FastLOF: An Expectation-Maximization based Local Outlier Detection Algorithm

Markus Goldstein
German Research Center for Artificial Intelligence (DFKI)
Markus.Goldstein@dfki.de

Abstract

Unsupervised anomaly detection techniques are becoming more and more important in a variety of application domains such as network intrusion detection, fraud detection and misuse detection. Today, unsupervised anomaly detection techniques are mainly based on quadratic complexity making it almost impossible to apply them on very large data sets. In this paper, an Expectation-Maximization algorithm is proposed which computes the Local Outlier Factor (LOF) incrementally and up to 80% faster than the standard method. Another advantage of FastLOF is that intermediate results can be used by a system already during computation. Evaluation on real world data sets reveal that FastLOF performs comparable to the best outlier detection algorithms although being significantly faster.

1 Introduction

Anomaly Detection is an area of machine learning, which deals with the detection and rating of outliers in data sets. According to its application domain, it is also known as outlier detection, intrusion detection (in the network security domain), behavioral analysis (in forensics) as well as misuse or fraud detection [4]. The latter two are often used with respect to the analysis of (financial) transactions or monitoring of database systems in data leakage prevention (DLP) applications. As a simple example in this context, accesses to a database server can be logged and features of individual users can be extracted and fed into an anomaly detection algorithm. As a result, users behaving different than the majority can be identified - for instance users requesting an out of ordinary amount of data records or users making requests at unusual times of the day helps to identify potential fraudsters.

From a mathematical point of view, all of these application domains have in common that normal behavior should be modeled and abnormal occurrences need to be detected. This leads to two simple assumptions on which anomaly detection is based:

- anomalies only occur rarely in the data set and
- their features significantly deviate from normal data.

In general, the definition of anomaly detection is often ambiguous. However, the survey paper [4] found some categorization among all publications in this area. One essential definition is determined with respect to the separation of training and testing data: *Supervised anomaly detection* uses a labeled training set with normal data and annotated anomalies. Usually, traditional classificators (such as neural networks, SVMs or k-nearest-neighbors) can be used in this area. *Semi-supervised anomaly detection* is based on training data containing normal data only and no anomalies. Important application domains in this scenario are intrusion detection and anti-virus applications, where normal operations are known but attacks and viruses are unknown and should be detected. For semi-supervised anomaly detection tasks, one-class classificators [11] such as One-Class-SVMs or Anomaly Trees [15] can be used. Finally, *unsupervised anomaly detection* is the most flexible but also the most error prone anomaly detection approach. In this context, no assumption of the data is made, i.e. the data contains normal and anomalous records and algorithms should separate the two without prior training. Many practical applications refer to this anomaly detection scenario, for example the analysis of log files [9], data leakage prevention and intrusion detection [13]. In this paper, also an unsupervised algorithm FastLOF is proposed which is based on LOF [3] and speeds up the computation significantly.

2 Related Work

2.1 Anomaly Detection

In unsupervised anomaly detection, there exist three main techniques [4]: (1) Nearest-neighbor based, (2)

Clustering-based and (3) Statistical methods. Statistical methods, parametric as well as non-parametric ones, mostly operate on univariate data whereas the few existing multivariate approaches are too compute intense for large data sets. Clustering-based methods such as the Clustering-based Local Outlier Factor (CBLOF) [7] can deal very well with large datasets since complexity can be low, for example $O(NlogN)$ if k-means is used. However, cluster-based methods have been found to perform worse than nearest-neighbor based methods [1] and are therefore rarely applied.

The vast majority of algorithms used in practical applications are nearest-neighbor based, for example the global k-nn algorithm [14, 2] or the well-known Local Outlier Factor (LOF) [3]. Many improvements based on the LOF idea have been presented, such as Connectivity-Based Outlier Factor (COF) [16], Local Outlier Probability (LoOP) [10], the Influenced Outlierness (INFLO) [8] or the parameter-free Local Correlation Integral (LOCI) [12]. The common ground of all these proposed methods is the determination of the k-nearest-neighbors in advance. Except for LOCI, this is the most compute intense part of the algorithms with a complexity of $O(N^2)$. Attempts to reduce this complexity are reviewed in Subsection 2.3.

2.2 Local Outlier Factor (LOF)

Since the Local Outlier Factor (LOF) introduced by Breunig et al [3] in 2000 is used frequently in practical systems today and also is the basis for the FastLOF algorithm, it is reviewed in the following.

In a first step, the k-nearest-neighbors according to the euclidean distance are found, which is the most compute-intense step. In general, all data instances have to be compared with all other remaining ones leading to a computational complexity of $O(N^2d)$, whereas d is the dimension of the data.

In a second step, the Local Reachability Density (LRD) is computed for all data points p based on the set of k neighbors $N_{min}(p)$, such that

$$LRD_{min}(p) = 1/\left(\frac{\sum_{o \in N_{min}(p)} reach\_dist_{min}(p,o)}{|N_{min}(p)|}\right) \quad (1)$$

The LRD can be understood as the (inverse of) the average reachability distance of the nearest neighbors. In this context, the reachability distance $reach\_dist_{min}$ is defined as the k-distance or the Euclidean distance of the two objects, depending on which value is larger [3].

In a third step, the LOF value can be computed by using the LRDs from the k-nearest-neighbors, such that

$$LOF_{min}(p) = \frac{\sum_{o \in N_{min}(p)} \frac{LRD_{min}(o)}{LRD_{min}(p)}}{|N_{min}(p)|} \quad (2)$$

From the above equation we see that the LOF is a ratio of the LRDs. A large LOF will be assigned if the density of all neighbors is higher than the density of the data instance p itself, indicating a possible outlier. On the other hand, if the densities of the neighbors are approximately as high as of the instance itself, the resulting ratio will be close to one. A rule of thumb states, that outliers have larger LOF scores than a threshold in the range of 1.2 to 2.0, depending on the data.

2.3 Performance Improvement Attempts

In practice, more than 99% of the computation time for LOF is spent with finding the nearest neighbors, whose complexity is $O(N^2d)$. Computing the LRDs and the LOF itself is only $O(N)$. Simplifying the LRD and LOF computation as suggested by Chiu et al [5] in LOF' and LOF" might thus not improve the total computation time. The authors also suggested GridLOF as an improvement by pruning away dense areas. However, it requires a good manual grid definition such that it fits the data well, which is usually not feasible in practice. The authors of aLOCI [12] also suffer from this problem and suggest to simply try out different grids and use the best. Amer [1] introduced some implementational improvements to the original LOF including a removal of duplicate data points from the data set and a smart parallelization based on the number of dimensions but still with quadratic complexity. Two well-known techniques for speeding up a k-nearest-neighbor classifier also have been applied: Space Partitioning and Locality Sensitive Hashing (LSH). Considering the first, search-trees such as kd-trees [9] or X-trees are used. Using trees as index structures has the advantage of having fast k-NN query times, depending on the tree down to $O(logN)$. This comes at the cost for building such a tree, for example $O(N(d+logN))$ for a kd-tree. The building time and the query time might be lower than a full search, but this usually only holds for low dimensions and a large number of samples ($N >> 2^d$). Another downside of using trees is the fact that usually the tree structure and thus all the data need to be kept in memory. Due to this disadvantages, Locality Sensitive Hashing (LSH) [6] is used very intense in the last years for high dimensional data nearest neighbor searches. The idea is that similar instances are mapped into the same "buckets" such that the nearest-neighbor search is only performed locally. LSH works very well in dense areas but not in areas with low density. This

makes it ideal for retrieval tasks but not for anomaly detection, where the opposite is needed: Precise matches for low-dense areas and approximate matches for high-dense areas.

3 FastLOF Algorithm

Our key observation that for LOF computation the precise nearest neighbors are important for the outliers and a good estimation is fair enough for normal data points, the FastLOF algorithm is introduced. The basic idea is inspired by using an Expectation-Maximization approach: Instead of computing the nearest neighbors first and then computing the LRD and LOF values, this is done alternately. First, the data set is randomly divided into data chunks. Then, for each data point, the k-nearest-neighbor search is performed only within a single chunk. Using this k-nearest-neighbor estimation, LRD and LOF for all data points are computed as in Equation 1 and 2. Then, the LOF values are used to determine for which data points it makes most sense to find better nearest neighbors determined by a threshold θ. Instances with an LOF close to one will not be processed further, instances with a higher LOF will be used for searching better neighbors. The pseudo code of FastLOF can be found in Algorithm 1. The alternating calculation of the LOF scores of active instances and the threshold based assignment of it is typical for an expectation-maximization algorithm. Note that the computation of LRD and LOF always have to be performed for all instances, because newly found nearest neighbors of outliers can also be new neighbors for currently not investigated instances. This also ensures, that the algorithm doesn't miss an outlier accidentally, which could have been declared as "normal" in a very early iteration (cf Figure 2).

Algorithm 1 The FastLOF algorithm

Input
  $D = d_1, \ldots, d_n$: data set with N instances
  c: chunk size (e.g. $\sqrt{N}$)
  θ: threshold for LOF
  k: number of nearest neighbors
Output
  $LOF = lof_1, \ldots, lof_n$: estimated LOF scores
function FASTLOF(D, c, θ, k)
  shuffle(D)
  Group $d_1, \ldots, d_n$ in $chunk_1, \ldots, chunk_c$
  $active \leftarrow D$
  while new NN^k found **do**
    for all $d_i \in active$ **do**
      $NN_i^k \leftarrow$ findNN($d_i, chunk_{c_i}$)
      Update NN_x^k for new neighbor x in NN_i^k
      c_i++
    $LRD \leftarrow$ LRD(D, NN^k)
    $LOF \leftarrow$ LOF(D, NN^k)
    $active \leftarrow 0$
    for all $d_i \in D$ **do**
      if $lof_i > \theta$ **then**
        $active \leftarrow d_i$
  return LOF

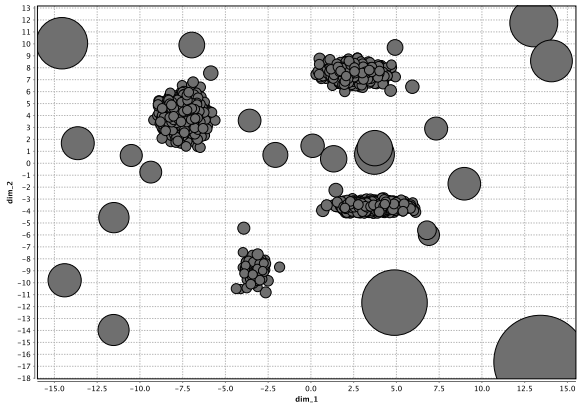

Figure 1. FastLOF results for 2D test data set. The size of the bubbles indicate the LOF score.

4 Experiments and Evaluation

For a first evaluation, a simple 2D data set was created using a mixture of four Gaussians with 3000 instances and 30 uniformly sampled outliers. Figure 1 illustrates the results for FastLOF using $k = 10$, chunk size $c = \sqrt{N} = 56$ and $\theta = 1.1$. A chunk size of $\sqrt{N}$ seems to be a good compromise between overhead (LRD/LOF computations) and distance computation savings, whereas θ should be set to the interval $[1.0, 2.0]$ indicating the minimum outlier suspicion [3]. Compared to LOF, the scores are almost the same, but the number of distance computations dropped by 95% from 4,588,935 to 228,819. To be fair, FastLOF requires additionally 336,000 computations for LRD and LOF in this example. In Figure 2, two points are exemplary chosen to visualize the FastLOF computation over time. For comparison, also the LOF values have been computed after each iteration such that the difference between LOF and FastLOF is clear. It can be observed that only very few chunks are computed using FastLOF for normal points (solid points at the bottom) whereas all chunks are computed for outliers. Note that LOF scores depend on the LRDs of the neighbors such that the LOF and FastLOF scores can differ. When using real world data with higher dimensions, the computational benefit even increases since the LRD and LOF computations do not depend on d. FastLOF was

Dataset	k	θ	LOF AUC	FastLOF AUC	FastLOF Calcs	Best Alg.	Best AUC	Worst AUC
Breast Cancer Wisconsin	10	1.10	0.9916	0.9882	18,5%	INFLO	0.9922	0.8389
Pen-based 4-anomaly (local)	10	1.01	0.9878	0.9937	16.0%	**FastLOF**	0.9937	0.7010
Pen-based 8-normal (global)	40	1.00	0.8864	0.9050	35.5%	u-CBLOF	0.9923	0.6808

Table 1. Performance and run time comparison of FastLOF.

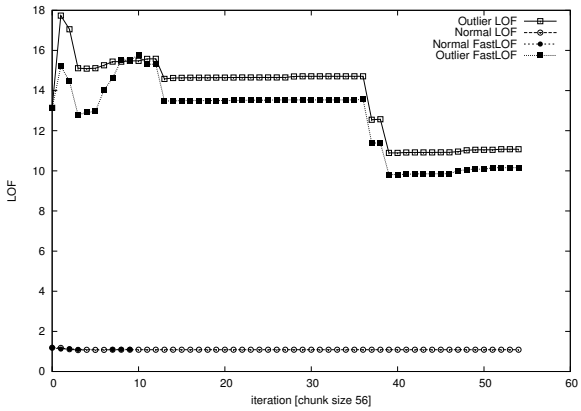

Figure 2. Comparing LOF and FastLOF. For normal points, FastLOF only computes a fraction of the nearest neighbors (solid circles). For outliers, the LOF is approximated well (top). In fact, LOF is only computed once (chunk 56), but for direct comparison it is shown here over time.

also evaluated on real world data from the UCI machine learning repository following the preprocessing in [1, 10]. A detailed description of the data sets and the calculation of the area under curve (AUC) performance measure can be found in these publications. The results of FastLOF in comparison with 8 other algorithms [1] are summarized in Table 1. For our experiments we used the optimal k as found in [1]. It can be seen that FastLOF requires 65-81% less distance calculations while having similar performance compared to LOF. On the pen-based dataset with the local anomaly problem, FastLOF even outperforms all other algorithms.

5 Conclusion

In this paper, FastLOF, an expectation-maximization algorithm for estimating a local outlier factor has been proposed. Similar detection performance is achieved with significantly less computational effort. Furthermore, intermediate LOF results can be obtained already during computation for systems where detection speed is more important than final precision.

Acknowledgment

This work is part of ADEWaS, a project of Deutsche Telekom Laboratories supported by German Research Center for Artificial Intelligence (DFKI) GmbH.

References

[1] M. Amer. Comparison of unsupervised anomaly detection techniques. Bachelor's Thesis, 2011. http://www.madm.eu/_media/theses/thesis-amer.pdf.

[2] F. Angiulli and C. Pizzuti. Fast outlier detection in high dimensional spaces. volume 2431 LNCS, pages 43–78.

[3] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, 2000.

[4] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):1–58.

[5] A. L. M. Chiu and A. W. Fu. Enhancements on local outlier detection. pages 298+, 2003.

[6] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. pages 518–529, 1997.

[7] Z. He, X. Xu, and S. Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10):1641 – 1650, 2003.

[8] W. Jin, A. Tung, J. Han, and W. Wang. Ranking outliers using symmetric neighborhood relationship. In *Advances in KDD*, volume 3918, pages 577–593. 2006.

[9] S. Kim, N. W. Cho, B. Kang, and S.-H. Kang. Fast outlier detection for very large log data. *Expert Syst. Appl.*, 38(8):9587–9596, Aug. 2011.

[10] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. Loop: local outlier probabilities. In *Proc. of the 18th ACM CIKM*, pages 1649–1652, 2009.

[11] M. Moya and D. R. Hush. Network constraints and multi-objective optimization for one-class classification. *Neural Networks*, 9(3):463–474, 1996.

[12] S. Papadimitriou, H. Kitagawa, and et al. Loci: Fast outlier detection using the local correlation integral. *Int. Conf. on Data Engineering*, 0:315, 2003.

[13] L. Portnoy, E. Eskin, and S. Stolfo. Intrusion detection with unlabeled data using clustering. In *Proc. of ACM CSS Workshop on DMSA*, pages 5–8, 2001.

[14] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. SIGMOD '00, pages "427–438".

[15] M. Reif, M. Goldstein, A. Stahl, and T. Breuel. Anomaly detection by combining decision trees and parametric densities. In *ICPR 2008*. IEEE.

[16] J. Tang, Z. Chen, A. Fu, and D. Cheung. Enhancing effectiveness of outlier detections for low density patterns. In *Adv. in KDDM 2002*, volume 2336, pages 535–548.