ISSN 2083 - 2567

JOURNAL OF ARTIFICIAL INTELLIGENCE AND SOFT COMPUTING RESEARCH

Volume No. 15

Issue No. 3

September - December 2025

ENRICHED PUBLICATIONS PVT. LTD

S-9, IInd FLOOR, MLU POCKET,
MANISH ABHINAV PLAZA-II, ABOVE FEDERAL BANK,
PLOT NO-5, SECTOR-5, DWARKA, NEW DELHI, INDIA-110075,
PHONE: - + (91)-(11)-47026006

ISSN 2083 - 2567

JOURNAL OF ARTIFICIAL INTELLIGENCE AND SOFT COMPUTING RESEARCH

ISSN 2083 - 2567

JOURNAL OF ARTIFICIAL INTELLIGENCE AND SOFT COMPUTING RESEARCH

Aims and Scope

Journal of Artificial Intelligence and Soft Computing Research (available also at Sciendo (De Gruyter)) is a dynamically developing international journal focused on the latest scientific results and methods in traditional artificial intelligence, neural networks and soft computing techniques. Our goal is to bring together scientists representing both approaches and various research communities. bring together scientists representing both approaches and various research communities

ISSN 2083 - 2567

JOURNAL OF ARTIFICIAL INTELLIGENCE AND SOFT COMPUTING RESEARCH

ISSN 2083 - 2567

ISSN 2083 - 2567

ISSN 2083 - 2567

JOURNAL OF ARTIFICIAL INTELLIGENCE AND SOFT COMPUTING RESEARCH

(Volume No. 15, Issue No. 3, September- December 2025)

Contents

ISSN 2083 - 2567

A COMPARATIVE STUDY FOR OUTLIER DETECTION METHODS IN HIGH DIMENSIONAL TEXT DATA

Cheong Hee Park
Department of Computer Science and Engineering, Chungnam National University,
220 Gung-dong, Yuseong-gu
Daejeon, 305-763, Korea
*E-mail: cheonghee@cnu.ac.kr

ABSTRACT

Outlier detection aims to find a data sample that is significantly different from other data samples. Various outlier detection methods have been proposed and have been shown to be able to detect anomalies in many practical problems. However, in high dimensional data, conventional outlier detection methods often behave unexpectedly due to a phenomenon called the curse of dimensionality. In this paper, we compare and analyze outlier detection performance in various experimental settings, focusing on text data with dimensions typically in the tens of thousands. Experimental setups were simulated to compare the performance of outlier detection methods in unsupervised versus semisupervised mode and uni-modal versus multi-modal data distributions. The performance of outlier detection methods based on dimension reduction is compared, and a discussion on using k-NN distance in high dimensional data is also provided. Analysis through experimental comparison in various environments can provide insights into the application of outlier detection methods in high dimensional data.

Keywords: Curse of dimensionality, Dimension reduction, High dimensional text data, Outlier detection.

Introduction

An outlier is defined as an observation which deviates so much from other observations enough to arouse suspicions that it was generated by a different mechanism [1]. Outlier detection has been a hot research topic in recent years and has been applied to a variety of problems, such as fraud detection, intrusion detection in computer networks, system fault detection, and unexpected error detection in databases [2, 3, 4, 5, 6].

Outlier detection methods can be classified into three categories according to the learning environment. The first category is a supervised method that detects outliers by learning a binary classifier when training data consisting of normal data and outliers is given. However, there is a high probability of unbalanced learning where the amount of outliers and normal data is significantly different. The second category is unsupervised learning without data labels where it detects data

samples that are highly likely to be outliers on the premise that most of the given data are normal and only a few outliers are included. The third category is semisupervised learning, which detects whether the test data are normal or outliers given the training data consisting only of normal data. In situations where it is easy to collect data under normal conditions,the approach of modeling the data distribution from the normal training data and detecting outliers in the test data based on it may be practical for some application problems.

On the other hand, according to computational methodologies employed in outlier detection methods, they can be roughly categorized to distancebased, density-based, tree-based, clustering-based, and neural network-based methods. A detailed survey of outlier detection methods can be found in several papers including [2, 7, 8]. The performance of the outlier detection method has been shown to be remarkable in many practical problems, but its application to high dimensional data is still difficult. In many cases, the experiments for high dimensional data were performed only for data with hundreds or fewer data dimensions, and not for data with more than tens of thousands of data dimensions such as text data [9, 10, 11].

Due to the curse of dimensionality in high dimensional space, the phenomenon referred to as data sparsity occurs where all pairs of data samples are almost equidistant in high dimensional data space [2]. This can be problematic in outlier detection methods which rely on distance computation such as in clustering or density estimation process. In this paper, we conduct comparative studies for outlier detection methods in high dimensional data. Focusing on text data with dimensions typically in the tens of thousands, the performance of outlier detection methods is compared in various experimental settings:

– Outlier detection in unsupervised mode of multimodal normal data.
– Outlier detection in unsupervised mode of unimodal normal data.
– Outlier detection in semi-supervised mode of multi-modal normal data.
– Outlier detection based on dimension reduction by feature selection or weighted feature combinations.

The remainder of the paper is organized as follows. In Section 2 wereviewoutlier detection methods which have been applied in high dimensional data. In particular, a dimension reduction based outlier detection method is introduced [12]. Dimension reduction is performed by a transformation maximizing kurtosis which can be interpreted as the degree of presence of outliers in the distribution, and in the transformed space outlier detection is applied. In Section 3, using text data, experimental

comparison and analysis are provided under various experimental setting of unsupervised or semi-supervised mode. The discussion follows in Section 4.

2 Outlier Detection methods

In this section, we review outlier detection methods that have shown good performance in various application problems.

2.1 Outlier Detection based on the Distance to k-Nearest Neighbors (KNN)

Distance-based outlier detection is simple and intuitive, but the outlier detection performance is often competitive to more complicated methods. In [13], the average or maximum of the distances to the k nearest neighbors is used as the outlier score of a data sample. The greater the distance to the k nearest neighbors, the more likely it is to be an outlier. Instead of computing outlier scores, a binary decision can be made that a data sample is determined as an outlier when less than k data samples lie within the radius R of the data sample [14].

The main challenge in distance-based methods is the scalability since distances between all pairs of data samples should be computed, and efforts to avoid high computational cost are being made. Partition-based pruning in [13] was used for speedup where data are first partitioned using a clustering algorithm and the partitions that cannot possibly contain the top n outliers are pruned. In [15], a sampling-based outlier detection method was proposed where a small set of samples are taken and an outlier score is measured by the distance from a data sample to its nearest neighbor in the sample set.

In various outlier detection methods, it is often necessary to calculate the distance between data samples during the process such as clustering or density estimation. However, in high dimensional space, the notion of distances may not work as in low dimensional data space. As the data dimension increases, the feature space becomes increasingly sparse and the maximum and the minimum distance between the pairs of data samples become in discernible compared to theminimumdistance[16]. In the experiments using text data of Section 3, the performance of a distance-based outlier detection method is experimentally compared too the rmethods and the impact from the curse of dimensionality on outlier detection is evaluated.

2.2 Angle-based Outlier Detection (ABOD)

Angle-based outlier detection (ABOD) computes an outlier score using the variances of angles

between the difference vectors to pairs of other data samples from a data sample [17].The idea is motivated by the following in tuition: For a data sample with in a cluster, the angles between difference vectors to pairs of other points differ widely, but for outliers, the angles to the most pairs of data samples will be small since most data samples are clustered in some directions[17].The angle-based out lier factor *ABOF (x)* for a data sample *x* is computed as

$$ABOF(x) = variance_{z_1,z_2 \in D}\left(\frac{< \overline{xz_1}, \overline{xz_2} >}{\|\overline{xz_1}\|^2 \|\overline{xz_2}\|^2}\right), \quad (1)$$

where D is a given data set, $<\cdot,\cdot>$ denotes the scalar product, and $\overline{xz_i}$ is the difference vector $z_i - x$. The angle is weighted less if the corresponding data sample is far from the query data sample. In order to reduce the time complexity arising from dealing with all pairs of data samples for each data sample, Fast ABOD approximate ABOD by using only the pairs between k nearest neighbors in stead of all pairs of data samples. In the experiments in Section 3, Fast ABOD was used.

2.3 Outlier Detection based on histograms (HBOS)

HBOS (histogram-basedoutlierscore) assumes independence of the features which makes it fast at the cost of less precision [18].For each feature, an univariate histogram is constructed by using static bin-width or dynamic bin-width histograms. The frequency of samples falling into each bin is used as an estimate of the density. After normalizing the histograms such that the maximum height is1, the outlier score for a data sample x is calculated using the corresponding height of the bins where it is located such as

$$HBOS(x) = \sum_{i=1}^{d} log\left(\frac{1}{hist_i(x)}\right), \quad (2)$$

where *d* is the number off eatures.

2.4 Outlier Detection based on One-class SVM (OSVM)

Given a data set from an underlying probability distribution P, one-class SVM (Support vector machine) tries to estimate a simple subset Sof input space such that the probability that a test point drawn from P lies out side of S is controlled by some pre-specified value ν between 0 and1[19]. It has been applied for outlier detection independently or in combination with other methods [20,21].

Given data $\{x_1, \cdots, x_n\}$, one-class SVM maps the data into the feature space corresponding to the kernel and finds a hyperplane to separate them from the origin with maximum margin by solving the

optimization problem

$$\min_{w,\rho,\xi} \frac{1}{2}\|w\|^2 + \frac{1}{\nu n}\sum_i \xi_i - \rho \qquad (3)$$
$$subject\ to\ w \cdot \Phi(x_i) \geq \rho - \xi_i, \quad \xi_i \geq 0,$$

where ξ_i' s are slack variables penalizing data points on the negative side of a separating hyperplane $w \cdot \Phi(x) - \rho = 0$. ν is an upper bound on the fraction of data points outside the estimated region and also a lower bound on the fractions of support vectors[19].

2.5 Outlier Detection based on Local Outlier Factor (LOF)

LOF (Local Outlier Factor) measures the ratio of the peripheral density of a given data sample to that of neighboring data samples [22]. It is known that LOF works well when the regions of different densities exist. While local reachability density (*lrd*) of a data sample *x* is computed from the inverse of the average reachability distance to the *k* nearest neighbors of *x*, kNN (x), LOF is defined as

$$LOF(x) = \frac{\frac{1}{k}\sum_{z \in kNN(x)} lrd(z)}{lrd(x)}. \qquad (4)$$

LOF provides an indication of whether x is in a denser or sparser region of the neigh borhood than its neighbors [23,6].While LOF addresses with the problem of the local density variation, selecting the value for k is not trivial and the performance of LOF can be sensitive to the value of k. If groups of points might be close to one another by chance, a small k will increase the outlier scores of data samples in their locality [2]. On the other hand, a large k might cause to miss local outliers.

2.6 Outlier Detection using Isolation Forest (IF)

Among various outlier detection methods, Isolation Forest [24] is known to be computationally efficient and very effective in detecting outliers. Isolation Forest builds an ensemble of binary trees which are grown by randomly selecting a splitting feature and a random split value between the maximum and minimum values of the selected feature at each node. Under the premise that outliers are susceptible to isolation than normal data, outlier scores are computed by the average path length on Isolation trees. It has shown the competent outlier detection performance in various problems [25, 26, 27, 28].

Isolation Forest is a widely used outlier detection method, but it is difficult to apply to high dimensional data. Since tree height is limited by $ceiling(log_2\psi)$ for the sub-sampling size ψ, the total number of attributes which can be selected for node partitioning is also limited. Isolation Forest can also suffer from the sparse, irrelevant and noisy attributes in high dimensional data. In [24], the weighted selection of attributes by the kurtosis value was proposed for the application in high dimensional data. Kurtosis is a statistical measure for the thickness of the tail in the probability distribution of a real-valued random variable, which can be interpreted as the degree of presence of outliers in the distribution [29]. In [30] and [31], a method of partitioning data by a hyperplane with a random slope at each node of an isolation tree has been proposed. However, the hypothesis space of all hyperplanes with random slopes is too large in high dimensional space and the experiments were performed only for the data with the dimension below 40 and the performance for high dimensional data was not tested [31].

2.7 Outlier Detection based on Feature bagging (BLOF, BKNN)

Subspace outlier detection finds outliers in subspaces of the original data space. In [32], the subspace outlier score of a data sample is given by the degree of the deviation from the neighbors in an axis-parallel hyperplane spanned by the neighbors. Subspace outlier detection is often performed by combining outlier detection results in subspaces by random selection into an ensemble [33].

In ensemble construction of [33], the subsample size is always the same as the original input sample size, but the features are randomly sampled from half of the features to all features. The outlier score is computed by averaging or taking the maximum of all base detectors. In [33], LOF is used as the base outlier detection method. However, any detector such as KNN could be used as the base detector. In the experiments in Section 3, we test feature bagging based on LOF and KNN, denoted as BLOF and BKNN, respectively.

2.8 Outlier Detection based on Principal Component Analysis (PCA)

Principal component analysis (PCA) is a traditional linear dimension reduction method where the projection into the directions with the largest variance in data is pursued [34]. The covariance matrix of the data is decomposed to orthogonal vectors, called eigenvectors, associated with eigenvalues, and the eigenvectors with high eigenvalues which capture most of the variance in the data are used for dimension reduction. However, when it comes to outlier detection, outliers and normal data samples can be better distinguished in the hyperplane of eigenvectors with small eigenvalues.

In [35] and [2], an outlier score is computed by the weighted sum of the projected distance of a data sample to the centroid along the direction of eigenvectors as follows:

$$Score(x) = \sum_{i=1}^{d} \frac{|(x-\mu)\cdot e_i|^2}{\lambda_i}, \qquad (5)$$

where λi is an eigenvalue corresponding to an eigenvector ei. d usually denotes data dimension, but when the number of data samples is smaller than data dimension such as in high dimensional text data, d can be set as the minimum value among the number of data samples and data dimension.

2.9 Outlier Detection based on AutoEncoder (AE)

Auto Encoder (AE) is a type of neural networks for learning useful data representations in an unsupervised manner. An autoencoder consists of two parts, the encoder ϕ and the decoder ψ, and it is trained so that the reconstruction error of data instances in a training set X

$$L = \sum_{x \in X} \|x - \psi(\phi(x))\|^2 \qquad (6)$$

is minimized. The larger the reconstruction error of a test instance is, the greater the degree of its outlierness is [2, 6].

2.10 Outlier Detection based on dimension Reduction maximizing kurtosis (IF/DR)

The kurtosis is the fourth standardized moment of univariate random variable X, defined as

$$kurtosis(X) = E\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] = \frac{\mu_4}{\sigma^4}, \qquad (7)$$

where μ4 is the fourth central moment and σ is the standard deviation. The meaning of kurtosis can be interpreted that data within one standard deviation of the mean contribute virtually little to kurtosis, since raising a number that is less than 1 to the fourth power makes it closer to zero. The only data values that contribute to kurtosis in any meaningful way are those outside the region of the peak, i.e., the outliers [36]. Kurtosis reflects the shape of a distribution and high kurtosis value means heavy tails than normal. High values of kurtosis can arise in the circumstances where the probability mass is concentrated in the tails of the distribution. In [37], 2p orthogonal directions maximizing or minimizing the kurtosis value are obtained by eigenvector computation, and in the projected space by the orthogonal directions outlier scores using a univariate measure of outlyingness are used for outlier detection. However, the experiments were performed only for very small data sets with the dimension

below 5 and simulated data with the dimension below 20. Recently an outlier detection method based on dimension reduction was introduced [12], where new features maximizing kurtosis are extracted by a transformation from the original feature space and in the transformed feature space Isolation Forest is modeled. Feature extraction by a transformation which maximizes kurtosis can be performed using a neural network with no hidden layers. Denoting the weight on the edge connecting the node j of an in put layer and the node i ($1 \le i \le k$) of an output layer as wij, kurtosis on the output node i can be computed for a given input data $\{\mathbf{x_1}, \cdots, \mathbf{x_n} \mid \mathbf{x_j} \in R^d\}$ as

$$kurtosis_i = \frac{1}{n}\sum_{j=1}^{n}\left(\frac{\mathbf{w_i x_j} + b_i - \mu_i}{\sigma_i}\right)^4, \qquad (8)$$

where $\mathbf{wi} = [wi1, \cdots, wid]$, and μi and $\boldsymbol{\sigma} i$ are the mean and standard deviation of $\{\mathbf{wixj} + bi \mid 1 \le j \le n\}$ which is the mapping of $\mathbf{xj}$'s to the node i of the output layer. Using the standardization $zij = \frac{\mathbf{w_i x_j} + b_i - \mu_i}{\sigma_i}$ and applying the activation function f on each output node, the objective function can be set as

$$minimize \frac{1}{kn}\sum_{i=1}^{k}\sum_{j=1}^{n} f(-z_{ij}^4). \qquad (9)$$

In the implementation by a mini-batch stochastic method, batch normalization can be applied in place of the standardization process $z_{ij} = \frac{\mathbf{w_i x_j} + b_i - \mu_i}{\sigma_i}$. Also instead of using all the original features as input features, a subset of features with high kurtosis can be used as input features of the neural network. In the transformed data which is the output of the neural network, Isolation Forest is applied for outlier detection. We denote this method as IF/DR. The process for IF/DR can be summarized as follows:

1. Select s features with the highest kurtosis in the original feature space.

2. Construct a neural network with s input nodes and k output nodes.

3. Train the neural network to optimize the objective function in Eq. (9) by applying batch normalization.

4. Compute the transformed representation of data samples by the outputs of the trained neural net work.

5. Perform Isolation Forest in the transformed data space.

Table 1. The description of text data sets used for performance comparison.

Data	classes	number of features (nf)	number of samples (ns)
bbc	5	17005	2225
reuter	3	15484	6656
20-ng	20	44713	18774
la12	6	21604	6279
sports	5	18324	8313
classic	4	12009	7094
ohscal	10	11465	11162
reviews	4	23220	3932

3 Experimental Comparison

In this section, we perform an experimental comparison for outlier detection methods reviewed in Section 2 in various experimental setups. Also a discussion on using k-NN distances in high dimensional data is provided.

3.1 Data Description

Eight text data sets were used for performance comparison, and detailed description is given in Table 1. The BBC News data consists of 2,225 news data which belong to five categories: business, entertainment, politics, sport, and tech [38]. It was preprocessed to have 17,005 terms by deleting special symbols and numbers and removing terms appearing in only one document. Reuters-21578 was downloaded from UCI machine learning repository and the documents belonging to 135 TOPICS categories were used. After preprocessing by stopwords removal, stemming, tf-idf transformation, and unit norm, and excluding documents belonging to two or more categories, there are 6,656 documents composed of 15,484 terms. The two largest categories of 1 and 36 and the collection of the remaining all the documents compose three classes. 20newsgroup (20-ng) data contains about 20,000 articles in 20 news groups divided into 5 categories 1. After preprocessing the 20news-bydate version, we constructed 18,774 text data with 44,713 terms. The remaining five data sets were downloaded from the site 2. The final data sets were constructed removing classes with less than 200 texts and terms with frequencies less than or equal to 1.

3.2 Parameter setting for outlier detection methods

Parameter setting of the compared methods is summarized in Table 2. Most of the methods were implemented using PyOD [39] which is a python toolkit for outlier detection and all parameters were set as default values in PyOD with few exception. For example, in outlier detection based on autoencoders, the mini-batch size was set to 256 instead of the default of 32, because of the frequent

interruptions caused by division-by-zero occurrences at batch size 32. The method, IF/DR, was implemented by PyTorch [40] and the parameter values to use as default values for all text data sets were determined by preliminary experiments on BBC data.

3.3 Unsupervised Mode: Multi-class Normal Data

Thefirst experiment was to test the performance of outlier detection methods in unsupervised mode where a small percent of outliers are mixed with normal data with their true labels unknown. In particular, normal data were randomly selected from multiple classes to simulate a multi-modal distribution for normal data. For each data set in Table 1, 10% of data from one class was randomly selected as outliers, and data of all other classes were set as normal data. The performance of the outlier detection method was measured using Area Under the Curve (AUC), and the average AUC was computed by repeating the experiment about 20 times while using each class as an outlier class the same number of times. In 20-newsgroup data, the experiment was performed while repeating 20 times of setting all

Table2. Parameter setting in outlier detection methods

Methods	Parameters	Values
KNN	number of neighbors for k neighbors queries	k=5
	metric used for distance computation	Euclidean
	outlier score: the distance to the k-th neighbor	
ABOD	number of neighbors to use for k neighbors queries	k=10
HBOS	number of bins	10
OSVM	RBF kernel function: $\exp(-\gamma\|x_1 - x_2\|^2)$	$\gamma = 1/nf$
	an upper bound on the fraction of training errors	ν =0.5
LOF	number of neighbors for local density estimation	k=20
	metric used for distance computation	Euclidean
IF	number of Isolation trees in the ensemble	100
	sub-sampling size for each Isolation tree	min(256, ns)
PCA	number of principal components to keep	min(ns, nf)
	standardization preprocessed	
BLOF	the base detector	LOF or KNN
BKNN	number of base estimators in the ensemble	10
	outlier score by the average of all detectors	
AE	the number of neurons per layers	[nf, 64, 32, 32, 64, nf]
	activation function in hidden layers	relu
	number of epochs to train the model	100
	mini batch size	256
	the percentage of data to be used for validation	0.1
	batch normalization, Adam optimizer, dropout rate=0.2	
	L2 regularizer of 0.1, standardization preprocessed	
IF/DR	input features: s features with the highest kurtosis	s = 0.15*nf
	number of nodes in the output layer	100
	activation function in the output layer	sigmoid

number of epochs to train the model	10
mini batch size	200
batch normalization	
Adam optimizer with learning rate 0.01, dropout rate=0.5	

(nf: number of features, ns: number of data samples)

Table3. The performance comparison of outlier detection methods in unsupervised mode of multi-class normal and one-class outlier.

	KNN	ABOD	HBOS	OSVM	LOF	IF	BLOF	BKNN	PCA	AE	IF/DR
bbc	0.84	0.721	0.557	0.738	0.707	0.533	0.753	0.703	0.852	**0.899**	0.867
20-ng	0.791	0.774	0.525	0.653	0.607	0.483	0.759	0.782	0.802	**0.824**	**0.82**
reuter	0.66	0.619	0.522	**0.752**	0.569	0.539	0.617	0.619	0.613	0.611	0.565
la12	**0.749**	0.647	0.519	0.682	0.639	0.5	0.7	0.657	0.704	0.728	0.709
sports	0.811	0.687	0.568	0.767	0.511	0.525	0.801	0.739	**0.833**	**0.83**	0.791
classic	0.591	0.491	0.596	0.748	**0.781**	0.556	0.693	0.614	0.702	0.742	**0.782**
ohscal	0.645	0.57	0.529	0.599	0.617	0.514	0.633	0.572	**0.669**	**0.665**	0.626
reviews	0.704	0.651	0.537	0.716	0.513	0.532	0.673	0.634	0.707	0.739	**0.75**
average	0.724	0.645	0.544	0.707	0.618	0.523	0.704	0.665	0.735	**0.755**	0.739

Table 4. The performance comparison of outlier detection methods in unsupervised mode of one-class normal and one-class outlier.

	KNN	ABOD	HBOS	OSVM	LOF	IF	FLOF	BKNN	PCA	AE	IF/DR
bbc	0.91	0.795	0.591	**0.955**	0.754	0.613	0.828	0.841	0.897	0.903	0.937
20-ng	0.835	0.821	0.535	0.748	0.708	0.54	0.8	0.822	0.821	0.854	**0.88**
reuter	0.687	0.643	0.533	**0.844**	0.533	0.571	0.642	0.65	0.648	0.638	0.626
la12	0.854	0.74	0.565	**0.904**	0.666	0.566	0.784	0.751	0.737	0.746	0.794
sports	0.88	0.751	0.586	**0.92**	0.62	0.594	0.85	0.845	0.817	0.801	0.82
classic	0.853	0.761	0.535	**0.9**	0.785	0.542	0.855	0.773	0.723	0.758	0.847
ohscal	0.853	0.742	0.601	**0.891**	0.744	0.568	0.766	0.733	0.843	0.822	0.777
reviews	0.799	0.693	0.542	**0.888**	0.482	0.573	0.702	0.753	0.766	0.767	0.809
average	0.834	0.743	0.561	**0.881**	0.662	0.571	0.778	0.771	0.782	0.786	0.811

data in one category as normal data and randomly selecting one class from the other category as an outlier class.

Table 3 summarizes the average AUC for the compared methods. Three methods, AE, IF/DR, and PCA, obtained the higher performance than other methods. One common characteristic in those methods is that they utilize a weighted combination of features. One of the differences between AE and IF/DR is that IF/DR trains a simple neural network with no hidden layers, whereas AE models an encoder and decoder with two hidden layers. Also, the large performance difference between IF and IF/DR shows that kurtosis and neural network-based feature extraction can construct a transformed space which is effective in constructing an isolation forest.

3.4 Unsupervised Mode: One-class Normal Data

While the experiment in the previous section simulated the environment where normal data are drawn from multi-modal distribution, this experiment simulates the case when the normal data come from uni-modal distribution. For each data set in Table 1, 5% among data from one class was randomly selected as outliers, and all data of one class among remaining classes was set as normal data. The average AUC was measured by repeating the experiment for every pair of classes. In 20-newsgroup data, the experiment was repeated 20 times with random selection of normal and outlier classes, respectively, from each pair of categories.

Table 4 summarizes the average AUC for the compared methods. For each data set, the highest AUCis marked as a bold face. Unlike in the experiment of multi-class normal and one-class outlier, OSVMoutperforms the other methods significantly and KNN showed the second best performance. It demonstrates that the outlier detection by OSVM can be a good choice when the normal data are considered to follow uni-modal distribution. The ensemble of one-class SVMs combined with a clustering algorithm has been applied for outlier detection or classification and showed better performance than single one-class SVM [41, 42]. Hence, for data with multi-modal distribution, we can expect the utility of one-class SVMs combined with a clustering method that can work well for highdimensional data.

3.5 Semi-supervised Mode: When Multiclass Normal Data is Given as Training Data

Unsupervised outlier detection assumes that no class label is provided and that a small fraction of outliers may exist in the given data. However, since it is relatively easy to collect normal data compared to outliers, it may be practical to perform outlier detection when training data consisting only of normal data is given. In this experiment, one class was set as an outlier class and the remaining classes were set as normal classes. Of the data from each normal class, 50% was used as training data. The remaining data samples of normal classes constituted the test set together with data 50% from the outlier class. An outlier detection model is learned using normal training data and an outlier score on test data is computed by applying the model. The AUC (Area Under the Curve) is computed based on out

Table 5. The performance comparison of outlier detection methods in semi-supervised mode

	KNN	ABOD	HBOS	OSVM	LOF	IF	BLOF	BKNN	PCA	AE
bbc	**0.917**	0.69	0.501	0.74	0.886	0.484	0.744	0.253	0.734	0.706
20-ng	**0.822**	0.726	0.462	0.672	0.778	0.462	0.79	0.465	0.706	0.693
reuter	**0.872**	0.728	0.52	0.867	0.671	0.559	0.734	0.574	0.767	0.739
la12	**0.804**	0.64	0.507	0.674	0.766	0.515	0.693	0.423	0.665	0.642
sports	**0.946**	0.671	0.523	0.833	0.87	0.482	0.861	0.331	0.851	0.813
classic	0.885	0.424	0.539	0.795	**0.902**	0.511	0.75	0.286	0.669	0.647
ohscal	**0.659**	0.575	0.519	0.611	0.642	0.497	0.615	0.487	0.601	0.586
reviews	**0.775**	0.631	0.535	0.739	0.719	0.524	0.704	0.417	0.695	0.667
average	**0.835**	0.636	0.513	0.741	0.779	0.504	0.736	0.405	0.711	0.687

lier scores of test data. This process was repeated about 20 times while changing the outlier class.

Table 5 summarizes the average AUC for the compared methods. When the experimental results are analyzed, it should be considered that some of the methods are more suitable for outlier detection in unsupervised mode. Nevertheless, the highest detection performance by KNN in semi-supervised mode is surprising, considering the phenomenon by the curse of dimensionality in high dimensional space. In the next section, we check the validity of using k-NN distances for outlier detection in highdimensional data.

3.6 Discussion on Using k-NN Distance in High Dimensional Space

It is known that due to data sparsity all pairs of data samples are almost equidistant in high dimensional data space and the difference between the maximumdistance and minimum distance compared to the minimum distance vanishes as the dimensionality increases. However, as shown in the experiments of previous sections, outlier detection based on the distance to k nearest neighbors showed the performance of the high rank among the compared methods, especially in semi-supervised mode where normal training data are given.

We try to explain the reason why k nearest neighbors-based outlier detection worked competently in spite of the phenomenon from the curse of dimensionality. The research in [16] has shown that the concentration effect of the distance measure only holds in the artificial scenario when the one-dimensional distributions are independent and identically distributed, and the curse of dimensionality is not the main problem for outlier detection in high dimensional data. We conducted the experiment to compare the distance from an outlier or normal data to other data samples using text data in Table 1.

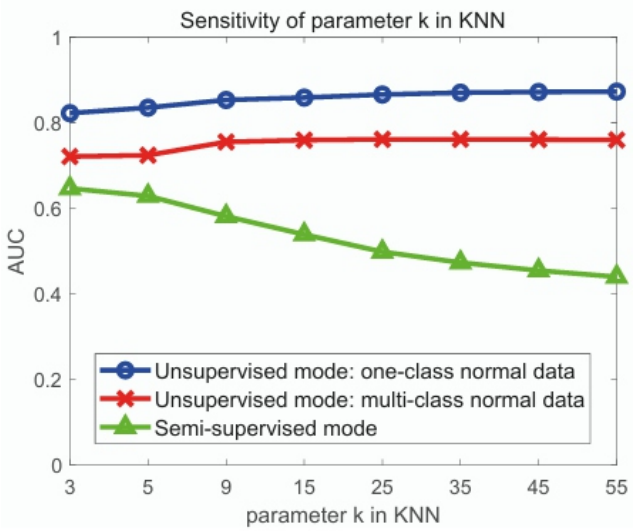
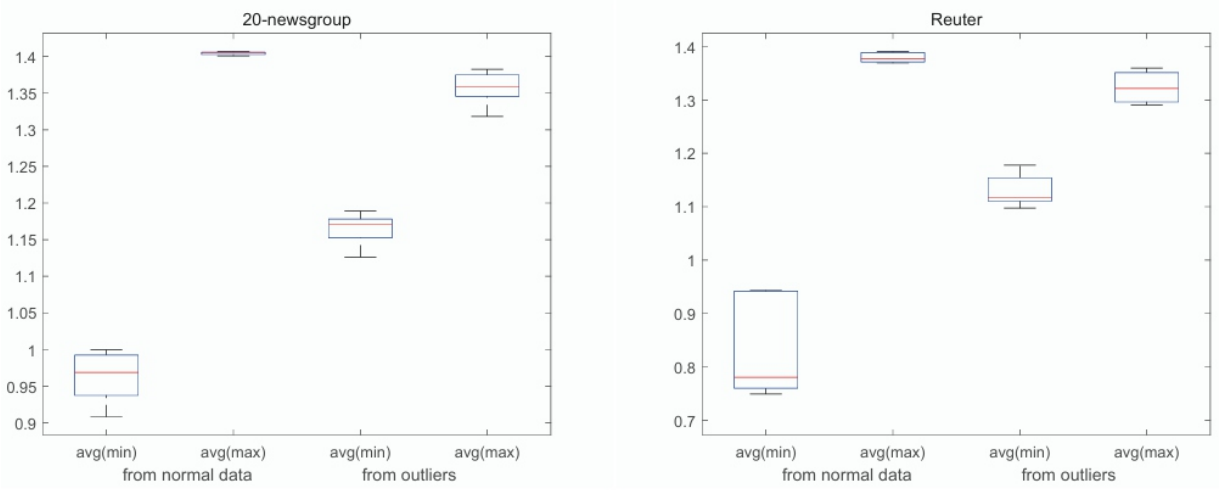


Figure 2. Comparison of outlier detection performance by KNN while changing the value of k. The average AUC in three experimental setups is shown.

Two figures in the top row of Figure 1 compare the maximum and minimum distances from normal or outliers of the test data set to normal data samples of the training set in semi-supervised mode of Section 3.5, and two figures in the bottom row compare the maximum and minimum distances between data samples in unsupervised mode of Section 3.3. The figures show that the average minimum distance from outliers is greater than that from normal data samples. On the other hand, the average maximum distance from outliers is almost equal to or smaller than that from normal data samples, implying that the distance concentration effect is stronger in outliers than in normal data samples. Hence, the distance to the nearest neighbors in high dimensional data can be used effectively for the discrimination



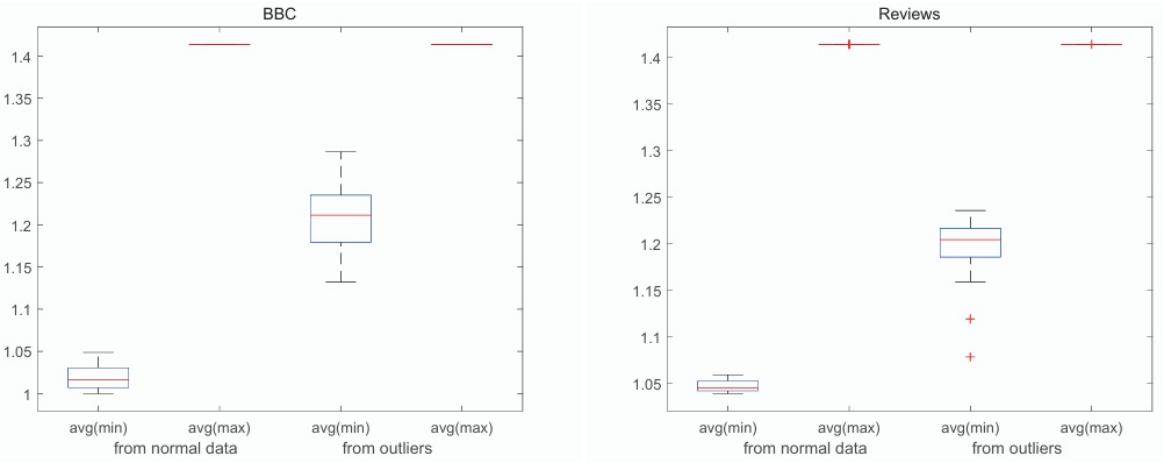


Figure1. Comparison of the maximum and minimum distances from normal or outliers to other data samples. top row: measured in semi-supervised mode, bottom row: unsupervised mode

of outliers and normal data samples with the careful selection of the value *k* ink-NN search.

Next, to test the sensitivity of the parameter *k* in KNN, we compared outlier detection performance by KNN while changing the value of *k*. Figure 2 shows the average AUC by the KNN method in three experimental setups. AsshowninFigure2, in the unsupervised mode, stable performance was observed in the range of 25to45,while in the semisupervised mode, high performance was obtained at small *k* values.

3.7 Running Time Comparison of Outlier Detection Methods

We measured the running time when performing the outlier detection method in unsupervised mode of Section 3.3. The computer used had a CPU Intel i9-9900X(3.50GHz), RAM32GB. Figure3(a)shows the measured CPU time in seconds while running on bbc and 20-newsgroup data respectively. The methods based on one-class SVM, PCA, and feature bagging showed are latively high execution time compared to the other methods. Figure 3(b) shows the average AUC values by outlier detection method scopied fromTable3,4,5.

4 Discussions

In this paper, a comparative study for outlier detection methods in high dimensional data was performed and experimental results using text data were analyzed. In particular, experimental setups were simulated to compare the performance of out lier detection methods in unsupervised versus

semisupervised mode and uni-modal versus multi-modal data distributions. Experimental results can be summarized as follows:

–Outlier detection methods utilizing feature transformation such as autoencoder, PCA, or kurtosis-baseddimensionreductionachievedthe highest performance in the unsupervised mode when normal data consisted of multiple classes. However, in the semi-supervisedmodewhere the class label of normal data is given, outlier detection methods such as KNN, LOF, or oneclass SVM were better than AEorPCA-based methods.

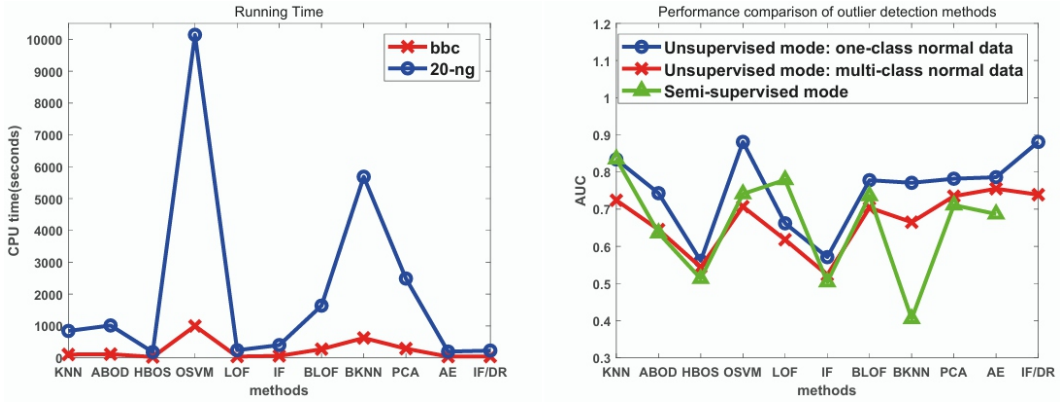


Figure 3. (a) Comparison of CPU time in seconds while running on bbc and 20-newsgroup data. (b) Performance comparison of outlier detection methods in three experimental settings by the average AUC.

– In unsupervised mode, bagging by feature selection using the base detector LOF achieved higher performance than using the single detector LOF, butit did not show the best performance among the compared methods. It is presumed to be caused by the characteristic that text data has many zero components.

– Outlier detection based on one-class SVM showed significantly higher performance when normal data consisted of one class. On the other hand, on multi-class normal data, the performance was lower than that of the KNN-based method.

– A dimension reduction method was introduced that maximizes kurtosis, which can be implemented using a simple neural network with no hidden layers. Experimental results have proven that the performance of the Isolation Forest built in a dimension reduced space is greatly improved.

– Outlier detection based on distance to k nearest neighbors worked well despite the curse of dimensionality in high dimensional space. Especially, it was prominent in semi-supervised mode

where normal training data is given. The feasibility of using k-NN distances for outlier detection in high-dimensional data was experimentally examined.

Experimental comparison has the limitation that it requires optimization of parameter values for each method and data set. However, parameter optimization is not easy unless a validation set of outliers is not provided. Instead, for all the data sets we used default parameter values recommended in the PyOD package with very little exceptional cases. Regardless of that limitation, consistent findings can provide insight into the application of outlier detection methods in high dimensional text data.

Acknowledgments

This work was partly supported by Institute of Information & communications Technology Planning Evaluation (IITP) grant funded by the Korea government(MSIT) (No.RS-2022-00155857, Artificial Intelligence Convergence Innovation Human Resources Development (Chungnam National University)) and research fund of Chungnam National University.

References

[1] D. Hawkins. Identification of Outliers. Chapman and Hall, 1980.

[2] C. Aggarwal. Outlier analysis (2nd ed.) Springer, 2017.

[3] Caroline Cynthia and Thomas George. An outlier detection approach on credit card fraud detection using machine learning: A comparative analysis on supervised and unsupervised learning. In: Peter J., Fernandes S., Alavi A. (eds) Intelligence in Big Data Technologies-Beyond the Hype. Advances in Intelligent Systems and Computing, 1167, 2021.

[4] H. Mazzawi, G. Dalai, D. Rozenblat, L. Ein-Dor, M.Ninio, O. Lavi, A. Adir, E. Aharoni, and E. Ker many. Anomaly detection in large databases using behavioral patterning. In ICDE, 2017.

[5] T. Li, J. Ma, and C. Sun. Dlog: diagnosing router events with syslogs for anomaly detection. The Journal of Supercomputing, 74(2):845–867, 2018.

[6] C. Park. Outlier and anomaly pattern detection on data streams. The journal of supercomputing, 75:6118–6128, 2019.

[7] H. Wang, M. Bah, and M. Hammad. Progress in outlier detection techniques: A survey. IEEE Access, 7, 2019.

[8] A. Boukerche, L. Zheng, and O. Alfandi. Outlier detection: Methods, models, and classification. ACMComputing Surveys, 53:1–37, 2020.

[9] X. Zhao, J. Zhang, and X. Qin. Loma: A local outlier mining algorithm based on attribute relevance analysis. Expert Systems with Applications, 84, 2017.

[10] X. Zhao, J. Zhang, X. Qin, J. Cai, and Y. Ma. Parallel mining of contextual outlier using sparse subspace. Expert Systems with Applications, 126, 2019.

[11] F. Kamalov and H. Leung. Outlier detection in high dimensional data. Journal of Information and Knowledge Management, 19, 2020.

[12] C. Park. A dimension reduction method for unsupervised outlier detection in high dimensional data(written in korean). Journal of KIISE. In press.

[13] S.Damaswanny,R.Rastogi, andK.Shim. Efficient algorithms for mining outliers from large data sets. In Proceeding of ACM SIGMOD, pages 427–438, 2000.

[14] E. Knorr and R. Ng. Finding intensional knowledge of distance-based outliers. In Proceeding of 25th International Conference on Very Large Databases, 1999.

[15] M. Sugiyama and K. Borgwardt. Rapid distancebased outlier detection via sampling. In International Conference on Neural Information Processing Systems, 2013.

[16] A. Zimek, E. Schubert, and H. Kriegel. A survey on unsupervised outlier detection in highdimensional numerical data. Statistical Analysis and Data Mining, 5:363–387, 2012.

[17] H. Kriegel, M. Schubert, and A. Zimek. Anglebased outlier detection in high-dimensional data. In Proceeding of KDD, pages 444–452, 2008.

[18] M.GoldsteinandA.Dengel. Histogram-based outlier score (hbos): a fast unsupervised anomaly detection algorithm. In Proceeding of KI, pages 5963, 2012.

[19] B. Scholkopf, J. Platt, J. Shawe-Taylor, and A. Smola. Estimating the support of a highdimensional distribution. Neural computation, pages 1443–1471, 2001.

[20] M. Amer, M. Goldstein, and S. Abdennadher. Enhancing one-class support vector machines for unsupervised anomaly detection. In Proceedings of the ACMSIGKDDWorkshoponOutlierDetection and Description, 2013.

[21] L. Ruff, R. Vandermeulen, N. Gornitz, L. Deecke, S. Siddiqui, A. Binder, E. Muller, and M. Kloft. Deep one-class classification. In Proceeding of international conference on machine learning, 2018.

[22] M. Breunig, H. Kriegel, R. Ng, and J. Sander. Lof: Identifying density-based local outliers. In Proceeding of the ACM Sigmod International Conference on Management of Data, 2000.

[23] P. Tan, M. Steinbach, and V. Kumar. Introduction to Data Mining. Addison Wesley, Boston, 2006.
[24] F. Liu, K. Ting, and Z. Zhou. Isolation forest. In Proceedings of the 8th international conference on data mining, 2008.

[25] G. Susto, A. Beghi, and S. McLoone. Anomaly detection through on-line isolation forest: An application to plasma etching. In the 28th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC), pages 89–94, 2017.

[26] L. Puggini and S. MCLoone. An enhanced variable selection and isolation forest based methodology for anomaly detection with oes data. Engineering Applications of Artificial Intelligence, 67:126135, 2018.

[27] J. Kim, H. Naganathan, S. Moon, W. Chong, and S. Ariaratnam. Applications of clustering and isolation forest techniques in real-time building energy-consumption data: Application to leed certified buildings. Journal of energy Engineering, 143, 2017.

[28] J. Hofmockel and E. Sax. Isolation forest for anomaly detection in raw vehicle sensor data. In the 4th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2018), pages 411–416, 2018.

[29] J. Livesey. Kurtosis provides a good omnibus test for outliers in small samples. Clinical Biochemistry, 40:1032–1036, 2007.

[30] F. Liu, K. Ting, and Z. Zhou. On detecting clustered anomalies using sciforest. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2010.

[31] S. Hariri, M. Kind, and R. Brunner. Extended isolation forest. IEEE transactions on knowledge and data engineering, 33:1479–1489, 2021.

[32] H. Kriegel, P. Kroger, E. Schubert, and A. Zimek. Outlier detection in axis-parallel subspaces of high dimensional data. In Proceedings of PAKDD, 2009.

[33] A. Lazarevic and V. Kumar. Feature bagging for outlier detection. In Proceedings of KDD, 2005.
[34] R. Duda, P. Hart, and D. Stork. Pattern classification (2nd ed.). Wiley-interscience, 2000.

[35] M. Shyu, S. Chen, K. Sarinnapakorn, and L. Chang. A novel anomaly detection scheme based on principal component classifier. In Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop, 2003.

[36] P. Westfall. Kurtosis as peakedness, 1905-2014. r.i.p. The American Statistician, 68(3):191–195, 2014.

[37] D. Pena and F. Prieto. Multivariate outlier detection and robust covariance matrix estimation. Technometrics, 43:286–310, 2001.

[38] D. Greene and P. Cunningham. Practical solutions to the problem of diagonal dominance in kernel document clustering. In Proceeding of ICML, 2006.

[39] Y. Zhao, Z. Nasrullah, and Z. Li. Pyod: A python toolbox for scalable outlier detection. Journal of Machine Learning Research, 20:1–7, 2019.

[40] A. Paszke, S. Gross, F. Massa, and et. al A. Lerer. Pytorch: An imperative style, high-performance deep learning library. In NeurIPS, pages 80268037, 2019.

[41] L. Abdallah, M. Badarna, W. Khalifa, and M. Yousef. Multikoc: Multi-one-class classifier based k-means clustering. Algorithms, 14(5):1–10, 2021.

[42] B. Krawczyk, M. Wozniak, and B. Cyganek. Clustering-based ensemble for one-class classification. Information sciences, 264:182–195, 2014.

Cheong Hee Park received her Ph.D. in Mathematics from Yonsei University, Korea in 1998. She received the M.S. and Ph.D. degrees in Computer Science at the Department of Computer Science and Engineering, University of Minnesota in 2002 and 2004 respectively. She is currently in the Department of Computer Science and Engineering, Chungnam National University, Korea as a professor. Her research interests include machine learning, data mining, and pattern recognition. https://orcid.org/0000-0002-8233-2206

ISSN 2083 - 2567

ANOMALY PATTERN DETECTION IN STREAMING DATA BASED ON THE TRANSFORMATION TO MULTIPLE

Taegong Kim and Cheong Hee Park*
Department of Computer Science and Engineering, Chungnam National University,
220 Gung-dong, Yuseong-gu
Daejeon, 305-763, Korea
*E-mail: cheonghee@cnu.ac.kr

ABSTRACT

Anomaly pattern detection in a data stream aims to detect a time point where outliers begin to occur abnormally. Recently, a method for anomaly pattern detection has been proposed based on binary classification for outliers and statistical tests in the data stream of binary labels of normal or an outlier. It showed that an anomaly pattern can be detected accurately even when outlier detection performance is relatively low. However, since the anomaly pattern detection method is based on the binary classification for outliers, most well-known outlier detection methods, with the output of real-valued outlier scores, can not be used directly. In this paper, we propose an anomaly pattern detection method in a data stream using the transformation to multiple binary-valued data streams from real-valued outlier scores. By using three outlier detection methods, Isolation Forest(IF), Autoencoder-based outlier detection, and Local outlier factor(LOF), the proposed anomaly pattern detection method is tested using artificial and real data sets. The experimental results show that anomaly pattern detection using Isolation Forest gives the best performance.

Keywords: anomaly pattern detection, multiple binary-valued streams, outlier detection, outlier score.

Introduction

An outlier detection method predicts whether a data sample is an outlier [1]. On the other hand, anomaly pattern detection in a data stream aims to find a time point where outliers suddenly begin to occur heavily. A sudden increase of outliers might indicate that an unusual event has happened [2]. The anomaly pattern detection method which was recently proposed in [3] utilizes an outlier detection method that performs the binary classification for outliers. By applying the outlier detection method to each data sample in a data stream, a data stream is transformed into the stream of binary values indicating normal or an outlier for each data sample. Then the occurrence of an anomaly pattern is detected on the stream of binary values by comparing binomial distributions in a reference window and a detection window.

Well-known outlier detection methods such as Isolation Forest(IF) [4], Autoencoder-based outlier detection [5, 6], and Local outlier factor(LOF) [7] compute outlier scores which measure the degree of anomaly in a data sample. Since the anomaly pattern detection method in [3] requires a binary outlier detection method in order to transform a data stream to a stream of binary values, most outlier detection methods, with the output real-valued outlier scores, can not be used directly. Although binary labels of normal or outliers can be obtained by setting a threshold over outlier scores, it is difficult to determine the optimal threshold.

In this paper, we present an anomaly pattern detection method in a data stream based on the transformation into multiple binary-valued data streams. Given a training set of normal data samples, an outlier detection model is constructed using the training set and it is applied to each data sample on a data steam, resulting in a stream of realvalued outlier scores. Multiple thresholds over outlier scores by which abnormal data samples could be distinguished from normal data samples are induced from outlier scores of normal training data samples. By applying the thresholds to a data stream of the real-valued outlier scores, multiple data streams of binary values are obtained. Two approaches for anomaly pattern detection on multiple binary-valued data streams are suggested. The proposed methods, APD-HT/IF, APD-HT/AE, and APD-HT/LOF based on Isolation Forest(IF), autoencoder-based outlier detection, and Local outlier factor(LOF)-based outlier detection methods respectively are tested comparing the performance with the method APD-HT in [3].

The contribution of the paper can be summarized as follows.

– Given normal training data, an anomaly pattern detection method on a data stream is proposed extending the method in [3].

– By setting multiple thresholds from outlier scores of training data samples, an ensemble of anomaly pattern detectors is constructed.

– Well-known outlier detection methods which give the output of outlier scores can be utilized in the proposed method.

The remainder of the paper is organized as follows. In Section 2, the anomaly pattern detection method in [3] is reviewed. In Section 3, we propose a method for anomaly pattern detection in a stream of real-valued outlier scores. Experimental results are given in Section 4 and discussion follows in Section 5.

2 Anomalypattern detection based on binary classification of outliers

While enormous research for outlier detection has been conducted, anomaly pattern detection in a data stream has not been studied extensively [8, 2]. For anomaly pattern detection, in [9, 10], a change in probability distribution on a data stream of real-valued outlier scores was detected based on a one-dimensional Gaussian distribution assumption. In [11, 12], anomaly pattern detection was performed through rule induction on categorical attributes. The method [3] which was published recently showed competent performance for anomaly pattern detection. It detects a burst occurrence of outliers on a binary-valued data stream which is induced by binary classification for outliers.

In [3], given a training set consisting of normal data samples, an outlier detection model is constructed which gives a binary label indicating normal or an outlier for a data sample. A clusteringbased outlier detection method was used for binary outlier prediction. The normal data region in the training set is modeled as a union of hyperspheres by performing k-means clustering on the training data. By partitioning training data to several chunks and applying k-means clustering for each chunk, the ensemble of cluster models can be constructed. Whenatest data sample is not included in the nearest hypersphere in any of the cluster models, it is predicted to be an outlier. By performing outlier prediction for each data sample in a data stream, a data stream is transformed into the stream of binary values where 1 stands for an outlier and 0 for normal.

Two methods for detecting a time point where a sudden burst of outliers occurs were proposed: APD-HT(Anomaly Pattern Detection by Hypothesis Testing) and APD-CC(Anomaly Pattern Detection by Control Charts). Since it was shown that the performance of APD-HT is generally higher than APD-CCin[3],wefocusonthereviewofAPD-HT. In APD-HT, a reference window is set in the beginning part of the binary-valued data stream which is considered as consisting of normal data samples and a detection window is moved forward as a new data sample arrives. Let X and Y be the number of outliers in the reference window and the detection window whose size is m and n respectively. Supposing that the samples in the reference window are generated from the binomial distribution of the proportion p1 and the samples in the detection window are generated from the binomial distribution of the proportion p2, a hypothesis test about the equality of proportions is performed. Under the null hypothesis H0 : p1−p2 =0 and alternative hypothesis H1 : p1−p2 <0,

$$
\begin{aligned}
&p\text{-value} = P(Z < z_0),\\
&\text{where } \hat{p}_1 = \frac{X}{m}, \hat{p}_2 = \frac{Y}{n}, \hat{p} = (X+Y)/(m+n),\\
&\text{and } z_0 = \frac{\hat{p}_1 - \hat{p}_2}{\sqrt{\hat{p}(1-\hat{p})(1/m+1/n)}},
\end{aligned}
$$

is computed. For a significance level α, if the pvalue is less than α, then H0 is rejected and anomaly pattern detection is declared. Otherwise, a detection window moves forward for a new hypothesis test.

3 Anomalypattern detection based on the transformation to multiple binary-valued data streams

Binary classification of outliers based on kmeans clustering performs reasonably well and the anomaly detection method in [3] is less affected by the performance of outlier prediction. However, if a well-known outlier detection method can be used, it will improve anomaly pattern detection as well as outlier pattern detection.

Given training data consisting of normal data samples, an outlier detection model is constructed which computes an outlier score of a data sample. From the distribution in outlier scores of training data samples, a threshold for outlier prediction can be induced. However, it is difficult to expect that one threshold value optimally distinguishes outliers from normal data.

Instead of choosing one threshold for outlier prediction, we can use multiple thresholds. For each threshold, the data stream of the real-valued outlier scores is transformed into a stream of binary values indicating outlier prediction with 1 and normal prediction with 0, and the anomaly pattern detection method, APD-HT, is applied in each binaryvalued data stream. Now there are two questions to answer: how to set multiple thresholds, and how to combine the results from APD-HT on multiple binary-valued data streams. In the next subsections, these problems are addressed.

3.1 Howtoset multiple thresholds for outlier prediction

Suppose that outlier scores are small when the anomalous degree of data samples is high, as in the implementation of Isolation Forest algorithm in scikit Learn [13]. Then it is generally expected that the score of outliers is smaller than that of normal data samples. Since a training set contains only normal data samples, thresholds can be chosen using outlier scores of normal training data. We choose the threshold using the percentile numbers in the distribution of outlier scores of normal training data samples. A percentile is a value below which a given percentage of observations in a group of observations falls. When the outlier scores of normal data samples and outliers in a test data set are expected to be separated well, threshold values smaller than 1th-percentile can work well. However, when the outlier scores of normal data samples and outliers in a test set are mixed in a wide range, using too small percentile values as a threshold can not differentiate outliers from normal data samples. In the experiments in Section 4, we test the effects of using various thresholds on the performance of anomaly pattern detection.

3.2 How to combine the results from multiple binary data streams

The anomaly pattern detection method, APDHT, is applied to each binary data stream. A detection window moves forward while the reference window is fixed at the front of the data stream. The distributions in two windows are compared to detect the burst of 1's in the detection window. Hence, a positive or negative prediction is made in each binary data stream and these predictions are combined to make a final decision. We tested two approaches for combining the predictions from multiple binary data streams: All-Agreed and HalfAgreed approaches. In the All-Agreed approach, when the predictions from all the data streams are positive, anomaly pattern detection is declared. On the other hand, in the Half-Agreed approach, when the majority of predictions are positive, anomaly pattern detection is declared. Table 1 summarizes the algorithm of the proposed method.

3.3 Outlier detection methods

Most of outlier detection methods calculate outlier scores which represent the degree at which a data sample deviates from the normal data range. Among various outlier detection methods, we used Isolation Forest and Autoencoder-based outlier detection and Local outlier factor(LOF)-based outlier detection in the experiments of Section 4.

Isolation Forest(IF) [4] is a well-known treebased outlier detection method. Isolation Forest is based on the assumption that outliers are easy to isolate from the remainders of the data. It grows a binary tree by selecting randomly an attribute and a splitting value of the attribute. The process is repeated recursively until all training data samples are isolated at the leaf nodes. The set of isolation trees built on subsets of a training data is called isolation forest. The length of a path where a data sample traverses from the root node to a leaf node is used to compute an outlier score.

Recently, various deep learning-based methods have been used for outlier detection. Autoencoder consists of two components of neural networks. An input data is encoded to low dimensional representation by the first network called an encoder, and it is again decoded to the original dimensional space by the second network called a decoder. Autoencoder is trained so that the error between the reconstructed one and the input data is minimized. Also, the error is used to compute an outlier score.

LOF(Local outlier factor) gives an outlier score based on local density around a data sample [14].

$$LOF(p) = \frac{\frac{1}{k}\sum_{q \in \mathrm{kNN}(p)} lrd(q)}{lrd(p)} \qquad (1)$$

kNN(p) denotes a set of k nearest neighbors of a data sample p. Local reachability density(lrd) of p is computed from the inverse of the average reachability distance to the k nearest neighbors of p. LOF provides an indication of whether p is in a denser or sparser region of the neighborhood than its neighbors [15]

Table 2. Data description

Data	attr.	samples	outlier ratio(%)
Creditcard	28	284,807	0.17
Kdd-Http	3	567,498	0.39
Annthyroid	6	7,200	7.42
Shuttle	9	49,097	7.15
Gaussian	1	102,500	2.44
RBFevents	5	100,000	10.2
Covtype	10	286,048	0.96
Satellite	36	5,100	1.4
Mammography	6	11,183	2.3

4 Experiments

4.1 Experimental Setup

To compare the performance of the proposed anomaly pattern detection method, we used nine data sets including real or artificial data. Detailed description is shown in Table 2. Creditcard data1 is a summary of credit card usage by some card holders in Europe in September 2013. Excluding the attribute indicating usage amount, 28 attributes were used. KDD-http data is a subset of KDD Cup data2, which is composed of data samples whose value of attribute service was http. Three attributes, duration, src-bytes, and dst-bytes, were used as in [16]. Gaussian data is 1-dimensional data where 100,000 normal samples were generated using 5 Gaussian mixture distributions with mean values 0, 1, 2, 3 and 4, and 2,500 abnormal data samples were generated from Gaussian distribution with the mean value 6. RBFevents data was constructed using the artificial streaming data generator RandomRBFevents of MOA [17] with normal data from five normal distributions and outliers from a random uniform distribution.

Other data sets were downloaded from the OpenML data repository. In Annthyroid data, the data samples in two classes, hyperfunction and subnormal functioning, were considered to be outliers. In Shuttle data, data samples of class 1 were treated

DARPA Intrusion Detection Evaluation Program:

Table1. The algorithm of the proposed method.

Input: X: a training set of normal data samples
$x_1, x_2, x_3, \cdots$: a test data stream
$\{p_1, p_2, \cdots, p_k\}$: a set of threshold values
Combining strategy: All-Agreed or Half-Agreed
Construct the outlier detection model F by using X.
Apply F for data samples in X and obtain a set of outlier scores G.
Let p_ith-percentile in G be s_i. $(i = 1, \cdots, k)$
Initialize k empty binary streams $B_1, \cdots, B_k$.
for $t = 1, 2, 3, \cdots$
Compute the outlier score for x_t by applying F.
The binary value obtained by the threshold s_i is added to the end of the binary stream B_i for $i = 1, \cdots, k$.
Apply APD-HT on each binary stream B_i $(i = 1, \cdots, k)$, where a reference window is fixed
in the beginning part of a data stream and a detection window is set at the end of the stream
including the newly arrived instance.
Combine the predictions from all k binary streams by the combining strategy.
if anomaly pattern detection is signaled
Give an alarm for anomaly pattern detection and exit
end if
end for

Table3. Comparison of fanomaly pattern detection performance. Thresholds of 0.1th, 0.5th, 1th, 2th, and 3th percentiles were reused.

	APD-HT [3]	APD-HT/IF		APD-HT/AE		APD-HT/LOF	
F1		All	Half	All	Half	All	Half
Creditcard	1	1	0.84	1	0.8	0	0.84
Kdd-Http	1	1	0.79	1	0.8	0	0.02
Annthyroid	0.96	0.96	0.98	1	0.95	1	0.98
Shuttle	0.84	1	0.93	0.6	0.91	1	0.91
Gaussian	1	1	0.9	1	0.88	0.99	0.84
RBFevents	1	1	0.92	1	0.9	1	0.87
Covtype	1	1	0.87	1	0.86	1	0.84
Satellite	0.96	0.98	0.96	1	0.99	1	0.98
Mammo.	0.92	1	0.98	1	0.98	0.04	0.99
mean	0.96	**0.99**	0.91	0.96	0.9	0.67	0.81

	APD-HT [3]	APD-HT/IF		APD-HT/AE		APD-HT/LOF	
Delay		All	Half	All	Half	All	Half
Creditcard	13.2	20.6	15.6	22.8	15.9	-	75.5
Kdd-Http	27.2	15.2	10.8	15.2	10.6	-	11
Annthyroid	72.7	129.6	43.6	15.8	9.4	67.7	32.9
Shuttle	7.6	15.6	9.5	215.9	12.1	15.8	9.4
Gaussian	5.9	17.1	11.1	15.4	9.6	18.8	11.8
RBFevents	5.0	14.7	8.9	14.7	9.6	16.3	10.8
Covtype	134.3	183.8	30.5	15.0	9.8	17.2	12.3
Satellite	21.9	101.3	32.9	14.5	9.4	22.7	15.8
Mammo.	50.0	20.5	14.7	16.1	10.2	210	42.7
mean	37.5	57.6	19.7	38.4	10.7	52.6	24.7

as normal data and data samples in the other classes except class 4 were set as outliers as in [4]. Covtype data targets the prediction of forest cover types from cartographic variables. 10 numeric attributes were used and data samples of class 2 were treated as normal data and data samples of class 4 were set as outliers. In Shuttle data, data samples of class cotton crop and soil with vegetable stubble were used as outliers.

Each data sample is split to a training set which consists of normal data corresponding to 30% of the total data samples and a test set of remaining data samples by which a test data stream is built. In order to simulate the occurrence of an anomaly pattern on a test data stream, outliers were arranged after a sequence of normal data, and the starting point of outliers was set as the actual anomaly pattern occurrence point. An outlier detection model is constructed by the training data and anomaly pattern detection is performed on a test data stream.

The experiment was repeated 100 times by randomly splitting to training and test data. For each test case, when anomaly pattern occurrence is predicted after the actual anomaly pattern occurrence point, it is counted as TP (True Positive) prediction. When an anomaly pattern is predicted before the actual anomaly pattern occurrence point, it is considered FP(False Positive) prediction. If anomaly pattern occurrence is not detected until the end of the test stream, it is marked as FN (False Negative) prediction. In a case of a true positive detection, a distance from the actual anomaly pattern occurrence point to the anomaly pattern prediction point is measured as Delay. After 100 experiments, from the accumulated TP, FP, and FN, the F1 value is computed by Equation 2 along with the average value of Delay.

$$F1 = \frac{2 * precision * recall}{precision + recall}, \qquad (2)$$

$$\text{where } precision = \frac{TP}{TP+FP}, \; recall = \frac{TP}{TP+FN}.$$

4.2 Comparison of anomaly pattern detection performance

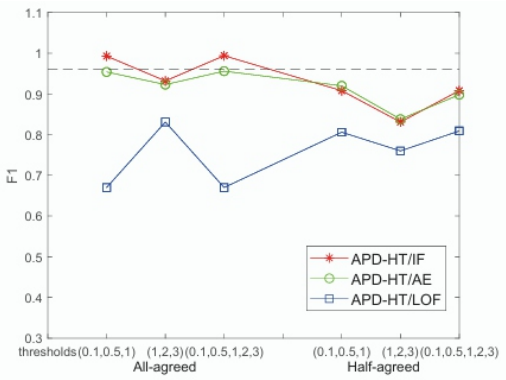
The performance of the proposed anomaly pattern detection method was compared with the method APD-HT in [3]. APD-HT [3] uses theensemble of k-means clustering for binary outlier prediction. As in [3], the ensemble of three clustering models with the number of clusters 30 was used. The proposed methods, APD-HT/IF, APDHT/AE, and APD-HT/LOF, are based on outlier detection methods of Isolation Forest(IF), Autoencoder(AE), and Local outlier factor(LOF), respectively. For Isolation Forest, the implementation in scikit-learn [13] was used, where the sub-sampling size was the number of training data samples and the ensemble size was 100. The tree height limit H is automatically set by the sub-sampling size ψ as H=ceiling(log2ψ). Outlier scores by Autoencoder and LOF were computed using the implementation in PyODwhich is a Python toolkit for outlier detection [18]. In Autoencoder, 6 hidden layers were set where the number of nodes in an encoder part increased 1.5 times per layer and decreased reversely per layer in a decoder part. Default parameter values in PyOD implementation were used where relu activation function was utilized in all the hidden layers and sigmoid function in the output layer, and Adam optimizer was used. In LOF, the number of neighbors was set to 20 which is a default value in PyOD, and the Euclidean distance metric was used. For Autoencoder and LOF methods, preprocessing of the standard normalization was performed for all the data sets except Gaussian data which is onedimensional data.

For all compared methods, the size of the reference window and the detection window was set as 400, and the significance level of the hypothesis test was set at 0.01 as in [3]. Five thresholds of 0.1th, 0.5th, 1th, 2th, 3th percentiles were set for the transformation from the stream of outlier scores to binary-valued streams. Table 3 shows the performance of the compared methods in 100 repeated tests by F1 and the average value of Delay. Isolation Forest outlier detection method combined with APD-HT showed the highest F1 value compared with other methods. The all-agreed strategy gave higher F1 value in APD-HT/IF and APDHT/AEthanthehalf-agreed approach, while the average delay is smaller when using the half-agreed approach.

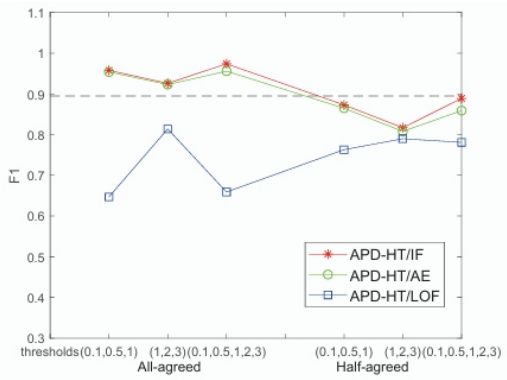
As in [3], we also tested anomaly pattern detection performance in noisy environments. This was simulated by inserting 10% outliers randomly into the normal data sequence and mixing the same

Table4. Comparison of anomaly pattern detection performance in the simulation of a noisy environment. Thresholds of 0.1th, 0.5th, 1th, 2th, 3th percentiles were used in the proposed methods.

	APD-HT [3]	APD-HT/IF		APD-HT/AE		APD-HT/LOF	
F1		All	Half	All	Half	All	Half
Creditcard	1	1	0.85	1	0.84	0	0.85
Kdd-Http	1	0.98	0.86	1	0.77	0	0
Annthyroid	0.95	0.95	0.96	0.99	0.96	1	0.96
Shuttle	0.74	0.96	0.87	0.57	0.82	0.95	0.84
Gaussian	0.84	0.98	0.85	0.97	0.84	1	0.86
RBFevents	0.68	0.91	0.8	0.92	0.8	0.94	0.78
Covtype	1	1	0.83	0.99	0.8	0.99	0.77
Satellite	0.94	0.98	0.98	1	0.98	1	0.98
Mammo.	0.91	1	0.99	1	0.92	0.04	0.98
mean	0.90	**0.97**	0.89	0.94	0.86	0.66	0.78
	APD-HT [3]	APD-HT/IF		APD-HT/AE		APD-HT/LOF	
Delay		All	Half	All	Half	All	Half
Creditcard	14.9	22.2	16.5	27.2	17.8	-	72.8
Kdd-Http	30.5	17.1	11.6	15.9	11.9	-	-
Annthyroid	83.3	138.5	53.3	18.7	15.7	86.6	37.0
Shuttle	12.7	18.7	13.6	377.3	17.6	21.8	15.4
Gaussian	9.3	19	14.5	18.4	13.1	20.7	16.1
RBFevents	15.9	22.2	17.6	20.8	18.0	22.9	17.0
Covtype	135.2	198.2	33.7	16.7	11.5	20.9	14.7
Satellite	27.2	105.6	39.1	16.5	11.3	24.4	18.0
Mammo.	55.5	24.2	17.4	20.1	13.5	233	48.7
mean	42.7	62.9	24.1	59.1	14.5	61.5	30.0



(a) In the experimental setting of Table3



(b) In the simulation of a noisy environment of Table4

Figure1. Comparisonofanomalypatterndetectionperformancewhendifferentthresholdvalueswere used.

number of normal data with the outlier data sequence. Table 4 compares the performance in the simulated noisy situations. APD-HT/IF showed the decrease from 0.99 to 0.97 in the F1 value on average compared with the performance in non-noisy situation of Table 3. On the other hand, in APD-HT [3] the decrease from 0.96 to 0.9 in the F1 value was obtained.

4.3 Performance comparison under various threshold values

Figure 1 compares the average F1 values in nine data sets when different threshold values were used in the experimental setting of Table 3 and 4. With All-agreed strategy, APT-HT/IF and APTHT/AE obtained high F1 values when relatively small threshold values such as 0.1th-percentile and 0.5th-percentile were included in the set of thresholds. On the other hand, APT-HT/LOF shows a different behavior pattern with APT-HT/IF and APTHT/AE. It is conjectured that when outlier scores do not well describe the outlier degree of data samples, it is not easy to determine thresholds from the outlier scores of normal training data samples.

5 Discussions

In this paper, we proposed an anomaly detection method in a data stream that utilizes outlier scores by a well-known outlier detection method. When a new data sample arrives, the outlier score for the data sample is computed by the outlier detection model which is constructed from normal training data. Then binary values obtained by applying multiple thresholds for the outlier score are added to the end of the binary-valued data stream corresponding to each threshold. The thresholds are computed from percentile values on the outlier scores of normal training data samples. The binomial distribution in a detection window that moves forward on a binary data stream is compared with the distribution on the reference window at the beginning part of the binary data stream. The predictions made in multiple binary data streams are combined to make a decision for anomaly pattern detection. If anomaly pattern occurrence is detected, an alarm signal is given. Otherwise, the process is repeated for the new incoming data sample.

Among the three outlier detection methods, IF, Autoencoder, and LOF, the anomaly pattern detection using IF, APD-HT/IF, demonstrated the best detection performance on average. It can detect the occurrence of anomalous events by focusing on the pattern where outliers are predicted rather than the

accuracy in outlier prediction for an individual data sample. As future work, we intend to apply the proposed anomaly detection method in a real application area such as breakdown detection in production facilities or new topic detection in a social network.

Acknowledgments

This work was supported by research fund of Chungnam National University.

References

[1] D. Hawkins. Identification of outliers. Springer Netherlands, 1980.

[2] C.H. Park. Outlier and anomaly pattern detection on data streams. The Journal of Supercomputing, 75:6118–6128, 2019.

[3] T. Kim and C.H. Park. Anomaly pattern detection for streaming data. Expert Systems with Applica tions, 149, 2020.

[4] F. Liu, K. Ting, and Z. Zhou. Isolation forest. In Proceedings of the 8th International Conference on Data Mining, 2008.

[5] Q. Feng, Y. Zhang, C. Li, Z. Dou, and J. Wang. Anomaly detection of spectrum in wireless communication via deep auto-encoders. The Journal of Supercomputing, 73(7):3161–3178, 2017.

[6] P. Remy. Anomaly detection in time setries using auto encoders. bolg positng from http://philipperemy.github.io/anomaly-detection.

[7] D. Pokrajac, A. Lazarevic, and L.J. Latecki. Incremental local outlier detection for data streams. In Proceedings of the CIDM, 2007.

[8] C. Aggarwal. Outlier analysis. Springer, 2017.

[9] D.Padilla, R. Brinkworth, and M. McDonnell. Performance of a hierarchical temporal memory net work in noisy sequence learning. In Proceedings of IEEE international conference on computational intelligence and cybernetics, 2013.

[10] S. Ahmad and S. Purdy. Real-time anomaly detection for streaming analytics, 2016. Retrieved from https://arxiv.org/pdf/1607.02480.pdf.

[11] W. Wong, A. Moore, G. Cooper, and M. Wagner. Rule-based anomaly pattern detection for detect ing disease outbreaks. In Proceedings of the 18th International Conference on Artificial Intelligence, 2002.

[12] K.Das, J. Schneider, and D. Neil. Anomaly pattern detection in categorical datasets. In Proceedings of KDD, 2008.

[13] F. et al. Pedregosa. Scikit-learn: Machine learning in python. Journal of Machine Learning Research, 12:2825–2830, 2011.

[14] M.M. Breunig, H-P. Kriegel, R.T. Ng, and J. Sander. Lof: Identifying density-based local outliers. In Proceedings of the 2000 ACM Sigmod International Conference on Management of Data, 2000.

[15] P. Tan, M. Steinbach, and V. Kumar. Introduction to data mining. Addison Wesley, Boston, 2006.

[16] S. Hawkins, H. Hongxing, G. Williams, and R. Baxter. Outlier detection using replicator neural networks. In Proceedings of the International Conference on Data Warehousing and Knowledge Discovery, 2002.

[17] A. Bife, G. Holmes, R. Kirkby, and B. Pfahringer. Moa: Massive online analysis. Journal of Machine Learning Research, 11:1601–1604, 2010.

[18] Y. Zhao, Z. Nasrullah, and Z. Li. Pyod: A python toolbox for scalable outlier detection. Journal of Machine Learning Research, 20(96):1–7, 2019.

Taegong Kim received his M.S. in Computer Science from Chungnam National University, Korea in 2020. He is currently working at Research Institute, SmartPro, Korea. His research interests include machine learning, deep learning, data mining.

Cheong Hee Park received her Ph.D. in Mathematics from Yonsei University, Korea in 1998. She received her M.S. and Ph.D. degrees in Computer Science from University of Minnesota, USA in 2002 and 2004, respectively. She has been on faculty at Department of Computer Science and Engineering, Chungnam National University, Korea since 2005, where currently she is a professor. Her research interests include machine learning, pattern recognition, data mining

ISSN 2083 - 2567

BROWSER FINGERPRINT CODING METHODS INCREASING THE EFFECTIVENESS OF USER IDENTIFICATION IN THE WEB TRAFFIC

Marcin Gabryel1,*, Konrad Grzanek2, Yoichi Hayashi3
1Department of Computer Engineering,
Czestochowa University of Technology,
al. Armii Krajowej 36, 42-200 Cze¸stochowa, Poland
2Information Technology Institute,
University of Social Sciences, 90- 113 Lodz
Clark University, Worcester, MA 01610, USA
3Department of Computer Science, Meiji University, Japan
*E-mail: marcin.gabryel@pcz.pl

ABSTRACT

Web-based browser fingerprint (or device fingerprint) is a tool used to identify and track user activity in web traffic. It is also used to identify computers that are abusing online advertising and also to prevent credit card fraud. A device fingerprint is created by extracting multiple parameter values from a browser API (e.g. operating system type or browser version). The acquired parameter values are then used to create a hash using the hash function. The disadvantage of using this method is too high susceptibility to small, normally occurring changes (e.g. when changing the browser version number or screen resolution). Minor changes in the input values generate a completely different fingerprint hash, making it impossible to find similar ones in the database. On the other hand, omitting these unstable values when creating a hash, significantly limits the ability of the fingerprint to distinguish between devices. This weak point is commonly exploited by fraudsters who knowingly evade this form of protection by deliberately changing the value of device parameters. The paper presents methods that significantly limit this type of activity. New algorithms for coding and comparing fingerprints are presented, in which the values of parameters with low stability and low entropy are especially taken into account. The fingerprint generation methods are based on popular Minhash, the LSH, and autoencoder methods. The effectiveness of coding and comparing each of the presented methods was also examined in comparison with the currently used hash generation method. Authentic data of the devices and browsers of users visiting 186 different websites were collected for the research.

Keywords: browser fingerprint, device fingerprint, LSH algorithm, autoencoder

Introduction

The process of identifying the users and their behavior on the Internet is rather commonplace. By collecting information about the user, the pages they visit, or their planned purchases, it is possible to develop a useful profile, for example, when communicating advertising content to them. The

mechanism for user identification is that of storing a unique identifier in a cookie on their device [1]. This type of method is widely criticized for the possibility of privacy violation. This problem has been noticed by the European Parliament and for several years now, information about the use of cookies has had to be placed on websites [2]. However, despite the obvious need for privacy, unambiguous identification of the device or the browser is extremely helpful in ensuring security and preventing abusive practices on the Internet. Numerous fraudulent activities include credit card payment scams [3], artificial generation of Internet traffic, the so-called abusive traffic [4], generating click fraud [5, 6] or automating the collection of web site contents [7]. Based on various reports [8, 9], losses due to fraud in online advertising alone can range from 6.5 to 19 billion dollars. Fraud prevention is primarily about identifying the perpetrator and blocking their actions. Unfortunately, the HTTP protocol used on the Internet only allows the user to be identified by the imperfect cookie mechanism. It is very easy to get around it by removing or ignoring cookies. Other methods, such as trying to eliminate fraud by blocking IP numbers of devices, do not give the expected results. One public IP number can be used by multiple users simultaneously. An IP number can also be assigned dynamically. Connections via proxy servers or VPN are also common. Hence, the best solution is to identify the user with the socalled browser fingerprint [10].

A browser fingerprint is a set of information about a given browser, device, operating system and environmental and location settings of the user [11]. Due to a great variety of this information, fingerprint can be treated as a unique identifier. Unfortunately, the values of the parameters collected may change over time. This happens when updating the browser or operating system version, resizing the screen, installing a new plugin, etc. Once a fingerprint has been obtained, on average, it remains valid for several days [12]. That is why developing a way of comparing fingerprints taking into account the changes occurring in them poses a rather demanding challenge. Using hash functions to encode and search for a fingerprint is no longer relevant, as hash functions generate completely different hash values for minor input changes, which is particularly undesirable in the case of fingerprinting. For example, once the browser version is changed, the same user will be recognized as two different persons.

The aim of this paper is to propose new methods of encoding fingerprint browser features and a method of comparing them, taking into account changes in the values of these features. The paper presents a research project showing the dynamics of changes taking place in the fingerprint during subsequent visits of a given user to the website. The features will be divided into a group of stable parameters and a group of unstable ones. This knowledge will be used to develop two new methods of comparing fingerprints:

– a new method of hash generation using the MinHash technique compatible with the LSH algorithm,

– adedicated neural network with an auto-encoder structure to encode the abovementioned two groups of fingerprint features.

The encoding effectiveness of both methods will be experimentally tested using the data obtained over a period of 3 months from 186 different websites.

This paper is organized as follows. Section 2 briefly discusses the related work highlighting the proposed methods: issues connected with the browser fingerprint, its practical applications and the possibilities offered by neural networks with the autoencoder structure for data encoding. Then, in Section 3, the definition of the browser fingerprint is given, the characteristics of the features selected for its creation are assessed and their changes over time are analyzed. Section 4 discusses the proposed algorithms for comparing browser fingerprints. The research tests showing their effectiveness are presented in Section 5. Section 6 concludes the paper and offers suggestions for future work.

2 Related works

In paper [13], it was noticed for the first time that there is a high likelihood of identifying the user through using various parameters extracted from the browser being used. However, the scale of the research at that time remained rather limited. Only in the next study [10], where the research was conducted on a larger number of users, was it shownthat fingerprinting can become a unique identifier. Works on browser or device fingerprinting usually contain comprehensive information on specific features comprising the fingerprint [3, 10, 11, 12, 14]. The studies also focus on the analysis of stability of particular parameters [3, 12], examining different types of browsers [11, 12] or applied security features limiting the possibility of obtaining fingerprint features [12]. There are also analyses of creating unique fingerprints over the years [15]. Despite many recent changes in browsers, browser fingerprinting is still effective in user identification. The possibilities of creating cross-browser fingerprints when one user uses several browsers on one device are also investigated [16].

The subject of practical use of browser fingerprinting appears in the literature in several contexts including web tracking [10, 14], bot and fraud prevention [7] and augmented authentication [3]. Computer security companies commonly use this technique to detect bots and unusual activity on websites [17, 18]. In [7], it is shown that fingerprinting is a good method of detecting crawler robots, but at the same time it can be bypassed with little effort. The paper [19] presents the capabilities offered by

browser fingerprinting that can be used in order to verify the software and hardware stack of a mobile or desktop client. The presented system can, for example, distinguish between traffic sent by an original smartphone running an original browser from an emulator or desktop client deceptively simulating the same configuration.

Anumberof publications address issues related to the abuse of online advertising and e-commerce. They mainly concern the problem of click frauds and credit card payments. Paper [4] proposes two novel inference techniques which can isolate click fraud attacks. One of them detects patterns of click reuse within an ad network clickstream and the second method, the bait-click defense, leverages the vantage point of an ad network to inject a pattern of bait clicks into a user's device. Further on, the authors in [20] deal with the problem of detecting Internet merchant fraud. Goods or services offered and sold at cheap rates, but never shipped is a simple example of this type of fraud. The authors suggest a framework to detect such fraudulent sellers with the help of the support vector machine approach.

Methods of detecting fraud on the Internet with the use of deep learning neural networks are also one of the subjects of many academic papers. In paper [21], the authors present a method of detecting credit card fraud. The main contribution of their work is the development of a fraud detection system that employs a deep learning architecture together with an advanced feature engineering process based on homogeneity-oriented behavior analysis. In [22], an ensemble neural network adapted as a hacking detection system to protect the computer system against cyber-attacks is presented. The presented ensemble neural network consists of an autoencoder, a deep belief neural network, a deep neural network and an extreme learning machine. The system's task is to monitor the activity within a network of connected computers so as to analyze the activity of intrusive patterns. In [23], an attempt was made to detect fraud in biometric systems. To detect this type of fraud, the authors propose a novel method for fingerprint spoofing detection using the Deep Boltzmann Machines (DBM) for the extraction of high-level features from images.

A special kind of neural networks are autoencoders. They make it possible to use their deepest layer to encode input data. An example here is the so-called semantic hashing published in [24], where this technique of efficient information retrieval is presented. A document is fed to the input of a neural network which generates a small binary vector. Two similar documents will have two identical or very similar hashes. By indexing a given document with this hash, it is possible to find other similar documents almost immediately– one only should calculate the hash of a given document and search for documents containing the same hash (or hashes that differ from each other by from one to two bits).

3 Generating and analyzing the browser fingerprint parameters

3.1 Definition

A browser or device fingerprint is a set of data related to the user device. It contains information about the hardware, operating system and browser, and its configuration [11]. The information is collected only directly from the browser of a user by Javascript and by a web server. The user remains unaware that they are being identified, as the use of browser fingerprinting leaves no trace.

For a quick search and comparison, the collected data set is given onto the input of a hash algorithm. The hash is an alphanumeric string of fixed length characters which becomes a unique identifier of a given browser [25]. This kind of fingerprint does not work in the case of mass-produced devices with limited configuration and upgrade possibilities, such as smartphones. In the case of one model of the device, the collected data is identical, generating the same fingerprints. Another problem is the instability of some features. For example, the software or operating system versions are regularly updated and then generate new fingerprints, too.

3.2 Parameters extracted from the browser

As mentioned in Subsection 3.1, the data necessary for the browser fingerprint are extracted directly from the browser. To cunduct the research under this paper the following features were selected:

– the features of the device (including device memory, color depth, logic cores, touch support, screen parameters, audio parameters)

– operating system parameters (i.e., its version, list of fonts, time zone)

– features of the browser (including its version, list of plug-ins, language list, User-agent header, adblock information, database information, Web Storage mechanisms, screen resolution available, window resolution available, Do Not Track header)

– graphics card information (canvas fingerprint, WebGLrenderer)

To calculate the level of identifying information in each of the fingerprint features mentioned above, the measure of entropy is used. The higher the entropy is, the more unique and identifiable a finger

print. Let H be the entropy, X- a discrete random variable with possible values $x_1,...,x_n$ and $P(X)$a probability mass function. The entropy follows this formula

$$H(X) = -\sum_i P(x_i) \log_b P(x_i). \qquad (1)$$

For $b = 2$ it is the Shannon entropy and the result is in bits. The fingerprint's features, along with the calculated entropy, are presented in Table 1. The data came from 131,326 users who made 365,209 visits to different websites over a period of three months. The table does not include parameters with the entropy below 0.1. Some parameters, such as screen_id and User-agent , were broken down into individual elements. For screen_id it is width , height , available_width and available_height . In the case of User-agent the whole sequence was divided into elements starting with prefix ua_ .

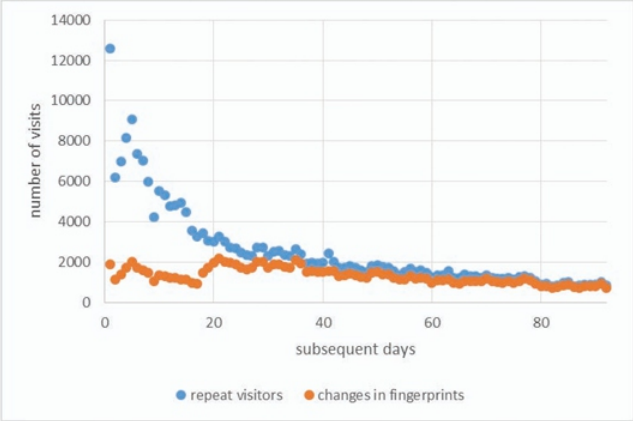


Figure 1. Changes in the browser fingerprints recorded daily in repeat visitors.

The same dataset was used to investigate the changes in the values of browser fingerprints. Figure 3.2 presents a graph showing the number of times that the website was accessed again over the following days. The graph also shows the number of users with a change in the value of at least one fingerprint feature. The graph shows that on the second day about 12,000 users returned to the website, and amongthemasmanyas2,000showedachange in at least one feature. The last day of the research recorded a return of approximately 1,000 users who had accessed the website on the first day of the project. All the repeat visitors displayed changes in at least one fingerprint feature. Figure 3.2 shows the percentage of visitors returning to the website over the following days. It also shows the percentage of users visiting the website on that day when each of them had a change in at least one feature. It can be seen on day 20 that 56% of the repeat visitors had already had a change of at least one feature. After 40 days, the changes had occurred in 80% of the investigated users. Table 1 shows for each fea ture the percentage of the users who had shown the changes. This means that the data can be divided into a stable data group and an unstable data group.

Table1. Obtainabledevicefingerprintfeatures

Feature	No. of bits of entropy	No. of changes in %	Group
device_memory	1.66	0.0	1
do_not_track_val_id	0.21	0.1	1
fonts	1.36	0.4	1
audio_params_id	0.50	1.2	2
webgl_vendor_id	1.66	0.0	1
webgl_renderer_id	5.35	0.1	1
logic_cores	1.18	0.0	1
platform	1.48	0.0	1
timezone	0.15	0.0	1
app_version	10.97	64.6	2
touch_enabled	0.68	0.0	1
max_touch_points	0.84	0.0	1
screen_id	4.78	1.7	-
width	2.60	1.2	2
height	4.32	1.1	2
av_width	2.63	1.4	2
av_height	4.58	1.4	2
adblock_enabled	0.26	0.6	1
canvas_2d_fingerprint	5.62	12.9	2
browser_plugins_hash	0.88	0.0	1
user-agent	10.98	64.7	-
br_version	4.26	62.9	2
os_version	2.8	4.9	2
app_version	10.97	64.4	2
platform	1.48	0.0	1
ua_device_brand_name	2.56	0.0	1
ua_device_model	6.51	0.0	1
ua_client_name	1.87	0.0	1
ua_client_version	4.76	63.3	2
ua_client_type	0.39	0.0	1
ua_device_type	1.08	0.0	1
ua_device_brand	2.56	0.0	1
ua_device_code	6.64	0.0	1
ua_os_name	0.78	0.0	1
ua_os_version	2.90	5.1	2
ua_preferred_client_name	2.32	0.2	1
ua_preferred_client_version	5.05	59.3	2
ua_preferred_client_type	0.31	0.2	1

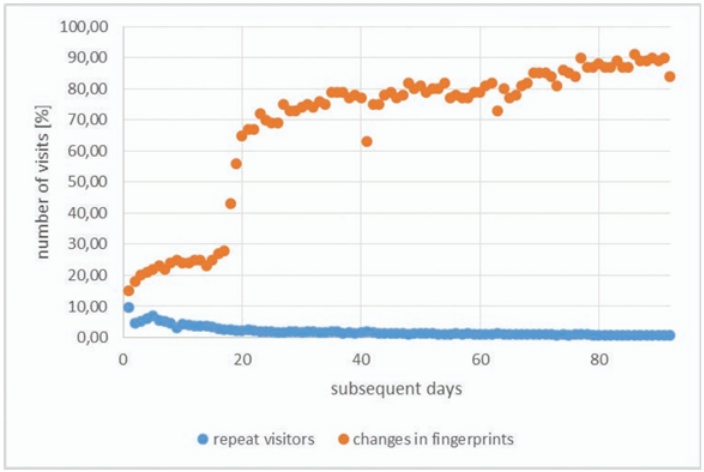


Figure2. Percentage of changes in the browser fingerprints recorded daily in repeat visitors.

4 Browser fingerprint encoding methods

4.1 Hashing

Hashing refers to the process of generating a fixed-sizeoutput from an input of a variable size. This is done through the use of mathematical formulas known as hash functions (implemented as hashing algorithms).A conventional hash function should have collision resistance. This is a feature that prevents the same hash value from being obtained from two different input sets. These two properties of the hash function, a fixed hash length and collision resistance, enable a quick search of large data sets. Instead of comparing dozens of parameters for identical values, only hash values are compared with each other. The introduction outlines the disad vantages of using this solution for searching and comparing browser fingerprints. In the following Subsections, new methods for obtaining hashes and their use in searching for similar browser fingerprints are proposed.

4.2 The LSH algorithm

The main task of the Locality-Sensitive Hashing (LSH) algorithm is to quickly compare documents in terms of their contents. The documents do not need to be identical, as it is in the case of the hash function, because the proposed method is not resistant to minor changes in the document. The LSH algorithm consists of three steps:

– transforming the document into a set of characters of length *k* (the shingling method, also known as *k*-

-shingles or *k*-grams method),

–compressing the shingles set using the "MinHashing"method, so that the similarity of the base sets of documents in their compressed versions can still be checked,

–the LSH algorithm, which allows us to find the most similar pairs of documents or all pairs that are above some lower bound in similarity.

Shingling is an effective method of representing a document as a set. To generate the set, we need to select short phrases or sentences from the document–the so-called shingles. This causes documents to have many common elements in their sets even if the sentences appear in documents in a different order.

The next step consists in creating the so-called characteristic matrix, where the columns contain sets of shingles of individual documents, and the consecutive lines correspond to individual shingles. In the matrix cells at the intersection of row I and column j, there is value 1 in the case of the i-th shingle in the j-th document.

In the MinHash algorithm a so-called SIG signature matrix is created with the dimensions m×n, where each of the m documents corresponds to n signatures. The matrix is calculated by performing random and independent n permutations of m rows of the characteristic matrix. The MinHash value for the column of the j-th document is the number of the first row (in the order resulting from the permutations), for which this column has value 1. These calculations are time-consuming, therefore instead of selecting random n row permutations, random n hash functions h1,h2,...hn are selected. The signature matrix is built taking into account each row in the given order. Let SIGk,j be an element of the signature matrix for the k-th hash function and column j of document dj. The next steps in generating the signatures matrix are shown in Algorithm 1.

Algorithm 1 Algorithm for generating the signatures matrix.

1. Initially, set $SIG_{k,j}$ to ∞ for all values of k and j.
2. For each i-th row of the characteristic matrix repeat points 2 and 3.
3. Calculate $h_1(j), h_2(j), \ldots, h_n(j)$.

4. For each column j, check if there is 1 in row i. If yes, then for each $k = 1, 2, \ldots, n$, set $SIG_{k,j} = \min(SIG_{k,j}, h_k(j))$.

The idea of the LSH algorithm allows checking the similarity of two elements. As a result of its operation, information is returned whether the pair forms a so-called "candidate pair", i.e. whether their similarity is greater than a specified threshold t (similarity threshold). Any pair that hashed to the same bucket is considered as a “candidate pair”. Dissimilar pairs that do hash to the same bucket are false positives. On the other hand, those pairs, which despite being similar, do not hash to the samebucket under at least one of the hash functions, are false negatives. A detailed description of particular parts of the LSH algorithm can be found in many works including [26] and [27].

In the proposed algorithm the *i*-th fingerprint parameters fi need to be divided into the stable fsi and unstable ones *fni* (see Section 3.2) and *fi* ={*fsi, fni*}. A MinHash algorithm starts operating for each set of parameters, which will generate two signature matrixes SIG*sk,j* and SIG*nk,j*. The algorithm starts the search process twice, i.e. for the stable and unstable parameters. The following steps of the algorithm are presented in Algorithm 2. Its operation results in returning a set of fingerprints *fqn* similar to fingerprint *fq*.

Algorithm 2 The LSH algorithm for searching for similar browser fingerprints in relation to parameter stability.

Initial procedure:

1. Prepare two groups of parameters: stable ones $f_s = \{f_{s1}, \ldots, f_{sm}\}$ and unstable ones $f_n = \{f_{n1}, \ldots, f_{nm}\}$, where m – number of fingerprints.

2. According to Algorithm 1 determine two signature matrices $SIGs_{k,j_s}$ and $SIGn_{k,j_n}$ (for stable and unstable parameters, respectively) for two sets d_s and d_n, $k = 1, \ldots, m$, $j_s = 1, \ldots, n_s$, $j_n = 1, \ldots, n_n$, n_s, n_n – number of signatures for stable and unstable parameters.

3. Determine the similarity thresholds t_s and t_n for stable and unstable parameters, respectively.

To find fingerprints f_{qn} similar to fingerprint f_q the following steps need to be carried out:

1. Start the LSH algorithm using the stable parameters f_s to find similar candidate pairs f_{qs}

$$f_{qs} = \mathrm{LSH}(SIGs(f_q), SIGs(f_s), t_s),$$

where: $SIGs(f_q)$, $SIGs(f_s)$ – signature matrix values for stable parameters obtained for the parameters of fingerprints f_q and f_s.

2. Having found fingerprints f_{qs} do one more search for a similar fingerprint, but this time using unstable parameters f_n

$$f_{qn} = \mathrm{LSH}(SIGn(f_q), SIGn(f_{qs}), t_n),$$

where: $SIGn(f_q)$, $SIGn(f_{qs})$ – signature matrix values for unstable parameters obtained for the parameters of fingerprints f_q and f_n.

3. Return obtained similar fingerprints f_qn.

4.3 Deeplearning methods– autoencoders

An autoencoder is a neural network with at least one hidden layer. The input and output have the same size. The autoencoder is trained in such a way that the values given onto its input are to be copied onto its output. The encoder aims to compress data to a low-dimensional representation, while the decoder aims to reconstruct the input data from the low-dimensional representation generated by the encoder [28].

The learning set is X = {x1,x2,...,xN} ∈Rm where xi is the m-dimensional feature vector, Nthe number of samples. The encoder maps input vector xi onto the hidden representation hi ∈ Rn using function fθ as in

$$s(t) = \frac{1}{1 + \exp^{-t}}. \qquad (3)$$

where W ∈Rm×n is a set of weights, n is the number of units in the hidden layer h, b ∈ Rn is a bias vector, θ is set {W,b}, and s(·) is the adopted activation function (sigmoid function) determined by the following formula

$$s(t) = \frac{1}{1 + \exp^{-t}}. \qquad (3)$$

The decoder maps back values hi obtained in the hidden layer onto the output vector yi ∈ Rm according to the following formula

$$y_i = g_{\dot{\theta}}(x_i) = s(\dot{W} h_i + \dot{b}), \qquad (4)$$

where ˙W ∈ Rn× m are weights, ˙ b ∈ Rm is a bias vector and ˙ θ = { ˙W, ˙ b}.

Thelearning of the autoencoder consists in minimizing the difference between input xi and output yi. To this end a loss function is calculated as shown in the following formula

$$L(x_i, y_i) = \|x_i - y_i\|^2 = \|x_i - s(\dot{W} s(W x_i + b) + \dot{b})\|^2. \qquad (5)$$

The aim of the learning is thus finding optimum values of parameters θ and ˙ θ facilitating the minimizing of the error between the input and output for thewhole training set

$$\theta, \dot{\theta} = \arg\max_{\theta, \dot{\theta}} L(x, y). \qquad (6)$$

The autoencoder presented above is discussed in relation to continuous data x. In the case of categorical data, one-hot encoded data are given into the input. In the case of one variable x of the categorical type, input dimension m is equal to the number of categories. Each category is numbered with consecutive natural numbers. Single vector xi is filled with zeros and contains a single value 1 in place j, where j is the category number. For this type of data function s(·) in formula (3) will take the value of softmax [29], where for each of the outputs k

$$s(t)_k = \frac{e^{t_k}}{\sum_{j=1}^{m} e^{t_j}}, \qquad (7)$$

where t ∈Rm, andtj is j-th element of vector t. For the softmax function, the loss function L(xi,yi) is calculated by using the categorical cross-entropy

$$L(x_i, y_i) = -\sum_{j=1}^{m} x_{ij} \log(y_{ij}), \qquad (8)$$

where m is the number of outputs (number of categories).

For encoding stable and unstable fingerprint features, there are two hashes required. The autoencoder

will, therefore, consist of two pairs of encoder-decoders for stable and unstable parameters, respectively. The autoencoder will thus have two hidden layers, h1 and h2, whose outputs will return the values of the fingerprint hashes given onto the network input. The diagram of such an autoencoder is shown in Figure 3.

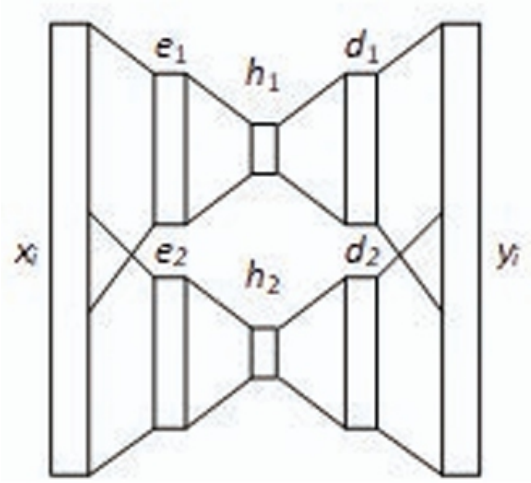


Figure 3. Dedicated structure of autoencoders.

There is a training set xi ∈ x1j,x2k ∈ Rm, where x1j ∈ Rm1, x2k ∈ Rm2– stable and unstable inputdata, m1– the number of stable data, m2– the number of unstable data, m = m1 +m2. The values of the hidden layers will then be

$$y_{1j} = g_{\dot{\theta}_1}(x_{1j}) = s(\dot{W}_1 h_{1j} + \dot{b}_1) \qquad (11)$$

and

$$y_{2k} = g_{\dot{\theta}_2}(x_{2k}) = s(\dot{W}_2 h_{2j} + \dot{b}_2). \qquad (12)$$

where: ˙W1 ∈ Rn1× m1, ˙W2 ∈ Rn2× m2 are the weights, b1 ∈ Rm1, ˙ b2 ∈ Rm2 are the bias vectors and ˙ θ1 = { ˙W1, ˙ b1} and ˙ θ2 = { ˙W2, ˙ b2}. For this autoencoder the loss function will be expressed by the following formula

$$\begin{aligned} L(x_i, y_i) &= \frac{L(x_{1i}, y_{1i}) + L(x_{2i}, y_{2i})}{2} \\ &= \frac{\|x_{1i} - y_{1i}\|^2 + \|x_{2i} - y_{2i}\|^2}{2} \\ &= \frac{\|x_{1i} - s(\dot{W}_1 h_{1j} + \dot{b}_1)\|^2 + \|x_{2i} - s(\dot{W}_2 h_{2j} + \dot{b}_2)\|^2}{2}. \end{aligned} \qquad (13)$$

After the learning process has been completed, only encoders are used for further research. They are treated as hash functions. The inputs of the encoders are given fingerprint feature values (as onehot encode data). The output values obtained by the two layers h1 and h2 are rounded to integers.

5 Experiments

For this research on browser fingerprints coding methods, authentic data were collected during visits to 186 different types of websites including online shops, companies offering various services and financial institutions, including banks. For 3 months, the total number of registered clicks totaled ca. 45 million. For the sake of the research, the data for which the fingerprints parameters changed5 to 9times in one user within the three months were selected. The number of the data filtered in this way amounted to ca. 213,000. About 33,000 unique users were identified. The data containing fingerprints were then divided in the 80/20 ratio into training and test groups. Most of the data were used to initiate the MinHash algorithm or to learn the neural network. The fingerprints from the test group were used to check the effectiveness of the search.

Precision and recall were used to evaluate the effectiveness of the tests [30]. Precision is the ratio of the number of correctly classified data to the total number of irrelevant and relevant data classified

$$precision = \frac{tp}{tp + fp}, \qquad (14)$$

and recall is the ratio between the number of data that are correctly classified to the total number of positive data

$$recall = \frac{tp}{tp + fn}, \qquad (15)$$

where tp– true positive, fp– false positive, fnfalse negative and they can be derived from a confusion matrix [30]. The parameter which combines the above two parameters is F1 score that it is the harmonic mean of precision and recall

$$F1 \text{ score} = \frac{2 \cdot precision \cdot recall}{precision + recall}. \qquad (16)$$

The first of the conducted experiments consists in testing the effectiveness of Algorithm 2. The initiating part of this algorithm requires that the data that constitute the browser fingerprint are divided into two groups, i.e. stable and unstable (according to the results obtained in Section 3.2). The initial values of the algorithm parameters must then be determined: the number of signatures for encoding ns, nn and threshold values ts, tn. The selection of these values requires a number of tests with different combinations of these parameters. The following values have been adopted for each parameter of the algorithm: $n_s,\ n_n \in \{2,4,8,16,24,32,64,96,128\},\ \ t_s = 1,\ t_n \in \{1,0.9,0.8,0.7,0.6,0.5\}.$

The best values were obtained for ns = 128. Table 2 presents a summary of the obtained results. The following rows show the results for different tn values, while the columns show the results for different nn values. The best was a set of parameters ns = 128, nn = 4, ts = 1 and tn = 0.5 lub tn = 0.6. Here the value F1=0.35forprecision=0.34andrecall=0.36 wasobtained.

Table2. The F1 score obtained for ns=128 and ts=1.

	n_n				
t_n	2	4	8	16	24
1	0.298	0.297	0.281	0.286	0.287
0.9	0.298	0.297	0.281	0.285	0.279
0.8	0.298	0.297	0.282	0.278	0.276
0.7	0.298	0.297	0.289	0.284	0.286
0.6	0.298	**0.350**	0.317	0.293	0.300
0.5	0.321	**0.350**	0.345	0.327	0.334
t_n	32	48	64	96	128
1	0.289	0.290	0.290	0.290	0.290
0.9	0.283	0.282	0.284	0.283	0.283
0.8	0.276	0.275	0.275	0.274	0.274
0.7	0.285	0.277	0.273	0.272	0.270
0.6	0.285	0.285	0.281	0.279	0.273
0.5	0.305	0.305	0.306	0.285	0.287

The next experiment concerned the use of aneural network with a dedicated structure consisting of two autoencoders (see Section4.3). Severalpossible combinations of neural network hyperpa rameters were tested. Each time a different number of encoder and decoder layers, units in each layer and the size of the deepest coding layers h1and h2. were selected. The optimization parameters were assumed as follows: categorical cross-entropyas the loss function, stochastic gradient descent optimizer, number of epochs–250 and batch size 32. The F1 score values for different network structures are presented in Table3.The subsequent rows show the examined structures of neural networks, specified number of units for encoding layers h1and h2,and the F1score value obtained for the test data. The best results were achieved by network No. 4 considering the obtained F1 score and network size.

Ta3ble.The F1score results obtained for
different structures of autoencoders.

No.	Type of the parameter group	Structure	No. of units of layer h	$F1$ score
1	1 2	512-256-64-256-512 512-256-4-256-512	64 4	0.327
2	1 2	512-256-32-256-512 512-256-8-256-512	64 4	0.303
3	1 2	640-368-128-368-640 256-128-8-128-256	128 8	0.361
4	1 2	640-368-96-368-640 256-128-8-128-256	96 8	0.377
5	1 2	256-128-32-128-256 256-128-8-128-256	32 8	0.298

The best results of the two coding and finger print benchmarking methods presented in this pa per are presented inTable4. The results are compared with the commonly used hashing method (see Section 4.1).Two experiments using hash function SHA1 were conducted. In the first case, stable and unstable features were given onto the hash function input. In the second case only stable features were given. The new methods proposed in the paper give much better results than commonly used hashing methods.

Table4. Comparison of the best results obtained.

Method	The best $F1$ score
hashing stable and unstable parameters (see 4.1)	0.276
hashing stable parameters	0.202
MinHash algorithms with LSH (see 4.2)	0.350
autoencoders (see 4.3)	0.377

6 Conclusion

The paper presents two new methods ofencoding and comparing browser fingerprints: an algorithm using MinHash encoding(cooperating) withthe LSH and a dedicated structure of aneural network consisting of two encoders. Both methods use the results of a previously conducted analysis of changes in fingerprint characteristics over time. This allowed selecting two groups of features: stable and unstable ones. The presented experimental results showed that the effectiveness of searching for similar fingerprints is much greater if the difference between these two groups is taken into account when encoding.

A great advantage which the proposed methods offer is the possibility to easily save the results of encoding in the database. This is possible both with the hashes obtained by MinHash algorithms and with the hashes obtained by the h1and h2 autoencoder layers. This gives the possibility to put

both algorithms into practice. Both methods can also be used when creating device fingerprints for different browsers used by one user [16]. In this case, it will be necessary to perform an appropriate analysis of the changes occurring in the fingerprint feature values beforehand and create appropriate groups of stable and unstable features.

Browser fingerprinting is a tool that helps toreduce the number of fraudulent activities on the In ternet. The new algorithms proposed in this paper may increase the level of detecting online fraudulent activities being carried out by some users by identifying them more accurately and efficiently.

References

[1] Kristol D.M., HTTP cookies: Standards, privacy, and politics, ACM Trans. Internet Techn. 1 (2) (2001) 151–198.

[2] Low C., Cookie law explained, 2016. on-line https://www.cookielaw.org/the-cookie-law/ (re trieved:03/2020).

[3] Alaca, F., Van Oorschot, P. C. (2016, December). Device fingerprinting for augmenting web authen tication: classification and analysis of methods. In Proceedings of the 32nd Annual Conference on Computer Security Applications (pp. 289-301).

[4] Nagaraja, S., Shah, R. (2019, May). Clicktok: click fraud detection using traffic analysis. In Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks (pp. 105 116).

[5] Mouawi, R., Elhajj, I.H., Chehab, A. et al. Crowdsourcing for click fraud detection. EURASIP J. on Info. Security 2019, 11 (2019)

[6] Dave, V., Guha, S., Zhang, Y. (2012, August). Measuring and fingerprinting click-spam in ad networks. In Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, ar chitectures, and protocols for computer communication (pp. 175-186).

[7] Vastel, A., Rudametkin, W., Rouvoy, R., Blanc, X. (2020, February). FP-Crawlers: Studying the Resilience of Browser Fingerprinting to Block Crawlers. In NDSS Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb'20).

[8] 2019. https://www.emarketer.com/content/digitalad-fraud-2019

[9] Barker S. , "Future Digital Advertising, Artificial Intelligence & Advertising Fraud 2019-2023", Juniper Research, 2019

[10] Eckersley P., How unique is your web browser? in: Privacy Enhancing Technologies, 10th International Symposium, PETS 2010, Berlin, Germany, July 21-23, 2010. Proceedings, 2010, pp. 1–18

[11] Laperdrix, P., Bielova, N., Baudry, B., Avoine, G. (2019). Browser Fingerprinting: A survey. arXiv preprint arXiv:1905.01051.

[12] Kobusinska, A., Pawluczuk, K., Brzezinski, J. (2018). Big Data fingerprinting information analytics for sustainability. Future Generation Computer Systems, 86, 1321-1337.

[13] Mayer J R. 2009. Any person... a pamphleteer": Internet Anonymity in the Age of Web 2.0. Undergraduate Senior Thesis, Princeton University (2009).

[14] Steven E. and Arvind N. 2016. Online Tracking: A 1-million-site Measurement and Analysis. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16). ACM, New York, NY, USA, 1388–1401.

[15] Gómez-Boix, A., Laperdrix, P., Baudry, B. (2018, April). Hiding in the crowd: an analysis of the effectiveness of browser fingerprinting at large scale. In Proceedings of the 2018 world wide web conference (pp. 309-318).

[16] Cao, Y., Li, S., Wijmans, E. (2017, March). (Cross) Browser Fingerprinting via OS and Hardware Level Features. In NDSS.

[17] 2020. The Evolution of Hi-Def Fingerprinting in Bot Mitigation- Distil Networks. https://resources.distilnetworks.com/all-blogposts/device-fingerprinting-solution-botmitigation

[18] 2020. Device Tracking Add-on for minFraud Services -MaxMind https://dev.maxmind.com/minfraud/device/

[19] Bursztein, E., Malyshev, A., Pietraszek, T., Thomas, K. (2016, October). Picasso: Lightweight device class fingerprinting for web clients. In Proceedings of the 6th Workshop on Security and Privacy in Smartphones and Mobile Devices (pp. 93102).

[20] Renjith, S. (2018). Detection of Fraudulent Sellers in Online Marketplaces using Support Vector Machine Approach. arXiv preprint arXiv:1805.00464.

[21] Zhang, X., Han, Y., Xu, W., Wang, Q. (2019). HOBA: A novel feature engineering methodology for credit card fraud detection with a deep learning architecture. Information Sciences.

[22] Ludwig, S. A. (2019). Applying a neural network ensemble to intrusion detection. Journal of Arti ficial Intelligence and Soft Computing Research, 9(3), 177-188.

[23] de Souza, G. B., da Silva Santos, D. F., Pires, R. G., Marana, A. N., Papa, J. P. (2019). Deep features extraction for robust fingerprint spoofing attack detection. Journal of Artificial Intelligence and Soft Computing Research, 9(1), 41-49

[24] Salakhutdinov, R., Hinton, G. (2009). Semantic hashing. International Journal of Approximate Reasoning, 50(7), 969-978.

[25] 2020. FingerprintJS. Fraud detection API. https://fingerprintjs.com/

[26] Leskovec J., Rajaraman A., Ullman J.D.: Mining of Massive Datasets, Cambridge University Press, 2014

[27] Azgomi, H., Mahjur, A. (2013). A Solution for Calculating the False Positive and False Negative in LSH Method to Find Similar Documents. Journal of Basic and Applied Research, 3, 466-472.

[28] Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016). Deep Learning. MIT Press

[29] Bengio Y., Learning deep architectures for ai Found. Trends Mach. Learn., vol. 2, no. 1, pp. 1127, Jan. 2009.

[30] Olson, D.L., Delen, D.: Advanced Data Mining Techniques, 1st edn. Springer, Heidelberg (2008).

Marcin Gabryel earned his Ph.D. degree in computer science at Czestochowa University of Technology, Poland, in 2007. He is an assistant professor in the Department of Computer Engineering at Częstochowa University of Technology. His research focuses on developing new methods in computational intelligence and data mining. He has published over 50 research papers. His present research interests include deep learning architectures and their applications in databases and security.

Konrad Grzanek, scientist, programmer and lecturer. Graduate of the Technical University of Łódź (FTIMS). Assistant professor at the Social Academy of Sciences. He holds a Ph.D. from Częstochowa University of Technology (CUT). His research interests focus on programming languages, software quality, software development processes, and artificial intelligence, in particular on combining machine learning methods with static software analysis. As a programmer, he is an advocate and promoter of the functional programming style. Author of over 30 publications related to various problems of computer science and software engineering.

Prof. Yoichi Hayashi received the Dr. Eng. degree in systems engineering from Tokyo University of Science, Tokyo in 1984. In 1986, he joined the Computer Science Department of Ibaraki University, Japan, as an Assistant Professor. Since 1996, he has been a Full Professor at Computer Science Department, Meiji University, Tokyo. He was a visiting professor at the University of Alabama at Birmingham and University of Canterbury (New Zealand). He authored over 230 published computer science papers. His current research interests include explainable AI, deep learning, rule extraction, high-performance classifiers, and medical informatics and medical imaging. He has been the Action Editor of Neural Networks and the Associate Editor of IEEE Trans. Fuzzy Systems. He has served as Editor-in-Chief and Associate Editor, Editorial Board Member, Review Board Member, Guest Editor and Reviewer in 60 academic journals, and was involved in the work for the European Research Council Executive Agency, National Sciences and Engineering Research Council of Canada (NSERC). He has been a senior member of the IEEE since 2000.

Instructions for Authors

Essentials for Publishing in this Journal

1 Submitted articles should not have been previously published or be currently under consideration for publication elsewhere.

2 Conference papers may only be submitted if the paper has been completely re-written (taken to mean more than 50%) and the author has cleared any necessary permission with the copyright owner if it has been previously copyrighted.

3 All our articles are refereed through a double-blind process.

4 All authors must declare they have read and agreed to the content of the submitted article and must sign a declaration correspond to the originality of the article.

Submission Process

All articles for this journal must be submitted using our online submissions system. http://enrichedpub.com/ . Please use the Submit Your Article link in the Author Service area.

Manuscript Guidelines

The instructions to authors about the article preparation for publication in the Manuscripts are submitted online, through the e-Ur (Electronic editing) system, developed by **Enriched Publications Pvt. Ltd**. The article should contain the abstract with keywords, introduction, body, conclusion, references and the summary in English language (without heading and subheading enumeration). The article length should not exceed 16 pages of A4 paper format.

Title

The title should be informative. It is in both Journal's and author's best interest to use terms suitable. For indexing and word search. If there are no such terms in the title, the author is strongly advised to add a subtitle. The title should be given in English as well. The titles precede the abstract and the summary in an appropriate language.

Letterhead Title

The letterhead title is given at a top of each page for easier identification of article copies in an Electronic form in particular. It contains the author's surname and first name initial .article title, journal title and collation (year, volume, and issue, first and last page). The journal and article titles can be given in a shortened form.

Author's Name

Full name(s) of author(s) should be used. It is advisable to give the middle initial. Names are given in their original form.

Contact Details

The postal address or the e-mail address of the author (usually of the first one if there are more Authors) is given in the footnote at the bottom of the first page.

Type of Articles

Classification of articles is a duty of the editorial staff and is of special importance. Referees and the members of the editorial staff, or section editors, can propose a category, but the editor-in-chief has the sole responsibility for their classification. Journal articles are classified as follows:

Scientific articles:

1. Original scientific paper (giving the previously unpublished results of the author's own research based on management methods).

2. Survey paper (giving an original, detailed and critical view of a research problem or an area to which the author has made a contribution visible through his self-citation);

3. Short or preliminary communication (original management paper of full format but of a smaller extent or of a preliminary character);

4. Scientific critique or forum (discussion on a particular scientific topic, based exclusively on management argumentation) and commentaries. Exceptionally, in particular areas, a scientific paper in the Journal can be in a form of a monograph or a critical edition of scientific data (historical, archival, lexicographic, bibliographic, data survey, etc.) which were unknown or hardly accessible for scientific research.

Professional articles:

1. Professional paper (contribution offering experience useful for improvement of professional practice but not necessarily based on scientific methods);
2. Informative contribution (editorial, commentary, etc.);
3. Review (of a book, software, case study, scientific event, etc.)

Language

The article should be in English. The grammar and style of the article should be of good quality. The systematized text should be without abbreviations (except standard ones). All measurements must be in SI units. The sequence of formulae is denoted in Arabic numerals in parentheses on the right-hand side.

Abstract and Summary

An abstract is a concise informative presentation of the article content for fast and accurate Evaluation of its relevance. It is both in the Editorial Office's and the author's best interest for an abstract to contain terms often used for indexing and article search. The abstract describes the purpose of the study and the methods, outlines the findings and state the conclusions. A 100- to 250-Word abstract should be placed between the title and the keywords with the body text to follow. Besides an abstract are advised to have a summary in English, at the end of the article, after the Reference list. The summary should be structured and long up to 1/10 of the article length (it is more extensive than the abstract).

Keywords

Keywords are terms or phrases showing adequately the article content for indexing and search purposes. They should be allocated heaving in mind widely accepted international sources (index, dictionary or thesaurus), such as the Web of Science keyword list for science in general. The higher their usage frequency is the better. Up to 10 keywords immediately follow the abstract and the summary, in respective languages.

Acknowledgements

The name and the number of the project or programmed within which the article was realized is given in a separate note at the bottom of the first page together with the name of the institution which financially supported the project or programmed.

Tables and Illustrations

All the captions should be in the original language as well as in English, together with the texts in illustrations if possible. Tables are typed in the same style as the text and are denoted by numerals at the top. Photographs and drawings, placed appropriately in the text, should be clear, precise and suitable for reproduction. Drawings should be created in Word or Corel.

Citation in the Text

Citation in the text must be uniform. When citing references in the text, use the reference number set in square brackets from the Reference list at the end of the article.

Footnotes

Footnotes are given at the bottom of the page with the text they refer to. They can contain less relevant details, additional explanations or used sources (e.g. scientific material, manuals). They cannot replace the cited literature.
The article should be accompanied with a cover letter with the information about the author(s): surname, middle initial, first name, and citizen personal number, rank, title, e-mail address, and affiliation address, home address including municipality, phone number in the office and at home (or a mobile phone number). The cover letter should state the type of the article and tell which illustrations are original and which are not.