Learning from Others: User Anomaly Detection Using Anomalous Samples from Other Users

Youngja Park[1](✉), Ian M. Molloy[1], Suresh N. Chari[1], Zenglin Xu[2], Chris Gates[2], and Ninghi Li[2]

[1] IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA
{young_park,molloyim,schari}@us.ibm.com
[2] Purdue University, West Lafayette, IN, USA
{xu218,gates,ninghui}@cs.purdue.edu

Abstract. Machine learning is increasingly used as a key technique in solving many security problems such as botnet detection, transactional fraud, insider threat, etc. One of the key challenges to the widespread application of ML in security is the lack of labeled samples from real applications. For known or common attacks, labeled samples are available, and, therefore, supervised techniques such as multi-class classification can be used. However, in many security applications, it is difficult to obtain labeled samples as each attack can be unique. In order to detect novel, unseen attacks, researchers used unsupervised outlier detection or one-class classification approaches, where they treat existing samples as benign samples. These methods, however, yield high false positive rates, preventing their adoption in real applications.

This paper presents a local outlier factor (LOF)-based method to automatically generate both benign and malicious training samples from unlabeled data. Our method is designed for applications with multiple users such as insider threat, fraud detection, and social network analysis. For each target user, we compute LOF scores of all samples with respect to the target user's samples. This allows us to identify (1) other users' samples that lie in the boundary regions and (2) outliers from the target user's samples that can distort the decision boundary. We use the samples from other users as malicious samples, and use the target user's samples as benign samples after removing the outliers.

We validate the effectiveness of our method using several datasets including access logs for valuable corporate resources, DBLP paper titles, and behavioral biometrics of user typing behavior. The evaluation of our method on these datasets confirms that, in almost all cases, our technique performs significantly better than both one-class classification methods and prior two-class classification methods. Further, our method is a general technique that can be used for many security applications.

1 Introduction

Driven by an almost endless stream of well publicized cases of information theft by malicious insiders, such as *Wikileaks* and *Snowden*, there is increased interest

© Springer International Publishing Switzerland 2015
G. Pernul et al. (Eds.): ESORICS 2015, Part II, LNCS 9327, pp. 396–414, 2015.
DOI: 10.1007/978-3-319-24177-7_20

for monitoring systems to detect anomalous user behavior. Today, in addition to traditional access control and other security controls, organizations actively deploy activity monitoring mechanisms to detect such attacks. Activity monitoring is done through enforced rules as well as anomaly detection using ML techniques. To best apply ML techniques, it is ideal if we can train a model with lots of both anomalous and benign samples. This is very difficult for security applications: it is often unrealistic to expect to gather enough anomalous samples for labeling. The lack of anomalous samples prohibits the applicability of more accurate classification techniques, and, thus, most monitoring applications have adopted anomaly detection or one-class classification techniques. These methods construct a profile of a subject's normal behavior using the subject's past behavior by treating them as benign sample and compare a new observed behavior with the normal profile resulting in high false positive cases. The lack of labeled data can also extend to samples of normal activity. In some situations, there may be a small number of samples to learn a user's normal behavior, or the sample contain anomalous cases. This makes it difficult to learn an accurate model for the data. Another problem of existing approaches is that they treat the samples in the training period as benign. However, the training data can contain anomalies, and, thus, training with this data can result in high false negative rates.

Prior work has addressed the issue of mapping such one class classification problems into two class classification problems [1–5]. However, earlier approaches generate samples for the second class randomly [1,2] or by following a predefined distribution such as uniform or Gaussian distribution [3–5]. While these data points are generated from the data, they do not represent actual behavior in most real-world problems.

In contrast, we observe that multiple users share the system and exhibit distinct behavioral patterns in many monitoring applications. Examples of such scenarios include user authentication determining the authenticity of a user based on users' keystroke patterns, insider threat detection identifying deviation of a user's access patterns from past behavior, and social network analysis detecting anomaly in a user's collaboration patterns. In these scenarios, we expect other users' behavioral pattern to be distinct from the target user's behavior. Thus, we can utilize other users' samples to estimate a target user's possible abnormal behavioral patterns. We leverage these "abnormal" samples to help the classifier learn a boundary between a user's expected behavior and unexpected behavior. There are no assumptions made about the distribution of anomalous samples, no manual labeling is necessary, and it is independent of the underlying learning algorithms.

The basic idea of our algorithm lies in the concept of a local density and is built on the Local Outlier Factor algorithm [6]. LOF is a density-based anomaly detection algorithm and finds anomalous data points based on their deviation with respect to their neighbours. The locality of a data point is defined by its k-nearest neighbors, and the distance to the k-neighbors is used to estimate the density. If the density of a data point has much lower density than its neighbors, the data point is considered as an outlier.

We extend the idea of LOF and propose a new local density-based method for selecting a good set of anomalous samples from the other users' sample set. For a given target user, we compute the Local Outlier Factor (LOF) value for all data points with respect to the target user's data points and choose data points from other users' samples that are distant from the target user's data as anomalous samples. Our method, named as *reference points-based LOF*, gives us an estimate of the degree of "outlier-ness" of the other data points with respect to the target user's behavior. Given this measure of LOF, we have explored two broad strategies to select abnormal samples: use the points with the highest LOF, which deviate the most from the target user's data points, or use the points with the lowest LOF above a certain threshold, which are just "slightly different" from the target user's data. Further, we use the reference points-based LOF to remove outliers from the target user's sample set and produce more coherent benign sample set.

We evaluate these approaches using four different data sets: keystroke dynamics data for user authentication, typing patterns for user recognition, user access patterns for a source code repository, and, finally, paper titles from the DBLP bibliography. For each test user, we generate both benign data points and abnormal data points using the LOF-based strategies. We then train two-class classifiers—Decision Tree, Logistic Regression and Random Forest—for evaluation. In each case, our evaluation has shown that the strategy of providing abnormal data points for users using the *Reference Points-based LOF* provides uniformly better results compared to the one class classifier approach and binary-class approach using synthetically constructed distributions of abnormal samples for training. Our methods produce better AUC (Area Under the ROC Curve) across all users in the various data sets. Our technique is promising and applicable to a large number of problems in security relying on anomaly detection and user profiling.

2 Approach

This paper addresses the critical problem of detecting anomalous user behavior, targeting use cases such as continuous user authentication and insider threat detection. The key challenge we aim to address is the difficulty in obtaining labeled anomalous samples. The primary goal is to detect anomalous behavior of a user, i.e., when a user's behavior deviates significantly from his/her own historical behavior. While user-specific modeling can provide higher accuracy and adaptability to changing environments, obtaining known anomalous samples for each individual user is made even more challenging.

Standard anomaly detection techniques, such as statistical analysis or one-class classification, aim to rank new samples based on their similarity to the model of the negative samples, assuming that all previously known samples are negative (benign). Many approaches use distance or density of the points as a measurement for the similarity, in which data points with the lowest density or the longest average distance to the previously known (negative) samples are considered most anomalous.

In contrast, our approach makes no assumption on the underlying data distribution. We assume that data samples in these applications are generated independently by many users with different underlying distributions. Consider, for example, the case of detecting anomalous user access to a source code repository shared by many employees. In this case, we expect that users' access patterns will depend on their role in the organization or project and will, in general, be different from each other. For instance, we expect software developers to exhibit similar access patterns e.g. access the repository regularly during business hours, and to be significantly different from the access patterns of testers, business managers, backup administrators etc. Further, we assume that, in these multi-user applications, malicious actors often change their behaviors subtly or try to impersonate another person to hide their malicious intention. Thus, an anomalous point of a user's behavior can look perfectly normal in the global view, and, anomaly detection per user can detect these stealth attacks better than a global anomaly detection. However, while user-specific modeling can produce more accurate detection, the data sparseness problem becomes even worse. In this case, in addition to the lack of anomalous cases, we may not have enough benign cases for some users, such as new users or not active users.

We address the lack of labeled samples by exploiting data samples from the other users in the target application. Our intuition is that, when there are many users, other users' behavior can provide additional insights on potential anomalies. We assume that a user's actions are similar each other and tend to form a few clusters occupying a small area in the data space. However, when we combine data samples from many users, they provide more accurate projection of the entire data space and help to estimate accurate boundaries between different users.

The main focus of this study is on how to generate anomalous samples automatically from other users' behaviors. To identify possibly anomalous samples for a target user, we adopt a common definition of anomaly which considers the data points in low density areas as anomalous [7]. We examine all the samples of all users in the data set and identify the samples that are considered different from a target user's data samples. We apply Local Outlier Factor (LOF) [6] to estimate the degree of "outlier-ness" and select anomalous samples for the target user from other users' data samples which have high LOF with respect to the target user's data samples. We call the target user's data points the *reference points*, and call the LOF computed based on the reference points as the *reference points-based LOF*. Figure 1 illustrates the difference between outliers based on the standard LOF and outliers based on the *reference points-based LOF*. Standard anomaly detection methods will identify two clusters of dense area and detect only the two data points p_1 and p_2 as outliers as shown in Fig. 1(a). However, the reference points-based outlier detection method will measure the density of all the points with respect to their distance to the reference points (C_1), and thus will consider all the data points in C_2 as outliers as well. The main differences of our approach from other density-based anomaly detection methods are in that we measure the outlier-ness of a data point with respect to

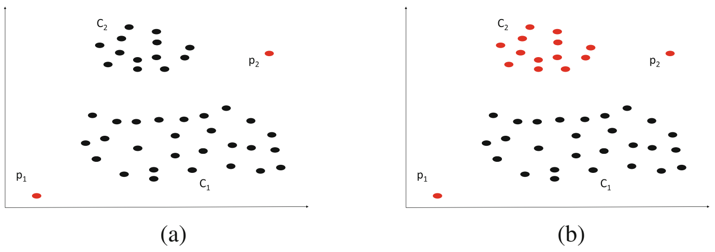


Fig. 1. Comparison of standard outlier detection (a) and reference points-based outlier detection (b). The data points in C_1 are the reference points, and the red points represent outliers (Color figure online).

a fixed set of existing data points in the space, and, we use low density samples as anomalous samples to build a binary classifier.

Apparently, if the data points of a user (i.e., reference points) are dispersed over a wide area in the data space and mingled with other users' samples, this method would not work well. To validate our assumption that one user's actions tend to form close clusters, we analyzed a data set of 51 distinct users containing 200 cases for each user (10,200 cases in total) from the dynamic keystroke analysis study [8]. We considered the 200 instances of the first user as the reference points, and computed the LOF scores for all 10,200 samples with respect to the 200 reference points. Figure 2 shows the LOF scores of the data points of all the users. The x-axis represents the 51 users, and each cross point in the chart represents a keystroke instance.

As we can see, all samples belonging to the first user have very low LOF scores, while other users' data points have much higher LOF scores, indicating that the data points belonging to a user are close each other, while data points from other users are separated. The analysis result supports our hypothesis on exploiting other users' data points to generate anomalous samples for the target user.

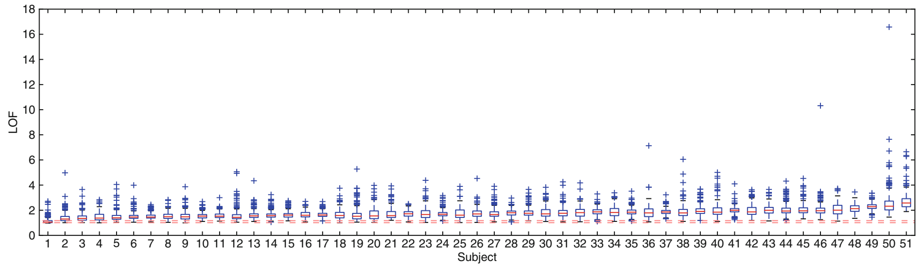


Fig. 2. The reference points-based LOF scores for the 51 users in the Keystroke Data Set [8] using the data points of the first user as the reference points. The dashed red line is the 95 % confidence interval for the target user (Color figure online).

3 Reference Points-Based LOF

In this section, we explain the reference points-based LOF method in detail. The task is to build an anomaly detection model for each user with both normal and anomalous samples. In the absence of labeled anomalous samples, we explore other users' samples as potential anomalous points for a target user. We find possible anomalous samples for each user from the other users' normal samples. The basic idea is to measure the degree of "outlier-ness" of all the training samples and identify the data points that deviate from the target user's samples.

In density-based anomaly detection, a data point is considered as an outlier if the local density of the point is substantially lower than its neighbors. In this work, we use the Local Outlier Factor (LOF) for local density estimation [6], where the local area is determined by its k nearest neighbors from the target user. LOF is computed as defined in Eqs. 1 and 2.

$$LOF(p) = \frac{\sum_{q \in kNN(p)} \frac{LRD(q)}{LRD(p)}}{|kNN(p)|} \tag{1}$$

The local reachability distance (LRD) is defined as in Eq. 2.

$$LRD(p) = \frac{|kNN(p)|}{\sum_{q \in kNN(p)} \max\{k\text{-}distance(q), dist(p,q)\}} \tag{2}$$

where $k\text{-}distance(q)$ be the distance of the point q to its k-th nearest neighbor.

Let $\mathcal{U}$ be the set of users, $\mathcal{D}$ be the set of training samples for all the users, $\mathcal{D}_u$ be the samples of a target user u, and $\overline{\mathcal{D}_u}$ be the samples from all other users except u, i.e., $\mathcal{D} = \mathcal{D}_u \cup \overline{\mathcal{D}_u}$. Unlike the standard LOF, where k-nearest neighbors are found from the entire data set, we compute the LOF values of all data points $p \in \mathcal{D}$ based on their distance to the k-nearest neighbors from the target user's data points, $\mathcal{D}_u$. Figure 3 shows a high level sketch of the reference points-based LOF.

```
Reference Points based LOF($\mathcal{D}, U, K$)
$\mathcal{D}_u$ = extract samples of user $u$ from $\mathcal{D}$
for ($p \in \mathcal{D}$) do
    for ($q \in \mathcal{D}_u$) do
        compute the distance between $p$ and $q$, $dist(p,q)$
    end for
end for
for ($p \in \overline{\mathcal{D}_u}$) do
    $kNN(p)$ = $K$ nearest neighbors of p in $\mathcal{D}_u$
    compute $LOF(p)$ using Equation 1
end for
```

Fig. 3. Algorithm for computing LOF based on a set of reference points

In this work, we compute the distance between two data points p and q using a normalized Manhattan distance.

$$dist(p, q) = \sum_i \frac{|p_i - q_i|}{max(i) - min(i)} \tag{3}$$

where $max(i)$ and $min(i)$ denote the maximum and minimum value for the i-th features respectively. Alternative to the k-nearest neighbors, one can use the ϵ-neighborhood as described in the DBSCAN clustering algorithm [9]. In this case, the degree of outlier-ness of a sample p can be computed as the average distance to the data points in its directly reachable neighbors.

4 Abnormal Behavior Detection

In this section, we explore several strategies to generate a labeled training set based on the reference points-based LOF. Sections 4.1 and 4.2 describe strategies for choosing normal samples and anomalous samples respectively. Note that the algorithm in Fig. 3 computes the LOF scores for all data points including both the target user's data points and other users' data points. We use the LOF scores to select both normal and abnormal samples to train a two-class classification model for each user.

4.1 Normal Sample Selection

We apply the following 2 different strategies for generating normal samples for training.

1. All Self Samples (Self): This method uses all the samples from the target user during the training period as normal samples, similarly to unsupervised anomaly detection or one-class classification approach.
2. No Outlier Samples (LowLOF): Note that we compute LOF values for the target user's own samples as well. The data points with relatively high LOF scores are outliers in the target user's samples. We discard these outlier samples from the target user's sample set and use the remaining samples as normal samples for training. This strategy can handle noisy data.

4.2 Abnormal Sample Selection

For anomalous training sample generation, we apply the following four strategies to extract anomalous samples for the target user. These strategies aim to find other users' samples that are outside of the target user's samples, i.e., outliers from the perspective of the target user.

1. Boundary Sampling (LowLOFAll): Out of all other users' samples that have LOF higher than a threshold, we choose the samples with lowest LOF scores. This method finds anomalous samples that are located close to the boundaries. These samples would have higher LOF scores than most of the target user's samples, but have lower LOF scores than most of the other users' samples.

2. Boundary Sampling Per User (LowLOFUser): This method is also intended to choose boundary samples. However, this method selects low LOF samples from each of the other users. If we want to generate N anomalous samples, and there are m other users, we generate approximately $\frac{N}{m}$ samples from each user.
3. Outlier Sampling (HighLOFAll): This method generates anomalous samples which deviate mostly from the target users' samples, i.e., samples with highest LOF scores from the sample set from all the other users as in LowLOFAll.
4. Outlier Sampling per User (HighLOFUser): This method is similar to LowLOFUser. The difference is that it chooses samples with highest LOF scores from each of the other users.

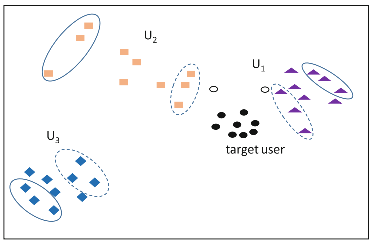
We note that our algorithm chooses anomalous samples which have an LOF score higher than a threshold to exclude other users' samples that are inside of or too close to the target user's region. Further, the LowLOF method for generating normal samples (Sect. 4.1) can also discard a few normal samples. Thus, for very small data sets like the Typist data set, we can generate a smaller number of samples than requested.

4.3 Training Sample Generation

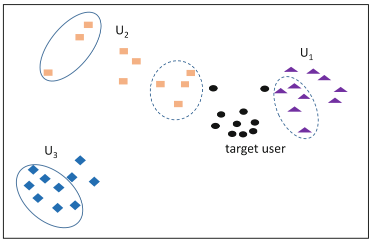
By combining the two methods for normal sample generation and the four for abnormal samples, we have eight methods for generating training samples. We label the methods as 'Normal Sampling Method'-'Abnormal Sampling Method' (e.g., Self-LowLOFAll and LowLOF-HighLOWUser). Figure 4 illustrates the differences of the sampling methods. Here, the circle points are the data samples of the target user, and the triangle, square and diamond points belong to the other three users, U_1, U_2, and U_3 respectively. Figure 4(a) shows the LowLOF method and Fig. 4(b) shows the Self method for selecting normal samples respectively. With Self, all circle points are selected as normal, while the two white circles are discarded because their LOF scores are high and considered as outlier with the LowLOF method. Suppose we plan to include nine anomalous samples in the training data set: Fig. 4(a) shows per-user basis sampling methods, LowLOFUser and HighLOFUser, and chooses three samples from each user. The points enclosed by dashed lines are selected by LowLOFUser, while the points enclosed by solid lines are chosen by HighLOFUser. Figure 4(b) shows anomalous samples selected by LowLOFAll (dashed line) and HighLOFAll (solid line).

4.4 Binary Classification

Having both normal and anomalous samples in the training data allows us to cast the anomaly detection task as a two-class classification problem, and learn a classifier that can discriminate the abnormal samples from the normal samples. Any classification algorithm can be applied and may be chosen based on the application. We use classification algorithms that produce the class probability as an output, rather than a binary decision. The advantage of having class



(a) LowLOF–LOFUser



(b) Self–LOFAll

Fig. 4. Illustrations of the reference points-based LOF sampling methods. The circles are the instances of the target user. (a) depicts LowLOF-LowLOFUser and LowLOF-HighLOFUser methods, and (b) depicts Self-LowLOFAll and Self-HighLOFAll methods. The points enclosed with dashed lines have low LOF values, and those enclosed with solid lines have high LOF values.

probability estimation over a binary decision of *normal* versus *abnormal* is that the system administrators can adjust the ratio of alarms according to available resources and costs. In this work, we conduct experiments with three classification algorithms: Decision Tree, Logistic Regression, and Random Forest. We use the implementations in RapidMiner [10] for all the experiments described in Sect. 5.

5 Experiments

We validated the proposed sampling methods with three publicly available data sets and one private data set from information security application. This section decribes the four data sets and the evaluation methods in details.

5.1 Data

We validate our algorithms for anomalous sample generation using the following four data sets: *Keystroke Dynamics Benchmark Data*; *Typist Recognition Data*; *DBLP Collaboration Network Data*; and *Access Log Data*. The first three data sets are publicly available and the last data set is private.

Keystroke Dynamics Benchmark Data: Killourhy and Maxion [8] collected keystroke data from 51 users typing the same strong password 400 times, broken into eight equal-length sessions[1]. They collected various timing features such as the length of time between each keystroke, and the time each key was depressed. They used this data set to compare the accuracy of fourteen one-class classifiers at identifying impostors.

Typist Recognition Data: Hempstalk, Frank and Witten [5] collected data on the typing patterns of ten different users and build a classifier to identify

[1] http://www.cs.cmu.edu/afs/cs/Web/People/keystroke/.

individual typists. The typing pattern are represented by eight features such as typing speed and error rate (backspace)[2]. The typing behavior of the users is broken into units, approximately one paragraph's worth of typing. Each user contains between 24 and 75 records with an average of 53.3.

DBLP Collaboration Network Data: DBLP[3] is a large database of publications from computer science journals, conferences, and workshops. We extracted DBLP records of "inproceedings" and "incollection" publications, and authors with publication records between 25 and 150 papers in the data set. We then randomly selected 200 authors from the extracted data and generated a corpus containing the publication records of the selected 200 authors. The paper titles are preprocessed by removing stop words and performing lemmatization on the remaining words, and each publication record is converted to a vector of term:count pairs found in the title. We build models to learn what a "normal" paper title is for an author.

Access Log Data: The access log data set comes from a source code repository used in a large IT company. The logs were collected over 5 years and consist of 2,623 unique users, 298,365 unique directories, 1,162,259 unique files, and 68,736,222 total accesses. Each log contains a timestamp, a user ID, a resource name, and the action performed on the resource. We process these logs down to individual periods per user which represent the user's behavior in a given week. The features include the number of total accesses, the number of unique accesses in that period, new unique accesses given a user's history, counts for the actions performed, counts for the file extensions accessed, and similarity scores to the target user as discussed in [11]. The similarity scores represent how similar a user is to the other users given the user's current access pattern and the other users' past access patterns.

5.2 Evaluation Method

While we assume that most of a target user's activity is benign, we need to prevent our training data from containing samples of malicious behavior to be detected. For example, if the target user's account is compromised by an adversary, the classifier should not have been trained on the activity of the adversary. For this reason, we train and test a classifier on different user groups. For each target user, we perform K-fold cross validation by dividing the user population into K disjoint sets of training and testing user groups. For example, suppose there are three users U_1, U_2 and U_3, and U_1 is the target user. We train a classifier on U_1 and U_2 and test on U_1 and U_3, and train a second classifier on U_1 and U_3, and test on U_1 and U_2. The user actions are also split into training and testing samples using a pivot point in time when applicable, that is, all training actions occur strictly prior to all testing actions. We choose anomalous samples only from the training user group and measure the performance on the evaluation user group. The training user group and the evaluation user group for each

[2] http://www.cs.waikato.ac.nz/ml/data/typist.arff.
[3] http://snap.stanford.edu/data/com-DBLP.html.

Table 1. Sizes of the four experiment data sets both for training and testing. The number of samples denote the mean values for each cross-validation set.

Data Set	Num. Users	Training		Testing	
		Normal	Abnormal	Normal	Abnormal
Keystroke	51	200	200	200	54
Typist	10	47	45	5	45
DBLP	28	91	85	23	34
Access Log	202	125	151	37	201

fold are mutually exclusive, so no evaluation user is seen during training. Table 1 shows the average size of the training and test data sets.

To ease comparison with some prior work, we evaluate the performance of a two-class classifier versus a one-class classifier for detecting changes in user behavior. Further, for all experiments, we report the average results over the cross-validation splits and compare the algorithms based on AUC (Area Under Curve), as it is the metric used in all previous work.

5.3 Baseline Methods

We use two baseline methods for evaluation. The first baseline method is a two-class classification using randomly selected anomalous samples from other users. We use all training samples from the target user as normal samples (i.e., Self), and apply a standard random sampling strategy (Random) on the other users' sample set. We call this baseline Self-Random, and compare this method with the eight different combinations of sampling methods described in the previous section.

We also compare two-class classification with one-class classification based on one-class SVM using all samples of a target user as normal samples. We use the SVM implementation from SVM KMToolbox [12][4] with RBF kernel and the upper bound on the number of errors ν was set to 0.1.

6 Results

6.1 Keystroke Dynamics Benchmark Data

Killourhy and Maxion [8] collected the Keystroke data to build one-class classifiers to detect imposters. There are 51 users in the data, where each user contributed 400 samples. They evaluated fourteen scoring methods, one of which is a one-class SVM. Each method is trained using normal samples obtained only from the target user and is not exposed to malicious samples during training. However, our methods need anomalous samples from other users. To make the

[4] Download available at http://asi.insa-rouen.fr/enseignants/~arakoto/toolbox/.

Table 2. AUC comparison for the Keystroke Dynamics Data. Our Algorithm (LOF) reports the best result obtained from the LOF-based strategies.

Cross validation	LOF	Self-Random	One-Class SVM
1	**0.979**	0.934	0.927
2	**0.967**	0.939	0.908
3	**0.971**	0.905	0.905
4	**0.945**	0.933	0.800
5	**0.968**	0.957	0.844
Avg.	**0.966**	0.934	0.877

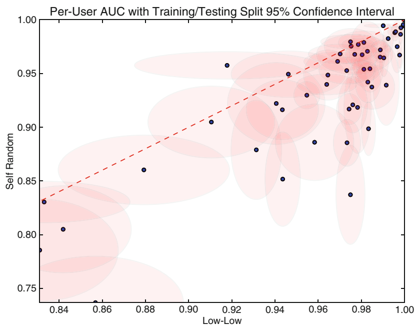


Fig. 5. AUC values of the classifiers for individual target users, and the variance across the five training-testing folds.

comparison objective, we divided the 51 users into 5-fold cross validation sets comprising a training user group and an evaluation user group as described in Sect. 5.2. Each training and testing group contains approximately 41 users and 10 users respectively.

Following Killourhy et al.'s convention, we considered one user from a training user group as the target user, and treat all remaining users in the training user group as malicious users. For each target user, we use the user's first 200 samples to select the benign training samples and extract five samples randomly from each malicious user as anomalous samples for training. Therefore, the training data set contains 200 benign cases and 200 anomalous cases. For testing, we use the remaining 200 samples from the target user and extract five samples randomly from each user in the testing user group as anomalous samples. The performance is measured using the average AUC over all 51 users. Table 2 shows the average AUC of the 5-fold cross validation results. As we can see from the table, the LOF-based method produces a higher AUC than the Self-Random baseline method and the one-class SVM for all folds.

Next, we evaluate how well the individual classifiers compare across each target user. Figure 5 compares the classifiers for individual target users and the error rates of the AUC across the five splits. Here, the x-axis is the AUC score of the LowLOF-LowLOFUser method, and the y-axis is the AUC for the random baseline method, Self-Random. The red error ellipse around each point has a diameter of one standard deviation for the AUC scores over the five splits. Points below the *y-equal-x* line (red-dashed) are classifiers where our LOF method produced strictly better results.

6.2 Typist Recognition Data

Hempstalk et al. [5] proposed a technique for combining density and class probability estimation for continuous typist recognition. They collected a dataset of

15 emails from each of ten participants to validate their method. To compare our methods with theirs, we conducted experiments using a stratified 10-fold cross validation on the same data set as described in [5]. For each user, we choose 90 % of randomly chosen samples as training samples and the remaining 10 % for benign testing samples. Due to the small user population, we don't split the users into disjoint training and evaluation groups.

To generate anomalous training samples for each target user, we first merge the training samples for all users, assuming the target user's samples as "normal" and samples from the other nine users as "abnormal". We then compute the reference points-based LOF scores all the samples in the training data, and generate abnormal samples as described in Sect. 3. To replicate the experiments by Hempstalk et al. as close as we can, we set the number of anomalous samples to the number of normal samples for the target users. However, our method often produces a smaller number of samples than requested as we noted in Sect. 4.2, because LowLOF normal sampling method discards outliers from the target user's sample set.

Table 3 compares the results of our algorithm, random sampling-based method, and two density estimation-based methods from [5]. The table shows the average AUC over the 10-fold cross validation for each user. As we can see, our method outperforms both of the density estimation methods, and is slightly better than the random sampling method for this data set. However, as we noted

Table 3. AUC results for the Typist Recognition Data. Our Algorithm (LOF) reports the best result obtained from the LOF-based strategies. The results of Gaussian and EM methods are obtained by the density and class probability estimation described in [5].

Participant	LOF	*Self-Random*	*Density**		One-Class SVM
			Gaussian	*EM*	
A	**0.946**	0.923	0.924	0.923	0.894
B	0.965	**0.984**	0.934	0.929	0.725
C	**0.852**	0.825	0.707	0.786	0.769
D	0.903	0.922	**0.924**	0.902	0.918
E	0.977	**0.982**	0.973	0.971	0.932
F	**0.902**	0.872	0.852	0.867	0.749
G	**0.960**	0.949	0.942	0.952	0.856
H	0.909	0.877	0.909	**0.914**	0.822
I	**0.976**	0.974	0.956	0.950	0.913
J	**1.000**	0.989	**1.000**	**1.000**	0.982
Avg.	**0.939**	0.930	0.912	0.919	0.856
Std.	**0.046**	0.056	0.082	0.060	0.087

Table 4. AUC results for the DBLP data set.

Authors	LOF	*Self-Random*	One-Class SVM
1	**0.783**	0.664	0.59
2	0.810	**0.859**	0.74
3	**0.926**	0.909	0.72
4	**0.805**	0.718	0.78
5	**0.837**	0.765	0.71
6	0.760	**0.821**	0.48
7	0.694	**0.760**	0.33
8	**0.845**	0.825	0.79
9	**0.918**	0.896	0.75
10	0.802	0.720	**0.84**
11	**0.807**	0.779	0.59
12	**0.810**	0.730	0.74
13	0.547	**0.630**	0.35
14	0.835	**0.868**	0.76
15	0.658	**0.822**	0.47
16	0.846	**0.852**	0.54
17	**0.750**	0.696	0.71
18	**0.760**	0.756	0.62
19	0.734	**0.824**	0.46
20	**0.659**	0.625	0.58
21	**0.890**	0.883	0.75
22	0.603	**0.607**	0.48
23	**0.814**	0.642	0.67
24	**0.735**	0.676	0.72
25	0.849	**0.865**	0.63
26	0.779	**0.858**	0.49
27	0.685	**0.686**	0.61
28	**0.628**	0.552	0.51
Avg.	**0.770**	0.760	0.621
Std.	**0.093**	0.100	0.137

earlier, the results demonstrate our algorithm's advantage, as it used a smaller number of training samples than the other three methods in most testing cases.

6.3 DBLP Collaboration Network Data

While the DBLP data contain publication information about many authors, each user has a small number of publications. Many authors do not have enough data points for training a reliable model. In this experiments, we selected authors with at least 50 publications in the data set, resulting in 28 distinct authors. We used the words in the publication titles as the features and represent each user with a bag of word vector after removing stop words and converting words to their base form, resulting in 9,670 unique features. We ran PCA (Principal Component Analysis) on the entire samples, and reduced the dimension to 200. Then, we conducted a 5-fold cross validation test similar to the password data sets. Table 4 summarizes the average AUC of each user over the 5-fold splits.

6.4 Access Log Data

The access log data contains a mixture of real user IDs and system IDs which periodically run batch processes. These system processes perform tasks such as nightly builds of source code and exhibit vastly different behavioral patterns from real users. We discarded these system IDs from the logs. Further, we eliminated any user that is active in fewer than 150 weeks (not active users) or more than 250 weeks (very active) out of 260 weeks during which the logs were collected. The final data set contain 202 unique users. The samples are then split by time into 80 % for training and 20 % for testing.

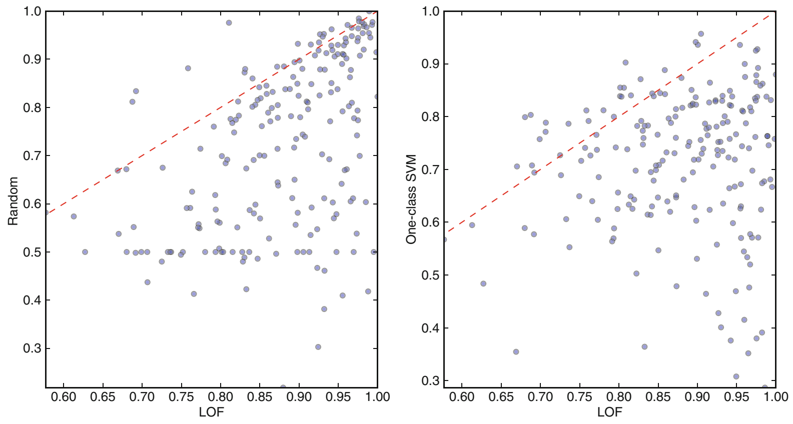


Fig. 6. Comparison of AUC scores for LOF, Random, and One-Class SVM across the 202 per-user classifiers.

Figure 6 shows the comparison of the three sampling methods in terms of the AUC scores for the individual classifiers per target user, and Table 5 shows the mean and standard deviation of the AUC scores of the 202 users.

In both scatter plots, the AUC scores of LOF are given on the x-axis, and the AUC scores for the competing methods, Self-Random for the left plot and one-class SVM for the right, are given on the y-axis. The dashed red-line is the *y-equals-x* line. For any points below the line, our LOF method outperformed the other methods for the given target user, producing better results across the majority of target users (89.1 % compared to Self-Random and 87.1 % compared to one-class SVM). Further, the improvement in AUC scores is significant: 0.155 (stdev 0.155) higher compared to Self-Random, and 0.163 (stdev 0.155) compared to one-class SVM. We also note that the biggest impediment to the use of analytics for insider threat detection is the high false positive rates. As shown above, our approach significantly reduces false positive rates, thus improving the applicability of anomaly detection mechanisms.

6.5 Comparison of Sampling Methods

Lastly, we compare the performance of the eight LOF-based sampling methods and the Self-Random baseline. The comparison was conducted using the Keystroke Data and the three binary classifiers—Decision Tree, Logistic Regression, and Random Forest—over a 5-fold cross validation, resulting in 6,885 experiments in total (51 *users* × 9 *sampling methods* × 3 *classifiers* × 5 *folds*). We counted how many times a sampling method preformed the best for each of the user and classifier combinations. When multiple sampling algorithms made a tie, we considered all the methods as the best performing one.

The comparison results are shown in Fig. 7. The x-axis denotes the five methods for generating anomalous samples—four LOF-based methods and the

Table 5. Mean and Standard Deviation for AUC scores of the 202 users in the Access Log data.

Method	Mean AUC	Stdev AUC
LOF	**0.877**	0.089
Self-Random	0.722	0.176
One-Class SVM	0.714	0.134

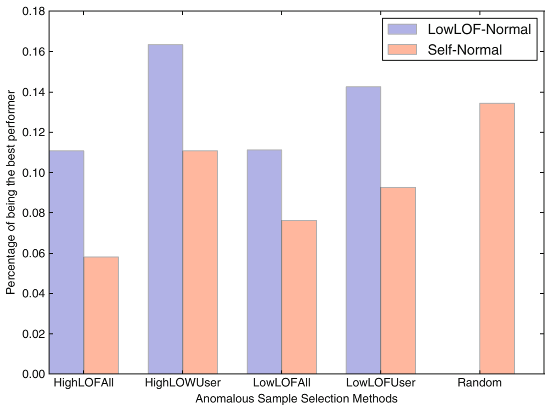


Fig. 7. The percentage of each sampling method being the best performer.

Random method. The blue bars represent the LowLOF method for generating normal samples, and the pink bars indicate that all samples of the target user were used as normal (Self). We can see that LowLOF-HighLOFUser was the best performing method most of the time, closely followed by LowLOF-LowLOFUser. The results confirm two findings. First, discarding outlier cases from the normal sample significantly increase the detection accuracy, as indicated by all blue bars (LowLOF outperforming red bards (Self). Second, sampling from each user is more beneficial than from the merged sample set (LOFUser vs. LOFAll for anomalous samples). The reason that there is no significant difference between HighLOFUser and LowLOFUser methods for anomalous sample generation is because the sample size was relatively small, and many of the selected anomalous samples appear in both training sets.

7 Related Work

Anomaly detection has been an important research problem in security analysis [13, 14]. Various techniques based on domain knowledge/statistics [15–19] or on data mining algorithms [20–22], haven been proposed for anomaly detection for a number of application domains. Due to the absence or insufficiency of labeled examples, most of the techniques have modeled anomaly detection as a unsupervised learning problem. However, unsupervised modeling suffers from a number of problems, e.g., the incapability of discriminative modeling and the difficulty of tuning parameters, leading to high false-positive rates.

To solve this problem, several anomaly detection methods tried to artificially generate samples as a second class based on some heuristics, posing a one-class classification task as a binary classification problem. For example, in a word spotting application, Chang and Lippmann [1] presented a method to artificially enlarge the number of training talkers to increase variabilities of training samples. They transformed one talker's speech pattern to that of a new talker by generating more varied training examples of keywords. Theiler and Cai [2] applied a resampling method to generate a random sample by choosing each of its coordinates randomly from the coordinate values that are in the data. Later, Fan et al. [3] proposed a distribution-based artificial anomaly generation method, which first measures the density of each feature value in the original data set D and then artificially generates anomaly points near to the normal data points by replacing low-density features with a different value in D. This method assumes that the boundary between the known and anomalous instances is very close to the existing data, hence "near misses" can be safely assumed to be anomalous. However, this methods is not applicable to data with a very high dimensionality or with continuous variables.

Further, Hastie et al. [4] summarized techniques for transforming the density estimation (unsupervised learning) problem into one of supervised learning using artificially generated data in the context of association rule learning. A reference model, such as uniform or Gaussian, can be used to generate artificial training samples as "contrast" statistics that provide information concerning departures of the data density from the chosen reference density. Following this principle, Hempstalk et al. [5] further proposed to employ the training data from the target class to generate artificial data based on a known reference distribution. But it restrict the underlying classification algorithm to produce class probability estimates rather than a binary decision.

Despite the success of the above methods, they suffer either from strong restrictions, which made them not applicable to problems with high dimensional data other application domains, or from the requirement of estimating the reference data distribution, which is usually not accurate and may lead to suboptimal performance. Our method addresses both limitations: (1) artificially generated samples do not reflect real cases, (2) assuming an underlying data distribution is unrealistic in multi-user environments. Instead, our LOF-based sampling provides a unified mechanism to filter out bad normal samples and generate potential anomalous samples for each target user.

8 Conclusion

This study focused on abnormal behavior detection for applications where multiple users share the system or application. Many applications exist in computer security such as user authentication, insider threat detection, and network security, and anomalous user activity detection in social network. In each case, we learn a target user's normal behavior from the user's training samples, and estimate the user's possible abnormal behavioral patterns from other users' training samples, who exhibit quite different behavioral patterns from the target user.

We proposed the reference points based LOF method which measures outlierness of a data point with respect to a set of known data points, and showed, through empirical evaluations, that reference points-based LOF methods find good anomalous samples from the behavior of other users. Our evaluation has shown that the our methods provide uniformly better results compared to the one class classifier approach and the approach of providing synthetically constructed distributions of abnormal samples for training. Our technique is promising and seems to be applicable to a large number of problems in security relying on anomaly detection and user profiling.

References

1. Chang, E.I., Lippmann, R.P.: Using voice transformations to create additional training talkers for word spotting. In: Advances in Neural Information Processing Systems, pp. 875–882 (1995)
2. Theiler, J., Cai, D.M.: Resampling approach for anomaly detection in multispectral images. In: Proceedings of the SPIE, pp. 230–240 (2003)
3. Fan, W., Miller, M., Stolfo, S., Lee, W., Chan, P.: Using artificial anomalies to detect unknown and known network intrusions. Knowl. Inf. Syst. **6**(5), 507–527 (2004)
4. Hastie, T., Tibshirani, R., Friedman, J., Franklin, J.: The elements of statistical learning: data mining, inference and prediction. The Math. Intelligencer **27**(2), 83–85 (2005)
5. Hempstalk, K., Frank, E., Witten, I.H.: One-class classification by combining density and class probability estimation. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part I. LNCS (LNAI), vol. 5211, pp. 505–519. Springer, Heidelberg (2008)
6. Breunig, M.M., Kriegel, H.-P., Ng, R.T., Sander, J.: LOF: Identifying density-based local outliers. In: ACM Sigmod Record, vol. 29, no. 2, pp. 93–104. ACM (2000)
7. Bishop, C.M.: Novelty detection and neural network validation. In: IEE Proceedings Vision, Image and Signal Processing, vol. 141, no. 4, pp. 217–222. IET (1994)
8. Killourhy, K.S., Maxion, R.A.: Comparing anomaly-detection algorithms forkeystroke dynamics. In: IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2009, pp. 125–134 (2009)
9. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Kdd, vol. 96, pp. 226–231 (1996)
10. Hofmann, M., Klinkenberg, R.: RapidMiner: Data Mining Use Cases and Business Analytics Applications. CRC Press, Boca Raton (2013)
11. Gates, C., Li, N., Xu, Z., Chari, S.N., Molloy, I., Park, Y.: Detecting insider information theft using features from file access logs. In: Kutyłowski, M., Vaidya, J. (eds.) ICAIS 2014, Part II. LNCS, vol. 8713, pp. 383–400. Springer, Heidelberg (2014)
12. Canu, S., Grandvalet, Y., Guigue, V., Rakotomamonjy, A.: Svm and kernel methods matlab toolbox. Perception Systmes et Information, INSA de Rouen, Rouen, France (2005)
13. Salem, M., Hershkop, S., Stolfo, S.: A survey of insider attack detection research. In: Insider Attack and Cyber Security, pp. 69–90 (2008)
14. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Comput. Surv. (CSUR) **41**(3), 15 (2009)
15. Denning, D.E.: An intrusion-detection model. IEEE Trans. Softw. Eng. **SE–13**(2), 222–232 (1987)

16. Javitz, H.S., Valdes, A.: The SRI IDES statistical anomaly detector. In: Research in Security and Privacy (1991)
17. Apap, F., Honig, A., Hershkop, S., Eskin, E., Stolfo, S.J.: Detecting malicious software by monitoring anomalous windows registry accesses. In: Wespi, A., Vigna, G., Deri, L. (eds.) RAID 2002. LNCS, vol. 2516, pp. 36–53. Springer, Heidelberg (2002)
18. Stolfo, S.J., Hershkop, S., Bui, L.H., Ferster, R., Wang, K.: Anomaly detection in computer security and an application to file system accesses. In: Hacid, M.-S., Murray, N.V., Raś, Z.W., Tsumoto, S. (eds.) ISMIS 2005. LNCS (LNAI), vol. 3488, pp. 14–28. Springer, Heidelberg (2005). http://dx.doi.org/10.1007/11425274_2
19. Chen, Y., Malin, B.: Detection of anomalous insiders in collaborative environments via relational analysis of access logs. In: CODASPY 2011: Proceedings of the First ACM Conference on Data and Application Security and Privacy, Feb 2011
20. Papadimitriou, S., Kitagawa, H., Gibbons, P.B., Faloutsos, C.: Loci: Fast outlier detection using the local correlation integral. In: Dayal, U., Ramamritham, K., Vijayaraman, T.M. (eds.) ICDE, pp. 315–326. IEEE Computer Society (2003)
21. Wang, Y., Parthasarathy, S., Tatikonda, S.: Locality sensitive outlier detection: A ranking driven approach. In: Abiteboul, S., Bhm, K., Koch, C., Tan, K.-L. (eds.) ICDE, pp. 410–421. IEEE Computer Society (2011)
22. Senator, T.E., Goldberg, H.G., Memory, A., Young, W.T., Rees, B., Pierce, R., Huang, D., Reardon, M., Bader, D.A., Chow, E., Essa, I., Jones, J., Bettadapura, V., Chau, D.H., Green, O., Kaya, O., Zakrzewska, A., Briscoe, E., Mappus, R.I.L., McColl, R., Weiss, L., Dietterich, T.G., Fern, A., Wong, W.-K., Das, S., Emmott, A., Irvine, J., Lee, J.-Y., Koutra, D., Faloutsos, C., Corkill, D., Friedland, L., Gentzel, A., Jensen, D.: Detecting insider threats in a real corporate database of computer usage activity. In: KDD 2013: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM Request Permissions, Aug 2013