THE 18TH EUROPEAN CONFERENCE ON MACHINE LEARNING
AND
THE 11TH EUROPEAN CONFERENCE ON PRINCIPLES AND PRACTICE
OF KNOWLEDGE DISCOVERY IN DATABASES

PROCEEDINGS OF THE INTERNATIONAL WORKSHOP ON KNOWLEDGE DISCOVERY FROM UBIQUITOUS DATA STREAMS

IWKDUDS 2007

September 17, 2007

Warsaw, Poland

Editors:
João Gama
LIAAD - INESC Porto L.A. & University of Porto, Portugal
Mohamed Medhat Gaber
Tasmanian ICT Centre, CSIRO ICT Centre, Australia
Jesús S. Aguilar-Ruiz
School of Engineering, Pablo de Olavide University, Seville, Spain

Preface

We are glad to have this year's international workshop on knowledge discovery from ubiquitous data streams at ECML/PKDD 2007. We have a strong workshop program with 12 papers, an invited talk and a tutorial.

Ting, Theodorou and Schaal have introduced a modified Kalman Filter that can perform real-time outlier detection. Spinosa, Carvalho, and Gama have extended OLINDDA (OnLIne Novelty and Drift Detection Algorithm) to multiple-class classification problems. Lei, Tang, Iglesias, Mukherjee, and Mohanty have presented a simlilarity-driven clustering approach to address the scalalbility probelms in large datasets with an application to Gravitational-Wave Astronomy Data. Yoshida and Hruschka Jr. have decribed and evaluated experimentally a Quasi-Incremental Bayesian Classifier that could be used in dynamic systems like sensor networks. Küçük, Inan, Boyrazoglu, Buhan, Salor, Çadirci, and Ermis have presented a data stream architecture for electrical power quality (PQStream). Phung, Gaber and Roehm have extended their ERA-Cluster clusiting algorithm to work in a distributed mode in wireless sensor networks. Karnstedt, Franke and Gaber have introduced and described mathematically a model for quality guaranteed resource-aware stream mining. Landwehr, Gutmann, Thon, Philipose, and Raedt have described a ubiquitous computing application to recognize human activities from sensory data. Last and Saveliev have enhanced the Information Network (IN) classification algorithm by preserving the model qulaity while reducing the coputational cost. Rodrigues and Gama have extended their clustering technique Online Divisive-Agglomerative Clustering (ODAC) using semi-fuzzy approach. Haghighi, Gaber, Krishnaswamy, Zaslavsky, and Loke have introduced an architecture for context-aware adaptive data stream mining. Finally, An and Park have introduced an efficient secure XML query processing method.

We hope that this proceedings will form an important and valuable addition to your library. Finally, we thank all the authors for their significant contributions to the workshop.

August 2007

João Gama
Mohamed Medhat Gaber
Jesús S. Aguilar-Ruiz

Workshop Organization

Workshop Chairs

João Gama (LIAAD - INESC Porto L.A. & University of Porto, Portugal)
Mohamed Medhat Gaber (Tasmanian ICT Centre, CSIRO ICT Centre, Australia)
Jesús S. Aguilar-Ruiz (School of Engineering, Pablo de Olavide University, Spain)

Publicity Chair

Pedro Pereira Rodrigues (LIAAD - INESC Porto L.A. & University of Porto, Portugal)

ECML/PKDD Workshop Chair

Marzena Kryszkiewicz (Warsaw University of Technology)

Workshop Program Committee

Andreas Hotho
André Carvalho
Antoine Cornuejols
Bernhard Seeger
Elaine Sousa
Eduardo Spinosa
Francisco Ferrer-Troyano
Auroop Ganguly
Geoff Holmes
Georges Hebrail
Hillol Kargupta
João Gama
Jesús Aguilar-Ruiz
Josep Roure
Mark Hall
Mark Last
Miroslav Kubat
Mohamed Medhat Gaber
Olufemi Omitaomu
Pedro Pereira Rodrigues
Philip S. Yu
Ralf Klinkenberg
Rasmus Pedersen
Ricard Gavalda
Sean Wang
Takashi Washio
Raju Vatsavai
Ying Yang

Table of Contents

Learning an Outlier-Robust Kalman Filter 1
Jo-Anne Ting, Evangelos Theodorou and Stefan Schaa

Learning novel concepts: beyond one-class classification with OLINDDA 13
Eduardo J. Spinosa, André Ponce de Leon F. de Carvalho, and João Gama

S-means: Similarity Driven Clustering and Its application in Gravitational-Wave Astronomy Data Mining 25
Hansheng Lei, Lappoon R. Tang, Juan R. Iglesias, Soma Mukherjee and Soumya Mohanty

Quasi-Incremental Bayesian Classifier 37
Murilo Lacerda Yoshida and Estevam R. Hruschka Jr.

PQStream: A Data Stream Architecture for Electrical Power Quality 47
Dilek Küçük, Tolga İnan, Burak Boyrazoğlu, Serkan Buhan, Özgül Salor, Işık Çadırcı and Muammer Ermiş

Resource-aware Distributed Online Data Mining for Wireless Sensor Networks . . 59
Nhan Duc Phung, Mohamed Medhat Gaber and Uwe Roehm

A Model for Quality Guaranteed Resource-Aware Stream Mining 72
Marcel Karnstedt, Conny Franke and Mohamed Medhat Gaber

Relational Transformation-based Tagging for Human Activity Recognition 83
Niels Landwehr, Bernd Gutmann, Ingo Thon, Matthai Philipose and Luc De Raedt

Enhanced Anytime Algorithm for Induction of Oblivious Decision Trees 95
Mark Last and Albina Saveliev

A Semi-Fuzzy Approach for Online Divisive-Agglomerative Clustering 107
Pedro Pereira Rodrigues and João Gama

An Architecture for Context-Aware Adaptive Data Stream Mining 117
Pari Delir Haghighi, Mohamed Medhat Gaber, Shonali Krishnaswamy, Arkady Zaslavsky and Seng Loke

Efficient Secure Query Processing in XML Data Stream 129
Dong-Chan An and Seog Park

Author Index 139

Learning an Outlier-Robust Kalman Filter

Jo-Anne Ting[1], Evangelos Theodorou[1] and Stefan Schaal[1,2]

[1] University of Southern California, Los Angeles, CA 90089
[2] ATR Computational Neuroscience Laboratories, Kyoto, Japan
{joanneti, etheodor, sschaal}@usc.edu

Abstract. In this paper, we introduce a modified Kalman filter that performs robust, real-time outlier detection, without the need for manual parameter tuning by the user. Systems that rely on high quality sensory data (for instance, robotic systems) can be sensitive to data containing outliers. The standard Kalman filter is not robust to outliers, and other variations of the Kalman filter have been proposed to overcome this issue. However, these methods may require manual parameter tuning, use of heuristics or complicated parameter estimation procedures. Our Kalman filter uses a weighted least squares-like approach by introducing weights for each data sample. A data sample with a smaller weight has a weaker contribution when estimating the current time step's state. Using an incremental variational Expectation-Maximization framework, we learn the weights and system dynamics. We evaluate our Kalman filter algorithm on data from a robotic dog.

1 Introduction

Systems that rely on high quality sensory data are often sensitive to data containing outliers. While data from sensors such as potentiometers and optical encoders are easily interpretable in their noise characteristics, other sensors such as visual systems, GPS devices and sonar sensors often provide measurements populated with outliers. As a result, robust, reliable detection and removal of outliers is essential in order to process these kinds of data. For example, in the application domain of robotics, legged locomotion is vulnerable to sensory data of poor quality, since one undetected outlier can disturb the balance controller to the point that the robot loses stability.

An outlier is generally defined as an observation that "lies outside some overall pattern of distribution" [1]. Outliers may originate from sensor noise (producing values that fall outside a valid range), from temporary sensor failures, or from unanticipated disturbances in the environment (e.g., a brief change of lighting conditions for a visual sensor). Note that some prior knowledge about the observed data's properties must be known. Otherwise, it is impossible to discern if a data sample that lies some distance away from the data cloud is truly an outlier or simply part of the data's structure.

For real-time applications, storing data samples may not be a viable option due to the high frequency of sensory data and insufficient memory resources. In

this scenario, sensor data are made available one at a time and must be discarded once they have been observed. Hence, techniques that require access to the entire set of data samples, such as the Kalman smoother [2] are not applicable. Instead, the Kalman filter [3] is a more suitable method, since it assumes that only data samples up to the current time step have been observed.

The Kalman filter is a widely used tool for estimating the state of a dynamic system, given noisy measurement data. It is the optimal *linear* estimator for linear Gaussian systems, giving the minimum mean squared error [4]. Using state estimates, the filter can also estimate what the corresponding (output) data are. However, the performance of the Kalman filter degrades when the observed data contains outliers. To address this, previous work has tried to make the Kalman filter more robust to outliers by addressing the sensitivity of the squared error criterion to outliers [5, 6]. One class of approaches considers non-Gaussian distributions for random variables (e.g., [7–9]), since multivariate Gaussian distributions are known to be susceptible to outliers. For example, [10] uses multivariate Student-t distributions. However, the resulting estimation of parameters may be quite complicated for systems with transient disturbances.

Alternatively, it is possible to model the observation and state noise as non-Gaussian, heavy-tailed distributions to account for non-Gaussian noise and outliers, e.g., [11, 12]. Unfortunately, these filters are typically more difficult to implement and may no longer provide the conditional mean of the state vector. Other approaches use resampling techniques (e.g., [13]) or numerical integration (e.g., [14]), but these may require heavy computation not suitable for real-time applications.

Yet another class of methods uses a weighted least squares approach, as done in robust least squares [15], where the measurement residual error is assigned some statistical property. Some of these algorithms fall under the first category of approaches as well, assuming non-Gaussian distributions for variables. Each data sample is assigned a weight that indicates its contribution to the hidden state estimate at each time step. This technique has been used to produce a Kalman filter that is more robust to outliers (e.g., [16, 17]). However, these methods usually model the weights as some heuristic function of the data (e.g., the Huber function [15]) and often require manual tuning of threshold parameters for optimal performance. Using incorrect or inaccurate estimates for the weights may lead to deteriorated performance, so special attention and care is necessary when using these techniques.

In this paper, we are interested in making the Kalman filter more robust to the outliers in the observations (i.e. the filter should identify and eliminate possible outliers as it tracks observed data). Identifying outliers in the state is an entirely different problem, left for another paper. We introduce a modified Kalman filter that can detect outliers in the observed data without the need for manual parameter tuning or use of heuristic methods. For ease of analytical computation, we assume Gaussian distributions for variables and states. We illustrate the performance of this robust Kalman filter on robotic data, comparing

it with other robust Kalman filter methods and demonstrating its effectiveness at detecting outliers in the observations.

2 Outlier Detection in the Kalman Filter

Let us assume we have data observed over N time steps, $\{\mathbf{z}_k\}_{k=1}^N$, and the corresponding hidden states as $\{\boldsymbol{\theta}_k\}_{k=1}^N$ (where $\boldsymbol{\theta}_k \in \Re^{d_2 \times 1}, \mathbf{z}_k \in \Re^{d_1 \times 1}$). Assuming a time-invariant system, the Kalman filter system equations are:

$$\begin{aligned} \mathbf{z}_k &= \mathbf{C}\boldsymbol{\theta}_k + \mathbf{v}_k \\ \boldsymbol{\theta}_k &= \mathbf{A}\boldsymbol{\theta}_{k-1} + \mathbf{s}_k \end{aligned} \tag{1}$$

where $\mathbf{C} \in \Re^{d_1 \times d_2}$ is the observation matrix, $\mathbf{A} \in \Re^{d_2 \times d_2}$ is the state transition matrix, $\mathbf{v}_k \in \Re^{d_1 \times 1}$ is the observation noise at time step k, and $\mathbf{s}_k \in \Re^{d_2 \times 1}$ is the state noise at time step k. We assume $\mathbf{v}_k$ and $\mathbf{s}_k$ to be uncorrelated additive mean-zero Gaussian noise: $\mathbf{v}_k \sim \text{Normal}\,(0, \mathbf{R})$, $\mathbf{s}_k \sim \text{Normal}\,(0, \mathbf{Q})$, where $\mathbf{R} \in \Re^{d_1 \times d_1}$ is a diagonal matrix with $\mathbf{r} \in \Re^{d_1 \times 1}$ on its diagonal, and $\mathbf{Q} \in \Re^{d_2 \times d_2}$ is a diagonal matrix with $\mathbf{q} \in \Re^{d_2 \times 1}$ on its diagonal. $\mathbf{R}$ and $\mathbf{Q}$ are covariance matrices for the observation and state noise, respectively. Fig. 1(a) shows the graphical model for the standard Kalman filter. Its corresponding filter propagation and update equations are, for $k = 1, .., N$:

Propagation:

$$\boldsymbol{\theta}'_k = \mathbf{A}\left\langle\boldsymbol{\theta}_{k-1}\right\rangle \tag{2}$$

$$\boldsymbol{\Sigma}'_k = \mathbf{A}\boldsymbol{\Sigma}_{k-1}\mathbf{A}^T + \mathbf{Q} \tag{3}$$

Update:

$$\mathbf{S}'_k = \left(\mathbf{C}\boldsymbol{\Sigma}'_k\mathbf{C}^T + \mathbf{R}\right)^{-1} \tag{4}$$

$$K'_k = \boldsymbol{\Sigma}'_k\mathbf{C}^T\mathbf{S}'_k \tag{5}$$

$$\left\langle\boldsymbol{\theta}_k\right\rangle = \boldsymbol{\theta}'_k + K'_k\left(\mathbf{z}_k - \mathbf{C}\boldsymbol{\theta}'_k\right) \tag{6}$$

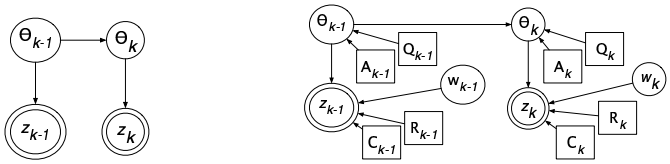
$$\boldsymbol{\Sigma}_k = \left(\mathbf{I} - K'_k\mathbf{C}\right)\boldsymbol{\Sigma}'_k \tag{7}$$

where $\left\langle\boldsymbol{\theta}_k\right\rangle$[3] is the posterior mean vector of the state $\boldsymbol{\theta}_k$, $\boldsymbol{\Sigma}_k$ is the posterior covariance matrix of $\boldsymbol{\theta}_k$, and $\mathbf{S}'_k$ is the covariance matrix of the residual prediction error—all at time step k. In this problem, the system dynamics ($\mathbf{C}$, $\mathbf{A}$, $\mathbf{R}$ and $\mathbf{Q}$) are unknown, and it is possible to use a maximum likelihood framework to estimate these parameter values [18]. Unfortunately, this standard Kalman filter model considers all data samples to be part of the data cloud and is not robust to outliers.

2.1 Robust Kalman Filtering with Bayesian Weights

To overcome this limitation, we introduce a novel Bayesian algorithm that treats the weights associated with each data sample probabilistically. In particular, we

[3] Note that $\langle\rangle$ denotes the expectation operator



(a) Kalman filter (b) Proposed robust Kalman filter

Fig. 1. Graphical Models: circular nodes are random variables, double circles are observed random variables, and square nodes are point estimated parameters.

introduce a scalar weight w_k for each observed data sample $\mathbf{z}_k$ such that the variance of $\mathbf{z}_k$ is weighted with w_k, as done in [19]. [19] considers a weighted least squares regression model and assumes that the weights are known and given. We model the weights to be Gamma distributed random variables, as done previously in [20] for weighted linear regression. Additionally, we learn estimates for the system dynamics at each time step. A Gamma prior distribution is chosen for the weights in order to ensure they remain positive. Fig. 1(b) shows the graphical model of this robust Kalman filter. The prior distributions are:

$$\begin{aligned} \mathbf{z}_k|\boldsymbol{\theta}_k, w_k &\sim \text{Normal}\left(\mathbf{C}\boldsymbol{\theta}_k, \mathbf{R}/w_k\right) \\ \boldsymbol{\theta}_k|\boldsymbol{\theta}_{k-1} &\sim \text{Normal}\left(\mathbf{A}\boldsymbol{\theta}_{k-1}, \mathbf{Q}\right) \\ w_k &\sim \text{Gamma}\left(a_{w_k}, b_{w_k}\right) \end{aligned} \tag{8}$$

We can treat this entire problem as an Expectation-Minimization-like (EM) learning problem [21, 22] and maximize the log likelihood $\log p(\boldsymbol{\theta}_{1:N})$ (known as the "incomplete" log likelihood with the hidden probabilistic variables marginalized out). Due to analytical issues, we only have access to a lower bound of this measure. This lower bound is based on an expected value of the "complete" data likelihood $\langle \log p\left(\boldsymbol{\theta}_{1:N}, \mathbf{z}_{1:N}, \mathbf{w}\right)\rangle$, formulated over all variables of the learning problem. Since we are considering this problem as a real-time one (i.e. data samples arrive sequentially, one at a time), we will have observed only data samples $\mathbf{z}_{1:k}$ at time step k. Consequently, in order to estimate the posterior distributions of the random variables and parameter values at time step k, we should consider the log evidence of only the data samples observed to date, i.e., $\log p\left(\boldsymbol{\theta}_{1:k}, \mathbf{z}_{1:k}, \mathbf{w}_{1:k}\right)$.

The expectation of the complete data likelihood should be taken with respect to the true posterior distribution of all hidden variables $Q\left(\mathbf{w}, \boldsymbol{\theta}\right)$. Since this is an analytically intractable expression, we use a technique from variational calculus to construct a lower bound and make a factorial approximation of the true posterior [22] as follows: $Q\left(\mathbf{w}, \boldsymbol{\theta}\right) = \prod_{i=1}^{N} Q\left(w_i\right) \prod_{i=1}^{N} Q\left(\boldsymbol{\theta}_i|\boldsymbol{\theta}_{i-1}\right) Q(\boldsymbol{\theta}_0)$. This factorization of $\boldsymbol{\theta}$ considers the influence of each $\boldsymbol{\theta}_i$ from within its Markov blanket, conserving the Markov property that Kalman filters, by definition, have. While losing a small amount of accuracy, all resulting posterior distributions over

hidden variables become analytically tractable. This factorial approximation was chosen purposely so that $Q(w_k)$ is independent from $Q(\boldsymbol{\theta}_k)$; performing joint inference of w_k and $\boldsymbol{\theta}_k$ does not make sense in the context of our generative model. The final EM update equations at time step k are:

E-step:

$$\boldsymbol{\Sigma}_k = \left(\langle w_k \rangle \, \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{C}_k + \mathbf{Q}_k^{-1}\right)^{-1} \tag{9}$$

$$\langle \boldsymbol{\theta}_k \rangle = \boldsymbol{\Sigma}_k \left(\mathbf{Q}_k^{-1} \mathbf{A}_k \langle \boldsymbol{\theta}_{k-1} \rangle + \langle w_k \rangle \, \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{z}_k\right) \tag{10}$$

$$\langle w_k \rangle = \frac{a_{w_k,0} + \frac{1}{2}}{b_{w_k,0} + \left\langle (\mathbf{z}_k - \mathbf{C}_k \boldsymbol{\theta}_k)^T \, \mathbf{R}_k^{-1} \, (\mathbf{z}_k - \mathbf{C}_k \boldsymbol{\theta}_k) \right\rangle} \tag{11}$$

M-step:

$$\mathbf{C}_k = \left(\sum_{i=1}^{k} \langle w_i \rangle \, \mathbf{z}_i \, \langle \boldsymbol{\theta}_i \rangle^T\right) \left(\sum_{i=1}^{k} \langle w_i \rangle \left\langle \boldsymbol{\theta}_i \boldsymbol{\theta}_i^T \right\rangle\right)^{-1} \tag{12}$$

$$\mathbf{A}_k = \left(\sum_{i=1}^{k} \langle \boldsymbol{\theta}_i \rangle \, \langle \boldsymbol{\theta}_{i-1} \rangle^T\right) \left(\sum_{i=1}^{k} \left\langle \boldsymbol{\theta}_{i-1} \boldsymbol{\theta}_{i-1}^T \right\rangle\right)^{-1} \tag{13}$$

$$r_{km} = \frac{1}{k} \sum_{i=1}^{k} \langle w_i \rangle \left\langle (\mathbf{z}_{im} - \mathbf{C}_k(m,:) \boldsymbol{\theta}_i)^2 \right\rangle \tag{14}$$

$$q_{kn} = \frac{1}{k} \sum_{i=1}^{k} \left\langle (\boldsymbol{\theta}_{in} - \mathbf{A}_k(n,:) \boldsymbol{\theta}_{i-1})^2 \right\rangle \tag{15}$$

where $m = 1, .., d_1$, $n = 1, .., d_2$; r_{km} is the mth coefficient of the vector $\mathbf{r}_k$; q_{kn} is the nth coefficient of the vector $\mathbf{q}_k$; $\mathbf{C}_k(m,:)$ is the mth row of the matrix $\mathbf{C}_k$; $\mathbf{A}_k(n,:)$ is the nth row of the matrix $\mathbf{A}_k$; and $a_{w_k,0}$ and $b_{w_k,0}$ are prior scale parameters for the weight w_k. (9) to (15) should be computed once for each time step k (e.g., [23] [24]) when the data sample $\mathbf{z}_k$ becomes available.

Since storing sensor data is not possible in real-time applications, (12) to (15)—which require access to all observed data samples up to time step k—need to be re-written using only values observed, calculated or used in the current time step k. We can do this by collecting sufficient statistics in (12) to (15) and rewriting them as:

$$\mathbf{C}_k = \sum_k^{\mathbf{w}\mathbf{z}\boldsymbol{\theta}^T} \left(\sum_k^{\mathbf{w}\boldsymbol{\theta}\boldsymbol{\theta}^T}\right)^{-1} \tag{16}$$

$$\mathbf{A}_k = \sum_k^{\boldsymbol{\theta}\boldsymbol{\theta}'} \left(\sum_k^{\boldsymbol{\theta}'\boldsymbol{\theta}'}\right)^{-1} \tag{17}$$

$$r_{km} = \frac{1}{k} \left[\sum_{km}^{\mathbf{w}zz} - 2\mathbf{C}_k(m,:) \sum_{km}^{\mathbf{w}\boldsymbol{z}\boldsymbol{\theta}} + \text{diag}\left\{\mathbf{C}_k(m,:) \sum_k^{\mathbf{w}\boldsymbol{\theta}\boldsymbol{\theta}^T} \mathbf{C}_k(m,:)^T\right\}\right] \tag{18}$$

$$q_{kn} = \frac{1}{k} \left[\sum_{kn}^{\theta^2} - 2\mathbf{A}_k(n,:) \sum_{kn}^{\boldsymbol{\theta}\boldsymbol{\theta}'} + \text{diag}\left\{\mathbf{A}_k(n,:) \sum_k^{\boldsymbol{\theta}'\boldsymbol{\theta}'} \mathbf{A}_k(n,:)^T\right\}\right] \tag{19}$$

where $m = 1, .., d_1$, $n = 1, .., d_2$, and the sufficient statistics, which are all a function of values observed, calculated or used in time step k (e.g., $\langle w_k \rangle$, $\mathbf{z}_k$,

$\langle \boldsymbol{\theta}_k \rangle$, $\langle \boldsymbol{\theta}_{k-1} \rangle$ etc.) are:

$$\sum_k^{\mathbf{w}\mathbf{z}\boldsymbol{\theta}^T} = \langle w_k \rangle \, \mathbf{z}_k \, \langle \boldsymbol{\theta}_k \rangle^T + \sum_{k-1}^{\mathbf{w}\mathbf{z}\boldsymbol{\theta}^T} \tag{20}$$

$$\sum_k^{\mathbf{w}\boldsymbol{\theta}\boldsymbol{\theta}^T} = \langle w_k \rangle \left\langle \boldsymbol{\theta}_k \boldsymbol{\theta}_k^T \right\rangle + \sum_{k-1}^{\mathbf{w}\boldsymbol{\theta}\boldsymbol{\theta}^T} \tag{21}$$

$$\sum_k^{\boldsymbol{\theta}\boldsymbol{\theta}'} = \langle \boldsymbol{\theta}_k \rangle \, \langle \boldsymbol{\theta}_{k-1} \rangle^T + \sum_{k-1}^{\boldsymbol{\theta}\boldsymbol{\theta}'} \tag{22}$$

$$\sum_k^{\boldsymbol{\theta}'\boldsymbol{\theta}'} = \left\langle \boldsymbol{\theta}_{k-1} \boldsymbol{\theta}_{k-1}^T \right\rangle + \sum_{k-1}^{\boldsymbol{\theta}'\boldsymbol{\theta}'} \tag{23}$$

$$\sum_{km}^{\mathbf{w}zz} = \langle w_k \rangle \, z_{km}^2 + \sum_{k-1}^{\mathbf{w}zz} \tag{24}$$

$$\sum_{km}^{\mathbf{w}z\boldsymbol{\theta}} = \langle w_k \rangle \, z_{km} \boldsymbol{\theta}_k + \sum_{k-1,m}^{\mathbf{w}z\boldsymbol{\theta}} \tag{25}$$

$$\sum_{kn}^{\theta^2} = \left\langle \theta_{kn}^2 \right\rangle + \sum_{k-1,n}^{\theta^2} \tag{26}$$

$$\sum_{kn}^{\theta\boldsymbol{\theta}'} = \langle \theta_{kn} \rangle \, \langle \boldsymbol{\theta}_{k-1} \rangle + \sum_{kn}^{\theta\boldsymbol{\theta}'} \tag{27}$$

A few remarks should be made regarding the initialization of priors used in (9) to (11), (16) to (19). In particular, the prior scale parameters $a_{w_k,0}$ and $b_{w_k,0}$ should be selected so that the weights $\langle w_k \rangle$ are 1 with some confidence. That is to say, the algorithm starts by assuming most data samples are inliers. For example, we set $a_{w_k,0} = 1$ and $b_{w_k,0} = 1$ so that $\langle w_k \rangle$ has a prior mean of $a_{w_k,0}/b_{w_k,0} = 1$ with a variance of $a_{w_k,0}/b_{w_k,0}^2 = 1$. By using these values, the maximum value of $\langle w_k \rangle$ is capped at 1.5. This set of values is generally valid for any data set and/or application and does not need to be modified, unless the user has a good reason to insert strong biases towards particular parameter values. Since some prior knowledge about the observed data's properties must be known in order to distinguish whether a data sample is an outlier or part of the data's structure, this Bayesian approach provides a natural framework to incorporate this information.

Secondly, the algorithm is relatively insensitive to the the initialization of $\mathbf{A}$ and $\mathbf{C}$ and will always converge to the same final solution, regardless of these values. For our experiments, we initialize $\mathbf{C} = \mathbf{A} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Finally, the initial values of $\mathbf{R}$ and $\mathbf{Q}$ should be set based on the user's initial estimate of how noisy the observed data is (e.g., $\mathbf{R} = \mathbf{Q} = 0.01\mathbf{I}$ for noisy data, $\mathbf{R} = \mathbf{Q} = 10^{-4}\mathbf{I}$ for less noisy data [25]).

2.2 Relationship to the Kalman Filter

With a little algebraic manipulation, we can show that the model derived in Section 2.1 is indeed a variant of the Kalman filter. If we substitute the propagation equations, (2) and (3), into the update equations, (4) to (7), we reach recursive expressions for $\langle \boldsymbol{\theta}_k \rangle$ and $\boldsymbol{\Sigma}_k$. By applying this sequence of algebraic

manipulations in reverse order to (9) and (10), we arrive at the following:

Propagation:

$$\boldsymbol{\theta}'_k = \mathbf{A}_k \langle \boldsymbol{\theta}_{k-1} \rangle \tag{28}$$

$$\boldsymbol{\Sigma}'_k = \mathbf{Q}_k \tag{29}$$

Update:

$$\mathbf{S}'_k = \left(\mathbf{C}_k \boldsymbol{\Sigma}'_k \mathbf{C}_k^T + \frac{1}{\langle w_k \rangle} \mathbf{R}_k \right)^{-1} \tag{30}$$

$$K'_k = \boldsymbol{\Sigma}'_k \mathbf{C}_k^T \mathbf{S}'_k \tag{31}$$

$$\langle \boldsymbol{\theta}_k \rangle = \boldsymbol{\theta}'_k + K'_k \left(\mathbf{z}_k - \mathbf{C}_k \boldsymbol{\theta}'_k \right) \tag{32}$$

$$\boldsymbol{\Sigma}_k = \left(\mathbf{I} - K'_k \mathbf{C}_k \right) \boldsymbol{\Sigma}'_k \tag{33}$$

Close examination of the above equations show that (9) and (10) in the Bayesian model correspond to standard Kalman filter equations, with modified expressions for $\boldsymbol{\Sigma}'_k$ and $\mathbf{S}'_k$ and time-varying system dynamics. $\boldsymbol{\Sigma}'_k$ is no longer *explicitly* dependent on $\boldsymbol{\Sigma}_{k-1}$ since $\boldsymbol{\Sigma}_{k-1}$ does not appear in (29). However, the current state's covariance $\boldsymbol{\Sigma}_k$ is still dependent on the previous state's covariance $\boldsymbol{\Sigma}_{k-1}$ (i.e. it is dependent through the other parameters K' and $\mathbf{C}_k$).

Additionally, the term $\mathbf{R}_k$ in $\mathbf{S}'_k$ is now weighted. Equation (11) reveals that if the prediction error in $\mathbf{z}_k$ is so large that it dominates the denominator, then the weight $\langle w_k \rangle$ of that data sample will be very small. As this prediction error term in the denominator goes to ∞, $\langle w_k \rangle$ approaches 0. If $\mathbf{z}_k$ has a very small weight $\langle w_k \rangle$, then $\mathbf{S}'_k$, the posterior covariance of the residual prediction error, will be very small, leading to a very small Kalman gain K'_k. In short, the influence of the data sample $\mathbf{z}_k$ will be downweighted when predicting $\boldsymbol{\theta}_k$, the hidden state at time step k.

The resulting Bayesian algorithm has a computational complexity on the same order as that of a standard Kalman filter, since matrix inversions are still needed (for the calculation of covariance matrices), as in the standard Kalman filter. In comparison to other Kalman filters that use heuristics or require more involved computation/implementation, this outlier-robust Kalman filter is principled and easy to implement.

2.3 An Alternative Kalman Filter

We explored a variation of the previously introduced robust Kalman filter. Instead of performing a full Bayesian treatment of the weighted Kalman filter, we use the standard Kalman filter equations, (2) to (7), and modify (4) so that the output variance for $\mathbf{z}_k$, $\mathbf{R}_k$, is now weighted—as in our original model in (8): $\mathbf{S}'_k = \left(\mathbf{C}_k \boldsymbol{\Sigma}'_k \mathbf{C}_k^T + \frac{1}{\langle w_k \rangle} \mathbf{R}_k \right)^{-1}$.

We learn the weights $\langle w_k \rangle$ using (11) from the robust Kalman filter and estimate the system dynamics at each time step using a maximum likelihood framework (i.e., using (16) to (19) from the robust Kalman filter). $\boldsymbol{\Sigma}_k$ is now

explicitly dependent on $\boldsymbol{\Sigma}_{k-1}$ (i.e. $\boldsymbol{\Sigma}_{k-1}$ appears in the propagation equation for $\boldsymbol{\Sigma}_k$). We introduce this somewhat unprincipled and arbitrarily derived filter for comparison with our weighted Kalman filter.

3 Experimental Results

We evaluated our weighted robust Kalman filter on data collected from a a robotic dog, LittleDog, manufactured by Boston Dynamics Inc. (Cambridge, MA), and compared it with three other filters. We omitted the filters of [16] and [17], since we had difficulty implementing them and getting them to work. Instead, we used a hand-tuned thresholded Kalman filter to serve as a baseline comparison. The three filters consist of i) the standard Kalman filter, ii) the alternative weighted Kalman filter introduced in Section 2.3, and iii) a Kalman filter where outliers are determined by thresholding on the Mahalanobis distance. If the Mahalanobis distance exceeds a certain threshold value, it is considered an outlier and ignored. This threshold value is *hand-tuned manually* in order to find the optimal value for a particular data set. If we have a priori access to the entire data set and are able to tune this threshold value accordingly, the thresholded Kalman filter gives *near-optimal* performance.

For this paper and these experiments, we are interested in the Kalman filter's prediction of the observed (output) data and detection of outliers in the observations. We are not interested in the estimation of the system dynamics or in the estimation (or outlier detection) of the states. Estimation of the system matrices for the purpose of parameter identification is a different problem, and details on this difference are highlighted in [26]. Similarly, detecting outliers in the states is a different problem and left to another paper.

3.1 LittleDog Robot

We evaluated all filters on a 12 degree-of-freedom robotic dog, LittleDog, shown in Fig. 2. The robot dog has two sources that measure its orientation: a motion capture (MOCAP) system and an on-board inertia measurement unit (IMU). Both provide a quaternion q of the robot's orientation: q_{MOCAP} from the MOCAP and q_{IMU} from the IMU.

q_{IMU} drifts over time, since the IMU cannot provide stable orientation estimation but its signal is clean. The drift that occurs in the IMU is quite common in systems where sensors collect data that need to be integrated. In contrast, q_{MOCAP} has outliers and noise, but no drift. We would like to estimate the offset between q_{MOCAP} and q_{IMU}, and this offset is a *noisy slowly drifting signal containing outliers*. There are various approaches to estimating this slowly drifting signal, depending on the quality of estimate desired. We can estimate it with a straight line, as done in [20]. Alternatively, if we want to estimate the signal more accurately, we can use the proposed outlier-robust Kalman filter to track it. For optimal performance, we manually tuned $\mathbf{C}$, $\mathbf{A}$, $\mathbf{R}$ and $\mathbf{Q}$ for the standard Kalman filter—a tricky and time-consuming process. The system dynamics of

Fig. 2. LittleDog

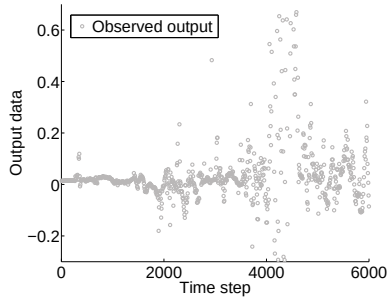
the thresholded Kalman filter were learnt using a maximum likelihood framework (i.e. using (16) to (19) without any weights). Its threshold parameter was manually tuned for best performance on this data set.

Fig. 3(a) shows the offset data between q_{MOCAP} and q_{IMU} for one of the four quaternion coefficients, collected over 6000 data samples, at 1 sample/time step. As expected, the standard Kalman filter fails to detect and ignore the outliers occurring between the 4000th and 5000th sample, as seen in Fig. 3(b). When comparing our weighted robust Kalman filter with the other remaining two filters, Fig. 3(c) shows that the thresholded Kalman filter does not react as violently as the standard Kalman filter to outliers and, in fact, appears to perform similarly to the weighted robust Kalman filter. This is to be expected, given we hand-tuned the threshold parameter for optimal performance (i.e. the thresholded Kalman filter is *near-optimal* in this experiment). Notice that the weighted robust filter does not track noise in the data as closely as the alternative filter. This is a direct result of higher Kalman gains and a consequence of $\boldsymbol{\Sigma}_k$'s *explicit* dependency on $\boldsymbol{\Sigma}_{k-1}$ in the alternative filter.

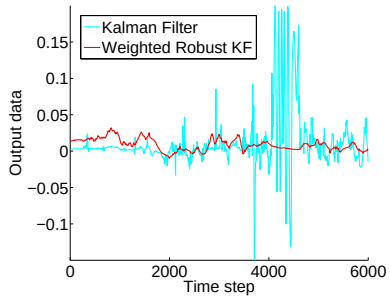
In this experiment, the advantages offered by the weighted Kalman filter are clear. It outperforms the traditional Kalman filter and alternative Kalman filter, while achieving a level of performance on par with a thresholded Kalman filter (where the threshold value is manually tuned for optimal performance).

4 Conclusions

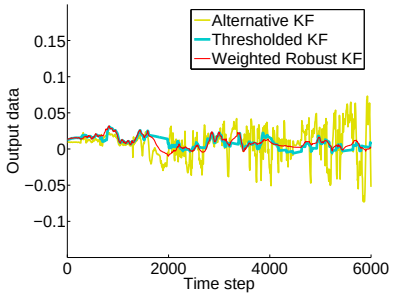
We derived a novel Kalman filter that is robust to outliers in the observations by introducing weights for each data sample. This Kalman filter learns the weights and the system dynamics, without needing manual parameter tuning by the user, heuristics or sampling. It performs as well as a hand-tuned Kalman filter (that required prior knowledge of the data) on real robotic data. It provides an easy-to-use competitive alternative for robust tracking of sensor data and offers a simple outlier detection mechanism that can potentially be applied to more complex, nonlinear filters.



(a) Observed data from LittleDog robot: a slowly drifting noisy signal with outliers



(b) Predicted data for the Kalman filter (KF) and weighted robust KF.



(c) Predicted data for the thresholded KF, alternative KF and weighted robust KF

Fig. 3. Observed vs. predicted data from LittleDog robot shown for all Kalman filters, over 6000 samples

Acknowledgments

This research was supported in part by National Science Foundation grants ECS-0325383, IIS-0312802, IIS-0082995, ECS-0326095, ANI-0224419, a NASA grant AC#98−516, an AFOSR grant on Intelligent Control, the ERATO Kawato Dynamic Brain Project funded by the Japanese Science and Technology Agency, and the ATR Computational Neuroscience Laboratories.

References

1. Moore, D.S., McCabe, G.P.: Introduction to the Practice of Statistics. W.H. Freeman & Company (March 1999)
2. Jazwinski, A.H.: Stochastic Processes and Filtering Theory. Academic Press (1970)
3. Kalman, R.E.: A new approach to linear filtering and prediction problems. In Transactions of the ASME - Journal of Basic Engineering **183** (1960) 35–45
4. Morris, J.M.: The Kalman filter: A robust estimator for some classes of linear quadratic problems. IEEE Transactions on Information Theory **22** (1976) 526–534
5. Tukey, J.W.: A survey of sampling from contaminated distributions. In Olkin, I., ed.: Contributions to Probability and Statistics. Stanford University Press (1960) 448–485
6. Huber, P.J.: Robust estimation of a location parameter. Annals of Mathematical Statistics **35** (1964) 73–101
7. Sorensen, H.W., Alspach, D.L.: Recursive Bayesian estimation using Gaussian sums. Automatica **7** (1971) 467–479
8. West, M.: Robust sequential approximate Bayesian estimation. Journal of the Royal Statistical Society, Series B **43** (1981) 157–166
9. West, M.: Aspects of Recursive Bayesian Estimation. PhD thesis, Dept. of Mathematics, University of Nottingham (1982)
10. Meinhold, R.J., Singpurwalla, N.D.: Robustification of Kalman filter models. Journal of the American Statistical Association (1989) 479–486
11. Masreliez, C.: Approximate non-Gaussian filtering with linear state and observation relations. IEEE Transactions on Automatic Control **20** (1975) 107–110
12. Schick, I.C., Mitter, S.K.: Robust recursive estimation in the presence of heavy-tailed observation noise. Annals of Statistics **22**(2) (1994) 1045–1080
13. Kramer, S.C., Sorenson, H.W.: Recursive Bayesian estimation using piece-wise constant approximations. Automatica **24**(6) (1988) 789–801
14. Kitagawa, G., Gersch, W.: Smoothness priors analysis of time series. In: Lecture Notes in Statistics. Springer-Verlag (1996)
15. Huber, P.J.: Robust Statistics. Wiley (1973)
16. Durovic, Z.M., Kovacevic, B.D.: Robust estimation with unknown noise statistics. IEEE Transactions on Automatic Control **44** (1999) 1292–1296
17. Chan, S.C., Zhang, Z.G., Tse, K.W.: A new robust Kalman filter algorithm under outliers and system uncertainties. In: IEEE International Symposium on Circuits and Systems. IEEE (2005) 4317–4320
18. Myers, K.A., Tapley, B.D.: Adaptive sequential estimation with unknown noise statistics. IEEE Transactions on Automatic Control **21** (1976) 520–523
19. Gelman, A., Carlin, J., Stern, H., Rubin, D.: Bayesian Data Analysis. Chapman and Hall (2000)

20. Ting, J., D'Souza, A., Schaal, S.: Automatic outlier detection: A Bayesian approach. In: IEEE International Conference on Robotics and Automation. (2007)
21. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. Journal of Royal Statistical Society. Series B **39**(1) (1977) 1–38
22. Ghahramani, Z., Beal, M.: Graphical models and variational methods. In Saad, D., Opper, M., eds.: Advanced Mean Field Methods - Theory and Practice. MIT Press (2000)
23. Ghahramani, Z., Hinton, G.: Parameter estimation for linear dynamical systems. Technical report, University of Toronto (1996)
24. Neal, R.M., Hinton, G.E.: A view of the EM algorithm that justifies incremental, sparse, and other variants. In Jordan, M.I., ed.: Learning in Graphical Models. MIT Press (1999) 355–368
25. Maybeck, P.S.: Stochastic models, estimation, and control. Volume 141 of Mathematics in Science and Engineering. Academic Press (1979)
26. Ting, J., D'Souza, A., Schaal, S.: Bayesian regression with input noise for high dimensional data. In: Proceedings of the 23rd International Conference on Machine Learning, ACM (2006) 937–944

Learning novel concepts: beyond one-class classification with OLINDDA

Eduardo J. Spinosa[1], André Ponce de Leon F. de Carvalho[1], and João Gama[2]

[1] University of São Paulo (USP), Institute of Mathematical and Computer Sciences (ICMC), Caixa Postal 668, 13560-970, São Carlos, SP, Brazil
ejspin@icmc.usp.br*, andre@icmc.usp.br
www.icmc.usp.br/~ejspin, www.icmc.usp.br/~andre

[2] University of Porto (UP), Artificial Intelligence and Computer Science Laboratory (LIACC), Rua Campo Alegre, 823, 4150, Porto, Portugal
jgama@liacc.up.pt
www.liacc.up.pt/~jgama

Abstract. OLINDDA (OnLIne Novelty and Drift Detection Algorithm) addresses the problem of novelty detection in an online continuous learning scenario as an extension to a single-class classification problem. This paper presents its current version, that evolved toward the discovery of new concepts initially as emerging clusters and further as cohesive sets of clusters. New strategies for validation and merging of clusters as well as for dynamically adapting the number of clusters are discussed and experimentally evaluated.

Key words: Novelty detection, Unsupervised learning, Clustering, K-Means

1 Introduction

Novelty Detection (ND) in the context of machine learning [6] is the identification of novel profiles in data. Along with the ability to deal with concept drift [10], it is an important attribute of any learning system applied to problems in which data distribution may change over time. Typical applications of ND include the identification of faults in various types of machines, attacks in computer networks, novel topics in news documents, regions of interest in medical images, among others.

Problems involving data streams can benefit from online learning algorithms capable of detecting novel profiles, such as the one presented herein. Working with data streams imposes, however, a series of restrictions [1] that are not addressed in this paper, whose focus is to present a new approach to ND while considering data streams as a promising future application.

The text is organized as follows. The next section presents the motivations for this work, while briefly mentioning some related works. Section 3 describes the

* Alternate e-mail: ejspin@yahoo.com

proposed approach. Sub-section 3.8 closes by commenting on the improvements of its current version over the one previously published [8]. Section 4 discusses experiments aimed at comparing both versions. Finally, Section 5 concludes the paper and presents future challenges.

2 Related works

ND is frequently treated as a *one-class classification* problem [9], where only information regarding a single class is available for training. That given class usually represents the *normal* or expected condition. Such an approach requires the ability to *learn from positive-only examples*, which is a much harder task than standard classification, since negative or counter-examples play an important role in defining the degree of generalization of a description.

Several machine learning techniques can be modified and have been employed to perform ND [4] [5]. However, most of them focus solely on solving the one-class classification problem, i.e., comparing a new example to a single static model to decide whether the example itself is a novelty. The terms *anomaly*, *surprising event* or *outlier* [2] are also sometimes applied, even though the meaning of these terms may not be the same.

Another aspect that is usually not addressed by the ND techniques described in the literature is the incorporation of novel profiles to the knowledge structure. This is an important attribute, considering that many typical applications of ND involve dealing with data distributions that change over time.

The k-means clustering algorithm, which is used by proposed approach, has been previously applied for one-class classification [9]. In that work, the decision boundaries are positioned in a way that forces a certain percentage of the examples of the target class to be placed outside the target description, which is defined by a fixed parameter. The number of clusters k is also manually set. Furthermore, as in many other ND techniques, any new example that does not fit the target description is considered novelty, and learning is performed in a single training phase.

In another cluster-based approach to ND [3], k-means is applied to improve the quality of clusters originally generated by a standard leader-follower algorithm. The detection of novelty is then performed by observing changes that may occur to each cluster's density function.

The approach proposed in this paper does not consider a single example as novelty, since a single example may appear due to noise, or be an outlier. It intends to take the single-class classification problem one step forward by attempting to discover novelty as a *novel concept*, represented by a cohesive cluster of examples that share similar characteristics and emerges from those that have not been previously and individually explained. We believe that this is a natural approach considering the continuous learning aspect inherent to ND: as time passes and new examples are received, those that have not been explained may start to make sense together in the form of a cohesive cluster. By doing so, we intend to produce robust predictions of emerging classes, in an attempt to

approximate to the real class structure at any time in an unsupervised way. In addition to that, the approach proposed in this paper continuously incorporates the knowledge of newly discovered concepts, since such knowledge is likely to be useful in explaining a larger number of examples in the future.

The following section describes the proposed approach.

3 Proposed approach

OLINDDA (OnLIne Novelty and Drift Detection Algorithm) implements our approach to ND. It builds on our initially proposed algorithm [8] and advances in the direction of continuously identifying, incorporating and merging concepts over time. This section describes the current version of the algorithm. Changes from the one previously published are discussed in Section 3.8.

3.1 Model structure

OLINDDA uses three hypersphere-based models to store knowledge about (1) the normal profile, (2) extensions to this normal profile and (3) novel profiles. The *normal* model is the only static one, remaining as a reference to the initial learning stage. It corresponds to what is usually employed by most ND techniques. The *extension* and *novelty* models can be created and continuously updated, allowing OLINDDA to deal with concepts that appear or change over time. Once newly discovered concepts become part of these two models, they will also help to explain future examples, thus reducing the cost of exploring regions of the feature space that have already been explored. Additionally, such an incremental structure allows the algorithm to start with a basic description, weakening the requirement of an initial set that thoroughly describes the normal profile.

3.2 Initial learning phase

OLINDDA starts by modeling the *normal* or expected behavior in the domain under investigation, by analyzing a set of normal examples. They usually belong to a single class, hence the term *one-class classification*. In the problem of intrusion detection in computer networks, for instance, this initial data set would be built from standard network traffic, without any examples of attacks.

To model the normal profile, OLINDDA produces k clusters using the k-means clustering algorithm. The normal model is composed of k hyperspheres, built in feature space, obtained directly from the clusters and represented by their centers and radii. Each hypersphere center is the centroid of its cluster, and its radius is the Euclidean distance from the centroid to the farthest example of the respective cluster.

3.3 Continuous unsupervised learning phase

The arrival of new (unseen) examples marks the start of a continuous unsupervised learning phase. For each new example, the algorithm first checks if it can be explained by the knowledge acquired until that point, represented by (up to) the three models previously described. If the coordinates of the example in feature space lie inside a hypersphere of any of the existing models, it is considered explained by the corresponding model; statistics are updated, and the example is discarded. Otherwise, the example is marked as a member of an *unknown* profile and moved to a short-time memory for further analysis.

Initially, OLINDDA is capable of distinguishing regions that correspond to the normal profile (inside any of the hyperspheres of the normal model) from those that have not been explored yet, named *unknown*. In a stable situation, the normal model is expected to explain the majority of the new examples. As new concepts emerge and are added to the *extension* and *novelty* models, it will also be able to explain examples of such concepts.

3.4 Learning new concepts by validating emerging clusters

OLINDDA learns new concepts initially as clusters, formed by examples previously considered *unknown*, that comply with certain restrictions. In order to discover these clusters, each time a new *unknown* example is found, k *candidate* clusters are generated from the examples currently available at the *short-time memory of unknown profiles*. These candidate clusters are then evaluated in an attempt to determine if any of them presents enough evidence of the appearance of a new concept, represented by a so-called *valid* cluster.

This is not a trivial task, since it is a totally unsupervised process. On the other hand, the fact that no labels are required allows its application to a large amount of data that could not be manually classified.

Several metrics can be used to evaluate clusters from various points of view. OLINDDA considers a cluster's density and the number of examples as the criteria for validating clusters. The density d of a cluster c is defined as:

$$d(c) = \frac{ne(c)}{V(c)} \tag{1}$$

where $ne\,(c)$ is the number of examples that belong to c and $V\,(c)$ is the volume of the hypersphere whose radius is the distance from the cluster's centroid to the farthest example that belongs to c. The volume $V\,(c)$ in an n-dimensional space is given by:

$$V\,(c,n) = \frac{\pi^{\frac{n}{2}} R^n}{\Gamma\left(\frac{n}{2}+1\right)} \tag{2}$$

where Γ is the gamma function, defined by:

$$\Gamma\left(\frac{n}{2}+1\right) = \begin{cases} \left(\frac{n}{2}\right)!, & for\ even\ n; \\ \sqrt{\pi}\frac{n!!}{2^{(n+1)/2}}, & for\ odd\ n. \end{cases} \tag{3}$$

For a candidate cluster to be considered valid, the first condition is that its density be equal to or higher than the minimum density among the clusters of the normal model.

To avoid clusters with too few examples, a minimum number of examples per cluster is required as a second condition, defined as a parameter. A value between 3 and 5 has been empirically determined as adequate.

3.5 Attempting to determine the nature of new concepts

Once a valid cluster is identified, OLINDDA proceeds to assess its similarity to the normal concept. We consider that an *extension* of the normal concept should naturally present some similarity to it, which, in terms of distances in feature space, means that the new concept should be located in the vicinity of the region associated to the normal concept. On the other hand, a new concept that is dissimilar to *normal* may represent a novel concept, or *novelty*.

To materialize this notion of vicinity of the normal concept, OLINDDA creates a hypersphere centered at the centroid of the centroids of the clusters of the normal model, and whose radius is the distance to the farthest centroid. If the centroid of the new valid cluster is located inside this hypersphere, the new concept is labeled *extension*. Otherwise, it is considered *novelty*.

As previously mentioned, newly discovered concepts update their corresponding models, which facilitates the classification of future examples. Since models are composed mainly of the coordinates of centroids and radii, besides a few other distances and statistics, model updating is fast and performed incrementally, which is an important issue in applications involving, for instance, data streams, where time and space constraints apply.

3.6 Merging similar clusters to produce cohesive concepts

A new valid cluster may itself represent a new concept. However, depending on the data distribution, a concept may be more adequately described by a set of clusters. For that reason, OLINDDA will also evaluate the similarity between the new concept and existing concepts of the corresponding model. It does that by checking if the new valid cluster intercepts any of the previous clusters. If it does not, then the cluster is considered a new concept on its own and receives a new label. However, if the new valid cluster intercepts one or more existing clusters, they are grouped under the same label and their statistics are merged.

A single cluster may trigger a sequence of mergers, and this process tends to produce a smaller number of concepts (labels) that are usually easier to analyze.

A typical experiment would be to present OLINDDA with examples of a single class (representing the normal profile) in the initial phase, and allow the algorithm to discover the remaining classes as novel concepts. In that scenario, our final goal would be to have produced a class structure as similar as possible to the real one, and the merging of concepts helps directing the algorithm toward that.

3.7 Dynamically adapting the number of clusters

The number of clusters k is an intrinsic parameter of the k-means clustering algorithm, which is used by OLINDDA (1) to create the initial normal model, as described in Section 3.2 and (2) to periodically generate candidate clusters in the online phase, as described in Section 3.4.

In the initial model, k is fixed and defined as a parameter, since it depends on the data distribution. For the generation of candidate clusters in the online phase, however, k is dynamically adapted to optimize the chance of discovering a valid cluster. This is done by increasing or decreasing k according to certain conditions. If the value of k is lesser than the optimum, the algorithm will generate clusters whose densities are lesser than the required threshold for cluster validation. On the other hand, if the value of k is greater than the optimum, the candidate clusters will tend to have fewer examples than the required minimum.

The automatic adaptation of k takes place after each iteration in which candidate clusters were generated. If at least one candidate cluster is considered valid, the value of k is maintained. Otherwise, OLINDDA checks what prevented each cluster from being accepted: too low density or too few examples. Then, considering the most frequent cause of failure for all candidate clusters, it decides how to adapt the value of k. If the majority of failures is due to low density, k is increased. If too few examples is the most frequent cause of failure, k is decreased. After a few iterations, k tends to stabilize around the optimum value that generates valid clusters.

3.8 Changes from the previous version

The initially published version of OLINDDA [8] has been greatly improved. It differs from the present version, described in Section 3, in the following aspects:

Management of *k* The previous version of the algorithm did not employ a dynamic adaptation of k aimed at optimizing the discovery of valid clusters. The value of k was not fixed either. It was set according to the number of examples available for clustering at the short-time memory of unknown profiles at any time. This was achieved by imposing an *average number of examples per cluster* as a parameter, and calculating k according to the number of examples available.

Cluster validation criteria The previous version considered the average distance between examples and the respective centroid as the criterion for cluster validation. The average of this metric for the normal model was taken as a higher threshold for acceptance.

Merging of clusters The previous version did not merge intercepting clusters. The merging of clusters allows OLINDDA to discover concepts formed by more than one cluster, reducing the number of labels, which may also ease the analysis.

4 Experimental evaluation

This section compares results obtained with the current version of OLINDDA to those that have been previously published [8]. OLINDDA was implemented in R [7] and makes use of its built-in implementation of k-means.

4.1 Data and setup

Experiments were performed with the data sets depicted in Figure 1. To simulate novel concepts, for each data set, each class was selected as the normal profile, while the remaining classes were considered novel concepts to discover. All metrics represent the average of 10 runs performed for each experiment. Datasets were scaled and shuffled prior to each run.

Data set (Source)	Number of attributes	Number of examples	Classes (Number of examples per class)		
			Class 1	Class 2	Class 3
Balance Scale (UCI)	4	625	Balanced (49)	Left (288)	Right (288)
Biomed (StatLib)	5	194	Carrier (67)	Normal (127)	-
Breast Cancer Wisconsin (UCI)	9	683	Benign (444)	Malignant (239)	-
Ionosphere (UCI)	33	351	Bad (126)	Good (225)	-
Iris (UCI)	4	150	Setosa (50)	Versicolor (50)	Virginica (50)
Lymphoma32A (Alizadeh, Reduced version)	32	47	A (23)	GC (24)	-
Mushroom (UCI)	22	8124	Edible (4208)	Poisonous (3916)	-

Fig. 1. Data sets used in the experiments.

Regarding parameters, the two versions of OLINDDA used different criteria for defining k (see Section 3.7 for the current version and Section 3.8 for the previous version). To produce comparable results, the value of k for the initial normal model has to be the same. This is achieved by setting the initial k in the current version as the number of examples in the initial batch of normal examples divided by the average number of examples per cluster that has been used in the experiments with the previous version.

4.2 Evaluating the distinction between normal and unknown

The distinction between normal and unknown examples, which corresponds to the one-class classification phase, is evaluated by the *false-unknown error rate*, defined by: $e_{FUnk} = \frac{FUnk}{Nor}$, where $FUnk$ is the number of *normal* examples wrongfully identified as unknown, and Nor is the total number of *normal* examples; and by the *false-normal error rate*, defined by: $e_{FNor} = \frac{FNor}{Unk}$, where $FNor$ is the number of *unknown* examples wrongfully identified as normal, and Unk is the total number of *unknown* examples.

As previously mentioned, in order to simulate novel concepts, one class is selected as the normal profile while the remaining classes serve as novel concepts

we wish to discover. For this initial distinction between normal and unknown, *unknown* examples are those that belong to any class other than the one which has been selected as normal.

A high value of one of these metrics indicates either overfitting or underfitting. To assess how the current version of OLINDDA compares to its previous, we analyze the average and standard deviation of these two error rates over 10 runs. Figure 2 displays the results.

Data set	OLINDDA's Version	Parameters			Normal concept is Class 1				Normal concept is Class 2				Normal concept is Class 3			
					Error rates				Error rates				Error rates			
					eFUnk		eFNor		eFUnk		eFNor		eFUnk		eFNor	
		Ex Nor Model	Avg Ex/Cluster	k Nor Model	Mean	SDev	Mean	SDev	Mean	SDev	Mean	SDev	Mean	SDev	Mean	SDev
Balance Scale	Previous	40	10	-	**0.13**	0.14	**0.80**	0.07	**0.19**	0.05	**0.23**	0.07	**0.17**	0.07	**0.25**	0.06
	Currrent	40	-	4	**0.04**	0.06	**0.83**	0.07	**0.09**	0.03	**0.24**	0.10	**0.10**	0.03	**0.24**	0.08
Biomed	Previous	20	10	-	**0.18**	0.11	**0.85**	0.12	**0.15**	0.09	**0.25**	0.08				
	Currrent	20	-	2	**0.19**	0.09	**0.72**	0.18	**0.10**	0.09	**0.29**	0.16				
Breast Wisconsin	Previous	50	10	-	**0.10**	0.06	**0.02**	0.03	**0.19**	0.08	**0.23**	0.19				
	Currrent	50	-	5	**0.06**	0.02	**0.03**	0.03	**0.12**	0.03	**0.34**	0.23				
Ionosphere	Previous	100	10	-	**0.25**	0.11	**0.98**	0.01	**0.15**	0.03	**0.07**	0.03				
	Currrent	100	-	10	**0.23**	0.10	**0.99**	0.01	**0.11**	0.05	**0.11**	0.07				
Iris	Previous	20	10	-	**0.13**	0.13	**0.00**	0.00	**0.12**	0.10	**0.16**	0.04	**0.12**	0.09	**0.28**	0.12
	Currrent	20	-	2	**0.11**	0.07	**0.00**	0.00	**0.14**	0.11	**0.14**	0.05	**0.13**	0.11	**0.29**	0.11
Mushroom	Previous	1000	8	-	**0.31**	0.01	**0.33**	0.02	**0.28**	0.02	**0.52**	0.01				
	Currrent	1000	-	125	**0.04**	0.01	**0.00**	0.00	**0.02**	0.01	**0.01**	0.01				

Fig. 2. Comparison between the current and the previous versions of OLINDDA in terms of distinction between normal and unknown.

In general, the current version performs as well as or better than the previous in terms of the distinction between normal and unknown. For the Mushroom data set, the performance gain is more evident. Of the 14 experiments, the current version only fails to obtain an adequate generalization degree in three cases, where very high values of e_{FNor} indicate underfitting (gray area). However, two of these three failures happen in experiments where an unnatural choice was made for the normal profile: Biomed with Class 1 (Carrier) as normal as opposed to the natural choice (Class 2: Normal), and Ionosphere with Class 1 (Bad) as normal as opposed to the natural choice (Class 2: Good).

It is important to stress that the error rates in one-class classification are not directly comparable to those of standard classification, since only members of a single class are available, which makes this a much harder task.

4.3 Evaluating the discovery of valid clusters

One of the major goals that guided OLINDDA's development has been the discovery of new concepts. This is naturally influenced by the cluster validation criteria, but also benefits from the dynamic adaptation of the number of clusters introduced by the current version.

To evaluate OLINDDA's ability to discover valid clusters, we analyze how such clusters are formed in terms of the real classes. Figure 3 compares the class distribution by the end of the learning process for the previous and current versions. Since cluster validation affects the algorithm's ability to discover new concepts as either *novelty* or *extension*, not the distinction between these two, we merged, under a single column named *discovered*, statistics corresponding to examples that formed valid clusters of either *extension* or *novelty* and examples identified as members of a previously discovered concept of either *extension* or *novelty*. Ideally, given Class 1 as the normal profile, for instance, a high percentage of the elements of this class would appear under *ident nor* (identified as normal), a high percentage of the members of the remaining classes would appear under *discovered*, leaving few examples in the *unknown* column.

In order to evaluate if valid clusters are composed of elements that share similar characteristics, we measure each cluster's *purity*, defined by the percentage of examples of the predominant class. The *purity* column in Figure 3 displays the average purity for all clusters discovered over 10 runs.

The increase in the average percentage of patterns discovered shows that the current version of OLINDDA highly improves the identification of valid clusters. Moreover, high values of purity indicate that the discovered concepts are indeed composed of a coherent set of examples, as intended.

4.4 Illustration of the merging of concepts

By merging similar concepts as described in Section 3.6, OLINDDA proceeds toward generating a class structure that aims at approximating to the real one. Figure 4 displays an example of the merging of clusters over time. These are the results of a single run with the Iris data set, using 30 examples of the class Iris-versicolor to build the initial normal model.

Each chart represents the class structure obtained by OLINDDA at a certain point of the learning process, from 10% to 100%. The set of charts on the left corresponds to concepts of the *novelty* model, and the ones on the right are concepts found as an *extension* of the normal behavior. The black bar indicates examples identified as normal. Each color represents a concept, including examples that either formed the clusters or were explained by them. There is no correspondence between the colors of concepts on the left and on the right set of charts, which means that, in this case, OLINDDA has found two different concepts marked in red: one small concept of extension and one of novelty. This red novelty concept is the result of at least 2 mergers: first with the green colored concept and later with the magenta colored concept. The resulting concept corresponds to the real class Iris-Setosa. The dark blue and light blue concepts have also merged into a single concept that corresponds to Iris-Virginica. This example shows OLINDDA's ability to build cohesive concepts that make sense in terms of the real class structure.

5 Conclusion

The detection of novel concepts is a very important aspect of a learning system. This paper presents OLINDDA, an approach that intends to take novelty detection beyond one-class classification. By detecting emerging coherent clusters of examples and, further, by merging intercepting clusters, OLINDDA proceeds toward the construction of a class structure that aims at reproducing the real one in an unsupervised continuous learning fashion.

Experimental results show that the current version improved in both the distinction between normal and unknown and the discovery of new concepts. Yet, several topics are still to be investigated, including other clustering algorithms that may provide more flexible models and/or improve stability, and the application of OLINDDA to problems involving data streams.

Acknowledgments The authors acknowledge the support of CNPq (Ministry of Science and Technology of Brazil), CAPES (Ministry of Education of Brazil), and FCT (Ministry of Science and Technology of Portugal, under the project Adaptive Learning Systems II, POSC/EIA/55340/2004).

References

1. B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS 2002)*, pages 1–16. ACM, 2002.
2. V. Barnett and T. Lewis. *Outliers in Statistical Data*. John Wiley & Sons, 3rd edition, 1995.
3. C. Gazen, J. Carbonell, and P. Hayes. Novelty detection in data streams: A small step towards anticipating strategic surprise. In *Novel Intelligence from Massive Data (NIMD) PI Meeting*, 2005.
4. M. Markou and S. Singh. Novelty detection: a review - part 1: statistical approaches. *Signal Processing*, 83:2481–2497, 2003.
5. M. Markou and S. Singh. Novelty detection: a review - part 2: neural network based approaches. *Signal Processing*, 83:2499–2521, 2003.
6. S. Marsland. Novelty detection in learning systems. *Neural Computing Surveys*, 3:157–195, 2003.
7. R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2006. ISBN 3-900051-07-0.
8. E. J. Spinosa, A. P. L. F. de Carvalho, and J. Gama. Olindda: A cluster-based approach for detecting novelty and concept drift in data streams. In *22nd Annual ACM Symposium on Applied Computing (SAC 2007)*, pages 448–452. ACM, 2007.
9. D. M. J. Tax. *One-class classification - Concept-learning in the absence of counter-examples*. PhD thesis, Delf University of Technology, Faculty of Information Technology and Systems, 2001.
10. G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, 1996.

Data set	OLINDDA's Version	Parameters			Classes	Normal concept is Class 1				Normal concept is Class 2				Normal concept is Class 3			
		Ex Nor Model	Avg Ex/Cluster	k Nor Model		Ident Nor	Discovered	Unkown	Purity	Ident Nor	Discovered	Unkown	Purity	Ident Nor	Discovered	Unkown	Purity
Balance Scale	Previous	30	7	-	1	.81	.17	.03		.49	.48	.03		.52	.43	.05	
					2	.71	.28	.01	.93	.74	.23	.03	.79	.18	.80	.02	.81
					3	.71	.28	.01		.18	.80	.02		.73	.24	.03	
	Current		-	4	1	.72	.27	.02		.47	.52	.01		.45	.52	.02	
					2	.65	.34	.01	.89	.74	.24	.02	.82	.15	.83	.01	.82
					3	.63	.36	.01		.18	.81	.01		.73	.26	.01	
Biomed	Previous	30	7	-	1	.82	.04	.14		.23	.11	.66					
					2	.67	.30	.04	.96	.76	.09	.15	.93				
					3												
	Current		-	4	1	.73	.12	.15		.27	.45	.28					
					2	.67	.32	.01	.88	.80	.16	.05	.91				
					3												
Breast Wisconsin	Previous	30	15	-	1	.95	.00	.05		.18	.81	.01					
					2	.04	.01	.95	1.00	.89	.01	.10	.96				
					3												
	Current		-	2	1	.96	.04	.01		.24	.76	.00					
					2	.08	.85	.07	.95	.93	.06	.01	.96				
					3												
Ionosphere	Previous	30	15	-	1	.85	.00	.15		.43	.01	.57					
					2	.99	.00	.01	NA	.93	.03	.04	.93				
					3												
	Current		-	2	1	.89	.08	.03		.34	.37	.29					
					2	1.00	.00	.00	.98	.90	.10	.01	.94				
					3												
Iris	Previous	30	7	-	1	.80	.05	.16		.00	.88	.12		.00	.90	.10	
					2	.00	.55	.45	.93	.84	.06	.10	.99	.38	.38	.24	.98
					3	.00	.40	.60		.16	.39	.45		.82	.05	.13	
	Current		-	4	1	.86	.03	.11		.00	.92	.08		.00	.92	.08	
					2	.00	.91	.09	.88	.75	.12	.14	.99	.37	.56	.07	.97
					3	.00	.88	.12		.13	.72	.15		.77	.14	.09	
Lymphoma32A	Previous	15	7	-	1	.84	.05	.11		.05	.67	.28					
					2	.03	.55	.43	1.00	.73	.02	.24	1.00				
					3												
	Current		-	2	1	.89	.05	.06		.02	.82	.17					
					2	.02	.84	.14	.97	.77	.09	.14	.98				
					3												
Mushroom	Previous	100	5	-	1	.59	.41	.01		.14	.85	.01					
					2	.00	.99	.01	1.00	.55	.44	.01	1.00				
					3												
	Current		-	20	1	.61	.39	.00		.12	.88	.00					
					2	.01	.99	.00	.92	.53	.47	.00	.94				
					3												

Fig. 3. Comparison between the current and the previous versions of OLINDDA in terms of discovery of valid clusters.

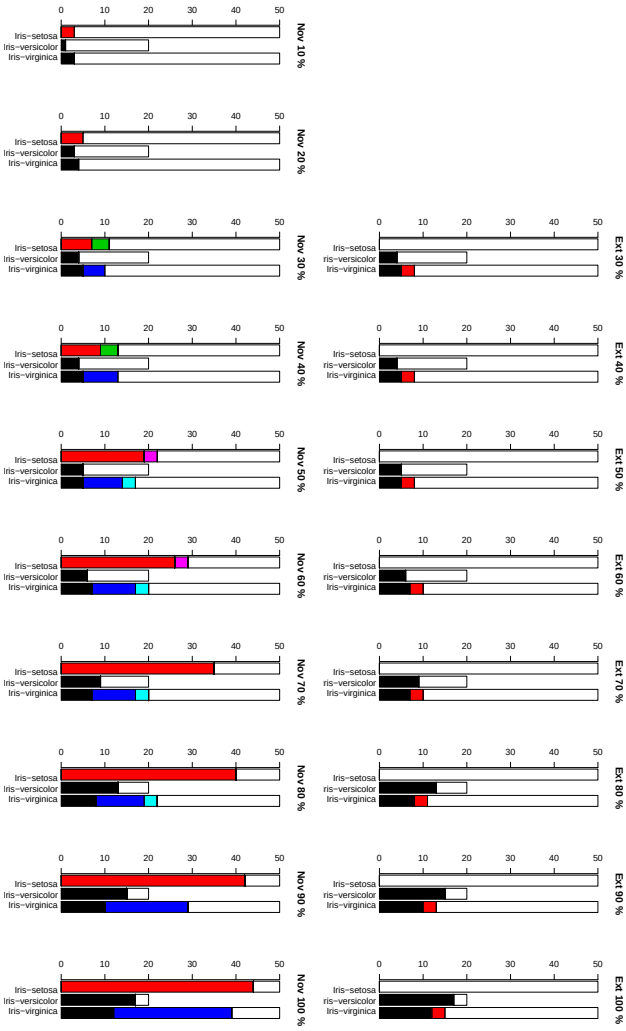


Fig. 4. Illustration of the merging of concepts.

S-means: Similarity Driven Clustering and Its application in Gravitational-Wave Astronomy Data Mining

Hansheng Lei[1], Lappoon R. Tang[1], Juan R. Iglesias[1]
Soma Mukherjee[2], and Soumya Mohanty[2]

[1] Computer Science Department
[2] The Center for Gravitational Wave Astronomy
The University of Texas at Brownsville
Brownsville TX 78520, USA
hansheng.lei@utb.edu

Abstract. Clustering is to classify unlabeled data into groups. It has been well-researched for decades in many disciplines. Clustering in massive amount of astronomical data generated by multi-sensor networks has become an emerging new challenge; assumptions in many existing clustering algorithms are often violated in these domains. For example, K means implicitly assumes that underlying distribution of data is Gaussian. Such an assumption is not necessarily observed in astronomical data. Another problem is the determination of K, which is hard to decide when prior knowledge is lacking. While there has been work done on discovering the proper value for K given only the data, most existing works, such as X-means, G-means and PG-means, assume that the model is a mixture of Gaussians in one way or another. In this paper, we present a similarity-driven clustering approach for tackling large scale clustering problem. A similarity threshold T is used to constrain the search space of possible clustering models such that only those satisfying the threshold are accepted. This forces the search to: 1) explicitly avoid getting stuck in local minima, and hence the quality of models learned has a meaningful lower bound, and 2) discover a proper value for K as new clusters have to be formed if merging them into existing ones will violate the constraint given by the threshold. Experimental results on the UCI KDD archive and realistic simulated data generated for the Laser Interferometer Gravitational Wave Observatory (LIGO) suggest that such an approach is promising.

1 Introduction

Clustering is unsupervised classification of unlabeled data, which has been a well-researched problem in many disciplines. A large portion of clustering algorithms were developed by computer scientists but much motivation came from applications of an interdisciplinary nature. It is common for modern applications in business data mining, physics, astronomy and environmental sciences to deal with a large amount of data.

While many clustering algorithms are available and work well in small scale data sets [9, 12, 16], comparative study showed that only K-means and its variants are suited for mining very large data sets [15, 22]. The K-means method is more computationally

efficient than other commonly used clustering methods such as hierarchical clustering [22, 7] and Kohonen's self organizing map (SOM) [14]. When data set size is large, hierarchical clustering and SOM can be computationally prohibitive.

However, K-means also has its own weaknesses: i) sensitive to initial partition, ii) converge to local minima, iii) the number of cluster, K, must be determined before hand, and iv) outliers from the centroid may pull the centroid away from the real one. Initial partition problem can be alleviated by repeating different initial seed settings. Local convergence is still an open problem. But in most cases, locally optimal solution is satisfactory if global optimization is too costly. Discovering K is a big weakness and several algorithms have been proposed to tackle the problem. The X-means algorithm was presented to learn K [19]. This algorithm tries many values of K and uses Bayesian Information Criterion (BIC) to score each resulting model. The K that produces the highest BIC score is chosen. Besides BIC, other scoring systems, such as Akaike Information Criterion [3] and Minimum Description Length [20] can be applied. X-means is a straightforward extension of regular K-means. The difficulty it faces is: how many K values should be chosen and compared? When the data set is large and data distribution is non-trivial, the range of possible number of clusters can be large.

Addressing problems in X-means where overfitting of data can occur, the G-means algorithm [13] is proposed to grow K from a small number. A statistical normality test is applied to each cluster to see whether it has high confidence of Gaussian distribution. If not, split the current cluster into two clusters and continue with the statistical test for the rest of the clusters. Like X-means, this algorithm is also a wrapper around K-means. It will generate a hierarchical tree of clusters. While the approach is intuitively meaningful, applying normality tests can become difficult when the set of data is extremely large (e.g. on the order of tens of thousands). The one dimensional projection of the data will be very high in dimension and tend to look Gaussian according to the Central Limit Theorem [6] and hence the need of splitting a cluster could not be detected even when it is not Gaussian. Powerful normality test like the Shapiro Wilk test [21] can handle a sample size of at most 5000. Also, the assumption of having Gaussian distribution in clusters is too strong in many real data, such as in Astronomy time series. It has been extensively tested within the LIGO community and it is known that LIGO data is not necessarily Gaussian in nature [2].

Similar to G-means, there are a number of algorithms that rely on statistical tests to check the "goodness of fit" of data according to some distribution. For example, PG-means projects both the data set and learned clusters to one dimension and then applies the Kolmogorov-Smirnov test (KS) to check the goodness of fit of the data to distribution implied by the clusters where model parameters are learned by Expectation Maximization (EM) [8]. Combining normality test and splitting for discovering K can be problematic due to application of possibly costly statistical tests and difficulty in applying the distribution test itself when data dimension is high.

Due to advances in multi-sensor networks, large amounts of astronomical data have been gathered in the form of sensor information. Discovering interesting patterns in these astronomical data has profound implication for making new discoveries in Astro-Physics, for instance, in opening up new understanding of the nature of the universe. Since sensor data can be gathered on a rate of terabytes per week, processing such

a gigantic amount of data requires at least semi-automated data mining mechanisms. Hence, clustering large amounts of astronomical data has recently become an interesting problem in the time series data community [10, 18]. Astronomical data are usually plagued with noise, very high in dimension, and not necessarily Gaussian in distribution. Limitations in current approaches motivated us to present a similarity driven clustering algorithm that we call S-means. Instead of specifying the number of clusters K, a similarity threshold T is used as a quality constraint in the search of optimal solutions to the clustering problem.

The rest of the paper is organized as follows. In Section 2, we provide a background on K-means, then the S-means algorithm is described afterward. Its time complexity and convergence are also discussed. In Section 3, we describe the experimental domains and experimental evaluation are demonstrated in which S-means is compared to existing approaches like K-means and G-means. Finally, conclusion and future work are presented in Section 4.

2 From K-means to Similarity driven clustering

Before we describe our similarity driven approach to clustering, we need to first revisit the classic K-means algorithm.

2.1 K-means

Two clustering algorithms are most popularly used: hierarchical clustering and K-means. Hierarchical clustering produces a nested hierarchy of clusters according to a pairwise distance matrix of all the given points. The hierarchy gives intuitive visualization. A user does not need to have expertise in Computer Science since no parameter excepts distance measure is needed in hierarchical clustering. However, the distance matrix limits its application to small data sets (both time complexity and space complexity are $O(n^2)$ or higher).

K-means basically divides a given data set into K clusters via an iterative refining procedure. The procedure simply consists of three steps:

1. initialize K centroids (c_i, $1 \leq i \leq K$) in the vector space.
2. Calculate the distances from every point to every centroid. Assign each point to group i, if c_i is its closest centroid.
3. Update centroids. Each centroid is updated as the mean of all the points in its group.
4. If no point changed its membership or no centroid moved, exit, otherwise, go to step 2.

The iterative procedure uses hill climbing to minimize the objective function:

$$J = \sum_{i}^{K} \sum_{j}^{N} \|x_j^{(i)} - c_i\|^2 \tag{1}$$

where $\|x_j^{(i)} - c_i\|^2$ denotes Euclidean distance between point x_j to corresponding centroid c_i. The Euclidean distance can be substituted by any distance measure.

Although the procedure will always terminate, K-means might converge to a local minima. K-means is a simple algorithm that has been employed in many problem domains. However, one of the major problems of K-means is that we do not know the right number of clusters in advance. There is no existing theoretical solution to find the optimal number of clusters for any given data set. A common approach is to score the results of multiple runs with different K values according to a given criterion. The criterion might incur new risk and parameter setting problems. We propose to use a similarity driven approach to clustering that does not require specification of K.

2.2 S-means: Similarity Driven Clustering

The clustering problem we need to solve is: *given N data points, group them into clusters such that within each cluster, all members have similarity $\geq T$ with the centroid where T is a user-defined threshold*. Similarity is a central notion in classification problem. The definition of cluster also implies that the cluster members should have high similarity with each other. The most popular Euclidean distance is a dissimilarity measure, which can be converted to a similarity measure in Gaussian form: $k(x_i, y_j) = exp(-\gamma \|x_i - y_j\|^2)$. This is also called the Radial Basis Function (RBF kernel) in kernel machines. Kernel methods all use similarity measures instead of dissimilarity. Similarity value is usually normalized to between 0 and 1; a confidence threshold in [0, 1] also makes intuitive sense to users. There are a large number of similarity measures available beside the RBF, such as correlation r, R-squared (the square of r). Indeed, any kernel function can be considered a similarity measure. Therefore, the clustering problem, if defined in terms of similarity, is more user-friendly and will likely gain more popularity due to the increasing amount of interests in kernel methods.

S-means starts from $K = 1$ by default and a user can specify any starting K. Note that the starting K is only an optional parameter in S-means. First, same as in K-means, we initialize K centroids. Second, calculate the similarities from every point to every centroid. Then, for any point, if the highest similarity to centroid c_i is $\geq T$, group it to cluster i, otherwise, add it to a new cluster (the $K + 1^{\text{th}}$ cluster). Third, update each centroid, using the mean of all member points by default. If one group becomes empty, remove its centroid and reduce K by 1. Repeat the second and third step until no new cluster is formed and none of the centroids moves.

Note that S-means is similar to K-means but with some differences. The major difference lies in the second step, which basically groups all the points to a *new* cluster whose highest similarity to *existing* centroids is below the given threshold . In K-means, all points must go to one of the existing K groups, which is unfair for some points when their similarities to corresponding closest centroid are very low. This simple difference makes big impact on the output of clusters. Also, we can let K starts from 1 and it will converge to a value, which eliminates the need of specifying a fixed K value. Also, there is a minor difference in the third step. While K is incremented by 1 if a new cluster is formed, it is decremented when some groups become empty. It is not unusual that as K keeps increasing, some old groups would disappear (as points in existing clusters could change membership as new clusters are formed). This way, K will not go beyond control.

The following is the pseudo code of S-means in Matlab style. The running code in Matlab is downloadable from our website[1].

S-means

Inputs: N data points $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]$ and similarity threshold T.
Outputs: number of clusters K , centroids $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_K]$ and labels Y.

1: $K = 1$; /* starting number can be specified*/
2: Randomly choose K centroids C;
3: change=1;
4: **while** change==1 **do**
5: $NewCluster = []$; /*initialize to empty */
6: **for** $i = 1$ to N **do**
7: **for** $j = 1$ to K **do**
8: $SimToClusters(i, j) = Similarity(x_i, c_j)$;
9: **end for**
10: **end for**/*End of similarity calculation */
11: **for** $i = 1$ to N **do**
12: $maxSim = max(SimToClusters(i, :))$;
13: **if** $maxSim \geq T$ **then**
14: $Y(i) = find(SimToClusters(i, :) == maxSim)$; /*Assign label */
15: **else**
16: $NewCluster = [NewCluster, x_i]$; /* Add to a new cluster */
17: $Y(i) = K + 1$; /*Assign label $(K + 1)$*/
18: **end if**
19: **end for**/*End of label assignment*/
20: **for** $i = 1$ to K **do**
21: $c_i = mean(find(X(:, find(Y == i))))$;
22: **if** c_i is empty **then**
23: remove c_i; /* K will also reduce by 1 */
24: **end if**
25: **end for**/*End of centroid update*/
26: **if** NewCluster is empty and no centroid changed **then**
27: change==0;
28: **end if**;
29: **end while**. /*end of algorithm*/

For the sake of simplifying description, two for loops are used to calculate the similarities in line 6-10. The loops are usually slow in Matlab. Using matrix dot product will be very efficient. Since many similarity measures can be implemented in dot product, the matrix product in Matlab can be utilized. The loop in line 11-19 varies from 1 to N. Matlab can also provide an efficient way to implement it using built-in functions like `find` and `max`. Interested readers should reference our Matlab real code.

The convergence of the S-means is guaranteed, because in the extreme case when K equals N every point has 100% similarity to itself. Of course, the extreme case is not desired. The result of K depends on threshold T. Intuitively, a high T produces more

[1] http://blue.utb.edu/hlei/Smeans/Smeans_V01.zip

clusters. When $T = 0$, S-means is reduced back to K-means. In this sense, S-means is a flexible generalization of K-means.

If S-means converges to K clusters, then time complexity is $O(N * (1 + 2 + \cdots + K)) \approx O(N * K^2/2)$. Recall that the time complexity of K-means is $O(NKL)$, where L is the number of iterations, strongly related to K and the distribution of data points. If using model selection based method to try different K and choose the best one, then the time complexity is approximately $O(N * K^2/2 * L)$, assuming K value varies from 1 to desired number of clusters. Besides avoiding the use of statistical tests (since both the number of data points and the data dimensionality could be high), S-means has advantages in low time complexity. In the following section, extensive experiments were performed to evaluate S-means from different perspectives.

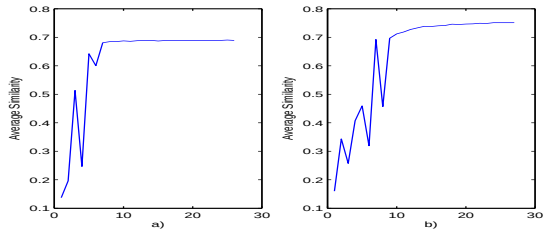


Fig. 1: S-means converges on dataset SCT in less than 30 iterations. a) the maximum number of clusters is restricted (up-bound is set 6). b) without restriction (up-bound is set 600).

3 Experiments

First, a small dataset was used to evaluate the convergence and execution speed of S-means. Second, a medium size dataset with ground-truth class labels was used to evaluate the accuracy of S-means in comparison with K-means and G-means. Third, we applied S-means to mine compact clusters in a simulated large dataset of Gravitational-wave time series. All the following experiments were performed in Matlab on a SUN Ultra 40 microsystem that has 2.8Ghz CPU and 3.0G RAM memory.

3.1 Convergence and Execution speed

We adopted the popular toy dataset Synthetic-Control time series dataset (SCT) to test the convergence and robustness of S-means [4]. SCT contains 600 samples of synthetically generated control charts. The length of each sample is 60. The similarity measure used is R-squared (squared Pearson's r) which is essentially equivalent to Euclidean distance after mean-variance normalization [17].

Fig. 1 show the convergence of S-means with $T = 0.7$ with/withoud restriction on the maximum number of clusters. Without restriction, S-means need more iterations

to finish and more clusters are generated. The average similarity of all points to their corresponding centroids is an equivalent measure of the objective function described in equation (1). Like K-means, S-means is also a hill-climbing algorithm. Although global optimum might not be reached, convergence is guaranteed.

We varied T from 0.05 to 0.75 by step 0.05 to watch the changing of the number of clusters on the SCT dataset. The result is illustrated in fig. 2, from which we can see that the number of returned clusters are sensitive to the threshold setting. Increasing similarity threshold T significantly increases the number of clusters, because it imposes the requirement that all cluster are compact (minimum similarity to the centroid is no less than T). Depending on the similarity adopted and the mining target, intuitive T should be properly set. Big T tends to lead to overfitting, which can be considered as one weakness of S-means. But from perspective of outlier detection, it is a good phenomenon that some outliers are grouped as single-item stand-alone clusters. That is, the clusters with only one elements are outliers which might interest user.

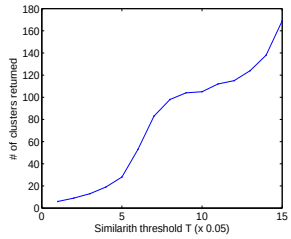


Fig. 2: The number of returned clusters with respect to the similarity threshold T setting.

S-means was compared in execution time with standard K-means and fast K-means with triangle inequality acceleration [11]. The source codes for standard K-means and fast K-means in Matlab are provided by the authors' of fast K-means. For K-means, we let K vary from 1 to 20 on the SCT dataset. For S-means, we let the up-bound number of clusters varied from 1 to 20. In this way, the comparison in execution time is fair. The results are plotted in Fig. 3. S-means has the same level of speed with fast the K-means when K beyond 5. S-means has a peak execution time when K=4. The cause is that S-means has to force itself to converge when the clusters reaches up-bound. S-means is fast because: i) in every iteration only one new cluster is added and ii) as the total number of clusters increase, a large portion of old cluster centroids do not move (already converged), thus, there is no necessity to recalculate distances from all the points to those converged centroids.

3.2 Clustering Accuracy

One of the major concerns for new algorithm is whether it is accurate. Classification accuracy usually depends on both (dis)similarity measure and classification strategy. To

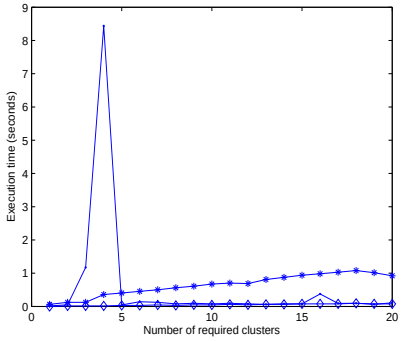


Fig. 3: Execution time comparison between S-means (dot point), standard K-means (star point and fast K-means (diamond point).

compare S-means and K-means fairly in accuracy, we still used the R-squared measure in S-means and equivalent Euclidean distance in K-means. The benchmark dataset used was Pendigit [5], which has been widely used in evaluating classification algorithms. It has 10 classes (digit 0 to 9). Total training samples is 7494 and test samples 3498. Each sample has 16 attributes (x-, y- coordinates). We concatenated x-, y- coordinates and made each sample a 32-length vector. All the vectors were normalized by mean-variance before input for clustering.

First, accuracy was compared in clustering the training dataset by K-means, G-means and S-means respectively. Suppose we don't know how many classes in Pendigit. For K-means, the necessary step is to guess K=1 up through to some up-bound P (P was set 20 in our experiments). In G-means, a confidence threshold is needed in place of K. The default confidence is set 0.001 in the original G-means package [13]. We let confidence vary from 0.0001 to 0.002 by step 0.0005. We recorded the accuracy of clustering results against the ground-truth labels in each step. For S-means, the necessary step is to vary T if no prior knowledge about the number of clusters. T was varied from 0 to 0.95 by step 0.05 and the up-bound number of clusters P was set 20, same as K-means.

Calculating clustering accuracy is a tricky task. Suppose K cluster are returned and the cluster assignment is $L = [l_1, l_2, \cdots, l_N]$, $1 \leq l_i \leq K$. And suppose the ground-truth number of clusters is K_t and the true labels are $Lt = [lt_1, lt_2, \cdots, lt_N]$, $1 \leq lt_i \leq K_t$. Note that K does not necessarily equal K_t. We used the following pseudo code to calculate accuracy:

for $i = 1$ to K **do**
  for $j = 1$ to K_t **do**
    $Common(i, j) =$
    $NumberOfCommonMemebers(find(L == i), find(Lt == j))$;

4: **end for**
5: **end for**
6: count=0;
7: **for** $i = 1$ to K_t **do**
8: $count = count + max(Common(:, i))$;
9: **end for**
10: $Accuracy = count/N$;

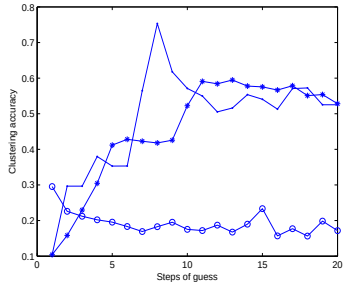


Fig. 4: Accuracy comparison between S-means (dot point), fast K-means (star point) and G-means (circle point). K varies from 1 to 20 in K-means. Confidence varies from 0.0001 to 0.002 in G-means. T varies from 0 to 0.95 with step 0.05 and up-bound of clusters is set 20 in S-means.

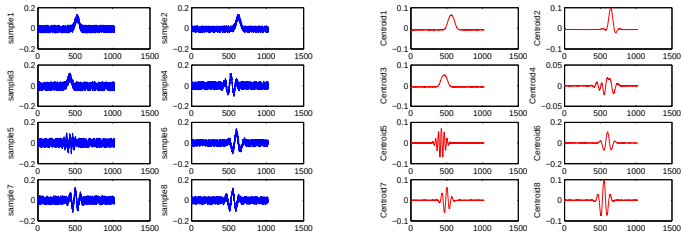
Line 3 computes the maximum common number of members between two clusters. The nested *for* loops (line 1-5) find the best matching between two sets of clusters. Line 7-9 sums up all the number of common members (which are correctly assigned). Fig. 4 shows the clustering accuracy with S-means and K-means on each step of guess. In step 9, S-means reaches its peak accuracy (when T=0.40). K-means reaches its peak accuracy when K=10. S-means's peak accuracy is significantly high than K-means's peak accuracy. After step 10, both algorithms decreases in accuracy, which is because the number of returned clusters run farther away from the true number of classes. G-means shows poor performance due to the weak assumption of Gaussian distribution. According to our experience, the real data's statistical distribution is usually not as simple as Gaussian.

3.3 Mining Gravitational-wave Astronomy time series

The detection of Gravitational waves is the next frontier in astronomy. Several large scale detectors have been constructed around the world, such as the Laser Interferometric Gravitaitonal wave Observatory (LIGO) in the U.S. [1]. These detectors are part of a world wide network that is collecting data at the rate of several Tb per week. Mining gravitational wave data for useful information is a daunting task and one of the major

challenges in the area of astronomical data analysis. The following simulation is an example of a typical clustering task that arises in such an analysis and also demonstrates the application of the S-means.

A dataset consisting of 20020 time series with length 1024 were artificially generated. Each sequence is first generated by a single Gaussian modulated sinusoid signal. The amplitude is scaled such that the matched filtering signal to noise ratio (SNR) is 1 in white Gaussian noise with zero mean and unit variance. Then, a single Gaussian pulse is added to the signal in random position. The pulse amplitude is also scaled with SNR=1 in white Gaussian noise. Fig. 5a shows the typical shape of the each cluster. The simulated time series is a close representation of the actual triggers in Gravitational-wave Astronomy time series.



(a) (b)

Fig. 5: (a) Samples of simulated Gravitational-wave time series. (b)Centroids mined by S-means when $T = 0.1$.

Although Gaussian process is used here, the possible clusters inside the dataset does not follow Gaussian distribution. As discussed in the introduction, GW events dot not follow any known statistical distribution. As a part of GW data analysis, clustering time series based on shapes (which can be matched by similarity measures) is a reliable method.

Same as experiments above, the simple similarity measure R-squared was used in S-means. Since the number of clusters in the dataset is unknown, our goal is to mine how many compact clusters exist. With high similarity threshold T, it is expectable that many time series will be stand-alone clusters. Therefore, we started with low T. Initially, we set T=0.1. We found that S-means converges to 8 clusters in less than 50 iterations. The centroids are as shown in Fig. 5b. The convergence was completed in 34.1 seconds. The average similarity of each item to its corresponding centroid was also converged to about 0.55, as shown in Fig. 6. Note that the average similarity times the number of items is equivalent to the objective function in equation (1). This means, S-means converges via a hill-climbing approach to minimize the object function and at the same time discover the number of clusters according to the similarity requirement.

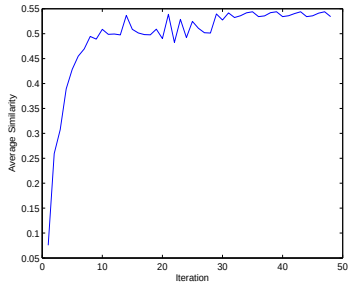


Fig. 6: S-means converges to 8 clusters in 50 interations when $T = 0.1$.

Then, we varied T from 0.2 to 0.6 by step 0.1 and recorded the number of clusters, iteration, execution time and average similarity on each step. The results are shown in Table 1 The number of clusters increases dramatically when $T = 0.4$ to unbound 1000. Correspondingly, and execution time and number of iterations changes sharply at step 4, while the average similarity increases steadily. So, we can see it is not worth to set $T \geq 0.4$ in mining this dataset. We can conclude that the simulated dataset has 8 to 16 compact clusters, depending on parameter T.

Table 1: Number of clusters, iterations, execution time and average similarity change as similarity threshold T increases.

Threshold T	0.1	0.2	0.3	0.4	0.5	0.6
Number of clusters	8	11	15	1000+	1000+	1000+
Number of iterations	48	29	20	1008	1003	1001
Execution time (secs)	34.1	23.1	23.3	1673.8	1221.8	1064.5
Average similarity	0.563	0.575	0.612	0.643	0.663	0.675

4 Conclusions and future work

S-means eliminates the necessity of specifying K (the number of clusters) in K-means clustering. An intuitive argument, similarity threshold T is used instead of K in S-means. Experiments demonstrates the efficiency and effectiveness of S-means in comparison with standard K-means, fast K-means and G-means. S-means mines the number of compact clusters in a given dataset without prior knowledge in its statistical distribution. We applied S-means to simulated Gravitations-wave time series analysis and discovered the existence of compact clusters.

While we believe S-means is promising in its simplicity, efficiency and effectiveness, we are aware that more extensive comparative experiments are needed to further

validate the algorithm with other clustering algorithms. For instance, the clustering result is very sensitive to threshold T and the number of returned clusters can be unexpectedly large when T is high (e.g, $T > 0.4$). Also, it is necessary to evaluate S-means with different similarity measures such as Dynamic Time Warping and kernel functions in our future work.

References

1. B. Abbott and et al. (LIGO Scientific Collaboration). Search for gravitational waves from binary black hole inspirals in ligo data. *Physics Review*, 73, 062001, 2006.
2. A. Abramovici and et al. LIGO: The laser interferometer gravitational wave observatory. *Science*, 256:325–333, 1992.
3. H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.
4. R. J. Alcock and Y. Manolopoulos. Time-series similarity queries employing a feature-based approach. In *Proceedings of the 7th Hellenic Conference on Informatics*, 1999.
5. A. Asuncion and D. Newman. UCI machine learning repository, http://www.ics.uci.edu/∼mlearn/mlrepository.html, 2007.
6. G. Casella and R. Berger. *Statistical Inference*. Duxbury Press, 2001.
7. R. D'andrade. U-Statistic hierarchical clustering, 1978.
8. A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(1, pages =).
9. C. Ding, X. He, H. Zha, and H. Simon. Adaptive dimension reduction for clustering high dimensional data. In *Proceedings of the 2nd IEEE International Conference on Data Mining*, pages 107–114, 2002.
10. S. G. Djorgovski, C. Donalek, A. Mahabal, R. Williams, A. Drake, M. Graham, and E. Glikman. Some pattern recognition challenges in data-intensive astronomy. In *the 18th International Conference on Pattern Recognition (ICPR 2006)*, page 856, 2006.
11. C. Elkan. Using the triangle inequality to accelerate kmeans. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 147–153, 2003.
12. U. M. Fayyad, C. Reina, and P. S. Bradley. Initialization of iterative refinement clustering algorithms. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, pages 194–198, 1998.
13. G. Hamerly and C. Elkan. Learning the k in k-means. In *Advances in Neural Information Processing Systems*, volume 17, 2003.
14. S. Haykin. *Neural Networks: A Comprehensive Foundation (2nd Edition)*. Prentice Hall, 1998.
15. A. Jain, M. Murty, and P. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3), September 1999.
16. K. Kalpakis, D. Gada, and V. Puttagunta. Distance measures for effective clustering of arima time-series. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 273–280, 2001.
17. H. Lei, S. Palla, and V. Govindaraju. ER2: An intuitive similarity measure for on-line signature verification. In *the 9th International Workshop on Frontiers in Handwriting Recognition*, pages 191–195, 2004.
18. S. Mukherjee. Multidimensional classification from kleine welle triggers from ligo science run. *Classical Quantum Gravity*, 23(S661-71), 2006.
19. D. Pelleg and A. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the 17th International Conference on Machine Learning*, pages 727–734, 2000.
20. J. Rissanen. Modeling by the shortest data description. *Automatica*, 14:465–471, 1978.
21. S. Shapiro and M. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3 and 4):591–611, 1965.
22. W. Sheng and X. Liu. A hybrid algorithm for k-medoid clustering of large data sets. In *IEEE Congress On Evolutionary Computation*, volume 1, pages 77–82, 2004.

Quasi-Incremental Bayesian Classifier

Murilo Lacerda Yoshida[1] and Estevam R. Hruschka Jr.[1]

[1] DC/UFSCar, Universidade Federal de São Carlos, São Carlos, Brazil
{murilo_yoshida, estevam}@dc.ufscar.br

Abstract. This paper describes and empirically evaluates a Quasi-Incremental Bayesian Classifier (QBC) designed to be used when a classification task must be performed in dynamic systems such as sensor networks, which are continuously receiving new piece of information to be stored in huge databases. Therefore, the knowledge that needs to be extracted from these databases is continuously evolving and the learning process may need to go on almost indefinitely. The induction proposed by QBC is performed in two steps; in the first one a traditional Bayesian Network (BN) induction algorithm is performed using an initial amount of data. As far as new data is available, only the numerical parameters of the classifier are updated. The conducted experiments showed that QBC tends to maintain the average correct classification rates obtained with non-incremental classifiers while decreasing the time needed to induce the classifier.

Keywords: Bayesian Networks, Bayesian Classifiers, Incremental Learning.

1 Introduction

Data Mining, also called KDD (Knowledge Discovery from Databases), is an important research area and its main objective is to study, investigate, propose and implement techniques, methodologies and algorithms to extract knowledge from great amounts of data.

One of the biggest challenges in KDD is to cope with huge datasets. These datasets are common in many real application domains as e-commerce and financial market. In addition, the most recent advances in miniaturization and sensor technology lead to sensor networks, gathering spatio-temporal data about the environment and revealing a new area that needs to deal with massive datasets. In such domains, thousands of measurements are done every day, thus the amount of information to be stored in databases is huge and continuously growing. Therefore, the knowledge extracted from these databases need to be continuously updated, otherwise it may become obsolete or incorrect.

The main problem of using traditional (non-incremental) learning algorithms (used to extract knowledge from databases in a KDD task) with these huge and continuously growing datasets, is the high computational effort needed. Therefore, it is not feasible to continuously execute them and an alternative approach must be used.

Incremental learning algorithms [23] represent a suitable approach to overcome the aforementioned problem [5]. The main idea behind these algorithms is to induce dynamic models that can be updated as far as new data is aggregated into a dataset. In this sense, the computational effort can be reduced and the knowledge extracted from the databases can be constantly updated.

Although there are some incremental algorithms proposed in the literature, this class of algorithm is not yet consolidated. In this paper, a Quasi-Incremental Bayesian Classifiers, named QBC, is proposed. The induction proposed by QBC is performed in two steps; in the first one a traditional Bayesian Network (BN) induction algorithm is performed using an initial amount of data and generating a Bayesian Classifier (BC). As far as new data is available, only the numerical parameters of the classifier are updated. Considering that a BC is formed by a graph structure and numerical parameters and, in addition, learning the graph structure demands a higher computational effort [18], QBC minimizes the use of a BN structure learning algorithm (applying it only once) and explores the numerical parameters to update the classifier. This idea is based on some previous work on BCs [2][16][25]; QBC, however, proposes some innovation to the traditional algorithms aiming at becoming a method suitable to real problems applications.

The remainder of this paper is organized as follows. The next section focuses on incremental learning, Bayesian Networks and related works found in the literature. Section 3 describes our Quasi-Incremental Bayesian Classifier (QBC), which will be evaluated in classification problems and the achieved results compared with those obtained using a traditional non-incremental learning algorithm. Section 4 reports our simulation results in four datasets that are benchmarks for data mining methods. Finally, Section 5 describes the conclusions and points out some future work.

2 Incremental Learning, Bayesian Classifiers and Related Work

The quasi-incremental approach proposed in this paper explores some incremental algorithms constraints and Bayesian Network Classifiers foundations. Therefore, this section reviews some basic concept of incremental learning and Bayesian Networks and Bayesian classifiers related works.

2.1 Incremental Learning

Some authors consider self-adaptation as a prerequisite for general intelligence [19]. Following along this line, the learning process should involve the ability to improve performance over time. In addition, it is known that humans acquire knowledge in an incremental fashion over time. These can be considered as the initial main motivations for developing incremental learning algorithms [8].

The recent and fast development of areas such as e-commerce, databases, electronic sensors and ubiquitous computation generated a new motivation for incremental learning algorithms investigation [24]. These technologies allow dynamic systems to be designed and employed in real world applications. Such dynamic

system are continuously receiving new piece of information to be stored in huge databases. Therefore, the knowledge present in the databases is continuously evolving and the learning process may need to go on almost indefinitely, thus a non-incremental learning algorithm may become ineffective. Most machine learning algorithm, however, are not incremental.

Incremental learning algorithms have been investigated focusing on many specific tasks. In [8] was proposed one of the first incremental learning algorithms and it was devoted to clustering tasks. In [11] an overview of incremental clustering algorithms developed in the 80's is presented and such algorithms can be considered precursors of incremental learning in machine learning and data mining.

Following the definition given in [15], an incremental learning algorithm should be able to use its learned knowledge to carry out its performance task at any stage of learning. It should also be computationally efficient when incorporating experience (training data) into memory during the learning procedure; and should not use unreasonable space (memory) to store its experience (already used training dataset).

An alternative definition can be found in [6][5]. In these works, the authors focus on learning from dynamic datasets that keep growing continuously. In this sense, to be considered incremental, a learning algorithm [5]:

- "must require small constant time per record, otherwise it will inevitably fall behind the data, sooner or later.
- must use only a fixed amount of main memory, irrespective of the total number of records it has seen.
- must be able to build a model using at most one scan of the data, since it may not have time to revisit old records, and the data may not even all be available in secondary storage at a future point in time.
- must make a usable model available at any point in time, as opposed to only when it is done processing the data, since it may never be done processing.
- Ideally, it should produce a model that is equivalent (or nearly identical) to the one that would be obtained by the corresponding ordinary database mining algorithm, operating without the above constraints.
- When the data-generating phenomenon is changing over time (i.e., when concept drift is present), the model at any time should be up-to-date, but also include all information from the past that has not become outdated."

The literature also presents other different definitions of incremental learning. In spite of this, instead of reviewing all these alternatives (to identify the most appropriate one to develop our approach), we propose to use a Quasi-Incremental algorithm which follows some of the aforementioned constraints, but do not need to be strictly in accordance with any incremental learning formal definition. Following this idea, some classical learning algorithms, such as k-NN [16] and naïve Bayes classifier [9] can be considered as having incremental learning features and based on this fact we developed our approach presented in section 3.

2.2 Bayesian Networks and Bayesian Classifiers Related work

Bayesian Networks (BNs) are graphical representations of multivariate joint probability distributions. They are described by directed acyclic graphs in which the nodes represent the attributes and the arcs represent probabilistic dependencies between connected nodes (attributes). The strength of each dependency is given by the conditional probability $P(x_i|\pi_{x_i})$, where x_i and π_{x_i} are the i-th attribute and the set of parents of x_i in the graph, respectively. The use of conditional independence is the key to the ability of BNs to provide a general-purpose compact representation for complex probability distributions [18][21].

BNs can be built directly from domain knowledge or they can be automatically learned from data. It is also possible to combine both strategies. Learning BNs from data became an effervescent research topic in the last decade, and there are two main classes of methods to perform this task [18]: methods based on heuristic search and methods based on conditional independence tests. Our work is based on the classic K2 algorithm [3], which uses a heuristic search to learn a Bayesian network from data.

Bayesian networks incremental learning is not a mature research field, hence, there are not many different approaches dealing with this theme. Nevertheless, there are some very relevant works that can be divided into two main groups, the first one considers the BN structure update and the second one considers the numerical parameters update.

When concerning the BN structure update some relevant ideas were presented in [14][10][24]. Nevertheless, considering that the method proposed in our work focuses on the numerical parameters update, the BN structure update approaches will not be further discussed in this paper.

Considering the numerical parameters update, in [2][16] and [25], the authors define the BN structure based on human expert knowledge and then, only the numerical parameters are updated in order to incorporate the new information in the knowledge represented by the BN. Such numerical parameters update may be done using one of the three different techniques: discretization of parameters, Dirichlet distributions, and Gaussian distributions. Still concerning only on the numerical parameters update, another relevant work is the one presented in [4] in which is assumed that the parameter distribution is given by a product of Gaussian distributions. The aforementioned approaches were designed to be used mainly in situations where the instances contained in the database are incomplete, the BN structure can be obtained from an external font (e.g. a human expert) and the form of the variables probability distributions are known (or can be estimated using *prior* knowledge). Under these circumstances, the proposed methods are good options when trying to perform an incremental learning of Bayesian Networks. Our proposed Quasi-Incremental Bayesian Classifier, on the other hand, is designed to be applied in situations where these assumptions do not hold.

3 Quasi-Incremental Bayesian Classifier - QBC

It is already known that when learning a BN from data, the correctness of the learned network (structure and numerical parameters) depends on the amount of training data available. Thus, when the training data is not large enough, the literature recommends employing prior knowledge about the domain to improve the accuracy of learned models [20]. Based on this understanding, some incremental BN learning algorithms require a domain expert to fully specify the network structure of the BN and focus only in the numerical parameter learning. Therefore, the network structure specified by the expert domain is used as a set of constraints to learn the conditional probability tables (CTPs) that rule the BN.

In real world applications, however, it is not common to have an expert domain at hand. In addition, as mentioned before, in domains such as sensor networks, the lack of data to be used in a learning task is not frequent. Therefore, the QBC approach considers that the initial amount of data available to be used in the learning task is large enough to induce a consistent network structure to be used in a classification task. Accordingly, the QBC initially uses a BN learning algorithm to induce a first model to be used as a classifier (as in a non-incremental procedure). As long as new data arise, only the numerical parameters are updated. Such an update is based on the previous induced BN structure. Considering that learning the BN structure needs more computational effort than learning the numerical parameters (when knowing the BN structure), the main idea is to minimize the need of inducing the entire BN every time that new data is available.

Another interesting feature present in the QBC is the numerical parameters induction approach. Instead of looking for an adequate probability distribution to model the variables behavior, as done in other proposals [2][16][25][4], our approach uses the dataset to estimate the relative frequency of the variables and thus, build the CPTs. It is important to state that we are not claiming that the frequentist approach is always better than the one pointed in the aforementioned papers. The frequentist approach, however, is simpler to implement and very adequate when no prior knowledge about the variables behavior is known, and because of its simplicity, it is commonly employed in data mining tasks. In addition, as showed in [26], assuming a variable distribution (e.g. gaussian) without being certain may implicate in bad results. On the other hand, when having a sufficiently large dataset, the relative frequency can be used to estimate the probability distribution [13].

Figure 2 presents the QCB in an algorithmic fashion. In this figure, it is shown that QBC receives a dataset D as input and, as output it produces a Bayesian Classifier BC and a data structure NP that represents the classifier numerical parameters.

The procedure `Learn_Structure` (in Figure 2) was implemented for this work using the K2 structure learning algorithm (described in [3]). This procedure, however, can be implemented using other approaches described in the literature [18]. The line 3 in the algorithm for example, can be omitted from the QBC when intending to use a naïve Bayes classifier [9] in the classification task. In spite of minimizing the computational effort in the first run of QBC, the use of a naïve Bayes classifier is not suitable in situations where the classification results are used for decision-making, for

example. It happens, mainly because precise estimates of class probabilities are crucial for decision-making. As stated in [12], a classifier is often only one part of a larger decision process, for which accurate class probability estimates provide additional utility. For instance, knowing the class probability may give much information about costs of incorrect predictions [22]. For this reason, QBC uses as classifier an unrestricted Bayesian Network (that gives more precise probability estimates) instead of a simpler Bayesian Classifier.

```
QBC Algorithm
Input:      D:  dataset
Output:     BC: Bayesian Classifier
            NP: updated Numerical Parameters
 1. Begin
 2.   If it is the first run then
 3.         Learn_Structure(D,Structure);
 4.   else
 5.         Load(NP);
 6.   end {if}
 7.   Learn_num_parameters(D,NP);
 8.   Build_Classifier(Structure,NP,BC);
 9.   Store(NP);
10.   Return(BC);
11. End.
```

Figure 2. QBC Algorithm

The first run of QBC must be executed when having a considerable amount of data. Doing so, in the first run the BN structure will be built (`Learn_Structure` procedure in Figure 2) using a fair sampling of the probability distribution that governs the variables domain. Subsequently, as far as new data arrives, QBC must be executed again. Considering that the BN structure is already built, from this point and on, only the numerical parameters will be learned and updated.

The procedure `Learn_num_parameters` receives as input a dataset D and the previous numerical parameters information (stored in a specific data structure NP), and updates NP. In the first run of the algorithm, NP is an empty input. The data structure used to store the learnt numerical parameters can be an AD-Tree [17], thus, only the sufficient statistics is stored. An AD-tree can be seen as a sparse data structure used to store counts of the records from datasets and its time performance is independent of the number of records in the dataset [24]. Therefore it is suitable in domains like sensor networks data streams learning.

The `Build_Classifier` procedure has as inputs a BN structure (previously built in line 3) and the data sufficient statistics stored in NP. Thus, it merges these inputs creating a complete BN having the structure and its corresponding numerical parameters which will be used as a classifier (BC).

As mentioned before, QBC is designed to be used in domains like sensor networks data streams learning. In such domains, it is worth having algorithms able to update the knowledge stored in a classifier in the light of new data instances using a reasonable computing time and memory space. In addition, as done in most incremental learning approaches [24], QBC is based on the assumption that all data in the stream are sampled from the same probability distribution, therefore, QBC does not need to handle concept drift as the underlying domain probability distribution does not change over time.

4 Experiments

Trying to verify the soundness of the proposed QBC approach, when compared to a non-incremental algorithm, a number of empirical classification experiments were conducted using the K2 algorithm as a non-incremental approach. The main aspects to be considered when concerning the QBC behavior are twofold: the Average Correct Classification Rates (ACCRs) and the time needed to build the classifier. The remaining of this section initially describes the knowledge domains used in the experiments as well as the experimental methodology adopted. The results from the experiments are then presented and analyzed.

Four well-known Bayesian Network domains, namely Alarm [1], Asia [7], Credit [7] and Engine fuel system [7] were used in our experiments. The description of each domain can be obtained in [7]. Table 1 summarizes datasets characteristics.

The main motivation of using domains described by known Bayesian Networks is the possibility to generated new data (using a sampling strategy) whenever it is necessary. In addition, it is possible to identify a suitable variable ordering to each domain.

Table 1. Datasets Description with dataset name (Data), number of attributes plus class (#Attributes), number of instances (#Instances) and number of classes (#Classes).

Data	Alarm	Asia	Credit	Fuel Engine
#Attributes	37	8	12	9
#Instances	30000	15000	15000	15000
#Classes	2	2	2	2

The experiments were conducted considering three scenarios for each domain. The main idea is to picture 3 different situations where the initial dataset (used to construct the complete BN: structure + numerical parameters) has different sizes. In the first scenario, the initial dataset can be considered as a small one and it keeps growing as far as new scenarios are created. Therefore, in the third scenario, the initial dataset is larger than the ones defined in the first and second scenarios. Table 2 shows the initial dataset size for each scenario considering each domain. The second and third scenarios are different in the Alarm domain (when compared to the other domains) because it is the largest domain in our experiments.

Table 2. Datasets Description with dataset name (Data), number of attributes plus class (#Attributes), number of instances (#Instances) and number of classes (#Classes).

	Alarm	Asia	Credit	Engine
1st scenario	1000	1000	1000	1000
2nd scenario	5000	3000	3000	3000
3rd scenario	10000	5000	5000	5000

Considering the Alarm domain, in the first scenario, after executing the first QBC run (with the initial dataset), 500 new instances were added to the experiment and the QBC was executed again. It was done 38 times, thus QBC was executed once using only the initial dataset (1000 instances) and then, executed other 38 times simulating new data arrivals. The last QBC execution in this scenario used a dataset containing 20000 instances.

For the second scenario, as the initial dataset contains 5000 instances, the QBC was executed 30 times to simulate the new data (500 instances) arrivals. Thus, in the last execution, the dataset was formed by 20000 instances. The third scenario followed the same strategy and, thus, QBC was executed once with the original dataset (10000 instances) and more 20 times with new data (500 instances) arrivals.

For each scenario, the process was repeated 10 times and Table 3 shows the obtained results. The time and ACCRs presented in Table 3 are the average values obtained after the 10 executions. To calculate the correct classification rates 10000 new instances (not used in the QBC executions) were used.

Analyzing results for the Alarm domain, it is possible to verify that the ACCRs obtained using QBC are close to the ones obtained with K2. On the other hand, the time needed to execute the non-incremental algorithm (every time that new data is available) is considerably lower with QBC. On average, QBC spent less than 35% of the time spent by K2.

One interesting aspect to be highlighted is that in the first scenario the QBC was faster than in the following ones. Considering that in the third scenario QBC was executed only 21 times, while in the first scenario it was executes 39 times, this result may seem counterintuitive. Observing, however, that in the third scenario the initial dataset has ten thousand instances while in the first scenario it has only one thousand, it is possible to understand such behavior. It means that the time spent in the first QBC run is higher in the third scenario than in the first one. In other words, the size of the initial dataset used in the first QBC run (when the BN structure is induced) has more influence in the average time showed in Table 3 than the number of times that QBC is executed.

For the next three domains, the same strategy employed to the Alarm domain was also used. The only difference is the size of the initial datasets which are smaller in Asia, Credit and Engine domains. Results presented in Table 3 show that the ACCRs obtained with both QBC and K2 are similar in all the three domains. When concerning the time needed by both algorithms, on average, QBC spent less than 30% of the time spent by K2. These results confirm the same behavior present in the Alarm domain.

As happened with the Alarm domain, in the experiments with the Credit domain,

the size of the initial dataset (used in the first QBC run) had more influence in the average run time than the number of times that QBC was executed. The same fact did not occur with Asia and Engine domains. Considering that Asia and Engine are smaller domains (8 and 9 variables respectively), such behavior was already expected. In this sense, the conducted experiments showed that, when concerning runtime obtained with employed datasets, large domains (having many variables) with large datasets tends to take more advantage of QBC characteristics.

Table 3. Results with Alarm, Asia, Credit and Fuel Engine domains. Where Time ± SD: time and standard deviation; ACCR: Average Correct Classification Rates; #exec: number of runs.

Alarm						
Scenario	Incremental – QBC			Non-incremental – K2		
	Time ± SD	ACCR	#exec	Time ± SD	ACCR	#exec
1st	31.7 ± 3.9	91.2 ± 8.3	39	2,566.0±30.9	93.9 ± 6.1	39
2nd	58.4 ± 0.7	85.6 ± 9.1	31	2,374.6±25.5	91.2 ± 8.4	31
3rd	112.2 ± 12.9	94.7 ± 6.1	21	2,075.7±294.0	95.3 ± 5.1	21
Asia						
Scenario	Incremental – QBC			Non-incremental – K2		
	Time ± SD	ACCR	#exec	Time ± SD	ACCR	#exec
1st	1.3 ± 0.6	99.9 ± 0.0	18	5.3 ± 0.9	99.9 ± 0,0	18
2nd	1.7 ± 0.6	99.9 ± 0.0	14	5.0 ± 0.7	99.9 ± 0,0	14
3rd	1.3 ± 0.6	100.0 ± 0.0	10	4.0 ± 0.8	100.0 ± 0,0	10
Credit						
Scenario	Incremental – QBC			Non-incremental – K2		
	Time ± SD	ACCR	#exec	Time ± SD	ACCR	#exec
1st	2.7 ± 0.6	69.3 ± 3.3	18	27.6 ± 2.0	64.2 ± 4.5	18
2nd	2.6 ± 0.3	70.7 ± 5.1	14	26.3 ± 0.9	65.1 ± 2.7	14
3rd	3.2 ± 0.6	64.4 ± 3.9	10	22.0 ± 0.5	64.7 ± 1.9	10
Engine						
Scenario	Incremental – QBC			Non-incremental – K2		
	Time ± SD	ACCR	#exec	Time ± SD	ACCR	#exec
1st	1.5 ± 0.5	99.5 ± 0.0	18	5.4 ± 0.4	99.5 ± 0.0	18
2nd	1.4 ± 0.6	99.5 ± 0.0	14	4.6 ± 0.4	99.5 ± 0.0	14
3rd	1.4 ± 0.4	99.6 ± 0.1	10	4.2 ± 0.6	99.6 ± 0.1	10

5 Conclusions and Future Work

In this paper we proposed a Quasi-Incremental Bayesian Classifier (QBC) designed to induce classifiers in dynamic systems such as sensor networks. QBC is a tow-fold approach, at first a BC is induced using an initial amount of data. Subsequently, as far as new data is available, only the numerical parameters of the classifier are updated.

The conducted experiments showed that QBC tends to maintain the ACCRs obtained with a non-incremental classifier while decreasing the induction time. In addition, large domains (having many variables) with large datasets tends to take more advantage of QBC characteristics than small ones. We intend next to investigate the appropriateness of other non-incremental BN learning as well as the use of simpler classifiers induction algorithms in conjunction to QBC.

Acknowledgments

Authors acknowledge the Brazilian research agency FAPESP for its financial support.

References

[1] Beinlinch, I.; Suermoundt, G.; Chavez, R.; Cooper, G. The ALARM monitoring system. In *Proceedings of the European Conference on AI and medicine*, 1989.
[2] Buntine, W. Operations for learning with graphical models. Journal of artificial intelligence research, 2: 159-225, 1994.
[3] Cooper, G.; Herskovits, E. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309-347, 1992.
[4] Díez, F. J., Parameter adjustment in Bayes networks. The generalized noisy OR-gate. In Proc. of the 9th Conf. on UAI, pages 99{105, Washington D.C., 1993. Morgan Kaufmann, San Mateo, CA.
[5] Domingos, P.; Hulten, G. A general framework for mining massive data streams. *Journal of Computational and Graphical Statistics*, 12, 2003.
[6] Domingos, P.; Hulten, G. Catching up with the data: Research issues in mining data streams. In *Workshop on Research Issues in Data Mining and Knowledge Discovery*, Santa Barbara, CA, 2001.
[7] Druzdel, M.J. SMILE: Structural Modeling, Inference and Learning Engine and GeNIe: A development environment for graphical decision-theoretic models. In Proc. AAAI-99, 902-903, 1999.
[8] Fisher, D.H. "Knowledge acquisition via incremental conceptual clustering". *Machine Learning*, 2:139-172, 1987.
[9] Friedman, N.; Geiger, D.; Goldszmidt, M. Bayesian network classifiers. Machine Learning, 29:131–163, 1997.
[10] Friedman, N.; Goldszmidt, M. Sequential update of Bayesian network structure. Proc. 13th UAI, 1997.
[11] Gennari, J.H.; Langley, P.; Fisher, D. Models of incremental concept formation. *Artificial Intelligence*, 40:11-61, 1989.
[12] Grossman, D. and Domingos, P., Learning Bayesian Networks Classifiers by Maximizing Conditional Likelihood. Proceedings of 21st ICML, Canada, 2004.
[13] Hays, W., Statistics. Wadsworth Publishing, 5 edition, 1994.
[14] Lam, W.; Bacchus, F. Using new data to refine Bayesian networks. In Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence, 383-390, 1994.
[15] Langley, P. Order effects in incremental learning. In *Learning in humans and machines: Towards an Interdisciplinary Learning Science*, Pergamon, 1995.
[16] Lauritzen, S.L. The EM algorithm for graphical association models with missing data. Computational Statistics and Data Analysis, 19:191-201, 1995.
[17] Moore, A. W. and Lee, M. S., Cached sufficient statistics for efficient machine learning with large datasets. Journal of Artificial Intelligence Research, 8:67-91, 1998.
[18] Neapolitan, R.E. *Learning Bayesian Networks*. Prentice Hall, Upper Saddle River, NJ, 2003.
[19] Newell, A. and Simon, H., Computer science as empirical enquiry: Symbols and search, Communications of the ACM 19(3), (1976), 113–126.
[20] Niculescu, R. S., Mitchell, T. M. and Rao, R. B., Bayesian Network Learning with Parameter Constraints 7(Jul):1357--1383, 2006.
[21] Pearl, J., "Probabilistic Reasoning in Intelligent Systems". Morgan Kaufmann, 1988.
[22] Provost, F. and Fawcett, T., Robust Classification for Imprecise Environments. Machine Learning, 42, 203-231, 2001.
[23] Provost, F. J. and Kolluri, V. "A survey of methods for scaling up inductive algorithms". Data Mining and Knowledge Discovery, 3(2):131-169,1999.
[24] Roure, J. Incremental methods for bayesian network structure learning. Research Report LSI-99-42-R, Software Department at the Technical University of Catalonia. 1999.
[25] Spiegelhalter, D.J.; Lauritzen, S.L. Sequential updating of conditional probabilities on directed graphical structures. Networks, 20:579 – 605, 1990.
[26] Yang, Y. and G. I. Webb, On Why Discretization Works for Naive-Bayes Classifiers. In T.D. Gedeon and L.C.C. Fung (Eds.), *LNAI Vol. 2903: Proc. of the 16th AI 03,* Perth, Australia, pp. 440-452, 2003.

PQStream: A Data Stream Architecture for Electrical Power Quality

Dilek Küçük[1], Tolga İnan[1], Burak Boyrazoğlu[1,2], Serkan Buhan[1,3], Özgül Salor[1], Işık Çadırcı[1,3], Muammer Ermiş[2]

[1] TÜBİTAK – Uzay, Power Quality Group, Ankara – Turkey
{dilek.kucuk, tolga.inan, burak.boyrazoglu, serkan.buhan, ozgul.salor}@uzay.tubitak.gov.tr
[2] METU, Electrical and Electronics Eng. Dept., Ankara – Turkey
ermis@metu.edu.tr
[3] Hacettepe University, Electrical and Electronics Eng. Dept., Ankara – Turkey
cadirci@ee.hacettepe.edu.tr

Abstract. In this paper, a data stream architecture is presented for electrical power quality (PQ) which is called PQStream. PQStream is developed to process and manage time-evolving data coming from the country-wide mobile measurements of electrical PQ parameters of the Turkish Electricity Transmission System. It is a full-fledged system with a data measurement module which carries out processing of continuous PQ data, a stream database which stores the output of the measurement module, and finally a Graphical User Interface for retrospective analysis of the PQ data stored in the stream database. The presented model is deployed and is available to PQ experts, academicians and researchers of the area. As further studies, data mining methods such as classification and clustering algorithms will be applied in order to deduce useful PQ information from this database of PQ data.

Keywords: Data Streams, Data Stream Applications, Electrical Power Quality.

1 Introduction

The proliferation of time-involving and data-intensive applications such as sensor networks, network traffic monitoring systems and financial applications led to the emergence of data stream models and issues related to the management of these models as well. Considerable research have been carried out on data streams including the studies on data stream management systems such as STREAM [1], those ones on tracking cross-correlation in data streams [2, 3], studies on mining data streams such as StreamCube [4], and finally those studies that present real world applications of data streams such as GigaScope [5]. We refer interested readers to [6, 7] for in-depth surveys of the literature on data streams.

In this paper, a data stream architecture is presented, which is called PQStream, for processing and managing electrical power quality (PQ) data. The feasibility and effectiveness of the proposed architecture is demonstrated on the PQ data obtained by a mobile PQ measurement system [8] monitoring the transformer substations of the

Turkish Electricity Transmission System where this measurement system is also part of the PQStream architecture.

The rest of the paper is organized as follows: Section 2 presents a brief review of electrical power quality and power quality parameters. In Section 3, general PQStream architecture and its main components are described in detail with its application on field PQ data. Finally, conclusions and future research directions are presented in Section 4.

2 Electrical Power Quality Parameters

Electrical power is one of the most essential items used by commerce and industry today. It is an unusual commodity because it is required as a continuous flow - it cannot be conveniently stored in quantity - and it cannot be subject to quality assurance checks before it is used [9]. The reliability of the supply must be known and the resilience of the process to variations must be understood. In reality, of course, electricity is very different from any other product – it is generated far from the point of use, is fed to the grid together with the output of many other generators and arrives at the point of use via several transformers and many kilometers of overhead and possibly underground cabling. Where the industry has been privatized, these network assets will be owned, managed and maintained by a number of different organizations. Hence assuring the quality of delivered power at the point of use is no easy task.

Consumers of electricity are being increasingly affected by PQ problems due to augmentation of disturbing loads in electric power systems. Throughout the study, PQ parameters given at IEC 61000-4-30 [10] are used. In the following subsections, basic electric power definitions and brief descriptions of some PQ parameters can be found. At the end of this section, event types such as sag, swell, unbalance and interrupt are explained as well.

2.1 Power

Electric power is defined as the amount of work done by an electric current, or the rate at which electrical energy is transferred. In alternating current circuits, energy storage elements such as inductance and capacitance may result in periodic reversals of the direction of energy flow. The portion of power flow that averaged over a complete cycle of the AC waveform, which results in net transfer of energy in one direction is known as real power. That portion of power flow due to stored energy that returns to the source in each cycle is known as reactive power. The relationship between real power, reactive power and apparent power can be expressed by representing the quantities as vectors (Fig.1). The apparent power vector is the hypotenuse of a right triangle formed by connecting the real and reactive power vectors [11].

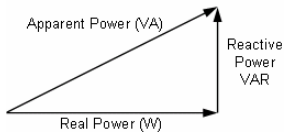


Fig. 1. Relation between Real, Reactive, and Apparent Powers

2.2 Demand

Electric power demand is directly proportional to the current demand of consumer. Hence consumer's power demand profile may obtained by sampling the current demand values. Power demand is mainly characterized by fundamental harmonic component of the load current.

2.3 Voltage and Current RMS

RMS (Root-Mean-Square) value is defined to be square root of the arithmetic mean of the squares of the instantaneous values of a quantity taken over a specified time interval. The average power consumed by a sinusoidally driven linear two-terminal electrical device is a function of the RMS values of the voltage across the terminals and the current passing through the device, and of the phase angle between the voltage and current sinusoids.

2.4 Frequency

The mains frequency is the frequency at which alternating current is transmitted from a power plant to the end user. In most parts of the world, it is typically 50 or 60 Hz. Mains frequency is fixed to 50 Hz for the Turkish Electricity Transmission System. However due to practical load demand variations, supply frequency appears to be in a frequency band rather than having a constant value. Hence, mains frequency turns out to be an important PQ parameter indicating the frequency stability of the particular utility grid.

2.5 Harmonics

Ideally, voltage and current waveforms are perfect sinusoids. However, as reported in [12], because of the increased popularity of electronic and other non-linear loads, these waveforms often become distorted. This deviation from a perfect sine wave can be represented by harmonics—sinusoidal components having a frequency that is an integral multiple of the fundamental frequency (Fig. 2). Thus, a pure voltage or current sine wave has no distortion and no harmonics, and a non-sinusoidal wave has distortion and harmonics. To quantify the distortion, the term total harmonic

distortion (THD) is used. The term expresses the distortion as a percentage of the fundamental (pure sine) of voltage and current waveforms.

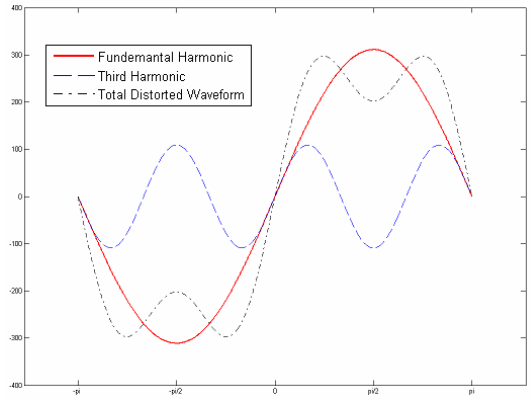


Fig. 2. Distorted Waveform Composed of Fundamental and 3rd Harmonic

2.6 Flicker

The power supply network voltage varies over time due to perturbations that occur in the processes of electricity generation, transmission and distribution. Interaction of electrical loads with the network causes further deterioration of the electrical PQ. High power loads that draw fluctuating current, such as large motor drives and arc furnaces, cause low frequency cyclic voltage variations that result in flickering of light sources which can cause significant physiological discomfort, physical and psychological tiredness, and even pathological effects for human beings. Hence, flicker is quantified based on models of light sources and human sensation [13].

2.7 Events

Voltage sag, swell, unbalance and interruption are detected as PQ events throughout the measurements. These events are briefly described below [10]:

- *Voltage Sag:* Sag indicates an under-voltage situation. On poly-phase systems, voltage sag begins when the voltage of one or more channels is below a sag threshold (%85 of nominal) and ends when voltage on all measured channels is equal to or above the sag threshold plus the hysteresis voltage.

- *Voltage Swell:* Swell indicates an over-voltage situation. On poly-phase systems, a swell begins when the voltage of one or more channel rises above the swell threshold (%110 of nominal) and ends when the voltage on all measured channels is equal to or below the swell threshold minus the hysteresis voltage.

- *Unbalance:* Channels of poly-phase systems should have sinusoidal voltages having same amplitude. Balanced three-phase system should have the same amplitude on all phases. Unbalance is a measure which indicates how much the amplitude of phases different from each other.

- *Interruption:* On poly-phase systems, a voltage interruption begins when the voltage of all channels is below the voltage interruption threshold (%5 of nominal) and ends when the voltage of any one channel is equal to or greater than the voltage interruption threshold plus the hysteresis.

3 PQStream Architecture

PQStream is an architecture offered for efficient processing and management of PQ data. PQ data has an inherent time-dependency and this data when measured at relevant frequencies requires almost unbounded storage and processing capabilities compared to other data types stored in conventional relational database management systems.

We have used PQStream architecture for mobile measurements of PQ data in the Turkish Electricity Transmission System and describe the architecture accordingly, yet it is a generic architecture and can be used for managing any PQ data acquired in other means with little or no customization.

An abstract representation of PQStream presented in Fig. 3. In the following subsections, we firstly introduce the mobile PQ measurements application which corresponds to the PQ measurement module in Fig. 3 and describe how the resulting data is transferred to the stream database. Following this, we describe the stream database with its conceptual data model and finally the PQStream GUI.

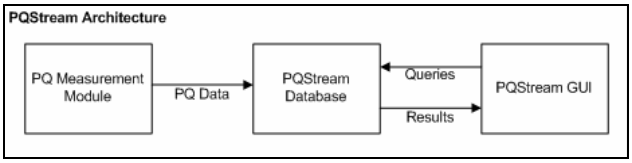


Fig. 3. Abstract Representation of PQStream Architecture

3.1 PQ Data Measurement Module

Mobile PQ measurements in the Turkish Electricity Transmission System are carried out for a period of seven consecutive days for each measurement point. Measurement points are feeders and busbars in the transformer substations. At the time this paper is written, measurements have been completed for 144 bus-bars, 205 feeders, 59 transformer substations all over the country.

Mobile measurement program is developed in LabView development environment [14] using its proprietary visual programming language called G where the sampling frequency of the program is 3200 Hz, that is, it acquires 3200 raw samples per second for each PQ parameter. The program calculates and outputs the averages corresponding to the PQ parameters according the PQ standards [10] in an online fashion. It outputs raw PQ data as well in case of events as will be clarified in the upcoming paragraphs.

Output traffic load of the PQStream measurement module is presented in Table 1 for each PQ data measurement point. Storage and processing requirements of PQStream database could be estimated using the total bit rate values in this table and number of measurement points.

Table 1. Outgoing Data Traffic of PQStream Measurement Module (Based on Mobile PQ Measurements of the Turkish Electricity Transmission System).

Parameter	Precision	Update Rate (Averaging Interval)	Three Phase	Average PQ Data Bit Rate (bps)
Active Power	Double	every second	Yes	192
Reactive Power	Double	every second	Yes	192
Apparent Power	Double	every second	Yes	192
Power Factor	Double	every second	Yes	192
33 Voltage Harmonics	Double	every 3 secs.	Yes	2.112
33 Current Harmonics	Double	every 3 secs.	Yes	2.112
RMS Current and Voltage	Double	every 0.2 secs.	Yes	1.920
Event Length	Integer	variable	No	4
Event Type	String	variable	No	10
Event Raw Current[1] Data	Double	variable	Yes	614.400
Event Raw Voltage Data	Double	variable	Yes	614.400
Short Term Flicker	Double	every 10 mins.	Yes	0,32
Demand	Double	every 15 mins.	Yes	0,213
Frequency	Double	every second	No	64
		Total (with Events)		**1.235.790,533**
		Total (without Events)		**6.990,533**

[1] 614400 bps = 3200 samples/sec*8 bytes/sample*8 bits/byte*1 sample/phase*3 phase

The output of the program is a set of directories and files for each PQ parameter where exact directory structure of this output is presented in Fig. 4.

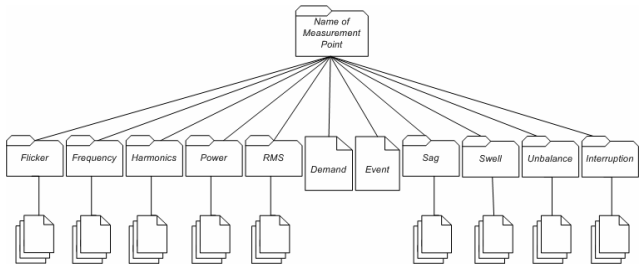


Fig. 4. Directory Structure of Mobile PQ Measurements

In order to store and manage this PQ measurement data in a stream database, a daemon program executes on the data in order to transfer them to the database. The program simply reads the measurement files for PQ parameters and inserts each sample in these files to the corresponding tables in the stream database.

However, not all values in all files in Fig. 4 are directly stored in the database due to space and processing limitations. The files under the directories of *Sag, Swell, Unbalance*, and *Interruption* include *raw data* corresponding to the actual samples during the entire period of each event and are simply stored as compressed files in the file system in a specific directory layout instead of storing their contents in the database. Only absolute paths of these files are stored in the database. *Event* measurement file in Fig. 4 has an entry for each of these events so that this information will be available through the database and if the actual *raw data* is required for an event, it will be provided to the user as a file. We refer interested readers to [8] for an in-depth description of the mobile PQ measurements application.

3.2 Data Stream Model for PQ Data

The output of the PQStream data measurement module, which corresponds to the computed averages of PQ parameters according to the averaging intervals provided in Table 1, should be effectively stored in a database for retrospective analysis of the PQ data. For this purpose, we have proposed a conceptual data model for PQ data and presented this model as a Unified Modeling Language (UML) class diagram in Fig. 5.

PQStream database is constructed by implementing each of the classes in Fig. 5 as tables of a database using open-source object relational PostgreSQL as the backend database system. These classes are briefly described below:

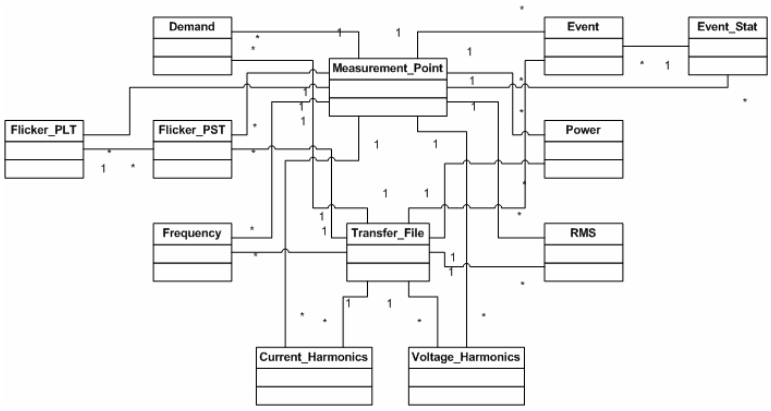


Fig. 5. Conceptual Data Model for PQ Data Represented with a UML Class Diagram

- *Measurement_Point* class holds information about the busbars or feeders (measurement points) where PQ measurements take place. The attributes of this class are crucial since they could be used to group stream data (hence will be in the *group by* clause when the required data is represented as a Structured Query Language (SQL) query or a query in one of the other languages based on SQL). Some of the most significant attributes of this class are *load_type* (which can take on one of the values of *Heavy Industry, Industry+Urban,* and *Urban Only*), *city_name, region_name,* and *voltage_level*.

- *Transfer_File* class is for holding information related to the transfer file and actual data transfer time. Since each PQ measurement sample has a corresponding timestamp; this value must also be stored in the stream database. But, frequency of each PQ parameter is determined according to PQ standards (provided as averaging intervals in Table 1), that is, duration between consecutive timestamps are known in advance, hence we use an attribute called *measurement_date* for each *Transfer_File* instance to represent the measurement time of the last sample in that file so that the timestamps for the remaining samples could be determined according to the PQ parameter type.

- *Event* class is used to model an entry for each and every event that occurred during the entire measurement period. The attributes of this class include *event_type* (one of *sag, swell, interruption* or *unbalance*), *event_starting_time*, *event_ending_time,* and *event_size_in_samples*. Although *raw data* corresponding to each event occurrence is also stored in the directory structure in Fig. 4 as a file for each event type under corresponding directories, they are not individually modeled in the

conceptual design due to space and processing limitations (the number of event occurrences is bounded only by the total measurement period) and we store these files in compressed form in a certain directory structure for each measurement point as explained at the end of Section 3.1. The absolute path of each of these raw data files is modeled with the *file_path* attribute of the *Event* class.

- *Event_Stat* is a class introduced for efficiency reasons. It is used to model a summary of the events occurred in a measurement point. The attributes of *Event_Stat* include *event_count, sag_count, swell_count, interruption_count* and *unbalance_count*. With this class implemented as a database table, most of the aggregation queries on PQStream will be faster (since they will scan *Event_Stat* table instead of the larger *Event* table).

- Among the remaining classes, *Flicker_PST* models short term, and *Fliker_PLT* models long term flicker measurements. Long term flicker (P_{lt}) is calculated from short term flicker (P_{st}) with the formula (1) taking N=12 where P_{sti} (i=1..N) are consecutive values of P_{st}. As their names imply, the classes *Demand*, *Frequency*, *RMS* and *Power* are for modeling respective PQ parameters.

$$P_{lt} = \sqrt[3]{\frac{\sum_{i=1}^{N} P_{sti}^3}{N}} \qquad \textbf{(1)}$$

3.3 PQStream Graphical User Interface (GUI)

A user-friendly interface is essential for the effective querying of the presented PQStream database. Furthermore, this interface should provide high-quality visualization facilities to its users since graphics is probably the best way to present PQ data.

For this purpose, a GUI for PQStream has been developed using Java programming language, with its Swing Application Programming Interface (API), in Eclipse development environment. The characteristics of PQStream GUI are summarized below:

- It enables its users to query each of the PQ parameters and results can be represented using different visualization options such as tables, bar/pie charts, or time-series graphics. Graphics facilities are implemented using the open-source JFreeChart API [15]. In Fig. 6, flicker panel of PQStream GUI is presented where the query provided through the GUI results in the graphical representation of short term flicker for a measurement point, namely, *EZİNE TM 154/34.5 KV TRAFO PRİMERİ*.

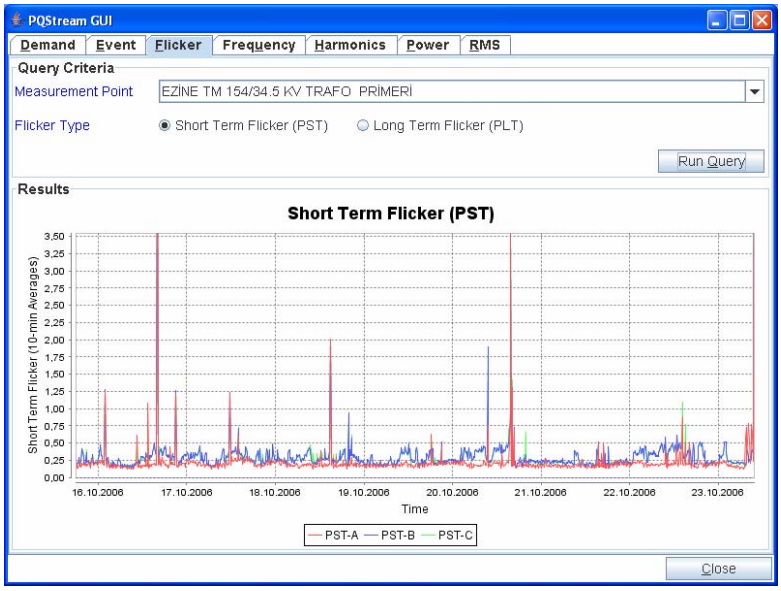


Fig. 6. PQStream GUI (with Flicker Query Panel)

- Users can provide aggregation queries through PQStream GUI so that summary information can be obtained about PQ parameters. For instance, the aggregation query provided in the Event query panel of the GUI in Fig. 7 can be represented in SQL as follows:

```
select      sum(es.sag_count), sum(es.swell_count),
            sum(es.unbalance_count),
            sum(es.event_count), mp.load_type
from        event_stat es, measurement_point mp
where       es.measurement_point_id = mp.id
group by    mp.load_type
```

- We have used Apache XML-RPC [16], Apache's Java implementation of XML-RPC protocol, for the communication between the stream database and PQStream GUI for its simplicity and compactness. Apache's Tomcat [17] web server is used to deploy server side PQStream code.

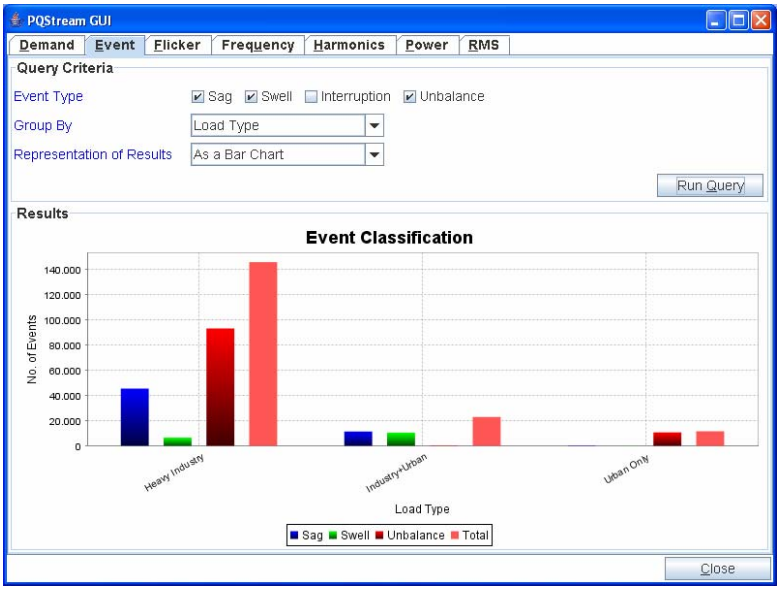


Fig. 7. PQStream GUI (with Event Query Panel)

4 Conclusion

Electrical PQ data is a time-evolving type of data and measuring it at high frequencies without any processing leads to unfeasibly large volumes requiring almost unbounded processing capabilities compared to other types of data stored in conventional relational database management systems. In this paper, we have described a data stream architecture for electrical PQ data, which, to our knowledge, is the first attempt to model PQ data as data streams, and shown its feasibility on real-world (field) PQ data.

The main modules of PQStream architecture are a measurement module which processes continuous PQ data and computes averages according PQ standards, a stream database for storing the averages that the measurement module computed and finally a GUI for retrospective analysis and visualization of the stored PQ data. PQStream chooses not to store raw PQ data to lower the storage requirements of the acquired PQ data and uses some summary relations to speed up query processing. It supports aggregation queries over PQ data and with its proprietary GUI it enables users to access summaries of PQ data with relevant visualization facilities such as bar/pie charts and time-series graphs which are typical ways of presenting PQ data.

As further studies, we will employ data mining techniques on PQStream database such as classification and clustering algorithms to group those measurement points from which PQ data are acquired as well as sequence mining techniques to see the time-evolution of PQ problems. With the results of these data mining attempts on PQStream, experts of PQ domain will be able to take the necessary measures to detect and reduce PQ problems in electricity transmission systems.

Acknowledgments. This research and technology development work is carried out as a subproject of the National Power Quality Project of Turkey (Project No. 105G129, http://www.guckalitesi.gen.tr). Authors would like to thank the Public Research Support Group (KAMAG) of the Scientific and Technological Research Council of Turkey (TÜBİTAK) for full financial support of the project.

References

1. Stanford Stream Data Management (STREAM) Project. http://www-db.stanford.edu/stream.
2. Zhu, Y. and Shasha, D. StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time. In Very Large Data Bases (VLDB), (2002).
3. Papadimitriou, S., Sun, J. and Yu, P.S. Local Correlation Tracking in Time Series. In Proceedings of the Sixth International Conference on Data Mining (ICDM), (2006).
4. Han, J., Chen, Y., Dong, G., Pei, J., Wah, B. W., Wang, J., and Cai, Y. D. StreamCube: An Architecture for Multidimensional Analysis of Data Streams. Distributed and Parallel Databases, 18(2): 173–197, (2005).
5. Cranor, C., Johnson, T., Spataschek, O., and Shkapenyuk, V. Gigascope: a stream database for network applications, In proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, June 09-12, (2003).
6. Golab, L. and Özsu, M. T. Issues in Data Stream Management, ACM SIGMOD Record, 32(2): 5–14, (2003)
7. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J. Models and Issues in Data Stream Systems, In: 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, (2002)
8. Özdemirci, E., Akkaya, Y., Boyrazoğlu, B., Buhan, S. et al. Mobile Monitoring System to Take PQ Snapshots of Turkish Electricity Transmission System. In IEEE Instrumentation and Measurement Technology Conference (IMTC), (2007)
9. Chapman, D. Introduction to Power Quality. Power Quality Application Guide, 1-4 (2001)
10. IEC 61000-4-30, Testing and Measurement Techniques - Power Quality Measurement Methods, International Electrotechnical Commission, (2003)
11. AC power - Wikipedia, http://en.wikipedia.org/wiki/AC_power
12. Harmonics, Electric Power Quality Notes, Pacific Gas and Electric Company, (1993)
13. Hanzelka, Z. and Bień, A. Flicker Measurements, Power Quality Application Guide, 1-12 (2005)
14. LabVIEW - Products and Services - National Instruments, http://www.ni.com/labview/
15. JFreeChart, http://www.jfree.org/jfreechart/
16. Apache XML-RPC, http://ws.apache.org/xmlrpc/
17. Apache Tomcat, http://tomcat.apache.org/

Resource-aware Distributed Online Data Mining for Wireless Sensor Networks

Nhan Duc Phung[1], Mohamed Medhat Gaber[2] and Uwe Roehm[1]

[1] University of Sydney, School of Information Technologies
SIT Building J12, NSW 2006, Australia
{dphu9727,roehm}@it.usyd.edu.au
[2] CSIRO ICT Centre, Tasmania, Hobart, TAS 7001
Mohamed.Gaber@csiro.au

Abstract. Online data mining in wireless sensor networks is concerned with the problem of extracting knowledge from a large continuous amount of data streams with an in-network processing mode. Unlike other types of networks, the limited computational resources require the mining algorithms to be highly efficient and compact. We propose a distributed resource-aware online data mining framework for wireless sensor networks which can be used to enable existing mining techniques to be applied to sensor network environments. We have applied the framework to develop and implement a distributed resource adaptive online clustering algorithm on the novel Sun Microsystem™ Small Programmable Object Technology Sun SPOT platform. We have evaluated the performance of the algorithm on the actual sensor nodes. Experimental results show that the clustering algorithm can improve significantly in resource utilization while maintaining acceptable accuracy level.

Keywords: distributed clustering, resource adaptivity, data mining, sensor networks

1 Introduction

Online data mining in wireless sensor networks has attracted research attention in recent years. This is because deployments of large-scaled distributed sensor networks are now possible owning to hardware advances and increasing software support. Online data mining, also called *data stream mining* is concerned with extracting patterns from continuous data streams such as those generated by sensor networks. Because of the massive amount of data and the speed of which the data are generated, many data mining applications in sensor networks require in-network processing such as aggregation to reduce sample size and the communication overhead.

Adding up to the challenges are the extremely limited size of memory, available energy and processing power of the sensor nodes. These factors imply that traditional data mining techniques in order to be used in sensor network need to be highly energy efficient and compact. One of the methods is to improve the resource utilization via enabling *resource-awareness* for the mining techniques. With resource-awareness, the mining algorithm can automatically adjust its configuration in real time according to resource

availability levels. This can prolong network lifetime and it can also improve the mining techniques performance under resource-scare scenarios. Whilst there is research work on resource adaptivity in wireless sensor network, none of them provide a generic mechanism to enable resource-awareness for data mining in sensor networks.

In this paper, we propose a distributed resource-aware online data mining framework for wireless sensor network which can be applied for many mining techniques that requires constantly monitoring, aggregation of data and in-network processing. We apply the framework to implement a distributed resource-aware online clustering algorithm, which we termed DERA-Cluster, on an actual sensor platform – the Sun SPOT. We have implemented and evaluated the algorithm on the actual sensor networks. Experimental results show that our clustering algorithm with resource-awareness greatly improves resource utilization while being able to maintain acceptable accuracy.

This paper is organised as follows. Section 2 reviews the related work in this field and Section 3 briefly discusses the background of the resource-aware framework. In Section 4, we introduce our DERA cluster algorithm, and we discuss implementation issues in Section 5. Section 6 evaluates the validity of this approach in terms of resource-awareness and accuracy. Section 7 concludes this paper.

2 Related Work

We discuss an approach to adapt mining data stream techniques to resource availability. Online data stream mining has attracted more and more research attention in recent years. Gaber et al. [3] have done an in-depth survey of mining data streams. There are several existing approaches to adapt data stream techniques to changes in resource constraints.

The first approach is the threshold-based approach for clustering algorithms. BIRCH [1] was the first threshold-based algorithm that uses an adjustable threshold to allow large datasets to fit into memory. Recently, it has been adopted in new algorithms such as CluStream [2] and LWC [3], which adds more features and/or modifies its structures to be able to adapt to streaming environments. Online stream clustering also has been termed by Aggarwal et al. [2] as *microclustering*.

The second family of algorithms is frequent itemset mining which concerns with finding sets of items occurring together frequently. Giannella et al. [4] have proposed a method to extend the traditional FP tree for finding frequent item sets to mine streaming data in a time-sensitive way. Franke et al. [5] have discussed methods to measure the quality of data stream mining algorithms. In [5], they have used these measurements to analyze and enhance a frequent itemset mining technique. The enhanced technique can estimate the quality of output depending on the current resource situation (mainly available memory) as well as allocate resources needed for guaranteeing user-specified quality requirements.

Teng el al. [6] have proposed the RAM-DS algorithm, which uses a wavelet-based approach to control the resource requirements. The algorithm is used to mine temporal

patterns and is be used in conjunction with a regression-based stream mining algorithm proposed by the authors.

An overview of recent research and application on distributed data mining can be found in [7]. Bandyopadhyay et al. described a K-Means-like technique for clustering homogeneously distributed data streams in a peer-to-peer environments like sensor networks [8].

3 Background

The resource-aware framework is a theoretical generic approach to provide resource-awareness for data stream mining first proposed by Gaber and Yu [9]. It promotes a holistic approach that jointly considers adjusting the settings of the mining algorithm input, output and/or processing endpoints according to resource availability. Gaber and Yu [9] have coined the algorithm input settings as Algorithm Input Granularity *AIG*, the algorithm output setting as Algorithm Output Granularity AOG and the processing settings as Algorithm Processing Granularity *APG*. In general, they are referred to as the Algorithm Granularity Settings or *AGS*.

The AIG represents the process of changing the data rates that feed into the algorithm such as sampling rates or data structure. The AOG represents the process of changing the output size of an algorithm such as the number of clusters formed by a clustering algorithm. The APG represents the process of changing the algorithm parameters to consume less processing power while changing the randomization factor is an example of an APG setting. The resource-aware framework consists of three main components:

1. A resource monitoring component that periodically monitors the availability of various resources. The implementation of the resource monitoring component is platform dependant and the resources to be monitored can also vary. Common resources are battery charge, remaining memory, CPU load, communication buffers or bandwidth.
2. The data mining algorithm processes data in real-time.
3. The algorithm granularity settings that is responsible for adjusting the mining algorithm parameters according to resource availability.

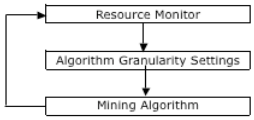


Fig. 1. The resource-aware framework by Gaber and Yu [9].

Gaber and Yu [9] have implemented a resource-aware clustering algorithm in Matlab, called RA-Cluster, which uses the resource monitoring component to adapt to resource availability. RA-Cluster adjusts its microcluster creation radius threshold according to remaining memory, sampling rate according to remaining battery and the randomization factor according to CPU utilization. By increasing the radius threshold, RA-Cluster

discourages the formation of new microclusters, thus, reduces memory consumption. This is done in combination with the removal of outliers and inactive microclusters to free more memory. The randomization factor affects a strategy called randomized assignment. The randomized assignment means that when determining a new data point, only a random number of existing microclusters are examined instead of all microclusters. The higher the randomization factor is, the less number of microclusters are examined. RA-Cluster uses adaptor threshold bounds to adjust the trade off between the resource adaptation and accuracy loss of the algorithms.

In our previous work, we have developed a generic resource-aware framework for wireless sensor networks. The framework has been used to implement a resource-aware clustering algorithm, which we termed Extended Resource-aware Cluster or ERA-Cluster. We have implemented and tested the framework in an actual sensor node. The sensor platform is the novel Sun Small Object Programmable Technologies sensor node from Sun Microsystems, a.k.a. Sun SPOT. Sun SPOT uses the Squawk Virtual Machine, which is a high performance JVM written mostly in Java and designed specifically for resource-constrained devices. Applications for the Sun SPOT node is written entirely in Java and can be deployed and run from the node. Details about the non-distributed ERA-Cluster algorithm can be found in [10].

This paper presents the complete distributed resource-aware framework for wireless sensor networks. By distributed, we mean a hierarchical structure, in which each node can do some data processing such as clustering but the results will be integrated at a parent node which in turn sends to other higher level parents or to base station to answer some queries or for further offline data mining. Firstly, we will discuss the issues coming up within the design of the framework, our solution as well as other alternatives. Secondly, we describe our specific implementation on the Sun SPOT platform.

4 Distributed Resource-aware Online Data Clustering

In the following, we describe our approach to distributed resource-aware data clustering in sensor networks, termed DERA-Cluster. We start by defining the problem we want to solve. After that, we describe our clustering algorithm and how it can adapt to computational resource availabilities. In particular, we focus on how to react to low battery resources in a distributed way in order to meet the lifetime goal with maximal result accuracy. We describe our solutions with respect to the feasibility of the development platform as well other possible alternatives.

4.1 Problem definition

We consider a system of a hierarchical or peer-to-peer wireless sensor network that comprises hundreds of nodes. Each node monitors the environments and does clustering over these collected online data. We propose DERA-Cluster, a distributed resource-aware online clustering algorithm, which can adapt to computational resource availabilities. In a distributed computational model, the main goal is that given a user-specified running time

and a task such as data clustering, the aim is that our network is able to complete the preset runtime and produce as accurate results as possible. The other objective is to minimize the accuracy loss in case few nodes die or stop working due to low availability of resources such as running out of battery, full of memory, and/or full of CPU utilization. Our approach is to migrate current results from a near-dead node to another 'best' neighbour. This gives rise to three main questions:

- Which neighbour to migrate to?
- When to migrate?
- How to migrate (and merge these clustered data)?

In general, the issues are divided into three aspects: migration of data, predicting dynamic thresholds and wireless sensor networking issues.

4.2 The core algorithm

The core of DERA-Cluster is based on our previous work in [10] where we developed a resource-adaptive online clustering algorithm called Extended Resource-aware Cluster or ERA-Cluster. Via ERA-Cluster, we wanted to show a typical AGS scheme – the way the algorithm adjusting to resource availability. *To the best of our knowledge, ERA-Cluster is the first resource-aware algorithm that runs on a sensor node with limited resource availability*. ERA-Cluster is an online threshold-based clustering algorithm, which can be used to reduce or summarize streaming data into microclusters. We allow mechanisms to control the accuracy of the algorithms.

In this paper, we extend this work to DERA-Cluster, a fully distributed clustering approach. The core algorithm runs locally on each node where it subscribes to the resource monitor to receive resource events, and adapts to changes in battery level, remaining memory and CPU utilization similar to what we introduced in [10]. Beside this local adaptation, we introduce a new distributed strategy: If the battery level drops below a minimal threshold, a node will migrate its microclusters to a suitable neighbour, where they will be merge with the existing microclusters. In the following, we present the details of our approaches to migrate microclusters – *when*, *where*, and *how* – in DERA-Cluster.

4.3 Using linear extrapolation model to estimate dynamic migration threshold

A node has to dynamically estimate if the node is able to complete the runtime at each timeframe. If not possible then it will migrate its current result to the best neighbour. In order to answer the question when to migrate, we use a simple linear regression model to dynamically and iteratively estimate three thresholds in descending order:

1. the *adaptive threshold*,
2. the *best-neighbour-finding threshold*, and
3. the *migrating threshold*.

The *adaptive threshold* is the one that triggers the resource adaptation process. This is described in details in [10]. However, there are cases which resource adaptation cannot improve much the situation. In that case, we choose to migrate its existing results before it dies. The second threshold is called *best-neighbour-finding threshold*. As its name

suggests, when resources drop below this threshold, the node starts to broadcast request to its neighbours. Information in the replies is the remaining resource levels. The link quality can also be estimated from the replies. From this information, a 'best' neighbour will be marked. Finally, when resources reach the *migrating threshold*, which typically just enough energy for it to send its data before it dies, this node will migrate its data to the selected neighbour. One simple approach to know when to adapt to resource availability or to migrate data is to use some *predefined* threshold. For instance, when the battery level reaches 70%, a node can start to adapt to resource availabilities; when battery reaches 30%, it starts to query for best neighbour and when the battery reaches 10%, it migrates results. This approach is simplest and also easiest to implement. Under some cases such as where all nodes do the same operation and the resources are consumed steadily, perhaps this is the best approach. However, we are also interested in developed a more dynamic scheme whereas user does not need to specify these predefined threshold but the node dynamically estimate these thresholds. We choose to use a simple linear extrapolation model to estimate whether a node is able to complete its specified runtime. It is the only suitable regression model because non-linear regression model are complicated to implement and cost a significant amount of energy and computational resources.

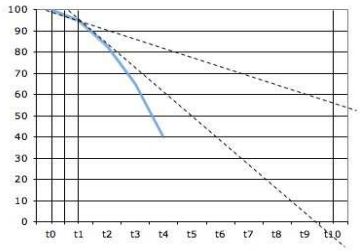


Fig. 2. Linear extrapolation model.

Fig. 2 shows our linear extrapolation model. Suppose, a node is programmed to run for 10 minutes, which is marked t10. At each time frame t0, t1, t2 the node checks its availability resources but it keeps only the most recent time frame resource record. The y-axis shows the battery level. Soon after started running, at time t0, the node measures its battery level. At time t1, it re-measures the battery and calculates the line equation through t0 and t1 which is used to check if it can complete 10 minutes runtime. In this case, it does so the node continues run normally. At time t2, the node re-measures the battery level. In this case, battery drops significantly and it detects that it cannot reach 10 minutes runtime. Thus, it starts the resource adaptation process. Later, if the node detects that resource adaptation cannot improve the situation, it starts to query for best neighbour. Finally, it will migrate result when battery level reaches the minimum amount necessary for sending data. Currently, this minimum battery level is pre-defined for the sake of

simplicity. Given a certain sensor platform, we can measure or estimate level by experimenting with the node.

4.4 Selecting 'best' neighbour

To find the 'best' neighbour to migrate data to, a node can broadcast a query to all of its neighbors asking for their current computational resources level. From the replies, it can also detect the link quality. Most platforms allow this feature. In our approach, we use such broadcasting and then built a two dimensional matrix to represent this information and we have a weighting scheme and a formula to determine the 'best' neighbour. For example, remaining battery is given the best priority; second comes link quality and remaining memory; last is CPU utilization.

4.5 Migrating data.

Data is sent in byte array format, not string, to minimal the amount of transferred data. As current SPOT' API does not support the *serializable* mechanism directly, we need to create our own mechanism to marshal/unmarshal objects to byte array. Basically, we define an *IPersistence* interface which contains the persist() and resurrect() methods. The Cluster class, which represents microcluster, extends this interface and implements these two methods defining how its attributes are actually persisted and revived. We also create a class called VectorHelper to serialize the Vector class, which contains collection of microclusters. Upon migration, ERA-Cluster persists all of its current microclusters to byte array then delegate to the Communicator class to send this data. Communicator is responsible for fragmenting this data into multiple datagram, adding appropriate header and flags before sending off the datagrams. At destination, the data is received and assembled by the Server class.

4.6 Merging data at destination.

At the destination node, the new arrival clustered data will be merged with the existing data on the node. The merging method depends on the mining algorithms. For our DERA-Cluster, the algorithm to merge the data is as follows:

```
FOR EACH new microcluster
   find the minimum distance min_dist to all existing microclusters
   IF min_dist > cluster_creation_ threshold
         keep this new microcluster
   ELSE
         merge this new microcluster with the microcluster with min_dist.
```

Fig. 3. DERA-Cluster's merging algorithm.

The merging formula can be a simple calculation of average mark with weights. Given two microclusters (a_1, a_2... a_N) with K number of records and (b_1, b_2... b_N) with L number of records. Each new attribute of the new microcluster is given by (1):

$$attribute_i = \frac{a_i \times K + b_i \times L}{K + L} \tag{1}$$

The number of records of the new cluster is $K + L$.

4.7 Networking issues

Firstly, most of current sensor node platform supports two basic type of communication: the packet-based or datagram-based communication and the streaming communication. The communication is, however, also one of the most significant factors that consume energy of the node. Thus, in general case, we choose datagram-based communication because it cost much less energy compared to streaming communication. The unreliability factor can be taken into account during implementation.

Secondly, when a node is querying for 'best' neighbour, broadcasting will be used as sensor networks may not necessarily have a robust routing system implemented. When broadcasting, we should assume we only get replies from 'direct' neighbours or the neighbours within the range of the sensor node. One issue that should also be noticed here is that the network follows a hierarchical structure, thus, one might consider the case that a child node always migrate to the parent node whenever it runs out of resources. That is a much simpler model and easier to implement. However, it is not always the best solution as it may lead to a bottleneck at the parent node. Migrating-to-parents can be used in a heterogeneous network in which parent nodes are of different kind than child nodes and have more resources. However, with a network that uses similar nodes, migrating to parent nodes is not the optimum solution.

5 Implementation of the Distributed Resource-Aware Framework

This section discusses issues we faced during the design and implementation of our distributed resource-aware framework for online data mining on the Sun SPOT platform.

5.1 Architectural design of the resource-aware framework

We use a couple of software design patterns to make the framework generic, extensible and maintainable and easy to implement on any platforms. Design patterns are classified in the well-known 'Gang-of-Four' book [11].

Firstly, we use the publish/subscribe pattern to decouple the resource monitor and the adaptive mining algorithms that subscribe to receive resource availability updates. By this way, we can support one or many processing techniques that subscribe to enable resource-awareness. Besides, future extension or modification can be made to the resource monitor

without any change to the rest of the system. As can be seen from Fig. 4, we have implemented our **ResourceMonitor** extends the **Publisher** class, which keeps a list of references to the subscribers. The algorithms that wish to receive resources updates need to implement the Subscriber interface. The **ResourceMonitor** can then use the method *notifySubscriber* (Object resourceEvent) to dispatch resource events.

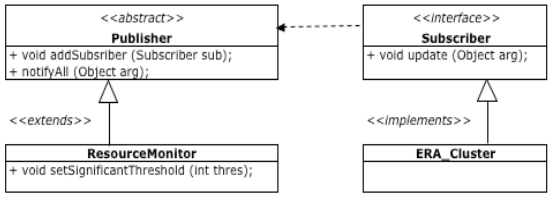


Fig. 4. Publish-Subscribe pattern of resource-awareness framework.

Secondly, we have implemented an abstract *factory* pattern for the data stream generator. We have a **Sensor** class to generate actual data stream sensed from the environment. Currently, data are light, temperature and 3D acceleration values, x, y and z. However for experimental purposes, there is the need for some synthetic data generator that gives us control on the evaluation parameters. For this purpose, we implemented a **RandomDS** class, which generates random data suitable to test our clustering algorithm.

In order to uncouple the data stream generator with the clustering algorithm, we use the factory design pattern. Following this pattern, we create a generic class called **DSGenerator** from which both **Sensor** and **RandomDS** extend. **DSGeneratorFactory** is responsible for creating **DSGenerator**. The implementation of **DSGenerator** is encapsulated and unknown to outsiders. Therefore, we can alternate between **Sensor** and **RandomDS** without changing the rest of the code. Fig. 5 shows the class diagram of the factory pattern.

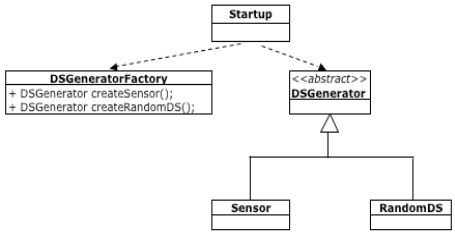


Fig. 5. Factory pattern.

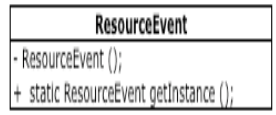


Fig. 6. Singleton pattern.

Thirdly, instead of creating a new resource event object for each update, we choose to have only one static singleton object of the **ResourceEvent** class. This can minimize the consumption of limited virtual memory of the node. Following this pattern (Figure 6), the constructor of each class is marked *private* instead of the normally *public* keyword. This means outside classes cannot arbitrarily create new object of this class. We then have a public and static method named *getInstance*() to return an object of this class. This method will return the existing object if there is already one or create a new object. Besides the resource event, some entities such as the battery simulation class and CPU utilization class are desired to be unique throughout the scope of the application. Therefore, we also apply the singleton pattern to these classes.

5.2 Resource Monitor

The responsibility of the resource monitor is to periodically examine remaining battery, memory and CPU utilization and publish the resource report, which contains status of various resource availabilities. We allow two ways of updating the resource report, *periodic* and *aperiodic* updating schemes. The periodic scheme is the traditional way of updating. This means that the resource monitor notifies the subscribed processing techniques over fixed time frames. The drawback of this approach is that if there is stability in the resource level, CPU utilization will be wasted as there is no need to adjust the algorithm settings. Thus, we have implemented an alternative method, which is the aperiodic scheme. The aperiodic scheme only notifies subscribed processing techniques when the *accumulative change* in resource level is greater than a *significant threshold*. This threshold is submitted to the resource monitor during the algorithm's subscription. For example, an algorithm can request to be notified only if there is more than 10% or 5% changes in resource level. This approach can greatly reduce processing and communication cost. To further reduce the use of the limited memory size of the node, there is only one resource event object follows the *singleton* pattern.

The current implementation of the resource monitor allows monitoring of battery charge, free memory and CPU utilization. For the memory, we use the available API provided by Sun SPOT as memory can be consumed quickly. However, we create two simulations for the battery and the CPU utilization to facilitate the manipulation of resource availability, thus, make it easier to experiment with resource adaptation and accuracy of the algorithm. The battery simulation employs a credit point system, which is used by Younis and Fahmy in [12]. With this approach, each activity of the sensor node is assigned an amount of points and the maximum battery capacity is defined. Activities such as sleep mode, send/receive radio signal, sensing data and computational processing are defined. During operation, the battery charge is decreased gradually according to the sensor activities. With the CPU simulation, we use a simple queuing model that has a fixed queue length and tasks with random generated service time. The CPU utilization is computed as the percentage of total service time of existing tasks in the queue over maximum load. Both simulations have methods to set the resource to a specified level to facilitate experimental setup.

5.3 High level architectural diagram

Fig. 7 illustrates the high level architecture of the system. The ERA-Cluster and the resource monitor block are existing components from our previous project. We add the Communicator to facilitate the sending of datagram and the ServerDeamon is responsible for listening for incoming request and act accordingly. These building blocks make up the basis for the system.

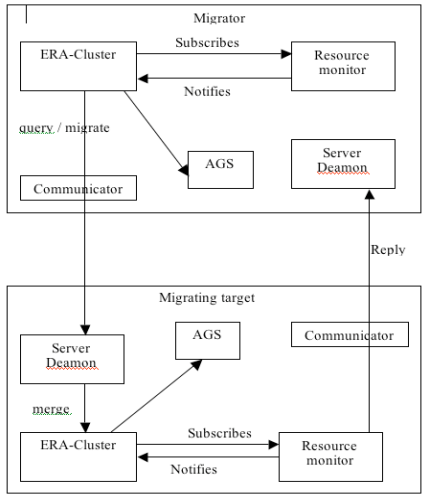


Fig. 7. Distributed Resource-aware Framework

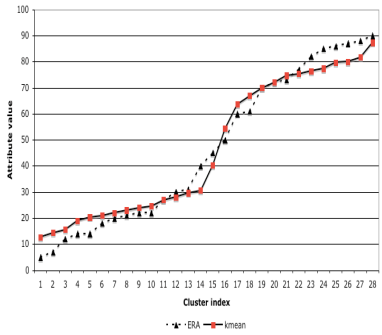
6 Experimental Evaluation

We conducted a small experimental evaluation of our resource-aware framework. We focus on proving two issues: The first is the resource adaptiveness of the framework. In other words, how effective the mining algorithms adapt to resource changes. This issue is to examine by comparing the resources – memory, battery, CPU utilization consumption pattern of the mining algorithm with against without the resource adaptiveness. The second issue of the evaluation deals with proving that the accuracy of the mining algorithm is acceptable even with its parameters adjusted to resource levels. This can be done by using another well known algorithm as a benchmark. For example, we compare the accuracy of our DERA-Cluster with the Weka's [13] simple K-Means clustering. Results show that DERA-Cluster's accuracy is comparable to Weka K-Means under normal operation (with resource adaptiveness). Under high CPU load, the accuracy will

be reduced. However, the overall accuracy is still acceptable. These experiments and results are detailed in [10].

For the distributed case, we aim to show that the accuracy of the migrated results at the destination node is acceptable. Similar to the previous approach, we use Weka K-Means clustering algorithm as a benchmark. The rationale behind using K-Means as a benchmark are also discussed in [10].

We use synthetic data for the experiments. Values are drawn from an uniform random integer of the range 0 to 100. We use a network comprising of two nodes for the experiments. We run node 1 for 10 seconds then migrate its clustered result to node 2. This migrated result is then merged with existing clustered result by the algorithm. We investigate the accuracy of this merged result. The original synthetic data set used up to that moment of node 1 and node 2 are combined. We then run K-Means 3 times over this synthetic data with k = n, n is the number of microclusters of the merged result. We sort all of the results of the merged result and three K-Means according to ascending order of mean value of the microclusters. We then plot the mean value of DERA-Cluster against the average mean value of K-Means. Figures 8 and 9 shows the results of this experiment.



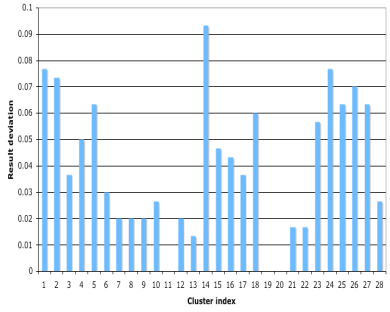


Fig.8. Accuracy of merged result compared to K-Means. **Fig.9.** Result deviation of merged result and K-Means.

From Figure 8, we can see a closed match between the two results despite some deviations. To justify these deviations, we calculate the result deviation of the merged result with K-Means, which is the absolute value of the deviation between the merge result and K-Means average over the maximum range of the mean value. Figure 9 shows that these deviations are small with the maximum accuracy loss are less than 10% while The average result deviation was less than 0.05. In other words, average accuracy loss is less than 5%.

7 Conclusions

This paper has presented a distributed resource-aware framework which can be used to enable resource adaptiveness for selective mining algorithms to be used in wireless sensor networks, in particular the Sun SPOT environment. The design of the framework is detailed and our approach to migrate results when nodes run out of battery are described.

Using the framework, we have implemented a distributed resource-aware online clustering algorithm termed DERA-Cluster. We have evaluated the accuracy of the migrated and merged results at the target node. Experimental results show that the loss of accuracy is acceptable. Possibilities for further work include: a) Evaluate the performance of the frame work on actual sensor data and with a network of multiple nodes. b) Implementation and study of the framework in another sensor platform such as Berkeley's Mote for comparison. c) Using the resource-aware framework to implement other online mining algorithms.

Acknowledgements. This work is supported by the Australian Research Council (ARC) under grant no. DP0664782, and by Sun Microsystems Laboratories.

References

[1] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering method for very large databases", *SIGMOD Rec.*, vol. 25 (2), June 2006.

[2] C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A Framework for Clustering Evolving Data Streams", in *Proc. of VLDB 2004*, 2003.

[3] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: a review", SIGMOD Record 34(2): 18-26, 2005.

[4] C. Giannella, J. Han, E. Robertson, and C. Liu, "Mining Frequent Itemsets over Arbitrary Time Intervals in Data Streams", technical report, Indiana U., 2003.

[5] C. Franke, M. Karnstedt, and K.-U. Sattler, "Mining Data Streams under Dynamically Changing Resource Constraints", in *KDML 2006*

[6] W.-G. Teng, M.-S. Chen, and P. S. Yu, "Resource-aware mining with variable granularities in data streams", *in SDM 2004*, 2004.

[7] S. Datta, K. Bhaduri, C. Giannella, R. Wolff, and H. Kargupta, "Distributed Data Mining in Peer-to-Peer Networks", *IEEE Internet Computing*, vol. 10, pp. 18-26, 2006.

[8] S. Banyopadhyay, C. Giannella, U. Maulik, H. Kargupta, S. Datta, and K. Liu, "Clustering distributed data streams in peer-to-peer environments", *Information Science*, vol. 176, 2006.

[9] M. M. Gaber and P. S. Yu, "A framework for resource-aware knowledge discovery in data streams: a holistic approach with its application to clustering",in *Proceedings of ACM SAC* 2006.

[10] D. N. Phung, M. M. Gaber, and U. Roehm, "Resource-aware Online Data Mining in Wireless Sensor Networks", in *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining*. Honolulu, USA, 2007.

[11] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns - Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1993.

[12] O. Younis and S. Fahmy, "HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks", *IEEE Trans. on Mobile Computing*, vol. 3 (4), pp. 366-379, 2004.

[13] I. H. Witten and E. Frank, Data Mining: Practical machine learning tools and techniques, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

A Model for Quality Guaranteed Resource-Aware Stream Mining

Marcel Karnstedt[1] Conny Franke[2] Mohamed Medhat Gaber[3]

[1] Technische Universität Ilmenau, Ilmenau, Germany
[2] University of California at Davis, Davis, CA, USA
[3] Tasmanian ICT Centre, CSIRO ICT Centre, Australia

Abstract. Data streams are produced continuously at a high speed. Most data stream mining techniques address this challenge by using adaptation and approximation techniques. Adapting to available resources has been addressed recently. Although these techniques ensure the continuity of the data mining process under resource limitation, the quality of the output is still an open issue. In this paper, we propose a generic model that guarantees the quality of the output while maintaining efficient resource consumption. The model works on estimating the quality of the output given the available resources. Only a subset of these resources will be used that guarantees the minimum quality loss. The model is generalized for any data stream mining technique.

1 Introduction

In the past years, data streams emerged as a new kind of data source. Analyzing data streams thus becomes more and more important as new areas of application are identified. Applications like click stream analysis and the analysis of records from networking and telephone services are among the most popular examples for data stream mining, i.e., the discovery of patterns and rules in the data. Another important area of application is the stream processing in sensor networks, where continuously generated data is processed as far as possible onboard the sensor node in order to preserve the limited bandwidth and energy.

Due to the unique characteristics of data streams, like their potentially infinite nature and the vast amount of data they are carrying, data stream mining requires a different processing than mining on databases and data warehouses. Efficient resource consumption is one of the major objectives when designing stream mining algorithms. Rather than storing the incoming data and processing it offline like in traditional data mining, data stream mining is much more constraint in terms of available resources.

Most data stream algorithms provide approximate results, often by using a summarization of the stream (called a *synopsis*) and determining precise error bounds. Thus, a notion of output quality is immediately associated with this process. Which information from the data stream is stored is crucial for the quality of the data mining results. Note that we explicitly refer to the output

quality of the mining technique, in contrast to aspects of the input quality, which is a related but still different field of research.

Well-known state-of-the-art of stream mining algorithms reveal that while many of the algorithms strive for minimizing the resources they need, most of them are not designed with regard to adaptation to resource availability. Specifically, they fail to provide well-defined routines for situations where the available resources are exhausted. Most algorithms are designed to work in a static manner, without taking into account that the algorithm's resource requirements might exceed the amount of resources provided. In such a case the algorithm's behavior depends in its implementation, and thus might be undefined. Recent approaches (e.g., [1]) identified this issue, but still lack a clear consideration of correlations between resource-adaptation and output quality.

When dealing with complex stream mining systems, where usually a set of queries runs continuously and resources are shared among them, we additionally have to consider the interactions between different mining operators. In such systems, algorithmic output quality is usually referred to as Quality-of-Service (QoS) [2]. Note that, when combining resource and quality aspects of multiple, possibly dependent, mining operators, we start closing the mentioned gap between input quality and output quality of mining techniques.

In this work, we consider all of the aforementioned aspects and integrate them into one single framework. We propose a generic three layer model for quality guaranteed resource-aware (QGRA) data mining on data streams. The model is designed to be applicable to a wide variety of stream mining techniques. Our model assesses the output quality and the current status of resources and adapts the algorithm's resource consumption accordingly. This way, we are able to maintain resource efficiency and we may use lesser resources to achieve the same level of accuracy in the output. At any time, we will be having the maximum achievable quality according to the resources. Most static data stream mining algorithms leave excess resources unused. With our framework, available resources are utilized in an optimal way at any point in time.

The model utilizes a set of functions that is provided by the applied algorithm to control the adaptation. One function is used to determine the algorithmic parameters from the assessed resources. Then, a second function is used to determine the output quality based on the chosen algorithmic parameters. Other functions are used to compute lower bounds for the algorithmic parameters in order to maintain the quality of the output. Using this strategy, our model bridges the gap between quality-aware mining and general resource-adaptivity in data stream mining by monitoring the resource consumption.

The remainder of this paper is organized as follows. In Section 2 we provide some background about data mining quality and resource-aware data stream processing. Our formal model is introduced in Section 3. There, we give a brief description of the functionalities and a formalization of all elements. Finally, Section 4 concludes this paper and outlines areas for future work.

2 Background

2.1 Data Mining Quality

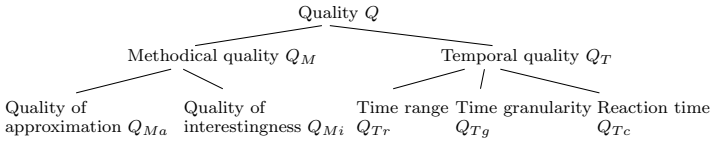


Fig. 1. Different quality measures

Most data mining algorithms have control parameters to determine how well their output approximates the actual result. These control parameters differ for each technique and data mining goal, but they can be arranged into classes of related parameters. Typical examples include the number of microclusters maintained during clustering, or the maximum frequency error in frequent itemset mining. The values of these parameters have a strong influence on the workload of the algorithm as well as on the size of the synopsis it maintains. In general, the better the approximation of the output should be, the more resources the algorithm consumes. Due to this close correlation between these parameters and the output quality of an algorithm, we will refer to this set of parameters as *adaptation factors*. After identifying the adaptation factors of an algorithm, we are able to adapt its resource requirements and output quality.

In analogy to a classification of adaption factors, affected quality measures can be classified, too. We distinguish several different classes of quality measures, which are categorized in Figure 1. This classification is comprehensive, yet extensible without restricting the proposed framework. All Q_{T*} are identical for different mining problems and symbolize concrete quality measures, while Q_{M*} represent classes of measures that are always specific to the investigated problem and the applied algorithm(s). For example, in the context of clustering these measures involve the clustering quality, e.g., SSQ, diameter and other standards to evaluate the final result of a clustering. One traditional measure for the problem of frequent itemset mining is the error rate ϵ, which defines the maximal deviation of the observed frequency to the actual frequency of an itemset. For several specific mining applications, special interestingness measures (Q_{Mi}) have been proposed in the literature (e.g., [3]). In the context of frequent itemsets the support is one such interestingness measure.

As many existing algorithms take time sensitiveness into account, we define time as another important quality measure. Q_{Tr} describes how far we can look back into the history of the processed data stream and Q_{Tg} how exact we can do this, which means which time granularity we can provide. Q_{Tc} corresponds to one of the main challenges of stream mining: the actual time necessary to register changes in the stream. As might be expected, adaptation factors sometimes

influence more than one of these quality measures, making them dependent from each other. For more details on the different quality classes we refer to [4]. For the remainder of this paper, if we refer to all quality measures as a whole, we will use the symbol of the superclass Q and the general term 'quality'.

2.2 Resource and Quality Awareness

Most data stream mining algorithms are designed to use as little resources as possible. However, they are often not aware of the actual amount of resources available and thus may either fail to utilize them completely or may not be able to work properly with the given resource constraints. We therefore distinguish algorithms that are aware of the available resources and are able to adapt their requirements accordingly. When talking about resources, we do not only consider memory consumption, despite this being the main factor in most streaming applications. In addition, since data streams are often generated at a rapid rate, algorithms must need only minimal time to process the data in order to keep up with the pace of the stream. In the example of sensor networks, bandwidth and battery power are additional constraint resources.

Apart from the adaptation factors, properties of the data stream also have a strong impact on an algorithm's resource requirements. One of the most important properties is the streaming rate. Another one are characteristics of the individual elements in the data stream, like the range and distribution of their values, and their size. This correlates to adaptation methods like sampling and load shedding [5], which can be used on the input level of mining techniques to reduce the workload, and thus the resource requirements, by decreasing the volume of the incoming stream. Due to their generic nature, they are applicable to all mining algorithms, since they do not require any changes to the algorithm itself. As a consequence, however, applying these methods results in the loss of guaranteed error bounds which the original algorithm may have provided. Instead, only probabilistic error bounds can be established. This may be a non-desirable tradeoff for some applications. Moreover, determining these probabilistic bounds should be expected to be very complicated and, due to the evolving character of streams, potentially erroneous.

Other levels of resource-adaptation throughout the whole process of stream mining have been identified and discussed in recent works. Methods that can be applied to most data stream mining algorithms have been proposed for example in [6]. Most other approaches either do not formalize their approaches accordingly or focus on single, rather limited, levels of adaption. Moreover, to the best of our knowledge, all of them lack the combination of resource and quality awareness. That means, although they deal with resource adaptation, they do not take quality aspects and guarantees into consideration.

3 QGRA Model

3.1 Model Description

Gaber et al. [6] have proposed and developed a generic model to adapt data stream mining algorithms to the current availability of computational resources. The model aims at prolonging the life-time of the running technique in critical situations of low availability of computational resources. It has been coined as *Algorithm Granularity (AG)*. AG can adapt the consumption of computational resources according to measured patterns of consumption over a pre-set time window.

AG has been classified into three classes according to the adaptation end-points. *Algorithm Input Granularity (AIG)* adapts the input streaming data to the mining algorithm. On the other hand, Algorithm Output Granularity (AOG) changes the output size of the algorithm. Finally, *Algorithm Processing Granularity (APG)* can adapt the algorithm parameters to consume less CPU cycles. The changes in AG affect the accuracy of the output. Therefore, the model sets bounds on the AG settings in order to keep the accuracy loss bounded. These AG settings have been used to develop an online clustering algorithm termed as *Resource-Aware Cluster (RA-Cluster)*.

Although the AG model has proven its applicability to change the resource consumption, the model is still facing the following issues:

- The bounds over the AG settings have no guarantee over the quality of the output. Because the quality relies on many other interleaving factors such as data distribution and the running mining technique.
- The changes in the AG settings are not quality-aware. That means the algorithm changes according only to the availability of computational resources. This may lead to accuracy loss and/or extra use of computational resources, because in some cases, we can gain the same accuracy using less resources.
- The AG settings do not take into consideration the interaction among the different settings. Addressing this issue can optimize the use of resources.

In this paper, we propose a new model QGRA that extends AG in order to address the above issues. The model is able to adapt in real-time according to resource consumption patterns as well as the quality of output. Franke et al. [4] have proposed a quality-aware data stream mining model. This model will be extended to assess the quality of the output in real-time. This assessment will be used to choose the best combination of AG settings that minimize resource consumption, and maximize the quality of output.

The model has three layers. The first one is the resource monitoring that works over dynamic time intervals. Unlike the AG model, the time window is dynamic and changes according to the criticality of the available computational resources. The second component is the real-time quality assessment. This will be able to provide information about the quality of the output given the availability of resources. It will also be able to provide the system with information about preserving computational resources while maintaining the same quality

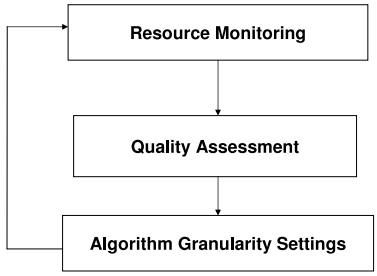


Fig. 2. Three layer model

level. The output of the second component will be passed to the AG settings (AGS) component. This third component feeds the mining algorithms with input, output and processing settings.

3.2 Formalization

The QGRA model relies on the notion of *variables* $v \in V$, which comprises all dynamic components the internal formulas and mechanisms are built on. We define a partitioning $\breve{V}$ over V, where each v belongs to a certain class $\hat{C}$ of variables, described as follows. As $\breve{V}$ is a partitioning,

$$\forall \hat{C}_1, \hat{C}_2 \in \breve{V}, \hat{C}_1 \neq \hat{C}_2 : \hat{C}_1 \cap \hat{C}_2 = \emptyset$$

holds. That is, each variable v belongs to exactly one class $\hat{C}$ of variables.

On the one side of the adaptation framework, we assume a set of *resources* $\hat{R} \in \breve{V}$,

$$\hat{R} := \{r | r \text{ is a limited resource consumed by the algorithm}\}.$$

Main representatives of $\hat{R}$ are the consumed memory and the number of CPU cycles needed. This corresponds to the kinds of resources already considered in previous works like [6, 4, 1].

The basis of the framework is enriched by quality awareness, defined in terms of *quality measures* $\hat{Q} \in \breve{V}$,

$$\hat{Q} := \{q | q \text{ is a quality measure of interest}\}.$$

There is a wide variety of quality measures that might be integrated into $\hat{Q}$. We present a brief classification in Section 2.1, which is based on the more detailed work in [4].

On the contrary side of the framework, we expect any stream mining algorithm to define a set of *parameters* $\hat{P} \in \breve{V}$,

$$\hat{P} := \{p | p \text{ influences resource requirements and/or output quality of the algorithm}\}.$$

These parameters can be seen as the "tuning knobs" of the particular method. This includes representatives of adaptation factors corresponding to the afore mentioned classes AIG, APG and AOG, e.g., sampling rate, internally used thresholds or number of output objects (like the number of output clusters). In addition, we also include parameters that cannot be adjusted but the method exhibits some dependence on, in either resource requirements, output quality, or both. Examples of this kind of parameters are distribution models the stream data follows or the fraction of noise. More on the specific requirements an algorithm has to meet can be found in Section 3.3.

In [6] the sets $\hat{R}$ and $\hat{P}$ were used in order to implement a one-way resource adaptation. This means, predictions for relevant $r \in \hat{R}$ are used to adapt $p \in \hat{P}$ accordingly, while the prediction of future resources is based on the observed recent resource consumption. Now, we introduce how to combine this idea with the quality-awareness proposed in [4].

We define an instance $F(\hat{C})$ as a materialization of all variables from class $\hat{C} \in \breve{V}$ which are actually involved in the adaptation process, i.e.,:

$$\begin{aligned} F &: \breve{V} \rightarrow V \times \Re \\ F(\hat{C}) &:= \{v, f(v) | v \in \hat{C} \wedge \nexists w \neq v : w \in \hat{C} \\ &\quad \wedge f(v) \text{ is the current value of } v\}. \end{aligned}$$

As v only defines which variable is concerned, $f(v)$ represents an actual value of that variable. To improve readability, we use C short for $F(\hat{C})$. In other words, C represents all variables from class $\hat{C}$ together with the corresponding values. For instance, we use R to represent all limited resources and their actual amount currently consumed by a specific algorithm.

Informally speaking, $\hat{C}$ represents the schema level of the variable classification, i.e., a description of which variables belong together in a class. Accordingly, C denotes an instance of $\hat{C}$, that is, it associates each variable $v \in \hat{C}$ with an actual value $f(v)$. In the above definition of C, we write $\nexists w \neq v : w \in \hat{C}$ to denote $\forall (v, f(v)) \in F(\hat{C}) : v \in \hat{C} \wedge \forall v \in \hat{C} : (v, f(v)) \in F(\hat{C})$. That is, on the instance level we have exactly one value $f(v)$ for each $v \in \hat{C}$. Note that distinguishing between classes $\hat{C}$ and instances C is not urgently necessary to make the model work. However, we believe that this makes the model more flexible, resulting in more algorithms and approaches being applicable to it. For instance, the introduced notion allows for an easy but still mathematically consistent integration of "schema-based" functions, e.g., which select actually considered quality measures from the set of all possible ones.

On the notion of the variables and instances we introduce two more kinds of sets. First, we define R_L to represent current resource limits and Q_L to represent requested quality guarantees. Whether these limits are met or not is described by two functions:

$$\Phi(R_L, R) := \begin{cases} \text{true} & \text{if } \forall (v, x) \in R_L : (v, y) \in R \wedge x \geq y \\ \text{false} & \text{else} \end{cases}$$

The resource limits R_L are met in R, i.e., $\Phi(R_L, R) = true$, if for each $v \in \hat{R}$ its value y in the instance R is less than or equal to its limit x in R_L.

$$\Psi(Q_L, Q) := \begin{cases} \text{true} & \text{if } \forall (v,x) \in Q_L : (v,y) \in Q \wedge x \leq y \\ \text{false} & \text{else} \end{cases}$$

The quality guarantees Q_L are met in Q, i.e., $\Psi(Q_L, Q) = true$, if for each $v \in \hat{Q}$ its value y in the instance Q is greater than or equal to its limit x in Q_L.

Finally, we define *timelined* variable instances C_T. A set C_T corresponds to an instance C enriched by a timestamp t, which represents the time the according values were effective. Thus,

$$C_T := \{v, x, t | (v,x) \in C \wedge \nexists (w,y) \neq (v,x) : (w,y) \in C \\ \wedge (v,x) \text{ was effective at time } t\}.$$

C_T associates each pair $(v,x) \in C$ with a timestamp t. In the following, if we refer to an instance of one specific time t we write C_t for short.

On the introduced sets we define the following functions:

$$\rho : R_L \times R_T \times P_T \to \{-1, 0, 1\} \tag{1}$$
$$\phi : P \to R \tag{2}$$
$$\psi : P \to Q \tag{3}$$
$$\tau : R_L \times R_T \times P_T \to P \tag{4}$$
$$\omega : Q_L \times Q_T \times P_T \times P \to P \tag{5}$$

The first formula ρ is used to decide whether future resource consumption should be increased (underload situation, $\rho = 1$), decreased (overload situation, $\rho = -1$) or nothing is to be done at all ($\rho = 0$). This decision is based on provided resource limits, recent resource consumption and recent parameters. There are different approaches for handling this issue. [6] proposes to calculate the number of time frames remaining until resources are exhausted, whereas [4] uses a filling factor describing the percentage of available resources already consumed.

ϕ and ψ take as input an instance of parameters and map them to the resulting instance of resources and quality measures, respectively. τ and ω can each be seen as a kind of inverse function, mapping an instance of resources, respectively quality measures, to an instance of parameters. As additional input, both τ and ω accept recent parameter values and recent resources/quality measures. In order to align the quality-based decision with a preceding resource-based decision, ω also takes suggested parameters P as an input.

Based on these sets and functions the QGRA model works as illustrated in Algorithm 1.

As mentioned before, the time intervals in which the QGRA method is applied are set dynamically. In the beginning, all parameters are set to achieve highest possible quality. The algorithmic steps from Algorithm 1 are then applied at any time t. Based on the current resource consumption in time t provided

Algorithm 1 General QGRA algorithm at time t

1: $R_t = \text{res-mon}(t)$
2: **if** $0 \neq \rho(R_L, \cup_{i=1}^{t} R_i, P_t)$ **then**
3: $\quad P = \tau(R_L, \cup_{i=1}^{t} R_i, P_t)$
4: $\quad P' = \omega(Q_L, \cup_{i=1}^{t} Q_i, P_t, P)$
5: $\quad$ **if** $\Phi(R_L, \psi(P'))$ **then**
6: $\quad\quad P = P'$
7: $\quad$ **end if**
8: $\quad$ **if** $!(\Phi(R_L, \phi(P)) \wedge \Psi(Q_L, \psi(P)))$ **then**
9: $\quad\quad$ **return** false
10: $\quad$ **end if**
11: **else**
12: $\quad P = P_t$
13: **end if**
14: $Q_{t+1} = \{q, x, t+1 | (q, x) \in \psi(P) \wedge \nexists (u, y) \neq (q, x) : (u, y) \in \psi(P)\}$
15: $P_{t+1} = \{p, x, t+1 | (p, x) \in P \wedge \nexists (o, y) \neq (p, x) : (o, y) \in P\}$
16: set parameters P
17: **return** true

by the resource monitoring component (line 1) and (dynamically) provided resource limits, ρ is used to decide whether resource utilization should be increased, decreased or maintained (line 2). If any adaptation is necessary, a new set of parameters is determined using τ on the provided resource limits and on recent resource consumption as well as parameters (line 3). In the next step, we refine these parameters using ω on the QoS requirements and on recent quality measures as well as recent parameters (line 4). In order not to work contrary, ω also takes the set of parameters suggested before by the resource adaptation as additional input. Only if the parameters modified like this still meet the resource limits (line 5), they are accepted (line 6). After all adaptation steps, the new parameters are checked for resource and QoS requirements again (line 8). If they are not acceptable, the failed resource and/or quality requirements are signalized and reaction is left to the system or user. Otherwise, the resulting qualities and parameters are stored for later access (lines 14 & 15) and finally set (line 16). Note that, if no adaptation takes place, parameters for time $t + 1$ are set to those from time t in order to build the timelined sets (line 12).

It is worth to note that formulas 4 and 5 involve solving a kind of optimization problem, which is left to the specific mining technique. By this, the interaction between different kinds of resources, quality measures and adaptation end-points is regarded. A significantly more complex approach is to include qualities and resources into *one single* optimization problem. But we expect any such problem to be much too complex in order to be solved in practicable time.

Applying the model as proposed, there is only one question unanswered until now: What quality is provided when querying the mining result of an arbitrary time interval? As stream mining algorithms should support such queries but quality measures differ between different time intervals, the answer to this question is fundamental to support meaningful quality awareness. Thus, we define a

last function

$$\xi : Q_T \times N \times N \rightarrow Q \tag{6}$$

which determines an instance Q of quality measures extracted from the gathered timelined qualities Q_T with respect to a given interval of time.

3.3 Requirements on Algorithms

Though we try to capture most of the existing data mining algorithms on data streams, the proposed framework is not applicable to all mining algorithms. There are some basic requirements that algorithms need to fulfill in order to be extendable using our model.

Naturally, one of the crucial requirements is that parameters must exist in the algorithm, so that we can choose adaption factors. Moreover, a strong correlation between the adaption factors and the algorithm's resource requirements aids precisely estimating the quality of the output. Also, in order to anticipate an algorithm's resource requirements as well as the quality of its output, the algorithm must show homogeneous behavior when provided with an input stream whose properties are maintained homogeneous as well. For example, most threshold based stream mining algorithms meet these requirements.

Another important property is that there should exist a partitioning into independent sections in the mining result of an algorithm. That means that different values of the adaption factors only have "local" effects in the mining results. This also indicates that it must be possible to query each of these independent sections separately, since otherwise the lowest quality setting of the adaption factors in the history of the data stream processing will determine the quality of the overall mining results. As an example, consider frequent itemset stream mining using a landmark window model, where the error threshold is one of the adaption factors. Since we can only query the complete history of the stream, the lowest value of the error threshold ever used while processing the stream will determine the overall output quality of this algorithm. Note that this last aspect is not a strict requirement but rather a helpful property. If an algorithm does not satisfy this requirement, it will still be applicable for the proposed framework, but it will not be able to "recover" from low quality adaption factor settings.

Currently, we are adopting the resource- and quality-aware mining techniques we proposed in former works to the presented framework. Despite some last formulas, this is almost done, which shows that the introduced formalization is suitable for application to existing and future proposals.

4 Conclusions and Future Work

Mining data streams stresses our computational resources with regard to processing power, memory requirements, energy and communication. In order to

ensure the continuity and consistency of a data stream mining process, adaptation to available resources is required. Although adaptation is crucial for the success of the data mining process, its effect on the quality is of concern. Thus, in this paper we propose a Quality Guaranteed Resource-Aware (QGRA) data stream mining approach. The objective of this approach is two-fold. Firstly, the adaptation is done while maintaining a guaranteed QoS set by the user. Secondly, utilization of resources is achieved through mapping of required resources to quality measures. If the same quality could be achieved with less resources, only the required resources are consumed with appropriate parameter settings.

Our work provides the mathematical foundation to add quality-guaranteed resource awareness to most stream mining algorithms. Concrete instances of the formulas given in this paper must be implemented by the respective algorithm. We are currently working on applying our model to existing algorithms for the three main data mining tasks, clustering, classification, and frequent itemset mining.

The paper presents a pioneering work to address the two most important challenges in data stream mining, namely, resource constraints and quality of the output model. We believe in the flexibility and suitability of the proposed framework in order to cover existing and following approaches as well as meeting the special challenges of stream mining. By this, we built a valuable and essential basis for future work on quality guaranteed resource-aware stream mining.

References

1. Jiang, N., Gruenwald, L.: Research issues in data stream association rule mining. SIGMOD Rec. **35** (2006) 14–19
2. Motwani, R., Widom, J., Arasu, A., Babcock, B., Babu, S., Datar, M., Manku, G.S., Olston, C., Rosenstein, J., Varma, R.: Query processing, approximation, and resource management in a data stream management system. In: CIDR. (2003)
3. Tan, P., Kumar, V.: Interestingness measures for association patterns: A perspective. KDD WS on Postprocessing in Machine Learning and Data Mining (2000)
4. Franke, C., Hartung, M., Karnstedt, M., Sattler, K.: Quality-Aware Mining of Data Streams. In: Information Quality (ICIQ). (2005) 300–315
5. Chi, Y., Wang, H., Yu, P.S.: Loadstar: load shedding in data stream mining. In: VLDB, VLDB Endowment (2005) 1302–1305
6. Gaber, M.M., Yu, P.S.: A framework for resource-aware knowledge discovery in data streams: holistic approach with its application to clustering. In: SAC. (2006) 649–656

Relational Transformation-based Tagging for Human Activity Recognition

Niels Landwehr[1], Bernd Gutmann[1], Ingo Thon[1], Matthai Philipose[2], and Luc De Raedt[1]

[1] Department of Computer Science
Katholieke Universiteit Leuven
Celestijnenlaan 200 A, B-3001 Heverlee, Belgium
firstname.lastname@cs.kuleuven.be
[2] Intel Research Seattle
1100 NE 45th Street
Seattle, WA 98105, USA
matthai.philipose@intel.com

Abstract. The ability to recognize human activities from sensory information is essential for developing the next generation of smart devices. Many human activity recognition tasks are — from a machine learning perspective — quite similar to tagging tasks in natural language processing. Motivated by this similarity, we develop a relational transformation-based tagging system based on inductive logic programming principles, which is able to cope with expressive relational representations as well as a background theory. The approach is experimentally evaluated on two activity recognition tasks and compared to Hidden Markov Models, one of the most popular and successful approaches for tagging.

1 Introduction

Smart systems that assist humans must be able to recognize the current context of the user and the activity she is performing in order to suggest or take actions in an intelligent manner. To recognize the context and activity, such systems can rely on streams of past activities, context, and dense sensory information, as it is often gathered in ubiquitous computing environments. Recognizing the current activity or context then corresponds to inferring the activity or context from such sequential information. From a machine learning perspective, this task is akin to many tagging tasks pursued in natural language processing (NLP). For instance, in part-of-speech tagging, a form of "shallow parsing", the words in a sentence are to be labeled with the corresponding parts-of-speech (word categories). In this paper, we will investigate this relationship and explore how tagging techniques can be applied in activity recognition domains.

In NLP, many techniques have been developed and employed for tagging purposes. Two popular techniques for part-of-speech tagging are Hidden Markov Models and transformation-based learning [1]. While Hidden Markov Models have been applied in many different areas, ranging from speech-recognition to activity recognition and bioinformatics, to the best of the authors' knowledge, transformation based learning has only seldomly been applied outside the field of natural language processing.

Because the structure of natural language is quite rigid as compared to that of typical activity recognition tasks, the existing transformation-based learners cannot directly be applied for activity recognition. Therefore, we develop a more flexible *relational* transformation-based tagger. This does not only provide an *expressive* representation, which allows to easily model complex sensory information, but also an easy way to incorporate prior domain knowledge in the learning process. Thus the key contribution of this paper is the application of the transformation-based tagging methodology to rich sensor data streams, which is realized by extending transformation-based tagging with a relational representation based upon inductive logic programming principles.

The presented relational transformation-based tagger also extends earlier work on relational transformation-based learning by [2] in that it focuses on *tagging* rather than *classification*. More specifically, from inductive logic programming (and the work by [2]) our technique inherits its search and refinement techniques (including a branch-and-bound algorithm) and from transformation-based learning the error driven stacking of rules.

The proposed method is evaluated in two activity recognition domains: "Activities of Daily Living" (ADL) recognition from a stream of "object interaction" data [3], and mobile phone profile prediction based on data collected by [4]. Experiments show that obtained tagging accuracies are competitive with those of HMM-based approaches, and it is easy to incorporate human-supplied background knowledge into the learning process. Furthermore, and that is perhaps the key advantage of the relational transformation-based tagger, the method can easily be extended to deal with variants of the tagging problem, for instance the prediction of structured output tags (as in Logical Hidden Markov Models [5]), and to cope with rich background knowledge.

2 Activity Recognition

Activity Recognition, in the broadest sense, is concerned with labeling a stream of sensor data with a context or activity label. The sensor data stream is typically collected from an ubiquitous computing environment, such as sensors embedded in everyday objects [3] or data gathered from mobile phones [4]. The context or activity label predicted at every step in time can characterize the physical activity a user is performing, her state of mind or intentions, or additional context information that cannot directly be observed through sensors. Activity Recognition and related problems are important challenges when designing the next generation of smart devices: knowing e.g. a user's intentions can make a device adapt to the current situation and thus improve usability. Furthermore, the collection of data about human behavior can be interesting in its own right, e.g. in elderly care. Here, it can be interesting to automatically generate reports about an elderly person's activities of daily living to detect medical problems at an early stage, without need for continous human supervision in an elderly care facility.

The gathered sensor data takes the form of a continous, dense and often heterogenous data stream. Depending on the complexity of the available sensor information, the data might also be structured, i.e. not easily representable with simple flat symbols from a fixed alphabet. As an example, consider the Activities of Daily Living (ADL) domain visualized in Figure 1. In ADL recognition, objects which are used in activities

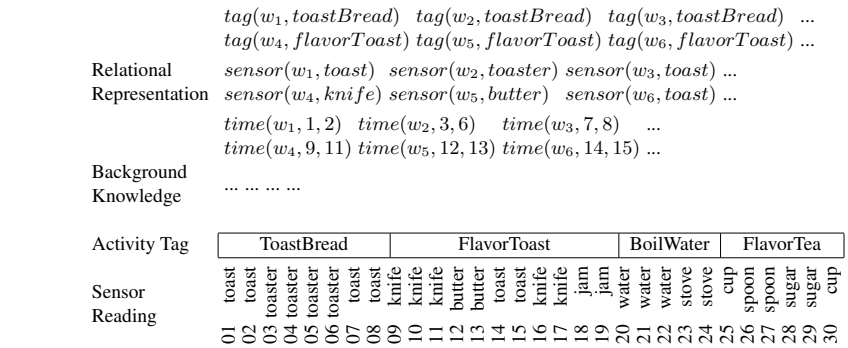


Fig. 1. Relational representation of the ADL recognition problem. While a person is performing activities of daily living (such as preparing breakfast), a stream of object interaction data is generated from a wearable RFID reader ("sensor reading"). This can be represented in a relational form by collapsing identical sensor readings to one sequence element w_i, and encoding the starting point and duration of the observation in another predicate. Furthermore, additional background knowledge can be used to encode prior knowledge about the domain.

of daily living such as making breakfast are equipped with small RFID tags that can be picked up by a wearable reader while a person performs an activity [3]. The task is to recover the activity currently performed from the stream of sensor data. Note that the stream of object data obtained from the sensor has some internal structure, as an object observation has a starting point and duration in time, which can be easily represented in a relational formalism.

At an abstract level, predicting activities is a sequence tagging problem, and exploring applications of sequence tagging methods to activity recognition will be the main focus of the paper. At the same time, the problem is an instance of *data stream mining*—the analysis of a continuous, potentially infinite stream of data. In this context, issues such as online learning (with only one pass through the data necessary) are of considerable interest. However, we will not address these issues in the paper, and instead assume that a limited amount of training data is given a priori. Extending the proposed methods to an online-learning scenario is an interesting direction for future work.

The following section reviews sequence tagging from a natural language processing perspective. Section 4 discusses a relational extension of transformation-based tagging for activity recognition. Finally, Section 5 presents experimental results and Section 6 conclusions and related work.

3 Sequence Tagging

Sequence tagging is the task of assigning to each element in a given sequence an appropriate label or *tag*. Let $W = \{w^1, ..., w^k\}$ denote the vocabulary of sequence elements,

Algorithm 1 Basic transformation-based tagging algorithm.

tb-tagging(input: sequences S; true sequence tags L)

```
1  L̂ := initial-tags(S, L)
2  initialize R = []
3  repeat
4        r := find-best-rule(S, L̂, L)
5        update L̂ := apply-rule(L̂, r)
6        update R := append(R, r)
7    until (no improvement)
8  return R
```

and $T = \{t^1, ..., t^m\}$ the vocabulary of tags. In natural language processing, the two most common tagging approaches are transformation-based taggers (rule-based) and probabilistic methods (hidden Markov models or related techniques). Both of these approaches yield competitive results, and have received much attention.

3.1 Transformation-based Tagging

Transformation-based learning is a rule-based learning approach which iteratively stacks rules on top of each other to improve performance [1]. The basic transformation-based learning algorithm for the tagging problem is summarized in Algorithm 1. The algorithm takes as input a set S of sequences with known true tags L. During learning, it maintains a set of current tags $\hat{L}$ for all $s \in S$. $\hat{L}$ is initialized with some simple scheme, such as assigning to every element $w \in W$ its most common tag $t \in T$ in the training data (procedure *initial-tags*). The algorithm then tries to improve the current tagging $\hat{L}$ with respect to the true tagging L by learning a list of *transformation rules* R. Transformation rules can re-tag sequence elements based on the context they appear in. A transformation rule has the form $t' \leftarrow t : context$ and simultaneously replaces all occurrences of tag t in all sequences with t' whenever the constraint $context$ is satisfied.

Example 1. As an example from NLP, the word "move" could be initially tagged as "verb", but would be re-tagged as "noun" if the preceding word was tagged as "article":

$$\textit{noun} \leftarrow \textit{verb} : \text{word} = \textit{move}, \text{preceding tag} = \textit{article}$$

The transformation rule languages employed in traditional transformation-based tagging are mostly simple instantiations of some template—for instance, querying in $context$ the word and tag at the current position and the next or preceeding position(s). In every iteration, the transformation rule which yields the greatest reduction in error between $\hat{L}$ and L is greedily selected (*find-best-rule*), applied to the current tagging $\hat{L}$ and appended to the rule list R. As conditions of rules in R match not only sequence elements but also currently predicted tags $\hat{L}$, rules can effectively bootstrap the current predictions. This makes transformation-based learning strictly more powerful than standard rule learning [1].

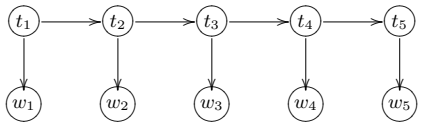


Fig. 2. Example lattice generated by unrolling a tagging HMM to a sequence $w_1, ..., w_5$. Inference in this model is carried out with the Viterbi algorithm, which yields the most likely joint state of the hidden variables $t_1, ..., t_5$ given the observations on $w_1, ..., w_5$.

3.2 Hidden Markov Model Tagging

Tagging with hidden Markov models is typically performed with a model in which there is a hidden state q_t for every possible tag t, and state emission symbols correspond to symbols $w \in W$. That is, the observed sequence of symbols is seen as being generated by the hidden sequence of tags. Formally, the joint probability of an observation sequence $s = w_1...w_n$ with hidden tag sequence $t_1...t_n$ is given by

$$P(w_1...w_n, t_1...t_n) = P(t_1) \prod_{i=1}^{n-1} P(t_{i+1} \mid t_i) P(w_i \mid t_i)$$

where $P(t_1)$ is an initial probability for tag t_1 and $P(w_i \mid t_i)$, $P(t_i \mid t_{i-1})$ are conditional probabilities for the emitted word w_i and next tag t_{i+1} given the current tag t_i. When such a model is applied to a sequence $w_1...w_n$, it is unrolled into a lattice as depicted in Figure 2, and the Viterbi algorithm [6] is employed to efficiently compute

$$\hat{t}_1...\hat{t}_n = \arg\max_{t_1...t_n} P(t_1...t_n \mid w_1...w_n),$$

the most likely sequence of tags for the given sequence. This technique has been used successfully for tagging problems in many domains.

4 Relational Transformation-based Tagging

The general motivation for our work is to apply the transformation-based tagging methodology to complex datastreams, which are generated for instance by sensors or sensor networks in ubiquitous computing environments. In many cases, such complex datastreams are most easily represented in an expressive relational formalism. Consequently, we will extend the template-based rule language traditionally used in transformation-based learning to a more flexible *relational* rule language, which can take advantage of such richer representations for sequence elements. Furthermore, this allows to incorporate domain-specific background knowledge into the learning process. Analyzing such relational sequences has received considerable attention recently, for instance with relational extensions of Hidden Markov Models [5] or n-gram models [8].

Example 2. As an example, consider the ADL ("Activities of Daily Living") recognition problem, which is visualized in Figure 1. It is obvious that this kind of data is less

rigidly structured than natural language data: there are no "grammatical rules" which determine the exact sequence of touching knife, toast, butter and jam when adding flavor to a toast. Nevertheless, context information can help determine the right tag. For instance, using a spoon can indicate activities FlavorTea or EatCereals. This ambiguity can be resolved by looking at the context: the observation of a spoon closely followed by sugar indicates activity FlavorTea, while observation of a spoon after milk and cereals indicates activity EatCereals. A relational rule language can exploit this structure, and express flexible rule conditions such as *object x has (not) been observed less than t seconds before/after the current time-step* or *the most frequent (currently estimated) tag around the current time-step is t* using manually defined background knowledge.

The next section will discuss the formal learning setting for relational transformation-based tagging, before discussing learning algorithms and experimental results.

4.1 Learning Setting

The relational transformation-based tagging problem can be formalized as follows:

Given

- a relational language $\mathcal{W}$ for describing sequence elements, i.e., a set of typed first-order logical predicates
- a set of tags T;
- a set of training sequences $S = \{s_1, ..., s_m\}$ with sequence elements described in $\mathcal{W}$ and corresponding true tags L over T;
- a scheme for setting initial tags given by a function $init$;
- a language $\mathcal{L}$ of transformation rules $t' \leftarrow t : q$ where $t, t' \in T$, $q = l_1, ..., l_r$ and the l_i are atoms in $\mathcal{W}$.

Find an ordered lists of transformations $R = [R_1, ..., R_l]$, $R_i \in \mathcal{L}$, such that applying the initial tagging scheme and afterwards transformation rules $R_1, ..., R_l$ minimizes

$$error(\hat{L}) = \sum_{s \in S} \sum_{i=1}^{n_s} \delta(l_{is}, \hat{l}_{is})$$

where n_s is the length of sequence s and l_{is}, $\hat{l}_{is}$ denote the tag assigned to element i in sequence s according to L and $\hat{L}$.

In contrast to standard (propositional) transformation-based tagging approaches, the languages $\mathcal{W}$ (sequence elements) and $\mathcal{L}$ (rules) employed are relational; that is, rule conditions q are first-order queries of the form $l_1, ..., l_k$ where the l_i are first-order logical atoms. Applying a first-order transformation rule $t' \leftarrow t : q$ means simultaneously replacing all tags t in $\hat{L}$ by t' wherever the first-order context constraint q matches the relational description of the corresponding sequence element.

Example 3. As an example rule in the ADL recognition domain consider

$$FlavorTea \leftarrow EatCereals :$$
$$sensor(X, spoon), near(X, sugar, 10), not(near(X, bowl, 5))$$

where the variable X is bound to the sequence element under consideration and the background predicate $near(X, bowl, 5)$ is true if the object $bowl$ has been observed within ± 5 seconds from X. This rule re-tags objects of type spoon from $EatCereals$ to $FlavorTea$ if implied by the context.

4.2 A Branch-and-Bound Learning Algorithm

For learning the list R of relational transformation rules, a large space of possible rules has to be searched. However, structure on the search space can be exploited to make this search more efficient. More specifically, the algorithm we use combines ideas from transformation-based learning (branch-and-bound search based on upper bounds for the error reduction of a transformation rule) and inductive logic programming (refinement search in a generalization/specialization lattice). It is closely related to the algorithm presented in [2].

Recall that the goal of learning is to find a list R of transformation rules which minimize $error(\hat{L})$ on a set of training sequences S with known true labels L. As in propositional transformation-based learning [1], the rule list is learned greedily: starting with an empty list, the algorithm incrementally adds one rule after the other, at every step selecting the rule which yields the greatest reduction in $error(\hat{L})$ and updating the current tagging $\hat{L}$ (cf. Algorithm 1).

When searching for an individual rule with maximum error reduction, a significant part of the search space can be pruned away by computing upper bounds for the error reduction a rule can achieve. One obvious bound for the reduction achievable by a transformation rule $t^i \leftarrow t^j : context$ is given by the number of sequence elements whose true tag (in L) is t^i and which are currently (in $\hat{L}$) assigned tag t^j. Let $\mathcal{M}$ denote the current confusion matrix, i.e., $\mathcal{M}[i, j]$ denote the number of sequence elements with true tag t^i currently tagged as t^j. This can be exploited by considering rules $t^i \leftarrow t^j : context$ in (decreasing) order of their potential $M[i, j]$ for error reduction and keeping track of the best error reduction Δ_{best} found so far. All rules of the form $t^i \leftarrow t^j : context$ with $\mathcal{M}[i, j] \leq \Delta_{best}$ can be removed from consideration (cf. [1]).

This idea can be taken one step further if it is combined with a general-to-specific search for the first-order constraint $context$ [2]. As a complete search in the space of first-order constraints is infeasible in most cases, a greedy general-to-specific search is performed. Specializations of the current condition q are generated by a so-called refinement operator ρ. For our purposes, the refinement operator specializes a condition $q = l_1, \cdots, l_n$ simply by adding a new literal l to the clause yielding $h \leftarrow l_1, \cdots, l_n, l$. This operator is monotone in the sense that for $q' \in \rho(q)$ the number of matches in the data can only decrease. Consequently, the maximum gain achievable from specializations of a transformation rule $t^i \leftarrow t^j : q$ can be bounded in terms of the current matches. More specifically, assume that a constraint q matches on a number of sequence elements in the training data S, and that for p_q of these it has a positive effect (current tag is t^j, but true tag is t^i) and for n_q it has a negative effect (current and true tag are t^j). The error reduction of applying the transformation $t^i \leftarrow t^j : q$ is $\Delta_q = p_q - n_q$. It is now obvious that no specialization $t^i \leftarrow t^j : q'$ with $q' \in \rho^*(q)$ can achieve an error reduction greater than $\Gamma_q = p_q$.

Algorithm 2 Branch-and-bound algorithm for relational transformation-based tagging

rtb-tagging(input: sequences S; true sequence tags L; language bias $\mathcal{L}$)

```
1  L̂ := initial-tags(S, L)
2  initialize R := []
3  repeat
4        initialize Δ_best := 0
5        compute M := confusion-matrix(L̂, L)
6        for all i, j ∈ {1, ..., k}, i ≠ j, sorted by M[i, j] descending do
7              initialize Γ := M[i, j]
8              initialize q := true
9              while (Γ > Δ_best) do
10                   for all q' ∈ ρ(q, L) do
11                         compute Δ_q' := error-reduction(t^j ← t^i : q')
12                         compute Γ_q' := max-reduction(t^j ← t^i : q')
13                   end for
14                   let q := argmax_q' Δ_q'
15                   let Δ_best := max(Δ_best, Δ_q)
16                   let Γ := Γ_q
17             end while
18       end for
19       let r := t^i ← t^j : q be a rule with error reduction Δ_best
20       update L̂ := apply-rule(L̂, r)
21       update R := append(R, r)
22   until (no improvement)
23 return R
```

A greedy branch-and-bound algorithm exploiting these two bounds is outlined in Algorithm 2. It takes as input a set of training sequences S, true sequence tags L, and the language bias $\mathcal{L}$. The algorithm starts with an empty rule list R and initial tags assigned in $\hat{L}$. Transformation rules are then greedily added to R, and their effect applied to the current tagging $\hat{L}$ (lines 3–21). Transformations are considered in order of decreasing $\mathcal{M}[i, j]$ (line 6). At every step of the search for a single transformation $t^i \leftarrow t^j : q$ (lines 6–18), the algorithm keeps track of the largest reduction Δ_{best} achieved by a rule so far. During refinements of the context constraint q (lines 9–17) a bound Γ_q for the maximum reduction that any specialization of a rule q can still achieve is computed (max-reduction), and only parts of the search space for which Γ is greater than Δ_{best} are explored.

5 Experiments

The proposed method was implemented in the RETRO (for RElatational TRansfOrmation-based tagging) system and experimentally evaluated in two real-world domains: Activity of Daily Living recognition (**ADL**) and mobile phone profile prediction (**Phone**).

Relation	Description
$sensor(Id, Object)$	The object observed at sequence element Id is $Object$
$duration(Id, T)$	The object observation at sequence element Id lasted T seconds
$close(Id, Obj, T)$	The object Obj has been observed within T seconds of sequence element Id
$time\_bin(T, Bin)$	The time span T falls into the bin $Bin \in \{short, medium, long\}$
$closest\_tag(Id, Act)$	The closest sequence position to Id for which an activity (i.e., a tag $\neq$ "no activity") is assigned in $\hat{L}$ is tagged with Act
$close\_used(Id, Act, T)$	Less than T seconds away from sequence element Id an object has been observed which is typically used in Act

Table 1. Example relations used to describe the activity data. Some relations are directly derived from the data (e.g. $sensor$, $duration$, $close$), others include human-supplied prior knowledge (e.g. $close\_used$).

Relational Representation

$cell(w_1, 6672)\ cell(w_2, 6671)\ cell(w_3, 6673)$ ...
$time(w_1, 1, 15)\ time(w_2, 16, 25)\ time(w_3, 26, 38)$ ...
$usr\_activity(w_1, act)\ usr\_activity(w_2, idle)\ usr\_activity(w_3, act)$ ...
$active\_app(w_1, 101)\ active\_app(w_1, 102)\ active\_app(w_3, 101)$ ...
$comm(125, sms, incoming)\ comm(390, call, outgoing)$ ... ...

Phone profile	normal		silent			normal		meeting	
Cell	6672	6671	6673	7409	6673	6671	7409	7410	6739

Fig. 3. Illustration of the Phone data (predicates for cell location, duration, user activity, active applications, and communication events).

In the ADL recognition domain, object-interaction data for a user having breakfast at home has been gathered by a wearable RFID reader and RFID tags on objects such as milk, cereals, kettle, water tap, cutlery etc. (23 objects in total). The stream of tags picked up by the RFID reader indicates which object is close (approximately 10–15 centimeters) to the wrist of the user at a particular point in time. A single object observation is returned at every second—if several tags are within reach, one is returned randomly. Note that the data is relatively noisy: tags might sometimes be missed, or a tag not related to a particular activity can be reported by the reader because the corresponding object is accidentally close. The task is to predict the current activity performed, out of a set of 24 possible activities such as boiling water, toasting bread, reading a newspaper or "no activity". The sequence data obtained from the RFID reader is represented in a relational form by collapsing identical observations into one observation with a starting point and duration in time (cf. Figure 1 for an illustration). Furthermore, additional background predicates have been defined, see Table 1 for examples.

In the Context Phone domain, data about user communication behavior has been gathered using a software running on Nokia Smartphones. The software automatically logs communication and context data, such as the current provider cell, incoming and outgoing calls and text messages, and other phone status information. The task is to

Algorithm	**ADL**	**Phone**
Majority tag	19.5 ± 22.3	56.7 ± 13.1
HMM Tagger	74.9 ± 12.5	56.7 ± 13.1
RETRO	75.4 ± 7.8	67.7 ± 10.3

Table 2. Average F-measure on the ADL Recognition and Phone problems based on a leave-one-sequence-out cross-validation.

Learned Rules
$ObtainNewspaper \leftarrow ReadNewspaper$: $close(Id, Obj, T), Obj = door,$ $time\_bin(T, medium)$
$FlavorTea \leftarrow EatCereals$: $closest\_tag(A, FlavorTea)$
$SteepTeaBag \leftarrow DrinkTea$: $close(Id, Obj, T), Obj = stove$
$PourCereal \leftarrow ObtainNewspaper$: $close\_used(Id, PourCereal, T),$ $not(close\_used(Id, ObtainNewspaper, T')), time\_bin(T, short)$
$SteepTeaBag \leftarrow noActivity$: $duration(Id, T), time\_bin(T, long),$ $closest\_tag(ID, SteepTeaBag)$

Table 3. Examples for rules learned by RETRO on the ADL dataset.

predict the active profile of the phone (silent, meeting, or normal) at every point in time. See Figure 3 for an illustration of the data and the predicates used.

For comparison, we have also conducted experiments with a (propositional) HMM tagger on the two datasets. As it is not possible to encode all relevant information propositionally, we have selected the most relevant information to be used as the propositional alphabet W. For the ADL recognition problem, this is the sequence of objects observed. For the phone domain, it is the sequence of cells the phone was located in.

For initializing the tagging $\hat{L}$ in the transformation-based tagger, RETRO simply assigns the most frequent tag given the propositional symbol $w \in W$:

$$\text{init}(w) = \underset{t \in T}{\text{argmax}}\, C(w, t)$$

where $C(w, t)$ is the number of times symbol w was tagged with t in the training data. More elaborate initialization schemes (such as using the HMM tagging as an initialization for the transformation-based tagger) are an interesting direction for future work. Furthermore, instead of a simple greedy search as outlined in Algorithm 2, a beam search with beam size $K = 10$ is used. The main loop of the algorithm is terminated if no rule with a gain of at least $min\_gain = 10$ is found.

Table 2 lists the average F-measure for RETRO and HMM tagging based on a leave-one-sequence-out cross-validation. For the ADL recognition problem, there is no significant difference between the two approaches. In the phone domain, the HMM tagger fails to improve upon the majority tag prediction, while RETRO yields a (borderline) significant increase in F-measure (paired sampled t-test, $p = 0.051$). This shows that transformation-based approaches can be competitive with probabilistic methods in complex tagging domains. However, the presented experiments are still preliminary, and more empirical evaluation is needed to assess the potential of the method in more

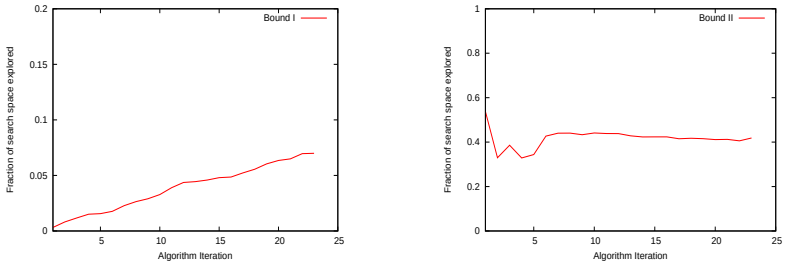


Fig. 4. Effectiveness of the two pruning schemes Bound I (maximum gain attainable from changing a certain tag into a certain other tag) and Bound II (maximum gain attainable from specializing a given rule). Results are averaged over a leave-one-sequence-out cross-validation.

detail. Note furthermore that although HMM tagging is a standard approach in activity recognition, more advanced probabilistic methods have recently been developed which would possibly yield slightly higher accuracy in this domain [9].

Examples for rules learned by RETRO on the ADL recognition task are shown in Table 3. For instance, consider the last rule: it encodes that if a sequence element corresponding to a long object observation is tagged with $noActivity$ and the closest currently predicted activity is $SteepTeaBag$, this sequence element should also be tagged with $SteepTeaBag$. This rule is useful for "filling in gaps" as $SteepTeaBag$ only causes characteristic object observations at the beginning and end of the activity.

Finally, Figure 4 visualizes the effectiveness of the pruning schemes based on the two upper bounds discussed above on the ADL recognition problem. More specifically, Figure 4 (left) shows the fraction of pairs (t^i, t^j) that have to be considered when searching for rules $t^i \leftarrow t^j$ in lines 6–18 of Algorithm 2 as a function of the algorithm iteration. This pruning scheme is very effective, reducing the search space by 93%–99%. It is more effective in earlier iterations as it is easier to find a rule with yields a large reduction in error. Figure 4 (right) shows which fraction of refinements is removed from the beam when rules are refined in lines 10–13 of Algorithm 2 because no further specialization can reach the performance of the best rule found so far. Note that this form of pruning does not affect the computational complexity of the algorithm but rather allows a more thorough search through the space of possible rules (given a limited beam size) by effectively reducing the branching factor of the search. On average, the branching factor is about halfed, this is independent of the algorithm iteration.

6 Conclusions and Related Work

Motivated by the needs of activity recognition problems, we have introduced a relational transformation-based tagging system. It tightly integrates principles of inductive logic programming (especially search, representations, operators, background knowledge) with transformation-based tagging (error-driven search, branch-and-bound idea). The approach has been evaluated on two activity recognition data sets and the results are competitive with those of a Hidden Markov Model approach. Perhaps more important than the experimental results obtained so far is the ease with which one can extend the transformation-based tagging approach beyond the propositional HMM setting. Important directions in this regard include: the use of rich sources of background knowledge (that take not only into account the inputs but also the already available produced tags), the prediction of structured output sequences (predicting sequences of logical atoms, cf. [10], such as *call(anna,10)* denoting the prediction that *anna* will be called in 10 minutes), and relaxing the purely sequential nature of the output (which is important for the ADL dataset where different activities may overlap in time, and therefore ordering them is not always possible).

Acknowledgments We would like to acknowledge support for this work from the Research Foundation-Flanders (FWO-Vlaanderen).

References

1. Brill, E.: Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. Computational Linguistics **21**(4) (1995) 543–565
2. Dehaspe, L., Forrier, M.: Transformation-based learning meets frequent pattern discovery. In Cussens, J., ed.: Proceedings of the 1st Workshop on Learning Language in Logic, Bled, Slovenia (1999) 40–51
3. Patterson, D., Fox, D., Kautz, H., Philipose, M.: Fine-grained activity recognition by aggregating abstract object usage. In: Proceedings of ISWC 2005, Osaka (2005)
4. Raento, M., Oulasvirta, A., Petit, R., Toivonen, H.: ContextPhone - a Prototyping Platform for Context-aware Mobile Applications. IEEE Pervasive Computing **4**(2) (2006) 51–59
5. Kersting, K., De Raedt, L., Raiko, T.: Logical hidden markov models. Journal of Artificial Intelligence Research **25** (2006) 425–456
6. Rabiner, L.: A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE **77**(2) (1989) 257–286
7. Wilson, D., Philipose, M.: Maximum a posteriori path estimation with input trace perturbation: Algorithms and application to credible rating of human routines. In: Proceedings of IJCAI 2005, Edinburgh, Scotland (August 2005)
8. Landwehr, N., De Raedt, L.: r-grams: Relational grams. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India (2007) 907–912
9. Wang, S., Pentney, W., Popescu, A.M., Choudhury, T., Philipose, M.: Common sense based joint training of human activity recognizers. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence. (2007) 2237–2242
10. Kersting, K., De Raedt, L., Gutmann, B., Karwath, A., Landwehr, N.: Relational sequence learning. In De Raedt, L., Frasconi, P., Kersting, K., Muggleton, S., eds.: Application of Probabilistic ILP. Springer (2007) to appear.

Enhanced Anytime Algorithm for Induction of Oblivious Decision Trees

Mark Last[1], Albina Saveliev[1]

[1]Department of Information Systems Engineering, Ben-Gurion University of the Negev, POB 653, Beer-Sheva, 84105 Israel
{mlast, albinabu}@bgu.ac.il

Abstract. Real-time data mining of high-speed and non-stationary data streams has a large potential in such fields as efficient operation of machinery and vehicles, wireless sensor networks, urban traffic control, stock data analysis etc.. These domains are characterized by a great volume of noisy, uncertain data, and restricted amount of resources (mainly computational time). Anytime algorithms offer a tradeoff between solution quality and computation time, which has proved useful in applying artificial intelligence techniques to time-critical problems. In this paper we are presenting a new, enhanced version of an anytime algorithm for constructing a classification model called Information Network (IN). The algorithm improvement is aimed at reducing its computational cost while preserving the same level of model quality. The quality of the induced model is evaluated by its classification accuracy using the standard 10-fold cross validation. The improvement in the algorithm anytime performance is demonstrated on several benchmark data streams.

Keywords: anytime algorithms, classification, information theory, Information Network algorithm, classification accuracy, computation cost

1 Introduction

Systems that deal with continuous data streams are becoming increasingly important primarily due to the emergence of sensors and similar small-scale embedded computing devices that continuously produce large volumes of data they obtain from their environment. The complex nature of real-world, streaming data has increased the difficulties and challenges of data mining applications in terms of knowledge induction and decision making within the limited time scope.

Data generated by wireless sensor networks (WSN) is one of the important examples. WSN are now used in many application areas including environment and habitat monitoring, health care, home automation, and traffic control. Each sensor node of such network records as streams time-stamped observations, taken at varying time frequency. A typical observation includes measurements of various physical or environmental parameters such as temperature, sound, vibration, pressure, as well as sensor location. While real-time tracking of environmental conditions is extremely

important for handling a chemo/bio contamination, seismic detection etc., continuous transmission of *all* recorded observations by the meter-reading chips to the nearest hub node and, subsequently, to the central station may be infeasible due to the limited battery life of the chips and the local hubs. The intuitive solution is to use data-mining techniques to analyze and induce time-dependent models of observed behavior and transfer these models to the central station rather then the streamed data. At the same time, the high rate of data changes requires to generate the model rapidly within the allocated time frame.

The *anytime algorithms* give intelligent systems the capability to trade computational time for the quality of results. This capability is efficient for solving time-constrained problems such as decision making in dynamic environment, sensor interpretation, and planning [17]. The term *anytime algorithm* was introduced by Dean et al. in the mid-1980s in their work on time-dependent planning [4], [5]. Similar approaches termed *flexible computation* by Horvitz [10], [11] and *imprecise computation* by Liu et al. [15] are based on a general idea that many computational tasks are too complicated to be completed at real-time speeds, therefore it is important to build a system that can generate good approximate results in a much shorter time period.

According to Zilberstein [17], the desired properties of anytime algorithms include the following: *measurable solution quality*, which can be easily determined at run time, *monotonicity* (quality is a non-decreasing function of time), *consistency* of the quality w.r.t computation time and input quality, *diminishing returns* of the quality over time, *interruptibility* of the algorithm, and *preemptability* with minimal overhead.

In this paper, we propose a new, enhanced version of an anytime algorithm for inducing a classification model called Information Network (IN). The original algorithm was developed by Last et al. [13]. The model is a tree-like structure that represents relationship between input (predictive) features and target (classification) attributes. Unlike most other decision-tree models, the information network uses the same input attribute across all nodes of a given layer (level) and thus it can be considered an *oblivious decision-tree*. The method was shown theoretically and empirically to have the basic properties of interruptible anytime algorithms [12]. The enhanced method presented in this paper is aimed at improving the anytime performance of the IN algorithm by reducing its computational time while maintaining the same quality level of the induced model. The most time-intensive operation in network construction is choosing, at each iteration of the algorithm, an input attribute, which provides the maximum significant increase in *mutual information* relative to the previous layer. Therefore the idea is to filter out the least significant attributes, before the classifier construction, and afterwards to build a model using a reduced subset of candidate input attributes. We evaluate the performance of the algorithm on eleven benchmark datasets from various sources (see Section 4).

The paper is organized as follows. Section 2 reviews the related works in the fields of anytime classification algorithms and resource-aware knowledge discovery in data streams. The enhanced anytime algorithm for induction of oblivious decision trees is described by us in Section 3. Experimental results are presented and discussed

in Section 4. Finally we conclude the paper and present the possible future research directions in Section 5.

2 Related work

2.1 Anytime Decision Tree Induction

Last et al. [12] introduced an interruptible anytime information-theoretic classification algorithm. Their method constructs a compact and accurate decision-tree model called Information Network. The algorithm has several objectives, such as: maximizing the mutual information between a set of predictive attributes and the target (classification) attribute, finding a minimal set of features involved in the induced model (hence, it can be also used as a feature selection method), and verifying the statistical significance of the discovered patterns.

Esmeir et al. [6] presented interruptible anytime algorithms for iterative improvement of decision trees. The motivation of their research is different from our goal of saving the computational resources. They explore the problem of how to produce better decision trees for hard-to-learn concepts when more time resources are available. Their framework consists of two anytime algorithms. The first one, called *Sequencing LSID3* converts the recent LSID3 contract algorithm to an interruptible version, which does not require the allocated time in advance and can be interrupted at any time. The second is *Interruptible Induction by Iterative Improvement* (IIDT) which repeatedly selects a sub tree whose reconstruction is estimated to yield the highest marginal utility and rebuilds it, exploiting extra time allocation.

2.2 Resource-aware Data Mining Techniques

Gaber et al. [7] presented a framework for resource-aware computing in data stream analysis. The streaming information is often generated, received or processed by computational devices such as wireless sensors. These devices are limited in terms of energy, memory, computational speed and communication bandwidth. The main goal of the research is to apply data mining techniques to continuous data streams within the scope of constrained device resources. This generic framework proposes Algorithm Granularity Settings (AGS). The idea is to periodically change algorithm settings from the input, output, and/or processing end points according to resource consumption pattern measurements performed over the last time period as well as a measure of resource criticality. In [7] this method is applied to a novel threshold-based micro-clustering algorithm, called RA-Cluster. The strategy of adapting the CPU demand is done using the *Randomized Assignment* approach. As the CPU load

increases, only a pre-specified fraction of the current micro-clusters is examined when making the micro-cluster assignment for a new data point.

Phung et al. [16] extended the previous work [7] for Wireless Sensor Networks. Their approach was applied to online clustering algorithm (ERA-Cluster), which uses the resource monitoring of the Sun SPOT sensor nodes from Sun Microsystem™ to adapt to resource availability. The CPU adaptation of [16] is also based on the Randomized Assignment approach.

2.3 Anytime Properties of the IN Algorithm

If the network quality is measured by its predictive accuracy, we can easily verify the algorithm conformity with the anytime properties defined by Zilberstein [17] using a line of arguments similar to [12]:

- *Measurable quality*. The predictive accuracy after each iteration of the algorithm can be estimated using 10-fold cross-validation or any other validation procedure.
- *Recognizable quality*. Due to the inherent compactness of IN models, counting the number of validation errors is a relatively fast procedure.
- *Monotonicity*. A new attribute is added by the algorithm to the set of input attributes only if it causes an increase in the mutual information. According to Fano's inequality [3],an increase in mutual information implies an expected decrease in the error rate.
- *Consistency*. The theoretical run time of the algorithm has been shown by us in [13] to be quadratic-logarithmic in the number of records and quadratic polynomial in the number of initial candidate input attributes.
- *Diminishing returns*. This property is very important for algorithm's practical usefulness: it means that after a small part of the running session, the results are expected to be sufficiently close to the results at the completion time. We could prove this property mathematically, if we could show that the mutual information is a concave function of the number of input attributes. Though the last proposition is not true in a general case, it is possible to conclude from Fano's inequality [3] that the mutual information is *bounded* by a function, which behaves this way. This conclusion is empirically confirmed by the results of Section 4.
- *Interruptibility*. The algorithm can be stopped at any time and provide the current list of selected attributes. Each iteration forms, what is called, a *contract anytime algorithm*, i.e. the corrections of predictive accuracy are available only after termination of an iteration.
- *Preemptability*. Since the algorithm maintains the training data, the list of selected input attributes, and the current structure of the information-theoretic network, it can be easily resumed after an interrupt. If the suspension is expected to be long, all relevant information may be stored on a hard disk.

3 Enhanced Algorithm for Anytime Induction of Oblivious Decision Trees

We aim at enhancing the Information Network algorithm by reducing the time needed to construct a classification model, while maintaining the same level of its predictive accuracy. At each iteration, the algorithm builds a new *hidden* layer by choosing an input attribute (either discrete, or continuous), which provides the maximum significant increase in mutual information relative to the previous layer. The computational complexity of evaluating a discrete attribute is the complexity of calculating its conditional mutual information $MI(A_i;T/z)$ **(1)**. The complexity of evaluating a continuous attribute consists of calculating its conditional mutual information MI (Th; T/S, z) for a given split **(2),** as well as discretizing it into a number of discrete intervals. Both these operations are performed in each hidden layer of information network for all candidates in that layer. Hence, to reduce the computational cost of the Information Network algorithm we propose the following "fast feature filtering" procedure to be applied before the network construction:

- Generate a random sample of training instances. The sample size is a pre-specified percentage of the training examples. Based on the experimental results described in Section 4, the recommended sample size can be as low as 5%.
- Compute the estimated mutual information for each candidate input attribute using the random sample of training instances. Due to the small sample size (5%), this calculation is expected to take much less time than the first iteration of the algorithm based on the entire training set. The mutual information calculated by the IN algorithm is shown in [14] to be a much more efficient feature selection method than two alternative feature selection algorithms (Relief and ABB).
- Filter out the least significant features, having the lowest values of estimated mutual information. The percentage of selected features is determined in advance. Based on the experimental results, described in Section 4, the recommended percentage is 30%, i.e., 70% of significant input attributes are removed from consideration by the algorithm. We call this approach *Fast Feature Filtering* (FFF).

The Information Network induction is performed subsequently on the subset of selected features using all training examples.

The pseudocode of the "fast feature filtering" procedure is given below:

Input: the set of n training instances; the set CI of m candidate input attributes (discrete and continuous); the target (classification) attribute T; the percentage of randomly selected training instances *sample_size*; the percentage of selected attributes from m candidate input attributes *significant_Set_size*.

Output: a set *I* of selected significant input attributes.

```
  I = ∅
  Create random sample of sample_size training instances.
  For each candidate input attribute A_i ∉ I do
      If A_i is discrete then
            Return the statistically significant
conditional mutual information cond_MI_i
between A_i and T.
      Else return the best threshold splits of A_i and the
statistically significant conditional
mutual information cond_MI_i between A_i and T.
      If cond_MI_i > 0, then
            Update the set I of selected input
attributes: I = I ∪ A_i.
  End do
  Sort the set I of selected input attributes according
to increasing its cond_MI_i
  For each i ← significantSet_size to |I|
      Exclude the less significant input attribute A_i from
the set I:
      I = I - A_i.
      i ← i + 1;
  End do
  Return a set I of selected significant input
attributes.
```

4 Experimental Results

According to [17], the performance profile (PP) of an anytime algorithm denotes the expected output quality as a function of the execution time *t*. Since there are many possible factors affecting the execution time, the performance profile, in many cases, has to be determined empirically and not analytically.

To study the performance profile of the enhanced method for induction of oblivious decision trees, we have applied it to eleven real-world datasets, including five datasets (Housing, Image Segmentation, Spambase, Waveform, Adult) from the UCI Machine Learning Repository [1], five Traffic Direction datasets provided by the Traffic Control Center of Jerusalem, and the Intrusion Detection database originally used for the Third International Knowledge Discovery and Data Mining Tools Competition (current available from the UCI KDD Archive [8]).The characteristics of each dataset are shown in Table 1. The size of the datasets varies between 506 and 10,000 cases. The total number of candidate input attributes is from 11 up to 57, including nominal and continuous features. It should be noted that the Traffic Direction, Intrusion Detection and Adult datasets have actually more than 10,000 instances, but due to the memory constraints we have confined ourselves to this amount of training examples.

We have measured the quality of the induced model by the standard 10-fold cross validation procedure. To evaluate the attribute filtering method we have experimented with three different sample sizes of 5%, 10% and 20% accordingly.

Using each sample of the training set, we have calculated the *mutual information* for all candidate input attributes and selected 20%, 30%, 40%, 50%, 60% and 70% of the most significant features. With each subset of selected significant attributes, we have built 10 *Information Networks,* using the ten-fold cross validation procedure. This experiment has been repeated eighteen times for each dataset, using six different amounts of selected attributes and three different samples of the training set. The results of each experiment, which are the averages of 10 cross-validation models, are compared to the results of the original method (not using fast feature selection). After each iteration of the algorithm, we have computed the accuracy of the current model and the time needed to induce the new hidden layer of that model. These parameters are compared with the same parameters of the original algorithm, which induces a classification model from all candidates, without filtering out less significant attributes.

Based on the results of experiments we can say that on average, only three *hidden* layers are built in all 10 models over 11 datasets. We have found also, that after the third iteration the cross-validation accuracy of most models stops to increase significantly (see the "simplicity first" approach proposed in [9]). Measuring the run time and the predictive accuracy of the enhanced algorithm over three different sample sizes (5%, 10%, 20%), we have found that the 5%-sample preserves the same performance level as the larger samples. Considering these facts we have presented in Figure 2 the performance profile of only three-layered networks induced from various sets of significant attributes selected by a 5% random sample. To simplify the comparison of the results of the novel approach with the original one, as well, for better illustration, we have normalized the execution time of each experiment with respect to the execution time of the original algorithm. For the run time equal to zero, the average accuracy over 11 datasets is computed by means of the majority rule.

Several important observations can be made from Figure 2. First, we can see, that the average performance profiles are *concave* functions of time. After the first iteration of the algorithm, the accuracy of the model is sufficiently close (85%) to the accuracy at completion time. It proves the very important anytime property of the algorithm: diminishing returns (see subsection 2.3). Second, we can observe that execution time of the enhanced approach varies between 20-50% of run time using the original method, where the lowest computational time of 20% refers to induction of the model from 30% of selected significant attributes and the highest time of 50% refers to construction of the model from 70% subset accordingly. Finally, we note that with a 20% subset of selected features, the induced model has only two layers in eight datasets out of eleven (Housing, Adult, five Traffic Direction datasets, and Intrusion Detection). Hence, we exclude the 20% subset of significant attributes from our study, and compute the average performance for a three-layered network, regarding this network as a minimal model in all 11 datasets.

The run time of the enhanced method with the 5% sample starts with 93.8 msec. for the Traffic-Direction2 datasets and goes up to 87,895 msec. for the Spambase dataset, which has 4,601 records and 57 continuous attributes. Due to space limitations, Figure 3 shows the performance profiles of five datasets only.

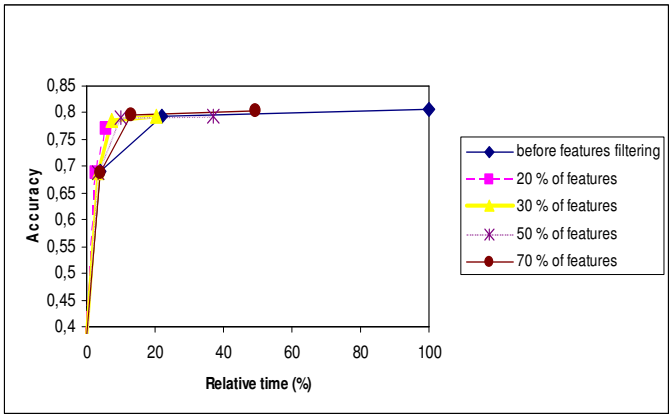


Figure2. Average performance profile of the enhanced anytime algorithm over eleven datasets, sample size 5%.

Our research is primarily aimed at reducing the computational time of the IN algorithm while keeping the same quality level of the classification model. To study how the sample size affects the accuracy and the execution time of constructing the Information Network, the average value of these parameters have been calculated for each sample size (see Table 2)

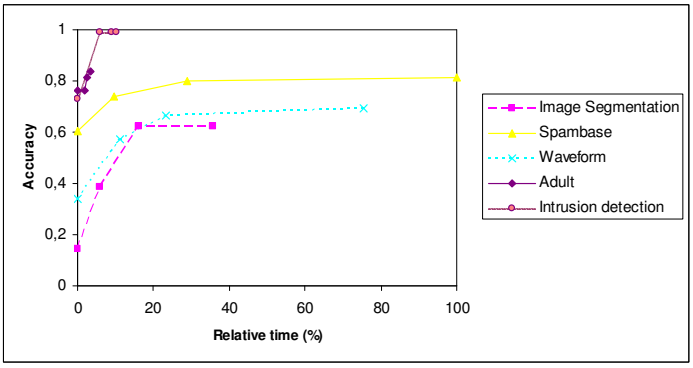


Figure3. Performance profiles of the enhanced anytime algorithm for five datasets, sample size 5%

Table 1. The characteristics of eleven benchmark datasets

Dataset	Data size	Classes	Continuous	Nominal	Total Attributes
Housing	506	3	12	1	13
Image Segment.	2,100	7	19	0	19
Spambase	4,601	2	57	0	57
Waveform	5,000	3	21	0	21
Adult	10,000	2	6	8	14
Traffic-Direction1	10,000	4	6	5	11
Traffic-Direction2	10,000	4	6	5	11
Traffic-Direction3	10,000	4	6	5	11
Traffic-Direction4	10,000	4	6	5	11
Traffic-Direction5	10,000	4	6	5	11
Intrusion Detect.	10,000	4	14	2	16

Table2. Average accuracy, execution time and standard deviation of three-layered model over eleven datasets and various percentages of selected significant attributes

Sample Size (%)	Average attributes filtering time (sec.)	Average accuracy	Average execution time (sec)	STDEV of mean accur.	STDEV of mean time	Slope ($*10^{-4}$)
5	1,8	0.79	31,7	0.013	6	2.5
10	1,8	0.80	32,1	0.013	6	2.49
20	2	0.79	32,2	0.014	6	2.45

As one can see from Table 2, the sample size affects the induction time of the classifier and does not affect its accuracy. To evaluate the trade-off between these characteristics we calculate their ratio called the *Slope* using the following equation:

$$\text{SLOPE} = \frac{\Delta Q(t)}{\Delta t} \qquad \textbf{(3)}$$

Where,

$\Delta Q(t)$ = the difference between the accuracy of the complete (three-layered) model and the initial (majority rule) accuracy;

Δt = the execution time of inducing a complete (three-layered) model

According to the value of *Slope* we can suggest that the 5% sample size is slightly more preferable than the 10% and 20% sample sizes.

Another question is which percentage of selected significant attributes is preferable for optimizing the accuracy-time relationship. To answer this question, we

are summarizing in Table 3, the average accuracy and execution time, for each subset of significant attributes, comparing these parameters to the results of the original method, without the fast feature filtering (FFF), where the average accuracy is 0.806 and execution time is 88,115 msec.

The decrease in accuracy and execution time (see Table 3, columns 2 and 3) is computed relative to the 100 % set of candidate attributes. As we can see, the maximal reduction of time (79.9%) is reached with the 30% set. It is important to note that, the decrease in accuracy vs. the original method (see Table 3, column 5) has not been found statistically significant as for various sample sizes, as for various percentages of selected attributes. To find the optimal percentage of significant features we have calculated the *Slope* for each subset of selected attributes. According to the *Slope* value we can say that the 30% percentage of significant features is optimal for accuracy-time optimization task.

Finally, we can conclude, based on analysis of the experimental results obtained for eleven datasets that best trade-off between the accuracy of the three-layered Information Network and computational time needed for its construction is achieved on a 30% subset of significant attributes selected by a 5% random sample. In this case, the execution time is reduced by almost 80%.

Table 3. Average accuracy, execution time and standard deviation of three-layered model, over eleven datasets and various sample sizes

Percent signif. attrib.	Aver. accur. after FFF	Aver. time (sec.) after FFF	Slope ($*10^{-4}$)	Decrease accur. after FFF (%)	Decrease time, after FFF (%)	STDEV of mean accuracy after FFF	STDEV of mean time, after FFF
30	0.788	17,6	4.38	4	80	0.018	5
40	0.792	25,7	3.08	2	71	0.018	6
50	0.793	33,2	2.39	2	62	0.018	8
60	0.801	39,3	2.04	1	55	0.015	9
70	0.804	44,0	1.83	0.3	50	0.015	11

One of the important benefits of the proposed FFF approach is that it allows capturing the tradeoff between the solution quality and the time saved and/or complexity of classification represented by the number of the most significant input attributes. The anytime interruptability of the algorithm allows stopping it after each iteration to provide an approximate solution that is close to the complete result. This can be crucial for real-time classification algorithms working with a large number of input attributes and/or with timing constraints.

5 Conclusions

In this paper, we have proposed a new, "Fast Feature Filtering" version of an anytime algorithm for constructing a classification model called Information Network (IN). We have studied and improved the important anytime property *diminishing returns* of the algorithm. The new method enables to reduce significantly its computation cost while preserving the same level of model quality. This goal is achieved by means of monitoring the relationship between the random sample size of training examples and the percentage of most significant input attributes selected by this sample. The proposed algorithm is evaluated on eleven benchmark datasets available from different sources. The quality of the induced model is measured by its classification accuracy using the standard 10-fold cross validation. The performance profiles of the new version have been shown to be concave functions of time. Based on the experimental results, the optimal tradeoff between accuracy of a three-layered Information network and execution time needed for its construction is achieved with a 30% subset of significant attributes selected using a 5% random sample. In this case, the accuracy rate is very close to the accuracy of the original algorithm, whereas the execution time is reduced by almost 80%. Topics for future research include predicting the expected quality for a given execution time (and vice versa), and integrating the enhanced version of the algorithm with real-time learning systems such as IOLIN [2].

References

1. Blake, C.L., Merz, C.J. UCI Repository of machine learning databases, http://www.ics.uci.edu/~mlearn/MLRepository.html, 1998.
2. Cohen, L., Avrahami, G., Last, M, Kandel, A., Kipersztok, O. "Real-Time Data Mining of Non-Stationary Data Streams from Sensor Networks", Information Fusion Journal, Special Issue on Information Fusion in Distributed Sensor Networks, in press. doi:10.1016/j.inffus.2005.05.005.
3. Cover, T.M., Thomas J.A., Elements of Information Theory, Second edition, Wiley, 2006
4. Dean, T.L., 1987. Intractability and Time-Dependent Planning. In *Reasoning About Actions and Plans*, Georgeff M.P., Lansky A. L., Eds. Morgan Kaufmann Publishers, San Francisco, California, 1986, pp. 245-266.
5. Dean, T.L, Boddy, M., An Analysis of Time-Dependent Planning, In *Proceedings of the American Association for Artificial Intelligence Conference (AAAI-88)* (Cambridge, Massachusetts, 1988), AAAI, MIT Press, pp. 49-54.
6. Esmeir, S., Markovitch, S., Interruptible Anytime Algorithm for Iterative Improvement of Decision Trees, In Proceedings of The 1st Workshop on Utility-Based Data Mining (UBDM-2005), held with The 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2005), pp 78-85
7. Gaber, M.M., Yu, P.S., A Framework for Resource-aware Knowledge Discovery in Data Streams: A Holistic Approach with Its Application to Clustering, in Proceedings of ACM, Symposium on Applied Computing, (SAC 2006), ACM Press, pp 649-656.
8. Hettich, S., Bay, S.D. (1999). The UCI KDD Archive [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science.
9. Holte, R.C., Very simple classification rules perform well on most commonly used datasets. Machine Learning, 11(1), pp 63-91, Apr. 1993.

10. Horvitz, E.J., Reasoning about Beliefs and Actions under Computational Resource Constraints, Proc. of the 1987 Workshop on Uncertainty in AI, Seattle, Washington, 1987, pp 429-444.
11. Horvitz, E.J., Suermondt, H.J., Cooper G.F., Bounded Conditioning: Flexible Inference for Decision under Scarce Resources. Proc. of the 1989 Workshop on Uncertainty in Artificial Intelligence, 182–193. New York: North-Holland, 1989.
12. Last, M., Kandel, A., Maimon, O., Eberbach, E., Anytime Algorithm for Feature Selection, Proceedings of the Second International Conference on Rough Sets and Current Trends in Computing (RSCTC'2000), pp. 532-539, Springer-Verlag, 2001.
13. Last, M., Maimon, O., A Compact and Accurate Model for Classification, IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 2, pp. 203-215, February 2004.
14. Last, M., Kandel, A., Maimon, O., Information-theoretic algorithm for feature selection, Pattern Recognition Letters 22(2001) pp. 799-811.
15. Liu, J.W.S., Lin, K.J., Shih, W.K., Yu, A.C., Chung, J.Y., Zhao, W., Algorithms for Scheduling Imprecise Computations, Computer, vol.24 no.5, pp.58-68, May 1991.
16. Phung, N.D., Gaber, M. M., Röhm, U., Resource-aware Online Data Mining in Wireless Sensor Networks, Proceedings of the 2007 IEEE Symposium on *Symposium on Computational Intelligence and Data Mining (CIDM 2007)*, pp 139-146.
17. Zilberstein, S., Using Anytime Algorithms in Intelligent Systems, AI Magazine, vol. 17, no. 3, pp. 73-83, 1996

A Semi-Fuzzy Approach for Online Divisive-Agglomerative Clustering

Pedro Pereira Rodrigues[1,2] and João Gama[1,3]

[1] LIAAD - INESC Porto L.A.
[2] Faculty of Sciences of the University of Porto
[3] Faculty of Economics of the University of Porto
Rua de Ceuta, 118 - 6 andar, 4050-190 Porto, Portugal
pprodrigues@fc.up.pt jgama@fep.up.pt

Abstract. The Online Divisive-Agglomerative Clustering (ODAC) is an incremental approach for clustering streaming time series using a hierarchical procedure over time. It constructs a tree-like hierarchy of clusters of streams, using a top-down strategy based on the correlation between streams. The system also possesses an agglomerative phase to enhance a dynamic behavior capable of structural change detection. However, the split decision used in the algorithm focus on the crisp boundary between two groups, which implies a high risk since it has to decide based on only a small subset of the entire data. In this work we propose a semi-fuzzy approach to the assignment of variables to newly created clusters, for a better trade-off between validity and performance. Experimental work supports the benefits of our approach.

Keywords: fuzzy clustering, streaming time series, hierarchical models.

1 Introduction

The task of clustering streaming time series is not widely studied. Data streams usually consist of variables producing examples continuously over time. The basic idea behind it is to find groups of variables that behave similarly through time, which is usually measured in terms of time series similarities. Clustering time series has been already studied in various fields of real world applications. Many of them, however, could benefit from a data stream approach. For example:

- in electrical supply systems, clustering *demand profiles* (ex: industrial or urban) decreases the computational cost of predicting each individual subnetwork load [2];
- in medical systems, clustering *medical sensor data* (such as ECG, EEG, etc.) is useful to determine correlation between signals [11];
- in financial markets, clustering *stock prices* evolution helps preventing bankruptcy [7];

All of these problems address data coming from a stream at high rate. Hence, data stream approaches should be considered to solve them.

In the next section we present an overview on ODAC and its main characteristics, while Section 3 proposes the new semi-fuzzy assignment criterion. Section 4 enunciates the validity indices used in Section 5 to validate our proposal, while Section 6 presents some concluding remarks.

2 ODAC Overview

The Online Divisive-Agglomerative Clustering *(ODAC)* is an incremental approach for clustering streaming time series using a hierarchical procedure [10]. It constructs a tree-like hierarchy of clusters of streams, using a top-down strategy based on the correlation between streams. The system also possesses an agglomerative phase to enhance a dynamic behavior capable of structural change detection. The splitting and agglomerative operators are based on the diameters of existing clusters and supported by a significance level given by the Hoeffding bound [5]. Accordingly, we observe that:

- the update time and memory consumption does not depend on the number of examples, as it gathers sufficient statistics to compute the correlations within each cluster; moreover, anytime a split is reported, the system becomes faster as less correlations must be computed;
- the system possesses an anytime compact representation, since a binary hierarchy of clusters is available at each time stamp, and does not need to store anything more than the sufficient statistics and the last example to compute the first-order differences;
- an agglomerative phase is included to react to structural changes; these changes are detected by monitoring the diameters of existing clusters;
- this online system was not designed to include new streams along the execution; however, it could be easily extended to cope with this feature;
- given its hierarchical core, the system possesses a inherently adaptable configuration of clusters;

As reported by the authors, this is one of the systems clearly proposed to address clustering of multiple streams. It copes with high-speed production of examples and reduced memory requirements, with constant time update. It also presents adaptability to new data, detecting and reacting to structural drift.

2.1 Dissimilarity Measure

The system must analyze distances between incomplete vectors, possibly without having any of the previous values available. Thus, these distances must be incrementally computed. The system uses Pearson's correlation coefficient [9] between time series as *similarity* measure. This way, the *sufficient statistics* needed to compute the correlation are easily updated at each time step.

2.2 Splitting Criterion

One problem that usually arises with approximate models is the definition of a minimum number of observations necessary to assure convergence. One approach is to apply techniques based on the Hoeffding bound [5] to solve this problem. The Hoeffding bound has the advantage of being independent of the probability distribution generating the observations [3], stating that after n independent observations of a real-valued random variable r with range R, and with confidence $1-\delta$, the true mean of r is at least $\overline{r}-\epsilon$, where $\overline{r}$ is the observed mean of the samples and

$$\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2n}} \tag{1}$$

As each leaf is fed with a different number of examples, each cluster c_k will possess a different value for ϵ, designated ϵ_k.

Let $d(a,b)$ be the distance measure between pairs of time series, and $D_k = \{(x_i, x_j) \mid x_i, x_j \in c_k, i < j\}$ be the set of pairs of variables included in a specific leaf c_k. After seeing n samples at the leaf, let

$$(x_1, y_1) = \underset{(x,y)\in D_k}{\operatorname{argmax}}\, d(x,y)$$

be the pair of variables with maximum dissimilarity within the cluster c_k, and in the same way considering $D'_k = D_k \backslash \{(x_1, y_1)\}$, let

$$(x_2, y_2) = \underset{(x,y)\in D'_k}{\operatorname{argmax}}\, d(x,y)$$

Let $d_1 = d(x_1, y_1)$, $d_2 = d(x_2, y_2)$ and $\Delta d = d_1 - d_2$ be a new random variable, consisting on the difference between the observed values through time. Applying the Hoeffding bound to Δd, if $\Delta d > \epsilon_k$, one can confidently say that, with probability $1-\delta$, the difference between d_1 and d_2 is larger than zero, and select (x_1, y_1) as the pair of variables representing the diameter of the cluster. With this rule, the ODAC system will only split the cluster when the true diameter of the cluster is known with statistical confidence given by the Hoeffding bound. However, to prevent the hierarchy from growing unnecessarily, another criterion is defined in ODAC which has to be fulfilled in order to perform the splitting, which falls out of the scope of this work.

2.3 Assigning Criterion

When a split point is reported, the *pivots* are variables x_1 and y_1 where $d_1 = d(x_1, y_1)$, which are separated into each of the newly created clusters. The system then assigns each of the remaining variables of the old cluster to the cluster which has the closest *pivot*. This crisp assignment is the key object of our proposal in this work. When considering the expansion of the structure, the strict splitting of variables appears as a possible drawback, in the sense that a previous decision of moving a variable to a leaf, when there was no statistical confidence on the decision of assignment, may split variables that should be together. Left plot

of figure 1 presents an example of a possible configuration where the problem could arise. An approach based on fuzzy sets [13] would let forthcoming examples decide what to do with those variables. Section 3 introduces our proposal to deal with this uncertainty.

2.4 Aggregation Criterion

The main setting of the system is the monitoring of existing clusters' diameters. On stationary data streams, the diameter of a cluster decreases every time a split occurs. Nevertheless, usual real-world problems deal with non-stationary data streams, where time series that were correlated in the past are no longer correlated to each other, in the current time period, and might be approaching time series of other clusters. The strategy that is adopted in ODAC to detect changes in the structure is based only on the analysis of the diameters. In fact, the diameter of each two new clusters should be less or equal than their parent's diameter. In this way, no computation is needed between the variables of the two siblings.

3 ODAC with a Semi-Fuzzy Assignment Criterion

When a split point is reported, ODAC determines two variables as *pivots* and assigns each of the remaining variables to the cluster which has the closest *pivot*. This is usually a good heuristic, as it often finds an optimal border hyperplane. It is a lot faster than the heuristic performed by DIANA [6], since it is not needed to compute the average distances to decide which leaf will receive each variable. However, this may lead to erroneous situations if the moving variable is equally distant from the two pivots, there is no way of determining to which cluster it should be assigned.

This issue has a possible solution. The Hoeffding bound can be used to control the expansion of a cluster. We could include this notion of the Hoeffding bound as a decision support tool to the decision of moving a variable considering the two pivots. Let x and y be the pivots of the clusters a and b, respectively, and m be the moving variable. The expansion is decided as follows:

- if $d(y, m) - d(x, m) > \epsilon_k$ move variable to cluster a;
- else if $d(x, m) - d(y, m) > \epsilon_k$ move variable to cluster b;
- else move m into **both** clusters a and b, with a given degree of membership;

An example of application of this *semi-fuzzy* assignment is explained in the right plot of figure 1. This option may in fact allow the system to try different combinations of objects. However, this expansion eliminates the characteristic of speeding up the process with structure growth. Another example of this behavior follows when, given a *crisp* data set, the final specification is presented in figure 2 illustrating the difference between the two approaches. Top plot shows the result for *strict* clustering, where nodes 4, 5, 6 and 7 represent the real clusters defined for the *crisp* data set.

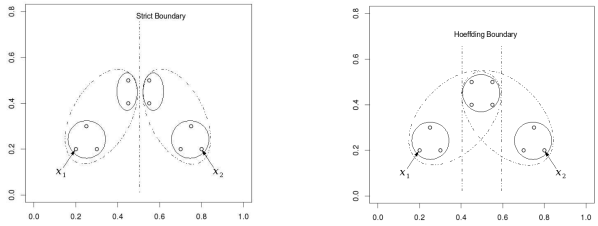


Fig. 1. Example of a dissimilarity structure between ten variables, produced by three clusters, and the comparison on the assignment method: *strict* (left) and *semi-fuzzy* (right). Variables x_1 and x_2 are the chosen *pivots* for splitting at first level this set of variables. Dot-dashed lines represent the first-level splitting while continuous lines present a second-level splitting.

The *fuzzy* approach enables a wider observation on the relations between clusters, as they appear in different configurations. However, if crisp sets are *fuzzified*, a later pruning action could be considered. The diameters of the first ancestors of the leaves or the leaves themselves could act as a post-prune criterion. Preliminary results suggest it may have a very important role in time series incremental clustering, and it is scheduled for future work. Moreover, this approach is a simple variation of the ODAC algorithm that might be useful in applications where the data reveals a fuzzy characteristic. Nevertheless, this simple technique must be complemented with the right membership function, in order to be evaluated by fuzzy clustering validity indices.

3.1 Membership Function

A *fuzzy* clustering procedure enables an object to belong to different clusters, with some membership function [8], using the concept of fuzzy sets [13]. However, in hierarchical procedures, different possibilities arise. A simple way of applying *fuzzy* clustering in ODAC is to consider that, at each split where a variable is not clearly closer to one *pivot* than the other, the degree of membership of the variable to the new clusters should be equal, assigning the variable to *both* clusters with same probability. This will enable the definition of accurate validity indices for fuzzy clustering structures.

The result of a fuzzy clustering procedure is usually defined as a matrix $U = [u_{ic}]$ where each u_{ic} is the degree of membership of a vector x_i to cluster c. In our case, the vectors are the variables. At the root level, all variables belong to one cluster, so if cluster r is the root node, $u_{ir} = 1$ for all variables i. Every time a split occurs on a cluster p, the membership of variable i to each offspring cluster c (it was assigned to) should depend on the degree of membership u_{ip}. So, our approach is to compute it as

$$u_{ic} = u_{ip} * \beta_{ip} \tag{2}$$

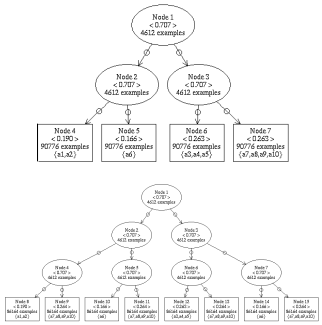


Fig. 2. ODAC structure comparison: *strict* (top) vs *fuzzy* (bottom) clustering (*crisp* data set).

	Variables									
	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10
High-level Nodes, u_{ic}										
node 1	1	1	1	1	1	1	1	1	1	1
node 2	1	1				2^{-1}	2^{-1}	2^{-1}	2^{-1}	2^{-1}
node 3			1	1	1	2^{-1}	2^{-1}	2^{-1}	2^{-1}	2^{-1}
node 4	1	1					4^{-1}	4^{-1}	4^{-1}	4^{-1}
node 5						2^{-1}	4^{-1}	4^{-1}	4^{-1}	4^{-1}
node 6			1	1	1		4^{-1}	4^{-1}	4^{-1}	4^{-1}
node 7						2^{-1}	4^{-1}	4^{-1}	4^{-1}	4^{-1}
Final Clusters, u_{ic}										
node 8	1	1								
node 9							4^{-1}	4^{-1}	4^{-1}	4^{-1}
node 10						2^{-1}				
node 11							4^{-1}	4^{-1}	4^{-1}	4^{-1}
node 12			1	1	1					
node 13							4^{-1}	4^{-1}	4^{-1}	4^{-1}
node 14						2^{-1}				
node 15							4^{-1}	4^{-1}	4^{-1}	4^{-1}

Table 1. U Matrix for the fuzzy clustering structure gathered for *crisp* data set (zero-valued cells removed).

where p is the parent cluster of c and β_{ip} is the distribution of membership due to possible fuzzy assignment of variable i when splitting cluster p. The β function can be computed based on several parameters, including the diameters of p, c and c's siblings. However, a first approach will consider $\beta = (n_{ip})^{-1}$, where n_{ip} is the number of new clusters to which variable i was assigned when splitting cluster p. For a clear strict assignment, we consider $\beta = 1^{-1} = 1$. For a fuzzy assignment after a binary split, we would have $\beta = 2^{-1} = 0.5$.

For the example presented in figure 2, the values for U are presented in table 1. It is easy to observe that the sum of membership values for each variable to all final clusters is 1. Values of U for non-leaf nodes are also presented to enable a clear insight on the splitting procedure.

4 Fuzzy Cluster Validity

Simple insights on the fuzziness of the clustering structure can be extracted using only the memberships values u_{ij}. Simple indices have been proposed such as the partition coefficient (PC) and the partition entropy (PE), defined next.

The partition coefficient index [1] is defined as

$$PC = \frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{nc} u_{ic}^2 \tag{3}$$

with range in $[1/nc, 1]$, where nc is the number of clusters. The closer the index is to 1 the crisper the clustering is. In case that all membership values to a fuzzy partition are equal, the closer the value of PC is to $1/nc$, and the fuzzier the clustering is.

The partition entropy coefficient [4] is a slight variation defined as

$$PE_a = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{nc} u_{ic} \cdot log_a(u_{ic}) \tag{4}$$

for values of U greater than zero, where a is the base of the logarithm, thus the index values range in $[0, log_a(nc)]$. The closer the value of PE is to 0, the crisp the clustering is.

Regarding the correspondence between the fuzzy clustering and the real data, a more robust criteria is the *Xie-Beni* index [12], also called the *compactness and separation* validity function. It is based on several measures of the clustering structure, with respect to the real data. The *fuzzy deviation* of variable i from cluster c, f_{ic}, is the distance between i and the center of cluster c, v_c, weighted by the fuzzy membership degree of data point i to cluster c, i. e.:

$$f_{ic} = u_{ic} \cdot d(i, v_c) \tag{5}$$

We can compute the variation of cluster c as $\sigma_c = \sum_{i=1}^{n_c} f_{ic}^2$. The sum of all the variations of final clusters, $\sigma = \sum_{c=1}^{nc} \sigma_c$, is called *total variation* of the data set. The *compactness* of cluster c is calculated as the average variation in cluster c, $\pi_c = \sigma_c / n_c$, where n_c is the number of variables belonging to cluster c. Hence the *compactness* of the whole partition is $\pi = \sigma / n$. The *separation* of the fuzzy partition, d_{min}, is defined as the minimum centroid linkage between any two clusters. The index is defined as

$$XB = \pi / d_{min} \tag{6}$$

Small values of XB are expected for compact and well-separated clusters. However, attention should be paid as it monotonically decreases with the number of clusters nc.

5 Experimental Evaluation

In order to compare the *fuzziness* of the structures gathered by the system, we have applied the algorithm to a real data set published in the 2004 ICML Physiological Data Mining Competition, which has no clear crisp structure. Visually, we can only stress that there is high correlation between *sensor3* and *sensor5*. For the data belonging to *user01* (93344 observations), presented in figure 3, it is easy to note that only *sensor3* and *sensor5* are clearly correlated. The resulting ODAC structure is the left-most plot of figure 5, and the membership values are presented in table 2. The indices for both the *crisp* and *user01* data sets present concordant directions. While the *crisp* data set was partitioned with $PC = 0.650$ and $PE_2 = 0.900$, the *user01* set produced a partitioning with $PC = 0.594$ and $PE_2 = 0.875$. One can note that the PC index reveals a fuzzier structure in the outcome of *user01*, as expected. Accordingly, the PE_2 shows that the entropy of *crisp* data set result is higher, revealing that less fuzzy structure was

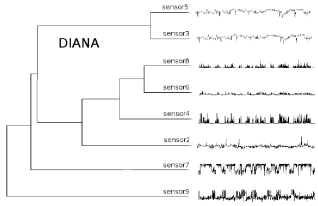


Fig. 3. Batch DIvisive ANAlysis clustering structure for the *user01* in PDMC data set, using the same correlation-based measure.

	Sensors							
	s2	s3	s4	s5	s6	s7	s8	s9
Final Clusters, u_{ic}								
node 14	1			1				
node 15						2^{-1}		2^{-1}
node 27			4^{-1}			2^{-1}		
node 31	2^{-1}							
node 32			4^{-1}		2^{-1}		2^{-1}	
node 29	4^{-1}							2^{-1}
node 33			2^{-1}		2^{-1}		2^{-1}	
node 34	4^{-1}							

Table 2. U Matrix for the final clusters obtained for the *user01* PDMC data set (zero-valued cells removed).

found. Evidence appears that the *user01* data set has some inherit fuzziness, which was absent in the *crisp* data set. The proposed method relies on a confidence test based on the Hoeffding bounds. This way, sensitivity analysis must be performed to assess the level of dependence of the method to this parameter. Figure 4 presents the analysis on the *user01* data set. We can observe that, for usual values of δ, the system reveals low sensitivity, being the best results observed for δ parameter values between 0.02 and 0.06. From this point, we chose to fix $\delta = 0.05$.

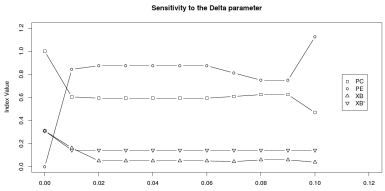


Fig. 4. ODAC quality sensitivity to the δ parameter, for the *user01* data, for PC, PE_2 and the *Xie-Beni* index for the *semi-fuzzy* (XB) and *crisp* (XB') partitioning.

	user01	*user06*	*user25*
Value, (*nc*)			
PC	0.594 (8)	0.672 (9)	0.938 (5)
PE_2	0.875 (8)	0.688 (9)	0.125 (5)
XB	0.051 (8)	0.123 (9)	0.134 (5)
$XB(strict)$	0.059 (4)	0.199 (4)	0.059 (4)

Table 3. Fuzzy validity indices for PDMC data sets. Last line presents the Xie-Beni index for the resulting structure using strict assignment ($\delta = 0.05$).

However, in order to assess the sensitivity of the method to different levels of fuzziness, we have applied the algorithm to two other users' data from the PDMC data set (80182 and 141251 observations). Figures 5 and 6 present the resulting structures for the three users. The resulting crisp clusters are the same in the three sets, although the hierarchy may be different. When using semi-fuzzy assignment, *user25* revealed much less fuzziness in the final structure than the remaining two sets, with almost the same clusters as the strict assignment method. Table 3 presents the values for the two partition indices

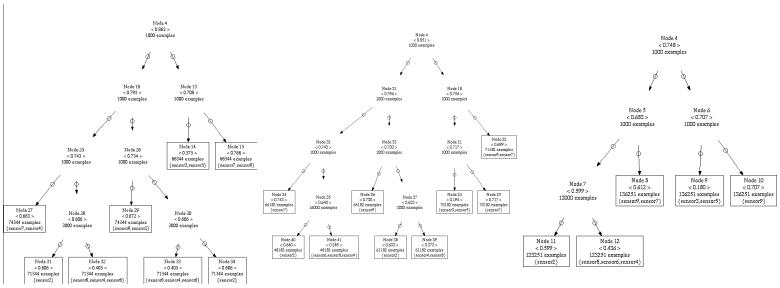


Fig. 5. ODAC semi-fuzzy structure for the PDMC users (*user01, user06* and *user25*).

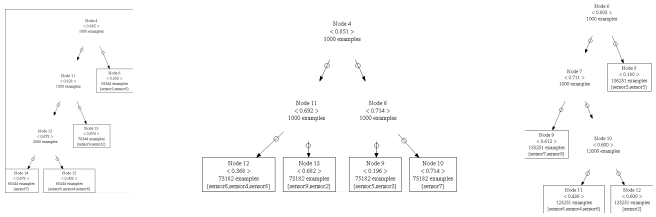


Fig. 6. ODAC strict structure for the PDMC users (*user01, user06* and *user25*).

and the *Xie-Beni* index for the three users, with the XB index also computed for the *strict* assignment method. While for the *user01* and *user06* data sets the semi-fuzzy approach resulted in better values of XB, the *user25* appears to contradict our approach. The strict assignment resulted in a much better value for the index, indicating that a fuzzy clustering would probably not be a good approach. Accordingly, when looking to the resulting hierarchy we can state that the semi-fuzzy assignment resulted in an almost strict partition of the variables, supporting the notion that, although semi-fuzzy, this approach will nonetheless find crisp partitions when data is inherently crisp.

6 Concluding Remarks

In this paper we have presented a semi-fuzzy variation to the assignment criterion of ODAC, a clustering system for streaming time series. ODAC uses a top-down strategy to construct a binary tree hierarchy of clusters with the goal of finding highly correlated sets of variables. The main underlying concept in ODAC is the clusters' diameter. The split decision used in the algorithm focus on the crisp boundary between two groups, generating uncertainty in the assignment since it has to decide based on only a small subset of the entire data. In this work we propose a semi-fuzzy approach to the assignment of variables to newly created

clusters, for a better trade-off between validity and performance. Experimental results show that this new assignment criterion will find better hierarchies when data is inherently fuzzy, without much loss in the quality of structures when the data is inherently crisp. Current and future work is concentrated on several areas, such as: the study of more complex membership functions; the inclusion of the membership function in the decision parameters of the entire system, such as the distance between variables and diameters computation; a post-prune criterion to reduce the size of the structure focusing on repeated instances of the same cluster; and a complete *fuzzy* assignment criterion for a complete *fuzzy* system.

Acknowledgment

Pedro P. Rodrigues is supported by a PhD grant awarded by FCT (SFRH/BD/29219/2006). The authors also wish to thank projects ALES II (POSC/EIA/55340/2004) and RETINAE (PRIME/IDEIA/70/00078).

References

1. J. C. Bezdek, R. Ehrlich, and W. Full. Fcm: The fuzzy c-means clustering algorithm. *Computers and Geoscience*, 10(2):191–203, 1984.
2. J. Gama and P. P. Rodrigues. Stream-based electricity load forecast. In *Knowledge Discovery in Databases: PKDD 2007*, LNAI 4702, pages 446–453. Warsaw, Poland, 2007. Springer Verlag.
3. P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80, Boston, MA, 2000. ACM Press.
4. M. Halkidi, Y. Batistakis, and M. Varzirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145, 2001.
5. W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.
6. L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley and Sons, Inc., New York, 1990.
7. R. N. Mantegna. Hierarchical structure in financial markets. *The European Physical Journal B*, 11(1):193–197, 1999.
8. Susana Nascimento. *Fuzzy Clustering Via Proportional Membership Model*, volume 119 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2005.
9. Karl Pearson. Regression, heredity and panmixia. *Philosophical Transactions of the Royal Society*, 187:253–318, 1896.
10. P. P. Rodrigues, J. Gama, and J. P. Pedroso. ODAC: Hierarchical clustering of time series data streams. In Srivastava, editors, *Proceedings of the Sixth SIAM International Conference on Data Mining (SDM 2006)*, pages 499–503,2006. SIAM.
11. D. M. Sherrill, M. L. Moy, J. J. Reilly, and P. Bonato. Using hierarchical clustering methods to classify motor activities of copd patients from wearable sensor data. *Journal of Neuroengineering and Rehabilitation*, 2(16), 2005.
12. X. L. Xie and G. Beni. A validity measure for fuzzy clustering. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 13:841–847, 1991.
13. L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

An Architecture for Context-Aware Adaptive Data Stream Mining

Pari Delir Haghighi, Mohamed Medhat Gaber, Shonali Krishnaswamy, Arkady Zaslavsky, and Seng Loke

Center for Distributed Systems and Software Engineering
Monash University, Australia
{pari.delirhaghighi, shonali.krishnaswamy, Arkady Zaslavsky}@infotech.monash.edu.au
CSIRO ICT Center, Australia
Mohamed.gaber@csiro.au
Department of Computer Science and Computer Engineering
La Trobe University, Australia
s.loke@latrobe.edu.au

Abstract. In resource-constrained devices, adaptation of data stream processing to variations of data rates, availability of resources and environment changes is crucial for consistency and continuity of running applications. Context-aware adaptation, as a new dimension of research in data stream mining, enhances and optimizes distributed data stream processing tasks. Context-awareness is one of the key aspects of ubiquitous computing as applications' successful operations rely on detecting changes and adjusting accordingly. This paper presents a general architecture for context-aware adaptive mining of data streams that aims to dynamically and autonomously adjust data stream mining parameters according to changes in context and resource availability in distributed and heterogeneous computing environments.

1. Introduction

Processing of data streams due to their unpredictable and continuous nature [1, 2] is a challenging area of study. In the literature, various techniques and approaches have been presented to address the issues associated with data stream processing both in data mining and querying. However, recently the emergence and growth of mobile computing and networking and importance of using mobile devices for data stream mining in certain application domains (e.g. health or bushfire monitoring applications) have introduced new research challenges that need to be addressed. Data stream applications running on resource-constrained devices need not only to consider limitations of computational resources such as memory, battery level and CPU speed but also to take into account the issues of variable data rates, mobility, disconnections and environmental changes.

Nearly all pervasive systems utilize context to perform their tasks and this makes context-awareness an essential requirement of these systems [3, 4]. To perform data stream mining in heterogeneous and distributed computing environments, applications need to monitor context changes and react or adapt to them in order to maintain consistent and continuous operations.

Studying the current state-of-art in data stream mining [5-7] indicates that there are methods and algorithms introduced for efficiently mining high speed data streams in mobile devices such as Personal Digital Assistants (PDAs) but they have limited dynamic ability to adjust to a multitude of changing contextual parameters and have not been adequately equipped to cope with the distributed and heterogeneous nature of applications or the mobility of the users/devices that these techniques aim to support. One of the innovative adaptive works in data stream mining on resource-constrained devices is Algorithm Output Granularity (AOG) [8, 9] that provides adaptability with respect to the available memory on a device. Examples of light-weight data stream mining algorithms that have been developed using the AOG include LWC, LWClass and LWF [10].

In this paper we introduce a novel architecture for context-aware adaptive data stream mining that aims to provide real-time and dynamic strategies for adaptation and cost-efficiency by factoring in current context, availability of resources and distribution of resources and processing. This approach will significantly contribute to a range of application areas such as the mobile workforce, Intelligent Transportation Systems and sensor network applications. The summary of our main contributions are as follows:

- Incorporating context-awareness into data stream processing as a meta-level concept (i.e. situations) based on the Naïve Context Spaces (NCS) model;
- Enabling real-time and cost-efficient adaptation by matching context changes to a set of pre-defined application-specific situations and responding to changes accordingly;
- Introducing adaptation strategies with data stream processing parameters that are dynamically set/adjusted at run-time based on contextual changes, and shifting from purely reactive to proactive behavior;
- Ensuring the continuity and consistency of running operations on resource-constrained devices;

To explain different parts of our architecture throughout the course of the paper, we will use an example of a health monitoring application for heart patients. The details of the examples are only provided for the purpose of illustrating the points and may lack the necessary medical accuracy and correctness. Applications for healthcare biosensor networks are recently gaining popularity among people as they provide a convenient and safe way to monitor patients remotely and generate warnings and emergency calls. One of the main biosensors used for monitoring heart patients is ECG (Electrocardiogram) sensors that send heart beat rates as a continuous data stream to a PDA [11] or to a base station using an ISM band [12]. Data stream querying or mining needs to be performed on the ECG sensor streams locally on a mobile device or on a central workstation for monitoring and analyzing heart beats.

This paper is structured as follows: Section 2 provides a general view of our proposed architecture for context-aware adaptive data stream mining. Section 3 discusses context and situation modeling and how situations are inferred. Section 4 focuses on adaptation strategy and correlation functions. Finally section 5 concludes the paper and discusses future work.

2. An Architecture for Context-Aware Adaptive Data Stream Mining

In this section, we introduce an architecture for context-aware adaptive data stream mining. The architecture consists of two parts as shown in Figure 1. The first part is a situation manager that provides context-awareness and includes components for context modeling and inference. The second part, strategy manager, is responsible for adjusting adaptation strategy parameters based on correlation functions and invoking strategies.

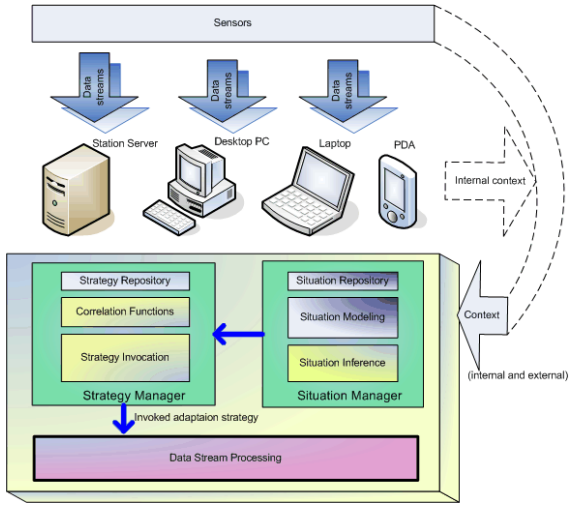


Figure 1. A general architecture for context-aware adaptive data stream mining

3. Situation Manager

The situation manager consists of three components: Situation repository, Situation Modeling and Situation Inference. These components work together to reason about the occurring situation based on the current context attribute values. Contextual information used for inferring situations may include sensed context collected from sensors, static context or internal context such as battery level of mobile devices.

3.1 Situation Modeling

We have based our context representation and modeling on the Naïve Context Spaces (NCS) Model [13, 14] but made changes to it to comply with our purpose. The NCS model and its extension in [15] are used as a powerful tool for reasoning about context and addressing uncertainties of sensed information. The core of the NCS model is the concept of situations. The NCS model represents contextual information as geometrical objects in multidimensional space called *situations* [13]. A *situation space* is a tuple of regions of attribute values related to a situation. Each *region* is a set of accepted values for an attribute based on a pre-defined predicate and each *context state* a collection of values of context attributes at the given time.

The NCS model extends the definition of context by describing it as "the set of facts, assumptions and predictions along with methods/algorithms of interpreting/ discovering/ processing that information" [15].

Using the NCS model, a situation occurs if every sensed context attribute value meets the predicate of the region set for the same type of attribute. We consider these situations as *known* situations. However, if the current context state does not match any of the pre-defined situations, it indicates the occurrence of an *unknown* situation. Any unknown situation can be *similar*/*dissimilar* to the situations already defined.

If we have an occurring situation S_i with a region of A_j, then for the sensed context attribute of a_i^t, we will have $a_i^t \in A_j$, and $A_{\min value} \le a_i^t \le A_{\max value}$ for continuous values. For any unknown situation there will be at least one context attribute value a_i^t that does not satisfy its associated region's predicate.

Considering our example for monitoring heart patients, from a list of related context attributes, we consider the following context attributes: temperature a_1 (0-50), age a_2 (20-120), location a_3 (HOME, NOT HOME), time a_4 (24 hours), heart rate a_5 (60-180), and Battery_level a_6 (0-100%). Weights are values from 0 to 1 that represent the importance of each context attribute in a situation and can have the total value of 1 per situation. Table 1 shows examples of pre-defined situations based on the aforementioned context attributes.

Table 1. Examples of situations

Situation	Context attributes	Regions and their predicates	Weight
S_1 sleep	a_1		
	a_2	<34	0.1
	a_3	<85	0.02
	a_4	HOME	0.03
	a_5	>10pm AND <7am	0.35
	a_6	<100	0.4
		>80%	0.1
S_2 Heat_stroke_ threat	a_1		
	a_2	>34	0.4
	a_3	>75	0.15
	a_4	NOT HOME	0.01
	a_5	>11am AND <8pm	0.01
	a_6	>100	0.4
		>40%	0.03
S_3 critical	a_1		
	a_2	>34	0.2
	a_3	>70	0.03
	a_4	NOT HOME	0.05
	a_5	>4pm AND <6pm (rush hour)	0.02
	a_6	>100	0.6
		<40%	0.1

3.2 Situation Repository

The situation repository contains a set of pre-defined situations that specify the most important regions of context attributes for the application however any major changes in context attribute values will be modeled as a situation (known or unknown) and used for adaptation of strategy parameters.

We have used XML schema (as shown in Figure 2) for defining our context model and XML documents for defining situations of different application domains.

We use JAXB to convert an XML document to java classes based on our XML schema. XML is a powerful and easy tool for the sharing of data across different applications. Applications can express their domain-related situations as an XML document in a simple way without requiring the knowledge of the underlying situation model.

```
<xsd:complexType name="Situations">
   <xsd:sequence>
      <xsd:element name="situation" minOccurs="1" maxOccurs="unbounded">
        <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="situationName" type="xsd:string"/>
          <xsd:element name="significance" type="xsd:string"/>
          <xsd:element name="regions" type="Regions"/>

      ...

<xsd:complexType name="Regions">
  <xsd:sequence>
    <xsd:element name="region" minOccurs="1" maxOccurs="unbounded">
     <xsd:complexType>
     <xsd:sequence>
          <xsd:element name="attributeName" type="xsd:string"/>
          <xsd:element name="weight" type="xsd:string"/>
          <xsd:element name="predicates" type="Predicates"/>

      ...

<xsd:complexType name="Predicates">
  <xsd:sequence>
    <xsd:element name="predicate" minOccurs="1" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
              <xsd:element name="condition" type="xsd:string"/>
                  <xsd:element name="valueType" type="xsd:string"/>
            <xsd:element name="values" type="Values"/>

      ...

<xsd:complexType name="Values">
  <xsd:sequence>
    <xsd:element name="attributevalue" minOccurs="1" maxOccurs="unbounded">
      <xsd:complexType>
```

Figure 2. XML schema of the situation model

3.3 Situation Inference

Situation inference is here referred to as discovering which pre-defined situation matches the current context state (collection of context attributes). Situation inference finds a perfect match for known situations and the closest match for unknown situations.

1. Let $R_{situations} = \{S_1, S_2, ..., S_n\}$ be the set of pre-defined situations in the *Situation Repository*.
2. Let context state $C = \{a_1^t, a_2^t, ..., a_m^t\}$ be the set of context attributes' values collected at time *t*.

3. We use the function $perfectMatch(C^t, R_{situations})$ to find a pre-defined situation that matches the current context state C_t. If there is not a perfect match, we find the most similar situation $S_{similar}$ using the State-Space difference Δ_{state}^{space} measures [15]. The function f calculates the distance between the context attribute and the region of accepted values, and $\hat{a}_i$ denotes the accepted region's absolute size.

$$\Delta_{state}^{space} = \sum_{i=1}^{n} w_i . f(a_i^s, a_i^r, \hat{a}_i) \cdot \quad (1)$$

4. After computing the state-space difference for all the pre-defined situations, the situation with the least difference value is selected as the most similar situation. The difference between a sensed context attribute and its closest value in the accepted region $(a_i^s - a_i^r)$ is later used for adjusting strategy parameters.

3.3.1 K-Nearest neighbor algorithm

To provide more accurate inference results we also perform the *k-nearest neighbor algorithm* to classify unknown situations based on the closest pre-defined situations. We first normalize instances and then measure Euclidean distance between the current context state and pre-defined situations to find the closet situation. We have tested the algorithm with k=1, k=3 and k=5, and the results are very similar. However, we intend to experiment this further with larger number of training examples and different application domains to determine the value of k.

4. Strategy Manager

To achieve real-time and dynamic adaptability, we use a set of parameterized adaptation strategies with their correlation functions. At run-time, the changes in context attribute values are used for adjusting parameter values in the corresponding strategy using its matching function. The strategy manager includes strategy repository, correlation functions and strategy invocation.

4.1 Strategy Repository

Every occurring situation may invoke a corresponding strategy to adjust the data stream mining parameters. We initially define a set of strategies with their parameters for each

pre-defined situation. Table 2 illustrates some examples of these strategies for our scenario.

Table 2. Examples of strategies

Strategies		parameters	Value	Situations
Str_1	Increase	window size (sec)	min= +5 max= +10 (non-linear)	S_1 Sleep
Str_2	Switch /change	process	lighter weight algorithm	
Str_3	Decrease	Window size (sec)	-6	S_2 heat_stroke
Str_4	Decrease	Window size (sec)	-12	S_3 Critical
Str_5	send warning/emergency call	-	-	

4.2 Strategy Invocation

Depending on which situation is currently occurring, a corresponding strategy is selected.

1. Let $R_{strategies} = \{Str_1, Str_2, ..., Str_n\}$ be the set of pre-defined strategies in the strategy repository.
2. The strategy invocation $SI(S_{occuring}, R_{repository}) = Str_i$ returns the adaptation strategy for the occurring situation.

Adaptation parameters need to be adjusted properly before they could be applied to data stream mining tasks. We use correlation functions to adjust the value of parameters based on context attribute values. For a data stream process P_i running on a data stream DS_i, the parameters that have been considered in resource-aware approaches [8, 16] include *Input rate, output rate, time frame, procesesing rate of data per unit, total time, available memory*, and *sampling method*. From the above parameters, those that can be adjusted by an adaptation strategy are: *Input rate*, *output rate*, *time frame*, and *sampling method*.

In addition to the above parameters, we consider the parameter of location L_i^{ds} for data streams, L_i^{device} for the mobile device D_i, and $L_i^{algorithm}$ for light weight algorithms that can be transferred from one location/device to another. The parameter of location addresses the aspect of mobility in devices and users as well as distribution of data streams and

algorithms on heterogeneous computing devices. Figure 3 illustrates adaptation of data stream processing.

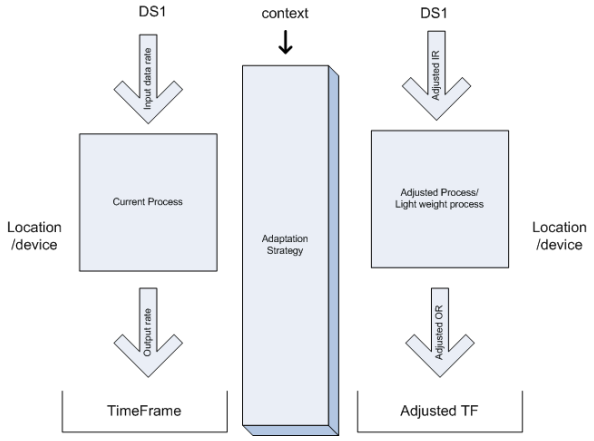


Figure 3. Adaptation of data stream processing

4.3 Correlation Functions

In order to make adjustments to strategy parameters, we use correlation functions that are pre-defined functions and vary from linear to non-linear. Correlation functions calculate strategy parameter values according to changes in context and the occurring situation. We envisage this concept by providing an example of context-aware adaptation from our scenario. In the heat-stroke-threat situation, the matching strategy reduces the window size from 30 sec (set value) to 24 sec (-6) as shown in Table 2. For a similar situation that meets all the accepted regions of the heat-stroke situation but its temperature value is 33 (refer to Table 1), we use a linear function (i.e. window size = $ax+b$ where $x=\Delta_{a_i^s}^{a_i^r}$ and a and b are set values) to adjust the parameter of window size. We discuss this further in the following steps:

1. Strategy invocation, $SI(S_{occuring}, R_{repository}) = Str_i$ returns the corresponding strategy Str_i for the current situation inferred by the situation manager.
2. Let $CF = \{f_1, f_2, ..., f_n\}$ be a set of pre-defined correlation functions.

3. Let $f_i(a_i, p_i) \rightarrow ap_i$ be a correlation function for Str_i that returns ap_i the adjusted value of the parameter p_i based on the relationship of the context attribute a_i and the parameter p_i where $p_i \in Str_i$ and $a_i \in S_{occuring}$.
4. The adjusted value ap_i of parameter p_i is then used in the strategy Str_i.

As an example of cost-efficient adaptation, we include a strategy for the sleep situation that uses a pre-defined non-linear function to increase the window size (i.e. reducing the reading rates of heart beat) based on the attribute of *time* during the night in order to save device resources. During the day (unknown situation), the window size is automatically adjusted according to the relationship of the time and window size in the sleep situation using a similar correlation function as shown in Figure 4. We consider sleep situation for patients who mainly suffer from heart irregularities under stress and tension that are more likely during the day.

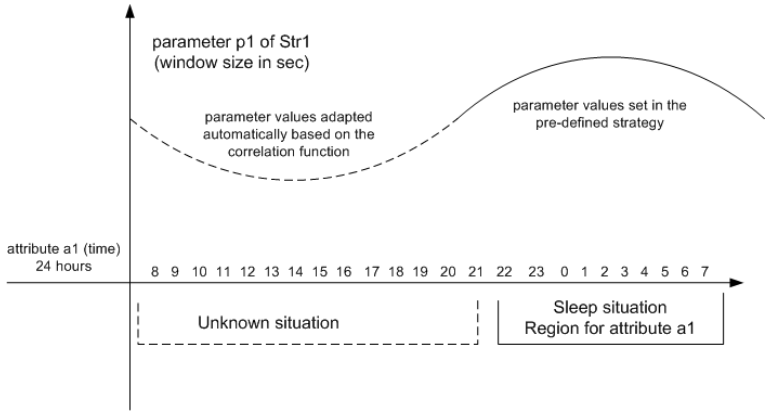


Figure 4. Using a quadratic function for adjusting parameters for sleep situation

5. Conclusion and Future Work

In distributed and heterogeneous environments, data stream applications have to cope with high data stream rates and limited computing resources such as memory or battery. Moreover, these applications need to address mobility, disconnections and environmental changes that are the norm in ubiquitous computing environments. Integrating data stream

processing with context-awareness enables applications to adjust to changes and continue their operations as expected.

In this paper we introduced a general architecture for context-aware adaptive data stream mining in heterogeneous computing environments. Our project is currently at the initial stages and our intention is to build a middleware based on the proposed architecture that will provide adaptation tasks and services on different data stream applications. While our current focus is on data stream mining, we see the potential for generalizing the approach for data stream processing including querying. Furthermore, there are still open issues in terms of distribution of data streams and mining algorithms that we intend to address in future work.

References

1. Babu, S., Widom, J.: Continuous Queries over Data Streams. ACM SIGMOD Record 30(3): pp. 109--120 (2001)
2. Golab, L., Ozsu, M.T.: Issues in Data Stream Management. SIGMOD Record 2003. 32(2): pp. 5--14 (2003)
3. Bunningen, A.H., Feng L. Apers, P.M.G.: Context for Ubiquitous Data Management. In: Proceedings of the 2005 International Workshop on Ubiquitous Data Management (UDM'05), pp. 17--28. IEEE Computer Society, Tokoyo (2005)
4. Davidyuk, O., Riekki, J., Rautio, V. Sun, J.: Context-Aware Middleware for Mobile Multimedia Applications. In: Proceedings of the 3rd International Conference on Mobile and Ubiquitous Multimedia. pp. 213--220. ACM Press, Maryland (2004)
5. Galan, M., Liu, H., Torkkola, K.: Intelligent Instance Selection of Data Streams for Smart Sensor Applications. SPIE Defense and Security Symposium, Intelligent Computing: Theory and Applications III., pp. 108-119. Orlando (2005)
6. Kargupta, H., Park, B, Pittie, S., Liu, L., Kushraj, D., pSarkar, K.: MobiMine: Monitoring the Stock Market from a PDA. SIGKDD Explorations, 3(2): pp. 37--46 (20020
7. Kargupta, H., Bhargava, R., Liu, K., Powers, M., Blair, P., Bushra, S., Dull, J., Sarkar, K., Klein, M., Vasa, M., Handy, D.: VEDAS: A Mobile and Distributed Data Stream Mining System for Real-Time Vehicle Monitoring. In: Proceedings of the SIAM International Data Mining Conference (SDM 2004), Orlando (2004)
8. Gaber, M.M., Krishnaswamy, Sh., and Zaslavsky, A., Resource-Aware Mining of Data Streams. Journal of Universal Computer Science. 11(8): pp. 1440--1453.
9. Gaber, M.M., Krishnaswamy, S., and Zaslavsky, A., On-board Mining of Data Streams in Sensor Networks, Advanced Methods of Knowledge Discovery from Complex Data, S. Badhyopadhyay, Maulik, U., Holder, L., and Cook,, D. (eds.), pp. 307--337. Springer, (2005)
10. Gaber, M.M., Krishnaswamy, Sh., and Zaslavsky, A. Cost-Efficient Mining Techniques for Data Streams. In: Proceedings of the 2nd Australasian Workshop on Data Mining and Web Intelligence (DMWI2004). Australian Computer Society, pp. 109--114. New Zealand (2004)
11. Brettlecker, G., Schuldt, H., and Schatz, R. Hyperdatabases for Peer-to-Peer Data Stream Processing. in Proceedings of the IEEE International Conference on Web Services (ICWS'04). pp. 358. IEEE Computer Society, San Diego (2004)

12. Chen, C., Agrawal, H., Cochinwala, M. and Rosenbluth, D. Stream query processing for healthcare bio-sensor applications. In: Proceedings of the 20th International Conference on Data Engineering (ICDE'04). pp. 791--794. IEEE Computer Society, Boston (2004)
13. Padovitz, A., Loke, S.W. and Zaslavsky, A. Towards a Theory of Context Spaces. In: Proceedings of the 2nd IEEE Annual Conference on Pervasive Computing and Communications, Workshop on Context Modeling and Reasoning (CoMoRea). IEEE Computer Society. Orlando (2004)
14. Padovitz, A., Loke, S.W., Zaslavsky, A. and Burg, B. . Towards a General Approach for Reasoning about Context, Situations and Uncertainty in Ubiquitous Sensing: Putting Geometrical Intuitions to Work. In: Proceedings of 2nd International Symposium on Ubiquitous Computing Systems (UCS'04). Japan (2004)
15. Padovitz, A., Context Management and Reasoning about Situations in Pervasive Computing, Caulfield School of Information Technology. Monash University: Australia. (2006)
16. Agarwal, I., Krishnaswamy, Sh., and Gaber M. M. Resource-Aware Ubiquitous Data Stream Querying In: Proceedings of the International Conference on Information and Automation. Sri Lanka (2005)

Efficient Secure Query Processing in XML Data Stream

Dong-Chan An, Seog Park

Department of Computer Science & Engineering, Sogang University
C.P.O. Box 1142, Seoul Korea 100-611
{channy, spark}@sogang.ac.kr

Abstract. As various users and applications require the distribution and sharing of information in XML documents, the need for an efficient secure access of XML data in a ubiquitous data stream environment has become very important. In this paper, we propose an efficient secure XML query processing method to solve the two problems by using role-based prime number labeling and XML fragmentation. A medical records XML document has the characteristic of an infinite addition in width rather than in depth because of the increment of patients. But a role-based prime number labeling method can fully manage the size of documents that increases to infinity and can minimize the maintenance cost caused by dynamic changes. Experimental evaluation clearly demonstrates that our approach is efficient and secure.

1 Introduction

XML [1] is recognized as a standard for information representation and data exchange, and the need for distribution and sharing in XML data basis is steadily increasing, making the efficient and secure access to XML data a very important issue. Despite this, relatively little work has been done to enforce access controls particularly for XML data in the case of query access. Moreover, the current trend in access control within the traditional environment has been a system-centric method under finite, persistent data environment and an access control under static data environment. However, more recently, access control policies have become increasingly needed in the continuous data stream [15] environment, and consequently, it has been accepted that the pre-existing access control methods do not meet these needs.

This paper proposes an efficient secure XML query processing using role-based prime number labeling method with regard to characteristics of XML data stream under ubiquitous environment. The proposed method enables efficient and secure real-time processing of a query in a mobile terminal, applying the characteristics of XML data stream.

2 Related Work

The traditional XML access control enforcement mechanism [4-13] is a view-based enforcement mechanism. The semantics of access control to a user is a particular view of the documents determined by the relevant access control policies. It provides a useful algorithm for

computing the view using tree labeling. However, aside from its high cost and maintenance requirement, this algorithm is also not scalable for a large number of users.

To overcome the view-based problems, M. Murata et al. [16] proposed the filtering method to filter out queries that do not satisfy access control policies. B. Luo et al. [17] took extra steps to rewrite queries in combination with related access control policies before passing these revised queries to the underlying XML query system for processing. However, the shared Nondeterministic Finite Automata (NFA) of access control rules is made by a user (or a user's role). Thus, the shared NFA involves many unnecessary access control rules from the user's query point of view, which further result in time-consuming decisions during which the user's query should have already been accepted, denied, or rewritten.

2.1 Access Control Policy

General access control brings additional processing time. However, it is overlooked that such time spent on unnecessary access control rules could be reduced when access control rules in XML document basis should be expressed in XPath [2] and users request queries in XPath. In other words, an XML document can, depending on a user's query, be classified into these three parts: ancestor node, descendant node, and sibling node (following-sibling node and preceding-sibling node). All access control rules needed, based on a user query, are just only access controls described in the ancestor node or descendant node. Access controls described in sibling nodes are unnecessary access control rules from the point of user query.

An authorization defined by a security manager is called explicit and an authorization defined by the system, on the basis of an explicit authorization, is called implicit in the hierarchical data model [3]. An implicit authorization is used with an appropriate 'propagation policy' to benefit the advantage of storage. With an assumption that the optimized propagation policy varies under each of the different environments, 'most-specific-precedence' is generally used. On the other hand, 'denial-takes-precedence' is used to resolve a 'conflict' problem that could be derived from propagation policy by such implicit authorization. Since positive authorization and negative authorization are also used together, 'open policy', which authorizes a node that does not have explicit authorization, and 'closed policy', which rejects access, are used. The policies 'denial-takes-precedence' and 'closed policy' are generally used to ensure tighter data security [16].

2.2 XFrag

The high practicality of mobile terminals and computing power is necessary for the feasibility of ubiquitous computing. The efficiency of memory, energy, and processing time is also especially needed. XML data has a hierarchical structure and the capacity might be very huge. A method that can take XML data into appropriate fragmentation so as to process it in pieces is consequently needed for the small memory of a mobile terminal to manage massive XML data [18, 19]. When XML streams data, which is generated under a sensor network, the data is structurally fragmented and transmitted and processed in XML piece stream, the efficiency of memory and the processing time of mobile terminals can be reconsidered. Moreover, when certain data is updated in an XML data stream, not the whole XML data but only the changed fragment needs to be transmitted, taking advantage of a reduced transmission cost.

The recent Hole-Filler Model [20, 21] has been proposed as a method that fragments XML data structurally. XFrag [21] and XFPro [22] proposed an XML fragmentation processing method adopting the Hole-Filler Model. Nonetheless, this method has problems of late processing time and waste of memory space due to additional information for the Hole-Filler Model. The XFPro method has improved processing time by improving the pipeline, but is not

a solution for the problems that the Hole-Filler Model has. A medical records XML document [25] is shown in Fig. 1, and a fragmented XML document by the Hole-Filler Model [21] is shown in Fig. 2.

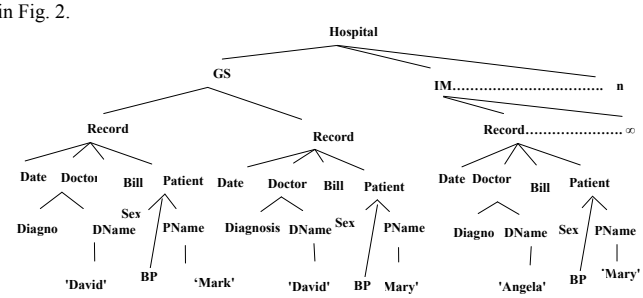


Fig. 1. Medical Records XML Document

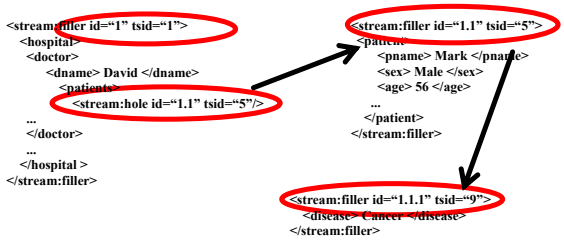


Fig. 2. Fragmented XML Document by Hole-Filler Model

2.3 Prime Number Labeling

For the query processing of a dynamic XML document, a labeling technique, which is easily applied to insert and delete elements, is needed. Some existing labeling techniques lack the document updating capability and search whole XML document trees again to re-calculate the overall label of the node, thus bringing costs higher [23].

A new labeling technique has shown up as a consequence of the appearance of the dynamic XML document. This technique is typical of the prime number labeling scheme [24] applied to information which rarely affects other labels. This technique assigns a label for each node, a prime number, to represent the ancestor-descendant relation and is designed not to affect the label of other nodes when updating the document. However, since it searches a considerable range of the XML tree again and re-records updated order information during the updating process, it presents a higher updating cost.

2.4 Problems

In sections 2.2 and 2.3, this paper pointed out the low practicability of existing access control under XML data streams. This paper proposes a fine-grained access control using role-based

prime number labeling method with regard to characteristics of XML data stream under a ubiquitous environment. The proposed method enables the efficient and secure real-time processing of a query in a mobile terminal, applying the characteristics of an XML data stream.

3 Proposed Method

The proposed environment of the role-based prime number labeling (RPNL) method is shown in Fig. 3. It depicts medical records that need accurate real-time query answers by checking the authority and the role of valid users via access control policy when a query is requested.

3.1 Role-Based Prime Number Labeling

The role-based prime number (RPN) labeling method is explained under a certain environment as in Fig. 3. First of all, considering the characteristics of the proposed environment, the fragmentation of the XML document in Fig. 1 is shown in Fig. 4. Problems such as low processing time and waste of memory space needed due to additional information for the Hole-Filler Model in existing XFrag [21] is minimized as shown in Fig. 4. This means that information such as tag structure is no longer needed because the order of XML documents no longer need to be considered.

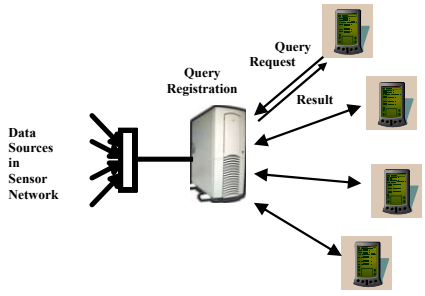


Fig. 3. Query Processing of Mobile Terminal over XML Data Stream Environment

```
<stream:filler id="1">
  <hospital>
   <deptname>GS</deptname>
      <record>
         <stream:hole id="1.1"/>
         <stream:hole id="1.2"/>
      …
      </record>
  </deptnamel >
  <deptname>IM</deptname>
      <record>
         <stream:hole id="2.1"/>
         <stream:hole id="2.2"/>
      …
      </record>
  </deptnamel >
  …
  </hospital>
</stream:filler>
```

```
<stream:filler id="1.1" >
  <date> 05-09-2007 </date>
  <doctor>
    <diagnosis> cancer </diagnosis>
    <dname> David </dname>
  </doctor>
  <bill>$40,000</bill>
  <patient>
    <pname> Mark </pname>
    <sex> Male </sex>
    <BT> 38 </BT>
    <BP> 150 </BP>
    ...
  </patient>
</stream:filler>
```

Fig. 4. Partial Fragmentation in XML Data Stream

After a fragmenting procedure of XML data stream, proper role-based prime numbers are assigned to nodes of the medical records XML data stream of Fig. 1 as shown in Table 1. Since roles are limited in any organization, it is possible to represent roles with a prime number and a prime number is expansible.

Table 1. Role-Based Prime Number Table

Roles	Role-Based Prime Number
Patient	2
Doctor	3
Researcher	5
Insurer	7

Referring to Table 1, a role-based prime number labeling algorithm is performed in Fig. 5.

(1) Level 1 : integer number assign to department (1~n)

- GS : 1, IM : 2

(2) Level 2 : 2nd level assign to sub node of department record

- GS record's sub node : (1.1), (1.2)...
- IM record's sub node : (2.1), (2.2)...

(3) Level 3 : 3rd level assign to role-based prime number in record's sub nodes

[department].[record].[sub node]

- Date(*.*.210) : product of 2,3,5,7
- Doctor(*.*.30) : product of 2,3,5
- Diagnosis(*.*.30) : product of 2,3,5
- DName(*.*.6) : product of 2,3
- attribute value of terminal node : inherit role-based prime number of super node
- Patient(*.*.210) : product of 2,3,5,7
- Sex(*.*.210) : product of 2,3,5,7
- PName(*.*.42) : product of 2,3,7
- BP(*.*.30) : product of 2,3,5

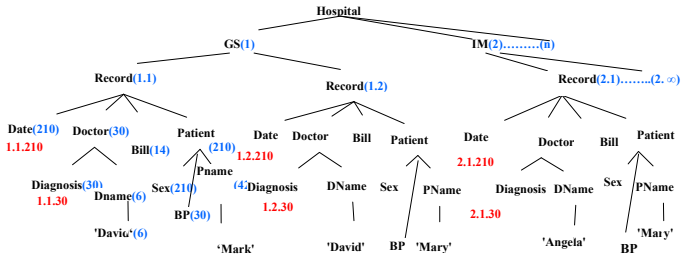
(4) Order : out of consideration

Fig. 5. Role-Based Prime Number Labeling Algorithm

XML document acquired through Fig. 5 is shown in Fig. 6-(a).

Roles	Role-Based Prime Number Label
Date	1.1.210
Doctor	1.1.30
Bill	1.1.14
Patient	1.1.210
PName	1.1.42
BP(Blood Pressure)	1.1.30

(a)



(b)

Fig. 6. (a) //GS/Record(1.1)'s RPN, (b) Medical Records XML Document's RPN

3.2 Query Processing by RPN

The proposed security system's architecture is shown in Fig. 7. The query processing procedure in Fig. 7 can be considered in two steps. The role check is done in Step 1 using the 'Role-Based Prime Number Table' and final access authority is checked at Step 2 using the 'Role Privacy Table'. Once a query from a mobile terminal is registered, access authority is checked at Step 1 by checking the prime number of the query terminal node. That is, access to Step 1 is permitted when the remainder of the RPN divided by the role of user becomes zero. Accessibility is finally checked at Step 2 referring to the 'Role Privacy Table' of Table 2. Moreover, as indicated in Section 2.1, query access is rejected by 'denial-takes-precedence'[16]. Details will be verified in the following example

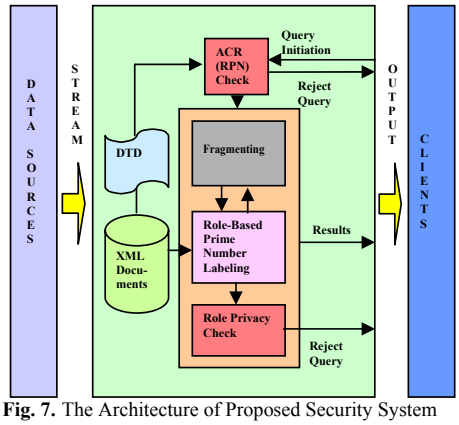


Fig. 7. The Architecture of Proposed Security System

• Example1 (predicate + positive access control + positive access control)

```
(1) //record/patient[pname=Mark]
(2) "David" with role of doctor requests a query
```

- step1, terminal node pname=Mark's label is verified : 1.1.42
- David's role = doctor : 3

- 42%3=0, access is permitted
- step2, Only 1.1.* and 1.2.* is permitted for David by 'Role Privacy Table'
- finally, 1.1.42*(//record/patient[pname=Mark])* access permitted
- response to the query

• Example 2 (positive access control + positive access control)
```
(1) //record/bill
(2) "AIG" with role of insurer requests a query
```
- step1, terminal node bill's label is verified : *.*.14
- insurer's role : 3
- 14%7=0, access is permitted
- step2, Only 1.1.* and 1.2.* is permitted for AIG by 'Role Privacy Table'
- finally, access of 1.1.14 and 2.1.14 is permitted
- response to the query

• Example 3 (predicate + positive access control + negative access control)
```
(1) //record/patient[pname=Mark]
(2) "Angela" with role of doctor requests a query
```
- step1, terminal node pname=Mark's is verified : 1.1.42
- Angela's role = doctor : 3
- 42%3=0, access is permitted
- step2, Only 2.1.* is permitted for Angela by 'Role Privacy Table'
- [pname=Mark] is 1.1.42, access rejected
- access to step1 permitted, access to step2 rejected.
- finally query access rejected

• Example 4 (negative access control)
```
(1) //record/patient/pname
(2) one with role of researcher requests a query
```
- step1, terminal node pname's label is verified : *.*.42
- researcher's role : 5
- 42%5≠0, access is rejected
- finally, query rejected

Table 2. Role Privacy Table

Dept.	Records	Roles			
		Patient	Doctor	Insurer	Researcher
1	1.1	Mark	David	ING	x
	1.2	Mark	David	AIG	x
	...	...			x
	1.∞		...		x
2	2.1	Mary	Angela	AIG	x
	...	...			x
	2.∞		...		x
...	...			...	x
n	n.∞				x

As shown in Example 4, the proposed method has a strong point in that it processes the rejection to a query quickly.

4 Experimental Evaluation

We used one PC with an Intel Pentium IV 3.0GHz CPU, with a main memory of 1GB using the MS Windows XP Professional OS. The Programming language used was JAVA (JDK1.5.0). Data for the experiment was used in the form of random medical records XML documents. Performance was compared mainly in two aspects.

4.1 Experiment I: Accurate Detection of Rejection Query

A rejection query is a user query that has to be rejected at all cases. Thirty intended rejection queries that suited each type of query were made up, and access control policy and actual number of detection of rejection queries was compared to this. The result is shown in Fig. 8. The experiment was conducted in three cases: "/" axis, "//" axis, and a case that has a predicate in a terminal node. The result demonstrates that the intended 30 rejection queries were detected 100%.

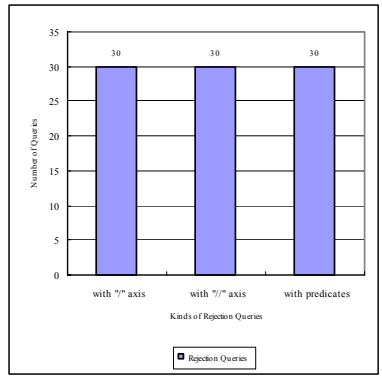
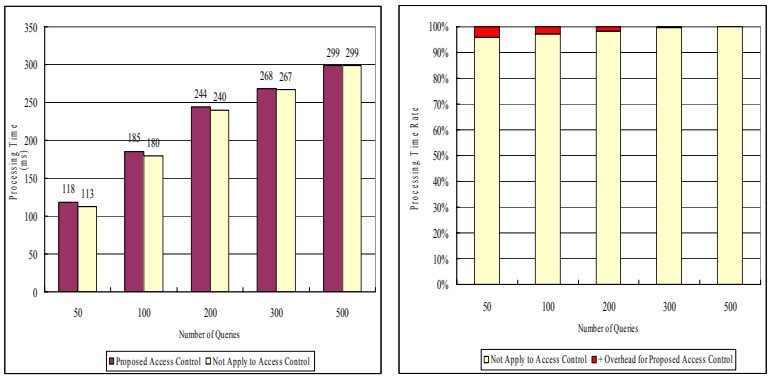


Fig. 8. The Result of Detection for Rejection Query

4.2 Experiment II: Processing Time of Query

Average query processing time was compared in two cases: one applied the access control method proposed in this paper and the other did not.

Average processing time was measured according to random samples of XPath query numbers (50, 100, 200, 300, and 500). Processing time is represented by an average time so that error of measurement can be minimized. Role-based prime number labeling time was not included in the response time in the proposed method because it is reconstructed when an update such as insertion or deletion of medical records XML documents is made. Referring to role prime number labeling which is generated before query process, and query processing time including the pure procedure of authority checking was measured. Nonetheless, the fact that referring to access control information does not affect the system performance was discovered. Fig. 9 shows the result.



(a) (b)

Fig. 9. (a) The Processing Time of The Security Check on Queries, (b) The Overhead of The Security Check on Queries

5 Conclusions

Considering the efficiency of memory, energy, and processing time in the performance of mobile terminals frequently used under a ubiquitous environment, the processing of massive XML data stream is impossible. Moreover, it is more challenge to implement access control on mobile terminals under a ubiquitous environment. In addition to these, security becomes a very important issue due to the increasing number of users, and the increase of the amount of data being used. We proposed an efficient secure XML query processing method to solve those two problems by using role-based prime number labeling. Medical records XML documents, and the proposed environment, in this paper, have the characteristic of infinite additions in width rather than in depth because of the increment of patients. In this manner, the role-based prime number labeling method was able to fully manage the size of documents that increase to infinity and can minimize the maintenance cost caused by dynamic changes. While the tree structure of XML documents, in addition, should be searched more than twice for the application of security during query processing in previous works, one search is possible for real-time processing resulting in minimization of transmission cost through fragmentation of XML documents by the proposed method. In terms of security, system load is minimized and a perfect access control is implemented by application of two-step security. First of all, a query by a user who does not have a role that does not meet access control rules is promptly rejected applying the characteristics of the prime number and a stricter access control can be applied by the application of two-step security. However, it is a little burdensome to keep the role privacy table. Nonetheless, the experiment still showed the superiority of our proposed method.

Acknowledgements

This work was supported by the Korea Science and Engineering Foundation(KOSEF) grant funded by the Korea government(MOST) (No. R01-2006-000-10609-0).

References

1. T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau. Extensible Markup Language (XML) 1.0, World Wide Web Consortium (W3C), 2004.
2. A. Berglund, S. Boag, D. Chamberlin, M. F. Fernández, M. Kay, J. Robie, and J. Siméon. XPath 2.0, World Wide Web Consortium (W3C), 2005.
3. F. Rabitti, E. Bertino, W. Kim and D. Woelk. A Model of Authorization for Next-Generation Database Systems. ACM Transaction on Database Systems, Vol 126, No 1. March 1991, PP. 88-131.
4. E. Damiani, S.Vimercati, S.Paraboschi, and P.Samarati. Securing xml document. In Proc. of the 2000 International Conf.on Extending Database Technology (EDBT2000), Konstan, Germany, March, 2000, pp.121-135.
5. E. Damiani, S. Vimercati, S. Paraboachk and P.Samarati.XML Access Control Systems: A Component-Based Approach. In Proc. IFIP WG11.3 Working Conference on Database Security, The Netherlands, 2000. 8.
6. E. Damiani, S. Vimercati, S. Paraboachk and P.Samarati. Design and Implementation of Access Control Processor for XML Documents. Computer Network, 2000.
7. E. Damiani, S. Vimercati, S. Paraboachk and P.Samarati.A Fine-grained Access Control System for XML Documents. ACM Trans. Information and System Sec., Vol.5, No.2, May 2002.
8. E. Bertino, S. Castano, E. Ferrari, and M. Mesiti. Specifying and Enforcing Access Control Policies for XML Document Sources. WWW Journal, Baltzer Science Publishers, Vol.3, N.3, 2000.
9. E. Bertino, S. Castano, and E. Ferrai. Securing XML documents with Author-x.IEEE Internet Computing, May.June, pp.21-31, 2001.
10. E. Bertino and E. Ferrari. Secure and Selective Dissemination of XML Documents. TISSEC, 5(3), pp. 237-260, 2002.
11. E. Bertino, M. Braun, S. Castano, E. Ferrari, and M. Mesiti. Author-X: A Java-Based System for XML Data Protection.In Proc. IFIP WG11.3 Working Conference on Database Security, Netherlands, 2002.
12. A. Gabillon and E. Bruno. Regulating Access to XML Documents. In Proc. IFIP WG11.3 Working Conference on Database Security, 2001.
13. A. Stoica and C. Farkas. Secure XML Views. In Proc. IFIP WG11.3 Working Conference on Database and Application Security, 2002.
14. XML 1.0, second edition, http://www.w3.org/TR/2000/REC-xml-20001006
15. B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and Issues in Data Stream Systems. Invited Paper in Proc. of PODS, 2002.
16. M. Murata, A. Tozawa, and M. Kudo. XML Access Control Using Static Analysis. In ACM CCS, Washington D.C., 2003.
17. B. Luo, D. W. Lee, W. C. Lee, and P. Liu. Qfilter: Fine-grained Run-Time XML Access Control via NFA-based Query Rewriting. CIKM 2004
18. XML Fragment Interchange, W3C Candidate Recommendation 2001.
19. Sujoe Bose, Leonidas Fegaras, David Levine and Vamsi Chaluvadi, A Query Algebra for Fragmented XML Stream Data, DBLP 2003.
20. Leonidas Fegaras, David Levine, Sujoe bose and Vamsi Chaluvadi, Query Processing of Streamed XML Data, CIKM 2002, pp. 126-133.
21. Sujoe Bose and Leonidas Fegaras, XFrag: A Query Processing Framework for Fragmented XML Data, Web and Databases 2005.
22. Huan Huo, Guoren Wang, Xiaoyun Hui, Rui Zhou, BoNing, and Chuan Xiao, Effiecient Query Processing for Streamed XML Fragments, DASFAA 2006.
23. Masatoshi Yoshikawa, Toshiyuki Amagasa, et al.,XRel: A Path-Based Approach to Storage and Retrieval of XML Documents Using Relational Databases, ACM Transaction on Internet Technology, 2001.
24. Xiaodong Wu, Mong Li, Lee Wynne Hsu, A Prime Number Labeling Scheme for Dynamic Ordered XML Trees, ICDE, 2004.
25. Wenfei Fan, Irini Fundulaki, Floris Geerts, Xibei Jia, Anastasios Kementsietsidis, A View Based Security Framework for XML, AHM, 2006.

Author Index

İnan, Tolga, 47
Çadırcı, Isık, 47

An, Dong-Chan, 129

Boyrazoğlu, Burak, 47
Buhan, Serkan, 47

Carvalho, André Ponce de Leon F. de, 13

De Raedt, Luc, 83

Ermiş, Muammer, 47

Franke, Conny, 72

Gaber, Mohamed Medhat, 59, 72, 117
Gama, João, 13, 107
Gutmann, Bernd, 83

Haghighi, Pari Delir, 117
Hruschka Jr., Estevam R., 37

Iglesias, Juan R., 25

Küçük, Dilek, 47
Karnstedt, Marcel, 72
Krishnaswamy, Shonali, 117

Landwehr, Niels, 83
Last, Mark, 95
Lei, Hangsheng, 25
Loke, Seng, 117

Mohanty, Soumya, 25
Mukherjee, Soma, 25

Park, Seog, 129
Philipose, Matthai, 83
Phung, Nhan Duc, 59

Rodrigues, Pedro Pereira, 107
Roehm, Uwe, 59

Salor, Özgül, 47
Saveliev, Albina, 95
Schaal, Stefan, 1
Spinosa, Eduardo J., 13

Tang, Lappoon R., 25
Theodorou, Evangelos, 1
Thon, Ingo, 83
Ting, Jo-Anne, 1

Yoshida, Murilo Lacerda, 37

Zaslavsky, Arkady, 117