Pattern Recognition Letters 36 (2014) 171–176

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Interval clustering algorithm for fast event detection in stream monitoring applications

Hyeon Gyu Kim [a], Cheolgi Kim [b,*]

[a] Division of Computer, Sahmyook University, Hwarangro 815, Nowon-Gu, Seoul 139-742, Republic of Korea
[b] School of EECS, Korea Aerospace University, Hwajeon-dong, Deogyang-gu, Goyang-si 412-791, Republic of Korea

ARTICLE INFO

Article history:
Received 27 March 2013
Available online 10 October 2013

Communicated by M. Kamel

Keywords:
Interval clustering
Abnormal event detection
Error control
Max–min vector
Data streams

ABSTRACT

In stream monitoring applications, it is important to identify rapidly abnormal events over bursty data arrivals. By clustering similar conditions used in event detection, it is possible to reduce the number of comparisons and improve the event detection performance. On the other hand, event detection based on these clustered conditions can produce inaccurate results. Therefore, to use this method for critical applications, such as patient monitoring, the number of event detection errors needs to be kept to within a tolerable level. This paper presents an interval clustering algorithm that provides an error control mechanism. The proposed algorithm enables a user to specify a permissible error bound, and then uses the bound as a threshold condition for clustering. The simulation conducted based on real data showed that the algorithm improves the performance of event detection by clustering conditions while observing a user-specified error bound.

1. Introduction

Recently, applications that monitor streams of data items such as sensor readings, network measurements, auction bids, stock exchanges, web page visits, etc., have attacted considerable interest. In stream monitoring applications, it is important to identify abnormal events over bursty data arrivals in a timely manner (Babcock et al., 2002; Golab and Ozsu, 2003). For example, a patients' body status can be monitored using biosensors in a medical center. In this application, abnormal events (e.g., life-threatening events) should be detected on time and be notified to the medical staff immediately. The delayed detection of critical events may not be acceptable in this case. Similar examples can be found in network intrusion detection, plant monitoring, and so on.

In many applications, the conditions used for event detection can be represented as conjunctions of intervals. For example, *Body-Media's Armband* (Teller, 2004) predefines the possible body status, such as sleeping, exercising, reading, etc. The application then uses a range of biosensor data to identify the user's body status. A body status can be represented as a conjunction of intervals, each of which describes a range of normal sensing values. For example, the status "sleep" can be represented as interval conjunction of body temperature [36.0, 37.0] (°C), heart bit rate [80, 120], gait [0, 20], and others. An event that does not belong to any given status can be considered abnormal.

The number of conditions can become large in real-world applications. In this case, to meet the real-time constraint over bursty data arrivals, similar conditions can be clustered to reduce the number of comparisons. Many existing algorithms can be used for this purpose. Lingras et al. employed the *rough set theory* based on the *K-means clustering algorithm* to reflect unknown overlapping sets in clustering (Lingras and West, 2004). The same authors also proposed the use of *fuzzy clustering* as an alternative (Lingras and Yan, 2004). Asharaf et al. used the *rough set theory* in clustering based on the *leader clustering algorithm* (Asharaf et al., 2006). Regarding the dissimilarity measure, Souza and Carvalho (2004) introduced an adaptive method based on the city-block distance, where the dissimilarity between two intervals is measured adaptively by different weights. Chavent and Lechevallier (2002) used the *Hausdorff distance* to compare interval data.

One the other hand, none of these algorithms provides a method to control the amount of classification errors resulting from clustering. If event detection is performed based on the clustered intervals, the number of detection errors should be kept to within a tolerable bound. As an example, the medical center would limit the percentage of errors to less than 1% to avoid detection results with noise.

This paper proposes an interval clustering algorithm that provides an error control mechanism. The proposed algorithm enables a user to specify a permissible error bound which can be defined as a percentage of false positive errors that can occur during event detection. Given an error bound, the algorithm uses it as a threshold condition for clustering. In particular, when clustering a condition,

* Corresponding author. Tel./fax: +82 2 300 0123.
E-mail addresses: hgkim@syu.ac.kr (H.G. Kim), cheolgi@kau.ac.kr (C. Kim).

0167-8655/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.patrec.2013.09.017

the algorithm estimates the expected ratio of false positive errors based on the distribution of input events. The new condition can then be clustered only when its expected ratio is smaller than the given error bound.

The remaining part of this paper is organized as follows. Section 2 introduces the preliminary concepts relevant to interval clustering. Section 3 describes an error estimation measure and the interval clustering algorithm based on it. Section 4 provides the experimental results of the proposed algorithm. Section 5 concludes the discussion.

2. Preliminaries

Let X be a set of n conditions such that $X = \{x_1, \ldots, x_n\}$. A condition, x_j, is represented as a vector of m intervals such that $x_j = (x_j^1, x_j^2, \ldots, x_j^m)$, where $x_j^t = [a_j^t, b_j^t] \in I = \{[a, b] : a, b \in \mathbb{R}, a \leqslant b\}$ $(1 \leqslant t \leqslant m)$. A partition, $P = (C_1, C_2, \ldots, C_k)$ of X into k equivalent classes $(k \leqslant n)$, can then be found. Fig. 1 gives an example of clustering n conditions into k classes; the conditions in a class (or a cluster) need not to be consecutive, as shown in the figure.

A cluster, C_i, is also represented as a vector. The reference vector, v_i, can be obtained in various ways. In the *K-means clustering algorithm* (MacQueen, 1967), k conditions are selected randomly from X as the initial reference vectors. Each condition in X is then added to its nearest cluster. To measure the dissimilarity between two vectors, v_i and x_j, a simple $L1$ norm can be used, whose definition is given as follows. It is also called the *city-block distance function* in literature.

$$D(x_i, x_j) = \sum_{t=1}^{m} D'(x_i^t, x_j^t) \qquad (1)$$

where $D'(x_i^t, x_j^t) = |a_i^t - a_j^t| + |b_i^t - b_j^t|$

After all conditions are added to their nearest clusters, the K-means clustering algorithm updates the reference vectors repeatedly to minimize the within-cluster sum of squares:

$$\sum_{i=1}^{k} \sum_{x_j \in C_i}^{k} D(v_i, x_j)$$

The algorithm continues until all v_is converge.

On the other hand, in the *leader clustering algorithm* (Spath, 1980), the reference vectors can be created and updated dynamically. Whenever a new condition, x_{n+1}, is given, the algorithm first searches a cluster C_i closest to x_{n+1}, which is called a *leader*. The algorithm then checks whether the distance $D(v_i, x_{n+1})$ is less than or equal to a predefined threshold, T. If so, x_{n+1} is clustered into C_i, and then v_i is updated with the new member. Otherwise, x_{n+1} becomes a new leader, C_{k+1}.

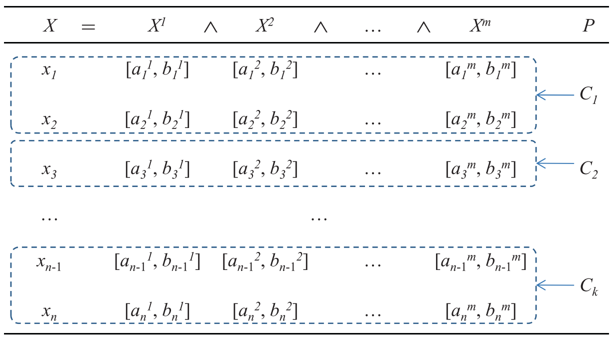
$X = X^1 \wedge X^2 \wedge \ldots \wedge X^m$
P
x_1 $[a_1^1, b_1^1]$ $[a_1^2, b_1^2]$ … $[a_1^m, b_1^m]$
x_2 $[a_2^1, b_2^1]$ $[a_2^2, b_2^2]$ … $[a_2^m, b_2^m]$
C_1
x_3 $[a_3^1, b_3^1]$ $[a_3^2, b_3^2]$ … $[a_3^m, b_3^m]$
C_2
…
x_{n-1} $[a_{n-1}^1, b_{n-1}^1]$ $[a_{n-1}^2, b_{n-1}^2]$ … $[a_{n-1}^m, b_{n-1}^m]$
x_n $[a_n^1, b_n^1]$ $[a_n^2, b_n^2]$ … $[a_n^m, b_n^m]$
C_k

Fig. 1. Clustering n conditions into k classes.

For monitoring applications, an incremental clustering algorithm can be used more appropriately. Many cases exist where the clusters need to be identified and updated gradually by monitoring on-the-fly input data. For example, the initial conditions of a biomedical device might have wide intervals to accommodate the diverse physiological status of each individual. On the other hand, as it is continuously used, the intervals will be adjusted to specific ranges of values to fit a user's body status and improve the accuracy of event detection.

The leader clustering algorithm has the best time and space complexity because it requires only a single scan of leaders to assign a new condition. On the other hand, the threshold T is determined heuristically in this algorithm. In general, as T becomes larger, more conditions can be clustered into leaders. Nevertheless, a large T value will generate more detection errors if event detection is performed based on the clustered conditions. Therefore, to use the algorithm for critical applications, such as patient monitoring, it is necessary to add an error control mechanism to this algorithm.

Now, let us discuss how errors can occur when the event detection is performed based on the clustered intervals. Suppose that two intervals, $x_1 = [a_1, b_1]$ and $x_2 = [a_2, b_2]$ $(a_1 < a_2 < b_1 < b_2)$, are added to a cluster whose reference vector v is a *mean vector* $[\alpha, \beta]$, where $\alpha = (a_1 + a_2)/2$ and $\beta = (b_1 + b_2)/2$. Consider that an input event, e, falls into the region $[a_1, \alpha)$. In this case, e is classified as abnormal because it does not belong to v, but it is normal in terms of x_1. On the other hand, if e falls into $[\alpha, a_2)$, it is considered normal though it is actually abnormal in terms of x_2. The former is called the *false positive error*, whereas the latter is called the *false negative error*.

As shown in the above, both types of errors can occur if the mean vector is used for clustering. Alternatively, it is also possible to use a *min–max vector* that covers all the intervals of the cluster's members as a reference vector (e.g., $v = [a_1, b_2]$). In this case, only false negative errors will occur.

Note that false negative errors cannot be accepted in monitoring applications. In the medical center example, life-threatening events of patients whose sensor values lie outside the normal body status must be classified as abnormal. Classifying those events as normal is unacceptable. On the other hand, false positive errors can be acceptable if the ratio of errors is tolerable, i.e., a small fraction of false alarms might be permissible. The proposed algorithm discussed below was designed to show only false positive errors when clustering intervals for event detection.

3. Proposed algorithm

The proposed algorithm can be viewed as an extension of the leader clustering algorithm, where an error control mechanism is augmented to it. To support error control, the algorithm estimates the amount of error incurred by clustering intervals. As a measure of the estimation error, the percentage of false positive errors was used, whose estimation is discussed in the first subsection. A clustering algorithm, which uses the estimation measure for error control, is then presented. To simplify our discussion, only one-dimensional intervals were considered in the algorithm. The algorithm was then extended to deal with multi-dimensional intervals, which is discussed in the following subsection.

3.1. Error estimation

To avoid false negative errors, the proposed algorithm uses a *max–min vector* to represent a cluster. Let each condition, x_j be given by a one-dimensional interval $[a_j, b_j]$. A max–min vector v_i for cluster C_i is represented by $[\alpha_i, \beta_i]$, where $\alpha_i = max_{x_j \in C_i}(a_j)$ and