

# Workshop on Ubiquitous Data Mining



<http://www.liaad.up.pt/udm/>  
in conjunction with the *19th European Conference on  
Artificial Intelligence (ECAI2010)*, in Lisbon, Portugal,  
August 16 - 20, 2010.

Edited by:

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



# Preface

Currently, two major technological evolutions modify our relationship with our environment:

- **Widely available and cheap computer power.** Simple objects that surround us are gaining sensors, computational power, and actuators, and are changing from static, into adaptive and reactive systems.
- **The explosion of communication networks of all kinds** offers new possibilities for the development and self-organization of communities.

The new characteristics of data reflect a World in Movement:

- **Time and space.** The objects of analysis exist in time and space. Often they are able to move.
- **Dynamic environment.** These objects exist in a dynamic and unstable environment, evolving incrementally over time.
- **Information processing capability.** The objects are endowed with information processing capabilities.
- **Locality.** The objects never see the global picture - they know only their local spatio-temporal environment.
- **Real-Time.** The models have to evolve incrementally in correspondence with the evolving environment.
- **Distributed.** The object will be able to exchange information with other objects, thus forming a truly distributed environment.

**Ubiquitous Data Mining (UDM)** uses Data Mining techniques to extract useful knowledge from data with these characteristics.

The goal of this workshop is to convene researchers (from both academia and industry) who deal with techniques such as: decision rules, decision trees, association rules, clustering, filtering, learning classifier systems, neural networks, support vector machines, preprocessing, post processing, feature selection, visualization techniques, etc. for UDM and related themes.

This is the first workshop in the topic. We received 14 submissions that were evaluated by 2 members of the Program Committee. The PC recommended accepting 11 papers. We have a diverse set of papers focusing from applications in medical domains to more theoretical aspects of computing statistics in data streams. All papers deal with different aspects of evolving data and/or distributed data.

João Gama, Paulo Cortez, Nuno Marques, Manuel F. Santos  
Program Chairs

# Organization

## **Program Chairs:**

João Gama, Paulo Cortez, Nuno Marques, Manuel F. Santos

## **Publicity Chairs:**

Pedro P. Rodrigues, 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
- Andre Carvalho, University of Sao Paulo (USP), Brazil
- Carlos A. Ferreira, Institute of Engineering of Porto, Portugal
- Elena Ikononovska, University St. Cyril & Methodius, Macedonia
- Ernestina Menasalvas, Technical University of Madrid, Spain
- Florent Massegia, INRIA, France
- João Gama, University of Porto, Portugal
- Manuel F. Santos, University of Minho, Portugal
- Mark Last, Ben Gurion University, Israel
- Matjaž Gams, Jozef Stefan Institute, Slovenia
- Michael May, Fraunhofer Bonn, Germany
- Mohamed Gaber, University of Portsmouth, 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
- Raquel Sebastião, University of Porto, Portugal
- Rasmus Pederson, Copenhagen Business School, Denmark
- Ricard Gavaldà, University of Barcelona, Spain
- Shonali Krishnaswamy, Monash University, Australia

# Table of Contents

|                                                                                                                 |    |
|-----------------------------------------------------------------------------------------------------------------|----|
| Understanding Clusters' Evolution .....                                                                         | 1  |
| <i>Márcia Oliveira, João Gama</i>                                                                               |    |
| Context-Aware Stream Ensemble for Ubiquitous Environments .....                                                 | 7  |
| <i>João Gomes, Pedro Sousa, Ernestina Menasalvas</i>                                                            |    |
| A Data Clustering Approach to Energy Conservation in Wireless .....<br>Sensor Networks                          | 13 |
| <i>Alexander Sagen, Cyril Labbé, Mohamed Gaber, Shonali Krishnaswamy,<br/>Agustinus Borgy Waluyo, Seng Loke</i> |    |
| Size and Speed Related Characteristic of Concept Drift .....                                                    | 19 |
| <i>Petr Kosina, João Gama, Raquel Sebastião</i>                                                                 |    |
| Towards Symbiotic Spam E-mail Filtering .....                                                                   | 25 |
| <i>Clotilde Lopes, Paulo Cortez, Pedro Sousa</i>                                                                |    |
| Memoryless Fading Windows in Ubiquitous Settings .....                                                          | 27 |
| <i>Pedro Pereira Rodrigues, João Gama, Raquel Sebastião</i>                                                     |    |
| GridClass: Strategies for Global Vs Centralized Model Construction in ...<br>Grid Data Mining                   | 33 |
| <i>Manuel Santos, Mathew Wesley, Henrique Santos</i>                                                            |    |
| Real-time Alarm Model Adaptation Based on User Feedback .....                                                   | 39 |
| <i>Violeta Mirchevska, Bostjan Kaluza, Mitja Lustrek, Matjaž Gams</i>                                           |    |
| Ubiquitous Data-Mining with Self-Organizing Maps .....                                                          | 45 |
| <i>Bruno Silva, Nuno Marques</i>                                                                                |    |
| Distributed and real time Data Mining in the Intensive Care Unit .....                                          | 51 |
| <i>Marta Vilas-Boas, Pedro Gago, Filipe Portela, Fernando Rua, Álvaro<br/>Silva, Manuel Filipe Santos</i>       |    |
| Learning Abnormal Gait Patterns of Elderly from Motion Capture .....                                            | 57 |
| System                                                                                                          |    |
| <i>Bogdan Pogorelc, Matjaž Gams</i>                                                                             |    |



# Understanding Clusters' Evolution

Márcia Oliveira and João Gama

**Abstract.** In this paper we are interested in the study of evolving data, whose distribution may be non-stationary. We address the problem of monitoring the evolution of clusters over time. We adopt two main strategies for cluster characterization - representation by enumeration and representation by comprehension -, and propose the MEC (Monitoring the Evolution of Clusters) framework, which was developed along the lines of Change Mining paradigm. MEC includes a taxonomy of various types of clusters' transitions, a tracking mechanism that depends on the cluster representation, and a transition detection algorithm. Our tracking mechanism can be subdivided in two methods devised to monitor and detect clusters' transitions: one based on graph transitions, and another based on clusters' overlap in the feature space. These are the most relevant contributions of this work. We experimentally evaluate our framework with a real world case study.

## 1 INTRODUCTION

The rapid progress made in science and technology has contributed to the emergence of a volatile and fast pace evolving world, which demands new perspectives in knowledge discovery upon data, such as time-oriented perspectives. The paradigm of *change mining* arises as a consequence of this evolution and encompasses Data Mining mechanisms that monitor models and patterns over time, compare them, detect and describe changes, and quantify them on their interestingness [5]. The challenge of *change mining* lays not only in the adaptation of models to changes in data distribution but also in the understanding of changes themselves. In this paper we propose MEC framework, which is built along these lines, to tackle the problem of monitoring clusters' transitions over time. We assume that clusters can be represented in two different ways - by enumeration and by comprehension - and we adopt two main strategies to monitor and classify changes experienced by clusters, to each possible representation. Thus, our framework encompasses a taxonomy of various types of cluster transitions, that may be external or internal, a tracking mechanism that depends on the cluster representation, and a transition detection algorithm.

The monitoring of clusters' structures is very important in many real world applications, since the detected transitions may correlate to some important or critical events in real applications or unveil the emergence of new ones. For these reasons, this study can benefit several areas, such as Marketing, Fraud Detection, Economy and Sociology. For example, the study of the evolution of customer's segments allow the detection of shifts in preferences and consumer habits, which can sustain the forecast of trends and consequent re-definition of Marketing's strategies and policies. This can contribute to the creation, or reinforcement, of the company's competitive advantages. Traditional Data Mining is not able to help companies

achieve these goals since it relies on static data and does not take into account its evolving nature.

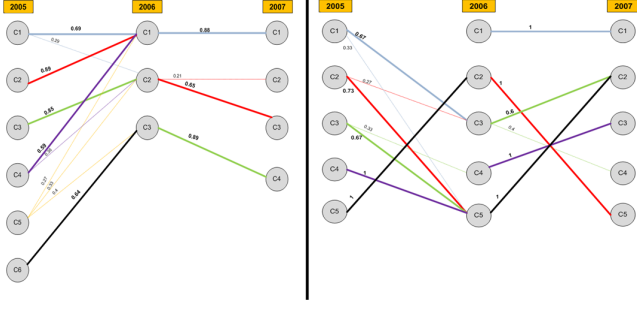
This paper is organized as follows. In Section 2 we provide a brief overview of the current state of the art. In Section 3 we distinguish two types of cluster representation and highlight their advantages and drawbacks. In Section 4 we formally introduce our MEC framework. In this section we present our taxonomy for clusters' transitions and explain the foundations of our tracking mechanism, for each type of representation. In Section 5 we show and discuss a real world case study. Section 6 concludes our study.

## 2 RELATED WORK

Despite the extensive study of the clustering problem, there is not much work conducted in the monitoring of clusters' transitions. In this context of change, the research endeavor has been mainly directed to the adaptation of clusters to changed populations. However, the dynamic nature of most datasets encouraged new directions in research. This effort is clearly present in the areas of Machine Learning and Data Analysis.

There are several algorithms that directly or indirectly aim to capture and, especially, understand the dynamic nature of these datasets, particularly susceptible to the occurrence of changes in the underlying structure. The last decade has been especially profuse in the design of transition detection algorithms. Based on recent literature, it was possible to deduce a preliminary classification for algorithms built in this context. In general, there are algorithms designed to operate in relatively static (snapshots) or highly dynamic (Data Streams) environments. The first may be focused on the transitions experienced by generic patterns [4], clusters [9, 6, 10] or association rules [3]. In the context of Data Streams, approaches focusing unclassified data are quite common (here we include clustering algorithms for mobile objects or spatio-temporal objects). These approaches [2, 1] elect clusters as its main data structure and are concerned with the efficiency and scalability of algorithms.

Based on our knowledge, currently there are two research works that address a problem related to ours [8, 9]. The framework proposed in [8] models the evolution of clusters by a weighted bipartite graph. MONIC framework [9] also proposes a cluster transition model to track cluster changes, supporting cluster comparisons across the time axis. MONIC uses a data ageing function that assigns lower weights to older observations and computes the overlap of clusters to capture its evolution. However, both frameworks only apply to clusters represented by enumeration, contrary to MEC, which is not restricted by the adopted method to represent clusters. MEC proposes a more complete framework to study the evolution of clusters which applicability to real world problems is much wider, since it can also consider clusters represented using summary statistics. This is the major con-



**Figure 1.** Bipartite graphs, corresponding to timestamps 2005, 2006 and 2007, with the thickness of the edges indicating weights that are equal or greater to Survival threshold; the edges whose weights are below the Split threshold were removed from the graphs, due to their insignificance. The left-hand side graph corresponds to the agglomerative hierarchical clustering and the right-hand side to the K-means clustering.

tribution of our work, compared to the referred ones.

### 3 CLUSTER CHARACTERIZATION

MEC framework assumes that clusters can be represented using two main strategies or representation schemes: **representation by enumeration** and **representation by comprehension** (can also be called extensional and intensional representation, using the mathematical nomenclature). In **representation by enumeration**, a cluster is characterized by its elements, ie, by the observations that were assigned to it by a given clustering algorithm (see Definition 1).

**Definition 1 - REPRESENTATION BY ENUMERATION:**

Let  $\vec{x}_i$ , ( $i = 1, \dots, N$ ) be the  $i$ th observation defined as a vector of real numbers in a  $d$ -dimensional space  $\vec{x}_i = (x_{i,1}, x_{i,2}, x_{i,3}, \dots, x_{i,d})$ . A possible temporal representation of a cluster is defined as follows:

$$C_j(t) = \{\vec{x}_1, \dots, \vec{x}_m\}$$

where  $m$  is the number of observations assigned to cluster  $C_j(t)$ ,  $j = (1, \dots, k)$ ,  $k$  is the number of clusters and  $t = (1, \dots, T)$ , where  $T$  corresponds to the last analyzed timestamp.

This type of representation does not involve information loss and enables the monitoring of each object over time, which contributes to the achievement of more reliable and accurate transition results. Though, it is not always possible to define clusters using this representation method, e.g. due to storage demands or privacy issues.

Alternatively, a cluster can be characterized through summary data, ie, using statistics that summarize its internal characteristics. This is the idea behind **representation by comprehension** (see Definition 2).

**Definition 2 - REPRESENTATION BY COMPREHENSION:**

According to this representation a cluster  $C_j$  is a temporal object characterized by the following statistics:

$$C_j = \{ID, t, m, sup, r, \rho, \vec{c}\}$$

where  $ID$  is the unique identifier of  $C_j$  ( $j = 1, \dots, k$ ),  $t$  is the time point where the cluster first appears ( $t = 1, \dots, T$ ),  $m$  is the number

of observations assigned to  $C_j$ ,  $sup$  is the cluster's support, used to assess the relative importance of the cluster,  $r$  is the cluster's radius,  $\rho$  represents the density of cluster and  $\vec{c}$  represents cluster's centroid.

To define this type of cluster representation we considered a centrality measure, a dispersion measure and a density measure. Due to simplicity issues, we adopted the cluster's centroid as our measure of centrality, the cluster's radius as our measure of dispersion and we use a definition of density that assumes spherical  $d$ -dimensional objects. Other measures could have been used, e.g., Mahalanobis metrics, but this would lead to more complex computations. The drawback of these measures is that they can only describe spherical or ball-shaped clusters<sup>1</sup>.

This kind of compact representations are very appealing to several real world applications, especially to those whose information are restricted due to privacy issues. However, it implies information loss, which may compromise the accuracy of *mapping* process.

To implement the transitions detection algorithm we have considered the definitions of **radius** and **centroid** presented in [11]. Thereby, in this work, the **radius** of a cluster is the average distance from member points to the centroid.

### 4 MEC FRAMEWORK

We developed the MEC framework in order to monitor the evolution of clusters' structures (also referred to as *clustering*). In this context, the concept of evolution refers to transitions undergone by clusters during time interval under observation  $[t_i, t_{i+\delta}]$  ( $\delta = 1, \dots, T - i$ ). Since there are, at least, two strategies to represent clusters, we designed a flexible tracking mechanism that is able to efficiently detect the transitions experienced by clusters, regardless of how they are represented. Therefore, our tracking mechanism can be subdivided in two different methods, each one adapted to a distinct clusters' representation. In the next subsections we present our taxonomy of transitions and the two developed methods.

#### 4.1 Taxonomy of Transitions

In the literature there are at least eight taxonomic schemes for the classification of transitions in clusters, patterns or concepts that evolve over time [6, 2, 1, 7, 9, 10]. To capture the changes likely to occur in clusters' structures we considered the following taxonomy: **Birth**, **Death**, **Split**, **Merge** and **Survival** of clusters. These transitions are **external**, as they relate to changes in the whole *clustering*. The key concept for the detection and evaluation of these transitions is the concept of *mapping*, which can be defined as the process of discovering the matches between clusters obtained at timepoint  $t_i$  and clusters obtained at a later timepoint  $t_{i+\delta}$ , in case they still exist. On the other hand, it is also possible to categorize **internal** transitions, ie, changes concerning the contents of each isolated cluster. For this purpose, we considered size and compactness transitions [9]. Size transitions encompasses **Expansion** or **Shrinkage**, depending on the increase or decrease of the surviving cluster's cardinality. Compactness transitions include **Compression** or **Dispersal**, which depends on the increase or decrease of the surviving cluster's density. These type of transitions can be easily traced through the monitoring of summary data, such as the cardinality and the density of the surviving clusters.

<sup>1</sup> In this work, we restrict to numerical features. For non-numerical features we will need appropriate measures for centrality and dispersion.

## 4.2 Method for Clusters Represented by Enumeration

In this method, the *mapping* process explores the concept of conditional probability and is restricted by a pre-defined threshold - Survival threshold  $\tau$  -, which assumes the minimum of  $\tau = 0.5$ . The use of conditional probabilities requires structurally identical datasets, ie, datasets composed by observations of the same object, in each timepoint under analysis. These conditional probabilities are computed for every pair of possible connections between clusters obtained at different timepoints and they represent the edge's weights in a bipartite graph. The use of bipartite graphs is related to its usefulness in modeling matching problems and the fact that graph based representations are visually appealing. The foundations of our transitions detection algorithm are based in this idea, which can be defined as follows (Definition 3).

### Definition 3 - WEIGHTED BIPARTITE GRAPHS:

Given the clusterings  $\xi_i, \xi_{i+\delta}$ , obtained at  $t_i, t_{i+\delta}$ , a graph  $G = (U, V, E)$  can be constructed, where  $U$  represents the first subset of vertices (clusters of  $t_i$ ),  $V$  represents the second subset of vertices (clusters of  $t_{i+\delta}$ ), and  $E$  denotes a set of weighted edges between any pair of clusters belonging to  $\xi_i$  and  $\xi_{i+\delta}$ . Formally, the weight assigned to the edge connecting clusters  $C_m(t_i)$  and  $C_u(t_{i+\delta})$  ( $m = (1, \dots, k_{t_i})$  and  $u = (1, \dots, k_{t_{i+\delta}})$ , where  $k_{t_i}$  and  $k_{t_{i+\delta}}$  are the number of clusters returned by a given clustering algorithm in timepoints  $t_i$  and  $t_{i+\delta}$ , respectively) are estimated in accordance with the conditional probability (Equation 1):

$$\text{weight}(C_m(t_i), C_u(t_{i+\delta})) = P(X \in C_u(t_{i+\delta}) | X \in C_m(t_i)) = \frac{\sum P(x \in C_m(t_i) \cap C_u(t_{i+\delta}))}{\sum P(x \in C_m(t_i))}$$

where  $X$  is the set of observations assigned to cluster  $C_m(t_i)$  and  $P(X \in C_u(t_{i+\delta}) | X \in C_m(t_i))$  represents the probability of  $X$  belonging to cluster  $C_u$  from  $t_{i+\delta}$  knowing that  $X$  belongs to cluster  $C_m$  obtained at a previous timestamp  $t_i$ .

To detect changes, we formally define the transitions that a cluster  $C \in \xi_i$  can experience, with respect to  $\xi_{i+\delta}$ . It was introduced a new threshold to help the definition of these transitions: the split threshold  $\lambda$ . This formal design is based on MONIC's framework's external transitions [9] and is depicted in Table 1. The drawback of this method boils down to the fact that monitoring based on graph transitions only allow the detection of external transitions.

## 4.3 Method for Clusters Represented by Comprehension

For compact representations of clusters, we developed a method to assess the similarity between clusters through the discovery of an intersection region in the feature space, formed by pairs of clusters from different timestamps. This procedure allows us to know if cluster  $C_m(t_i)$  in clustering  $\xi_i$  is the match of cluster  $C_u(t_{i+\delta})$  in the later clustering  $\xi_{i+\delta}$ . To accomplish this task, we compute the Euclidean distance between clusters' centroids (see Definition 4) for every pair of clusters belonging to clusterings generated at different time points.

### Definition 4 - CENTROID EUCLIDEAN DISTANCE:

Let  $C_m$  and  $C_u$  be two clusters obtained at  $t_i$  and  $t_{i+\delta}$ , respectively,

and let  $\vec{c}_m = (c_{m,1}, c_{m,2}, \dots, c_{m,d})$  and  $\vec{c}_u = (c_{u,1}, c_{u,2}, \dots, c_{u,d})$  be the corresponding centroids. The Centroid Euclidean Distance is used to measure the similarity between  $C_m(t_i)$  and  $C_u(t_{i+\delta})$ , and is defined as:

$$d(\vec{c}_m, \vec{c}_u) = \sqrt{\sum_{j=1}^d (c_{m,j} - c_{u,j})^2} \quad (1)$$

Posteriorly, to assess the existence of overlap (or intersection) between clusters, we compare the centroids' distance with the sum of the radius of clusters. If the centroids' distance is equal or greater than the sum of clusters' radius (2), then we can deduce that clusters don't intersect and, therefore, the cluster of  $t_i$  cannot be the match of the cluster of  $t_{i+\delta}$ . Otherwise (3), we can assume that these pair of clusters intersect and conclude that they form a match.

$$d(C_m(t_i), C_u(t_{i+\delta})) \geq r_{C_m(t_i)} + r_{C_u(t_{i+\delta})} \quad (2)$$

$$d(C_m(t_i), C_u(t_{i+\delta})) < r_{C_m(t_i)} + r_{C_u(t_{i+\delta})} \quad (3)$$

To detect external and internal transitions and discover the changes undergone by clusters, we formally define the transitions, using the presented concepts. The formal design is depicted in Table 2 and Table 3. Note that internal transitions are only monitored for surviving clusters and, also, that the usage of Euclidean distance recommends the standardization of the attributes.

The implementation of the transitions detection algorithm, in software R 2.10.0, was supported by the definitions presented in Table 1, Table 2 and Table 3.

## 5 REAL WORLD CASE STUDY

In order to show the feasibility and application of the proposed framework and glean insights about the evolution of clusters, we experimented MEC in three real datasets extracted from Banco de Portugal's Central Balance-Sheet Database (CBSD). Each dataset corresponds to a year (2005, 2006 and 2007) and consists of 439 observations characterized by 10 continuous attributes. The objects represent activity sectors, according to the higher granularity level of CEA (Portuguese Classification of Economic Activities), and the attributes are financial and economic aggregated indicators (e.g. net income, investment rate and labor productivity). It should be noted that economic activity sectors and performance indicators are exactly the same, for all periods under analysis, which is a requirement for the application of MEC framework. The attributes were normalized using z-scores, due to significant differences on scale and dispersion.

### 5.1 REPRESENTATION BY ENUMERATION

To discover the clusters (input of our study) from the CBSD datasets, we conducted experiments using two different algorithms for clustering: the agglomerative hierarchical algorithm using Ward's index, and K-means partitional algorithm. In both algorithms, the determination of the critical clustering structure, through the identification of the best number of clusters (best  $k$ ), was supported by

**Table 1.** Formal definition of the external transitions of a cluster represented by enumeration

| Transitions' Taxonomy | Notation                                                                           | Formal Definition                                                                                                                                                                                    |
|-----------------------|------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Cluster's Birth       | $\emptyset \rightarrow C_u(t_{i+\delta})$                                          | $0 < weight(C_m(t_i), C_u(t_{i+\delta})) < \tau \forall_m$                                                                                                                                           |
| Cluster's Death       | $C_m(t_i) \rightarrow \emptyset$                                                   | $weight(C_m(t_i), C_u(t_{i+\delta})) < \lambda \forall_u$                                                                                                                                            |
| Cluster's Split       | $C_m(t_i) \xrightarrow{\subseteq} \{C_1(t_{i+\delta}), \dots, C_r(t_{i+\delta})\}$ | $(\exists_u \exists_v : weight(C_m(t_i), C_u(t_{i+\delta})) \geq \lambda \wedge weight(C_m(t_i), C_v(t_{i+\delta})) \geq \lambda) \wedge \sum_{u=1}^r weight(C_m(t_i), C_u(t_{i+\delta})) \geq \tau$ |
| Cluster's Merge       | $\{C_1(t_i), \dots, C_p(t_i)\} \xrightarrow{\subseteq} C_u(t_{i+\delta})$          | $(weight(C_m(t_i), C_u(t_{i+\delta})) \geq \tau) \wedge \exists C_p \in \xi_i \setminus \{C_m\} : weight(C_p(t_i), C_u(t_{i+\delta})) \geq \tau$                                                     |
| Cluster's Survival    | $C_m(t_i) \rightarrow C_u(t_{i+\delta})$                                           | $(weight(C_m(t_i), C_u(t_{i+\delta})) \geq \tau) \wedge \nexists C_p \in \xi_i \setminus \{C_m\} : weight(C_p(t_i), C_u(t_{i+\delta})) \geq \tau$                                                    |

**Table 2.** Formal definition of the external transitions of a cluster represented by comprehension

| Transitions' Taxonomy | Notation                                                                           | Formal Definition                                                                                                                                                                                         |
|-----------------------|------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Cluster's Birth       | $\emptyset \rightarrow C_u(t_{i+\delta})$                                          | $d(C_m(t_i), C_u(t_{i+\delta})) \geq r_{C_m(t_i)} + r_{C_u(t_{i+\delta})} \forall_m$                                                                                                                      |
| Cluster's Death       | $C_m(t_i) \rightarrow \emptyset$                                                   | $d(C_m(t_i), C_u(t_{i+\delta})) \geq r_{C_m(t_i)} + r_{C_u(t_{i+\delta})} \forall_u$                                                                                                                      |
| Cluster's Split       | $C_m(t_i) \xrightarrow{\subseteq} \{C_1(t_{i+\delta}), \dots, C_r(t_{i+\delta})\}$ | $(d(C_m(t_i), C_u(t_{i+\delta})) < r_{C_m(t_i)} + r_{C_u(t_{i+\delta})}) \wedge \exists C_r \in \xi_{i+\delta} \setminus \{C_u\} : d(C_m(t_i), C_r(t_{i+\delta})) < r_{C_m(t_i)} + r_{C_r(t_{i+\delta})}$ |
| Cluster's Merge       | $\{C_1(t_i), \dots, C_p(t_i)\} \xrightarrow{\subseteq} C_u(t_{i+\delta})$          | $(d(C_m(t_i), C_u(t_{i+\delta})) < r_{C_m(t_i)} + r_{C_u(t_{i+\delta})}) \wedge \exists C_p \in \xi_i \setminus \{C_m\} : d(C_p(t_i), C_u(t_{i+\delta})) < r_{C_p(t_i)} + r_{C_u(t_{i+\delta})}$          |
| Cluster's Survival    | $C_m(t_i) \rightarrow C_u(t_{i+\delta})$                                           | $(d(C_m(t_i), C_u(t_{i+\delta})) < r_{C_m(t_i)} + r_{C_u(t_{i+\delta})}) \wedge \nexists C_p \in \xi_i \setminus \{C_m\} : d(C_p(t_i), C_u(t_{i+\delta})) < r_{C_p(t_i)} + r_{C_u(t_{i+\delta})}$         |

the analysis of an internal validation measure: the silhouette width. Afterwards, we applied our transitions detection algorithm, setting a fixed  $\lambda = 0.2$  and  $\tau = 0.5$ , whose choice was guided by a thresholds' sensitivity analysis. These parameters are more relaxed and allow the detection of a wider variety of transitions. The resulting graphs, showing the transitions between clusters, are depicted in Figure 1.

On the left-hand side of Figure 1 we observe two bipartite graphs, one corresponding to transitions experienced during time interval [2005, 2006] and another one corresponding to transitions occurred during [2006, 2007]. In the former, there were two surviving clusters ( $C_3(2005) \rightarrow C_2(2006)$  and  $C_6(2005) \rightarrow C_3(2006)$ ), a merge of three clusters ( $\{C_1(2005), C_2(2005), C_4(2005)\} \xrightarrow{\subseteq} C_1(2006)$ ) and a split ( $C_5(2005) \xrightarrow{\subseteq} \{C_1(2006), C_2(2006), C_3(2006)\}$ ). In the latter, all clusters survived ( $C_1(2006) \rightarrow C_1(2007)$ ,  $C_2(2006) \rightarrow C_3(2007)$  and  $C_3(2006) \rightarrow C_4(2007)$ ) and a new one has emerged ( $\emptyset \rightarrow C_2(2007)$ ). The right-hand side of Figure 1 corresponds to K-means algorithm's bipartite graphs. Using the same strategy and the silhouette width validation measure, K-means algorithm resulted in a different partition (five clusters to all timepoints under analysis). The detected transitions also differ from the ones obtained for agglomerative hierarchical algorithm. This was expected since the number of clusters suggested for each algorithm is different. Nevertheless, transition's results are stable within the same algorithm (e.g., using different initializations of K-means). Thereby, during [2005, 2006] we detect the survival of two clusters ( $C_1(2005) \rightarrow C_3(2006)$  and  $C_5(2005) \rightarrow C_2(2006)$ ), a merge of three clusters ( $\{C_2(2005), C_3(2005), C_4(2005)\} \xrightarrow{\subseteq} C_5(2006)$ ) and two births ( $\emptyset \rightarrow C_1(2006)$  and  $\emptyset \rightarrow C_4(2006)$ ). At period [2006, 2007], four clusters survived ( $C_1(2006) \rightarrow C_1(2007)$ ,  $C_2(2006) \rightarrow C_5(2007)$  and  $C_4(2006) \rightarrow C_3(2007)$ ), two clusters merged ( $\{C_3(2006), C_5(2006)\} \xrightarrow{\subseteq} C_2(2007)$ ) and a new one has emerged ( $\emptyset \rightarrow C_4(2007)$ ).

In this experiment, the differences in terms of transitions between the hierarchical and the K-means algorithm are more evident in time interval [2006, 2007]. The merge occurred in [2005, 2006] was captured by both clustering algorithms, so we assumed this was an important transition. The inspection of the dataset suggested that these three clusters were grouped into one because the activity sectors assigned to them experienced a worsening in their economic and financial performance, which was reflected in the mitigation of their initial differences. But why did this happen? To answer this question we searched for relevant information about the topic and we found out that, in 2006, there has been a homogenization of the information process within the SIMPLEX portuguese program, which aimed to incorporate in one document (the Simplified Business Information) and a single delivery operation the information that companies are required to provide to public institutions. Moreover, the reporting of the business information became mandatory, which contributed to the increase of the coverage degree of CBSD. For this reason, the data became more reliable and complete, reflecting a more realistic image of the country, which may be the cause of the detected cluster's merge.

## 5.2 REPRESENTATION BY COMPREHENSION

To test the tracking mechanism for compact representations of clusters we use the same datasets from CBSD. We transform the original representation of data into a comprehension representation scheme, according to Definition 2. We show the results using the same clustering algorithms and briefly compare the solutions obtained for each representation scheme.

During [2005, 2006], and using the hierarchical agglomerative clustering algorithm (partition in six, three and four clusters, for each year), our method detected the merge of three groups of clusters ( $\{C_1(2005), C_2(2005)\} \xrightarrow{\subseteq} C_1(2006)$ ,  $\{C_1(2005), C_3(2005)\} \xrightarrow{\subseteq}$

**Table 3.** Formal definition of the internal transitions of a cluster represented by comprehension

| Transitions' Taxonomy | Notation                                           | Formal Definition                                  |
|-----------------------|----------------------------------------------------|----------------------------------------------------|
| Cluster's Expansion   | $C_m(t_i) \nearrow C_u(t_{i+\delta})$              | $\#C_m(t_i) < \#C_u(t_{i+\delta})$                 |
| Cluster's Shrinkage   | $C_m(t_i) \searrow C_u(t_{i+\delta})$              | $\#C_m(t_i) \geq \#C_u(t_{i+\delta})$              |
| Cluster's Compression | $C_m(t_i) \xrightarrow{\bullet} C_u(t_{i+\delta})$ | $\lambda_{C_m(t_i)} < \lambda_{C_u(t_{i+\delta})}$ |
| Cluster's Dispersal   | $C_m(t_i) \xrightarrow{*} C_u(t_{i+\delta})$       | $\lambda_{C_m(t_i)} > \lambda_{C_u(t_{i+\delta})}$ |

$C_2(2006)$  and  $\{C_5(2005), C_6(2005)\} \searrow C_3(2006)$ ), a split ( $C_1(2005) \searrow \{C_1(2006), C_2(2006)\}$ ) and the death of one cluster ( $C_4(2005) \rightarrow \emptyset$ ). During [2006, 2007] we observe the split of two groups of clusters ( $C_1(2006) \searrow \{C_1(2007), C_4(2007)\}$  and  $C_2(2006) \searrow \{C_1(2007), C_2(2007), C_3(2007), C_4(2007)\}$ ) and two merges ( $\{C_1(2006), C_2(2006)\} \searrow C_1(2007)$  and  $\{C_1(2006), C_2(2006), C_3(2006)\} \searrow C_4(2007)$ ). Using the K-means algorithm we obtain a partition in five clusters, and different transitions results. During [2005, 2006] a cluster was born ( $\emptyset \rightarrow C_1(2006)$ ), four clusters died ( $C_3(2005) \rightarrow \emptyset$ ,  $C_4(2005) \rightarrow \emptyset$ ,  $C_5(2005) \rightarrow \emptyset$  and  $C_6(2005) \rightarrow \emptyset$ ) and two clusters survived ( $C_1(2005) \rightarrow C_3(2006)$  and  $C_2(2005) \rightarrow C_2(2006)$ ). For the surviving clusters we looked for transitions concerning its contents. The detected internal transitions were only related to changes in size. Thus, we found that cluster  $C_1(2005)$  shrinks ( $C_1(2005) \searrow C_3(2005)$ ), reducing the number of observations assigned to it, and that cluster  $C_2(2005)$  suffers the reverse process - Expansion - increasing its cardinality ( $C_1(2005) \nearrow C_2(2006)$ ). During [2006, 2007], we detect the death of one cluster ( $C_1(2006) \rightarrow \emptyset$ ), the birth of two clusters ( $\emptyset \rightarrow C_2(2007)$  and  $\emptyset \rightarrow C_4(2007)$ ) and the survival of another two clusters ( $C_2(2006) \rightarrow C_3(2007)$  and  $C_3(2006) \rightarrow C_1(2007)$ ). Both surviving clusters shrink ( $C_2(2006) \searrow C_3(2007)$  and  $C_3(2006) \searrow C_1(2007)$ ).

Comparing the results obtained for different representation schemes we can deduce that transitions detection algorithm's behavior differs. The most obvious difference is the number of deaths and births, which is much higher in the method of representation by comprehension.

Afterwards, we conducted an experiment with a traditional technique to study evolution: we merged all CBSD' datasets into one (each object appears 3 times using different labels -  $obj_{2005}$ ,  $obj_{2006}$  and  $obj_{2007}$ ), applied a clustering algorithm (we used K-means for  $k = 5$ ) and analyzed the contents of each cluster. The initial hypothesis is: objects that didn't evolve must be allocated to the same cluster and objects that experienced significant changes over time must be assigned to different clusters. The inspection of clusters suggested that "surviving" activity sectors were assigned to the same cluster and the ones who experienced merge, split, birth or death were assigned to different clusters, confirming the hypothesis. However, this is a technique with limited capacity to represent evolution, since it only returns a dichotomic result ( $obj1$  experienced evolution or  $obj1$  didn't experienced evolution).

## 6 CONCLUSION

In this paper we introduced a framework for addressing the problem of monitoring the evolution of clusters' structures. We define and propose two different strategies to represent clusters. We also propose two methods to monitor clusters' transitions, for each type of cluster representation. The detected transitions can be internal or external and were defined in the taxonomy. We present a real world

case study using datasets extracted from Banco de Portugal to prove the applicability and feasibility of our framework.

The novelty of these methods lies in the use of bipartite graphs and conditional probabilities to monitor the transitions of clusters represented by enumeration, and in the computation and assessment of clusters' overlap, for compact representation of clusters.

As future work we intend to develop similar methods to monitor the evolution of social networks. We also aim to experiment our framework in data streams. In this special case, and since our methods are offline, they have to be applied on the top of an online clustering algorithm.

**Acknowledgements:** Thanks to the support of the project Knowledge Discovery from Ubiquitous Data Streams (PTDC/EIA-EIA/098355/2008).

## REFERENCES

- [1] Charu C. Aggarwal, 'A framework for change diagnosis of data streams', in *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, June 9-12, 2003, San Diego, California, USA*, pp. 575–586. ACM, (2003).
- [2] Charu C. Aggarwal, 'On change diagnosis in evolving data streams', *IEEE Transactions on Knowledge and Data Engineering*, **17**(5), 587–600, (2005).
- [3] Steffan Baron and Myra Spiliopoulou, 'Monitoring the evolution of web usage patterns', in *Proceedings of the 1st European Web Mining Forum, Cavtat-Dubrovnik, Croatia, September 22, 2003*, volume 3209 of *Lecture Notes in Computer Science*, pp. 181–200. Springer, (2004).
- [4] Ilaria Bartolini, Paolo Ciaccia, Irene Ntoutsis, Marco Patella, and Yannis Theodoridis, 'The panda framework for comparing patterns', *Data and Knowledge Engineering*, **68**(2), 244–260, (2009).
- [5] Mirko Böttcher, Frank Höppner, and Myra Spiliopoulou, 'On exploiting the power of time in data mining', *SIGKDD Explorations*, **10**(2), 3–11, (2008).
- [6] Tanja Falkowski, Jörg Bartelheimer, and Myra Spiliopoulou, 'Mining and visualizing the evolution of subgroups in social networks', in *Proceedings of the 2006 IEEE / WIC / ACM International Conference on Web Intelligence, Hong Kong, China, 18-22 December, 2006*, pp. 52–58. IEEE Computer Society, (2006).
- [7] S. Kaur, V. Bhatnagar, S. Mehta, and S. Kapoor, 'Concept drift in unlabeled data stream', Technical report, University of Delhi, (2009).
- [8] Marcia Oliveira and Joao Gama, 'Bipartite graphs for monitoring clusters transitions', in *Proceedings of the Ninth International Symposium on Intelligent Data Analysis, Tucson, Arizona, USA, May 19-21, 2010*, volume 6065 of *Lecture Notes in Computer Science*, pp. 114–124. Springer, (2010).
- [9] Myra Spiliopoulou, Irene Ntoutsis, Yannis Theodoridis, and Rene Schult, 'Monic: modeling and monitoring cluster transitions', in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, pp. 706–711. ACM, (2006).
- [10] Hui Yang, Srinivasan Parthasarathy, and Sameep Mehta, 'A generalized framework for mining spatio-temporal patterns in scientific data', in *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*, pp. 716–721. ACM, (2005).
- [11] Tian Zhang, Raghu Ramakrishnan, and Miron Livny, 'Birch: An efficient data clustering method for very large databases', in *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, pp. 103–114. ACM Press, (1996).



# Context-Aware Stream Ensemble for Ubiquitous Environments

João Bártolo Gomes<sup>1</sup> and Santiago Eibe<sup>2</sup> and Pedro A. C. Sousa<sup>3</sup> and Ernestina Menasalvas<sup>4</sup>

## Abstract.

The dynamic and unstable nature of ubiquitous environments manifests itself in learning systems through changes in data, context and resource availability. Ubiquitous data stream learning systems must be aware and adapt to such changes so that incoming data can be classified with high accuracy. Ensemble approaches have been shown successfully in dealing with concept changes. Nevertheless, the contextual information available in ubiquitous environments could be exploited to improve change detection or even anticipate recurring concepts. In this work, we enhance ensemble approaches for the problem of concept drift with recurring concepts by dynamically adding and removing weighted classifiers in response to changes not only in concepts but to context and resources. Representing and comparing context in a context-concept history is one of the challenges addressed in this paper. This is used in the adaptation to recurring concepts and in the management of knowledge from previous learned concepts while adapting to resource constraints. We present and discuss preliminary experimental results with synthetic and real datasets.

## 1 Introduction and Related Work

Learning from data streams in ubiquitous environments introduces several challenging issues, especially the known problem of concept drift[12], where changes in the underlying concept require the actual decision model to be revised. The concept drift problem is observed across several ubiquitous application domains, such as health assistance; sensor networks; intelligent vehicles; and recommender systems. In such domains concepts are context-dependent, which means that changes in concepts are due to changes in context. Still, the relation between context and concepts, referred in the literature as *hidden context*[15, 14, 6], is not known. Concept recurrence[16, 10, 8, 4] is also common in such domains.

Building an ensemble using classifiers learned from sequential chunks of stream data[11, 13, 8] and adapting the weights of the classifiers according to the underlying concept has been shown a successful approach to deal with concept drift[11, 13] and recurring concepts[10, 8]. In such approaches, the main challenge is the determination of classifier weights and the size of chunks. An approach to deal with concept recurrence that does not require this was proposed by Gama and Kosina in[4]. It uses the drift detection method presented in[5] to identify the stable concepts and memorizes learned classifiers that represent these concepts. Referees are used to choose

the most appropriate classifier to reuse after drift is detected in situations of recurrence (i.e. the referee prediction about the applicability of the classifier is greater than a pre-defined threshold). However, none of the previous approaches exploits context information to deal with recurrence.

Technological advances enable context representation in many ubiquitous computing environments, through the information gathered from different sensors. Context-awareness is one of the key aspects of ubiquitous computing environments as applications need to monitor context changes and adapt to them[2, 9].

Context representation is a problem studied by many researchers as they attempt to formally define context. The well known definition from Dey[2] states that ‘Context is any information that can be used to characterize the situation of an entity’. In contrast Brézillon and Pomerol[1] argue that no particular knowledge can be objectively called context, as context is in the eye of the beholder, stating that ‘knowledge that can be qualified as ‘contextual’ depends on the context!’.

In situations where context information is related to the underlying concepts, this knowledge could be exploited to detect and adapt to recurring concepts. Nevertheless, those relations are not known apriori, and it is even possible that given the available context information it is hard or impossible to find such relations. Still, in many real-world problems we find examples where the available context does not explain all global concept changes, but partially explains some of those. For example, user preferences can change with time or location. A user has different interests during the weekends, weekdays, when at home or at work. In general, different concepts can recur due to periodic context (e.g. seasons, locations, week days) or non-periodic context (e.g. rare events, fashion, economy).

The idea of using context to track changes is not new and has been studied in existing works[14, 6]. In these works, it is assumed that learning systems should be able to detect and adapt to concept changes without explicitly being informed about those changes, using only the available contextual features and the performance of the base learner as a measure to detect change in the underlying concept.

Widmer in[14] exploits what is referred as contextual clues and proposes a meta-learning method to identify such clues. Contextual clues are context-defining attributes or combinations of attributes whose values are characteristic of the underlying concept. When more or less systematic changes in their values are observed this might indicate a change in the target concept. The method automatically detects contextual clues on-line, and when a potential context change is signaled, the knowledge about the recognized context clues is used to adapt the learning process in some appropriate way. However, if the *hidden context* is not represented in the contextual clues it is not possible to detect and adapt to change.

<sup>1</sup> Universidad Politécnica Madrid, joao.bartolo.gomes@alumnos.upm.es

<sup>2</sup> Universidad Politécnica Madrid, Spain, email: seibe@fi.upm.es

<sup>3</sup> Universidade Nova de Lisboa, Portugal, email: pas@fct.unl.pt

<sup>4</sup> Universidad Politécnica Madrid, Spain, email: emenasalvas@fi.upm.es

The approach of conceptual clustering proposed by Harries[6], identifies stable *hidden contexts* from a training set by clustering the instances assuming that similarity of context is reflected by the degree to which instances are well classified by the same concept. A set of models is constructed based on the identified clusters. This idea proved to work very well with recurring concepts and real world problems. However, its main drawback is the off-line training required to obtain the conceptual clusters, as these could lead to inaccuracy with concepts or patterns that were not seen during training.

In this work we propose to integrate context information in an ensemble based approach to deal with concept drift and recurring concepts while in a data stream mining scenario[12, 3]. The proposed system maintains a pool of classifiers that represent previously learned concepts. Context information is integrated with those concepts to create a context-concept history and explicitly addresses the problem of recurring concepts. Context is represented by a set of attribute-value variables. We assume that an increase in the classification error-rate is the observable effect of a concept change[5, 16, 4]. To represent the current underlying concept an ensemble is built using classifiers from the pool. The classifiers used in the ensemble and its weights are dynamically determined using context information and the classifiers estimated error. If the ensemble is unable to represent the current concept, a new classifier is learned and added to the pool.

Furthermore, the approach adapts itself to the underlying resource constraints in terms of memory and processing time, in order to optimize the trade-off between accuracy and efficiency. The drawback of this approach is the memory costs as it requires memorizing concept representations and also the context-concept history, which could be problematic when executing in ubiquitous devices with strong memory constraints. To deal with this problem we present a strategy to discard models in situations of memory scarcity. Our preliminary experiments show that despite the memory consumption the efficacy of the process increases.

Associating context and concepts is not trivial and when exploiting such relations to improve adaptation to recurring concepts, one could argue that context information should be simply added as additional attributes in the base learner. However, that would introduce noise in the learning process, as concepts may change due to factors that are not expressed as context attributes. Therefore we believe that such context information should be integrated carefully in a meta-learning level (as proposed by some researchers[14, 6]).

Another problem to be solved are possible conflicts with recurring concepts in the presence of different context. Our approach is sensitive to this because the context-concept history represents such changes, and because context is not the only factor when calculating the weights of the classifiers in the ensemble.

This paper is structured as follows: In section 2 the Context-Aware Stream Ensemble system and its challenges are presented, and we discuss how such challenges are addressed by the proposed solution. In section 3 we evaluate the behavior and accuracy of the approach using synthetic and real datasets. In section 4 we express our concluding remarks and outline future research work.

## 2 Context-Aware Learning System

It has been shown that an ensemble of classifiers can outperform a single classifier[13, 10]. The proposed learning system maintains a pool of classifiers, and creates an ensemble to classify incoming unlabeled records using the classifiers that are able to represent the current underlying concept. The system adds and removes classifiers

to the ensemble and calculates its weights based on accuracy and context. Reusing previously learned classifiers in situations where concepts recur leads to faster adaptation to changes and gains in accuracy, while saving computational costs associated with relearning a concept (i.e. train a classifier), improving the system added value in terms of accuracy-efficiency.

We assume a stream of records  $(\vec{x}, y)$  that arrive one at a time, where  $\vec{x}$  is a vector of attribute values and  $y$  is the class label. If the underlying concept is new (i.e. not a recurring concept), an incremental base learner is used to learn a classifier from the incoming labeled records, to represent this new concept. The obtained classifier is added to the pool and is used to make predictions about the incoming records class label. We also assume a stream of records  $\vec{c}$  that is a vector of values of context attributes and that  $m_{limit}$  is the value of memory available in the device. Using the history of concept changes and the context stream, we learn relations between context and learned concepts. Exploiting context-concept relations we are able to improve the adaptation to recurring concepts. The adaptation strategy dynamically defines the weights of the classifiers so the ensemble represents the current underlying concept given the available context. The proposed system has to deal with several challenges: i) represent and find context relations with learned concepts ii) what classifiers to use in the ensemble and how to calculate their weights iii) what classifiers should be kept or discarded. The solution to address such challenges is explained in the following subsections.

### 2.1 Preliminaries

#### 2.1.1 Context representation and similarity

The literature contains a large number of studies that attempt to formally define context[1, 2, 9]. We represent a context state as an object in a multidimensional Euclidean space. A context state  $C_i$  is defined as a tuple of  $N$  context attribute-values,  $C_i = (a_1^i, \dots, a_n^i)$  where  $a_n^i$  represents the value of context attribute  $a_n$  for context state  $C_i$ .

The available context information should be represented in a formal manner and depends on the learning environment and data mining problem. Context information can represent simple low level sensors (i.e. temperature, humidity) or a high-level context (i.e. season, location, user activity) defined by domain experts or inferred by other means beyond the scope of the problem discussed in this work.

Context similarity is not a trivial problem[9] as inferred context is associated with inherent ambiguity and inaccuracy which is often modeled by fuzzy functions. For the purposes of this work, we use a boolean similarity function that calculates the normalized Euclidean distance between two context states. If the distance between two context states  $C_i, C_j$  is below a certain threshold, these are considered similar. For an attribute  $a_k$  with nominal values we define the distance as 1 if  $a_k^i$  and  $a_k^j$  have different values and 0 otherwise.

#### 2.1.2 Base learner - Concept representation and similarity

The base learner is used to learn a classifier that represents the data stream records' underlying concept. Any classification algorithm able to learn incrementally can be used for this task, and the decision can be made according to the nature of data and the ubiquitous device capabilities, choosing the algorithm that best suits them (e.g. high accuracy, handles noise, memory consumed, fast execution). For this task we used the NaiveBayes algorithm, because it represents the concepts in a compact form (i.e. memory efficiency), is incremental, easy to implement[3], handles nominal as well as continuous attributes and has shown good results as a base learner[8, 4, 14].

To determine whether a certain classifier represents a new concept or a reappearing concept a concept similarity measure is required. We use *Conceptual equivalence* measure, proposed by Yang et al.[16]. Given two classifiers  $C_1, C_2$  and a dataset  $D_n$  of  $n$  records, it calculates  $ce$  as their degree of equivalence represented in the range  $[-1, 1]$ . The bigger the output value, the higher the degree of conceptual equivalence. For the records in  $D_n$  it compares how  $C_1$  and  $C_2$  classify the records. If the classifiers agree in their prediction 1 is added to the result otherwise -1 is added. The final  $ce$  is the result of the previous calculation divided by the number of records in  $D_n$ . The authors[16] argue that the accuracy and the conceptual equivalence degree are not necessarily positively correlated. The reasoning is that, despite  $C_1$  and  $C_2$  classify  $D_n$  with low accuracy, their equivalence degree can be very high if their classifications match, even when both misclassify. Moreover, that does not represent conceptual equivalence as they can still achieve the same accuracy and misclassify different parts of the attribute space. The drawbacks of this measure are related to the selection of dataset  $D_n$ , because we must pre-define its minimum size and also for the purpose of our approach, the records in  $D_n$  should belong to the same underlying concept.

We consider that if the obtained  $ce$  value is above a pre-defined threshold, the models are similar and thus represent the same underlying concept. However, the decision of such thresholds is not trivial and has impact on the results obtained.

### 2.1.3 Concept Drift detection

To detect when changes in the underlying concept occur we use the drift detection method proposed by Gama et al.[5]. It considers the classifier error-rate (i.e. false predictions) to be a random variable from Bernoulli trials. The binomial distribution gives the general form of the probability of observing an error. For each record  $i$  in the sequence being sampled and  $error_i$  the number of misclassifications at  $i$ , the error-rate is the probability of misclassifying  $p_i = (error_i/i)$ , with standard deviation given by  $s_i = \sqrt{p_i(1 - p_i)/i}$ . It is assumed that  $p_i$  will decrease while the number of records increases if the distribution of the examples is stationary. A significant increase in  $p_i$ , indicates that the class distribution is changing. The values of  $p_i$  and  $s_i$  are calculated incrementally and their minimum values ( $p_{min}, s_{min}$ ) are recorded when  $p_i + s_i$  reaches its minimum value. A warning level and a drift level are defined using  $p_i, s_i, p_{min}, s_{min}$ , representing the different levels of confidence. In the proposed system, after a warning level is signaled the incoming records are processed using the base learner to represent a potential new concept, and such records are kept in a warning window.

## 2.2 Learning context-concept relations

The idea to explore the concept history has also been used in [6, 16, 4], where a model representing the concept history is built from the changes observed between concepts. This way, if history repeats itself when concepts recur, it is possible to anticipate the detection of change.

To represent concept history and better adapt to recurrence, here we associate the concepts (i.e. learned classifiers) with context information. Such history is also of great interest to model the concept changes and associated context as domain experts can analyze these structures to better understand the nature of change or in applications where modeling concept changes is required.

To learn the relations between the context attributes and the underlying concepts, we train a classifier, where for every batch of a pre-defined number of context records, the most frequent context  $\vec{c}$  in the batch is associated with the classifiers  $C_i$  that represent the current underlying concept (i.e. are used in the ensemble). Therefore the training records for this learner will be in the form  $(\vec{c}, y = C_i)$ . As learning algorithm we used the NaiveBayes algorithm, because it is simple to modify incrementally the frequency tables as new concepts are learned. Also it allows us to estimate the probability  $P(C = C_i|\vec{c})$ , that represents the frequency for each classifier  $C_i$  given context  $\vec{c}$ . This is used in calculation of classifiers weights as it is described in subsection 2.3.1.

## 2.3 Ensemble management

When change is detected two situations are possible: i) the underlying concept is new (i.e. no equivalent concept is represented in the pool) and the base learner will learn the new underlying concept by processing the current incoming labeled records. The incremental classifier that is being learned will also classify the incoming unlabeled records as anytime classification is assumed; ii) the underlying concept is recurrent (i.e. has been learned previously). In this situation the classifiers in the pool that represent the underlying concept integrate the ensemble that is created to classify incoming unlabeled records. Further description of this process and the weight that is assigned to each classifier in the ensemble is detailed in 2.3.1.

### 2.3.1 Classifier weighting

The main objective of the ensemble is to represent the current underlying concept and thus be able to classify the incoming records with high accuracy. One of the most important challenges of stream ensemble approaches is on how to select/weight the classifiers in the ensemble[13, 10]. We combine an accuracy weighted ensemble approach as proposed by Wang et al.[13] and the information learned from the context-concept history described in 2.2. In the accuracy weighted approach the classifier  $C_i$  accuracy is estimated using the Mean Square Error. This measure estimates the error of the classifier for a set of records  $W_n$ . The Mean Square Error  $MSE_i$  for classifier  $C_i$ , using the window  $W_n$  of  $n$  records in the form of  $(x, c)$ , where  $c$  is the true class label for that record, can be expressed as:

$$MSE_i = \frac{1}{|W_n|} \sum_{(x,c) \in W_n} (1 - f_c^i(x))^2$$

where the error of  $C_i$  on record  $(x, c)$  is  $1 - f_c^i(x)$ , and  $f_c^i(x)$  is the probability given by  $C_i$  that  $x$  is an instance of class  $c$ .

The  $MSE_r$  is the error estimation of a random classifier and is used as part of the filtering criterion (i.e. decide which classifiers are used in the ensemble), here we could have used the conceptual equivalence measure discussed in 2.1.2. However, conceptual equivalence degree and accuracy are not necessarily correlated, and also the accuracy based approach is less restrictive as it allows for more diversity in the ensemble, which may achieve higher accuracy. Classifiers  $C_i$  with  $MSE_i * acceptanceFactor$  greater than  $MSE_r$  and context distance with the occurring context greater than a pre-defined threshold are not included in the ensemble. The *acceptanceFactor* is in the range  $[0, 1]$  and was proposed in [10], to avoid that classifiers that show minimal improvement from  $MSE_r$  participate in the ensemble.

The weight  $W_i$  assigned to Classifier  $C_i$  in the ensemble is formulated, using the  $MSE_i$  calculated using dataset  $W_n$  and with probability  $P(C = C_i | \vec{c}_o)$  that  $C_i$  represents the underlying concept given the occurring context  $\vec{c}_o$  (i.e. the most frequent context observed for the records in  $W_n$ ):

$$W_i = acc_w * (MSE_r - MSE_i) + ctx_w * P(C = C_i | \vec{c}_o)$$

The  $acc_w$  and  $ctx_w$  are pre-defined weights assigned to error and context factors in the calculation of  $W_i$ . The sum of  $acc_w$  and  $ctx_w$  is 1. The weights are normalized according with the number of classifiers in the ensemble.

## 2.4 Classifier pruning strategy

Reusing classifiers may lead to higher accuracy and savings in the computational costs associated with relearning a recurrent concept. However, the memory cost associated with storage must also be taken into account. The system is aware of the space consumed by the classifiers in the pool and ensures that this value is within the memory limit  $m_{limit}$ . In situations where its not possible to add new classifiers to the pool without exceeding the memory limits, an intelligent strategy is used to discard the classifier that is considered to have the lowest utility. This is defined using the criterion in the following order: i) equivalent classifiers (i.e. have the highest  $ce$  value) ii) classifiers that are associated with similar contexts (i.e. the contexts have lower distance) iii) oldest, in case of ties when using the other criterions. The goal of this utility definition is to promote heterogeneity in the pool, and thus the overall accuracy-efficiency value of the learning system.

## 2.5 Learning process

The continuous learning process consists of the following steps:

1. if the underlying concept is new (state variable  $newConcept$ ), the classifier is incrementally updated (base learner) with the incoming labeled records. This classifier is used to classify unlabeled records, as all the other classifiers have weight 0 in this situation.
2. otherwise (i.e. concept is recurrent) use the ensemble to classify unlabeled records.
3. the drift detection method monitors the error-rate of the system predictions.
4. the classifiers in use are associated with context records.
5. when the error-rate goes up, the drift detection method indicates:
  - **warning level:** Prepare a new base learner  $C_{new}$  that processes incoming records while the warning level is signaled. The records observed after the warning level are kept in a warning window  $W_n$ . After  $n$  records in this level, the ensemble is recalculated.
  - **drift level:** use classifier  $C_{new}$  prepared during the warning level to classify new incoming records. When the warning window has a size of  $n$  records (i.e. stable underlying concept), compare  $C_{new}$  with every classifier  $C_i$  in the pool in terms of conceptual equivalence. If the  $ce$  value is above a pre-defined threshold for any  $C_i$  (i.e. recurrent concept) then discard  $C_{new}$ , recalculate the ensemble and set the state variable  $newConcept$  to *false*; otherwise  $C_{new}$  is added to the pool, its weight is set to 1.0 and all the other classifiers set to 0, and set  $newConcept$  to *true*, so it continues to be updated.
  - **false alarm (normal level after warning):** the warning window is cleared and the learner  $C_{new}$  is discarded.

## 3 Evaluation

To test the proposed approach, an implementation was developed in Java, using the MOA[7] framework as a test-bed. The available evaluation features, the implementation of the drift detection method(DMM)[5] and the NaiveBayes (as base learner) algorithm have been used. We used synthetic and real world datasets to evaluate the behavior and accuracy of the proposed approach. The experiments compared our approach with the *SingleClassifierDrift* algorithm in MOA[7], that uses the Gama et al. DMM[5] to detect drift and adapts by learning a new classifier (i.e. discards previous concept representations).

### 3.1 Experimental setup

The SEA Concepts[11] is a synthetic *benchmark* for data streams that uses different functions to simulate concept drift, allowing control over the underlying concepts and its recurrence in our experiment. The SEA Concepts dataset has two classes {class0, class1} and three features but only the first two features are relevant. Four target concept functions are defined[11]. We used a numerical context feature space with two features  $a_1$  and  $a_2$  with values between 1 and 4,  $a_1$  is correlated with the target concept, and  $a_2$  value generated randomly. We generated 250000 records and changed the underlying concept every 15000 records. The test was executed with a 10% noise value as in the original paper[11], which means that the class value of the training record is wrong in 10% of the records.

The real world emailing list (elist) dataset[8] is a stream of email messages from different topics that are labeled by a user as interesting or junk according to his preferences. The dataset is a stream of 1500 examples and contains 2 recurrent concepts that change every 300 records. The first concept represents messages where user is only interested in medicine and in the second concept the interest changes to space and baseball. A contextual attribute location that is correlated with the target concept was used in the context stream.

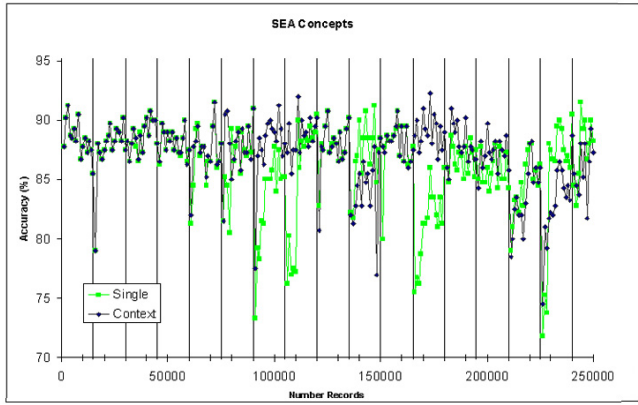
A window with 300 records and 15 records was used during the warning level in the SEA and elist datasets respectively. For parameters, 0.75 was used as threshold for equivalence and context, *acceptanceFactor* was set to 0.5 and the weights  $acc_w$  and  $ctx_w$  were set to 0.5.

### 3.2 Results

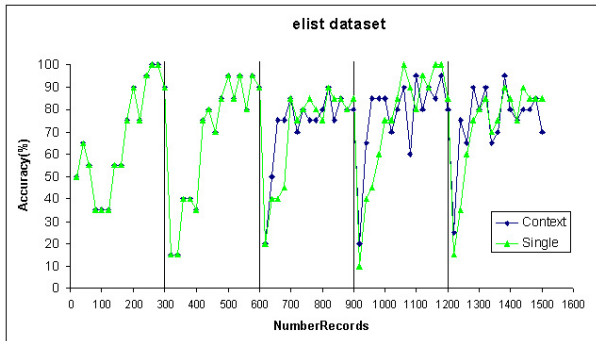
The preliminary results show that the proposed approach achieves better overall accuracy, adapts faster to drift and in situations of recurrent concepts is able to improve the accuracy between changes. It also reduces the number of processed records (i.e. training records used by the base learner).

In the SEA dataset the proposed approach achieves 87% accuracy and processed 14983 less records than the *SingleClassifierDrift* which achieved 86.2% accuracy. The proposed approach was able to identify context and reuse past classifiers, however due to the noise in the context stream sometimes (e.g. around record 135000) the adaptation is not as expected. We believe this is something that can still be improved as discussed in future work.

For the email list dataset experiment, the approach behaved as expected as it adapted faster, was able to identify the context, achieved 71.9% accuracy and processed 641 less records than *SingleClassifierDrift* which achieved 70.3% accuracy. This dataset has less target concepts than SEA dataset and it is free from noise, either in the target class as well as in the context records, which makes the comparison between the approaches more clear.



**Figure 1.** Comparison between the proposed approach(Context) and SingleClassifierDrift(Single) with the SEA dataset. Black lines indicate drift



**Figure 2.** Comparison between the proposed approach(Context) and SingleClassifierDrift(Single) with the elist dataset. Black lines indicate drift

## 4 Conclusion and Future Work

In this work, we have proposed a Context-Aware Stream Ensemble learning system. The system exploits available context information to improve existing ensemble approaches to handle drift and recurring concepts. We have presented the challenges associated with the approach and how the solution addresses those.

We also presented preliminary results with the SEA concepts and the Email List datasets. We observed that the proposed approach adapts faster in recurring concept situations while observing some improvement on the accuracy and a reduction in the number of records processed. We should note that even small accuracy gains in certain domains, for example in detection of credit card fraud, could have a significant financial impact, and that the ability to process less records could be of significant interest, particularly in ubiquitous devices where processing records is costly. The approach is also sensitive to memory scarcity, which is another factor to consider in ubiquitous devices. However, this requires further testing as in our experimental settings the memory used by the classifiers was within memory limits. We should also say that the proposed system can still be improved, in terms of the naive representation and function used to compare context, making it more robust to uncertainty and noise. Also the system still requires setting several parameters, particularly the warning window size, which represents an additional problem related to the selection of such parameters.

Consequently, future work will focus on context similarity and in this sense we are currently working on the use of fuzzy functions. We would also like to make the system less dependent of pre-defined parameters. Finally, we plan to study in depth the accuracy-efficiency trade-off of the approach when executing in devices where resources are severely limited, analyzing overhead costs both in resource consumption and response time enabling the system to execute in different ubiquitous devices.

## ACKNOWLEDGEMENTS

The work of J.P. Bártolo Gomes is supported by a Phd Grant of the Portuguese Foundation for Science and Technology (FCT). This research is partially financed by project TIN2008-05924 of Spanish Ministry of Science and Innovation.

## REFERENCES

- [1] R. Brezillon and J.C. Pomerol, 'Contextual knowledge sharing and cooperation in intelligent assistant systems', *TRAVAIL HUMAIN*, **62**, 223–246, (1999).
- [2] A.K. Dey, 'Understanding and using context', *Personal and ubiquitous computing*, **5**(1), 4–7, (2001).
- [3] J. Gama and M.M. Gaber, *Learning from data streams: processing techniques in sensor networks*, Springer-Verlag New York Inc, 2007.
- [4] J. Gama and P. Kosina, 'Tracking Recurring Concepts with Meta-learners', in *Progress in Artificial Intelligence: 14th Portuguese Conference on Artificial Intelligence, Epia 2009, Aveiro, Portugal, October 12-15, 2009, Proceedings*, p. 423. Springer, (2009).
- [5] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, 'Learning with drift detection', *Lecture Notes in Computer Science*, 286–295, (2004).
- [6] M.B. Harries, C. Sammut, and K. Horn, 'Extracting hidden context', *Machine Learning*, **32**(2), 101–126, (1998).
- [7] G. Holmes, R. Kirkby, and B. Pfahringer. MOA: Massive Online Analysis, 2007 - <http://sourceforge.net/projects/moa-datastream/>.
- [8] I. Katakis, G. Tsoumakas, and I. Vlahavas, 'Tracking recurring contexts using ensemble classifiers: an application to email filtering', *Knowledge and Information Systems*, 1–21.
- [9] A. Padovitz, S.W. Loke, and A. Zaslavsky, 'Towards a theory of context spaces', in *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pp. 38–42, (2004).
- [10] S. Ramamurthy and R. Bhatnagar, 'Tracking recurrent concept drift in streaming data using ensemble classifiers', in *Proc. of the Sixth International Conference on Machine Learning and Applications*, pp. 404–409, (2007).
- [11] W.N. Street and Y.S. Kim, 'A streaming ensemble algorithm (SEA) for large-scale classification', in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 377–382. ACM New York, NY, USA, (2001).
- [12] A. Tsymbal, 'The problem of concept drift: definitions and related work', *Computer Science Department, Trinity College Dublin*, (2004).
- [13] H. Wang, W. Fan, P.S. Yu, and J. Han, 'Mining concept-drifting data streams using ensemble classifiers', in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 226–235. ACM New York, NY, USA, (2003).
- [14] G. Widmer, 'Tracking context changes through meta-learning', *Machine Learning*, **27**(3), 259–286, (1997).
- [15] G. Widmer and M. Kubat, 'Learning in the presence of concept drift and hidden contexts', *Machine learning*, **23**(1), 69–101, (1996).
- [16] Y. Yang, X. Wu, and X. Zhu, 'Mining in anticipation for concept change: Proactive-reactive prediction in data streams', *Data mining and knowledge discovery*, **13**(3), 261–289, (2006).



# A Data Clustering Approach to Energy Conservation in Wireless Sensor Networks

Alexander Sagen<sup>1</sup> and Cyril Labbé<sup>2</sup> and Mohamed Medhat Gaber<sup>3</sup>  
and Shonali Krishnaswamy<sup>4</sup> and Agustinus Borgy Waluyo<sup>5</sup> and Seng Loke<sup>6</sup>

## Abstract.

This paper presents a new cluster-based power preservation scheme. Our clustering strategy for power saving is defined based on the “similarity of data” coming out from the sensors. The proposed clustering works in conjunction with our learning algorithm to obtain an optimum sleeping time of the sensors without disrupting the monitoring activity. The algorithm helps to effectively regulate the activation or de-activation of the sensor node’s radio transceiver, which in turn prolong the lifetime of the network.

We have carried out several real-world experiments concerning power utilisation of wireless sensor devices in different scenarios and found that the proposed method leads to a significant power efficiency improvement with up to four times longer battery lifetime than other cases without such scheme.

## 1 Introduction

The progressive advances of wireless sensor devices have made the adoption of such devices become increasingly essential and pervasive. Each sensor devices is equipped with a small processor, wireless transmitter, and battery. These sensors are capable of sampling, processing, and communicating environmental parameters including temperature, humidity and light or physiological signs such as blood pressure and movement [10]. After receiving the data from the sensor nodes, the gateway or base station processes the data for specific applications. Typical applications associated with these sensors include environmental monitoring [11], surveillance [8], location tracking [1] and activity recognition [12] to name a few.

As wireless sensor devices are reasonably small-sized and lightweight, energy efficiency is extremely critical considering that sensor nodes are powered only by short life-span batteries, which cannot be recharged during deployment [9]. Consequently, this requires power saving features in order to prolong the sensor node’s battery life and ensure that sensors remain operational for the longest time possible.

In light of this issue, this paper aims to investigate power utilization of the sensor node in different conditions and apply a clustering scheme as part of our power saving feature, which is designed to regulate the activation or de-activation of the

sensor node’s radio transceiver. The proposed cluster-based power saving method is expected to preserve power of the sensor node without overly disrupting the monitoring activity. The clustering of the sensors is done based on the reading values coming out from the sensors. Subsequently, the algorithm determines the sleep time and the cluster boundaries derived from a pre-determined learning process, which takes into account the historical event occurring in the clusters.

By clustering data coming from sensors, we aim to cluster sensors according to “similar” behavior and their sensed data, yielding the notion of sensors being “similar” because they provide “similar” data readings. This knowledge can then be used to save energy, particularly when approximate values are required. A sensor that has just been assigned (by virtue of its history of readings) a cluster can afford to sleep for a while, waking up from time to time to see if it needs to transmit its readings to the base station. A sensor assigned to a cluster (of “similar” data) does not need to transmit its readings (and can even go into deep sleep), unless and until, on waking up, it finds that its current reading has changed enough to, effectively, put the sensor outside of its assigned cluster. This ability of the sensor to sleep (perhaps quite often due to our strategy) helps the sensor to save energy, while still providing a fairly accurate view of the world (compared to when it does not sleep). Because the approach is data-centric, exploiting only a simple characteristic of data values, it is applicable across sensor platforms, different sensor network sizes, different types of sensor data, and applications. This strategy is also more general and effective than the simple strategy of a sensor sending data only when there is change (compared to a previous value, for example), since it takes into account the *significance* of such changes, as measured relative to the other sensors in the cluster (change that is not enough to put the sensor outside of its assigned cluster is ignored).

The remainder of this paper is organized as follows. We present our proposed clustering and energy saving algorithm in Section 2. The infrastructure used to implement the proposed scheme is described in Section 3, followed by our experiments utilizing the newly implemented feature in Section 4. Some existing works related to this paper is given in section 5. Finally, Section 6 concludes the paper.

## 2 The Approach: clustering data for energy saving

In our method, given a set of data values coming from sensors, these values are clustered, and given this set of clusters (and associated information such as boundaries, centroids),

<sup>1</sup> Monash University, Australia, asagen@gmail.com

<sup>2</sup> University of Grenoble, France, Cyril.Labbe@imag.fr

<sup>3</sup> University of Portsmouth, UK, mohamed.gaber@port.ac.uk

<sup>4</sup> Monash University, Australia, Shonali.Krishnaswamy@infotech.monash.edu.au

<sup>5</sup> Monash University, Australia, awaluyo@infotech.monash.edu.au

<sup>6</sup> La Trobe University, Australia, S.Loke@latrobe.edu.au

---

**Algorithm 1** Algorithm on each sensor: sensors are sleeping following the base station instruction

---

```

1: Relearn  $\leftarrow$  True ; TimeSlept  $\leftarrow$  0
2: NextSleepTime  $\leftarrow$  0 ; Radio  $\leftarrow$  Off
3: Boundaries  $\leftarrow$   $[-\infty, +\infty]$ 
4: loop
5:   Value  $\leftarrow$  Sense()
6:   if Radio = Off and (Value  $\notin$  Boundaries or Relearn = True) then
7:     Radio  $\leftarrow$  On
8:   end if
9:   if Relearn = False and (Value  $\notin$  Boundaries or TimeSlept  $\geq$  MaxReportTime) then
10:    Send(Value) to base station.; TimeSlept  $\leftarrow$  0;
11:    Receive(Boundaries, Relearn, NextSleepTime) from base station.
12:   end if
13:   if Relearn = True then
14:     while Relearn do
15:       Send(Value) to base station.
16:       Receive(Boundaries, Relearn, NextSleepTime) from base station.
17:       Radio  $\leftarrow$  Off; GoToSleep(NextSleepTime)
18:     end while
19:   end if
20:   Radio  $\leftarrow$  Off;
21:   TimeSlept  $\leftarrow$  TimeSlept + NextSleepTime
22:   GoToSleep(NextSleepTime)
23: end loop

```

---

corresponding sensors are assigned to a cluster. Such information is used to decide whether or not data from a particular sensor is needed, and to decide a sleep time for the sensor.

A strategy, calculating the most desirable sleep time, as a compromise between battery preservation and data resolution has been developed. Section 2.1 states necessary definitions and section 2.2 and 2.3 give algorithms executed on each sensor and on the base station.

## 2.1 Method overview

Lets assume a system composing of  $n$  sensors enumerated and denoted by  $s_1, s_2, s_3, \dots$ . The stream  $d_i$  of data values from a sensor  $i$  is an ordered set  $d_{i,1}, d_{i,2}, d_{i,3}, \dots$  of data values.

Clustering of available data results in a set of  $k$  clusters denoted by  $c = \{c_1, \dots, c_k\}$  where  $c_j$  is the centroid of cluster  $j$ . Each sensor can then be assigned to a cluster according to the last data reading available for this sensor. Let  $d_{i, \cdot}$  be the last value sent by sensor  $i, \forall i \in [1, n]$ , then each time a new data value is processed by the clustering algorithm, sensor  $i$  is assigned to cluster  $j$  if and only if  $dist(d_{i, \cdot}, c_j) = \min_{l=1}^k dist(d_{i, \cdot}, c_l)$  defining thus natural "boundaries" between clusters.

The idea is to use these boundaries to detect significant change in the system. For a given sensor, a new sensed data can either be *inside* or *outside* boundaries of the cluster to which the sensor was assigned regarding the previous sensed data. Making a sensor aware of the cluster boundaries it is in allows the sensor to know when its reading changes in a significant way according to the global view of the system.

At the beginning, a learning phase is needed to ensure that enough data values are available to allow the clustering algorithm to build relevant clusters.

## 2.2 On the sensor

We assume three different levels of energy consumption for a sensor: (i) a deep sleep mode where computing, sensing and radio are off, (ii) an intermediate mode where sensing and

---

**Algorithm 2** Algorithm on base station: ....

---

```

1:  $\{S[i] \in \{Assigned, Relearn\}$  state of sensor  $i$ .  

2:  $\{B[i]$  bounds of the cluster to which  $i$  is assigned.  

3:  $\{C[i]$  centroid of the cluster to which  $i$  is assigned.  

4:  $\{R[i]$  number of needed data values from sensor  $i$ .  

5: for  $i \leftarrow 1$  to  $n$  do  

6:    $R[i] \leftarrow InitNbData$  ;  $S[i] \leftarrow Relearn$   

7:    $B[i] \leftarrow [-\infty, +\infty]$  ;  $C[i] \leftarrow -\infty$   

8: end for  

9: loop  

10:  Receive(data) from sensor  $i$   

11:  OldB  $\leftarrow$   $B$ ; OldC  $\leftarrow$   $C$   

12:   $C \leftarrow UpdateClusters(d)$ ;  $B \leftarrow UpdateClusters(d)$   

13:  if  $S[i] = Relearn$  then  

14:    if  $R[i] \neq 0$  then  

15:       $R[i] \leftarrow R[i] - 1$   

16:      NextSleepTime  $\leftarrow RealerningSleepTime$   

17:    else  

18:       $S[i] = Assigned$ ;  

19:      NextSleepTime  $\leftarrow Compute(...)$   

20:    end if  

21:  else  

22:    if  $OldC[i] \neq C[i]$  then  

23:       $S[i] \leftarrow Relearn$ ;  $R[i] \leftarrow InitNbData$   

24:      for  $j$  such as  $B[j] = B[i]$  do  

25:        {for all sensor  $j$  in  $i$ 's new cluster}  

26:         $S[j] \leftarrow Relearn$ ;  $R[j] \leftarrow InitNbData$   

27:      end for  

28:      NextSleepTime  $\leftarrow 0$   

29:    else  

30:      NextSleepTime  $\leftarrow Compute(...)$   

31:    end if  

32:  end if  

33:  Send( $i, B[i], (R[i] = Relearn), NextSleepTime$ )  

34: end loop

```

---

computing are on but radio is off, (iii) lastly, the most resource consuming mode where computing, sensing and radio are on.

On sensor, the first step of the process is a *learning phase* for a fixed time interval (and relearning when required) where readings from sensors are fed into the clustering algorithm.

On each sensor the algorithm 1 is used. As a result of the learning phase (lines 6,13-18), a *sleep time* for the sensor is computed by the base station (see section 2.3) and sent to the sensor, which goes into "deep-sleep" (line 22).

On waking up, the sensor node takes a reading (line 5) and then determines if its reading has changed significantly enough to put it outside its current cluster (by comparing the reading with boundary values : lines 6 and 9 ). If not, the sensor goes back to sleep (line 22) - this is where the energy savings happen. If the change is significant (lines 10-12), the sensor reports its reading to the base station. The base station then sends new cluster information where the sensor may or may not now be assigned into a new cluster, as well as a new sleep time (line 11). Note that the first *NextSleepTime* received from the base station is the *first sleep time*. The sensor then goes back to sleep for the given time, and then wakes up to check if its reading is significant enough, and the process repeats itself.

To ensure that a sensor still reports its value from time to time to the base station, a threshold *max report time* is used. The *max report time* is the maximum amount of time given to a sensor between two consecutive reports of its data reading to the base station; if the sensor sent a reading at time  $t$  and max report time is  $r$ , it must send the next report by  $t + r$ .

## 2.3 On the base station

**Overview** On the base station algorithm 2 is used. There is an initial learning phase where the base station waits for

data readings from the sensor nodes for a fixed time interval. The base station receives these readings and feeds them into the clustering algorithm obtaining clusters of data values, and so, clusters of sensors (corresponding to the sensors the data values belong to). The base station then sends clustering information (centroid, boundaries, etc) to each sensor; the base station also computes and sends the first sleep time to each sensor. Each sensor is now aware of which cluster it belongs to and sleeps for the given time.

From the base station's perspective, a sensor can be in two different states: *Assigned* or *Relearn*. A sensor in the *Assigned* state is a sensor that has been assigned to a cluster after a relearning phase, whereas a sensor in a *Relearn* state is (i) a sensor which is to change cluster (i.e., the sensor reported to the base station that its reading now puts it outside its current cluster boundaries), or (ii) a sensor that is in the same cluster as the sensor which is to change cluster (i.e., a sensor assigned to a cluster in which a new sensor has been assigned "triggering a learning" phase).

**Computing sleeping time : line 19 and 30.** The sleeping time is computed using two different *activity* notions: the notion of *cluster activity*  $\mathcal{A}_c$ ; this activity is related to the frequency of important events for this cluster. Such events are: boundaries change and the fact that any sensor (in this cluster) changes cluster. for each sensor, we also consider the *sensor activity*  $\mathcal{A}_s$ , which is related to the frequency of important events regarding this sensor and its cluster. Such event occurs when the sensor itself changes cluster. Let  $\delta_{c,i}$  (resp.  $\delta_{s,i}$ ) be the  $i^{th}$  inter-arrival time between event  $i$  and event  $i-1$  for a given cluster (resp. sensor). At the time when event number  $k$  occurs, the activity of this cluster (resp. sensor) is defined as follows (with a total of  $(k+1)$  events, starting from event 0 to event  $k$ ):

$$\mathcal{A}_c = \sum_{i=1}^k \frac{\delta_{c,i}}{2^{(k-i)}} \quad \mathcal{A}_s = \sum_{i=1}^k \frac{\delta_{s,i}}{2^{(k-i)}}$$

This can be seen as a weighted mean of inter-arrival time giving more importance to recent events against older ones. The *NextSleepTime* for a sensor is defined as

$$\text{Min}\left(\frac{\mathcal{A}_c + \mathcal{A}_s}{2}, \text{MaxSleepTime}\right)$$

*MaxSleepTime* is an upper bound on the sleep time. The intuition behind this formula is that both cluster events and events at the sensor are taken into account. This means that each sensor can receive a sleep time tailored to it. Also, the more active the cluster or the sensor (or both), the more events there are and the smaller the inter-arrival times, and the smaller the sleep time, which means that the sensor is more active (sleep less) when more events are happening.

### 3 Implementation: Software Infrastructure

The exposed method has been implemented using the following tools: SStreaMWare, a middleware aiming at managing large heterogeneous sensors farms, an open-mining tool kit, which provide several way to cluster data. We describe the overall system (3.1) and the clustering technique we used (3.2).

#### 3.1 Architecture of our Testbed

Our system consists of four components, (i) Clustering, (ii) Controller, (iii) Middleware, and (iv) Sensor. The first three components reside in the base station, which is a PC in our case. As the name suggests, our proposed method is defined in the Controller component. The Controller is directly linked to the user and is responsible for maintaining cluster definition and user's query. To ensure a smooth data exchange, communication and translation, we deploy our service-oriented middleware called SStreaMWare [5, 6, 7] This Middleware receives a query from the Controller and evaluates the query. It is also the first component that receives sensor data readings transmitted directly from the sensor devices to the base station over a wireless link.

The data flow of the system can be described as follows. First, the query is built from the Controller and passed on to the Middleware. Second, upon receiving the query from Controller, the Middleware will evaluate and execute the query. Third, the Middleware receives data readings from the sensor nodes based on the defined query and send the relevant data to the Controller. Fourth, the Controller, after receiving data from Middleware, feeds the data to the Clustering component. Fifth, a cluster definition is then obtained and passed this on to the sensor nodes. The sensor nodes will adjust its parameters based on the given cluster definition.

#### 3.2 Clustering Technique

Different data stream clustering methods have been proposed in the literature [2]. Owing to the scarce resources of individual sensors, adaptation to availability of resources is crucial to the success of the proposed method that potentially can run on a cluster head in a physically clustered sensor network. We have adopted our *RA-Cluster* technique [3] that uses the *Granularity-based Approach* [4] to adapt to memory, battery and processing power variability in real-time. RA-Cluster is a sublinear data stream clustering technique that has proved to produce accurate results while adapting to resource availability by maintaining higher and upper bounds on the different adaptation parameters.

*RA-Cluster* processes data stream instances as they arrive. Memory adaptation is handled by changing a distance threshold that encourages or discourages the creation of new clusters. Changing the sampling rate is done in response to low availability of battery having the sensor node drains its power rapidly by continuously sensing or listening. Utilising the CPU power is done through randomisation of assignment of new instances by examining a calculated percentage of the already created clusters. Releasing outliers and inactive clusters periodically from memory are two strategies used by RA-Cluster to address the dynamic nature of the streaming data as well as efficiently using the limited available memory on-board the sensor. Memory, battery and CPU adaptations are referred as Algorithm Output Granularity (*AOG*), Algorithm Input Granularity (*AIG*) and Algorithm Processing Granularity (*APG*) respectively.

In our implementation of the proposed method, both *AIG* and *APG* have been disabled allowing the clustering technique to boost its accuracy. This is done because *RA-Cluster* runs at the base station with high availability of resources in terms of processing power. Additionally, the base station is not battery powered and is always connected to the power

source. Hence, both *APG* and *AIG* have been disabled. It is worth mentioning that if the proposed method is to run on a cluster head sensor node, both *AIG* and *APG* should be enabled.

## 4 Experimentation

| Type of data | Initial life time | life time with energy saving |
|--------------|-------------------|------------------------------|
| Luminosity   | 12 hours          | 70 hours                     |
| Temperature  | 12 hours          | 100 hours                    |

**Table 1.** Improving life time with clustering

All reported results are coming from real-life experiments. They have been conducted using a set of two sunspots <sup>7</sup>.

**Sending less and listening doesn't improve energy saving.** First set of strategies have been tested on the assumption that saving data emission from sensors to the base station will improve energy consumption. In these strategies sensors are still listening to the base station so that changes in the cluster can trigger as quickly as possible a relearning phase for involved sensors. Experiments show that these kind of strategies are not efficient on the tested sensor. This is mainly due to the fact that the cost of sending data is small compared to the cost of letting the radio turned on.

This shows that for this set of sensor, energy savings can only be done by turning the radio off. In that case, sensor can be put to sleep.

**Impact of sleeping strategy on accuracy.** The aim of this set of experiments is to analyze the difference of perception that is introduced by the fact that sensors are sleeping. A 12 hours run aims at gathering data from sensors every 2 second. The clustering algorithm is run on these data: this is the *reference*. This is what an application sees of the data without energy saving. The same set of data is replayed with energy saving algorithm in action. If the set of clusters found are not too different and none of the important events are missed, then the clustering approach for energy saving is "accurate".

Found clusters are highly dependent on clustering parameters, mainly, on the frequency of update, that is to say the number of data items that have to be waited for for an update of the cluster. When the algorithm for energy saving is running, the frequency of data sent drops and to obtain similar clusters, the frequency of update has to be increased. The value that has to be chosen for the frequency of update depends on variability of data.

That's why two sets of experiments have been conducted: one with quite steady data (temperatures) and one with highly variable data (luminosity). Figure 2 and 1 shows the resulted clusters from the associated data.

**Impact of sleeping strategy on life time** Table 1 report results of two experiments aiming at testing the improvement of the live time of a systems composed of two sensors using the proposed method. This result shows that huge improvement have been optioned using this method: the life time is between 6 and 8 time longer than without energy saving. They also show that improvement depends on the variability of data. The longest lifetime is obtained with steady data (temperature) and highly variable data (Luminosity) leads to less improvement, as expected from our sleep time formula.

<sup>7</sup> <http://www.sunspotworld.com/>

## 5 Related Work

Several approaches have been developed that focus on reducing the amount of data communicated in-network and thus, prolong sensor network lifetime. The specific type of approach that can be applied to reduce communication is dependent upon the granularity of data required for the WSN application. For instance, a critical-sensing WSN application such as smart home health care systems [13,18] would require accurate values from sensor nodes at all times to monitor patient health conditions, whereas coarse-granularity data would suffice for certain event detection systems [16,17,22]. As a consequence, various data granularity control mechanisms have been proposed to cater to the data requirements of WSN applications, utilizing approximation, prediction and use of associations/correlations. Conversely, in applications where approximations in the collected data can be tolerated, approximations on sensory data can be performed instead to further reduce data transmissions. The application of approximation to reduce data transmissions in-network has been explored in works such as [23,21,13]. The third class of techniques that can be applied to reduce network data transmissions is prediction. Prediction techniques at the node level derive spatial and temporal relationships or probabilistic models from sensory data to estimate local data readings or readings of neighbouring nodes. When sensory data of particular sensor nodes can be predicted, these sensor nodes are then suppressed from transmitting the data to save communication costs. Similar to compression and approximation, prediction-based techniques are also required to run in a light-weight manner on sensor nodes. In the literature, prediction techniques have been proposed as algorithms for enhancing data acquisition in [19,15,20] and as generic light-weight learning techniques to reduce communication costs in transmissions. In [23-26] an algorithm for learning highly correlated rules from sensory data is presented that prunes/leverages the associations to develop "context-aware" strategies for energy-efficient control of sensors.

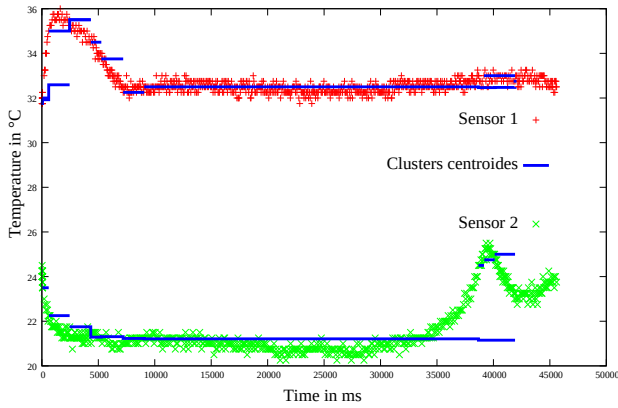
In essence, these approaches have explored how sensor data and knowledge obtained from this data can be used in some measure reduce data transmission and therefore prolong sensor lifetime. However, to the best of our knowledge none of these techniques have used changing cluster patterns as a control mechanism for energy efficient operations in a sensor network, which is the core contribution of this paper.

## 6 Conclusion and Future Work

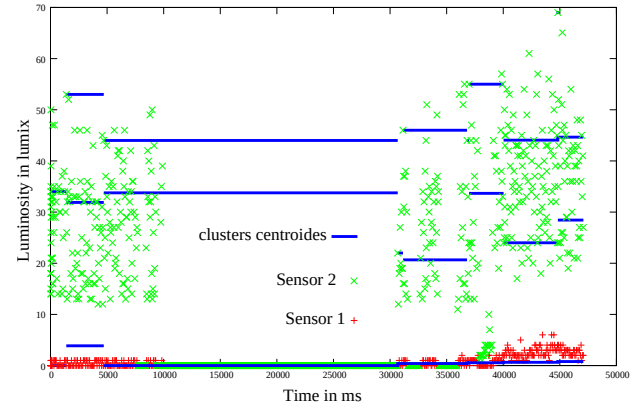
Depending on the sensor, reducing communication does not always leads to energy saving, in that case the only way to provide energy saving is by turning the radio off. Finding a good sleeping time is then crucial and results in a compromise between energy consumption and accuracy.

As shown here, clustering data from sensors can be a valuable tool to determine the sleeping time of sensor and extends the life of systems. The first results reported here are promising as the life time has been increased by 6 and 8 times the initial life time. The proposed method is quite general and can be fitted to any type of sensor on which radio can be turned off.

Some important future works are needed such as to further investigate a better way of defining cluster boundaries and clearly quantify the difference of "accuracy" introduced by the deep sleep of sensors. Impact of the two introduced



**Figure 1.** Temp. and clusters, sleeping for energy saving.



**Figure 2.** Lum. and clusters, sleeping for energy saving.

threshold on accuracy and energy saving have to be carefully studied and parameters of the clustering algorithm should be computed and adjusted dynamically according to characteristics of data streams. Clustering could also be done on the sensors themselves in order to achieve more efficient energy savings.

Other clustering techniques should also be investigated as well as the impact of the number of sensors involved in the systems.

## REFERENCES

- [1] M.X. Chen and Y.D. Wang. An efficient location tracking structure for wireless sensor networks. *Computer Communications*, 32(13-14), 2009.
- [2] M.M. Gaber, A. Zaslavsky, and S. Krishnaswamy. Mining Data Streams: A Review. *ACM SIGMOD Record*, 34(1), 2005.
- [3] M.M. Gaber, and P.S. Yu. A Holistic Approach for Resource-aware Adaptive Data Stream Mining. In *Journal of New Generation Computing*, 25(1):95-115, 2006.
- [4] M.M. Gaber. Data Stream Mining Using Granularity-based Approach. In *Foundations of Computational Intelligence Vol. 206/2009*, Springer, Germany, 2009.
- [5] L. Gurgen, C. Roncancio, C. Labbe, A. Bottaro and V. Olive. SStreamWare: a service oriented middleware for heterogeneous sensor data management. In *Proc. of The 5th ICPS*, 2008.
- [6] L. Gurgen, C. Roncancio, C. Labbe and V. Olive. Update Tolerant Execution of Continuous Queries on Sensor Data. In *Proc. of The 5th INSS*, 2008.
- [7] L. Gurgen, J. Nystrom-Person, A. Cherbal, C. Labbe, C. Roncancio, and S. Honiden. Plug-Manage Heterogeneous Sensing Devices. In *Demonstration in 6th Int'l Workshop DMSN*, 2009.
- [8] H. Liu, P.J. Wan and N. Pissinou. Maximal lifetime scheduling in sensor surveillance networks. In *Proc. of The 24th INFOCOM*, 2005.
- [9] A. Hac. *Wireless Sensor Network Designs*. John Wiley and Sons, 2003.
- [10] I. F. Akyildiz, W. Su and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38:393-422, 2002.
- [11] T. Wark, W. Hu, P. Corke, J. Hodge, A. Keto, B. Mackey, G. Foley, P. Sikka and M. Brnig". Springbrook: challenges in developing a long-term, rainforest wireless sensor network. In *Proc. of The 4th Int'l ISSNIP*, 2008.
- [12] W. S. Yeoh, W. Jiankang, I. Pek, Y. H. Yong, X. Chen and A. B. Waluyo. Real-time tracking of flexion angle by using wearable accelerometer sensors. In *Proc. of The 5th International Workshop BSN*, 2008.
- [13] C.R. Baker, K. Armijo, S. Belka, M. Benhabib, V. Bhargava, N. Burkhart, A. D. Minassians, G. Dervisoglu, L. Gutnik, M. B. Haick, C. Ho, M. Koplow, J. Mangold, S. Robinson, M. Rosa, M. Schwartz, C. Sims, H. Stoffregen, A. Waterbury, E.S. Leland, T. Pering, and P.K. Wright. Wireless sensor networks for home health care. In *21st Advanced Information Networking and Applications Workshops*, Niagara Falls, Canada, 2007.
- [14] J.Y. Chen, G. Pandurangan, and D. Xu. Robust computation of aggregates in wireless sensor networks: Distributed randomized algorithms and analysis. In *4th International Symposium on Information Processing in Sensor Networks*, California, USA, 2005.
- [15] B. Gedik, L. Liu, and P.S. Yu. Asap: An adaptive sampling approach to data collection in sensor networks. *IEEE Transactions on Parallel and distributed systems*, 18(12), 2007.
- [16] T. He, S. Krishnamurthy, J.A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh. Energy-efficient surveillance system using wireless sensor networks. In *2nd International Conference on Mobile Systems, Applications and Services*, pages 270-283, Boston, USA, 2004.
- [17] M. Hefeeda and M. Bagheri. Wireless sensor networks for early detection of forest fires. In *IEEE International Conference on Mobile Adhoc and Sensor Systems*, pages 1-6, Pisa, Italy, 2007.
- [18] A. Keshavarz, A.M. Tabar, and H. Aghajan. Distributed vision-based reasoning for smart homecare. In *ACM SensSys Workshop on Distributed Smart Cameras*, pages 105-109, Boulder, USA, 2006.
- [19] S. Madden and M.J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In *18th ICDE*, San Jose, USA, 2002.
- [20] M.A. Sharaf, J. Beaver, A. Labrinidis, and P.K. Chrysanthis. Tina: A scheme for temporal coherency-aware in-network aggregation. In *ACM Workshop on Data Engineering for Wireless and Mobile Access*, pages 69-76, California, USA, 2003.
- [21] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and beyond: New aggregation techniques for sensor networks. In *ACM Conference on Embedded Networked Sensor Systems*, pages 239-249, Baltimore, USA, 2004.
- [22] A. Tavakoli, J. Zhang, and S.H. San. Group-based event detection in undersea sensor networks. In *2nd International Workshop on Networked Sensing Systems*, San Diego, USA, 2005.
- [23] J. Zhao, R. Govindan, and D. Estrin. Computing aggregates for monitoring wireless sensor networks. In *1st IEEE International workshop on sensor network protocols and applications*, California, USA, 2003.
- [24] S.K. Chong, M.M. Gaber, S. Krishnaswamy, and S.W. Loke. A rule learning approach to energy efficient clustering in wireless sensor networks. In *2nd International Conference on Sensor Technologies and Applications*, Cap Esterel, France, 2008.
- [25] S.K. Chong, S. Krishnaswamy, and S.W. Loke. A context-aware approach to conserving energy in wireless sensor networks. In *3rd IEEE International Conference on Pervasive Computing and Communications Workshops*, Hawaii, USA, 2005.
- [26] S.K. Chong, I. McCauley, S.W. Loke, and S. Krishnaswamy. Application of context-awareness to data muling. In *1st ACM International Workshop on Context-Awareness for Self-Managing Systems*, Toronto, Canada, 2007.



# Size and Speed Related Characteristic of Concept Drift

Petr Kosina<sup>1</sup> and João Gama<sup>2</sup> and Raquel Sebastião<sup>3</sup>

**Abstract.** Stream mining draws more and more attention in data mining since the volume of available data becomes too large and problematic for conventional mining tools. One typical feature of data stream is a change over time in the underlying data distribution. This feature is particularly interesting for studying, and vital for quick adaptation to the new situation.

The focus of this paper is in small detail in the process of drift detection - the connection between the area affected by the change and number of errors. The goal is to provide the information about the size and speed characteristics of the change to the user.

**Keywords:** Data Streams, Concept Drift, Drift Rate

## 1 Motivation

In many situations where there is a vast amount of data, such as in biomedicine, fraud detection, fault detection in industrial processes etc., there is also a need for tools that are taking into account the change detection. Such applications are of high importance and interest. Therefore the change in data has been subject of research in many works, which are dealing with data stream mining, mostly in a pursuit of fast and adequate reaction.

The process of change does not necessarily need to be always the same, for example weather in the mountains can change very quickly to completely different conditions - from warm, sunny to chilling and snowing - whereas in some lowland places the conditions are quite stable during the whole season. That suggests that the ability and complexity to detect a difference in the underlying concept varies. In this paper, we were interested in the speed and amount of change between different concepts in data e.g. its rate. We introduce a simple, yet effective method for measuring the characteristic of drift to describe the difficulty of concept drift recognition with respect to the number of examples needed to be seen before the detection.

The rest of the paper is organized as follows. In the next section related work on concept drift detection in data stream is presented. In section 3 we described the drift detection method we used and also our method for measuring the drift characteristic. The experiments are evaluated in the section 4 and finally section 5 concludes the paper.

## 2 Related Work

Concepts drift is a topic in many existing works dealing with data stream mining [5, 13, 15]. In [13] the phenomenon of concept drift is defined as number of instances  $\Delta x$  between two stable concepts. When  $\Delta e = 1$  drift is instantaneous and if  $\Delta e > 1$  the concept

is changing over a number of instances. The higher  $\Delta e$  the more gradual (slower) the drift is. Besides the instantaneous drift, there are also presented two subsets of gradual drift - moderate and slow.

The rest of the related work can be grouped into two main categories. The first one is drift detection methods and the second is severity and rate of drift definitions and works.

### 2.1 Drift Detection Methods

**DDM.** The first drift detection method used in experiments in this paper was proposed in [5]. It assumes that if the distribution of the examples is stationary, the error-rate of the learning algorithm will decrease when the number of examples increases.

Suppose the sequence of examples is in the form of pairs  $(\mathbf{x}_t, y_t)$  and the prediction of the model  $(\hat{y}_t)$  can be either True ( $\hat{y}_t = y_t$ ) or False ( $\hat{y}_t \neq y_t$ ). For each point  $i$  in the sequence, the error-rate is the probability,  $p_t$ , of observing False in a sample of  $n$  examples with standard deviation given by  $s_t = \sqrt{p_t(1 - p_t)/t}$ .

A significant increase in the error signals the change in class distribution. For a sufficient large number of the examples, the Binomial distribution is closely approximated by a Normal distribution with the same mean and variance. Considering that the probability distribution is unchanged when the context is static, then the  $1 - \alpha/2$  confidence interval for  $p$  with  $n > 30$  examples is approximately  $p_t \pm \alpha * s_t$ . The parameter  $\alpha$  depends on the confidence level. This drift detection method manages two registers during the training of the learning algorithm,  $p_{min}$  and  $s_{min}$ . Every time a new example  $t$  is processed those values are updated when  $p_t + s_t$  is lower than  $p_{min} + s_{min}$ . We use a warning level to define the optimal size of the context window. The context window will contain the old examples that are on the new context and a minimal number of examples on the old context. Suppose that in the sequence of examples there is an example  $i$  with correspondent  $p_t$  and  $s_t$ . In the experiments the confidence level for warning has been set to 95 %, that is, the warning level is reached if  $p_t + s_t \geq p_{min} + 2 * s_{min}$ . The confidence level for drift has been set to 99 %, that is, the drift level is reached if  $p_t + s_t \geq p_{min} + 3 * s_{min}$ . The Algorithm 1 contains the pseudo-code for the drift detection system.

Suppose a sequence of examples where the error of the actual model increases reaching the warning level at example  $e_w$ , and the drift level at example  $e_d$ . This is an indication of a change in the distribution of the examples. A new context is declared starting in example  $e_w$ , and a new decision model is induced using only the examples starting in  $e_w$  till  $e_d$ . It is possible to observe an increase of the error reaching the warning level, followed by a decrease. We assume that such situations correspond to a false alarm, without changing the context.

**Page-Hinkley Test.** Another drift detection method is based on Page-Hinkley (PH) statistics [11, 7, 9], which are commonly used to

<sup>1</sup> LIAAD-INESC Porto, FI-Masaryk University, Brno email: soso@mail.muni.cz

<sup>2</sup> LIAAD-INESC Porto, FEP-University of Porto email: jgama@fep.up.pt

<sup>3</sup> LIAAD-INESC Porto, FC-University of Porto email: raquel@liaad.up.pt

---

**Algorithm 1** Drift Detection Algorithm

---

**Require:**  $p_t$  and  $s_t$  {error-rate and standard deviation at time  $t$ }**Ensure:**  $state$  {state of classifier}

```
if  $p_t + s_t < p_{min} + s_{min}$  then
   $p_{min} \leftarrow p_t$ 
   $s_{min} \leftarrow s_t$ 
end if
if  $p_t + s_t \geq p_{min} + 2 * s_{min}$  then
   $state \leftarrow warning$ 
  if  $p_t + s_t \geq p_{min} + 3 * s_{min}$  then
     $state \leftarrow drift$ 
  end if
else
   $state \leftarrow stationary$ 
end if
return  $state$ 
```

---

---

**Algorithm 2** Page-Hinkley Test Algorithm

---

**Require:**  $p_t$  and  $n_t$  {error-rate and number of example at time  $t$ } $\delta, \lambda_w, \lambda_d$  {user defined constant and thresholds}**Ensure:**  $state$  {state of classifier}

```
 $mean \leftarrow ((n_t - 1) * mean + p_t) / n_t$ 
 $m_t \leftarrow m_t + p - mean - \delta$ 
if  $m_t \leq M_t$  then
   $M_t \leftarrow m_t$ 
end if
if  $m_t - M_t > \lambda_w$  then
   $state \leftarrow warning$ 
  if  $m_t - M_t > \lambda_d$  then
     $state \leftarrow drift$ 
  end if
else
   $state \leftarrow stationary$ 
end if
return  $state$ 
```

---

detect a change in signal processing. The PH statistical test monitors the difference between two variables  $m_t$  and  $M_t$ .

Variable  $m_t$  is the cumulated difference between observed values (error-rate in our case) and their mean till the current time  $t$ :  $m_t = \sum_{i=1}^t (p_i - \bar{p} - \delta)$  where  $\bar{p} = \frac{1}{t} \sum_{i=1}^t p_i$  and parameter  $\delta$  corresponds to the magnitude of change that should not raise an alarm.

The second variable,  $M_t$ , is then defined as the minimum observed value of  $m_t$  e.g.  $M_t = \min(m_i, i = 1 \dots t)$ .

The change in the distribution is reported when the monitored difference of the two variables is above a given threshold  $\lambda$ :  $m_t - M_t > \lambda$ . The value of parameter  $\lambda$  determines the rate of false alarms. Increasing the value will entail less false alarms but some changes could be missed.

For the purpose of measuring the rate we used two different threshold values. The first,  $\lambda_w$ , signals the warning phase and the second,  $\lambda_d$ , signals the drift in data as shown in the Algorithm 2.

## 2.2 Rate and Severity of Drift

**Drift rate.** The drift rate in [1, 2] is defined as the probability of two consecutive concept functions to provide different label on random

example  $Prob(f_t \neq f_{t+1})$ , where  $f_t = E \rightarrow \{0, 1\}$  assigns the label 0 or 1 to the referred example  $e_t$  from some domain  $E$  at time  $t$ .

In [3] drift rate refers to the change of angle per time step. The data distribution in their example can be depicted as checker-board with static sampling window. The checker-board distribution rotates thus changing the distribution of data in sampling window and causing a drift. Three cases of rate of drift can be distinguished then: constant, exponentially increasing and sinusoidal (rate drops to zero - short pause in rotation - and rises again). The rate measure is relevant to this very dataset, because it is reflecting the rotating nature of the distribution.

**Severity.** In the paper [10] we can encounter the term severity of drift. Suppose examples of context  $C_i$  and examples of the following context  $C_{i+1}$  are from some input space  $S$ . During the drift the examples from both contexts,  $C_i$  and  $C_{i+1}$ , can appear. The severity of drift is defined as the percentage of  $S$  that has its class label changed after the drift is complete, e.g. examples of  $C_{i+1}$  are emitted.

## 3 The Measure

### 3.1 Measuring the Characteristic

The drift rate of evolving data significantly affects the detection difficulty. For some artificial datasets we are able to compute the severity or we know the information about speed, but not many works provide more detailed information about the event and we lack measure that could be used for different datasets.

According to the speed, drifts are commonly divided into two basic groups: abrupt and gradual. In abrupt drift the amount of change is immediately present whereas in gradual drift the change is distributed over longer period of time, which can be considered as very small amount of immediate change. A value describing the rate of change may be of use in studying and comparing the behaviour of algorithms on different datasets and it may also reveal some interesting characteristics. This measure shows how difficult is for the system to recognize that there was a drift.

The drift detection methods from the previous part have two states relevant for the speed and size related characteristic. The first one is a warning state. This period is indicating that there is some change and it is the time during which we keep simple statistics about the number of processed examples and the number of misclassified ones. Once we reach the second state e.g. drift is detected and the change in data is confirmed, we easily compute the rate as  $\frac{\#misclassified}{\#examples\_in\_warning}$ . Only statistics from warning phase that was followed by drift are taken into account, therefore the examples which were a cause of false alarm are not included in the computation.

## 4 Experiments

In this section we describe the datasets and present the results obtained from the experiments. There are two drift detection methods to be applied together with two different classifiers. First classifier is an incremental version of Naive Bayes [6] and the second is Hoeffding Tree [4] as second. The settings of the Hoeffding Tree classifier is following:  $\delta = 10^{-6}$ ,  $\tau = 0.05$ ,  $graceperiod = 200$ , where  $\delta$  is the split confidence constant,  $\tau$  is the tie breaking constant and grace period stands for number of examples that has to be processed before computing the split test. Numeric attributes are handled by exhaustive method with binary tree structures introduced in [4].

The PH test detection is used only with Naive Bayes classifier since the slower learning process of Hoeffding Tree in some of the

datasets did not work with it. In the experiments the values of PH test parameters are set as follows:  $\delta = 0.001$ ,  $\lambda_d = 2.5$ , and  $\lambda_w = 2$ . The value of  $\lambda_w$  is set to be 80 % of  $\lambda_d$ .

## 4.1 Experimental Methodology

In this section we describe the metrics used in the experiments and how to measure them. The design of the experiments that concern about the speed of drift we use following strategy. Assume that there are  $2 * N$  examples in the dataset,  $N$  for old concept,  $N - DriftLength$  for new concept, where  $DriftLength$  is desired number of examples before the drift is complete. During the drift the examples from new concept are generated with probability:  $p_{new}(t) = \frac{t-N}{DriftLength}$ , and examples from old concept with:  $p_{old}(t) = 1 - p_{new}(t)$ , where  $N < t < N + DriftLength$ . There are three speed levels in these experiments: Low, Med, and Hi corresponding to 1, 0.25\*N, and 0.5\*N respectively.  $N$  is set to 4,000 examples. The severity levels Low and Hi stands for 15 % and 85 % respectively. The metrics for measuring the rate and severity of drift are based on: The *WarningPoint* in the experiments is the time when warning followed by drift is first reported and similarly *DriftPoint* is the time when drift is reported. The *Difference* then corresponds to the length of warning e.g.  $DriftPoint - WarningPoint$ . In the warning state the slope can be measured. It is the slope of the line between the value of the statistic test at the *WarningPoint* and at *DriftPoint* as:

$$\arctan\left(\frac{ph(DriftPoint) - ph(WarningPoint)}{Difference}\right)$$

where  $ph(t)$  is the value of PH test for PH or the value of  $p(t) + s(t)$  for DDM at time  $t$

The *AvgTimeError* is a statistic for average number of correctly classified examples between two consecutive misclassified examples in the warning state.

The severity of drift in SEA dataset is computed as follows. The two relevant attributes of the data defines two-dimensional area. The threshold line divides the area into two parts, one for each class. By subtracting the areas defined by two consecutive concept thresholds we obtain the area affected by drift.

## 4.2 Datasets

The 8 artificial datasets used in the experiments are described below. First 6 were taken from the [5], SEA Concepts were generated based on [14], STAGGER and LED was generated by [12].

1. **SINE1.** Abrupt concept drift, noise-free examples. The dataset had two relevant attributes. Each attribute had values uniformly distributed in  $[0, 1]$ . In the first context all points below the curve  $y = \sin(x)$  were classified as positive. After the context change the classification was reversed. Concept length: 1,000 examples.
2. **SINE2.** The same two relevant attributes. The classification function was  $y < 0.5 + 0.3 * \sin(3\pi x)$ . After the context changed the classification was reversed. Concept length: 1,000 examples.
3. **SINIRREL1.** Presence of irrelevant attributes. The same classification function of SINE1 but the examples had two more random attributes with no influence on the classification function. Concept length: 1,000 examples.
4. **SINIRREL2.** The same classification function of SINE2 but the examples had two more random attributes with no influence on the classification function. Concept length: 1,000 examples.

5. **MIXED.** Abrupt concept drift, boolean noise-free examples. Four relevant attributes, two boolean attributes  $v, w$  and two numeric attributes from  $[0, 1]$ . The examples were classified positive if two of three conditions were satisfied:  $v, w, y < 0.5 + 0.3 * \sin(3\pi x)$ . After each context change the classification was reversed. Concept length: 1,000 examples.
6. **SEA.** Abrupt concept drift, 10 % noise. Random points in a three-dimensional feature space with two classes. All three features had values between 0 and 10, where just the first two attributes were relevant. Data point belonged to class 1 if  $f_2 + f_3 \leq \Theta$ , where  $f_2$  and  $f_3$  represented the first two features and  $\Theta$  was a threshold value between the two classes. Many different thresholds were used with concept length: 45,000 examples.
7. **LED.** Abrupt, 10% noise. 24 attributes from which 7 were relevant and they represented digits on a 7-segment display. Each attribute had the value 1 or 0 indicating if the LED was on or off. Drift was caused by changing the number of attributes with drift. Concept length: 250,000 examples.
8. **STAGGER.** Abrupt concept drift, symbolic noise-free examples. Instances had three categorical attributes with values as follows: **size** = {small, medium, large}; **color** = {red, green, blue}; **shape** = {square, circular, triangular}. There were three concepts defined as (1) **size** = small and **color** = red, (2) **color** = green or **shape** = circular and (3) **size** = medium or large. Concept length: 20,000 examples.

## 4.3 Results

This part of the paper will provide some of the experimental results to support the suggested dependence between the area of drift and the number of errors.

The SEA datasets, that were generated with many different thresholds  $\Theta$ , gave us an additional information to examine. The information is the area difference of two concepts represented by the severity. The proposed measure does not measure the severity, however it is highly correlated to severity as could be seen in Tab. 1. Fig. 1 depicts the dependency.

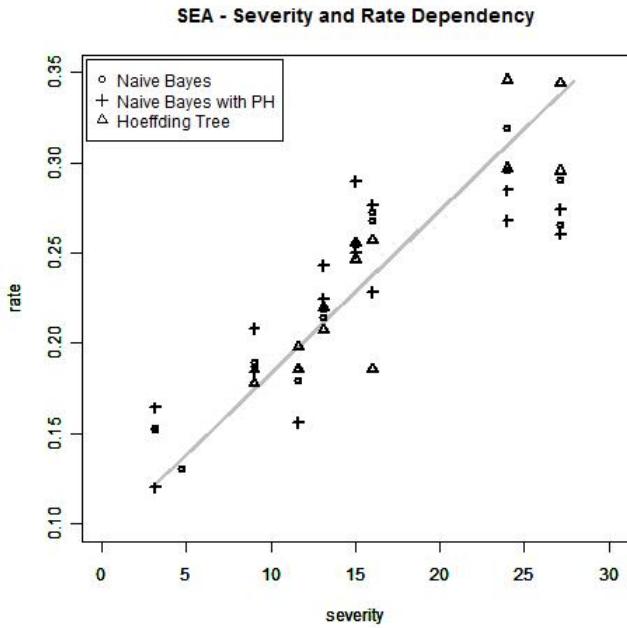
Considering speed and severity studies in the Tables 2,3,4 we can see that the warning length indicates the severity. It is longer in low severity data and shorter in high severity data. The measure is influenced by both speed and severity, but the dependency is stronger to the severity.

Datasets from [5] had different number of concepts but each concept had 1,000 examples. The gradual drift in CIRCLES dataset was problematic especially for Hoeffding trees, since the number of examples was quite small considering the grace period was set to 200. There were some unrecognized concept drifts using the trees as well as Naive Bayes with PH detection. In Tab. 5 could be seen the results of the four datasets based on the sine function, which have abrupt change in concepts and have high severity due to the fact that the class labels are reversed. The rate in these sets is mostly very high indicating that the change was detected very quickly and easily. It also shows that the deviation of rate values with different approaches and even between different concepts is much bigger than in the cases with smaller rate. That is caused by the fact that the warning phase is very short and every misclassification causes bigger change in rate.

LED dataset consisted of two-concept sets with following tuples, which represents numbers of drift attributes:  $[2, 3]; [2, 4]; [2, 5]; [3, 4]; [3, 5]; [4, 5]$  respectively. Each concept had 250,000 examples. Since there were 7 equally important attributes and 10 class values, it would require many examples to

**Table 1.** SEA Concepts results showing the dependency of the characteristic measure on severity on two datasets with the same threshold  $\Theta$  values.

| Severity | Rate           |                |                | $\Theta$ |     |
|----------|----------------|----------------|----------------|----------|-----|
|          | Hoeff. Tree    | Naive Bayes    | NB with PH     |          |     |
| 3.125    | -              | 0.15298        | 0.16431        | 6        | 6.5 |
|          | -              | 0.15206        | 0.12           |          |     |
| 4.725    | -              | -              | -              | 9        | 8.5 |
|          | -              | 0.13034        | -              |          |     |
| 9        | -              | 0.18715        | 0.20792        | 8.5      | 9.5 |
|          | 0.17755        | 0.18966        | 0.18396        |          |     |
| 11.625   | 0.18538        | 0.18557        | 0.15589        | 8.5      | 7   |
|          | 0.19795        | 0.17957        | 0.15601        |          |     |
| 13.125   | 0.20737        | 0.21883        | 0.24324        | 8        | 9.5 |
|          | 0.21964        | 0.21418        | 0.22436        |          |     |
| 15       | 0.25545        | 0.25440        | 0.25000        | 6.5      | 8.5 |
|          | 0.24650        | 0.25549        | 0.28966        |          |     |
| 16       | 0.25710        | 0.26804        | 0.22819        | 7        | 9   |
|          | 0.18538        | 0.27254        | 0.27684        |          |     |
| 24       | 0.34591        | 0.31944        | 0.26772        | 9.5      | 6.5 |
|          | 0.29712        | 0.29620        | 0.28467        |          |     |
| 27.125   | 0.34383        | 0.29018        | 0.26050        | 9.5      | 6   |
|          | 0.29541        | 0.26596        | 0.27397        |          |     |
| corr     | <b>0.90345</b> | <b>0.90766</b> | <b>0.78851</b> |          |     |



**Figure 1.** The dependency of the measure on area in SEA data set from Tab. 1 showing the high linear correlation.

**Table 2.** Results on SEA dataset with different speed and severity levels for Hoeffding Tree with Drift Detection Method.

| WarningPoint | severity |       |
|--------------|----------|-------|
|              | Low      | Hi    |
| speed        | 2,293    | 2,517 |
| Low          | 1,855    | 2,461 |
| Med          | -        | 2,028 |
| Hi           | -        | -     |

| DriftPoint | severity |       |
|------------|----------|-------|
|            | Low      | Hi    |
| speed      | 2,862    | 2,681 |
| Low        | 2,397    | 2,793 |
| Med        | -        | 2,067 |
| Hi         | -        | -     |

| Difference | severity |     |
|------------|----------|-----|
|            | Low      | Hi  |
| speed      | 569      | 164 |
| Low        | 542      | 332 |
| Med        | -        | 39  |
| Hi         | -        | -   |

| Rate  | severity |       |
|-------|----------|-------|
|       | Low      | Hi    |
| speed | 0.367    | 0.451 |
| Low   | 0.334    | 0.614 |
| Med   | -        | 0.769 |
| Hi    | -        | -     |

| AvgTimeError | severity |       |
|--------------|----------|-------|
|              | Low      | Hi    |
| speed        | 1.731    | 1.233 |
| Low          | 2.006    | 0.631 |
| Med          | -        | 0.310 |
| Hi           | -        | -     |

| Slope | severity |         |
|-------|----------|---------|
|       | Low      | Hi      |
| speed | 0.00003  | 0.00006 |
| Low   | 0.00002  | 0.00011 |
| Med   | -        | 0.00025 |
| Hi    | -        | -       |

**Table 3.** Results on SEA dataset with different speed and severity levels for Naive Bayes with Drift Detection Method.

| WarningPoint | severity |       |
|--------------|----------|-------|
|              | Low      | Hi    |
| speed        | 2,409    | 2,228 |
| Low          | 2,494    | 2,171 |
| Med          | 2,139    | 2,020 |
| Hi           | -        | -     |

| DriftPoint | severity |       |
|------------|----------|-------|
|            | Low      | Hi    |
| speed      | 2,647    | 2,358 |
| Low        | 2,770    | 2,216 |
| Med        | 2,249    | 2,056 |
| Hi         | -        | -     |

| Difference | severity |     |
|------------|----------|-----|
|            | Low      | Hi  |
| speed      | 238      | 130 |
| Low        | 276      | 45  |
| Med        | 110      | 36  |
| Hi         | -        | -   |

| Rate  | severity |       |
|-------|----------|-------|
|       | Low      | Hi    |
| speed | 0.206    | 0.277 |
| Low   | 0.210    | 0.489 |
| Med   | 0.282    | 0.75  |
| Hi    | -        | -     |

| AvgTimeError | severity |       |
|--------------|----------|-------|
|              | Low      | Hi    |
| speed        | 3.896    | 2.657 |
| Low          | 3.754    | 1.095 |
| Med          | 2.600    | 0.346 |
| Hi           | -        | -     |

| Slope | severity |         |
|-------|----------|---------|
|       | Low      | Hi      |
| speed | 0.00003  | 0.00006 |
| Low   | 0.00003  | 0.00016 |
| Med   | 0.00007  | 0.00029 |
| Hi    | -        | -       |

**Table 4.** Results on SEA dataset with different speed and severity levels for Naive Bayes with Page-Hinkley Test.

| WarningPoint | severity |       |
|--------------|----------|-------|
|              | Low      | Hi    |
| speed        | 3,074    | 2,650 |
| Low          | 3,115    | 2,435 |
| Med          | 2,998    | 2,164 |
| Hi           | -        | -     |

| DriftPoint | severity |       |
|------------|----------|-------|
|            | Low      | Hi    |
| speed      | 3,131    | 2,671 |
| Low        | 3,163    | 2,452 |
| Med        | 3,054    | 2,178 |
| Hi         | -        | -     |

| Difference | severity |    |
|------------|----------|----|
|            | Low      | Hi |
| speed      | 57       | 21 |
| Low        | 48       | 17 |
| Med        | 56       | 14 |
| Hi         | -        | -  |

| Rate  | severity |       |
|-------|----------|-------|
|       | Low      | Hi    |
| speed | 0.211    | 0.524 |
| Low   | 0.188    | 0.756 |
| Med   | 0.196    | 0.786 |
| Hi    | -        | -     |

| AvgTimeError | severity |       |
|--------------|----------|-------|
|              | Low      | Hi    |
| speed        | 3.583    | 0.727 |
| Low          | 3.556    | 0.231 |
| Med          | 4.000    | 0.300 |
| Hi           | -        | -     |

| Slope | severity |         |
|-------|----------|---------|
|       | Low      | Hi      |
| speed | 0.00880  | 0.02334 |
| Low   | 0.01051  | 0.02910 |
| Med   | 0.00897  | 0.03440 |
| Hi    | -        | -       |

train the Hoeffding Tree with our default settings. We decided to set the options to higher values because this experiment was focused on the change, not on speed of learning, therefore we could afford this setting:  $\tau = 0.15$ ;  $\delta = 10^{-5}$ . The results in Tab. 5 show the expected behaviour - the larger the difference between concepts (e.g. the severity represented by number of attributes) the higher the rate.

STAGGER had 20,000 examples in one concept and only three concept drifts which were taken in the following order: concept (1) and (2), (1) and (3), and (2) and (3) as defined previously. This dataset has abrupt change with very different concepts which is also confirmed by the high rate results.

**Table 5.** Rate of drift results for Naive Bayes and Hoeffding Tree classifiers on different datasets.

|           |       |       |            |
|-----------|-------|-------|------------|
| SINE1     | HT    | NB    | NB with PH |
|           | 0.808 | 0.889 | 0.889      |
|           | 0.917 | 0.917 | 0.889      |
|           | 0.778 | 1.0   | 1.0        |
|           | 0.818 | 1.0   | 0.889      |
|           | 0.92  | 0.588 | 0.889      |
| SINE2     | HT    | NB    | NB with PH |
|           | 0.808 | 0.455 | 0.9        |
|           | 0.65  | 0.606 | 0.7        |
|           | 0.417 | 1.0   | 0.8        |
|           | 0.75  | 0.813 | 0.875      |
|           | 0.833 | 0.789 | 0.8        |
| SINIRREL1 | HT    | NB    | NB with PH |
|           | 0.913 | 1.0   | 0.889      |
|           | 0.846 | 1.0   | 1.0        |
|           | 0.694 | 1.0   | 1.0        |
|           | 0.678 | 0.9   | 1.0        |
|           | 0.95  | 0.818 | 0.778      |
| SINIRREL2 | HT    | NB    | NB with PH |
|           | 0.944 | 0.489 | 0.9        |
|           | 0.8   | 1.0   | 0.9        |
|           | 0.857 | 0.882 | 0.9        |
|           | 0.71  | 0.773 | 0.818      |
|           | 0.857 | 0.941 | 0.9        |
| MIXED     | HT    | NB    | NB with PH |
|           | 0.59  | 0.75  | 0.308      |
|           | 0.59  | 0.36  | 0.571      |
|           | 0.541 | 0.5   | 0.5        |
|           | 0.571 | 0.545 | 0.417      |
|           | 0.56  | 0.389 | 0.462      |
| LED       | HT    | NB    | NB with PH |
|           | 0.408 | 0.401 | 0.339      |
|           | 0.486 | 0.493 | 0.444      |
|           | 0.651 | 0.613 | 0.565      |
|           | 0.343 | 0.341 | 0.312      |
|           | 0.55  | 0.521 | 0.508      |
|           | 0.475 | 0.433 | 0.471      |
| STAGGER   | HT    | NB    | NB with PH |
|           | 0.59  | 0.5   | 0.674      |
|           | 0.434 | 0.8   | 0.795      |
|           | 0.462 | 0.6   | 0.468      |

## 5 Conclusions

In this paper we introduced a measure that is simple, yet effective, indicator of amount and speed of change between different concepts in

data. We experimentally proved on artificial datasets that the higher the rate measured is the bigger the change between concepts is and the easier it is to recognize the drift. It was also supported by high correlation of the obtained value to the severity of drift, .

This measure provides valuable information for user about the characteristics of the changes that are hidden in their data. It may be useful in reaction to concept drift when selecting appropriate single classifier or classifier to ensemble. Such use could be subject for future work.

## Acknowledgements

This work was supported by the project Knowledge Discovery from Ubiquitous Data Streams ( PTDC/EIA-EIA/098355/2008 ). The work of Raquel Sebastião is supported by the Portuguese Foundation for Science and Technology (FCT) under the PhD Grant SFRH/BD/41569/2007.

## REFERENCES

- [1] Bartlett, P. L., Ben-David, S., and Kulkarni, S. R. *Learning Changing Concepts by Exploiting the Structure of Change*. Mach. Learn. 41, 2 (Nov. 2000), 153–174.
- [2] Dwi H. Widyantoro, John Yen. *Relevant Data Expansion for Learning Concept Drift from Sparsely Labeled Data*. IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 3, pp. 401–412, March, 2005.
- [3] Elwell, Ryan and Polikar, Robi. *Incremental Learning of Variable Rate Concept Drift*. MCS '09: Proceedings of the 8th International Workshop on Multiple Classifier Systems, 2009, page 142–151, Springer-Verlag Berlin Heidelberg.
- [4] Gama, João and Fernandes, Ricardo and Rocha, Ricardo. *Decision Trees for Mining Data Streams*. Intelligent Data Analysis, IOS Press, Pages 23-45, 2006.
- [5] Gama João, Medas Pedro, Gladys Castillo, and Rodrigues Pedro. *Learning with Drift Detection*. Proc. of the 17th Brazilian Symposium on Artificial Intelligence (SBIA 2004), Volume 3171, page 286–295 - October 2004.
- [6] Gama, João, Gaber, Mohamed Medhat (Eds.). *Learning from Data Streams: Processing Techniques in Sensor Networks*. Springer, 2007.
- [7] Gama, J., Sebastião, R., and Rodrigues, P. P. *Issues in evaluation of stream learning algorithms*. In Proceedings of the 15th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining (Paris, France, June 28 - July 01, 2009). KDD '09. ACM, New York, NY, 329-338.
- [8] M. Harries. *Splice-2 Comparative Evaluation: Electricity Pricing*. Technical Report UNSW-CSE-TR-9905, Artificial Intelligence Group, School of Computer Science and Engineering, The University of New South Wales, Sydney, Australia, July 1999.
- [9] C. Hartland, S. Gelly, N. Baskiotis, O. Teytaud and M. Sebag. *Multi-Armed Bandit, Dynamic Environments and Meta-Bandits*. NIPS 2006 Workshop, On-line Trading of Exploration and Exploitation.
- [10] Minku, F. L. and Yao, X. *Using diversity to handle concept drift in on-line learning*. In Proceedings of the 2009 international Joint Conference on Neural Networks (Atlanta, Georgia, USA, June 14 - 19, 2009). IEEE Press, Piscataway, NJ, 2968-2975.
- [11] E. S. Page. *Continuous inspection schemes*. Biometrika, 41(1/2):100–115, 1954.
- [12] Kirkby Richard. *Massive Online Analysis*. University of Waikato, Hamilton, New Zealand, 2007. <http://sourceforge.net/projects/moa-datastream/>
- [13] Stanley K.O., *Learning concept drift with a committee of decision trees*. Tech. Report UTAI- TR-03-302, Department of Computer Sciences, University of Texas at Austin, USA, 2003.
- [14] W.Nick Street and YongSeog Kim. *A streaming ensemble algorithm SEA for large-scale classification*. In Proc. seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pages 377-382. ACM Press, 2001.
- [15] Tsymbal, Alexey. *The Problem of Concept Drift: Definitions and Related Work*. Department of Computer Science, Trinity College Dublin, Ireland, 2004.



# Towards Symbiotic Spam E-mail Filtering

Clotilde Lopes<sup>1</sup> and Paulo Cortez<sup>1</sup> and Pedro Sousa<sup>2</sup>

**Abstract.** This position paper discusses the use of symbiotic filtering, a novel distributed data mining approach that combines content-based and collaborative filtering for spam detection.

## 1 INTRODUCTION

Unsolicited e-mail (spam) is a serious problem, as it consumes resources (e.g. time spent reading unwanted messages) and it is also used to spread malicious content (e.g. viruses). Currently, spam represents around 80-90% of all email messages sent [11].

Several solutions have been proposed to fight spam under two main categories [6]: Collaborative Filtering (CF) and Content-Based Filtering (CBF). CF involves sharing information about spam emails messages (e.g. blacklists with IP addresses of known spammers). The use of social networks has also been proposed to increase the CF potential (e.g. propagation of whitelists among socially connected users) [6]. CBF is the most used anti-spam solution and it is based on using a text classifier (e.g. Naive Bayes) that learns to detect spam from message features (e.g. word frequencies) that are extracted from the past messages of a given email account [9].

Both pure CBF and CF solutions have drawbacks. CBF performance is poor for new users, as it requires a large number of representative examples. Also, there may be a large gap between high-level concept (e.g. spam image) and the low-level message features (e.g. bit colors). Moreover, CBF is vulnerable to contamination attacks, where spammers mix spam with normal words. On the other hand, CF suffers from first-rater, i.e. difficulty to classify emails that have not been rated before, and sparsity of data, i.e. when users rate few items. Also, spam is ultimately a personal concept and often CF systems discard this issue [7]. In recent work, we proposed the novel Distributed Data Mining (DDM) approach, called Symbiotic Filtering (SF), which combines useful features from both CBF and CF [4]. In this position paper, we discuss several SF issues.

## 2 SYMBIOTIC FILTERING (SF)

Symbiosis is a close interaction among different entities and this phenomenon is present not only in biological species but also in business enterprises. Under the Web 2.0 concept, the idea is to use the Internet to gather distinct users interested on similar but not identical Data Mining (DM) goals. While SF could be applied to other personalized filtering applications (e.g. web pages with offensive content), we discuss here the case of spam e-mail filtering, since it involves several challenging issues (e.g. concept drift and privacy). It is assumed that each individual runs a local CBF (e.g. Naive Bayes). Instead of sharing data (which rises privacy issues), SF exchanges information

learned locally (e.g. the filter). This reduces communication costs, while preserving privacy. The final SF goal is to foster mutual relationships, where all (or most of) users benefit from the collaboration.

SF puts emphasis on accessing (indirectly) more data rather than using more complex local filters. To achieve SF, there are two interesting sharing possibilities: relevant features or DM models (i.e. filters). In this paper, we will discuss the latter approach, which can be achieved by using ensembles, where a combination function is used to aggregate distinct predictions into a single response. As spam suffers from concept drift (i.e. the learning concept evolves through time), such aggregation function should be dynamic. In [4], we proposed a hierarchical learning ensemble, where the outputs of the distinct filters are used as the inputs of another (meta-level) learner. Hence, each user has a local meta-learner that is dynamically trained (e.g. each time a new filter is received) to get a high accuracy on the user recent past data (e.g. last 100 emails). Such approach was successfully applied to spam filtering, by using a realistic mixture of real spam and ham messages. Promising results were obtained by the SF, which outperformed a local CBF under several scenarios (e.g. using fixed and incremental symbiotic groups) and for a small number of users (from 3 to 5). The increase of performance was particularly high for new users. Also, SF was more robust to word contamination attacks. By dynamically combining filters from distinct users, we believe that a stronger protection is achieved against spam. Next, we discuss in detail several important SF issues.

### 2.1 SF and Distributed Data Mining

The main DDM goal is to obtain a global model by aggregating several local data mining analysis [10]. SF can be viewed as a DDM variant. SF is a natural distributed response to spam, as it reuses data mining models that are already locally available at the user level. Yet, SF is distinct from classic DDM since each user will have an aggregated model that is tailored to her/his needs. Thus, there is a different motivational issue, since if a given user does not benefit from the SF, she/he could easily leave the collaboration.

### 2.2 SF and Other Hybrid CBF-CF Systems

SF is different from other hybrid CBF-CF works that were devised for centralized systems (e.g. collaborative ensemble learning) [13]. In SF, the data and filters are distributed through different entities, thus there are issues not only of user motivation but also distribution (how and what should be shared), privacy and security (e.g. users are not willing to share legitimate email).

### 2.3 Privacy and Security

Sharing DM models is less sensitive than exchanging data, but there are still privacy issues. If user A has access to the filter of B, then

<sup>1</sup> Dep. Information Systems, University of Minho, Portugal, email: {clopes, pcortez}@dsi.uminho.pt

<sup>2</sup> Dep. Informatics, University of Minho, Portugal, email: pns@di.uminho.pt

A may feed a given token (or set of tokens) into the model and thus know with some probability that such token was classified by B as spam or ham. Our privacy solution resides in an anonymous exchange of models. Privacy can be increased if each individual does not know the symbiotic group composition. Yet, for social networks it may also be attractive that the symbiotic group composition could be assessed by all the SF participants. To harvest an individual e-mail may be easy for spammers, yet knowing who belongs to a given SF group is more difficult, as security can be increased by using encrypted SF communication. Also, the use of trust weights can be used to prevent fake users from joining SF groups [12].

## 2.4 Communication and Scalability

When compared to CF, SF requires less communication costs. For example, a filter built from a millions of emails can be described by a few hundreds or thousands of bytes (depending on the filter algorithm used) [5]. To exchange the filters, a standard format should be adopted, such as the Predictive Model Markup Language (PMML) [8], which is compatible with a large number of data mining tools. Also, the SF exchange (e.g. filters) can be set in an asynchronous fashion and triggered only after receiving  $m$  messages or when it is detected that the filter suffers a concept shift [3].

The blind exchange of features or filters can be set using two implementations: a centralized server or a Peer-to-Peer (P2P)-like application. Under the first option, all users register into a centralized and secure service. This service could be implemented by large companies or email providers (e.g. Gmail or Hotmail), when all emails are stored at a given server. The second distributed implementation is more natural for the SF concept (e.g. under social networks), although it requires the users to trust the P2P software.

SF is linearly affected by the number of “foreign” filters. To improve scalability, we propose using text clustering [2] (e.g. using the CBF word features) to segment users with similarities, thus reducing the SF group and avoiding the inclusion of inadequate or malicious filters. Another option is the use social networks, where users could choose their “friends”. Such system could be implemented in social network sites (e.g. Facebook).

## 2.5 Current and Future Work

To experimentally test the SF capabilities, we are currently approaching the Enron corpus [1], which includes real ham messages from 148 users. Using the perl language, we defined the Enron relationships based on the number of received/sent messages between any two Enron users. Figure 1 shows an example of the social network related to user allen.p. Using a methodology similar to [4], we will select real spam from public repositories (e.g. <http://untroubled.org/spam/>) and use a realistic and controlled mixture of ham and spam (e.g. using different and changing spam/ham ratios). Next, we intend to apply CBF to each Enron user and test several SF approaches (e.g. text clustering and use of the Enron social network), in order to measure performance and scalability issues. At a second stage, we intend to encode SF in e-mail clients (e.g. Thunderbird extension), in order to gather feedback from real users.

## 3 CONCLUSION

Rather than improving a single classifier or exchanging messages, the emphasis of SF is on accessing information learned from other filters to improve personal filtering. In this position paper, we discuss several SF issues when applied to spam e-mail detection.

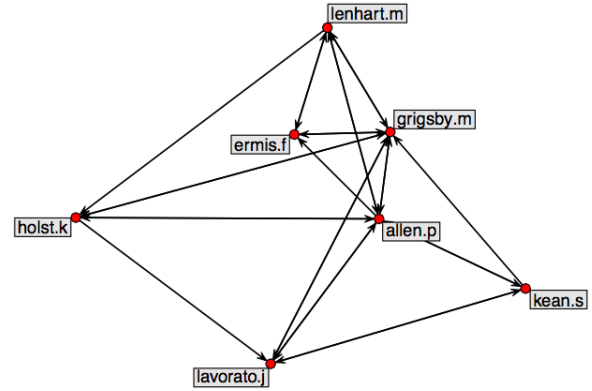


Figure 1. Example of the Enron social network

## ACKNOWLEDGEMENTS

The work is supported by FCT project PTDC/EIA/64541/2006 and FCT grant SFRH/BD/44840/2008.

## REFERENCES

- [1] R. Beckermann, A. McCallum, and G. Huang, ‘Automatic categorization of email into folders: benchmark experiments on Enron and SRI corpora’, Ir-418, University of Massachusetts Amherst, (2004).
- [2] R. Bilisoly, *Practical text mining with Perl*, Wiley Publishing, 2008.
- [3] G. Castillo and J. Gama, ‘Adaptive Bayesian network classifiers’, *Intelligent Data Analysis*, **13**(1), 39–59, (2009).
- [4] P. Cortez, C. Lopes, P. Sousa, M. Rocha, and M. Rio, ‘Symbiotic Data Mining for Personalized Spam Filtering’, in *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI-09)*, pp. 149–156. IEEE, (2009).
- [5] A. Garg, R. Battiti, and R. Cascella, ‘May I borrow your filter? Exchanging filters to combat spam in a community’, in *Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on*, volume 2, (2006).
- [6] S. Garriss, M. Kaminsky, M.J. Freedman, B. Karp, D. Mazières, and H. Yu, ‘RE: reliable email’, in *Proceedings of the 3rd conference on Networked Systems Design and Implementation (NSDI)*, pp. 297–310, San Jose, CA, (2006). USENIX Association Berkeley, CA, USA.
- [7] A. Gray and M Haahr, ‘Personalised, Collaborative Spam Filtering’, in *1st Conference on E-Mail and Anti-Spam CEAS*, (2004).
- [8] R. Grossman, M. Hornick, and G. Meyer, ‘Data Mining Standards Initiatives’, *Communications of ACM*, **45**(8), 59–61, (2002).
- [9] T. Guzella and W. Caminhas, ‘A review of machine learning approaches to Spam filtering’, *Expert Systems with Applications*, **36**, 10206–10222, (2009).
- [10] F. Provost, *Advances in Distributed and Parallel Knowledge Discovery*, chapter Distributed data mining: Scaling up and beyond, MIT Press, 2000.
- [11] M86 Security Team, ‘Security Labs Report’, Tech. report – jul-dec 2009, available at: <http://www.m86security.com/labs>, Orange, CA, USA, (January 2010).
- [12] H. Yu, M. Kaminsky, P. Gibbons, and A. Flaxman, ‘Sybilguard: Defending against sybil attacks via social networks’, *IEEE/ACM Transactions on Networking (TON)*, **16**(3), 576–589, (2008).
- [13] K. Yu, A. Schwaighofer, V. Tresp, W. Ma, and H. Zhang, ‘Collaborative Ensemble Learning: Combining Collaborative and Content-Based Information Filtering via Hierarchical Bayes’, in *19th International Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 353–360. ACM, (2003).

# Memoryless Fading Windows in Ubiquitous Settings

Pedro Pereira Rodrigues<sup>1</sup> and João Gama<sup>2</sup> and Raquel Sebastião<sup>3</sup>

**Abstract.** In streaming scenarios, recent data is usually more important than old data. Probably the most common approach of algorithms focusing on evolving recent data are sliding windows. In this work we propose the use of weighted sliding windows, where data points inside the window lose weight exponentially with time. However, in ubiquitous streaming data sources, such as sensor networks, resources like memory and processing power are scarce. Hence, most of the times there is not even enough memory to store all the data points inside the window, leading to approximated results. In this paper we propose the use of fading windows, a memoryless model which uses all previous data to approximate a weighted window. We present theoretical proof and empirical evidence of the approximating ability of such method for moving averages, also defining bounds to the error of such approximation. Finally, we derive a precise method to define the fading window parameters so that we can approximate a weighted average for a window of a given size, within certain error bounds. Future work will focus on the definition of tighter bounds for the error, study the application to correlation and histograms, and the possible connection with concept drift detection.

## 1 Introduction

Data streams are unbounded in length. It is impractical to store all data to execute queries that reference past data. High-quality approximate answers can be an acceptable solution. In such cases, queries are executed over a summary, a compact data-structure that captures the distribution of the data. Certainly, large summaries provide more precise answers. Either way, there is a trade-off between the size of summaries and the overhead to update summaries and the ability to provide precise answers [1].

Approximation methods in the data streams setting can be defined using the  $(\epsilon, \delta)$ -approximation schema [4]: given any positive number  $\epsilon < 1$  and  $\delta < 1$  compute an estimate that with probability  $1 - \delta$  is within relative error  $\leq \epsilon$ . Time and space required to compute an answer depends on  $\epsilon$  and  $\delta$ . Some results on tail inequalities provided by statistics are useful to accomplish this goal. The basic general bounds on the tail probability of a random variable (that is, the probability that a random variable deviates greatly from its expectation) include the Markov, Chebyshev and Chernoff inequalities [8]. However, usual stream data points are not produced at random according to a stable distribution, diminishing the possibility of probabilistic error bounds.

In most applications recent data is the most relevant one. To target this subset of data, a popular approach consists of defining a time window covering the most recent data [9]. Actually, time windows are a commonly used approach to solve queries in open-ended data streams. Instead of computing an answer over the whole data stream, the query (or operator) is computed, eventually several times, over a finite subset of tuples. In this model, a time stamp is associated with each tuple. The time stamp defines when a specific tuple is valid (e.g. inside the window) or not [4].

The paper is organized as follows. Next section presents an overview of some simple well-known window models. Section 3 presents the  $\alpha$ -weighted window, a sliding window model with exponential forgetting within the window. Then, Section 4 presents the main contribution of the paper, the  $\alpha$ -fading window as an approximation of the  $\alpha$ -weighted window, and the corresponding fading statistics. Section 5 concludes the paper with some remarks.

## 2 Window Models

Several window models have been used in the literature. The most relevant are: *landmark*, *sliding* and *time-biased* windows.

### 2.1 Strict Window Models

Landmark windows [6] identify relevant points (the landmarks) in the data stream and the aggregate operator uses all record seen so far after the landmark. Successive windows share some initial points and are of growing size. In some applications, the landmarks have a natural semantic. For example, in daily basis aggregates the beginning of the day is a landmark. Examples of the use of such windows appear in [10, 11]. Most of the time, we are only interested in computing statistics in the strictly recent past. The simplest approach are sliding windows of fixed size  $w$  [9]. However, due to the need to forget old observations, we need to maintain in memory all the observations inside the window. Nevertheless, sliding windows approaches have been successfully applied in summarizing recent data, especially for stream evaluation purposes [5].

### 2.2 Time-Biased Window Models

Previous windows models use a catastrophic forget, that is, any past observation either is inside the window or it is not inside the window. Usually in streaming settings, the concept generating data evolves smoothly, so old data is less but still important [3]. A smoother approach are tilted time windows, where time scale is compressed. The most recent data is stored inside the window at the finest detail (granularity). Oldest information is stored at a coarser detail, in an aggregated way, with the level of granularity depending on the application [7]. Another approach, presented next, could weight each obser-

<sup>1</sup> LIAAD - INESC Porto L.A. & SBIM-FMUP - Faculty of Medicine & DCC-FCUP - Faculty of Sciences, University of Porto, email: pprodrigues@med.up.pt

<sup>2</sup> LIAAD - INESC Porto L.A. & FEP - Faculty of Economics, University of Porto, email: jgama@fep.up.pt

<sup>3</sup> LIAAD - INESC Porto L.A. & DM-FCUP - Faculty of Sciences, University of Porto, email: raquel@liaad.up.pt

vation inside the window according to the age of the data point: older data would get less weight.

### 3 Weighted Sliding Windows

In data streams scenarios, recent data is usually more important than old data [4]. Even within a sliding window, the most recent data point is usually more important than the last one which is about to be discarded. This way, a simple approach could consider giving weights to data points depending on their age within the sliding window. Several weighting models could apply: linear, loglinear, etc. Given its particular characteristics, we will present an exponential approach, where the weight of a data point decreases exponentially with time.

**Definition** Let  $i$  be the number of observations of a given variable  $X$  from which we are monitoring a sliding window of size  $w$ . The  $\alpha$ -**weighted window** is the set of points

$$\dot{X}_{\alpha,w}(i) = \left\{ \alpha^{i-j} x_j \mid j \in [i-w, i], 0 < \alpha < 1 \right\}. \quad (1)$$

The main advantages of this window model are two-fold. First, compared to traditional sliding windows, more importance is given to recent data points, as the weight of each observation decreases exponentially with time. Second, compared to other weighting approaches, it can be maintained on the fly. At first, we are tempted to say that the set  $\dot{X}_{\alpha,w}(i)$  must be computed at every new observation  $i$ . However, given its exponential definition, we can compute it using the recursive form

$$\dot{X}_{\alpha,w}(i) = \{x_i\} \cup \alpha \times \dot{X}_{\alpha,w}(i-1) \setminus \{\dot{x}_{i-w}\}. \quad (2)$$

#### 3.1 Weighted Statistics

As previously stated, almost every systems have the need to compute simple statistics over recent data. Examples of those statistics include averages and correlation, which are based on sums and counts. In the following we will focus on averages. Using the  $\alpha$ -weighted window model, these statistics need to be defined slightly differently.

**Definition** Let  $i$  be the number of observations of a given variable  $X$  from which we are monitoring a sliding window of size  $w$ . A  $\alpha$ -**weighted increment**  $N_{\alpha,w}(i)$  is

$$N_{\alpha,w}(i) = \sum_{j=1+i-w}^i \alpha^{i-j} = \sum_{j=0}^{w-1} \alpha^j, \quad (3)$$

with  $0 < \alpha < 1$ .

**Definition** Let  $i$  be the number of observations of a given variable  $X$  from which we are monitoring a sliding window of size  $w$ . The  $\alpha$ -**weighted moving sum**  $S_{x,\alpha,w}(i)$  is a weighted sum of the observations of  $X$  within a recent window of size  $w$ , where

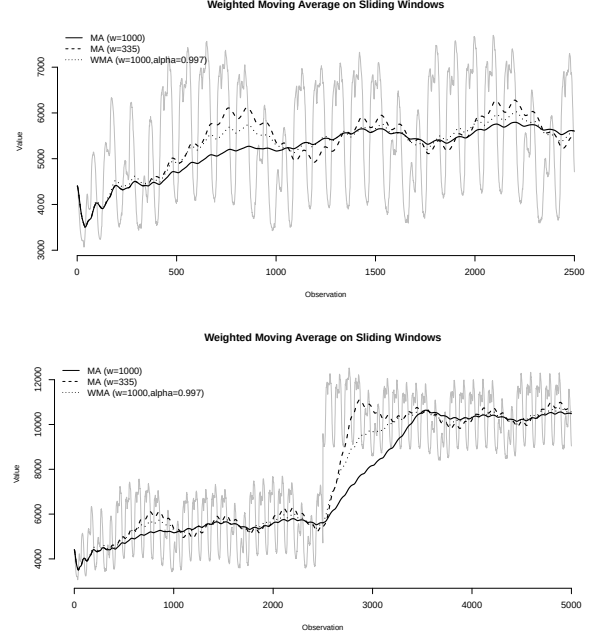
$$S_{x,\alpha,w}(i) = \sum_{j=1+i-w}^i \alpha^{i-j} x_j, \quad (4)$$

with  $0 < \alpha < 1$ .

**Definition** Let  $S_{x,\alpha,w}(i)$  be the  $\alpha$ -weighted moving sum of variable  $X$  after  $i$  observations. The  $\alpha$ -**weighted moving average**  $\bar{X}_{\alpha,w}(i)$  is a weighted moving average within the window of the  $w$  most recent observations of  $x$ , so that

$$\bar{X}_{\alpha,w}(i) = \frac{\sum_{j=1+i-w}^i \alpha^{i-j} x_j}{N_{\alpha,w}(i)}, \quad (5)$$

with  $0 < \alpha < 1$ .



**Figure 1.** Comparison between traditional and weighted moving averages. The data stream (light thin line) is sketched as a moving average with  $w = 1000$  (solid thick line), as a moving average with  $w = 335$  (long dashed thick line) and a weighted moving average with  $w = 1000$  and  $\alpha = .997$  (dashed thick line).

#### 3.2 Comparison with the Sliding Window Model

Figure 1 (top) presents the comparison among this window model and a traditional window model in the computation of averages. The reader can note that, in the example, while having a catastrophic forgetting of data points older than 1000, the  $\alpha$ -weighted moving average keeps a smooth forgetting of values newer than that. This way, it can approximate better the effect of a small sliding window (better adaptation) while keeping information from older data (smooth behavior). This effect is even more profound when dealing with data with concept drift. Figure 1 (bottom) also presents the same evaluation but for the harsh scenario where the data drifts in terms of the mean of values.

The main feature of the weighted sliding window model is the use of smooth forgetting proposed recently as a fading factor [5]. In fact, this approach was firstly proposed to deal with evaluation of learning algorithms, and the usage of all previous data. However, its application to window models presents some advantages to the processing of data streams, as its recursive form enables the computation of weighted data points on the fly.

### 4 Memoryless Fading Windows

To avoid keeping all data in the window when computing statistics which are based on sums of the data points, and in order to include a smooth forgetting of information, the previous approach can be applied to achieve an approximated value for the elementary statistics on a data stream. This is strongly related to a weighted sum of the points in the sliding window, with more weight being given to most recent data points. The fading factors are memoryless, an important property in streaming scenarios [5]. In this section we will define

fading quantities, which do not require to keep the data points within the sliding window, but showing at the same time that they are, in fact, approximations of the same statistics computed with data points from the sliding window.

## 4.1 Fading Statistics

Similar to the  $\alpha$ -weighted window definitions, different statistics can be defined, based on counts and sums, which make use of all the previous data points. When possible, we will use the recursive forms to illustrate its applicability to data streams.

**Definition** Let  $i$  be the number of observations of a given variable  $X$ . A  $\alpha$ -fading increment  $N_\alpha(i)$  is

$$N_\alpha(i) = \begin{cases} 1, & i = 1 \\ 1 + \alpha \times N_\alpha(i-1), & i > 1 \end{cases} \quad (6)$$

with  $0 < \alpha < 1$ .

A first look at the  $\alpha$ -fading increment clarifies its differences to usual increments.

**Theorem 1**  $N_\alpha(i)$  converges to  $\frac{1}{1-\alpha}$  when  $i \rightarrow +\infty$ .

**Proof** Using simple arithmetics one can see that  $N_\alpha(i) = \sum_{j=0}^{i-1} \alpha^j$  which is a special form of the infinite geometric series  $\sum_{k=0}^{\infty} ar^k$ , built from the geometric progression with initial value  $a = 1$  and common ratio  $r = \alpha$ . For values of  $|r| < 1$ , these series are known to converge to  $\frac{a}{1-r}$ , hence  $\lim_{i \rightarrow \infty} N_\alpha(i) = \frac{1}{1-\alpha}$ . ■

Given its convergence property,  $\alpha$ -fading increments will have a massive effect in the properties of the remaining count-based statistics.

**Definition** Let  $i$  be the number of observations of a given variable  $X$ . The  $\alpha$ -fading sum  $S_{x,\alpha}(i)$  of  $X$  is a weighted sum of the observations of  $X$ , where

$$S_{x,\alpha}(i) = \begin{cases} x_1, & i = 1 \\ x_i + \alpha \times S_{x,\alpha}(i-1), & i > 1 \end{cases} \quad (7)$$

with  $0 < \alpha < 1$ .

As for the weighted window model, the  $\alpha$ -fading increment is clearly related with the  $\alpha$ -fading sum.

**Lemma 2**  $N_\alpha(i)$  equals the total amount of weight given to observations used in the  $\alpha$ -fading sum  $S_{x,\alpha}(i)$ .

**Proof** The  $\alpha$ -fading sum  $S_{x,\alpha}(i)$  can be rewritten as  $S_{x,\alpha}(i) = \sum_{j=1}^i \alpha^{i-j} x_j$ . Hence, the sum of weights is  $\sum_{j=1}^i \alpha^{i-j} = \sum_{j=0}^{i-1} \alpha^j = N_\alpha(i)$ . ■

Given the previous definitions, we can approximate the weighted window model when dealing with sum-based statistics.

## 4.2 Approximating the Weighted Window Model

Since we are using all previous data in the computation of  $\alpha$ -fading increments and  $\alpha$ -fading sums, to approximate a window model of size  $w$ , an important aspect to check is how much of the data that is older than the objective  $w$  is being in fact used in this computation.

**Definition** Let  $w$  be the size of a window with the most recent examples  $x_j, \forall j \in [i-w, i]$ . The **ballast weight**  $B_{\alpha,w}(i)$  is the proportion of weight given to old observations (with respect to  $w$ ) in the computation of the  $\alpha$ -fading sum  $S_{x,\alpha}(i)$ , i.e.

$$B_{\alpha,w}(i) = \frac{\sum_{j=w}^{i-1} \alpha^j}{N_\alpha(i)}. \quad (8)$$

Apparently, as the window slides on, the amount of old data points being used in the computation increases. However, the special form of the fading factors yields an important property.

**Theorem 3**  $B_{\alpha,w}(i)$  converges to  $\alpha^w$  when  $i \rightarrow +\infty$ .

**Proof** Given that  $\lim_{i \rightarrow \infty} B_{\alpha,w}(i) = \lim_{i \rightarrow \infty} \frac{\sum_{j=w}^{i-1} \alpha^j}{\sum_{j=0}^{i-1} \alpha^j}$ , we can apply Theorem 1 to the expression, getting  $\lim_{i \rightarrow \infty} B_{\alpha,w}(i) = \frac{\frac{1}{1-\alpha} - \sum_{j=0}^{w-1} \alpha^j}{\frac{1}{1-\alpha}}$ . Then, for  $\alpha < 1$ , with simple arithmetics we get  $\lim_{i \rightarrow \infty} B_{\alpha,w}(i) = 1 - (1-\alpha) \frac{1-\alpha^w}{1-\alpha} = 1 - 1 + \alpha^w = \alpha^w$ . ■

Now, given a fading factor  $\alpha$  and an objective window size  $w$ , we can know exactly how much of the sum is based on data points older than  $w$ . This way, depending on the application domain, we can define an admissible value for the ballast weight, so that we are approximating a sliding window with minimum error: the  $(\alpha, \varepsilon)$ -fading window for  $S_{x,\alpha}(i)$ .

**Theorem 4** Let  $\varepsilon$  be an admissible value for the ballast weight of a given  $\alpha$ -fading sum  $S_{x,\alpha}(i)$ . If  $w > \log_\alpha \varepsilon$  the ballast weight converges to  $B_{\alpha,w}(i) < \varepsilon$  when  $i \rightarrow +\infty$ .

**Proof** From Theorem 3 comes that  $\lim_{i \rightarrow \infty} B_{\alpha,w}(i) = \alpha^w$ . Hence, for  $\lim_{i \rightarrow \infty} B_{\alpha,w}(i) < \varepsilon$ , then  $\alpha^w < \varepsilon \Leftrightarrow w > \log_\alpha \varepsilon$ . ■

**Definition** The  $(\alpha, \varepsilon)$ -fading window of a  $\alpha$ -fading sum  $S_{x,\alpha}(i)$  is the window of the  $w$  most recent observations of  $x$ , with  $w = \lceil \frac{\log \varepsilon}{\log \alpha} \rceil$ .

This window has the nice property of being the smallest one for which the ballast bounds apply.

**Corollary 5** Let  $B'_{\alpha,w}(i)$  be the proportion of weight given to examples in the window of the  $w$  most recent examples, i.e.  $B'_{\alpha,w}(i) = 1 - B_{\alpha,w}(i)$ . The  $(\alpha, \varepsilon)$ -fading window is the smallest recent window with total weight proportion  $B'_{\alpha,w}(i) \geq 1 - \varepsilon$  in the  $\alpha$ -fading sum  $S_{x,\alpha}(i)$ .

**Proof** Given the definition of  $B'_{\alpha,w}(i)$ ,  $\lim_{i \rightarrow \infty} B'_{\alpha,w}(i) < 1 - \lim_{i \rightarrow \infty} B_{\alpha,w}(i)$ . From Theorem 4 comes that  $\lim_{i \rightarrow \infty} B_{\alpha,w}(i) < \varepsilon$  if  $w > \frac{\log \varepsilon}{\log \alpha}$ , so  $w = \lceil \frac{\log \varepsilon}{\log \alpha} \rceil$  is the smallest integer value of  $w$  for which  $B_{\alpha,w}(i) < \varepsilon \Leftrightarrow B'_{\alpha,w}(i) \geq 1 - \varepsilon$ . ■

After proving some properties of the  $\alpha$ -fading sum, we can take the step of approximating the  $\alpha$ -weighted moving average.

**Definition** Let  $S_{x,\alpha}(i)$  be the  $\alpha$ -fading sum of variable  $X$  after  $i$  observations, and  $N_\alpha(i)$  the corresponding  $\alpha$ -fading increment. The  $\alpha$ -fading average  $M_{x,\alpha}(i)$  is a weighted average of observations of  $x$ , where

$$M_{x,\alpha}(i) = \frac{S_{x,\alpha}(i)}{N_\alpha(i)}, \quad (9)$$

with  $0 < \alpha < 1$ .

Similar approaches can be made for  $\alpha$ -fading correlation and  $\alpha$ -fading histograms, where the sufficient statistics needed to compute the final quantity are kept as  $\alpha$ -fading sums. Nonetheless, the definition of the  $\alpha$ -fading average is in itself a milestone in the processing of streaming data. However, the approximation to the  $\alpha$ -weighted average is not error free.

### 4.3 Approximation Error Bounds

When dealing with approximated results, a level of error should be associated with it. Hence, research has evolved into the definition of bounds for those errors [2]. Considering  $M_{x,\alpha}(i)$  an approximation of  $\bar{X}_{\alpha,w}(i)$ , the error can be clearly bounded.

**Definition**  $\Delta_{x,\alpha,w}(i)$  is the error of approximating the  $\alpha$ -weighted moving average  $\bar{X}_{\alpha,w}(i)$  with  $M_{x,\alpha}(i)$ , i.e.

$$\Delta_{x,\alpha,w}(i) = \|\bar{X}_{\alpha,w}(i) - M_{x,\alpha}(i)\|. \quad (10)$$

**Theorem 6** Let  $\varepsilon < 1$  be an admissible ballast weight for the  $\alpha$ -fading sum, and  $R$  the known range of variable  $X$ . Then,  $\Delta_{x,\alpha,w}(i) \leq 2\varepsilon R$ .

**Proof** From previous definitions comes that

$$\Delta_{x,\alpha,w}(i) = \left\| \frac{\sum_{j=1+i-w}^i \alpha^{i-j} x_j}{\sum_{j=1+i-w}^i \alpha^{i-j}} - \frac{\sum_{j=1}^i \alpha^{i-j} x_j}{\sum_{j=1}^i \alpha^{i-j}} \right\|$$

There are two sources of error when approximating  $\bar{X}_{\alpha,w}(i)$  with  $M_{x,\alpha}(i)$ , that become visible if we separate the computation of  $M_{x,\alpha}(i)$  in terms of the sums of points inside and outside the  $\varepsilon$ -fading window.

$$\Delta_{x,\alpha,w}(i) = \left\| \Delta_{x,\alpha,w}^{in}(i) - \Delta_{x,\alpha,w}^{out}(i) \right\|,$$

where

$$\Delta_{x,\alpha,w}^{in}(i) = \frac{\sum_{j=1+i-w}^i \alpha^{i-j} x_j}{\sum_{j=1+i-w}^i \alpha^{i-j}} - \frac{\sum_{j=1+i-w}^i \alpha^{i-j} x_j}{\sum_{j=1}^i \alpha^{i-j}},$$

and

$$\Delta_{x,\alpha,w}^{out}(i) = \frac{\sum_{j=1}^{i-w} \alpha^{i-j} x_j}{\sum_{j=1}^i \alpha^{i-j}}.$$

The first one occurs because, in  $M_{x,\alpha}(i)$ , we are giving less weight to the “good” data points (the ones within the recent window  $w$ ): while  $\bar{X}_{\alpha,w}(i)$  divides by  $\sum_{j=1+i-w}^i \alpha^{i-j}$ ,  $M_{x,\alpha}(i)$  divides by  $\sum_{j=1}^i \alpha^{i-j}$ . The second source of error comes from including “bad” data points (the ones outside the recent window  $w$ ). Of course the definition of “good” and “bad” is just illustrative; they might actually have the paradoxical effect in the average. Since we are looking for an upper bound on the error, the worst case scenario is that these two sources of error do not cancel out, rather adding up their effect:

$$\Delta_{x,\alpha,w}(i) \leq \left\| \Delta_{x,\alpha,w}^{in}(i) \right\| + \left\| \Delta_{x,\alpha,w}^{out}(i) \right\|.$$

Hence

$$\Delta_{x,\alpha,w}(i) \leq \left\| \frac{(\sum_{j=1}^{i-w} \alpha^{i-j})(\sum_{j=1+i-w}^i \alpha^{i-j} x_j)}{(\sum_{j=1+i-w}^i \alpha^{i-j})(\sum_{j=1}^i \alpha^{i-j})} \right\| + \left\| \Delta_{x,\alpha,w}^{out}(i) \right\|.$$

Then, from previous definitions comes that

$$\begin{aligned} \Delta_{x,\alpha,w}(i) &\leq \left\| \frac{B_{\alpha,w}(i) N_{\alpha}(i) \sum_{j=1+i-w}^i \alpha^{i-j} x_j}{B'_{\alpha,w}(i) N_{\alpha}(i) N_{\alpha}(i)} \right\| + \left\| \Delta_{x,\alpha,w}^{out}(i) \right\| \\ &= \left\| \frac{B_{\alpha,w}(i) \sum_{j=1+i-w}^i \alpha^{i-j} x_j}{B'_{\alpha,w}(i) N_{\alpha}(i)} \right\| + \left\| \frac{\sum_{j=1}^{i-w} \alpha^{i-j} x_j}{N_{\alpha}(i)} \right\|. \end{aligned}$$

Since we are looking for an upper bound on the error, let us analyse the worst case scenario. For the first term of the error, the worst case scenario is when we are decreasing weight to data points whose impact in the moving average is higher, those that are farther apart from the average ( $\max \delta_j$ , with  $\delta_j = |x_j - \bar{X}_{\alpha,w}(i)|$ ). For the second term of the error, the worst case scenario is when we are giving weight to data points that will have a higher impact in the average, hence, also the farthest apart from it. If the variable being measured is bounded by a certain range  $R$  (e.g. each observation is a probability) then we can define an upper bound to each  $\delta_j$ , i.e.  $\delta_j \leq R, \forall j$ . If  $R$  is not known, we can use  $R \sim \hat{R}(i) = \max X_{1..i} - \min X_{1..i}$ , the observed range of previous observations of  $X$ . Since keeping the range of the variable in a sliding window is a blocking operator, we can only keep track of the global range  $\hat{R}(i)$  of the variable. Nonetheless, this range is always at least as large as the range of the values in the recent window, so the bound stays robust. Hence, the upper bound on the error is given by considering all  $x_j = R$ :

$$\begin{aligned} \Delta_{x,\alpha,w}(i) &\leq \left\| \frac{B_{\alpha,w}(i) \sum_{j=1+i-w}^i \alpha^{i-j} x_j}{B'_{\alpha,w}(i) N_{\alpha}(i)} \right\| + \left\| \frac{\sum_{j=1}^{i-w} \alpha^{i-j} x_j}{N_{\alpha}(i)} \right\| \\ &= \left\| \frac{B_{\alpha,w}(i) B'_{\alpha,w}(i) N_{\alpha}(i) R}{B'_{\alpha,w}(i) N_{\alpha}(i)} \right\| + \left\| \frac{B_{\alpha,w}(i) N_{\alpha}(i) R}{N_{\alpha}(i)} \right\| \\ &= \|B_{\alpha,w}(i) R\| + \|B_{\alpha,w}(i) R\| \\ &= 2 \|B_{\alpha,w}(i) R\|. \end{aligned}$$

Fixating  $\varepsilon$  for an admissible ballast weight,  $\Delta_{x,\alpha,w}(i) \leq 2\varepsilon R$ . ■

Now, given a  $\alpha$ -fading average  $M_{x,\alpha}(i)$  of a variable  $X$  with range  $R$ , we can state that this average is an approximation of the  $\alpha$ -weighted moving average  $\bar{X}_{\alpha,w}(i)$ , with  $w = \lceil \frac{\log \varepsilon}{\log \alpha} \rceil$ , within an error interval of  $M_{x,\alpha}(i) \pm 2\varepsilon R$ , where  $\varepsilon$  is the admissible ballast weight. Given this, we can clearly define the parameters needed to achieve the objective approximation.

**Corollary 7** Let  $X$  be a variable with range  $R$ . In order to compute an estimate  $M_{x,\alpha}(i)$  of the  $\alpha$ -weighted moving average  $\bar{X}_{\alpha,w}(i)$  on a sliding window with size  $w$ , so that  $M_{x,\alpha}(i) \in \bar{X}_{\alpha,w}(i) \pm 2\varepsilon R$ ,  $\alpha$  must be set to  $\alpha = \varepsilon^{\frac{1}{w}}$ .

**Proof** From Theorem 4 and Theorem 6 comes that, since for having  $\lim_{i \rightarrow \infty} B_{\alpha,w}(i) < \varepsilon$ ,  $w > \log_{\alpha} \varepsilon$ , then to achieve  $M_{x,\alpha}(i) \in \bar{X}_{\alpha,w}(i) \pm 2\varepsilon R$ ,  $\alpha = \varepsilon^{\frac{1}{w}}$ . ■

The main objective of approximating  $\bar{X}_{\alpha,w}(i)$  with  $M_{x,\alpha}(i)$  is to have a memoryless approach to compute an approximated quantity that would require  $w$  values to be kept. Moreover, this quantity has the properties of being a weighted average that gives more weight to most recent examples, while older examples loose weight exponentially. The connection between the  $\alpha$ -weighted moving average and a traditional moving average is highly distribution-dependent, so we keep the bias introduced in the current average model as a good property for the data stream processing. From Corollary 7 we get a precise way of defining  $\alpha$  so that our  $\alpha$ -fading average is an approximation of the  $\alpha$ -weighted moving average of size  $w$ , within a known bound of  $\bar{X}_{\alpha,w}(i) \pm 2\varepsilon R$ .

#### 4.4 Empirical Evidence of Error Bounds

Figure 2 presents a comparison of the  $\alpha$ -fading average with the  $\alpha$ -weighted moving average. The scenario was to approximate a sliding window of size  $w = 1000$ , with an admissible ballast weight  $\varepsilon = 0.05$ . To achieve the approximation with the error bounded by  $2\varepsilon R$ , the fading factor was set to  $\alpha = .005 \frac{1}{1000} = .997$ .

Apparently, in the first experiment, the error bounds seem too loose compared to the actual error of the approximation. However, this effect is due to the stability of the original data streams. The benefits of using this strategy are clear. Not only we maintain the benefits of the weighted sliding window model compared to traditional sliding windows, but we achieve it without the need to keep any data point from the sliding window. Only the corresponding  $\alpha$ -fading sums,  $\alpha$ -fading increment, and the observed range of the variable, are monitored.

Even for drifting contexts, where old data must be forgotten, the advantages of the  $\alpha$ -fading average linger. See Figure 2 for a comparison of the  $\alpha$ -fading average with the  $\alpha$ -weighted moving average on the same example. To check the ability of the model to react and protect (by bounding the error) to drift, Figure 2 presents the same comparison for a unitary step artificial data stream, where data ranges theoretically between 0 and 1 (e.g. probability), being constant at zero on the first concept, and one in the second concept. In these drifting cases, it becomes clear that the error bounds are actually only tight and loose enough to protect the concept drift region. Even in the most problematic part of the stream, the error of the approximation is bounded. The  $\alpha$ -fading window model clearly delivers.

#### 5 Concluding Remarks

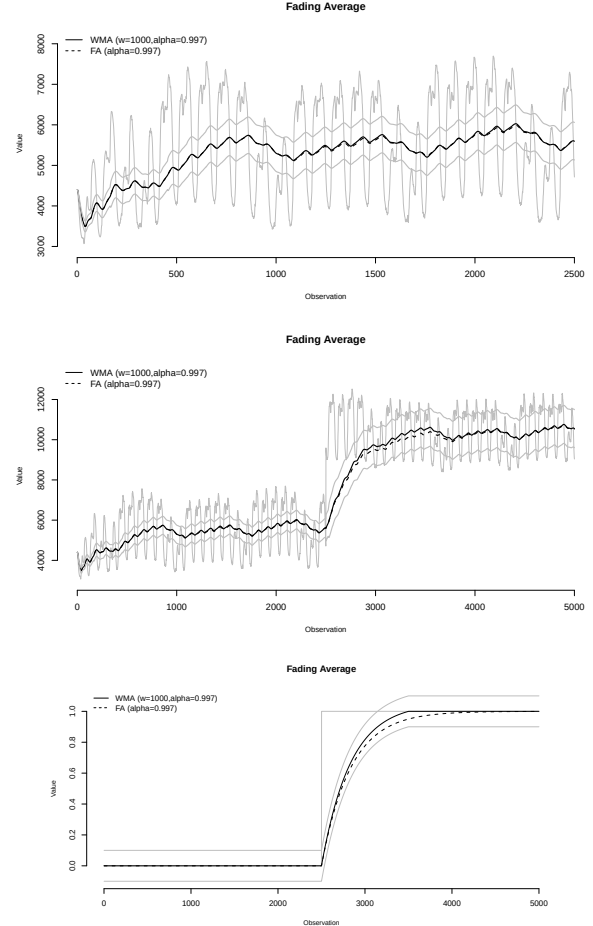
Usual ubiquitous streaming algorithms operate with limited memory and processing power. In this paper we have presented a memoryless approximation of a weighted window model for data stream summarization. The average computed on the  $\alpha$ -fading window is shown to approximate the average computed on the  $\alpha$ -weighted window, within precise error bounds. This way, we have a clear method to compute approximated averages without keeping all the data in a sliding window. A major advantage of this approach is that it is independent of the distribution generating the examples, as no probabilistic assumption is made on the data.

#### ACKNOWLEDGEMENTS

The work of P.P. Rodrigues and R. Sebastião is supported by the Portuguese Foundation for Science and Technology (FCT) under the PhD Grants SFRH/BD/29219/2006 and SFRH/BD/41569/2007, respectively. The authors thank FCT's Plurianual financial support attributed to LIAAD. This work was developed in the scope of project KDUDS (PTDC/EIA-EIA/98355/2008).

#### References

- [1] M. Datar, A. Gionis, P. Indyk, and R. Motwani, 'Maintaining stream statistics over sliding windows', in *Proceedings of 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 635–644. Society for Industrial and Applied Mathematics, (2002).
- [2] Pedro Domingos and Geoff Hulten, 'A general method for scaling up machine learning algorithms and its application to clustering', in *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pp. 106–113, (2001).
- [3] João Gama, Pedro Medas, Gladys Castillo, and Pedro Pereira Rodrigues, 'Learning with drift detection', in *Proceedings of the 17th Brazilian Symposium on Artificial Intelligence (SBIA 2004)*, eds., Ana L. C. Bazzan and Sofiane Labidi, volume 3171 of *Lecture Notes in Artificial Intelligence*, pp. 286–295, São Luiz, Maranhão, Brazil, (October 2004). Springer Verlag.



**Figure 2.** Comparison between  $\alpha$ -weighted moving average and the  $\alpha$ -fading average for a real data stream (top and middle) and an artificial unitary step data stream (bottom). The data stream (light thin line) is sketched as weighted moving average with  $w = 1000$  and  $\alpha = .997$  (solid thick line) and a  $\alpha$ -fading average with  $\alpha = .997$  (dashed thick line). Also sketched are the error bounds (solid grey lines).

- [4] João Gama and Pedro Pereira Rodrigues, 'Data stream processing', in *Learning from Data Streams - Processing Techniques in Sensor Networks*, eds., João Gama and Mohamed Medhat Gaber, chapter 3, 25–39, Springer Verlag, (2007).
- [5] João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues, 'Issues in evaluation of stream learning windows', in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2009)*, pp. 329–337, Paris, France, (2009). ACM Press.
- [6] Johannes Gehrke, Flip Korn, and Divesh Srivastava, 'On computing correlated aggregates over continual data streams', in *SIGMOD Conference*, pp. 13–24, (2001).
- [7] Jiawei Han and Micheline Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 2001.
- [8] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.
- [9] S. Muthukrishnan, *Data Streams: Algorithms and Applications*, Now Publishers Inc, New York, NY, 2005.
- [10] 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).
- [11] Pedro Pereira Rodrigues, João Gama, and João Pedro Pedrosa, 'Hierarchical clustering of time-series data streams', *IEEE Transactions on Knowledge and Data Engineering*, **20**(5), 615–627, (May 2008).



# GridClass: Strategies for Global Vs Centralized Model Construction in Grid Data Mining

Manuel Filipe Santos<sup>1</sup> and Wesley Mathew<sup>2</sup> and Hendrique Santos<sup>3</sup>

**Abstract.** This paper highlights 4 different distributed data mining strategies for making a global data mining model from local models in distributed sites. These models are Specific Classifier Method, Generalized Classifier Method, Majority Voting Method and Weighted Classifier Method. These strategies differ from the conventional centralized data mining technique, which makes a global model by mining a large data set collected from different distributed sites. Eleven Multiplexer problem is used in the experimental work. The supervised learning classifier system is used for data mining at distributed sites. The paper compares the efficiencies of the 4 different strategies and describes how the global model size affects the testing accuracy.

**Keywords.** Generalized Classifier, Specific Classifier, Weighted Classifier, Majority Voting.

## 1 INTRODUCTION

Recently there has been increased potential for data mining in the parallel and distributed environments. Today's global organizations that generate data in multiple locations have expanded the scope of data mining applications to global scale [1].

Ubiquitous data mining is the process of examining data generated from distributed sources on mobile devices or embedded devices [2]. Increasing use of the computational capability of mobile devices and rapid growth of wireless networks are contributing to the importance of ubiquitous computing [2].

Two approaches can be used to create a global model from distributed sites; 1) the centralized method in which global model is created from a large data set that combines data from distributed site[3], 2) the distributed method in which global model is created by first creating a local model at each distributed site and combining them [3,4]. This paper describes four strategies for the distributed method: Specific Classifier Method (SCM), Generalized Classifier Method (GCM), Majority Voting Method (MVM) and Weighted Classifier Method (WCM).

---

Department of Information System, University of Minho, Portugal, email: [mfs@dsi.uminho.pt](mailto:mfs@dsi.uminho.pt)<sup>1</sup>, [wesley@dsi.uminho.pt](mailto:wesley@dsi.uminho.pt)<sup>2</sup>, [hsantos@dsi.uminho.pt](mailto:hsantos@dsi.uminho.pt)<sup>3</sup>

The distributed data mining is more complex than the centralized data mining because centralized data mining involves only one data mining operation whereas distributed data mining has multiple data mining operations and it is difficult to combine local models in the global site without losing some classifiers from the local models. Eleven Multiplexer problem is used in the experimental work. The supervised learning classifier system is used for data mining at distributed sites. The paper compares the efficiencies of the 4 different strategies and describes how the global model size affects the testing accuracy.

Section 2 describes the background details of the experiments. It examines the design and working of the proposed system and presents two components: gridclass learning classifier system, and global model generator. It also demonstrates the different strategies for making global model. Section 3 explains the experimental work and results. This section describes the experimental setup. Section 4 discusses the result and performance of the proposed system. The final section presents the future work and conclusion.

## 2 BACKGROUND

The sUpervised learning Classifier System for gridclass (UCSgrid) is the tool used for data mining in the Grid environment [3]. UCSgrid is one of the most appropriate systems in the learning classifier system [3]. The basic components of proposed system are gridclass learning classifier system and global model generator.

### 2.1 Gridclass Learning Classifier System

Gridclass Learning Classifier System is a grid structured for data mining in a parallel fashion. Grid based Supervised classifier system (UCSgrid) is installed in every node in the distributed environment. The distributed system has basically 3 functions: 1) Reading predefined classifier, 2) Training, 3) Testing the local model. Here, training data file and testing data file are in CSV format, and the predefined classifier file is in XML format. Each distributed learning classifier system writes its local model in a XML file. UCSgrid develops the learning model based on the training data. The first step of this system is to read the predefined classifiers (population). If the predefined classifier is not given, UCSgrid starts the training process with empty population.

The training process is the process of maintaining a population based on the training data. Each training data has one condition part and one action part. The covering algorithm and genetic algorithm are two processes for maintaining classifiers in the population. When the training is employed, the condition part of the training data is matched with the condition part of the classifier in the population. Those classifiers that are correctly matched with the condition part of the train data become the match set (M) [3,4,5,6]. If classifiers in the match set's action part is matched with action part of the trained data, these classifiers are considered as correct set (C) [3, 4, 5, 6, 7]. If the size of correct set is empty, then the covering will be executed [5, 6]. If, the average experience of classifier in the correct set is greater than the user defined constant GA\_Threshold, then genetic algorithm will be triggered [3, 5, 6]. While covering, training data will add to population as a new classifier. The condition part of the training data is used as the condition part of the new classifier and action part of the training data is used as the action part of the new classifier. While GA, two classifiers are selected with probability proposition to fitness from correct set and doing crossover and mutation to generate two new classifiers. [6]

Testing is another task in supervised classifier system. Testing is used to check the accuracy of population. Each node in the distributed site has a separate testing data set.

## 2.2 Global Model vs. Local Models

A challenge in creating global model is combining local model without losing the benefits of each classifier [3]. Local models only represent their own training data (problem defined in the node), but the global model should represent the entire local model from the grid environment [3]. Two major concerns are addressed while generating global model from the local models in the grid environments: 1) managing the size of global model, 2) keeping the benefits of each classifier while modifying the parameters. The size of the local models is fixed. Local model size is defined based on the training data. But, in the case of global model, system cannot add all the classifiers from the grid environment. So, global model size is defined based on the different strategies used for making global model. Updating the parameters of classifiers in the local model is done only during the training process, but the parameters of the classifiers in the global model are updated during the merging of different local models.

Basically, there are two situations when we need to update the parameters: First, if two classifiers are the same, then there is no need to add repeated classifier in the global model. In this case, keep one classifier in the global model and update the parameters of that classifier with the parameters of the other. Second, if one classifier is more general than the other one, then the more general classifier is kept in the global model and the less general classifier is removed from the global model. The parameters of the more general classifier are updated with parameters of the less general classifier.

Next we present four different strategies for combining local models: WCM, MVM, SCM, and GCM. Figure 1 shows different methods for making global model. These four methods are newly described in this paper.

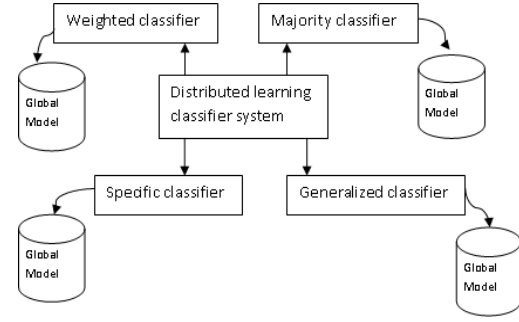


Figure 1. Different methods for combining local models

## 2.3 Different methods for generating global model

### 2.3.1 Weighted classifier method (WCM)

Weighted classifier method is used to find the highest weighted classifiers from the local models. Global population size is set by average size of local models. Weight of the classifier (Wgt) is derived from the parameters of Number of Match (Nm), Number of Correct (Nc), and Accuracy (Acc)

$$Wgt = Nm * Nc * Acc \quad (1)$$

When a new classifier comes into the global model it is matched with all the other classifiers that are already stored in the global model. If the weight of the new classifiers is higher than the weights of any classifier in the global model, then the new classifier is added to the population. If the population size already reached the maximum, then one classifier, whose weight is less than those of all the other classifiers in the population, is remove from the population. The global population size should not change dynamically.

### 2.3.2 Specific Classifier Method (SCM)

In this method, the global model preserves all discrete classifiers in the global model. While the global model collected from all local models in the distributed site may have many similar classifiers. Discrete classifier means that, no two classifiers have same condition and action part. Two classifiers are considered to be similar if the condition and action part of one is the same as the condition and action part of the other. For example, consider three classification: c1="0,#,0", c2="0,1,0", and c3="0,0,0". The c1, c2, and c3 are the condition part of the three classifiers. Here '#' is a wildcard [8]. Classifier c1 is more general than c2 and c3; but in SCM, does not consider the generality of classifiers. In SCM all three classifiers are added to the global model because they are consider as different from each other.

If a classifier is repeated, its parameters are updated with those of the repeating classifier. Size of the global model generated by SCM is dynamic, and it is greater than the size of global model generated by other methods.

### 2.3.3 Majority Voting Method (MVM)

Majority Voting is a third method for generating global model from distributed local models. This approach finds one cut-off-threshold [3] value to benchmark the classifiers in the global population, and there is no fixed size for global population. Here, the system first selects the discrete classifiers from the local models. If a new classifier is already in the global model, then parameters of the existing classifier are modified based on the parameters of the new classifier.

After reading the entire local models from the distributed sites, the system calculates the average value of accuracy, which is considered as the cut-of-threshold value of Majority Voting. If the classifier accuracy is less than this threshold value, then that classifier is removed from the global model. This method helps to maintain the valid classifiers in the global model.

### 2.3.4 Generalized Classifier Method (GCM)

The fourth method is the Generalized Classifier Method. This strategy is used to maintain only more general and discrete classifiers in the global model. A new classifier may come to global model under any one of the following distinct situations: 1) global model may have the same classifier, 2) global model may have more general classifier than the new classifier, 3) new classifier may be more general than the classifier in the global model, and 4) new classifier may be completely new.

Now we describe how this method handles each one of the 4 cases. In the first case, classifier just updates its value and does not allow a new classifier to enter the global model. In the second case, classifiers in the global model, which are more general, update their parameter values with new classifier parameters and do not let a new classifier to enter the global model. In the third case, classifiers that are less general are removed from the global model, and the parameters of the new classifier are updated with the parameters of the classifiers that are removed from the global model. The new classifier is added to the global model. In the fourth case, new classifiers directly enter into the global model. The size of the global model in this method is dynamic and it is smaller than the size of the global model generated by other methods.

## 3 EXPERIMENTAL WORK AND RESULTS

Experiment was done with 11 Multiplexer problem. Two sets of experiments were done in multiplexer problem. Each experiment differs by the way it generates training data and testing data. For training, all experiments were done with 5000 iterations. Each experiment in this work included four nodes in the grid environment. So each experiment has four different local models in grid environment. All four strategies are applied in each problem.

### 3.1 Experimental setup for 11 multiplexer problem

The Boolean multiplexer in 11 multiplexer problem has 11 bits therefore it has 2048 different ordered list. The data format of the

11multiplexer problem is a0, a1, a2, d0, d1, d2, d3, d4, d5, d6, d7. The first 3 bits (a0 - a2) are considered as the address bits and subsequent 8 bits (d0 - d7) are considering as answer bits [9]. The Two experiments of the 11 multiplexer are differentiated by the way they choose training and testing data. The problem was divided into two parts, 70% of data was taken for training, and 30% of data for testing. The training data was divided into 4 equal portions for four nodes. Similarly, the testing data also was divided into 4 equal portions for four nodes. So the size of training data set is 340 and the size of testing data set is 172.

In the first experiment, training data in each node was completely different from the data in other nodes, and the same was true for the testing data. In the second experiment the training and testing data in the four nodes shared some common classifiers because the data for each node was selected randomly. The centralized training data set was created by combining four training data sets in the grid environment; so, centralized training data set size was 1360 (340×4). Similarly the centralized testing data was created by combining four testing data sets in the grid environment resulting in a centralized testing data set size of 688(172×4).

## 3.2 Results

Table 1 displays the different testing accuracies of global models, which are generated by using the four different strategies in both experiments. The global model size of WCM was set to 400 and the size of the other 3 methods was not fixed.

Training and testing data sets in both experiments have the same size. In the first experiment, SCM has the best accuracy (0.9534884) of all other methods used in the grid environment. Other methods such as WCM and MVM also have good accuracy. In the second experiment, SCM and WCM have the best accuracy. In both experiments, testing accuracy of GCM is very low. Based on the accuracy, SCM is the best of all strategies.

**Table 1.** Testing accuracies of global model from 11 multiplexer problems.

| First Experiment       |           |                   |
|------------------------|-----------|-------------------|
| Strategies of global   | Accuracy  | Global model size |
| Generalized Classifier | 0.7848837 | 238               |
| Specific classifier    | 0.9534884 | 772               |
| Majority voting        | 0.8953480 | 503               |
| Weighted classifier    | 0.9127904 | 400               |
| Second Experiment      |           |                   |
| Generalized classifier | 0.7965114 | 220               |
| Specific classifier    | 0.9302326 | 775               |
| Majority voting        | 0.9186047 | 507               |
| Weighted classifier    | 0.9476744 | 400               |

Table 2 displays the testing accuracy of centralized learning models of both experiments. It shows testing accuracies for different sizes of learning models ranging from 100 to 800.

The size of the centralized learning model is varied so that the results can be compared with those of the distributed method. Five thousand training iterations were done in each execution of the centralized method.

This table shows that the accuracy of learning model is dependent on the size of learning model, because when the global model size increases, the accuracy also increases. Here, it is difficult to compare the results of distributed and centralized methods, because of the differences in the sizes of learning models. Comparison of SCM with the centralized model shows that in both the experiments SCM have almost similar accuracy according to the relating global model size. Comparison of WCM with the centralized model shows that in both experiments WCM has better accuracy than centralized method according to the relating global model size.

**Table 2.** Testing accuracies of the centralized model.

| Centralized model of 11 multiplexer problem |                  |                     |
|---------------------------------------------|------------------|---------------------|
| Experiment                                  | Testing Accuracy | Learning model size |
| 1                                           | 0.539244         | 100                 |
| 2                                           | 0.649709         | 100                 |
| 1                                           | 0.726744         | 200                 |
| 2                                           | 0.813953         | 200                 |
| 1                                           | 0.922965         | 300                 |
| 2                                           | 0.854651         | 300                 |
| 1                                           | 0.901162         | 400                 |
| 2                                           | 0.877907         | 400                 |
| 1                                           | 0.902616         | 500                 |
| 2                                           | 0.959302         | 500                 |
| 1                                           | 0.944767         | 600                 |
| 2                                           | 0.962209         | 600                 |
| 1                                           | 0.962209         | 700                 |
| 2                                           | 0.949128         | 700                 |
| 1                                           | 0.997093         | 800                 |
| 2                                           | 0.956395         | 800                 |

## 4 DISCUSSION

In order to provide more insights into the experimental results, we consider two criteria: 1) efficiencies of different strategies for generating global model, and 2) efficiencies of distributed and centralized data mining.

When global model is generated by GCM and SCM, there is no loss of classifiers. No loss means that each and every classifier in the local model has one representation or copy in the global model. When global model is generated by WCM or MVM, some classifiers are removed from the local models. The WCM keeps only the top classifiers based on the weight. The number of classifiers kept is equal to the size of the global model. In MVM, the classifiers which are above the threshold value are kept in the global model. The size of SCM is bigger than those of all other methods and the accuracy of the global model also is comparatively better.

The result of centralized classifier method used in the 11 multiplexer problems also shows the importance of size in the global model. This result reveals that the size of the global model partially affect the testing accuracy of the global model.

In the first experiment of 11 multiplexer problems, centralized accuracy and distributed accuracy are almost same, but in the second experiment, the centralized testing accuracy is better than the distributed testing accuracy.

A major difficulty of centralized data mining is the communication costs to transfer large amounts of data from the grid environment to central repository. In addition, since the size of data in the central repository is very large, the computational cost of centralized data mining is higher [3]. On the contrary, the distributed data mining only needs to send the local models from the gridclass to the global site. The communication cost and computational cost of distributed data mining are lower [3] because data mining is applied only to each small data set in the gridclass. The main advantage of centralized method is that the global model of the centralized method represents the whole data set. So the global model generated by the centralized method is more general than the global model generated from the distributed method.

## 5 CONCLUSION AND FUTURE WORK

The results of the experiments that are described in this paper reveal the possibility for implementing distributed data mining in grid environment. In general, the centralized method has higher accuracies than distributed data mining although SCM of the distributed data mining achieves accuracies that are closed to those of centralized method. The distributed data mining has the advantage of lower communication and computational costs compared to centralized data mining. WCM come second, and overall, GCM is the last.

This paper also shows the influence of the grid environment on the global mode. The experimental results show that the Global model size affects the testing accuracy. Therefore the efficiency of SCM is higher than those of other methods.

This work only focuses on the homogeneous data, but the future work should include heterogeneous data and more dynamic learning systems. Likewise, the future system should develop more adequate strategy for generating more compact and feasible global models. Future work will be tested with huge number of diverse set of real world data. Also that would include evaluation of communication and computation cost of each strategy with the results.

## ACKNOWLEDGEMENTS

The authors would like to express their gratitude's to FCT (Foundation of Science and Technology, Portugal), for the financial support through the contract GRID/GRI/81736/2006.

## REFERENCES

- [1] J. Luo, M. Wang, J. Hu, Z. Shi, Distributed data mining on Agent Grid: Issues, Platform and development toolkit. *Future Generation computer system* 23 2007 61-68.
- [2] O. Horovitz, S. Krishnaswamy, M. M. Gaber, A fuzzy approach for interpretation of ubiquitous data stream clustering and its application in road safety. *Intelligent Data Analysis* 11 2007 89- 108.
- [3] M. F. Santos, W. Mathew, T. Kovacs, H. Santos, A grid data mining architecture for learning classifier system. *WSEAS TRANSACTIONS on COMPUTERS* Volume 8, 2009 ISSN: 1109-2750.
- [4] H. H. Dam, A scalable Evolutionary Learning Classifier System for Knowledge Discovery in Stream Data Mining, M.Sci. University of

- Western Australia, Australia, B.Sci. (Hons) Curtin University of Technology, Australia. Thesis work 2008.
- [5] A. Orriols-Puig, A Further Look at UCS Classifier System. *GECCO'06*, July 8–12, Seattle, Washington, USA, 2006.
  - [6] A. Orriols, EsterBernado-Mansilla, Class Imbalance Problem in UCS Classifier System: Fitness Adaptation..., *Evolutionary Computation*, The 2005 IEEE congress, 604-611 Vol.1 2005.
  - [7] M. F. Santos, Learning Classifier System in Distributed environments, University of Minho, School of Engineering, Department of Information System. *PhD Thesis* 1999.
  - [8] T. Kovacs, Strength or Accuracy: Credit Assignment in Learning Classifier Systems. *Distinguished Dissertation*., Springer-Verlag London limited 2004
  - [9] <http://www.gene-expression-programming.com/webpapers/FerreiraCS2001/Section6/SS5/SSS2.htm>, May 2009.



# Real-time Alarm Model Adaptation Based on User Feedback

Violeta Mirchevska<sup>1</sup>, Boštjan Kaluža<sup>2</sup>, Mitja Luštrek<sup>2</sup>, and Matjaž Gams<sup>2</sup>

**Abstract.** This paper presents real-time adaptation of alarm detection models in a remote health monitoring system based on user feedback. Real-time adaptation enables systems to fine-tune to the needs and preferences of the user and changes in the environment. This way, the system performance is improved in terms of technical accuracy and subjective user wishes. Two types of alarm detection models were used: (1) models in the form of rules created by domain expert and (2) models induced by machine learning. The problem of adaptation for the rule-based models is defined as Markov decision process. Machine-learning models are adapted by rebuilding the model every time new data is obtained. We tested the adaptation capabilities of the two types of alarm detection models based on their accuracy and time-to-alarm (needed length of possibly critical activity, such as lying on the ground, which causes the models to raise an alarm). Both types of models achieved 90% alarm detection accuracy. The rule-based models decreased time-to-alarm when user-triggered alarms were raised and increased it when the user indicated false alarm. We did not observe this process for the machine-learning models.

## 1 INTRODUCTION

Remote health monitoring is gaining attention in developed countries. One of its main goals is to allow users, particularly the elderly, to continue living at home instead of being admitted to a nursing home or hospital. On the one hand, this increases independence and quality of life, and on the other hand, it significantly reduces elderly-support costs. Many research projects address the problem of design and development of systems that can intelligently, continuously and pervasively monitor the health conditions of its user.

In order for remote health monitoring to achieve its goals, it must be highly reliable, which means that it must detect all major health problems without raising too many false alarms. Given the diversity of users, it is difficult to develop a system that will suit each user in each possible circumstance from the start. Therefore, real-time system adaptation based on user feedback is an important topic of research.

We have studied the problem of real-time system adaptation based on user feedback in Confidence [1], ubiquitous care system to support independent living of the elderly. This system is able to detect emergency situations such as falls, which result in the user lying/sitting at an inappropriate place (e.g. on the ground). The user of the system may provide feedback concerning the detection of such situations through portable device. Emergency-situation detection is performed by rule-based alarm detection agents incorporating domain knowledge and machine-learning alarm detection agents. In this paper we present and compare the adaptation capabilities of the both agent types to user feedback.

The paper is organized as follows: Section 2 presents related health monitoring systems and adaptation capabilities in the intrusion detection domain. In Section 3 we present Confidence. The alarm detection system, which constitutes a module in the Confidence system, is presented in detail. Section 4 describes the procedure for adapting the rule-based and machine-learning alarm detection agents. Section 5 presents the experiments for evaluating the adaptation of these agents as well as achieved results. Section 6 concludes the paper and states open problems for future work.

## 2 RELATED WORK

Remote health monitoring systems mainly focus on fall detection in domestic environment. Several systems were introduced able to detect falls and trigger an alarm. Zang et al. [2], for example, designed a SVM-based fall detector using a waist-worn accelerometer. Similar functionalities were presented by Kangas et al. [3] and also by Bourke et al. [4] using gyroscope, Fu et al. [5] using video, and Lustrek and Kaluza [6] using a localization system. There are also commercially available products, for example, by AlertOne [7] and Zenio [8]. Cesta et al. [9] proposed an agent-based approach for detection of inconsistencies in the execution of daily activities and reacting to exogenous events. All these systems constructed a model from learning data that remained unchanged after deployment at a user, meaning that the model was not able to adapt to the user and the environment.

On the other hand, there were several successful applications of adaptive incremental learning in intrusion detection. Pietraszek [10] introduced an approach for reducing false alarms by having a human analyst review the alarms and mark them as true or false, after which a new model was built by machine learning. NIDES system [11] generates user profiles that are constantly aged, meaning that new behavior is compared to the most recently observed one. Cannady [12] demonstrated the use of reinforcement learning method for adaptation according to user feedback, while Hossain and Bridges [13] proposed a fuzzy association rule system for adaptive anomaly detection.

We were unable to find any paper related to adaptive learning in remote care which was partly motivation for our work. Adaptation seems a necessity if one wants to build a truly user-friendly system. We propose methodology for adaptation of both machine-learning-based and rule-based systems. The latter is particularly relevant in the field of remote health monitoring, since a large number of remote health monitoring systems found in literature are based on expert knowledge.

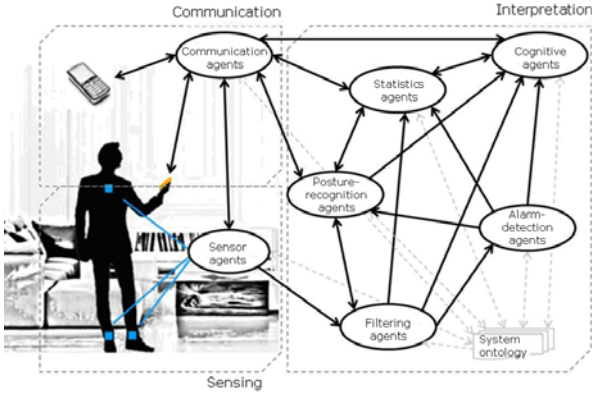
## 3 ALARM DETECTION IN CONFIDENCE

We have designed the Confidence system as a multi-agent system since agents are flexible, adaptive and robust. Figure 1 presents the architecture of Confidence. The user is tracked using *sensor agents* part of a localization system. The raw data is filtered by *filtering agents* and the posture of the user is recognized using *posture*

<sup>1</sup> Result d.o.o., Bravničarjeva 11, 1000 Ljubljana, Slovenia, e-mail: violeta.mircevska@result.si

<sup>2</sup> Jožef Stefan Institute, Department of Intelligent Systems, Jamova 39, 1000 Ljubljana, Slovenia

*recognition agents*. History of user posture and movement is used for alarm detection by *alarm detection agents*. When alarming situation is detected, it is reported to the user by *communication agents* either over a portable device that is part of the system or by a computer screen. In this paper we study real-time adaptation of alarm detection agents based on user feedback. Unusual movement such as limping is also detected in Confidence by tracking a variety of statistics (e.g. walking signature) by *statistics agents*. In this paper we will not present the statistics agents since they are independent of the alarm detection agents.



**Figure 1.** Architecture of Confidence.

For localization, we selected the commercially available Ubisense system [14], which can determine positions of a set of wearable tags using radio technology. Based on studies of posture recognition accuracy that can be achieved with different tag number and placement configurations [15], we placed wearable tags on the following parts of user's body: chest, belt (optional), left and right ankle. In a typical open-environment, the localization accuracy of Ubisense is about 15 cm, but in practice it may drop to 200 cm or more. What is more, sensor data is not necessarily available at each moment in time. Therefore, filtering agents were developed [16] in order to tackle the problems with the Ubisense system.

User's posture is classified as one of the following 5 postures: standing, sitting, lying, on all fours and intermediate states (falling, going down and standing up). Posture recognition was performed using rule-based and machine-learning agent, whose predicted posture is combined using merge agent in order to obtain final posture classification.

The rule-based posture recognition agent is used to reduce the gap between real-world and data captured in training phase. It consists of a set of rules for posture classification. Attributes that need to be included in a rule condition were determined by using knowledge extracted from decision trees and human know-how [17]. Two approaches for determination of limits for each condition in the rules were considered: (1) genetic algorithms and (2) approach which computes information gain for each attribute included in a particular rule in order to determine the most suitable rule condition limit. In this paper we use the information gain approach in order to determine the condition limits of the posture recognition expert knowledge agent.

The machine-learning posture recognition agent contains posture recognition knowledge captured in training data. It takes 10 successive sets of attributes computed for successive time frames as an attribute vector with the prevailing posture in those frames as the

class value. We have tested a variety of classifiers, including C4.5, Naïve Bayes, SVM, kNN, Bagging and AdaBoost [6]. Based on these studies, we selected machine-learning posture recognition agent to use the random forest ensemble classifier since it offers the highest classification accuracy.

The posture-recognition-merge agent takes into account the ability of the rule-based and machine-learning posture recognition agent to correctly classify each posture of interest. This ability can be easily determined by exploiting classification's confusion matrix from each of the two agents. In particular, for each posture of interest we compute its probability to be the true posture using the predictions by both the rule-based and machine-learning agent taking into account their ability to correctly classify each posture of interest. The posture with the highest probability is considered as true. This way the strengths of each agent are exploited when determining true user posture.

**Table 1.** Confusion matrix of joint classification using random forest and expert rules.

|                    | Standing | sitting | lying | intermediate state | on all fours |
|--------------------|----------|---------|-------|--------------------|--------------|
| Standing           | 72       | 9       | 0     | 19                 | 0            |
| sitting            | 7        | 83      | 0     | 3                  | 7            |
| lying              | 2        | 21      | 63    | 8                  | 6            |
| intermediate state | 36       | 19      | 6     | 37                 | 2            |
| on all fours       | 4        | 11      | 9     | 21                 | 55           |

The confusion matrix of the posture-recognition-merge agent is shown in Table 1. The left column shows the label of the correct posture, and the top row shows the assigned label. Except intermediate state and on all fours, all basic postures are detected reliably. Most errors appear on transitions from one state to another. These errors do not influence system performance since they introduce just time shift.

Alarm detection agents raise alarms when falls and sudden health problems of the user are detected. These are situations which are reflected by person lying/sitting at an inappropriate place (e.g. on the ground) for a prolonged period of time. Similarly to activity recognition, we considered a rule based and machine-learning based approach for alarm detection.

Rule-based alarm detection agents contain rules created by domain experts concerning situations in which an alarm should be raised. We considered four types of alarm situation (1) falling detected and person lying/sitting immovable at inappropriate place, (2) falling detected and person lying/sitting movable at inappropriate place, (3) person lying/sitting immovable at inappropriate place and (4) person lying/sitting movable at inappropriate place. Each rule-based alarm detection agent captures one type of alarm situation which it gives as output.

Machine-learning alarm detection agents contain knowledge about alarm situations that are induced from available data concerning such situations. The alarm detection module in Confidence contains two such agents, one containing SVM classifier, the other C4.5 classifier. They obtain percentage of all observed user's postures and movement in periods of 5 s, 10 s and 15 s as input. These intervals are suitable for our experiments because we considered a reasonable period of lying/sitting at inappropriate place after which an alarm should be raised somewhere between 5 and 15 s. In real life a longer period might

make more sense, in which case the intervals used in attributes should be lengthened. The output of the machine-learning alarm detection agents, unlike the rule-based agents, is only the presence or absence of emergency situation, without any information about its type.

When an alarm is raised, the communication agents notify the user either over a portable device that is a part of the system or by a computer screen. The user may (1) confirm the alarm (explicitly by pressing a button or implicitly by doing nothing), meaning it is true alarm and the user needs help, (2) inhibit the alarm, which means that this is a false alarm and (3) suspend alarm, meaning that the situation was actually a true alarm, but the user can help himself/herself. Additionally, the user may trigger alarm by himself/herself.

## 4 ADAPTIVE ALARM MODEL

Users behave differently and what might be alarm-worthy for one is normal for another. To give two extreme examples, one user might never voluntarily lie or sit on the ground because of a physical disability which prevents him/her from getting up again, whereas another might exercise regularly on the living room carpet. Our goal is to automatically adapt the alarm model to such differences, fine-tuning it to the needs and preferences of a particular user. This section presents the adaptation mechanisms of the rule-based and machine-learning alarm detection agents.

### 4.1 Rules adaptation

The problem of adapting the rule-based agents, whose agent function is represented in the form of a rule, is defined as a Markov decision process (MDP). Each rule can be thought of as a point in N-dimensional space, where N is the number of adjustable parameters in the rule. For example, the space of the rule «IF person lying on the ground **P**% in **T** seconds THEN alarm» is two dimensional, with one dimension representing the set of possible percentage (P) values and the other representing possible time interval (T) values (Figure 2). The goal state of the rule-based agents is dynamically changing as user's needs change. This adds uncertainty to the reward received for each agent action. Available agent actions in each state are all combinations of parameter value changes. Therefore, agent actions, unlike in problems usually modeled with MDP, are deterministic. The rule-based agents must determine optimal parameter values under uncertainty about their goal state.

The initial state of each rule-based agent is set to values predefined by domain expert. Returning to our example agent (Figure 2), using domain knowledge its agent function is initialized to «IF person lying on the ground **90%** in **8 seconds** THEN alarm».

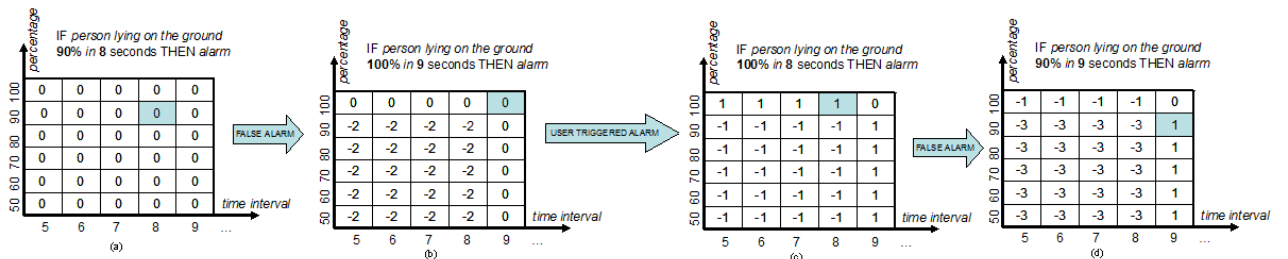


Figure 2 Rule-based agent adaptation

The reward matrix is initialized to zero for all states (Figure 2a). Without any feedback from the user, we do not have any information whether there exists a state which better suits user's needs. The agent does not change its state as long as there is no state with better utility than the utility of the current state. For example, because all states have zero utility at the beginning, the agent does not change its state, i.e. rule parameters. When an alarm is raised and the user responds with false alarm, the reward of the current state and all states that are dominated by it is reduced by penalty amount  $pa$ , in our case -2. Then the utility of being in the current state is reassessed. When the reward matrix is updated after the first false alarm (Figure 2b), it is easy to see that the current parameter values are no longer optimal. The agent chooses randomly one of the states with best possible utility. In the example in this figure, after the first false alarm the agent function is changed to «IF person lying on the ground **100%** in **9 seconds** THEN alarm». When the user triggers alarm by himself/herself, the reward of all states which are dominated by the current state is incremented by reward amount  $ra$ , in our case 1 (Figure 2c). Again, the utility of being in the current state is reassessed. After the user triggered alarm, the agent function of the example agent is changed to «IF person lying on the ground **100%** in **8 seconds** THEN alarm». Notice how the initial state is avoided because of the negative reward received during the first false alarm. This way the rule-based agents autonomously adapt their parameter values to achieve optimal utility in an environment with non-deterministic goal state.

### 4.2 Machine learning adaptation

The agent function of the machine learning agents is adapted by re-inducing the alarm detection classifiers every time new alarm situation examples are obtained. User triggered alarms and false alarms provide new true or false alarm examples. In the case of a user triggered alarm, the instance at the point when the alarm is triggered by the user and all following instances that dominate that instance with respect to the amount of person lying at an inappropriate place are added as true alarm examples. In the case of a false alarm, all the instances that were classified as true alarms are added as false alarm examples to the dataset. In order to escape classifier bias in case of unbalanced dataset, after each addition, the weight of the database instances is updated in order to bring the ratio of true alarm to false alarm to neutral examples (includes standing, sitting and lying on the bed) to 40 to 30 to 30. A new machine learning classifier for alarm detection is built on this dataset

## 5 EXPERIMENTS

To design and test performance of Confidence a room resembling a studio apartment equipped with a bed, few chairs and tables was set up. The apartment was divided into five regions: kitchen, living room, toilet, sleeping area (bed) and corridor.

### 5.1 Data

In order to test the adaptation capabilities of the rule-based and machine-learning alarm detection agents we created a set of true and false alarm scenarios. True alarm scenarios include (1) user falling quickly and then lying on the ground moving for 15 s, (2) user falling quickly and then lying immovable for 15 s, (3) user falling slowly and then lying moving for 15 s and (4) user falling slowly and then lying immovable for 15 s. The cases (1) and (2) represent tripping; (2) results in an injury that prevents movement. The cases (3) and (4) represent falling due to dizziness or fainting. In all true alarm scenarios the user raised alarm after lying on the ground for 7 seconds. Three types of false alarm scenarios were also recorded: (1) user on all fours on the ground for 10 s, (2) user on all fours for 5 s, then lying on the ground for 5 s and (3) user lying on the ground for 10 s. The user is moving in all three cases. These scenarios may represent the user searching for things under the table or bed. They differ from the true alarm scenarios by the length the user stays on the ground and in some cases the amount of movement.

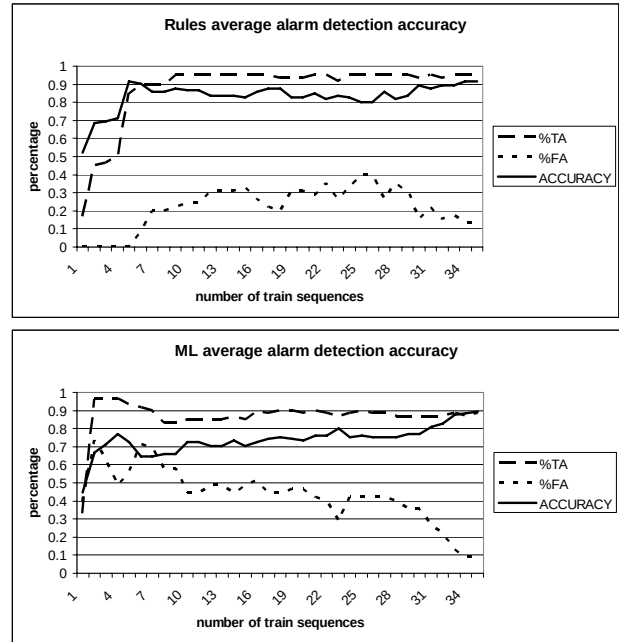
Each true and false alarm scenario was recorded five times for the purpose of training. Additionally, all scenarios were recorded five times for testing.

### 5.2 Results

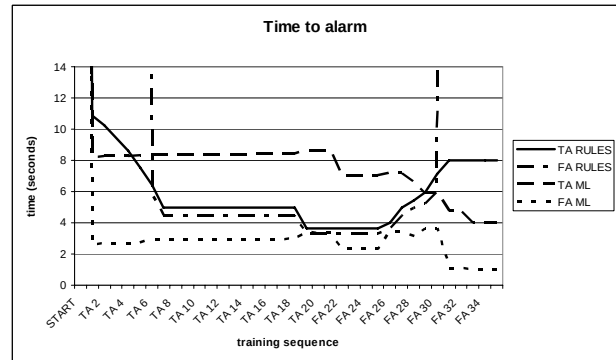
We tested the adaptation capabilities of the rule-based and machine-learning alarm detection agents based on two measures: alarm detection accuracy and time-to-alarm. Each test started with agents that are not able to recognize any true alarm situation in the test sequences. The training sequences were provided to the agents one by one. After each new training sequence, the agents were adapted and their alarm detection accuracy and time-to-alarm was tested on each of the five test sequences. As previously stated, in all true alarm scenarios, the user triggered alarm was raised after 7 seconds. Designing true alarm scenarios this way, we intended to force the classifiers to reduce their time-to-alarm to 7 seconds or less. However, if the agents fire alarm in the true alarm scenarios within 7 seconds, they will also fire alarms in the false alarm scenarios since they last for 10 seconds. Designing the false alarm situations this way, we intended to force the classifiers to require longer period of person lying on the ground (more than 10 seconds) before they raise an alarm. Obtaining information how the alarm detection accuracy of the agents changes over time provides information how good the agents can distinguish true from false alarm scenarios. Measuring the time-to-alarm, on the other hand, provides information whether the agents separate true and false alarm sequences by the period of user lying on the ground, which we consider most relevant for this task.

In order to test how agent's alarm detection accuracy changes after each adaptation step, three test runs were executed. In the first, all true alarm scenarios were presented to the alarm detection agents first, followed by all false alarm scenarios. In the remaining two test runs, the true and false alarm training scenarios were given randomly to the agents. By adapting the agents by various orders of

training scenarios, we gain information not only how their alarm detection accuracy improves, but also how much it depends on particular order of training scenarios.



**Figure 3** Average rule-based and machine learning agent alarm detection capabilities. %TA presents the percentage of correctly detected true alarms. %FA presents the percentage of alarms raised for false alarm test scenario examples. ACCURACY presents the overall achieved accuracy.



**Figure 4** Average rule-based and machine-learning agent time-to-alarm. TA RULES and TA ML present average time-to-alarm for the true alarm test scenario examples for the rule-based and machine learning agent respectively. FA RULES and FA ML present their time-to-alarm for the false alarm test scenario examples.

Figure 3 presents how alarm detection accuracy of the rule-based and machine-learning based agents changes after each adaptation step. The %TA line presents the percentage of correctly raised true alarms in the test alarm sequences. The %FA line presents the percentage of incorrectly raised false alarms in the test alarm sequences. The ACCURACY line presents the overall alarm detection accuracy of the alarm detection agents. From this graph we can see that both rule-based and machine-learning alarm

detection agents can adapt to reliably detect the alarms presented in the test sequences, reaching an accuracy of 90%.

We tested how agent's time-to-alarm changes by presenting all true alarm scenarios to the agents first, followed by all false alarm scenarios. Our intention was to: (1) teach the agents to raise alarm when the user has lied on the ground for a period of 7 seconds or less after all true alarm scenarios are presented and (2) increase time-to-alarm to 10 seconds or more when the agents are presented all false alarm examples.

Figure 4 presents how agent's time-to-alarm changes in such test run. The *TA RULES* line presents average time-to-alarm for the true alarm examples in the test sequences for the rule-based alarm detection agent, whereas the *FA RULES* line presents its time-to-alarm for the false alarm examples. Time-to-alarm for the true and false alarm examples of the machine-learning alarm detection agent is presented with the lines *TA ML* and *FA ML*, respectively. Rule-based agent's time-to-alarm follows our expectations. It falls to around 4 seconds after the rules are presented with all true alarm training scenarios, then raises to 8 seconds after the classifier receives the false alarm ones. The machine-learning agent, on the other hand, does not follow our expectations. Its time-to-alarm stayed fairly constant when the true alarm training scenarios were presented to the agents, then started to fall when false alarm training scenarios were presented to it. The time-to-alarm fell to 4 seconds in the end. This means that machine-learning agents learned to separate true from false alarm scenarios not according to the length of person lying on the ground, but by some other attribute. As future work we need to test if the machine-learning agents have created alarm detection classifiers that work in the general case and whether they are not over-fitted to these particular scenarios.

## 6 CONCLUSIONS AND FUTURE WORK

We presented adaptation of rule-based and machine-learning alarm detection agents based on received user feedback in Confidence, ubiquitous care system to support independent living. Markov decision process formalism was used for adapting the rule-based agents, whereas machine learning was adapted by re-inducing the alarm detection classifiers every time new data is obtained.

Experimental results show that both agents are able to adapt their agent function according to provided user feedback. On average they both achieved alarm detection accuracy of 90% on the performed test runs. The rule-based agent followed our expectations with respect to the length of user lying on the ground (which we considered as the most relevant for this task) needed in order to fire alarm. User triggered alarms influenced their time-to-alarm to be reduced, whereas false alarms influenced their time-to-alarm to be enlarged. On the other hand, machine-learning alarm detection agents did not follow our expectations. True alarms did not influence them by reducing their time-to-alarm. What is more, time-to-alarm was reduced when these agents were presented with false alarm training sequences. This means that the machine-learning agents found attribute other than the period of user lying on the ground for separating true from false alarm examples.

As future work we need to test whether the machine-learning alarm detection agents designed in this way work well in the general case and are not over-fitted to the particular scenarios used in this test. We also need to develop more intelligent way of choosing best rule-based agent state when the agent faces a situation with more than one state with the same highest utility.

## ACKNOWLEDGMENTS

This research is partly financed by the European Union, European Social Fund, partly by the Slovenian Research Agency under the Research Programme *P2-0209 Artificial Intelligence and Intelligent Systems*, and partly from the European Community's Framework Programme FP7/2007–2013 under grant agreement No. 214986. We would like to thank Domen Marinčič, Rok Piltaver, Boža Cvetkovič, Damjan Kužnar and Blaž Strle for suggestions, discussion and help with programming.

## REFERENCES

- [1] Confidence. <http://www.confidence-eu.org>, 2010-05-19.
- [2] T. Zhang, J. Wang, L. Xu, and P. Liu, 'Fall detection by wearable sensor and one-class SVM algorithm', *Lecture Notes in Control and Information Sciences*, pp. 858–863, (2006).
- [3] M. Kangas, A. Konttila, I. Winblad, and T. Jamsa, 'Determination of simple thresholds for accelerometry-based parameters for fall detection', *Proceedings of the 29th Annual International Conference of the IEEE, Engineering in Medicine and Biology Society*, pp. 1367–1370, (2007).
- [4] A. K. Bourke and G. M. Lyons, 'A threshold-based fall-detection algorithm using a bi-axial gyroscope sensor', *Medical Engineering & Physics* 30, pp. 84 – 90, (2008).
- [5] Z. Fu, E. Culurciello, P. Lichtsteiner, and T. Delbruck, 'Fall detection using an address-event temporal contrast vision sensor', *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 424–427, (2008).
- [6] M. Lustrek, and B. Kaluza, 'Fall detection and activity recognition with machine learning', *Informatica* 33(2), pp. 197–204, (2009).
- [7] AlertOne Services, Inc. iLife™ Fall Detection Sensor. <http://www.falldetection.com>, 2010-05-19.
- [8] Zenio, 'Zenio Fall Detector', <http://www.zenio.be/product/8.html>, 2010-05-19.
- [9] A. Cesta, and F. Pecora, 'Integrating intelligent systems for elder care in RoboCare', W. C. Mann and A. Helal (Eds): *Promoting Independence for Older Persons with Disabilities*, IOS Press, pp. 65-73, (2006).
- [10] T. Pietraszek, 'Using adaptive alert classification to reduce false positives in intrusion detection', *Proceedings of the Symposium on Recent Advances in Intrusion Detection*, pp. 102-124, (2004).
- [11] T. Lunt, 'Detecting intruders in computer systems', *Proceedings of the Symposium on Computer Security, Threats, and Countermeasures*, pp. 110–121, (1990).
- [12] J. C. Georgia, 'Next generation intrusion detection: autonomous reinforcement learning of network attacks', *Proceedings of the 23rd National Information Systems Security Conference*, (2000)
- [13] M. Hossain, and S.M. Bridges, 'A framework for an adaptive intrusion detection system with data mining', *Proceedings of the 13th Annual Canadian Information Technology Security Symposium*, (2001)
- [14] Ubisense. <http://www.ubisense.net>, 2010-05-19.
- [15] M. Lustrek, B. Kaluza, E. Dovgan, B. Pogorelc, and M. Gams, 'Behavior analysis based on coordinates of body tags', *Proceedings of the European Conference on Ambient Intelligence*, pp. 14–23, (2009).
- [16] B. Kaluza, and E. Dovgan, 'Glajenje trajektorij gibanja človeškega telesa zajetih z radijsko tehnologijo', *Proceedings of the 13th International Multiconference Information Society vol. A*, pp. 97-100, (2009).
- [17] V. Mirchevska, M. Lustrek, I. Velez, N. G. Vega, and M. Gams, 'Classifying Posture Based on Location of Radio Tags', *Ambient Intelligence Perspectives II*, pp. 85 – 92, (2009).



# Ubiquitous Data-Mining with Self-Organizing Maps

Bruno Silva<sup>1</sup> and Nuno Marques<sup>2</sup>

**Abstract.** Advances in technology are turning our mobile phones or PDAs into powerful computing devices capable of executing data mining algorithms. This paper discusses how the Self Organizing Map (SOM) algorithm can be adapted to a cooperating network of these devices. This approach opens the door to several new applications of data mining, including active data-collecting devices and on-demand knowledge visualization. Our results show that the method proposed is viable to use in ubiquitous environments.

## 1 Introduction

Nowadays, computing power is omnipresent, i.e. ubiquitous. In our every day life we carry with us a computing device - our mobile phone or PDA. Advances in technology (e.g., processing power, battery capacity) are pushing these devices towards a more powerful computing resource able to cope with demanding data-mining (i.e., knowledge extraction) algorithms. Hence, extracting knowledge from our day-to-day life is becoming feasible, opening a wide range of research in the near future. Moreover, the quantity of these devices that is present at any given time in a particular place gives us the opportunity to establish networks of devices where the processing can be distributed and knowledge shared.

The main idea of this paper is to take these facts into account and present a methodology that uses Self-Organizing Maps (SOM) to process acquired data from mobile devices and to share the individual knowledge of each device as a pair of devices is able to interact. We call this process *proximity-based knowledge sharing*. The SOM is an artificial neural network that is trained with an unsupervised algorithm to produce highly visual interpretations and clustering of the processed data. Situations such as:

- **City hot-spot mapper (modeling a distribution).** Several individuals are collecting random hot-spot coordinates on a map and sharing their knowledge when they meet.
- **An electronic gardener network.** Here, each person is actively measuring flowers, and from time to time, shares its knowledge with other gardeners.

seem to be very interesting real-world problems that can be tackled by self-organizing maps. The SOM is trained with observations (input samples), that describe a particular problem, and have the form of vectors which contain the values of attributes (features). The attributes characterize an observation and their number defines the dimensionality of the input space. For example, an observation describing a flower can take the form of  $\vec{x} = [SepalLength, SepalWidth, PetalLength, PetalWidth]$ .

The set of observations is a *dataset*. This is the type of data that a device may be acquiring over time. Both particular examples are illustrated in our experiments.

After we provide the motivation for this work in Section 2 and 3, we present our two main contributions contained in this paper:

1. We describe the SOM in terms of its architecture and training algorithm, characterizing it as an interesting tool for data-mining in ubiquitous environments. Here, we show the weaknesses of its original variant for such environments (Section 3). We then propose, in alternative, the use of a less-known variant of the SOM - the *Parameter-less SOM*, that can sustain an indefinite *plasticity*, where we justify such requirement (Section 3.1).
2. We present a methodology for using the SOM in a collaborative learning setting in ubiquitous environments in Section 3.2. We describe the problems associated with this goal and propose an interesting solution on how knowledge can be efficiently shared between pairs of devices. To the best of our knowledge, this has not been done before.

In Section 4 we present our experiments and results with the proposed method, simulating an ubiquitous environment where devices interact at random points in time. Our results show that the collaborative learning strategy allows for devices to acquire knowledge that, otherwise, they could not acquire independently. Finally, in Section 6 we present our conclusions and future work.

## 2 Motivation

At this moment, one of our scientific research areas is the distributed processing of neural-network algorithms. We believe that the computing power available nowadays, and scattered everywhere, can be used to tackle large problems, in a collaborative way. We divide these large problems into two kinds: first, we can have large amounts of data or large networks that need to be processed in a suitable timespan. In these cases we already have all the data and the challenge is to efficiently distribute the training process. We already addressed this problem with the SOM in a previous work [9]; second, we envision situations where neural-networks are *scattered around* processing a particular subset of data (belonging to the same problem) that they obtained independently of each other. Our pursuit in this latter case is to allow the sharing of "knowledge" between the individual neural-networks, i.e., to incorporate the knowledge obtained by one neural-network into another. We see this as a kind of distributed processing, where pieces of a problem are being tackled separately and the results combined to obtain the "big picture". We make our first attempt in combining knowledge from self-organizing maps in this paper. Ubiquitous environments present themselves as a typical scenario where this can be interesting and useful, focusing on mobile devices that are able to run a neural-network algorithm.

<sup>1</sup> Escola Superior de Tecnologia de Setúbal - IPSetúbal, Portugal, email:brunomnsilva@gmail.com

<sup>2</sup> Universidade Nova de Lisboa - Faculdade de Ciências e Tecnologia, Portugal

Our main idea resides in exchanging knowledge between devices when they are in presence of others. In technological terms, this can relate to a concept known as *personal area networks* (PAN), whose reach is typically a few meters. Wireless PANs (WPAN) can be established using technologies such as IrDA and Bluetooth. Bluetooth is nowadays a widely disseminated standard available on most mobile devices. It uses radio frequencies so devices do not need to be in line of sight of each other, in opposition to IrDA. Therefore, the establishment of PAN networks presents itself as an opportunity for collaborative learning strategies, which we explore in this paper.

### 3 Ubiquitous Self-Organizing Maps

The Self-Organizing Map (SOM) [6] was introduced by Kohonen in the early 1980s. It can be visualized as a sheet-like neural-network array, whose neurons become specifically tuned to various input vectors (observations) in an orderly fashion. They provide a way of representing multidimensional data on much lower dimensional spaces - usually two dimensions.

The training of these maps results in networks that store "compressed" information in their neurons. This particular process is often called *vector quantization* [8].

**Vector quantization** allows the modeling of probability density functions by the distribution of *prototype* vectors. It works by dividing a large set of vectors into groups having approximately the same number of points closest to them. Each group is represented by its centroid point, as in *k-means* [5], or by a neuron as in the SOM.

However, in SOMs, topological relationships in the input data are maintained on trained maps, allowing the application of visualization techniques. The SOM can be used for tasks of classification [3], clustering [4] and inspection of correlations in the underlying data [10].

The SOM is an *unsupervised* learning algorithm. Consider  $x = [x_1, x_2, \dots, x_n]$  an input sample belonging to an input space of dimensionality  $n$ . The basic architecture of the SOM network consists in  $K$  neurons arranged in a rectangular lattice, where the adjacency between neurons define their neighborhood. The size of the map is defined by its width and height, e.g.,  $10 \times 15$  SOM, hence  $K = \text{width} \times \text{height}$ . Attached to each neuron  $k$  is a weight (*prototype*) vector of the same dimensionality of the input space (Equation 1).

$$w_k = [w_{k1}, w_{k2}, \dots, w_{kn}], \quad k = 1, 2, \dots, K \quad (1)$$

The training process works iteratively and is dependent of time  $t$ , i.e., an input sample  $x(t)$  is presented to the network at each iteration. It is necessary to present the set of input data (the dataset) several times to the network, i.e., inputs are recycled. A complete presentation of the dataset is called an *epoch* and the training process usually endures for several epochs. The initial state of the network is typically random, using random prototype vectors. The training process can be resumed in two basic steps that are repeated over time, for each  $x(t)$ :

1. **Find the best matching unit for  $x(t)$ .** The distances from the input sample  $x(t)$  to all prototype vectors are calculated, usually with the Euclidean metric (Equation 2). Note that  $w_k(t)$  represents the prototype vector at instant  $t$ , since they will suffer changes over time.

$$d_k(t) = \|x(t) - w_k(t)\| \quad (2)$$

The best prototype that represents this input sample, i.e., the one that yields the smaller distance  $d_k$  to the input is found. The neuron that holds this prototype is called the *best matching unit* (BMU), denoted by the index  $c$  in Equation 3.

$$d_c(t) = \min_{\forall k} d_k(t) \quad (3)$$

2. **Adjust weights of the prototype vectors.** The BMU and its surrounding neighbors are "moved" towards the input, i.e., the weights are adjusted in order to become more similar to the input. The adjustments of the prototype vectors are obtained using Equation 4, where we denote the presence of scaling factors in the magnitude of the updates, i.e.,  $\alpha$  and  $h_{ck}$ . These are related to **training parameters** that influence the overall training procedure and resulting maps.

$$w_k(t+1) = w_k(t) + \alpha(t)h_{ck}(t)[x(t) - w_k(t)] \quad (4)$$

$$h_{ck}(t) = e^{\frac{-d(c,k)^2}{\sigma(t)^2}} \quad (5)$$

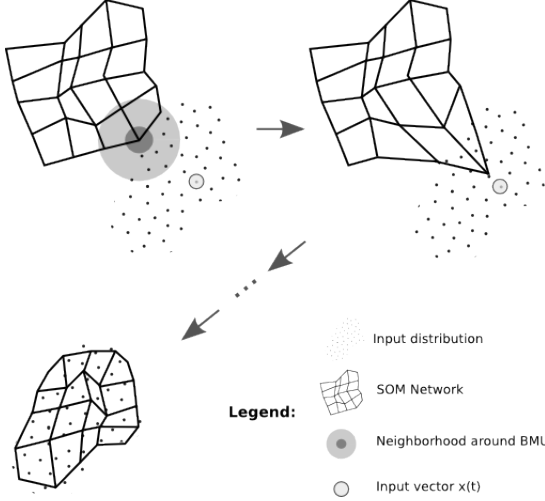
In the traditional SOM,  $\alpha$  is called the *learning rate* and  $\sigma$  stands for the *neighborhood radius*. The first defines how sensitive the updates should be regarding a input sample, while the second determines the region around a neuron that should be adjusted. It is indirectly present in the update equation through the neighborhood function (Equation 5). This function represents a Gaussian around the BMU that scales the magnitude of the updates based on the distance of a neuron to the BMU, i.e., the updates are smaller as the distance  $d(c, k)$  of a neuron  $k$  to the BMU neuron  $c$  in the lattice increases. These parameters are usually defined by their initial and final values and are decreased monotonically over time, hence  $\alpha(t)$  and  $\sigma(t)$ . After many iterations they hold small values and in consequence the updates made are very small and only in small regions - this is a process called *fine-tuning* - producing a fine-grained vector quantization.

The training process is illustrated in Figure 1 and results in a map that converges to a shape that models the density distribution of the input space in a topological ordered manner. On top of these trained maps, visualization techniques can be applied to extract knowledge from the underlying input space, namely clusters and correlations between features (variables that compose the vectors). We provide an example latter on, in Section 4.

#### 3.1 Preserving Plasticity in SOMs

The described training process of the SOM is ideal in most data-mining applications where all data is present *a priori*. However, the lack of *plasticity* of the map, i.e. the small adjustments at later iterations of the training, is not desirable in an ubiquitous environment where we are generating data on-the-fly, modifying our location and interaction with the device and interested in incorporating knowledge from other sources. We need a variant of the SOM that preserves some plasticity over time, such that it is able to cope with very different input samples, from what it has seen so far, at any point in time. For example, take Figure 1. If at a latter time in the training process the distribution was changed, where some input samples existed far away from the learned samples, the SOM would not learn the "new" distribution satisfactorily.

To overcome this, we propose in this paper the use of an under-used variant of the SOM, called the *parameter-less SOM* (PLSOM)



**Figure 1:** Illustration of the SOM training process. At each iteration an input sample is drawn from the input data, the BMU is calculated and the prototypes in its neighborhood are adjusted. This process repeats itself for a predefined number of iterations/epochs. At the end of training the network is able to model the input distribution density.

[2] (more precisely, its *improved* version [1]), for ubiquitous environments. PLSOM has only one parameter  $\beta$  that needs to be specified in the beginning of the training process, after which  $\alpha$  and  $\sigma$  are estimated dynamically at each iteration. The parameter  $\beta$  is called the *neighborhood range* and is kept constant throughout the training process. It defines the maximum value for the neighborhood radius. The basic idea behind the PLSOM is that for an input pattern that the network already represents well, there is no need for large adjustments - learning rate and neighborhood radius are kept small. On the other hand, if an input vector is very dissimilar of what was seen previously, then those parameters are adjusted to produce large adjustments.

Let  $qe(t)$  be the quantization error of a particular iteration, i.e., the normalized Euclidean distance of the input sample  $x(t)$  to the current prototype vector of the BMU  $w_c(t)$  (Equation 6).

$$qe(t) = \|x(t) - w_c(t)\| \quad (6)$$

which is basically the same as Equation 3. In the PLSOM the amplitude and range of the prototype updates are not dependent on the iteration number, but on how well the map fits the input data. Hence, the values of  $\alpha$  and  $\sigma$  are estimated using Equations 7 and 8.

$$\alpha(t) = \min\left(\frac{qe(t)}{S}, 1\right) \quad (7)$$

where  $S$  is computed with the algorithm in Figure 2, which estimates the diameter of the input space learned so far. The function  $diam()$  gives the diameter of a set, i.e., the largest distance between any two members of the set according to the Euclidean distance. Therefore, if  $\alpha(t)$  is large, the network fits the data poorly and needs large readjustments. Conversely, if  $\alpha(t)$  is small, the fit is likely to already be satisfactory for that input and no large update is necessary at time  $t$ .

$$\sigma(t) = \beta \cdot \ln(1 + \alpha(t)(e - 1)) \quad (8)$$

where  $\ln()$  is the natural logarithm,  $e$  is the Euler number and the neighborhood range:  $\beta = \text{constant}, \forall t$ . A suitable value for this pa-

Let  $d = 1 + n$ , where  $n$  is the dimensionality of the input space, since  $n + 1$  is the smallest number of vertices that can span a  $n$ -dimensional volume.

```

Initialize empty set A
S ← -1
for each input  $x(t)$  do
   $s \leftarrow diam(A \cup x(t))$ 
  if  $s > S$  then
     $S \leftarrow s$ .
    while  $size(A) \geq d$  do
      Remove the element of A that is closest to  $x(t)$ 
    end while
    Add  $x(t)$  to A
  end if
end for

```

**Figure 2:** Algorithm to estimate the diameter of the input space at every iteration of the PLSOM algorithm.

rameter can be  $\beta = \sqrt{(width - 1)^2 \times (height - 1)^2}$ , corresponding to the largest diagonal of the map.

This variant of the SOM retains an indefinite *plasticity*, which allows the SOM to react to very different input samples from what has been presented to it, at any point in time. We believe this is ideal for the objectives of our application in ubiquitous environments. The disadvantage of the PLSOM, regarding the standard SOM, is that the vector quantization achieved is not so fine-grained [2]. Nevertheless, we believe that the loss of precision of the quantization process is acceptable, given its clear advantage in ubiquitous environments. Subsequence references to the SOM, will address this particular variant - the PLSOM.

### 3.2 Exchanging Knowledge Between SOMs

The remaining problem is: *how to share the knowledge that each device acquires over time?* At this point we shall explicit the meaning of the words *data* and *knowledge* in this work. Data by itself does not necessarily represent knowledge in its primitive form. That is why data-mining algorithms are applied to sets of data to produce knowledge, i.e., information that is readily understandable. We consider a trained map to contain knowledge in an indirect form. The knowledge is concealed within the prototype vectors and the topology of the map. Visualizations techniques applied over trained maps allow the knowledge to arise in a visual form. As mentioned before, we present an example in Section 4.

So what should be shared between devices, and consequently between SOMs? The "dirty" solution would be to transmit the training data acquired by a device, incorporating it in the other device's training data. However, since these devices typically have short storage-space capabilities and bandwidth limitations, this can become impractical over time as new data is constantly being acquired. Please recall at this time that the SOM performs a type of vector quantization, i.e., the prototypes contain "compressed" data that models the input space density. An additional observation is that the quality of the quantization is given by the size of the map, i.e., larger maps produce better quantizations. Therefore, **we opt to share these prototype vectors**, which are typically in much less quantity than the number of inputs that are fed to the SOM and are **representative of the data acquired** and processed by a single device.

We call *shared prototypes* to the prototypes received from another device's SOM. They must be stored and recycled during training, so that the map is able to converge to a form that also models the shared knowledge (Section 3). Two problems arise: *how to incorporate the shared-prototypes received from several devices? And how to combine them with the samples being acquired independently by the device over time?*

First of all, a device may be acquiring new training samples over time, but this does not mean that this is being done in a constant rate or that new samples are being obtained regularly. If we just feed the input sample *on-the-fly* and throw it away, the SOM will not learn it well, e.g., if the processed number of samples is small, so they must be stored and recycled as well. On the other hand, there may be situations where storing all the input samples is impractical, e.g., when their number is very large. We solve this by using a *sample-buffer* that holds a specified number of input samples. While the buffer has space, the samples are being continuously recycled if needed. When it is full, the first acquired sample is discarded and the new one is added. This implements a type of moving-window across samples obtained over time. This way the SOM is adapting itself to the most recent data (ideal in a changing environment). However, if we do so, we can possibly lose the knowledge acquired earlier.

We address both problems by having two different SOM maps in each device - the *local-SOM* and the *shared-SOM*, which are processed in parallel:

**Local-SOM** It is responsible for processing the data in the *sample-buffer*, containing the latest data acquired independently by the device. It contains the *local prototypes*.

**Shared-SOM** It serves two purposes: retain "memory" of local past knowledge (from the Local-SOM) and incorporate knowledge shared by other devices. This map is trained exclusively with prototypes, which are stored in the *shared-buffer*. Its the prototype vectors of this SOM that are shared (transmitted) to other devices, since they contain all the knowledge that the device acquired over time.

The prototypes in the shared-buffer have different origins and this buffer is managed in the following manner: **when a device interacts with another, this buffer is emptied and filled with the local-prototypes, the prototypes from the local shared-SOM and the prototypes from the other device's shared-SOM.** The train of this map continues normally, but with this new set of samples. This is because its current state is an appropriate initialization to continue, since it already quantizes a percentage of the input space contained in the shared-buffer (the local shared-SOM prototypes).

One important observation is that if we don't discriminate the *weight* between local prototypes and shared prototypes while training the shared-SOM, this will cause the SOM to concentrate more prototypes in the input region defined by the local prototypes. This is a consequence of the ratio between the number of local prototypes and shared prototypes. For example, consider that both local-SOM and shared-SOM are  $10 \times 15$  SOMs. Also consider that the shared-SOM already contains knowledge from other devices acquired over time. The coverage of the overall input space should be significantly larger for the prototypes of the shared-SOM (combined knowledge), although the number of prototypes in the local-SOM and the shared-SOM is the same: 150. This means that the shared prototypes must have more "weight" in the update equation (Equation 4) than the local prototypes in the shared-SOM. We solve this by having attached to each shared-SOM a *variable* that counts how many times the

shared-SOM has been "combined" with others. This gives an overall measure of the "importance" of its prototypes. Hence, we refer to this variable as the *quantization-weight*, and it is denoted by  $\phi$ . It is incremented at each sharing of prototypes between shared-SOMs.

Given that the shared-SOM is always trained with prototype vectors, its update equation must be modified to cope with the prototype's "weight". We call this scaling factor *weight of the prototype* and denote it by  $\varphi$ . So,  $\varphi(x(t))$  is the weight of the prototype  $x$  fed to the shared-SOM at time  $t$ . Hence, the new update equation for the shared-SOM is formalized in Equation 9.

$$w_k(t+1) = w_k(t) + \varphi(x(t))\alpha(t)h_{ck}(t)[x(t) - w_k(t)] \quad (9)$$

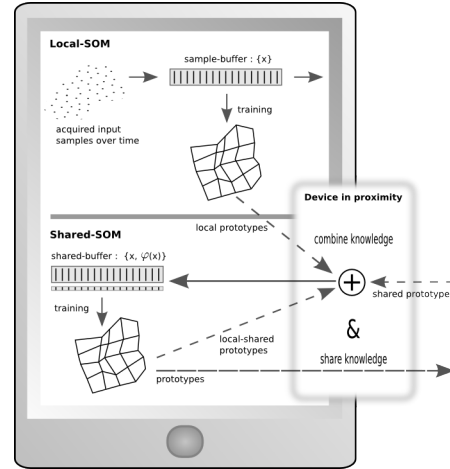
This way, the training input samples of the shared-SOM are tuples  $\{x, \varphi(x)\}$ , where the weight  $\varphi$  of the prototype  $x$  is assigned given its origin and its relative importance in the overall quantization process, i.e., given two devices  $A$  and  $B$ , each having its own shared-SOM, we first compute  $\phi_{max}$  with Equation 10.

$$\phi_{max} = \max(\phi_A, \phi_B) \quad (10)$$

where  $\phi_{max}$  holds the maximum quantization weight between the two device's shared-SOMs. Then, the  $\varphi(x)$  values are calculated using Equation 11.

$$\varphi(x) = \frac{\phi_M}{\phi_{max}} \quad (11)$$

where  $M$  stands for a local/shared SOM. The value of  $\phi$  for the Local-SOM (its quantization weight) is always 1.



**Figure 3:** Illustration of the methodology proposed for ubiquitous self-organizing maps (description in text).

We present an illustration of this methodology in Figure 3. One very important particularity derived from our experiments, is that after the *shared-buffer* has been emptied and filled with all the prototypes from the local-SOM, the local shared-SOM and the other device's shared-SOM, the **order by which the prototypes are drawn from this buffer must be random**, i.e., either by randomizing the buffer itself or drawing them in random. This is because sometimes the order by which samples are presented to the SOM affects the correct convergence of maps, especially in situations where subsequent sets of samples describe different regions of the input space. This

is usually the situation we get when simply appending the different prototypes by their source.

Yet, in situations where two devices have very similar knowledge, there should not exist the need to share prototypes. This can be seen as a performance enhancement in crowded environments. This is reserved for future work and it is discussed in Section 6.

## 4 Experiments and Results

We tested our proposed methodology with two different experiments that exemplify how an input space is collaboratively quantized and how the same knowledge can be extracted from different trained maps [9]. These translate to two requirements that we believe are the most important in proving that the method is viable: capability to model a distribution collaboratively and extraction of knowledge from a shared problem.

We simulated an ubiquitous environment where devices are put into presence of others and random points in time. The different runs had the duration of 400 seconds and each device was able to process a sample at every 0.5 seconds.

### 4.1 Experiment 1 - Modeling a Distribution

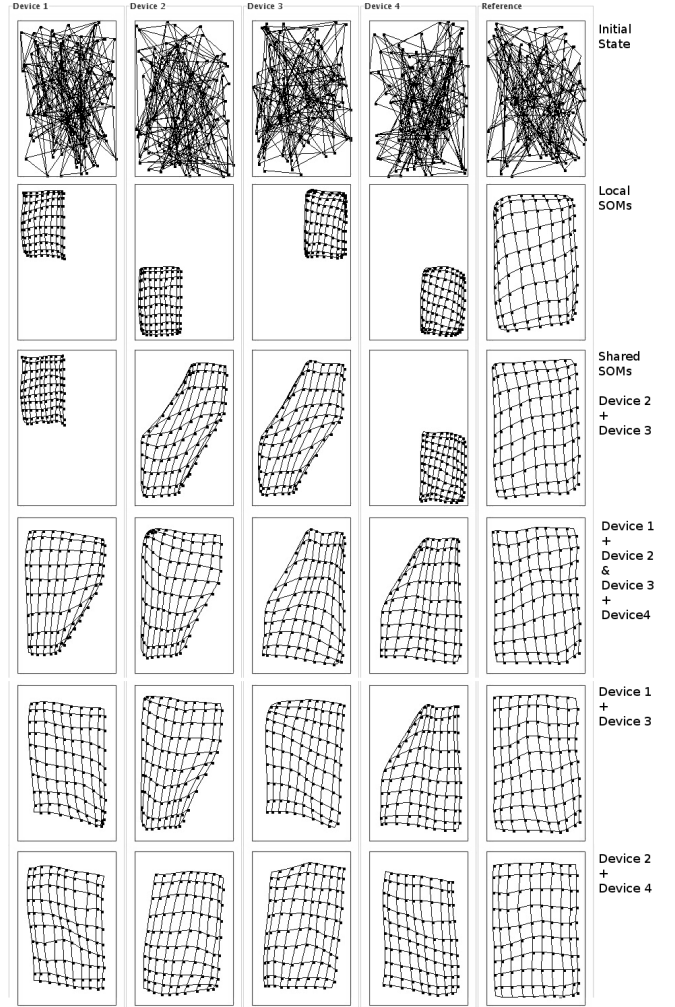
In this experiment we generated 800 two-dimensional uniform random samples in the range  $[0, 1]$ , i.e., points in a 2D plane, with an equal number of samples (200) in each quadrant. Each device was assigned the data from a different quadrant. The objective was to observe how well could the devices model the complete 2D plane (overall input space) by exchanging knowledge with each others. A "reference" SOM was trained with all the samples for comparison purposes. The maps were all  $10 \times 10$  in size.

One run was chosen to illustrate this experiment and can be seen in Figure 4. We used a mathematical measure called the *average quantization error* (AQE) to determine how well could the resulting SOMs quantize the input space. This gives a score between 0 and 1, that measures how well can a trained map quantize the input space. The value of  $AQE = 0$  means that the map can quantize the input space *perfectly*. This can only happen if the number of neurons in the map is equal, or larger, to the number of input samples and is rarely the case. Therefore, low values of AQE are desirable. The complete dataset is used on trained maps to produce these measures. Please recall that each device only had directly a portion of the input space, i.e., a quadrant, while it learned the rest of the input space by *proximity-based knowledge sharing*. We present the mean AQE for 10 runs of the simulated ubiquitous environment in Table 1.

| Mean Average Quantization Errors |          |          |          |          |
|----------------------------------|----------|----------|----------|----------|
| Reference                        | Device 1 | Device 2 | Device 3 | Device 4 |
| 0.041                            | 0.054    | 0.055    | 0.054    | 0.053    |

**Table 1:** Mean average quantization errors over 10 runs of the experiment for each device against the reference SOM.

It can be seen that the results are very similar. First of all, it should be noted that the SOM algorithm is *stochastic*, i.e., with differently initialized networks the resulting maps are rarely the same. Secondly, it is expected for the AQEs of the shared-SOMs to be slightly higher than the reference SOM's, because using the prototypes to describe an input space is a "lossy" procedure. Nevertheless, what matters is



**Figure 4:** One run of the ubiquitous environment for Experiment 1 (description in text). On the right side we present some relevant events that occurred over time.

how well they can quantize the input space and if the same knowledge can be extracted from different trained maps [9]. This experiment yielded very good results, which were validated empirically and mathematically.

### 4.2 Experiment 2 - The Electronic Gardner and Knowledge Extraction

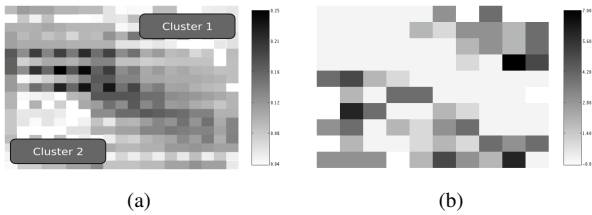
In this experiment we were interested to see if the "correct" knowledge from a data-mining problem could be extracted from the shared-SOMs. We used the well-known *Iris* dataset<sup>3</sup>, which consists of 150 observations of three types of flowers: *Iris-Setosa*, *Iris-Virginica* and *Iris-Versicolor*. Each flower has 50 observations. The observations are characterized by the following attributes (features): Sepal Width, Sepal Height, Petal Width and Petal Height. These define a 4-dimensional input space. Consequently, the SOM networks are

<sup>3</sup> We called this dataset *The Electronic Gardner Network* due to the adaptation to our ubiquitous simulation.

trained in this 4-dimensional space, not allowing a simple visualization - in opposition to Experiment 1.

Since the dataset at hand is wellknown, we know exactly what should be derived from a correctly trained map:

1. Two clusters should become visible with the U-Matrix. This dataset has two linearly separable clusters: one contains the Iris-Setosa flower; the other contains the Iris-Virginica and Iris-Versicolor flowers.
2. The features Petal Length and Petal Width are strongly correlated, i.e., one grows in proportion to the other. This should become visible through the Component Planes, where similar component planes (there exists one per each feature) indicate correlation.

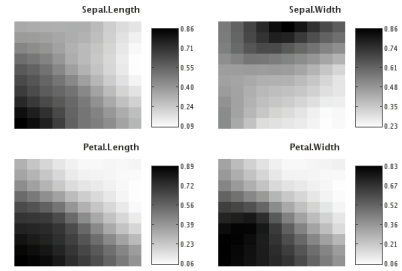


**Figure 5:** (a) U-Matrix of a trained shared-SOM with the Iris dataset, where dark areas denote the separation between clusters. Two clusters are clearly visible, as expected. (b) Histogram of projected samples over the trained shared-SOM. This allows to see how many input samples does one prototype represents, i.e., for which it was the BMU. The darker the color, the more input samples they represent. This is in accordance to the U-Matrix.

Each device was assigned the data for a particular flower, maintaining the conditions of Experiment 1. We took the obtained shared-SOM from one device and applied three visualizations techniques. Figure 5(a) shows the U-Matrix where two clusters are clearly visible. This is further enlighten by the map's histogram of projection of input samples in Figure 5(b). Finally, in Figure 6 we show the component planes for all four features. Observe that the component planes of Petal Length and Petal Width are identical. This clearly demonstrates that the method is able to derive global knowledge through the collaboration between devices and knowledge sharing.

## 5 Related Work

The work presented in this paper is closely related to the distributed training of the SOM. What differentiates this work is the fact that data is being acquired and processed independently by computing and sensing devices, e.g., mobile phones or PDAs. Then, the trained maps are combined at random points in time, allowing each device to incorporate the knowledge obtained by others. In other words, portions of a problem are being tackled separately and combined later on. We have already addressed the distributed training of the SOM in situations where all the data is available prior to training. In [9] we have combined two common partitioning schemes, e.g. data-partition and network-partition, in a hybrid algorithm that is very scalable. At this time, we have no knowledge of the application of self-organizing maps in ubiquitous environments and in the conditions presented. However, other machine learning algorithms have been ported to mobile devices, namely the k-neighbor classifier in [7].



**Figure 6:** Component Planes of the shared-SOM. Each feature is represented by a component plane and it can be seen that Petal Length and Petal Width are correlated, i.e.m their component planes are identical.

## 6 Conclusions

In this paper we presented a novel approach for using Self-Organizing Maps (SOM) in ubiquitous environments. We explored the everyday proximity/interaction between mobile devices to achieve a process that we call *proximity-based knowledge sharing*. It is a collaborative process in which the SOM in each device combines and exchange its knowledge with other devices. We proposed a underused variant of the SOM - the *parameter-less SOM* (PLSOM) - to be used in ubiquitous environments. We also proposed a solution that addresses the way in which knowledge can be combined between SOMs, preserving the importance of *locally* and *shared* acquired knowledge. We simulated an ubiquitous environment where devices interacted and were able to learn between each other. The results presented show that the methodology proposed works and it is quite interesting.

Future work will address mainly a measure that we call *SOM signature*, that will be used to compare SOMs in terms of the region of an input space that they cover. This will be very useful to compare two SOMs and decide if the exchange of knowledge is really necessary. This is particularly relevant in crowded environments.

## REFERENCES

- [1] E. Berglund, 'Improved PLSOM algorithm', *Applied Intelligence*, **32**(1), 122–130, (2010).
- [2] E. Berglund and J. Sitte, 'The parameter-less SOM algorithm', in *Proc. ANZIS*, pp. 159–164, (2003).
- [3] D.E. Callan, R.D. Kent, N. Roy, and S.M. Tasko, 'Self-organizing map for the classification of normal and disordered female voices', *Journal of Speech, Language and Hearing Research*, **42**(2), 355, (1999).
- [4] Arthur Flexer, 'On the use of self-organizing maps for clustering and visualization', in *Principles of Data Mining and Knowledge Discovery*, pp. 80–88, (1999).
- [5] E. Forgy, 'Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications', *Biometrics*, **21**(3), 768, (1965).
- [6] Teuvo Kohonen, *Self-Organizing Maps*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.
- [7] Ville Kononen, Jani Mantyjarvi, Heidi Simila, Juha Parkka, and Miikka Ermes, 'A computationally light classification method for mobile wellness platforms', *Conf Proc IEEE Eng Med Biol Soc*, 1167–70, (2008).
- [8] J. Li and C. N. Manikopoulos, 'Multi-stage vector quantization based on the self-organization feature maps', *Visual Communications and Image Processing IV*, **1199**, 1046–1055, (1989).
- [9] Bruno Silva and Nuno Marques, 'A hybrid parallel som algorithm for large maps in data-mining', in *New Trends in Artificial Intelligence*, (2007).
- [10] Bruno Silva and Nuno Marques, 'Feature clustering with self-organizing maps and an application to financial time-series portfolio selection'. To appear in International Conference on Neural Computation, 2010.

# Distributed and real time Data Mining in the Intensive Care Unit

Marta Vilas-Boas<sup>1</sup> and Pedro Gago<sup>2</sup> and Filipe Portela<sup>1</sup> and  
Fernando Rua<sup>3</sup> and Álvaro Silva<sup>3</sup> and Manuel Filipe Santos<sup>1</sup>

**Abstract.** Pervasive computing, ubiquitous computing and ambient intelligence are progressively influencing health care and medicine. In this paper we present our work regarding the development of the INTCare System for decision support in Intensive Care Units (ICU). INTCare gathers data from several sources (e.g. bedside monitors, drugs system, lab results, the Electronic Nursing Record, etc) and is able to use that data to make predictions about organ failure and final outcome for each of the patients in the ICU. Moreover, the system continuously monitors its performance levels and is able to automatically adapt so that performance doesn't drop over time.

## 1 Introduction

The high volume of clinical information that doctors have to deal with everyday in Intensive Care Units (ICU's) explains the need to rely on Information and Communication Systems (ICS), referred in this article as a tool for the decision making process.

Hence the interest by intensive care clinicians and researchers in the development of prediction models based on Data Mining techniques (DM) to support clinical decision. Knowledge discovery and data mining are related to finding new information, knowledge and patterns which may predict future behavior and improve business performance [1]. The emergence of network-based computing environments established a new dimension to this area, regarding distributed sources of data and computing. The growing use of laptops, palmtops and wearable devices is making ubiquitous access to large quantity of distributed data a reality [2]. ICS are present in nowadays ICU's; however, there is an emergent trend in regard of this matter. Pervasive computing has been pointed as a new paradigm for the 21st century. Computing has evolved itself beyond the desktop PC. Significant hardware developments, as well as advances in location sensors, wireless communications, and global networking, have advanced towards technical and economic viability [3].

Pervasive computing, ubiquitous computing and ambient intelligence are progressively influencing health care and medicine [4].

The concept of pervasiveness is often applied to refer "numerous, casually accessible, often invisible computing devices, frequently mobile or embedded in the environment and connected to an increasingly ubiquitous network structure" (National Institute of Standards and Technology, 2001). A study by [4] reveals that 51% of the systems in pilot phases feature mobile devices. Regarding the systems' function, 60% are related to analytical and diagnostic support, followed by alerting (60%). There are no references to systems addressing medical treatments.

This paper presents the INTCare system as an Intelligent Decision Support System for ICU in real time, based on Knowledge

Discovery in Databases (KDD) and Data Mining (DM), and its pervasive and ubiquitous features.

## 2 Background

Ubiquitous computing, pervasive computing and ambient intelligence are concepts evolving in a growing number of applications in health care and are increasingly influencing it [4]. However, as pointed in the study of [4], the majority of these systems are in their prototype stages and there is a need for advanced research on their deployment, mainly clinical studies, economic and social analyses and user studies.

Mark Weiser had described his vision of ubiquitous computing, also called pervasive computing, whose essence is that "the most profound technologies are those that disappear." [5]. The highlight is the development of environments with computing and communication capability and yet smoothly integrated with human users.

Ubiquitous computing can transform some key features of user interaction: their location, the scope of the service, and its duration and frequency [6].

Safer, less expensive, and higher-quality health care can be achieved using clinical decision support systems (Neves, 2008).

Computer-based decision support tools are supposed to help practitioners avoid errors, ensure quality and improve efficiency in healthcare. The main focus is the prevention and early detection of disease. As tools designed to improve clinical decision making, computerized clinical decision support systems enable that characteristics of individual patients are matched to a computerized knowledge base, and software algorithms generate patient-specific recommendations [7]. Health care providers can manually enter patient characteristics into the computer system, but ideally, information retrieval, registration and display should be as automated as possible. Computer-assisted decision support systems must be able to combine the heterogeneous clinical data types into a seamless system. Intelligent agents, an approach that has been used essentially in business applications, provides a structure that can combine not only data types but also a variety of reasoning methodologies in the same decision support system [8].

Advanced analysis of distributed data for extracting useful knowledge is the next expected step in the increasingly connected world of ubiquitous computing [1]. In this context, ubiquitous data mining is attracting a great amount of attention. Much of the data mining tasks which are being done currently focus on databases or data warehouses whose data is physically located in one place. However, the scenario arises where information may be located in different physical locations. Therefore, the goal is to effectively mine distributed data streams which are located in heterogeneous sites [9].

<sup>1</sup> Departamento de Sistemas de Informação, Universidade do Minho – Portugal – mvb@dsi.uminho.pt, mfs@dsi.uminho.pt, cfp@dsi.uminho.pt

<sup>2</sup> Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria – Portugal – pedro.gago@ipleiria.pt

<sup>3</sup> Serviço de Cuidados Intensivos, Hospital Geral de Santo António, Porto – Portugal – moreirasilva@clix.pt, fernandoruia.sci@hgsa.min-saude.pt

### 3 The INTCare system

INTCare is a situated Intelligent Decision Support System (IDSS) for intensive medicine that is being developed in the ICU of the HGSA in Porto, Portugal. The development and deployment of such system is driven by intentions of improving healthcare in organizational and medical perspectives.

INTCare is based on KDD and DM models and it makes use of intelligent agents that perform autonomous actions in order to meet its goals. The system's agents have the capability of acquiring data that will feed data mining models, which will predict, in real-time, organ failure of six organic systems (cardiovascular, respiratory, renal, liver, coagulation and neurological) and mortality assessment at the time of discharge (dead or alive) and, according to these predictions, suggest therapeutic treatment. In critical situations, such rapid access to knowledge saves lives. It should consider the inclusion of the data available as it is registered (i.e. online data) to guarantee that all relevant clinical information is available and regarded in decision support in real-time.

INTCare is being developed to work in real time and online, i.e. the data to generate the models is gathered in real time and automatically feeds the models so that the prediction of the patient condition is immediate. The inclusion of the data available as it is registered (i.e. online data) guarantees that all relevant clinical information is taken into account for decision support in real-time. Also, it avoids the need for manual data preparation and integration, due to its semi-autonomous features.

The development of the system within the ICU must be in synchronization with the whole information system and activities within the unit and the hospital. Patient care is supported by complex information systems, which brings the need for integration of the various types and sources of data.

The INTCare system uses data gathered by the bedside monitors (e.g. heart rate) and the Electronic Nursing Record (ENR) (e.g. lab results). These data sources produce and feed information to the INTCare system. "Everywhere, anywhere, always on, anytime" are keywords that have been pointed as underlying concepts regarding pervasiveness and ubiquity and, in this perspective, the INTCare system accomplishes that all the needed data must be available, readable and interpretable anywhere in the context of clinical decision support and by the medical staff (e.g. laptops with Wi-Fi connectivity inside and outside ICU, PDA's, mobile phones) (Fig. 1 and 2).

Also, Radio Frequency Identification (RFID) technology is a good example of a pervasive technology where miniscule tags embedded in the environment support automatic sensing (Kietzmann). In the INTCare system, patients will be identified using this technology to guarantee that the various data streams of clinical data are related to a particular patient and consequently, avoid serious damage that might occur if the data received does not match the patient identified. In conclusion, RFID identification will provide real-time interactivity between the context of work and the information recorded about that context. This context-aware scenario has also been addressed by Hospital of the future [10] and Hospital without walls [11].

Furthermore, computer-generated recommendations, reminders and alerts are delivered to the medical staff through the electronic nursing record, PDA's and sound devices. Predictive analysis, recommendations and alerts are generated combining clinical information of various sources (e.g. bedside monitors, electronic patient record) and treated by DM algorithms. This collaboration process is transparent to the medical staff, and it is context-aware,

features that are closely related to pervasive and ubiquitous computing [12].

Figure 1 represents the system's functionality in Real Time and in Real World. It shows all the possible communications that can exist in the Hospital throughout the world. The system is implemented in an ICU in Oporto. It is accessible from the web and all data is stored in a data Server. Anyone who is authorized may consult data or see the patient's clinical information; they only need a browser to connect to the internet and access the web-based interface of the system. The type of connection will be secure and only users with privileges to access the information can do so. If there are alerts or pertinent situations regarding a patient, the system sends alerts to doctors by message or email. Doctors can access all patient information independently of their location, as shown in Figure 1.

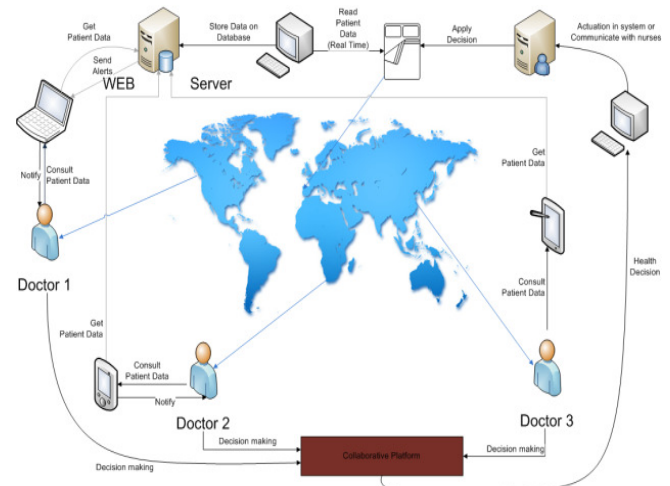


Figure 1. INTCare system

This decision support system is also sustained by a collaborative platform and intelligent agents. In a healthcare unit with a distributed computational environment, different services and people must be in contact and communicate, exchanging data and knowledge, as shown in Figures 2 and 3. Users of this system can communicate and discuss to find the better solution for the patient. After that, the decision will be communicated to the nursing staff in order to treat the patients. Furthermore, doctors can configure the system to adjust procedures and medications related to the patients (e.g. configure ventilation system, schedule an exam, administrate a drug).

In Figures 2 and 3 it is shown the system's ubiquity and how it works. It comprises two phases. The first phase is how doctors can consult the patient data, and the second how they can use that information to decide. In the first phase doctors can get patient information by laptop, mobile phone, tablet or other web-based device. The data is on the Electronic Nursing Record (ENR) and comes from four heterogeneous types of data sources. Doctors can receive alerts by the ENR and can access data whenever they want to. They can consult the information in two ways: Processed information generated by the predictive DM models, or data in raw. After consulting the information, doctors may step to the next phase, which is the decision phase. In the second phase doctors decide if they want a single decision or not. If they wish, they may decide with collaboration of other doctors through the collaborative platform. They can decide based on the decision models

recommend by the INTCare system about the best action towards the patients best interest. After the decision is done, doctors can use mobile devices with internet connection to disseminate the decision, whether by actuating directly in the system or communicating with the nursing staff. Finally, the decision will be applied and two things may be improved: the patients' health condition and the predictive and decision models that doctors can use. These tasks can be performed any time in any place.

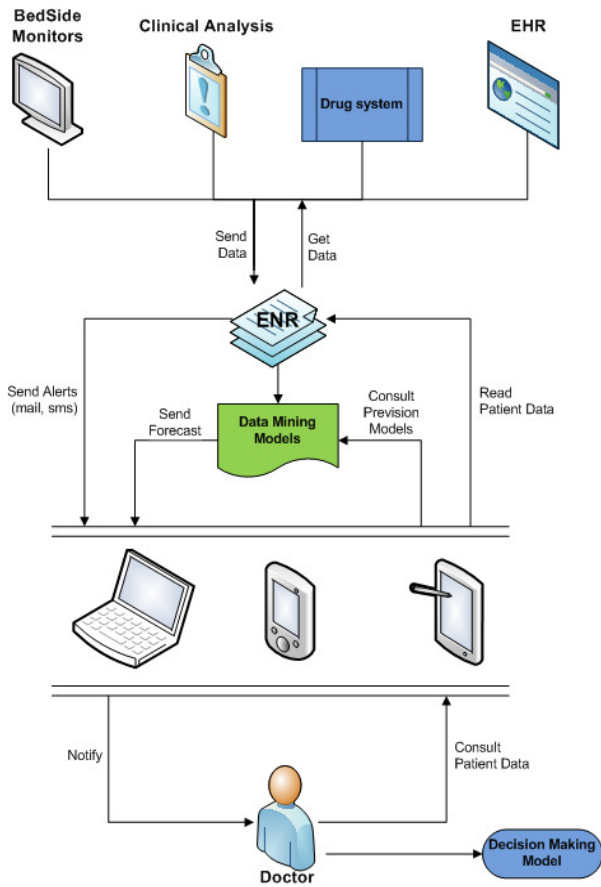


Figure 2. Data access

The Data that can be consulted is retrieved from Sensor Networks. The DM models are obtained through processes of learning from Sensor Networks and information systems. That process uses sensors to collect data to database. The sensors work in the network, they receive information from the patients (e.g. vital signs) and send it to the database and some application that use that data in moment that these were collected.

Next, this process uses On-line Learning Algorithms to put information on web (e.g. ENR) that can be accessible online and in Real Time.

The data is acquired from Bedside Monitors, Drugs System, Electronic Health Record, lab results and other electronic devices that might collect information regarding the patients' clinical condition. The Electronic Nursing Record is a tool for integrating and subsequently accessing patient data. The digital format of the ENR allows data to be easily searched and retrieved. Due to the medical information sources being distributed, heterogeneous and complex, the ENR takes advantage of the Agency for Integration,

Diffusion and Archive of Medical Information (AIDA) implemented at the hospital [13], [14]. It is known that the ENR simplifies the work of the nursing staff, reduces number of paper-based records and minimizes the access time to information [15].

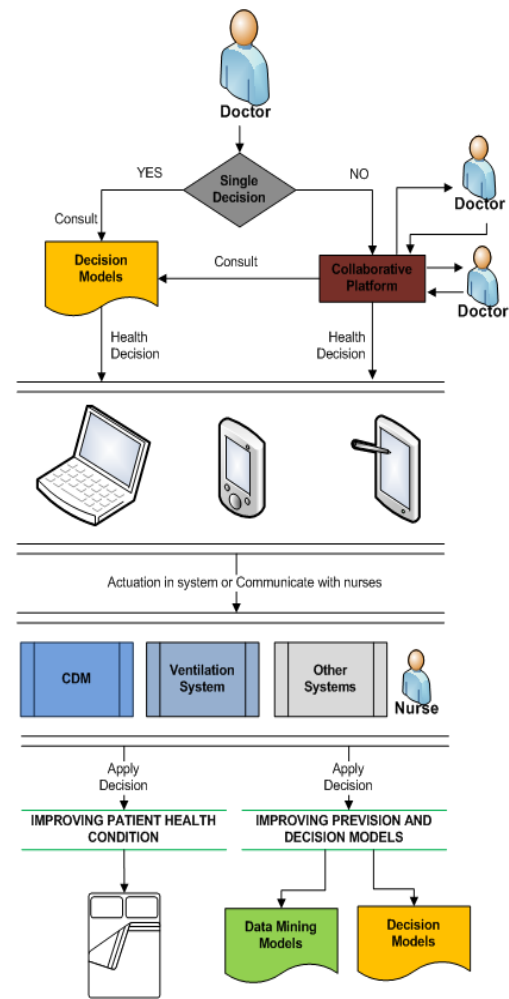


Figure 3. Decision Support

Group Decision Support Systems (GDSS) is the collaborative platform in use in the hospital, and it comprises a set of computer-based systems that aim to support collaborative work. Particularly, it increases the benefits and decreases the losses associate to collaborative work [16]. Users may call meeting to all the processes necessary to the conclusion of a specific collaborative task [17].

### 3.1 Data Mining Models

The main purpose in ICU is to avoid or reverse organ failure. We expect that, with an hourly prediction of the patients' clinical condition, it will be possible for the healthcare professionals to have a timely intervention and a proactive attitude so that worst complications for the patients may be avoided. Models were developed for the prediction of organ dysfunction/failure of the cardiovascular, respiratory, renal, coagulation and liver systems, as well as the final outcome.

The DM techniques used for the experiment were Artificial Neural Networks (ANN), Decision Trees (DT), Regression and Ensembles.

### 3.1.1 Data description and features selection

The data used to generate the DM models originates from three distributed and heterogeneous sources: EHR (lab results), bedside monitors (blood pressure, heart rate and oxygen saturation) and nursing records (urine output). Variables containing the case mix (information that remains unchanged during the patient's stay - age, admission from, admission type) were also considered. It was also included some calculated variables: Critical Events (CE), SOFA scores and a set of ratios relating the previous variables to the patients' length of stay. The data was gathered in the ICU of HGSA and it was collected in the first five days of stay of thirty two patients.

For the prediction of each organic system and outcome, it was explored three scenarios regarding the inclusion of the variables mentioned above – M1, M2, and M3 – where

M1 = {Hour, Case Mix, CE}

M2 = {Hour, Case Mix, CE, Ratios}

M3 = {Hour, Case Mix, CE, SOFA}.

### 3.1.2 Results

The DM models generated so far show promising results, with high accuracies. Table 1 shows the results for cardiovascular, respiratory, renal, liver, coagulation and neurological and outcome in terms of sensitivity (i.e. percentage of failure and death correctly classified as such) as well as the scenario that produced the best results. The models were developed for hourly prediction with the intent to make predictions as fast as possible, in the patients best interest.

**Table 1.** Sensitivity of the DM models by system and outcome.

| System         | Sensitivity (%) | Scenario |
|----------------|-----------------|----------|
| Cardiovascular | 93.4            | M3       |
| Respiratory    | 96.2            | M2       |
| Renal          | 98.1            | M3       |
| Coagulation    | 97.5            | M2       |
| Liver          | 98.3            | M3       |
| Outcome        | 98.3            | M1       |

## 4 Adaptive Data Mining

The performance of predictive models is expected to decrease as new data arrives. However, this is not a desired result and human intervention is usually required in order to restore previous performance levels. Indeed, the DM process must be repeated in order to obtain new (and hopefully more accurate) prediction models.

Nevertheless, this procedure is time consuming and demands an expert to be on site. This is clearly undesired and we have been working on methods that lead to semi-automation of the models updates. In order to do so we experimented using ensembles and were able to define a methodology that not only avoided the

decrease in performance as new data arrived but even increased the system's performance [18]. Tests were conducted on historical data (collected manually by the nursing staff).

## 4.1 Experimental Setting

To evaluate our system's ability to adapt we used an ensemble of predictive models. Ensembles are known to outperform individual models [18].

Using algorithms from the Weka library [19] we created an ensemble of decision trees and neural networks using part of the data for the initial training of the models in the ensemble.

Next, each of the remaining records was used for evaluation and evolution of the ensemble. During that phase we allowed the weights to be changed so that the best models got higher weights while the worst models had their weights decreased. More information about this weight update procedure can be found in [18].

## 4.2 Results

We compared a "traditional" ensemble (where all the models contributed the same to the solution) to the ensemble where model weights were updated dynamically.

To evaluate the results we used the average of the values of the area under the Receiver Operating Characteristic curve (AUC ROC) obtained after 30 runs of each experiment. The results are presented in Table 2. It is evident that results are better when we dynamically adjust the weights in the ensemble.

**Table 2.** Area under the ROC curve.

| System                   | AUC ROC (%)  |
|--------------------------|--------------|
| "Traditional" Ensemble   | 74.05 ± 0.77 |
| Dynamic weights Ensemble | 77.41 ± 0.32 |

## 5 Discussion

In our work we collected data from several sources and combined it in order to obtain a better understanding of the patient condition. Next, several Data Mining techniques were used to create predictive models and the results achieved are very encouraging in some of the scenarios evaluated. But, even if good results were achieved, decision support systems should be able to adapt and maintain good performance without the need for human intervention. Results achieved with our work on adaptive Data Mining seem to indicate that it is possible to build such decision support systems. When using an ensemble where we dynamically adjusted the models weights according to their performance we consistently achieved better results than when using a static ensemble.

## 6 Conclusion

More and more medical devices allow data collection and efforts towards electronic health records can be found in most hospitals. This creates huge demands for data merging as it is necessary to

gather data from a growing number of sources. Also, medical professionals are eager to have available the tools that will enable them to make use of all this data to improve quality of care. An effort must be done so that access to relevant information is timely and very user friendly. Moreover, more and more doctors are making use of mobile devices and plans must be made to allow them to use those tools as they can ease their tasks. Additionally, there is a need for real-time decision support systems capable of using the wealth of data that is becoming available. Besides providing decision support those tools should also be able to adapt to changes in their environment. The INTCare system is providing answers to several of these questions.

## REFERENCES

- [1] E. I. Neaga and J. A. Harding, "An enterprise modeling and integration framework based on knowledge discovery and data mining," *International Journal of Production Research*, vol. 43, pp. 1089-1108, 2005.
- [2] H. Kargupta, "CAREER: Ubiquitous Distributed Knowledge Discovery from Heterogeneous Data," 2001.
- [3] D. Saha and A. Mukherjee, "Pervasive computing: a paradigm for the 21st century," *Computer*, pp. 25-31, 2003.
- [4] C. Orwat, *et al.*, "Towards pervasive computing in health care \$x02013; A literature review," *BMC Medical Informatics and Decision Making*, vol. 8, p. 26, 2008.
- [5] M. Weiser, "The computer for the 21st century," *Scientific American*, vol. 272, pp. 78-89, 1995.
- [6] A. Fano and A. Gershman, "The future of business services in the age of ubiquitous computing," *Communications of the ACM*, vol. 45, p. 87, 2002.
- [7] A. X. Garg, *et al.*, "Effects of computerized clinical decision support systems on practitioner performance and patient outcomes: a systematic review," *Jama*, vol. 293, p. 1223, 2005.
- [8] D. L. Hudson and M. E. Cohen, "Use of intelligent agents in the diagnosis of cardiac disorders," *Computers in Cardiology*, pp. 633-636, 2002.
- [9] J. Hsu, "Data mining trends and developments: The key data mining technologies and applications for the 21st century," 2002, pp. 1542-7382.
- [10] J. E. Bardram, "Hospitals of the future—ubiquitous computing support for medical work in hospitals," 2003.
- [11] D. Cramp, *et al.*, "Hospital without walls," *Computer Methods and Programs in Biomedicine*, vol. 64, pp. 153-154, 2001.
- [12] H. Ogata and Y. Yano, "Context-aware support for computer-supported ubiquitous learning," 2004, p. 27.
- [13] A. Abelha, *et al.*, "Agency for Integration, Diffusion and Archive of Medical Information."
- [14] A. Abelha, *et al.*, "Health data management in the medical arena," 2004, pp. 1-6.
- [15] R. E. McMillen, "End of life decisions: nurses perceptions, feelings and experiences," *Intensive and Critical Care Nursing*, vol. 24, pp. 251-259, 2008.
- [16] G. Marreiros, *et al.*, "ABS4GD: A Multi-agent System that Simulates Group Decision Processes Considering Emotional and Argumentative Aspects," 2008, pp. 88-95.
- [17] M. Miranda, *et al.*, "A group decision support system for staging of cancer," *Electronic Healthcare*, pp. 114-121.
- [18] P. Gago and M. Santos, "Evaluating Hybrid Ensembles for Intelligent Decision Support for Intensive Care," *Applications of Supervised and Unsupervised Ensemble Methods*, pp. 251-265, 2009.
- [19] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*: Morgan Kaufmann Pub, 2005.



# Learning Abnormal Gait Patterns of Elderly from Motion Capture System

Bogdan Pogorelc<sup>1,2</sup> and Matjaž Gams<sup>1,2</sup>

**Abstract.** We propose a system for learning abnormal gait patterns of elderly from motion capture system data for the purpose of supporting their independent living. Gait is captured with the motion capture system, which consists of tags attached to the body and sensors situated in the apartment. Sensors acquire position of the tags and the resulting time series of position coordinates are analyzed with machine learning algorithms in order to identify the abnormal gait, indicating health problem. The presented study proposes novel features for training a machine learning classifier that classifies the user's gait into 1 normal and 4 abnormal classes. Results show that decision tree classifier is overall less accurate than support vector machines, which needs at least 8 tags and up to 10 mm noise to reach the classification accuracy of 98 %.

## 1 INTRODUCTION

The developed countries are facing a hard demographic problem. Due to the ever-growing share of elderly, the active population's capacity for taking care of its elderly members is decreasing [1].

Moreover, elderly want to live in their homes for as long as possible, even alone. But for elderly living at home alone the risks increase, e.g. of a potential health problem undetected until serious consequences emerge. To overcome this problem, we propose a system for automatic ubiquitous elderly health care, for which we have developed advanced techniques for recognition of common health problems manifesting in gait of elderly. In case the system recognizes a health problem, it automatically notifies the physician and displays the explanation of the diagnosis through visualization of the kinematic model. In this way, the system provides elderly with constant health care at their homes, relatives get assured that elderly are not left alone, and physicians get a valuable assistance in constant surveillance. The physicians are still the only one to confirm/reject the automatic diagnosis, but the information can be revealed to the elderly or their relatives if desired by the elderly. In this case elderly gain constant (ubiquitous) support, providing them safety and confidence.

The target health problems for the automatic recognition are: hemiplegia (usually the result of a stroke), Parkinson's disease, pain in the leg and pain in the back. The gait of the user is captured with the motion capture system, which consists of the

tags attached to the body. The position of the tags is acquired by the sensors situated in the apartment and the resulting time series of position coordinates are analyzed with machine learning algorithms in order to recognize the specific health problem.

The first objective of the research is to discover to what extent the automatic recognition of the health problems with motion capture system is feasible. The second objective is to investigate the classification accuracy achievable using various numbers/placements of tags on the user's body and various amounts of noise in tag coordinates. Tag placement must achieve a trade-off between usability and technical requirements – the users prefer as few tags as possible, but too few tags cannot ensure sufficient accuracy. Both the finding regarding noise and tag placement can affect motion-capture system selection and further development and applications of care systems for elderly.

## 2 RELATED WORK

**Motion capture.** For the automatic recognition of the movement (gait) pattern, the movement must first be captured. For this purpose many types of motion capture devices exist. Widely used are inertial sensors composed of accelerometers or gyro sensors [2]-[4]. The second widely used approach uses video-image processing for the reconstruction of the human body movement [20], [23]. The third approach uses cameras in combination with tags attached to the body. Usually infra-red (IR) cameras are used and the body posture is reconstructed from the position of the retroreflective tags [12], as in our approach.

There also exist some specific measurement devices for the recognition of tremor – a symptom in Parkinson's disease, but not in hemiplegia, pain in the leg and pain in the back. Tremor can be acquired with variety of measuring approaches, including sensors for a measurement of the angle of the joint deflection in tremor-type joint movements [5] and with electromyography [6].

We did not address the recognition of the activities of daily living, such as walking, sitting, lying, etc., and the detection of falling, which have already been successfully addressed [9], [15]. We are focused on solving a more challenging task, i.e. the recognition of gait-related health problems.

**Recognition of health problems.** In related work [10], [11], physicians and physical therapists usually diagnose health problems which manifest in gait just by visually observing the user's gait. If they cannot decide for a certain diagnosis easily, they can use a semiautomatic approach in which they capture the movement using motion capture system and then they analyze it by observing the time series of the movement (such as time series

---

<sup>1</sup> Jožef Stefan Institute, Dept. of Intelligent Systems, Slovenia  
<sup>2</sup> Špica International d.o.o., Slovenia,  
email: {bogdan.pogorelc, matjaz.gams}@ijs.si

of angles of legs, etc.).

They use the same procedure for pre-rehabilitation planning (e.g. physical therapy) and after-rehabilitation evaluation to calculate the difference in movement between those states. They also use some special medical tests, e.g., a test for assessment of balance. However, none of those approaches can provide constant real-time observation of the elderly at home, for automatic recognition of changes in movement (gait), indicating a health problem or progress in home rehabilitation.

An example of the semiautomatic approach is also [23], where a system for long-term monitoring of the gait in Parkinson's disease is presented. The characteristics of every stride taken were acquired using a lightweight ankle-mounted sensor array that was sending the data into a small pocket PC. In comparison to our approach, the described approach was meant for the monitoring of progress of Parkinson's disease for the known Parkinson's disease patient and was not used for the early automatic recognition of various health problems.

The work [5] presents sensors for measurement of the angle of joint deflection in tremor-type joint movements, which can also be used to assess Parkinson's disease. However, the sensor systems are too big and would prevent users from doing the activities of daily living if the systems were to be worn all day. Similar to the system described previously it has a drawback in comparison to our approach, because the system cannot automatically recognize Parkinson's disease or any other health problem.

Using similar motion-capture system as the one in our approach, the automatic distinguishing between health problems such as hemiplegia and diplegia is presented in [14]. A classification accuracy of 92.5 % was reported. This was achieved with Self-Organizing Maps, whose features were wavelet-transformed gait characteristics such as walking speed and stride length.

To present the motion recognition field more generally, also fall detection and activity recognition will be mentioned here. The review is broken down by the choice of motion capture system.

**Fall detection and activity recognition using inertial sensors.** Fall detection with accelerometers and gyro sensors [20] is quite common, particularly when using simple threshold algorithms [13], achieving classification accuracies close to 100 %. With a more advanced approach using the One-Class SVM machine learning algorithm whose features were accelerations, changes in acceleration etc., an accuracy of 96.7 % was reported [22]. It should be noticed that these results were obtained in rather academic circumstances, because accuracies like the one reported are not feasible in real life.

Accelerometers can also be used for activity recognition. Five tri-axial accelerometers distinguished 30 physical activities of various intensities with an accuracy of 94.9 % with person-dependent training and 56.3 % with person-independent training [19]. This was achieved with C4.5 decision trees using various time- and frequency-domain features.

**Fall detection and activity recognition from video.** Fall detection and activity recognition from video are also quite common. In an example of fall detection [20], objects in the video were first identified and then rules were applied to the aspect ratios of their bounding boxes, their horizontal and vertical gradients and angles, achieving an average accuracy of 83 %. In an example of activity recognition [23], ten states related to the activities of daily living were distinguished with an average accuracy of 74.1 %. This was achieved by first reconstructing the human body in the video

and then matching its state against models defined using a specialized language. Video detection has drawback in comparison to other methods, i.e. the often unacceptable identity obtrusion.

**Fall detection and activity recognition from video and tags.** This work, using similar motion capture devices as ours [18], used 43 body tags sampled with 30 Hz to distinguish between seven activities related to military operations, reporting an accuracy of 76.9 %. This was achieved with the SVM machine learning algorithm whose features were the tag coordinates belonging to two postures separated by 1/3 of a second, reduced in number to 20 using Principal Component Analysis.

Another paper on activity recognition [17] used 41 body tags sampled with 120 Hz to distinguish between 21 dance gestures, reporting an accuracy of 99.3 %. The gestures were represented with Gaussian mixture models of joint angles. The high accuracy can in part be attributed to the high quality of the input data, the strictly defined gestures and the fact that testing was always done on the same dancer as the training. This research is based on a large number of tags, not appropriate for elderly.

**Investigation of the optimal placement of motion capture devices.** An important part of the research presented in this paper is the study of the impact of the placement of tags on the user's body and the amount of noise in tag coordinates on the classification accuracy. The closest work in this respect that we are aware of investigated the placement of accelerometers for fall detection [8], [13]. Their finding was that the head provides optimal accuracy, but is impractical, the wrist is not appropriate, and the waist is a good option.

### 3 METHODS AND MATERIALS

**Targeted Health Problems.** For the development of our health problem recognition system we have focused on four health problems and normal walking as a reference in accordance with the suggestions received from the collaborating medical expert. The specific health problems for recognition were suggested based on the incidence in the elderly aged 65+, the medical significance and the feasibility of their recognition from the observed subjects' movements. The following four health problems were chosen as the most appropriate [10], [11]: Parkinson's disease, hemiplegia, pain in the leg and pain in the back.

**Construction of the Features for Machine Learning.** A physician usually diagnoses target health problems while observing a patient's gait (i.e. posture and the walking pattern). Since the gaits of patients with the observed four health problems and normal gait look similar to each other, a physician needs to pay attention to many details that need to be transformed into computable quantities. In practice, the observed five health states can be detected by the distinctive walking patterns [10], [11]. For the task of the automatic health-problem recognition we proposed and tested 13 features that are based on the tag locations for 12 tags, placed on the shoulders, elbows, wrists, hips, knees and ankles of the elderly.

The proposed features are listed as follows:

- Absolute difference between i) average distance between right elbow and right hip and ii) average distance between right wrist and left hip.
- Average angle of the right elbow.
- Quotient between maximal angle of the left knee and maximal angle of the right knee.

- Difference between maximal and minimal angle of the right knee.
- Difference between maximal and minimal height of the left shoulder.
- Difference between maximal and minimal height of the right shoulder.
- Quotient between i) difference between maximal and minimal height of left ankle and ii) maximal and minimal height of right ankle.
- Absolute difference between i) difference between maximal and minimal speed (magnitude of velocity) of the left ankle and ii) difference between maximal and minimal speed of the right ankle.
- Absolute difference between i) average distance between right shoulder and right elbow and ii) average distance between left shoulder and right wrist.
- Average speed (magnitude of velocity) of the right wrist.
- Frequency of angle of the right elbow passing average angle of the right elbow.
- Average angle between i) vector between right shoulder and right hip and ii) vector between right shoulder and right wrist.
- Difference between average height of the right shoulder and average height of the left shoulder.

The features for the identification of the chosen four health problems were designed with the help of a medical expert. They are afterwards used for modeling using the machine learning methods.

#### Modeling Target Health Problems using Machine Learning.

To construct a predictive model which can be subsequently used to automatically recognize health problems in the subjects yet to be observed (our first objective from the Introduction), we employed supervised learning methods from the field of machine learning (sub-field of the artificial intelligence field). In supervised learning, a training data set of already labeled subjects (i.e. classified into one of the target five classes) is used to construct a model, which is later used to predict the class of the subjects for which we wish to detect the health problem.

Our task was therefore to classify the recordings of walking into five classes: four with selected health problems (classes *hemiplegia*, *parkinson*, *pain-leg*, *pain-back*) and one without it (*normal*).

Data for the evaluation of the proposed approach was collected by recording the gaits of test subjects with particular walking patterns, of which each subject was recorded 4-5 times. The final data set of 141 recordings consisted of:

- 25 recordings of normal walking,
- 45 recordings of walking with hemiplegia,
- 25 recordings of walking with Parkinson's disease,
- 25 recordings of walking with a limp due to a pain in the leg,
- 21 recordings of walking with a limp due to a pain in the back.

The recordings consisted of the position coordinates for the 12 tags worn on the body, sampled with 60 Hz. The tag coordinates were acquired with Smart IR motion capture system with 0.5 mm standard deviation of noise.

For each subject, the locations of the sensor tags were recorded in a session which lasted 5-8 seconds from which a vector of 13 proposed features was computed. These learning examples were labeled with the type of the representing health problem, yielding the final data on which the classifier was trained.

## 4 EXPERIMENTS AND RESULTS

In our experimental work we have focused on analyzing the classification accuracies of model, built using machine learning methods. The experimental classification accuracies were obtained using stratified 10-fold cross validation. We compared decision tree (DT) and support vector machines machine learning (SVM) algorithms [21].

The 10-fold cross-validation resulted in classification accuracy of 90.1 % and 97.9 % for decision tree and support vector machines classifier, respectively.

**Table 1.** Confusion matrices of matrices of decision tree (left) and support vector machines classifier (right).

|            |   | classified as |    |    |    |    |
|------------|---|---------------|----|----|----|----|
|            |   | H             | L  | N  | P  | B  |
| true class | H | 40            | 0  | 1  | 4  | 0  |
|            | L | 2             | 23 | 0  | 0  | 0  |
|            | N | 2             | 0  | 23 | 0  | 0  |
|            | P | 5             | 0  | 0  | 20 | 0  |
|            | B | 0             | 0  | 0  | 0  | 21 |

|            |   | classified as |    |    |    |    |
|------------|---|---------------|----|----|----|----|
|            |   | H             | L  | N  | P  | B  |
| true class | H | 45            | 0  | 0  | 0  | 0  |
|            | L | 1             | 24 | 0  | 0  | 0  |
|            | N | 0             | 0  | 25 | 0  | 0  |
|            | P | 2             | 0  | 0  | 23 | 0  |
|            | B | 0             | 0  | 0  | 0  | 21 |

Table 1 shows the confusion matrices, where H=hemiplegia, L=pain in the leg, N=normal (healthy) walking, P=Parkinson's disease and B=Pain in the back. Numbers denote quantity of the classified examples. Confusion matrices are used for visualization, how many examples of a certain true class (in rows) are classified in one of possible five classes (in columns). For the real-world cases, they are for three purposes:

- We can observe how many false positives (false alarms) can be expected using these classifiers. When in real world use the system would report false alarm, e.g., normal walking is classified as some health problem, ambulance could drive to pick up the elderly which would cause unnecessary costs. In case of decision tree, normal walking was in 2 of 25 examples erroneously classified as hemiplegia.
- We can see how many false negatives can be expected using these classifiers. False negatives could mean potentially risky situation for the elderly, as his/her health problem would not be recognized automatically. In case of decision tree, hemiplegia was in 1 of 45 examples erroneously classified as normal walking.
- We can identify between which health states (classes) the errors (misclassifications) occurs. Consequently, we can add additional features to help distinguish between those particular classes. The misclassifications happened very rarely.

The results show that in the proposed approach false positives/negatives are very rare, i.e., they would not cause much unnecessary ambulance costs. Since the method accurately classified most true health problems, it represents high confidence and safety for the potential use in elderly care.

### 4.1 Variation of noise

To test the robustness of the approach, we added Gaussian noise with varying standard deviation (and zero mean) to the raw coordinates. The standard deviation of noise was varied from 0 mm to 50 mm in steps of 5 mm. As a preprocessing step, a Kalman filter was used to smooth the potentially unrealistic difference between the positions of two consecutive time samples, caused by the addition of Gaussian noise to the captured positions [16].

The cells of Table 2 show the classification accuracy and the significance level of the t-test for varying the amount of standard

deviation of noise in the range 0 – 50 mm. The two-tailed paired t-test was performed between the cross-validation folds of the initial setting (0 mm noise) and the corresponding amount of noise denoted by the table column. Dark shading denotes the experiments in which the classification accuracy significantly changed ( $\alpha < 0.05$ ). The results show that varying the amount of noise has a different impact on the used classifiers. The performance of the decision tree does not degrade significantly with an increase in the noise; however, its classification accuracy is lower than of support vector machines in the initial setting. On the other hand, the more accurate support vector machines in the initial setting, showed significant decrease in classification accuracy at 30, 45 and 50 mm of noise.

**Table 2.** Classification accuracies (CA in [%]) of the decision tree (second row) and the support vector machines classifier (third row) built on data with added noise.

| Noise [mm] | 0    | 5                        | 10                       | 15                       | 20                       | 25                       | 30                       | 35                       | 40                       | 45                       | 50                       |
|------------|------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| CA (DT)    | 90.1 | 92.9<br>$\alpha = 0.380$ | 94.3<br>$\alpha = 0.168$ | 93.6<br>$\alpha = 0.245$ | 92.2<br>$\alpha = 0.510$ | 92.2<br>$\alpha = 0.542$ | 87.9<br>$\alpha = 0.403$ | 90.1<br>$\alpha = 0.976$ | 87.2<br>$\alpha = 0.524$ | 87.2<br>$\alpha = 0.453$ | 84.4<br>$\alpha = 0.182$ |
| CA (SVM)   | 97.9 | 98.6<br>$\alpha = 0.558$ | 98.6<br>$\alpha = 0.558$ | 97.9<br>$\alpha = 0.999$ | 98.6<br>$\alpha = 0.343$ | 98.6<br>$\alpha = 0.591$ | 94.3<br>$\alpha = 0.049$ | 95.0<br>$\alpha = 0.228$ | 97.9<br>$\alpha = 0.999$ | 90.8<br>$\alpha = 0.008$ | 90.8<br>$\alpha = 0.004$ |

Even with the highest amount of noise added, the classification accuracies of both classifiers were much higher than the accuracy of the majority class classifier with classification accuracy of 31.9 %. The majority class classifier defines the lowest acceptable classification accuracy, which can be trivially achieved by classifying all examples into the majority class, i.e., class with the largest number of training examples. Therefore, the models are robust to the addition of the noise.

## 4.2 Reduction of the number of tags

Since wearing the full complement of 12 tags may be annoying to the user, we investigated ways to reduce the number of tags. We started with all 12 tags and removed them in the order that retained the largest number of the features when decreasing the number of tags by one. Consequently, the “best” tag placement for each number of the 12 tags was obtained.

The classification accuracies for the best tag placements for various numbers of tags and without addition of noise are shown in Table 3. The table cells contain the classification accuracy and the significance level of the two-tailed paired t-test which was performed between the initial setting (12 tags) and the corresponding number of tags. Dark shading denotes the experiments in which the classification accuracy decreased significantly ( $\alpha < 0.05$ ).

**Table 3.** The classification accuracies achieved with the best tag placements for each number of tags for the decision tree (second row) and the support vector machines classifier (third row).

| Nr. of tags | 12   | 11                       | 10                       | 9                        | 8                        | 7                        | 6                        | 5                        | 4                        | 3                        | 2                        | 1                        |
|-------------|------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| CA (DT)     | 90.1 | 91.5<br>$\alpha = 0.168$ | 90.1<br>$\alpha = 0.999$ | 91.5<br>$\alpha = 0.168$ | 90.1<br>$\alpha = 0.999$ | 92.9<br>$\alpha = 0.363$ | 90.1<br>$\alpha = 0.974$ | 92.2<br>$\alpha = 0.443$ | 94.3<br>$\alpha = 0.114$ | 87.2<br>$\alpha = 0.486$ | 71.6<br>$\alpha = 0.001$ | 43.3<br>$\alpha = 0.001$ |
| CA (SVM)    | 97.9 | 99.3<br>$\alpha = 0.168$ | 98.6<br>$\alpha = 0.594$ | 98.6<br>$\alpha = 0.343$ | 98.6<br>$\alpha = 0.591$ | 95.0<br>$\alpha = 0.228$ | 99.3<br>$\alpha = 0.168$ | 92.9<br>$\alpha = 0.097$ | 87.2<br>$\alpha = 0.004$ | 75.2<br>$\alpha = 0.001$ | 59.6<br>$\alpha = 0.001$ | 46.1<br>$\alpha = 0.001$ |

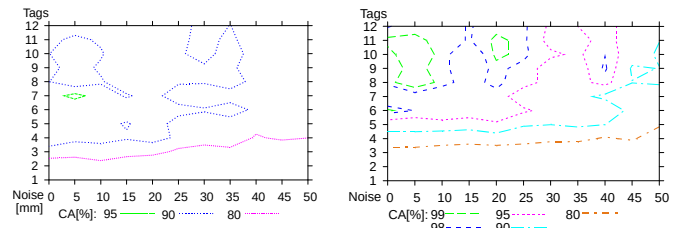
From Table 3 we can observe that decreasing the number of tags

results in the decrease of classifier’s performance and that the smallest number of tags for which the performance of classifiers is insignificantly different compared to the initial setting is three for decision tree and five for support vector machines.

For recognizing target health states, the positions of tags from the most to the least appropriate are: shoulder, wrist, elbow, hip, ankle and knee.

## 4.3 Dependence of the classification accuracy on the tag placement and noise level

Besides experimenting with tag placement, we also varied the standard deviation of noise from 0 to 50 mm. Figure 1 shows the dependence of the classification accuracy (CA) on the tag placement and the noise level. We can observe a variation of the noise in the standard deviation from 0 to 50 mm on horizontal axis and the best tag placement for each number of tags from 12 to 1 tag on the vertical axis. Each curve of different color and shape (e.g. dotted, dashed) connects points of the particular classification accuracy.



**Figure 1.** Classification accuracy with respect to the number of tags and the noise level for the recognition of the target health problems using the decision tree (left) and support vector machines classifier (right).

Figure 1 (left) illustrates the classification accuracies for the decision-tree classifier; the highest classification accuracy over 95 % is reached only for 7 tags and 5 mm noise. To surpass the classification accuracy of 90 % we need at least 4 tags in the noise range 0-20 mm and at least 6 tags in the noise range 25-35 mm, except for very small “islands”, where the classification accuracy is insignificantly lower. To exceed an accuracy of 80 % at least 3 tags are required for 0-20 mm and 4 tags for 25-50 mm noise.

Figure 1 (right) shows the classification accuracies for the support vector machines classifier. With 8-10 tags and at most 0-5 mm noise or with 10-11 tags with around 20 mm noise we can achieve a classification accuracy of over 99 %. We need at least 8 tags and at most 0-10 mm of noise to surpass the boundary of 98 %, 6 tags with 0-25 mm noise for 95 % and 5 tags with 0-40 mm noise for 90 %. Four tags are enough to overcome the boundary of 80 % for all the noise amounts except for the 50 mm noise, where 5 tags are needed.

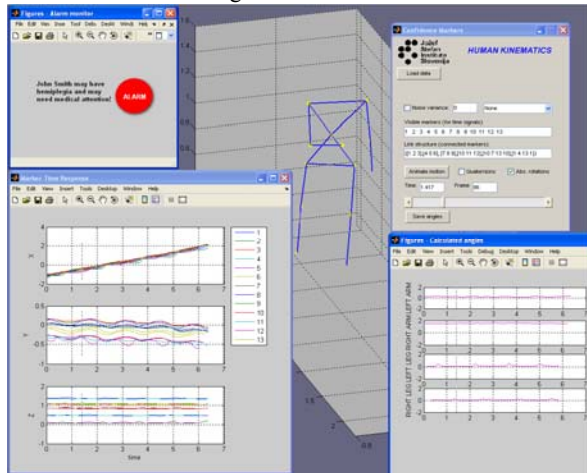
Although the classification accuracies of the support vector machines classifier are overall better than of decision tree, decision tree has important advantage in interpretability of the predictions. In case the system recognizes some health state it can explain the reasons for classification to the physician.

## 5 EXPLANATION OF THE DETECTION

Because it is very important for the physician to obtain an explanation for the interpreted health state, we have developed a control-panel prototype, which is intended to be used by

physicians (Figure 2).

In the middle of the prototype screen, there is visualization of the kinematic model of the elderly moving through the room. In the upper-right-hand corner there are controls for saving and loading the captured movement. Underneath there are the time series of the calculated angles.



**Figure 2.** Panel prototype for the explanation of the interpreted health state

In the lower-left-hand corner the time series of the x, y and z positions are shown for all 13 tags (the 13th tag is calculated from the positions of the nearest tags). When the health problem is recognized, the red alarm button appears in the left upper-left-hand corner with the necessary description of the recognized case for the relatives and/or for the medical center.

## 6 CONCLUSION

The results show that in the initial setting decision tree achieved average classification accuracy of 90.1 % and support vector machines achieved 97.9 %. There were a few false positives/negatives, thus not much unnecessary ambulance costs are expected in practice. The results of varying the amount of noise and the results of varying the number of tags indicate that both algorithms are robust, the decision tree being a bit more robust. The results of investigation of the dependence of the classification accuracy on the tag placement and noise level show that the support vector machines was more accurate than decision tree. Because it is very important for the physician to obtain an explanation of the interpreted health state, we paid a lot of attention to it in the developed control-panel prototype.

For future work, we intend to adapt the system for automatic evaluation of rehabilitation process (e.g. after stroke) at home.

## ACKNOWLEDGEMENTS

This work is partially financed by the European Union, the European Social Fund. The authors thank Martin Tomšič, Bojan Nemec and Leon Žlajpah for their help with data acquisition, Anton Gradišek for his medical expertise and Zoran Bosnić, Mitja Luštrek and Rok Piltaver for helpful discussions.

## REFERENCES

- [1] S. Toyne, Ageing: Europe's growing problem. BBC News. <http://news.bbc.co.uk/2/hi/business/2248531.stm>, 2009-01-19.
- [2] D. Strle, V. Kempe, "MEMS-based inertial systems", *Informacije MIDEM* 37(2007)4, pp. 199-209.
- [3] D. Jurman, M. Jankovec, R. Kamnik, M. Topič, "Inertial and magnetic sensors: The calibration aspect", *Informacije MIDEM* 37(2007)2, pp. 67-72.
- [4] F. Dimic, B. Mušič, R. Osredkar, "An example of an integrated GPS and DR positioning system designed for archeological prospecting", *Informacije MIDEM* 38(2008)2, pp. 144-148.
- [5] S. Ribarič, J. Rozman, "Sensors for measurement of tremor type joint movements", *Informacije MIDEM* 37(2007)2, pp. 98-104.
- [6] J. Trontelj, J. Trontelj and L. Trontelj, "Safety Margin at mammalian neuromuscular junction – an example of the significance of fine time measurements in neurobiology", *Informacije MIDEM* 38(2008)3, pp. 155-160.
- [7] Bourke, A.K., and Lyons, G.M. A threshold-based fall-detection algorithm using a bi-axial gyroscope sensor. *Medical Engineering & Physics* 30, 1 (2006), 84–90.
- [8] Bourke, A.K., Scanail, C.N., Culhane, K.M., O'Brien, J.V., and Lyons, G.M. An optimum accelerometer configuration and simple algorithm for accurately detecting falls. In *Proc. BioMed 2006* (2006), 156–160.
- [9] Confidence: Ubiquitous Care System to Support Independent Living. <http://www.confidence-eu.org>.
- [10] Craik R., and Oatis C. *Gait Analysis: Theory and Application*. Mosby-Year Book (1995).
- [11] Perry J. *Gait Analysis: Normal and Pathological Function*. McGraw-Hill, Inc., 1992.
- [12] eMotion. Smart motion capture system. <http://www.emotion3d.com/smart/smart.html>.
- [13] Kangas, M., Konttila, A., Lindgren, P., Winblad, P., and Jamsa, T. Comparison of low-complexity fall detection algorithms for body attached accelerometers. *Gait & Posture* 28, 2 (2008), 285–291.
- [14] Lakany, H. Extracting a diagnostic gait signature. *Pattern recognition* 41 (2008), 1627–1637.
- [15] Luštrek M., Kaluža B., Fall Detection and Activity Recognition with Machine Learning. *Informatica (Slovenia)* 33(2): 197-204 (2009).
- [16] Maybeck, P.S. Stochastic models, estimation, and control. *Mathematics in Science and Engineering* 141 (1979).
- [17] Qian, G., Guo, F., Ingalls, T., Olson, L., James, J., and Rikakis, T. A gesture-driven multimodal interactive dance system. In *Proc. ICME '04* (2004), 1579–1582.
- [18] Sukthankar, G., and Sycara, K. A cost minimization approach to human behavior recognition. In *Proc. AAMAS 2005* (2005), 1067–1074.
- [19] Tapia, E.M., Intille, S.S., Haskell, W., Larson, K., Wright, J., King, A., and Friedman, R. Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor. In *Proc. ISWC 2007* (2007), 37–40.
- [20] Vishwakarma, V., Mandal, C., and Sura, S. Automatic detection of human fall in video. *Lecture Notes in Computer Science* 4815 (2007), 616–623.
- [21] Witten, I.H., and Frank, E. *Data Mining: Practical Machine Learning Tools and Techniques* (2nd edition). Morgan Kaufmann (2005).
- [22] Zhang, T., Wang, J., Liu, P., and Hou, J. Fall detection by wearable sensor and One-Class SVM algorithm. *Lecture Notes in Control and Information Science* 345 (2006), 858–863.
- [23] Zouba, N., Boulay, B., Bremond, F., and Thonnat, M. Monitoring activities of daily living (ADLs) of elderly based on 3D key human postures. In *Proc. ICW 2008* (2008), 37–50.
- [24] Moore ST, et al., Long-term monitoring of gait in Parkinson's disease, *Gait Posture* (2006), doi:10.1016/j.gaitpost.2006.09.011