LSC-Mine: Algorithm for Mining Local Outliers

Malik Agyemang
Department of Computer Science
University of Calgary
2500 University Drive N.W.
Calgary, AB, Canada

agyemang@cpsc.ucalgary.ca

Christie I. Ezeife*
School of Computer Science
University of Windsor
401 Sunset Avenue
Windsor, Ontario, Canada

cezeife@uwindsor.ca

Abstract

Data objects which differ significantly from the remaining data objects are referred to as outliers. Density-based algorithms for mining outliers are very effective in detecting all forms of outliers, where data objects with fewer neighbors are likely to be outliers than are those with more neighbors. However, existing density-based algorithms engage in huge repetitive computation and comparison for every object before the few outliers are detected. Expensive computations might make scalability of these techniques to important applications like quick fraud detection unfeasible. This paper proposes LSC-Mine algorithm based on the distance of an object and those of its knearest neighbors. In addition, data objects that are not possible outlier candidates are pruned which reduces the number of computations and comparisons in LOF technique resulting in an improved performance.

1. Introduction

Data mining is a non-trivial process of identifying valid, potentially useful and understandable patterns in data [6]. Most data mining tasks have concentrated on finding frequent patterns while discarding the less frequent ones, but the less frequent patterns that are usually eliminated contain another group of objects often described as nuisance, noise or outliers. Outliers are observations that deviate from other observations within the same group to arouse suspicion that they were generated using a different mechanism [7]. In many data mining applications identifying exceptions or rare events can often lead to the discovery of interesting and unexpected knowledge in areas such as credit card fraud detection, cellular phone cloning fraud and detection of suspicious activities. Some existing algorithms in machine learning and data mining have only tolerated outliers in whatever problems those algorithms are solving [12]. ROR in [1] removes identified outliers before any further processing is done on the data. The problem of locating outliers in a large dataset is like finding needles in a haystack. The difficulty of identifying outliers has resulted in different definition and

* This research was supported by the National Science and Engineering Research Council (NSERC) of Canada under an operating grant (OGP-0194134) and a University of Windsor grant

techniques for mining them. Knorr et al. [12, 13] described an object O in dataset T as a DB(p,D)-outlier if at least fraction p of the objects in T lies at a distance greater than D from O. Ramaswamy et al. [16] however, described an outlier as an observation that has the greatest distance from all other observations when the calculated distances are ranked. In [4], an outlier is viewed as an observation with the highest local outlier factor within a given neighborhood. The local outlier factor is a measure of the distribution or density of objects around a particular point. A high local outlier factor means that the neighbors are far away, whereas a low local outlier factor indicates nearer neighbors. In this paper, outlier mining algorithms are grouped into 4 categories; distribution-based, depth-based, distance-based and density-based outlier mining techniques. We concentrate on density-based outlier mining techniques with a proven ability of efficiently identifying all forms of outliers [4].

1.1. Related Work

Distribution-based techniques are mostly found in statistics where standard statistical distributions (e.g. Normal, Poisson, Student t etc) are used to fit data points. Outliers are objects that show different characteristics from the rest of the data objects [5, 7]. Distribution-based methods require upfront knowledge of the statistical distribution of the data, and are available mostly for single variable data. In depth-based techniques, data objects are organized into layers with the expectation that shallow layers are more likely to contain outlying data than are deep layers. Ruts et al. [15] developed ISODEPTH for computing 2-D depth contours. In [11], an algorithm robust against collinear points is developed to address the major drawbacks of ISODEPTH. Depth-based methods are supposed to work for high dimensional data but in practice they do not work for more than three dimensions.

Knorr et al. [12, 13] proposed the distance-based outlier concept to address the problems of the earlier outlier mining techniques. A distance-based outlier is described as 'fraction *of data objects with distance greater than the minimum outlier distance'*. The choice of the minimum outlier distance is left to the discretion of the miner. The nested-loop and the index–based algorithms proposed for mining distance-based outliers have a time complexity of $O(kN^2)$ which is linear in the number of dimensions but quadratic in data size. In [15], a new definition for outliers is proposed which does not require the user to specify the *minimum outlier distance which also* allows outliers to be ranked. In [2], a distance-based outlier that takes into account weighted neighborhood distances is proposed.

Outliers are data objects with larger weights computed using the *Hilbert space filling curve algorithm*. The proposed algorithm scales linearly in both dimensionality and data size.

Breunig et al. [4] claim that every object has some degree of "outlierness" in it and that being an outlier is not just a binary property. The degree of outlierness called local outlier factor (LOF) depends on the remoteness of an object with respect to its surrounding neighborhood. Outliers are objects that tend to have high LOF values. The LOF algorithm is able to detect all forms of outliers including those that could not be detected by the distance-based algorithms. In [10], a micro-cluster-based algorithm for finding top-n local outliers is proposed. The pruning of the clusters based on the computed bounds reduces the number of LOF value computations but it still relies on computing reachability distances and local reachability densities for every object in the data set. The notion of a semantic outlier that integrates semantic knowledge of the underlying data is presented in [8]. A semantic *outlier is defined as a data point which behaves differently with other data points in the same class. The* degree of semantic outlying is called semantic outlier factor (*SOF*). The concept is very interesting and useful but it works mostly on categorical data and as such can only supplement the other outlier mining algorithms rather than replace any of them.

1.2. Contributions

This paper proposes LSC-Mine algorithm for mining outliers based on the distances of objects from their nearest neighbors without actually computing their reachability distances and local reachability densities. In addition, data objects that are not possible outlier candidates are pruned as soon as they are detected, making LSC-Mine algorithm less expensive and more efficient compared to other related outlier mining algorithms described in the literature.

1.3. Outline of Paper

Section 2 presents an example mining of outliers with LOF algorithm to expose the contributions of the proposed technique. Section 3 presents LSC-Mine algorithm. Performance analysis of the algorithms is presented in section 4. Finally, section 5 presents conclusions and future work.

2. An Example Outlier Mining with LOF

Since LSC-Mine algorithm being proposed in this paper is related to LOF algorithm, this section uses an example to show outlier mining with LOF [4]. In LOF algorithm, outliers are data objects with high *LOF* values whereas data objects with low *LOF* values are likely to be normal with respect to their neighborhood. High *LOF* is an indication of low-density neighborhood and hence high potential of being an outlier. The sequence of steps involved in computing *LOF* value of an object p in a dataset are: (1) computing kdistance of p, (2) finding kdistance neighborhood of p where k is a positive integer called the Minpts, (3) computing reachability distance of p, (4) local reachability density of p, and (5) computing and ranking of LOF. MinPts is the minimum number of objects desired to be in a neighborhood. Let D be a database having four objects denoted as P_1, P_2, P_3 and P_4 with distances $P_1P_2 = 4$, $P_1P_3 = 3$, $P_1P_4 = 7$, $P_2P_3 = 5$, $P_2P_4 = 6$, and $P_3P_4 = 8$ obtained using a known distance function and MinPts (k) = 2.

Step1: Computing kdistance of p:

The motive for computing kdistance of p is to determine the neighbors of p. In simple terms, k-distance of p is the maximum distance from object p when every object in the data set is considered to have at least kneighbors. kdistance of p, denoted as kdistance(p) is obtained as follows:

i. First, compute the distances of all objects from P_1 using a distance function. The distances are $P_1P_2 = 4$, $P_1P_3 = 3$, $P_1P_4 = 7$ (assumed previously).
ii. Next, select the first 2 distinct minimum distances from P_1. All distances from P_1 are ordered and the first 2 minimum distinct distances are chosen (i.e., Min($P_1P_2 = 4$, $P_1P_3 = 3$, $P_1P_4 = 7$)).
iii. Finally, the maximum of the first 2 minimum distinct distances is selected as kdistance of P_1. Thus, kdistance(P_1) = max(3, 4), hence, kdistance(P_1) = 4. The kdistances of the remaining objects are similarly obtained.

Step 2: Finding kdistance neighborhood of p

The kdistance neighborhood of p denoted ($N_k(p)$), contains every object with distance not greater than kdistance(p). The rationale for computing the kdistance neighborhood is to find the nearest neighbors of each object. For instance, kdistance neighborhood of P_1 contains P_2 and P_3 since kdistance (P_1) is 4 and the distances of P_2 and P_3 from P_1 are each not more than 4 (i.e., $P_1P_2 = 4$, $P_1P_3 = 3$)

Step 3: Computing reachability distance of p

The reachability distance of an object p with respect to object o is the distance(p, o) or kdistance(o) whichever is larger ($reachdist_k(p,o) = \max\{kdistance(o), distance(p,o)\}$). The objective is to ensure that

all the objects within a neighborhood are homogeneous. In addition, LOF stabilizes when the objects within a neighborhood are uniform even if MinPts (k) changes. The fluctuations in the reachability distances can be controlled by choosing large values for k [4]. The reachability distance of P_1 is computed as follows: First, identify kdistance neighborhood of P_1 (i.e., $N_k(P_1) = (P_2, P_3)$). The reachability distance of P_1 is computed with respect to P_2 and P_3 since they constitute the neighbors of P_1.

For P_2 within the neighborhood of P_1: $reachdist_k(P_1, P_2) = \max(kdistance(P_2), distance((P_1,P_2)) = \max(5,4) = 5$. Since $kdistance(P_2) = 5$ and $distance(P_1,P_2) = 4$

For P_3 within the neighborhood of P_1: $reachdist_k(P_1,P_3) = \max\{kdistance(P_3), distance(P_1,P_3)\} = \max(5, 3) = 5$. Hence, $reachdistk(P_1,o) = (5, 5)$, which is the combination of reachability distances of the neighbors of P_1.

Step 4: Computing the local reachability density of p

The local reachability density of an object p, denoted $lrd_k(p)$ is the inverse of the average reachability distances from kdistance neighbors of p. It provides a means for comparing reachability distances.

$$lrd_k(p) = 1 \Big/ \left(\frac{\sum_{o \in N_k(p)} reachdist_k(p,o)}{|N_k(p)|} \right)$$

The local reachability density for P_1 is computed as follows: $lrd_k(P_1) = 1/\{(5+5)/2\} = 2/10$, since (5, 5) constitutes the local reachability distance of P_1 and the number of kdistance neighbors is 2. Also, $lrd_k(P_2) = 2/9$, $lrd_k(P_3) = 2/9$ and $lrd_k(P_4) = 2/13$.

Step 5: Local outlier factor of p

The local outlier factor is a ratio that determines whether or not an object is an outlier with respect to its neighborhood. The local outlier factor of an object p denoted $LOF_k(p)$ is the average of the ratios of local reachability density of p and that of p's knearest neighbors.

2.1. Limitations of the LOF Algorithm

The major drawback of LOF algorithm lies in computing reachability distances defined as $reachdist_k(p,o) = max\{kdistance(o),distance(p,o))$. Computing reachability distance of p involves computing distances of all objects within p's neighborhood, and each compared with the kdistance of that neighborhood which is very expensive when MinPts is large. Secondly, LOF has to be computed

for every object before the few outliers are detected. This is not a desirable exercise since outliers constitute only a small fraction of the entire dataset

3. The LSC-Mine Algorithm

The identified problems are addressed by proposing LSC-Mine based on the original idea of local outliers. LSC-Mine avoids computing reachability distances and local reachability densities which are considered very expensive in LOF. Instead, local sparsity ratio derived from the neighborhood distances is computed. Additionally, LSC-Mine prunes data objects that are not possible outlier candidates using pruning factor computed from the neighborhood distances. The remaining data forms the candidate set on which outliers are determined.

Definition 3.1: The local sparsity ratio of an object p denoted $lsr_k(p)$ is defined as the ratio of the cardinality of kdistance neighborhood of p to the sum of all the actual distances in that neighborhood.

$$lsr_k(p) = \frac{|N_k(p)|}{\sum_{o \in N_k(p)} distofN_k(p)}$$

,where $distofN_k(P)$ consists of actual distances of objects in kdistance neighborhood of p. The local sparsity ratio measures the concentration of objects around an object p. Objects with low local sparsity ratios have high potential of being outliers and vise versa. The final declaration of outliers depends on the local sparsity coefficient rather than the ratio. The pruning factor is rooted on the assumption that the local sparsity ratio of an object p in a dataset should not be less than a similar ratio computed from the entire data if that object is not an outlier. It gives an upper bound for any object that is an outlier candidate.

Definition 3.2: The pruning factor (Pf) is the ratio of the sum of the absolute neighborhood distances to the overall sum of the actual neighborhood distances. Mathematically,

$$Pf = \frac{\sum |N_k(p)|}{\sum \sum_{o \in N_k(p)} distof N_k(p)}$$

Once the pruning factor is obtained, any object with a local sparsity ratio less than Pf is removed since it cannot be a potential outlier candidate. The pruning can remove more than half of the data thereby enhancing the performance of LSC-Mine.

Definition 3.3: The local sparsity coefficient of p denoted $LSC_k(p)$ is the average ratio of the local sparsity ratio of p to that of its knearest neighbors.

$$LSC_k(p) = \frac{\sum_{o \in N_k(p)} \frac{lsr_k(o)}{lsr_k(p)}}{|N_k(p)|}$$

A high local sparsity coefficient indicates the neighborhood around an object is not crowded and hence a higher potential of being an outlier whereas a low local sparsity coefficient indicates a crowded neighborhood and hence a lower outlying potential. The formal sequence of steps followed by the LSC-Mine algorithm for computing outliers is presented in Figure 1. The LSC-Mine algorithm will compute for each object (1) its kdistance, (2) its kdistance neighborhood, (3) its local sparsity ratio, (4) its pruning factor, (5) candidate set that is not to be pruned, (6) LSC of objects in candidate set and finally, (7) rank objects with the highest LSC as strongest outliers. Definitions 3.1 to 3.3 give the formulars for computing these values.

Algorithm LSC-mine()
Input: Data objects, integer k
Outputs: Ranked list of n objects with highest LSC
Other variables: kdistance neighborhood of each object,
local sparsity ratio of each object. Candidate set, pruning factor
(1) Compute the kdistance of each object
(2) Find kdistance neighborhood of each object
(3) Compute local sparsity ratio of each object
(4) Compute the pruning factor of each object
(5) Obtain the candidate set
(6) Compute LSC using the candidate set
(7) Rank outliers as those with the highest local sparsity coefficients
end. // LSC-mine//

Figure 1: The LSC-Mine outlier

4. Performance Analysis

The performance analysis of LSC-Mine and LOF algorithms are presented in this section. All experiments were performed on a 1.8GHz Intel Pentium 4 PC with 256 megabytes main memory, running Windows 2000 Professional Edition. The programs are written in Java. Two sets of data are used to test the algorithms for correctness and response time. The data for testing correctness is

obtained from the National Hockey League (NHL) player statistics for 1995 whereas the data for testing response time was generated.

4.1. Testing for Correctness

The National Hockey League (NHL) player statistics of 1995 used for this experiment had 805 records with unknown number of outliers. Four fields (name, total score, plus/minus, and penalty minutes) were chosen from the statistics to constitute our data set. In addition, each record was assigned a unique identifier for evaluating the results. The experiment was to determine how many outliers identified using LOF are correctly identified by LSC-Mine. In particular, we were interested in the number of top-n records determined as outliers, as well as their positional rankings in the two algorithms. The two algorithms were run using MinPts of 100, 200, 300 and 400. For MinPts 100 and 200, the two algorithms identified the same records as top 10 outliers with slight differences in their positional rankings. LSC-Mine and LOF produced the same top-10 outliers for MinPts 300 and 400.

4.2 Testing for Response Time

The data generated for testing response time contained 16,000 records with structure similar to [14] and unknown number of outliers. Two sets of experiments were conducted for response time. The first was to ascertain the effect of MinPts on response time while the other was to find the effect of data size on response time for the two algorithms. The execution times for both LSC-Mine and LOF algorithms were recorded. The results of the experimental runs are discussed in subsequent sections.

The Effect of MinPts on response time was conducted using different MinPts. The results depicted in Figure 2 shows that LSC-Mine performs better than LOF with LOF showing steady increases in response time with increasing MinPts. LSC-Mine on the other hand, shows almost a constant response time with increasing MinPts.

Finally, the effect of data size on response time was tested using MinPts of 500. The two algorithms were run using different data sizes and the response time taken. The results show irrespective of MinPts LOF and LSC-Mine algorithms perform almost the same with data sizes less than 2000, but as the data size increases, the response time for LOF also increases rapidly. The response time for LSC-Mine increases steadily as data size increases. Figure 3 depicts the obtained

results. The difference is the response times the two algorithms will be very remarkable with very large datasets hence making LSC-Mine a very good choice for warehouse applications.

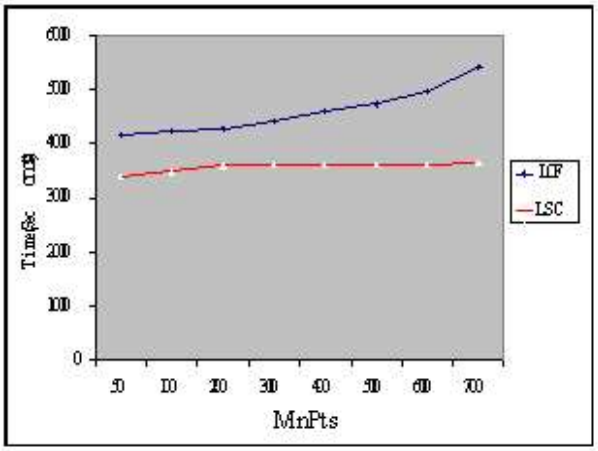

Figure 2: Effect of MinPts on response time

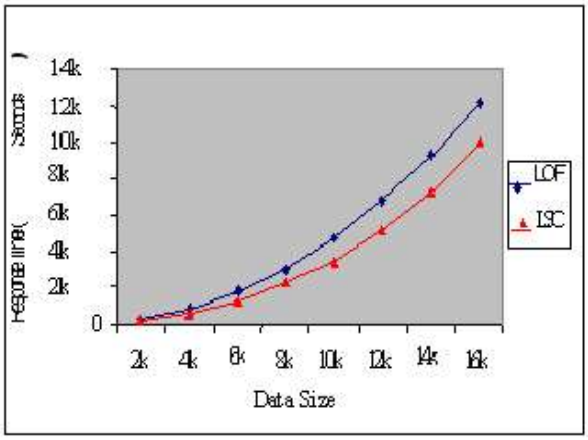

Figure 3: Effect of data size on response time

5. Conclusions and Future Work

Outlier mining is very important data mining activity, which has not received much attention in the research community. But finding rare activities such as detecting credit card fraud or cellular phone cloning is likely to be more interesting than finding how often a regular customer visits the ATM. Density-based approach to outlier-mining makes monitoring customer activities even more interesting since every customer activity has a potential of being exceptional (outlier). This paper contributes an enhancement to the Local Outlier Factor (LOF) algorithm called LSC-Mine algorithm. LSC-Mine improves upon the response time of LOF by avoiding the computation of reachability distances and local reachability densities. In addition, data objects that are not likely outlier candidates are pruned as soon as they are identified. The pruning drastically reduces the number of Local Sparsity Coefficient Computation. The outlier candidate set is then used as input for calculating local sparsity coefficient. Experimental results show that LSC performs better than LOF with respect to response time for all sizes of data and MinPts.

Areas of future research include finding what fraction of the entire data should be assigned to MinPts and the application of outlier mining techniques to text data and exploring more on the application of the outlier mining to spatial data and Web data. Application of outlier mining techniques to spatial data is also very promising.

References

[1] A. Adam, E. Rivlin, and I. Shimshoni, "ROR: Rejection of Outliers by Rotations" IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 23 No. 1 Jan. 2001.

[2] F. Anguilli, and C. Pizzuti, "Fast Outlier Detection in High Dimensional Spaces" T. Elomaa et al. (Eds): PKDD, LNAI 2431, 2002, pp 15-27.

[3] C. Aggarwal, and P.S Yu, "Outlier Detection for High Dimensional Data", In Proc. of ACM SIGMOD 2001 Int. Conf. on Management of Data, Santa Barbara, CA,USA. May 2001.

[4] M. Breunig, H-P. Kriegel, R.T. Ng, and J. Sander, "LOF: Identifying Density-Based Local Outliers", Proc. of ACM SIGMOD 2000 Int. Conf. on Management of Data, Dallas, TX 2000.

[5] V. Barnett, and T. Lewis, "Outliers in Statistical Data ", John Willey, 1994

[6] U. Fayyad, G. Piatesky-Shapiro, and P. Smyth, "Knowledge Discovery and Data Mining: Towards a Unifying Framework" Proc. 2nd Int. Conf. on knowledge Discovery and Data Mining, Portland, OR 1996, pp82-88.

[7] D. Hawkins, "Identification of Outliers", Chapman and Hall, London, 1980

[8] Z. He, S. Deng, and X. Xu, "Outlier Detection Integrating Semantic Knowledge" X. Meng, J. Su and Y. Wang (Eds): WAIM 2002, LNCS 2419, 2002, pp 126-131.

[9] M.V Joshi, R.C. Agarwal, and V. Kumar, "Mining Needles in Haystack: Classifying Rare Classes via Two-Phase Rule Induction" In Proc. of ACM SIGMOD, Santa Barbara, CA, USA. May 2001.

[10] W. Jin, A.K.H. Tung, and J. Han, "Mining Top-n Local Outliers in Large Databases", In Proc. of KDD 2001, San Francisco, CA, USA, 2001.

[11] T. Johnson, I. Kwok, and R. Ng, "Fast Computation of 2-D Depth Contours", In Proc. of KDD 1998, pp 224-228.

[12] E.M. Knorr and R.T. Ng, "A Unified Notion of Outliers: Properties and Computation", In Proc. of KDD 97, 1997, pp 219-222.

[13] E.M. Knorr, and R.T. Ng, "Algorithms for Mining Distance-Based Outliers in Large Dataset", In Proc. of the 24th VLDB Conference, New York, USA, 1998.

[14] Notational Hockey League Player Statistics, "www.lcshockey.com", Player Statistics for 1995 and 1996.
[15] I. Ruts, and P. Rousseuw, "Computing Depth Contours of Bivariate Points Cloud", Computational Statistics and Data Analysis, 1996, 23:153-16.

[16] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient Algorithms for Mining Outliers from Large Data Sets" In Proc. of ACM SIGMOD 2000, USA, pp127-138.