Information Systems 38 (2013) 331–350

Contents lists available at SciVerse ScienceDirect

Information Systems

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

Mining neighbor-based patterns in data streams

Di Yang [a,*], Elke A. Rundensteiner [b], Matthew O. Ward [b]

[a] 1 Oracle Dr, Nashua, NH 03062, United States
[b] WPI, United States

ARTICLE INFO

Article history:
Received 15 September 2011
Received in revised form
2 June 2012
Accepted 13 August 2012
Recommended by: K.A. Ross
Available online 26 September 2012

Keywords:
Streaming
Pattern mining
Clusters
Outliers
Algorithms

ABSTRACT

Discovery of complex patterns such as clusters, outliers, and associations from huge volumes of streaming data has been recognized as critical for many application domains. However, little research effort has been made toward detecting patterns within *sliding window* semantics as required by real-time monitoring tasks, ranging from real time traffic monitoring to stock trend analysis. Applying static pattern detection algorithms from scratch to every window is impractical due to their high algorithmic complexity and the real-time responsiveness required by streaming applications. In this work, we develop methods for the incremental detection of neighbor-based patterns, in particular, density-based clusters and distance-based outliers over sliding stream windows. Incremental computation for pattern detection queries is challenging. This is because purging of to-be-expired data from previously formed patterns may cause birth, shrinkage, splitting or termination of these complex patterns. To overcome this, we exploit the "predictability" property of sliding windows to elegantly discount the effect of expired objects with little maintenance cost. Our solution achieves guaranteed minimal CPU consumption, while keeping the memory utilization linear in the number of objects in the window. To thoroughly analyze the performance of our proposed methods, we develop a cost model characterizing the performance of our proposed neighbor-based pattern mining strategies. We conduct an analysis study to not only identify the key performance factors for each strategy but also show under which conditions each of them are most efficient. Our comprehensive experimental study, using both synthetic and real data from domains of moving object monitoring and stock trades, demonstrates superiority of our proposed strategies over alternate methods in both CPU processing resources and in memory utilization.

1. Introduction

We present a new framework for detecting "neighbor-based" patterns in streams covering two important types of patterns, namely density-based clusters [18,17] and distance-based outliers [24,5] applied to sliding windows semantics [7,8]. Many applications providing monitoring services over streaming data require this capability of real-time pattern detection. For example, to understand the major threats of an enemy's airforce, a battle field commander needs to be continuously aware of the "clusters" formed by enemy warcrafts based on the objects' most recent positions extracted from the data streams reported from satellites or ground stations. We evaluate our techniques for this class of applications by mining clusters in the ground moving target indicator data stream [16]. As another example, a financial analyst monitoring stock transactions may be interested in the "outliers" (abnormal transactions) in the transaction stream, as they are potential indicators for new trends in the market. We evaluate our techniques for this class of application by mining outliers in the NYSE transaction stream [23].

Background on neighbor-based patterns: Neighbor-based pattern detection techniques are distinct from global clustering methods [32,22], such as *k*-means clustering. Global clustering methods aim to summarize the main characteristics of

* Corresponding author. Tel.: +1 508 243 9636.
E-mail addresses: di.yang@oracle.com, matt@cs.wpi.edu (D. Yang).

0306-4379/$ - see front matter © 2012 Published by Elsevier Ltd.
http://dx.doi.org/10.1016/j.is.2012.08.001

huge datasets by first partitioning them into groups (e.g., in Fig. 1, the objects in the same circles are considered to be in the same cluster), and then provide abstract information about the identified clusters, such as cluster centroids, as output. In these works, the cluster memberships of individual objects are not of special interest and thus not determined. In contrast, the techniques presented in this work target a different scenario, namely when individual objects belonging to patterns are of importance. For example, during the battlefield monitoring scenario, the commander may need to drill down to access the specific information about individual objects in the clusters formed by enemy warcraft. This is because some important characteristics of the clusters, such as the composition of each cluster (e.g., how many bomb carriers and fighter planes each cluster has) and the positions of the "super threats" in each cluster (e.g., the bomb carriers with nuclear bombs) can be learned from this specific information. Similarly, specific details about each outlier in the credit card transactions scenario may point to a credit fraud that may cause serious loss of revenue.

Thus our techniques focus on identifying specific objects that behave individually (for outliers) or together (for clusters) in some special manner. More specifically, the neighbor-based patterns are composed of object(s) with specific characteristics with respect to their local neighborhoods. Precise definitions of the patterns will be given in Section 2. Fig. 2 shows an example of two density-based clusters and a distance-based outlier in the dataset from Fig. 1.

Motivation for sliding window scenario: Another important characteristic distinguishing our work from previous efforts [13,12] is that we aim to mine for neighbor-based patterns within the sliding window scenario. The sliding window semantics, while widely used for continuous query processing [7], have rarely been applied to neighbor-based pattern mining. Sliding window semantics assume a fixed window size (either a fixed time interval or a fixed number of objects), with the pattern detection results generated based on the most recent data falling into the current sliding window. However, in previous clustering work [20,19,13,12], objects with different time horizons are either treated equally or assigned weights decaying as their recentness decreases. These techniques capture the accumulative characteristics of the full data stream seen so far, rather than isolating and reflecting about the features in the most recent stream portion. Using our earlier example, the position information of the warcraft may only be valid for a certain time period due to the movement of the monitored objects. In such cases, the sliding window technique is necessary as it forces the system to discard the out-of-date information and form the patterns only based on the most recent positions of the moving objects.

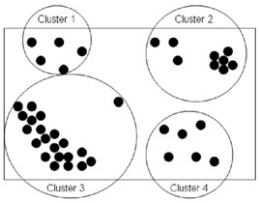

Fig. 1. Four global clusters determined by global clustering algorithms, such as *K*-means.

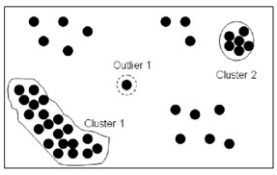

Fig. 2. Two density-based clusters and one distance-based outlier determined by neighbor-based pattern detection algorithms.

Challenges: Detecting neighbor-based patterns for sliding windows is a challenging problem. Naive approaches that run the static neighbor-based pattern detection algorithms from scratch for each window are often not feasible in practice, considering the conflict between the high complexity of these algorithms and the real-time response requirement for stream monitoring applications. Based on our experiments (see Section 11), detecting density-based clusters from scratch in a 50 K-object window takes around 100 s on our experimental platform, which does not meet the real-time responsiveness requirement for many interactive applications.

A straightforward incremental approach, which relies on incrementally maintaining the *exact* neighbor relationships (we will henceforth use the term "*neighborship*" for this concept) among objects, will also fail in many cases. This is because the potentially huge number of *neighborships* can easily raise the memory consumption to unacceptable levels. In the worst case, N^2 *neighborships* may exist in a single window, with N the number of data points in the window. Our experiments confirm that this solution consumes on average 15 times more memory compared to the from-scratch approach when applied to real datasets [16,23].

To overcome this serious strain on memory consumption, while still enabling the incremental computation, we introduce several *neighborship* abstractions that guarantee a linear in N memory consumption. However, designing solutions based on abstracted *neighborships* now come with a new shortcoming. Namely, the absence of exact *neighborships* makes discounting the effect of the expired objects from previously detected patterns become highly expensive in terms of CPU resources. This is because it is difficult to track what pattern structural changes, such as "splitting" or "termination", will be triggered by objects' expiration, without knowing which objects are directly connected to the expired objects and thus are affected.

Proposed methods: To make the abstracted *neighborships* incrementally maintainable in a computationally efficient manner, we propose to exploit an important characteristic of sliding windows, namely the "predictability" of the objects' expiration. Specifically, given a query window with fixed window size and slide size, we can predetermine all the windows in which each object can survive. A further insight gained from this "predictability" property leads us to propose the notion of the "predicted views". Namely given the objects in the current window, we can predict the pattern structures that will persist in subsequent windows and abstract them into the "predicted view" of each individual future window. The "view prediction" technique