Neurocomputing 181 (2016) 19–28

Contents lists available at ScienceDirect

Neurocomputing

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

An efficient algorithm for distributed density-based outlier detection on big data

Mei Bai *, Xite Wang, Junchang Xin, Guoren Wang

College of Information Science and Engineering, Northeastern University, Liaoning, Shenyang 110819, China

ARTICLE INFO

Article history:
Received 13 February 2015
Received in revised form
27 April 2015
Accepted 22 May 2015
Available online 3 December 2015

Keywords:
Density-based outlier
Local outlier factor
Distributed algorithm

ABSTRACT

The outlier detection is a popular issue in the area of data management and multimedia analysis, and it can be used in many applications such as detection of noisy images, credit card fraud detection, network intrusion detection. The density-based outlier is an important definition of outlier, whose target is to compute a Local Outlier Factor (LOF) for each tuple in a data set to represent the degree of this tuple to be an outlier. It shows several significant advantages comparing with other existing definitions. This paper focuses on the problem of distributed density-based outlier detection for large-scale data. First, we propose a Gird-Based Partition algorithm (GBP) as a data preparation method. GBP first splits the data set into several grids, and then allocates these grids to the datanodes in a distributed environment. Second, we propose a Distributed LOF Computing method (DLC) for detecting density-based outliers in parallel, which only needs a small amount of network communications. At last, the efficiency and effectiveness of the proposed approaches are verified through a series of simulation experiments.

1. Introduction

The outlier detection plays an important role in the area of data management and multimedia analysis, such as detection of noisy images and credit card fraud detection. According to Hawkins [1], "An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism". Thus far, there have been a large amount of studies aiming at outlier detection, and various kinds of definitions have been proposed, *e.g.*, DB-outlier [2], top-n outlier [3] and density-based outlier [4].

There exist lots of excellent algorithms for the outlier detection. However, most of them only focus on centralized environments. With the increasing amount of data, the processing efficiency of these algorithms becomes limited and cannot meet users' requirements. For instance, in the area of electronic commerce, we consider the users' trade information as a data set, and the abnormal trade records as outliers. Then the techniques of the outlier detection can help us to find the theft of user accounts and avoid the property damage. For many online shopping websites (*e.g.*, eBay and Amazon), huge amount of trade information is generated every day. It takes hours even days if we use traditional centralized algorithms to compute outliers. The time-effectiveness cannot be guaranteed. In this case, economic losses cannot be avoided. Therefore, it is quite necessary to design a parallel algorithm that can use multiple machines to accelerate the outlier computing.

This paper focuses on the problem of the outlier detection in distributed environments, and we follow the definition of density-based outlier [4]. Specifically, in a data set, for each tuple p, we calculate its local outlier factor (LOF) that represents the degree of p to be an outlier. LOF can reflect the isolation situation of p with respect to (w.r.t.) its surrounding neighbors. We will give the formalized description in Section 3. Comparing with other definitions of outliers, the density-based outlier has two advantages. (a) In other existing approaches [2,3], the outlier detection is considered as a binary problem (either an object in the data set is an outlier or not). Instead, for the density-based outlier, they tend to assign each object a degree of being an outlier, and they show it is more meaningful in many complex scenarios. (b) They point out that in many real-world data sets, whether a tuple t can be an outlier or not depends on its surrounding neighborhood (only the tuples which are closed to p). In this situation, LOF achieves better expressiveness than other existing approaches.

There exists only one study [5] focusing on the same problem with our paper. In [5], they adopt a master–slave architecture for distributed computing. Each slave node calculates its neighborhood set (it is the matrix that contains all the local tuples and their respective neighborhood) and sends it to the master node. The master node collects all the partial neighborhood sets and calculates LOFs of all the tuples. Clearly, this approach is not suitable for distributed outlier detection on large-scale data, because a large

* Corresponding author.
E-mail address: baimei861221@163.com (M. Bai).

http://dx.doi.org/10.1016/j.neucom.2015.05.135
0925-2312/

number of tuples are aggregated to the master node, and lots of calculations are needed to obtain the result. The master node becomes the bottleneck when the data scale is large.

In this paper, to detect density-based outliers in distributed environments efficiently, we propose several practical techniques, which are summarized as follows:

1. We propose the Gird-Based Partition algorithm (GBP) for data preprocessing. The algorithm first splits the whole data set into several grids, and then allocates these grids to the datanodes in a distributed environment. Using GBP, we can balance the workload on each datanode and reduce the network overhead while computing outliers.
2. We propose the Distributed LOF Computing method (DLC) for detecting density-based outliers in parallel, which includes two portions. First, based on the characters of LOF, we classify the tuples in a grid into two categories: grid-local tuples and cross-grid tuples. The grid-local tuples can be processed locally, and the network communications are required only for the cross-grid tuples. Then, we design a tailored method to minimize the number of tuples that need to be transmitted across the network.
3. We evaluate the performance of the proposed approaches through a series of simulation experiments. The experimental results show that the density-based outliers can be computed in parallel efficiently using GBP and DLC. The performance of our methods can meet the requirements of practical applications.

The rest of this paper is organized as follows. In Section 2, we briefly review the related work. Section 3 states the problem of density-based outlier detection in a distributed environment. Section 4 describes the details of GBP and DLC. Section 5 presents the experimental results. Finally, we conclude this paper in Section 6.

2. Related work

We summarize the existing definitions of outliers and related computing methods in Section 2.1. Then the previous approaches of distributed outlier computing are described in Section 2.2.

2.1. Definitions of outliers

The concept of outlier was presented by Hawkins [1] in 1980. In the recent decades, it has attracted the attention of many scholars, and several formal definitions of outliers were proposed.

Several earlier studies [1,6,7] focus on statistic-based approaches to detect outliers, where the tuples are modeled as a distribution and outliers are the tuples who show significant deviations from the assumed distribution. However, for high dimension data the statistic-based techniques are unable to build an appropriate model, which leads to performance degradation. Some non-statistical (model-free) approaches are proposed and they do not rely on the assumed data distribution. Knox and Ng [2] proposed the distance-based (DB) outlier. Given two parameters k, r, the neighborhood of a tuple p are the tuples whose distances to p are smaller than or equal to r, and p is determined to be a DB outlier if the number of its neighborhood is smaller than a given threshold k. They also presented a nested-loop (NL) method to compute DB outliers. Ramaswamy et al. [3] pointed out that Edwin's method lacks the ranking information for outliers. Thus they proposed a new definition, called D_n^k outlier, and proposed some efficient algorithms to compute outliers using clustering techniques.

Another model-free approach is the density-based outlier [4] that is adopted in our paper. Instead of directly determining whether a tuple is an outlier or not, a *local outlier factor* (LOF) that represents the *degree* of this tuple to be an outlier is assigned to each tuple. The LOF of a tuple p is computed by analyzing its local neighborhood. Comparing with other definitions, the density-based outlier shows several significant advantages that have been illustrated in Section 1.

Some studies aim at improving the computational efficiency for the outlier detection. Bay and Schwabacher [8] proposed a nested loop algorithm using randomization and a simple pruning rule, and they showed that the algorithm has near linear time performance on many large real data sets. Angiulli and Fassetti [9] proposed DOLPHIN. Through maintaining a small portion of data in the main memory, the entire data are required to be scanned twice to calculate outliers. Furthermore, numbers of indexing techniques (*e.g.*, R-tree [10]) are employed to accelerate the computing speed. Recently, there also emerge some outlier detection algorithms for special purposes, such as uncertain data [11], streaming data [12], and high dimensional data [13].

2.2. Outlier detection in distributed environments

Faced with the large-scale data, it takes a long time for outlier detection if we still use the centralized algorithms, and the efficiency is not satisfactory in most cases. Therefore, some researchers start to utilize multiple machines to speed up the calculation, and several methods [14–16] for distributed outlier detection were proposed.

Otey et al. [17] proposed an outlier definition for the data with mixed attributes, and designed a distributed method to detect outliers. They considered the data sets which contain a mixture of categorical and continuous attributes. The computation process includes two steps. In the first step, they designed the anomaly score function and computed the locally anomaly scores for all the tuples. In the second step, a global schema is used to recalculate the anomaly scores of tuples whose locally scores are larger than the threshold. Finally, the outliers were chosen according to their anomaly scores.

Angiulli et al. [18] proposed a method for top-n outlier detection in distributed environments. In the first iteration, each salve node randomly selects n tuples and transfers them to the master node. In the master node, the neighborhoods of these tuples are calculated and the top-n tuples are chosen to filter out the local tuples in the salve nodes. In the second iteration, another n tuples in each salve node are randomly chosen and transferred to the master node. The master node recalculates the top-n tuples and uses them to prune local tuples. The process is repeated until all the local tuples are pruned. This method is complex and needed multiple iterations. Furthermore, most of the computations occur on the master node which would be a bottleneck for large-scale data.

Lozano and Acufia [5] proposed a distributed algorithm to compute density-based outliers, which targets at the same issue in our paper. The proposed algorithm is a parallel version of Breuning's method [4]. However, as we mentioned in Section 1, they adopt a master–slave architecture. In the slave node, the local neighborhood of each tuple is calculated locally. Then all the tuples and their neighborhood are transferred to the master node. The final result is calculated in the master node. Since all the tuples are transferred to the master node, the workload on the master node is quite heavy. Thus, this approach cannot achieve good performance when the data scale is large.

Outlier detection has some applications in the graph-based data management. Specifically, many graph-based models are used as geometric image descriptors [20] to enhance image categorization. Besides, these methods can be used as image high-order potential descriptors of superpixels [21–23]. Further, graph-based descriptors can be used as a general image aesthetic