agg\{w_{Ci}^m \times \alpha_{Ci}^m\} \quad (1)$$

Where, *Agg* denotes the aggregation function which can be maximum or sum. *w* is equal to the weight or quality of item(e.g. hotel with 5 star can have weight of 5 and hotel with one star can have weight of 1) and *i* is an index of retrieved candidates. α is influence function which is equal to 1 for Nearest Neighbor and Range score and is equal to the equation 2 for Influence score.

$$\alpha = 2^{-\frac{d(p, f_m^i)}{R}} \quad (2)$$

Where *d* denotes the distance between point P and facility *i* of category *m.* and *R* is a pre-defined radius.

Then the result of Top-K spatial preference query is a sorted list of Sp for all point of interests (*P*).

3 DATA SET

Data set is extracted from a well-known online tourism information source *TripAdvisor* which is the most biggest and richest source for travelers around the world to find the relevant information and other user feedbacks about hotels, restaurants and point of interests. One of interesting service of *TripAdvisor* is providing a raking of all tourism locations. The ranking criteria are not visible to the users but in general is a combination of on users opinions and ratings and other sources. Nowadays many users around the world choose their destination, hotels and places to visit based on this ranking.

We extracted all hotels and all near restaurants and point of interests(POI) corresponding to city of Lisbon, Portugal. All GPS

[1] LIAAD-INESC Porto, University of Porto , hadi.fanaee@fe.up.pt

[2] http://www.tripadvisor.com

coordinates and quality factors were extracted from the Raw crawled HTML pages

We then transferred extracted records to the MySQL databases for further process. Finally we had three tables hotels, restaurants and attractions with 355, 563 and 276 records respectively.

Since for some locations , the GPS coordinates were not available, we employed Google Map API[5] and Yahoo Map API[6] Geocoding service to fetch GPS coordinates. Then we removed the places which their coordinate was not available after the Geocoding step. We also removed those hotels which for them ranking was not available in TripAdvisor.

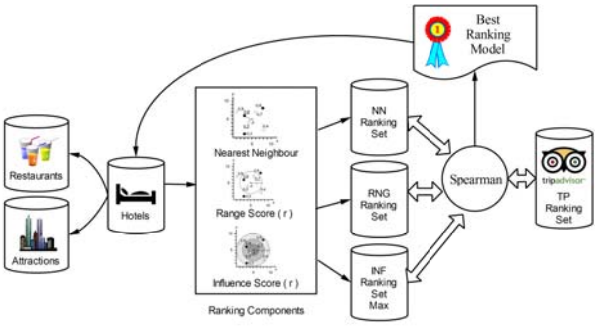


Figure 1. Experiment overview

4 EXPERIMENTS

Two significant problems regarding the Top-k spatial preference query is that first no evaluation on the ranking results is presented yet and second there is not any solution for estimating the radius value in two of algorithms range score and influence score. In other words, when a ranking is made how we can make sure about the correctness of that, or better say how the ranking model correctly assign the spatial objects to the true ranks.

Solving this problem is impossible unless we could compare two generated and real ranking sets together. TripAdvisor real ranking set enable us to perform such comparison and measurement. Our performed experiments are illustrated in figure 1. we first apply Top-K spatial preference query algorithms on the data set and generate three ranking set namely *NN*, *RNG* and *INF* which stands for Nearest Neighbor , Range Score and Influence score respectively. Then in order to evaluate the ranking model we benefit from Spearman's rank correlation coefficient[7]. After this step we find out that which model with which parameters is the best model for predicting the ranking of a hotel. Thus in the next step we employ our best model to rank all the hotels in Lisbon.

As mentioned in the section 2, *Nearest neighbor* is not dependant on radius *R*, so this algorithm doesn't have any input parameters, instead, two other algorithms *Range score* and *Influence score* has radius *R* as their input. In order to study the impact of quality weight on *Influence Score* method, we defined two kind of Influence score, *INFMAX0* and *INFMAX1* so that in the latter one, w is considered to be equal to 1. it means INFMAX1 just consider the spatial property of place and ignores the weight(*w*).

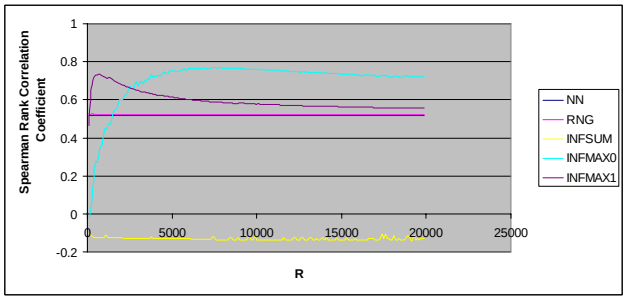


Figure 2. Spearman's rank correlation for different R from 100m to 20000m for 4 rankers NN (nearest neighbor), range score(RNG), influence score with sum module(INFSUM), influence score with max module with considering the rating of attrac-tions(INFMAX0), influence score with max module and not considering the rating of attractions(INFMAX1)

On of the important problem regarding the *Influence Score* approach is determination of R. In order to estimate the best *R* we generated 5 ranking set for R from 100 to 19900 meter by granularity of 100 meter. For both RNG and NN we used maximum aggregation while for INF we tested both maximum(*INFMAX0* and *INFMAX1*) and sum function (*INFSUM*). Then we compute spearman rank correlation coefficient for each 5 generated rankings sets to the TripAdvisor Real ranking set.

Results are shown in figure 2. The vertical axis represents the spearman rank correlation coefficient and the horizontal axis shows the *R* value. The best rankers are those that have the biggest area under their curve. Therefore green curve which is related to the *INFMAX0* would be identified as the best model. INFMAX1 which do not consider the facilities quality is also placed at the second place. The maximum correlation (73.4%) is obtained at R=700m for *INFMAX0* ranker and for INFMAX1 77% correlation is obtained at R=7500m. In terms of *RNG* have a constant behavior between 0.522 and 0.526 very near to *NN* which is always equal to 0.519 and doesn't change by the increasing of *R*.

	InfMax	InfMaxP	InfMaxR	InfSum	InfSumP	InfSumR	NN	NNP	NNR	Rng	RngP	RngR	Review	TPRank	Best
InfMax	1	0.983	0.92	0.159	0.152	0.146	0.781	0.535	0.732	0.784	0.784	0.784	0.56	0.77	0.755
InfMaxP	0.983	1	0.879	0.111	0.107	0.099	0.753	0.517	0.701	0.758	0.759	0.758	0.476	0.724	0.664
InfMaxR	0.92	0.879	1	-0.285	-0.301	-0.3	0.94	0.867	0.925	0.941	0.942	0.941	0.37	0.674	0.642
InfSum	0.159	0.111	-0.285	1	0.991	0.999	-0.882	-3.368	-1.217	-0.761	-0.761	-0.761	0.036	-0.124	-0.76
InfSumP	0.152	0.107	-0.301	0.991	1	0.984	-0.9	-3.403	-1.239	-0.777	-0.776	-0.777	0.011	-0.122	-0.784
InfSumR	0.146	0.099	-0.3	0.999	0.984	1	-0.897	-3.404	-1.233	-0.776	-0.776	-0.776	0.032	-0.138	-0.776
NN	0.781	0.753	0.94	-0.882	-0.9	-0.897	1	1	0.999	0.999	0.999	0.999	0.105	0.519	0.44
NNP	0.535	0.517	0.867	-3.368	-3.403	-3.404	1	1	1	1	1	1	-1.171	-0.173	-0.373
NNR	0.732	0.701	0.925	-1.217	-1.239	-1.233	0.999	1	1	1	1	1	-0.093	0.407	0.294
Rng	0.784	0.758	0.941	-0.761	-0.777	-0.776	0.999	1	1	1	1	1	0.124	0.522	0.441
RngP	0.784	0.759	0.942	-0.761	-0.776	-0.776	0.999	1	1	1	1	1	0.12	0.523	0.441
RngR	0.784	0.758	0.941	-0.761	-0.777	-0.776	0.999	1	1	1	1	1	0.124	0.522	0.441
Review	0.56	0.476	0.37	0.036	0.011	0.032	0.105	-1.171	-0.093	0.124	0.12	0.124	1	0.666	0.864
TPRank	0.77	0.724	0.674	-0.124	-0.122	-0.138	0.519	-0.173	0.407	0.522	0.523	0.522	0.666	1	0.802
Best	0.755	0.664	0.642	-0.76	-0.784	-0.776	0.44	-0.373	0.294	0.441	0.441	0.441	0.864	0.802	1

Figure 3. Spearman's rank correlation for R=7500m and Influence Score with Max module and considering the attractions rating

5 Ranking of Lisbon Hotels

We used our best ranking model (*INFMAX0* with R=7500m) to rank all hotels in Lisbon. Figure 6 shows Spearman's rank correlation computed between all generated rankings sets. P and R at the end of titles stands for attractions and restaurants respectively. For instance *InfMaxR* means that the corresponding

generated ranking set is obtained by just taking into account the restaurants and by using Influence score method. *Best* column represents our best ranking model. The columns that doesn't have any R or P at the end of their title are those which both restaurants and attractions are considered in the ranking generation. Also another two columns review and TPrank denote the number of reviews done for that item in TripAdvisor and the corresponding rank in TripAdvisor respectively.

Some interesting facts can be extracted from this table. For instance intersection of *InfMax* and *TPrank* shows that generated ranking set by InfMax has +0.77 correlation to the real ranking provided by TripAdvisor. Also some other interesting results can be obtained from this table. For example we can realize that Influence score with max aggregation if applied on just restaurant data set has +0.94 correlation with ranking set generated with Nearest Neighbor. We also understand that influence score with sum aggregation never performs good and always show a negative correlation to *TPrank*. If we look the correlation between *NN* and *RNG* we discover an interesting fact. It reveals that by using R=7500m ranking set get highly correlated to nearest neighbor ranker with 99.9% confidence.

6 DISCUSSIONS & CONCLUSION

In this paper we presented a new method for evaluation of Top-k spatial preference query. One of the direct result we obtained was the high performance of original influence score ranker with max aggregation function that shows 77% correlation to real ranking of TripAdvisor. It means that when there is no ranking set available, this method can be a good alternative since it generates close ranking set. Second we proved that despite by a first glance, influence score with sum aggregation could have a wide cover on all attractions and thus could have a better ranking result, the opposite happened and it generally didn't provide a good result.

When we are dealing with very large data set, the computation cost will be the most important factor to choose a solution. Nearest neighbor and Range score can be a good choice since provide constant correlation of approximately 50%.

As we also observed there is not considerable difference between *INFMAX0* and *INFMAX1* them. Even in R<700m not considering INFMAX1 that doesn't consider the quality of facilities performs better. It reveal an important fact. Tourist usually use to visits close attractions to their hotel without considering the quality of them. However when distance goes upper than 700m the quality of that attraction gets important and they pay attention to the rating of that place with the goal of not wasting their time and money in transfer. In other words, tolerance threshold of travelers is the intersection of two curves InfMax0 and InfMax1 which is 2700m. It means that by increasing the distance from 700m to 2700m from the hotel, the motivation of travelers to look for rating of the attractions is increased.

The reason why *RNG* and *NN* show a constant value is this fact that most hotel owners establish their hotel in a place that is close to at least some attractions. Except some minor cases, no hotel company invests on a place that is very far from all attractions. So when there is for example 4-5 attractions near to the hotels, their *NN* and *RNG* is affected by the rating of them and thus doesn't change a lot. Because always it is possible to find one high quality attraction near to the hotel.

The reason why influence score with sum aggregation gets negative correlation is this fact that it counts all attractions and thus consider very far attractions and thus distance in equation 2 goes upper and deduct the overall score.

REFERENCES

[1] Man Lung Yiu; Hua Lu; Mamoulis, N.; Vaitis, M.; , "Ranking Spatial Data by Quality Preferences", IEEE Transactions on Knowledge and Data Engineering, vol.23, no.3, pp.433-446, March 2011

[2] J.B. Rocha-Junior , A.Vlachou , C.Doulkeridis , K.Nørvåg , Efficient processing of top-k spatial preference queries , Journal Proceedings of the VLDB Endowment ,Volume 4 Issue 2, November 2010

[3] Hauke J., Kossowski T., Comparison of values of Pearson's and Spearman's correlation coefficient on the same sets of data. Quaestiones Geographicae 30(2), Bogucki Wy-dawnictwo Naukowe,

[4] 4. Barcelona Field Studies Centre S.L. Spearman's Rank Correlation Coefficient http://geographyfieldwork.com/SpearmansRank.htm

[5] https://developers.google.com/maps/

[6] http://developer.yahoo.com/maps/

[7] C. Spearman, The Proof and Measurement of Association between Two Things, *The American Journal of Psychology*, Vol. 15, No. 1 (Jan., 1904), pp. 72-101

Identification of Opinions in Arabic Texts using Ontologies

Farek Lazhar and **Tlili-Guiassa Yamina**

Abstract. A powerful tool to track opinions in forums, blogs, e-business sites, etc., has become essential for companies, politicians as well as for customers, and that because of the huge amount of texts available which make the manual exploration more and more difficult and useless. In this paper, we present our approach of identification of opinions based on an ontological exploration of texts. This approach aims to study the role of domain ontologies and their contributions in the identification phase. In our approach, domain ontology and sentiments lexicon are needed as pre-requirements.

1 INTRODUCTION

The views available on the Internet have a significant impact on users, for example, if users have already researched opinions on a product, they are willing to pay more for a product whose opinion is more favorable than another, and the product will be more marketed than another whose opinion is less favorable [14].

Companies, politicians, and customers need a powerful tool to track opinions, sentiments, judgments, and beliefs that people can express in blogs, comments, or in the form of texts, toward a product, a service, a person or an organization, etc. [13].

In opinion mining area, the use of expressions as a "bag of sentiment words" to detect the semantic orientation of the overall content of a text needs to give values to those expressions as positive, negative or neutral towards a given topic [10].

Generally, research works in this area can be grouped into three main categories:

- Development of linguistic and cognitive models for opinion mining where all approaches based on dictionary or corpus are used automatically or semi-automatically to extract opinions based on the semantic orientations of words and phrases [2];
- Opinions extraction from texts, where all the local opinions are aggregated to determine the overall orientation of a text [1],[2],[6];
- Features based opinion mining, where all the opinions expressed towards the characteristics of a product or an object are extracted and summarized [5], [8], [9].

This article focuses on identification and classification of opinions in Arabic texts, which aims to calculate the semantic orientation of the entire content of a text as positive or negative toward a subject or an object from the subjective expressions carrying the semantic orientations of the different features, but the key questions that we should ask are:

- How to get this set of features?
- What features are related to each other?
- What model of knowledge representation to be used to produce an understandable summary for the studied domain?

To answer these questions, we propose in this paper to study the role of ontologies used in opinion mining, and more specifically, our goal is to study how domain ontology can be used to:

- Structure the features;
- Extract explicit and implicit features from the texts;
- Produce summaries based on reviews and user comments.

The paper is organized as follows: We present in Section 2, state of the art of the main approaches used in the field and the motivations of our work. We present in the next section, our approach and the general architecture of opinions identification process.

2 STATE OF THE ART

1.1 Related Work

Overall, two main types of work are distinguished, those that are based on simple features extraction from the texts, and those who organize features into a hierarchy using taxonomies or ontologies. The extraction process mainly concerns explicit features. We can distinguish two main families:

- **Opinion Mining without Knowledge Representation Models**

All approaches that do not use knowledge representation models are based on the use of algorithms to discover the different characteristics of a product or an object. Only the expressions of opinions (adjectival and adverbial) are extracted, then a summary is produced to show for each characteristic, the positive and the negative opinions and the total number of these categories [2], [8].

The main limitation of these approaches is that there is a large number of extracted features and a lack of organization. In addition, similar concepts are not grouped (for example, in some domains, the words "موعد" and "لقاء" witch have the same meaning "appointment"), and possible relationships between the features of an object are not recognized (example: "قهوة" "coffee" is a specific term of "شرب" "drink"). Thus, analysis of polarity (positive, negative or neutral) of the text is done by assigning the dominant polarity of opinion words, regardless of the polarities associated with each feature individually [10].

- **Opinion Mining with Knowledge Representation Models**

The family itself can be divided into two subfamilies:

(a) Use of Taxonomies

This kind of approaches does not seek a list of features, but rather a hierarchical organized list by the use of taxonomies. We recall that a taxonomy is a list of terms organized hierarchically through a sort of "is a kind of". In [5] the author use predefined taxonomies and semantic similarity measures to automatically extract the features and calculate the distances between concepts.

Generally, the use of taxonomies is coupled with a classification technique; the sentences corresponding to the leaves of the taxonomy are extracted. At the end of the process, a summary that can be more or less detailed is produced.

(b) Use of Ontologies

These approaches aim to organize the features using elaborated representation models. Unlike taxonomies, ontology is not restricted to a hierarchical relationship between concepts, but can describe other types of paradigmatic relations such as synonymy, or more complex relationships such as relations of composition or spatial relationships.

Generally, the extracted features correspond exclusively to terms contained in the ontology. The feature extraction phase is guided by a domain ontology, built manually [11], or semi-automatically [7], [9], which is then enriched by a process of automatic extraction of terms, corresponding to new features identification.

Similar features are grouped together using semantic similarity measures.

Ontologies have also been used to support polarity mining. For example, in [4], the authors manually built an ontology for movie reviews and incorporated it in the polarity classification task which substantially improved the performance of their approach.

1.2 Ontology Based Opinion Mining

In [13], the use of a hierarchy of features improves the performance of features based identification systems. However, works using domain ontologies exploit the ontology as a taxonomy using only "is a" relations between concepts. They do not really use all data stored in an ontology, such as the lexical components and other types of relationships. We believe that we can get several advantages in the domain of opinion mining by the full use of domain ontology capabilities:

- Structuring of features: Ontologies are tools that provide a lot of semantic information. They help to define concepts, relationships, and entities that describe a domain with an unlimited number of terms;
- Extraction of features: Relationship between concepts and lexical information can be used to extract explicit and implicit features.

3 OUR APPROACH

1.3 Description

For each studied domain, our approach requires three basic elements:

- A domain ontology O, where each concept and each property is associated to a set of labels that correspond to their semantics;
- A lexical resource L of opinion expressions;
- A set of texts T as comments and views.

Based on the conceptual model described in [10], and on the definition described in[3] witch define an elementary discourse unit (EDU) as a clause containing at least an elementary opinion unit (EOU) or a sequence of clauses that address a rhetorical relation to a segment expressing an opinion. Note that an EOU is an explicit opinion expression composed of an explicit noun, an adjective or a verb with its possible modifiers (negation and adverbs).

In a review, the opinion holder comments a set of features of an object or a product using opinion expressions. Each feature corresponds to a concept or a property in the ontology O.

For each extracted EDU, the system:

- Extracts EOUs using an approach based on rules;
- Extracts features that correspond to the process of terms extraction using the domain ontology;
- Associates, for each feature within the EDU, the set of opinion expressions;

We detail below, these steps:

(a) Extraction of Elementary Opinion Units: Nouns, adjectives or verbs may be associated with certain modifiers such as words of negation and adverbs. For example, "ممتاز", "excellent", "ليس جيدا", "not good" are EOUs.

For example in the following comment, the EDUs are between square brackets, the EOUs are underlined, and the characteristics of the object are in bold. There is an inverse relationship between the EDUa and the EDUb, representing the review expressed in the EDUd.

[يوم أمس ، اشتريت **جهاز هاتف**]a
[حتى إذا كان **الهاتف** ممتازا]b
[فان **التصميم** بسيط جدا]c
[الشيء المخيب للآمال في هذه **العلامة**]d

[Yesterday, I purchased a **phone**] $_{a.}$ [Even if the **phone** is excellent]$_b$, [the **design** is very basic]$_c$, [which is disappointing in this **mark**]$_d$.

Figure 1. Example showing EOUs Extraction

(b) Features Extraction

This step aims to extract for the comment all the labels of the ontology. As each concept is an explicit feature, we simply project the lexical components of the ontology on the text to obtain, for each EDU, all the features. To extract the implicit features, ontology properties are used. We recall that these properties are to define the relationships between concepts of the ontology. For example, the property "يسوق","drive" links the concepts "سائق","conductor" and "سيارة","car".

(c) Linking Opinions Expressions with Extracted Features

In this step, extracted opinions expressions in step (a) have to be linked to the features extracted in step (b), i.e. we should associate with each EDUi the set of pairs (fi, OEi). During this step, we distinguish the following cases:

- **Known Opinionated Features and Known Opinions Expressions:** In this case, opinionated features match to the used opinions expressions. For example, if our lexicon contains the concept "طبيعة", "nature", and sentiments lexicon contains the word "خلاب", "amazing", from the EDU "طبيعة خلابة", "amazing nature", it is easy to extract the couple (خلابة, طبيعة), (nature, amazing) from the text.
- **Known Opinionated Features and Unknown Opinion Expressions:** Expressions, as in the EDU "نتائج مقبولة", "acceptable results", where the opinion word "مقبول", "acceptable" was not extracted in step (a) (see section 3.1). In this case, the lexicon of opinions can be automatically updated with the recovered opinion word.
- **Unknown Opinionated Features and Unknown Opinion Expressions:** As in the EDU "غابة مطرية رائعة", "wonderful rainforest" where the feature "مطرية","rainforest" has not been extracted in step (b) (see section 3.1), in this case, the domain ontology can be updated by adding a new concept or a new property in the right place.
- **Opinion Expressions Only:** As in the EDU "بطيء", "It's slow". This kind of EDU expresses an implicit feature. In this case, we use the ontology properties to retrieve the associated concept in the ontology.
- **Features Only:** An EDU with features alone can also be an indicator of the presence of an implicit opinion expression towards the feature as in " الحديقة أصبحت ملجأ للمنحرفين", "the park became a haven for perverts", witch express a negative opinion towards "الحديقة", "the park".

1.4 Architecture of our Approach

In this section, we present the general architecture of our approach and the different modules constituting our system:

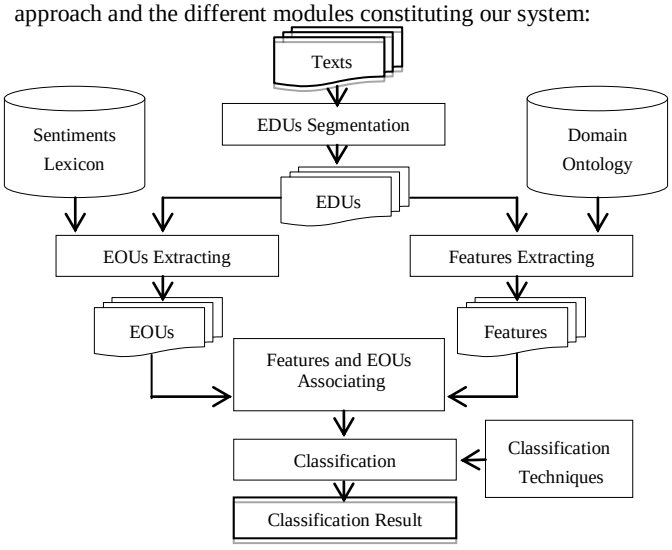


Figure 2. General architecture of our approach

As indicated in the last figure, our system contains the following modules:

1. **Texts EDUs Segmentation:** Generally, extraction of elementary discourse units (EDUs), depends on the use of delimiters such as "." , ",", "?" "!";

2. **EOUs Extracting:** Elementary opinions units EOUs and semantic orientations are usually extracted using a lexicon of emotions specific to domain of study;

3. **Features Extraction:** Features can be extracted by a simple projection of the ontology on the elementary discourse units (EDUs);

4. **Associating UEOs to Features:** Each extracted feature should be associated to one or more elementary opinions units in order to extract its semantic orientation;

5. **Classification:** The last phase of our work is to classify the identified opinions into positive or negative classes using supervised classification techniques.

4 CONCLUSION

In this paper we presented our approach based on an ontological exploration of Arabic texts. Our method is promising because the use of ontologies improves the extraction of features and facilitates the association between opinions expressions and opinionated features of the object. On the one hand, domain ontology is useful within its list of concepts which carry much semantic data in the system. The use of ontology concepts labels can recognize terms that refers to the same concepts and provides a hierarchy between these concepts. On the other hand, ontology is useful to its list of properties between concepts that can recognize the opinions expressed on the implicit features.

REFERENCES

[1] Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan, 'Thumbs up? Sentiment Classification using Machine Learning Techniques'. *Proceedings of EMNLP*, (2002)

[2] Turney, Peter D., and Michael L. Littman, 'Unsupervised Learning of Semantic Orientation from a Hundred-Billion-Word Corpus'. *National Research Council, Institute for Information Technology, Technical Report ERB-1094. (NRC#44929)*, (2002)

[3] Asher Nicholas and Lascarides Alex, 'Logics of Conversation'. *Cambridge University Press*, (2003)

[4] Pimwadee Chaovalit, Lina Zhou, 'Movie Review Mining: a Comparison between Supervised and Unsupervised Classification Approaches', *HICSS*, (2005)

[5] Carenini, Giuseppe, Raymond T. Ng, and Ed Zwart, 'Extracting Knowledge from Evaluative Text', *In Proceedings of the 3rd international conference on Knowledge capture*, (2005)

[6] Kim, Soo-Min, and Eduard Hovy, 'Extracting Opinions, Opinion Holders, and Topics Expressed in Online News Media Text', *In Proceedings of ACL/COLING Workshop on Sentiment and Subjectivity in Text*, Sydney, Australia, (2006)

[7] Feiguina, Olga, 'Résumé automatique des commentaires de Consommateurs'. *Mémoire présenté à la Faculté des études supérieures en vue de l'obtention du grade de M.Sc. en informatique, Département d'informatique et de recherche opérationnelle, Université de Montréal*, (2006)

[8] Hu et al. 'Mining and Summarizing Customer Reviews', *In Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining*, (2008)

[9] Cheng, Xiwen, and Feiyu Xu. 'Fine-grained Opinion Topic and Polarity Identification', *In Proceedings of the Sixth International Language Resources and Evaluation (LREC' 08)*, Marrakech, Morocco, (2008)

[10] Farek Lazhar et al., 'Identification d'opinions dans les textes arabes', *IC*, (2009)

[11] Zhao, Lili, and Chunping Li, 'Ontology Based Opinion Mining for Movie Reviews', *In Proceedings of the 3rd International Conference on Knowledge Science, Engineering and Management*, (2009)

[12] Asher, Nicholas, Farah Benamara, and Yvette Y. Mathieu. 'Appraisal of Opinion Expressions in Discourse, *Lingvisticæ Investigationes, John Benjamins Publishing Company, Amsterdam, Vol. 32:2*, (2009)

[13] Anaïs Cadilhac et al., 'Ontolexical resources for feature based opinion mining: a case study', Beijing, (2010)

[14] Gillot Sébastien, 'Fouille d'opinions, Rapport de stage', (2010)

[15] Alexander Pak et al., 'Classification en polarité de sentiments avec une représentation textuelle à base de sous-graphes d'arbres de dépendances', *TALN 2011,Montpellier, 27 juin – 1er juillet*, (2011)