Ensemble learning using frequent itemset mining for anomaly detection

Saeid Soheily-Khah and Yiming Wu

SKYLADS Research Team, Paris, France
{saeid,yiming}@skylads.com

Abstract

Anomaly detection is vital for automated data analysis, with specific applications spanning almost every domain. In this paper, we propose a hybrid supervised learning of anomaly detection using frequent itemset mining and random forest with an ensemble probabilistic voting method, which outperforms the alternative supervised learning methods through the commonly used measures for anomaly detection: accuracy, true positive rate (i.e. recall) and false positive rate. To justify our claim, a benchmark dataset is used to evaluate the efficiency of the proposed approach, where the results illustrate its benefits.

Keywords

Ensemble learning, anomaly detection, frequent (closed / maximal) itemset mining, random forest, classification

1. Introduction

Anomaly detection, also known as 'outlier' detection [1], is a technique used to identify unusual or abnormal patterns that do not conform to the expected behaviors [2]. It has a wide variety of applications in business, from security intrusion detection to system health monitoring, and from fraud detection in credit card transactions or (online) ads clicks to fault detection in operating environments, military surveillance and etc [3, 4, 5, 6, 7]

Detecting anomalies (or outliers) has been studied in statistics community as early as in the 19th century [8], and it was proposed for Intrusion Detection Systems (IDS) in 1980s [9]. Over time, in the recent years, anomaly detection has attracted great attention in the machine learning and data mining community [10, 11, 12]. Anomaly detection works by taking the baseline of normal traffic and activities, from which a model of normal behaviors is built. It detects known and previously unknown attacks. However, in many cases, it may fail to detect malicious behaviours or even raise alarms for normal data assuming erroneously that it is an attack. Thus, applying data mining techniques on network traffic data is a promising solution which helps to develop better anomaly detection systems. Several anomaly detection techniques have been proposed in the literature, such as density-based techniques (k-nearest neighbor [13] or local outlier factor[14]), correlation-based outlier detection [15], one-class support vector machines [16], neural networks, bayesian networks, hidden markov models [17], fuzzy logic-based outlier detection and etc. In this paper, we discuss the wel-known machine learning methods for network anomaly detection and propose a hybrid supervised learning approach to detect anomalies in networks more effectively. The main contribution of this work is actually to boost base (weak) learners to strong learners by ensemble learning, which can make very accurate classifiers.

The remainder of the paper is organized as follows: Section 2 provides the state-of-the-art of the work. Next, in Section 3, we characterize the proposed ensemble learning approach

Natarajan Meghanathan et al. (Eds) : CCSEA, CLOUD, SIPRO, AIFU, SEA, DKMP, NCOM - 2019
pp. 373-389, 2019. © CS & IT-CSCP 2019 DOI: 10.5121/csit.2019.90931

for anomaly detection in detail. Section 4 presents the experimental results, and lastly Section 5 concludes the paper.

2. State-of-the-art

In this section we first present an overview of the learning approach used in this paper, i.e. the ensemble learning approach. Moreover, we describe the data mining, machine learning and pattern mining algorithms used in this research work such as Support Vector Machine (SVM), k-Nearest Neighbor (k-NN), Multi Layer Perceptron (MLP), and Random Forest (RF). In addition, we simply mention their computational complexities, advantages and disadvantages.

2.1. Ensemble Learning

Ensemble learning is a statistic and machine learning approach that uses multiple learning algorithms to solve a problem with better predictive performance. Unlike traditional machine learning approaches in which a single hypothesis is learned from training data, ensemble approaches attempt to build a set of hypotheses and combine them to build a new hypothesis [18].

Ensemble learning systems can be useful to deal with the big data. When the size of training data is too large to make a single classifier, the data can be partitioned into smaller subsets by different strategies, and each subset can be used to train a separate classifier. Then the different classifiers can be combined using an appropriate combination rule. On the other hand, while the data is not that big, several base learners are built on the whole data, and then these learners are combined through several combination techniques such as majority voting [19].

In a simple way, the main objective of an ensemble learning is to improve the performance of a predictive model by combining multiple learners. Previous researches and studies show that generally an ensemble learning performs better than the individual (base) learners [20]. In the following, we describe the state-of-the-art of the well-known individual (base) learning methods.

2.2. Individual (base) learning

2.2.1. Machine learning-based approaches

A common data mining task, with the foundations of machine learning is classification. Classification-based anomaly detection techniques analyze, evaluate and classify the data in two classes (i.e. normal or abnormal). They are used when the available training data are labeled. In the following, we present the most common classification techniques for anomaly detection applications.

Support Vector Machine-based anomaly detection Support Vector Machine (SVM) is an effective technique for detecting anomalies (or outliers). Typically, SVM is associated with supervised learning, but its extensions (e.g. OneClass-SVM) can be used to identify anomalies as an unsupervised problems, where the training data is not labeled. However, the SVM is one of the most successful classification algorithms, but it is a time-consuming task in the training step, which limits its use. In addition, generally the SVM considers the features of data equally, while in real datasets, many features are unneeded, redundant or less important. Due to the shortcomings of the standard SVM for detecting anomalies, in the recent years, variant of SVM are suggested [21, 22].

Density-based anomaly detection Density-based anomaly detection is based on the k-Nearest Neighbors (k-NN) classification algorithm. The nearest set of k data points are evaluated using a metric such as Eucledian and Hamming distance. The k-NN is a simple and non-parametric lazy learning technique and it is one of the oldest methods of classification. While k-NN classifier usually works well in the terms of accuracy, it is slow in the recognition step, because the distances between the new data point and all the training data need to be computed. So, in the literature, there have been attempts to make it faster [23, 24], and research works are in progress to investigate its reliability and scaling properties [25, 26].

Naive Bayesian-based anomaly detection Bayesian networks have been used for anomaly detection in the multi-class setting by predicting the class membership probabilities. They work based on the Bayes' theorem, with strong independence assumptions between the features to simplify the task. The Bayes' rule allows unknown probabilities to be computed from known conditional probabilities, usually in the causal direction. Naive Bayesian networks are fast to train and classify, space efficient, not sensitive to irrelevant features and easy to implement. But their main disadvantage is that the Naive Bayes classifiers make a very strong assumption on the shape of data distribution (i.e. independence of features). However, in practice, they can work surprisingly well and comparable in performance with other classification algorithms, even when the conditional independence assumption is not true [27, 28, 29]. Over the last decade, several variants of the basic Naive Bayes technique have been proposed for anomaly detection [30, 31].

Neural network-based anomaly detection Neural Networks have been applied to anomaly detection in multi-class as well as one-class setting. They consist of a connected set of processing units distributed several layers, namely input, hidden and output layers, where each connection is characterized by a 'synaptic' weight that determines how the signal will propagate from one unit to another one. By adjusting these weights, the neural networks benefit from their learning algorithms to learn the relationship between inputs and outputs and to predict the correct class label of the input data. The neural networks are a simple manner to signify nonlinear relationships between features. However, they are computationally expensive to train and generally, require a large set of training data. A basic neural networks-based anomaly detection technique works in two steps. Firstly, a neural network is trained on the normal training data to learn the different normal classes, and then, each test data is provided as an input to the neural network. If the network accepts the test input, it is normal and otherwise, it is an anomaly [32]. In the literature, several variants of the basic neural network technique have been proposed for anomaly detection [33, 34].

Rule-based anomaly detection Rule-based anomaly detection methods learn rules that capture the normal behavior. While a test instance is not covered by any such learned rule, it is considered as an anomaly. These techniques have been applied in one-class and multi-class setting. One of the most common rule-based techniques used in anomaly detection is associated with decision trees, where they can be used to detect anomalies in large datasets. A decision tree algorithm generates a tree structure where each internal node stands for a decision on a feature and each leaf node take a class label. So, there is a path from the root node to the labeled leaf node, considered as a set of rules, which makes it easy to classify new unlabeled data. Decision trees have several advantages compared to the other machine learning based classification approaches, which make them more suitable for anomaly detection. In particular, they have a simply explainable framework and they are less sensitive to problem of the curse of dimension [1].

Random Forests (RF) [35] is a widely used ensemble learning method for classification and regression and operates by constructing a plenty of decision trees during train and test procedure. The term came from random decision forests that was first proposed in 1995 [36]. While in the standard decision tree, each node is bisect using the best split among all the features, in a random forest algorithm, each node is bisect among a small subset of randomly selected input features. In general, the more trees in the forest the more robust the forest looks like, and the higher the number of trees in the forest gives the more accurate results. The main advantages of the random forest algorithm are:

- It has ability to handle unbalanced datasets
- It is robust against over training and over-fitting
- It runs efficiently on very large datasets with many features
- It can handle thousands of input features without feature deletion
- It gives estimates of which features are important in the classification

and lastly, it is unexcelled in accuracy among many current anomaly detection algorithms [37, 38, 39].

In summary, random forest is a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the aim of reducing the variance, where it is becoming a popular algorithm for both classification and regression, because it does not have many tuning parameters, is a highly flexible classifier and often works quite well. However, random forest has been observed to over-fit in classification for some noisy datasets. In addition, for data including categorical features with different number of levels, it is biased in favor of those features with more levels [40, 41]. Since, random forest has been regarded as one of the most efficient approaches in classification (and anomaly detection) [42], in this work, we try to deploy the random forest into an ensemble learning algorithm using a pattern mining-based method to acquire even higher performance.

2.2.2. Pattern mining-based approaches

The problem of pattern mining has been widely studied in the literature because of its numerous applications to a variety of data mining and machine learning problems such as clustering and classification. It consists of developing (or using) data mining algorithms to discover interesting, useful or unexpected patterns in the data. Pattern mining can be applied to various types of data such as strings, transaction, sequence (time series), spatial data, and graphs. Typically, an interesting pattern is a pattern that appears frequently in the data. Therefore, in simple words, pattern mining is a way to find all of frequent patterns whose occurrence frequency in the data is 'no less' than the pre-defined threshold value. But there are many types of patterns such as sequential patterns, frequent itemsets, frequent subgraphs, frequent episodes, etc, and all of those types of patterns can be said to be frequent patterns. The most common one is the support-based framework, in which itemsets with frequency above a given threshold are found. In the following, we discuss more about the frequent itemset mining in detail.

Frequent itemset mining-based anomaly detection Frequent itemset plays an essential role in many mining tasks which trys to find interesting patterns from the datasets. The original motivation for searching the frequent itemsets came from the need to analyze the supermarket transaction data, that is, to examine customer behavior in terms of the purchased products [43], while the frequent itemsets of products describe how often items are purchased together. But what is an itemset and the frequent itemsets?

An Itemset (or element) is a non empty set of items $(x_1, x_2, ..., x_m)$, and a frequent itemset is an itemset whose support is greater than or equal to a minimum support threshold.

Here, the support (σ) is the frequency of occurrence of an itemset in a dataset. The task of discovering all frequent itemsets is quite challenging. The search space is exponential in the number of items occurring in the database. Furthermore, the major problem with the frequent itemset mining methods is the explosion of the number of the results, while it is difficult to find the most interesting frequent itemsets. Hence, we need to seek the most efficient techniques to solve this problem.

Frequent closed itemset mining is a task of discovering frequent itemsets whose support counts are different than those of their supersets. It means that an itemset is 'closed frequent' if none of its immediate supersets has the same support count as the itemset. Therefore, the size of frequent closed itemsets are much smaller than all the frequent itemsets, while we do not lose any information. In a nutshell, the frequent closed itemsets provide a compact yet lossless representation of the frequent itemsets.

Maximal frequent itemset is a frequent itemset for which none of its immediate supersets in the database is frequent. This representation is valuable because it provides the most compact representation of the frequent itemsets, and so when the search space is an issue or when we have a very large dataset, it is very helpful.

Let us make it more clear by one example. Consider 4 sequences as {a,b,c,d,e}, {a,b,d}, {b,e,a,c}, and {b,c,d,e}, where the minimum support (minsup) is equal to 2. {b,c} is a frequent itemset because it appears in two sequences (it has a support of 2). {b,c} is not a closed frequent itemset, because it is contained in a larger sequential pattern {b,c,d} having the same support. {b,c,d} has a support of 2. It is also not a closed frequent itemset, because it is contained in a larger sequential pattern {b,c,d,e} having the same support. {b,c,d,e} is a closed frequent itemset, because it is not included in any other sequential pattern having the same support. In this case, {b,c,d,e} is also a maximal frequent itemset, since none of its immediate supersets in the data is frequent.

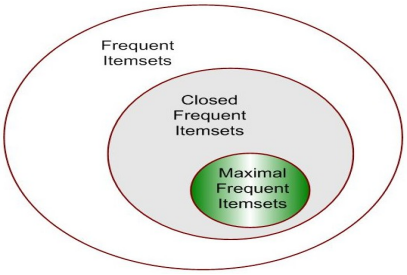

Figure 1: The relationship between frequent itemsets representations

Lastly, Figure 1 illustrates the relationship between frequent itemsets, closed frequent itemsets and maximal frequent itemsets representations. As we mentioned earlier closed and maximal frequent itemsets are subsets of frequent itemsets, but maximal frequent itemsets is a more compact representation, since it is a subset of the closed frequent itemsets. Notice that the closed frequent itemsets are more widely used than maximal frequent itemset [44]. So, the quesion now is how to get the frequent itemsets.

The most popular algorithm for itemset mining is without a doubt Apriori algorithm [45], which is designed more than 20 years ago, and is the basis of many efficient algorithms developed later. However, it is originally designed to be applied on a transaction data to discover patterns in transactions made by customers in stores, it can also be applied in several other applications. Apriori-based algorithms take as input a minimum support threshold and output all frequent itemsets, i.e. groups of items shared by no less than minimum support transactions in the input data. Algorithm 1 illustrates the apriori-all algorithm in a simple way.

Algorithm 1 Apriori(-All)

input: <**minsup**> minimum support threshold
output: frequent itemsets

do scan data once to get frequent 1-itemsets
repeat
    generate length-(k+1) candidate itemsets from length-k frequent itemsets
    test candidates againts DB to find frequent (k+1)-itemsets
    set k = k + 1
until no frequent or candidate set can be generated
return frequent itemsets

This can be done easily for a small data. If we have n items in the data, there will be 2^n possible itemsets. This is not a lot while the data size is small. But consider a large dataset having 1,000 items, the number of possible itemsets would be: $2^{1000} = 1.26e30$, which is huge, and practically not possible to use a apriori-based approach to find the frequent itemsets. In general, candidate counting , problem of I/O minimization, reducing the number of comparisons as well as handling large data are the most challengs in frequent itemset mining.

Later in [46], authors proposed a faster algorithm than Apriori-All, called GPS, which scales linearly with the number of data-sequences. The basic structure of the GSP for finding frequent itemsets is as follows: multiple-passing, candidate generation and testing (see Algo 2). Notice that in the loop cycle, one can generate length-(k+1) candidate sequences using the Apriori method. But still using the GPS algorithm, a huge set of candidates could be generated, we need to have multiple scans of data, as well as difficulties at mining long sequence patterns are exist.

Algorithm 2 GPS: Generalized Sequential Pattern

input: <**minsup**> minimum support threshold
output: frequent itemsets

repeat for each level (e.g. length-k)
    scan data to find length-k frequent sequence
    generate length-(k+1) candidate sequences from the length-k frequent sequences
    set k = k + 1
until no frequent or candidate set can be generated
return frequent itemsets

The above Apriori-based approaches are horizontal format-based and use the breadth-first search to mine with a hierarchical structure. Nearly two decades ago, the vertical format-based methods are proposed, where they tried to read the data, convert it to a vertical representation, and perform a depth-first search by joining items to each patter. The most well-known ones are SPADE (Sequential PAttern Discovery using Equivalence classes) [47], SPAM (Sequential PAttern Mining using a bitmap representation) [48] and LAPIN (LAst Position INduction) [49] and its improved version LAPIN-SPAM [50]. SPADE algorithm, which uses equivalence classes to discover the sequential patterns (itemsets), is an Apriori-based hybrid miner and can be either breadth-first or depth-first. It exploits sequential patterns by utilizing a vertical id-list database format and a vertical lattice structure. By using the SPADE algorithm a huge set of candidates could be generated, and also it waste

a lot of time on merging ID lists of the candidates, which prevent its usage. SPAM is similar to SPADE, but SPAM uses bitmap representation and bitwise operations rather than regular and temporal joins. First of all, SPAM consider all the sequences arranged in a sequence tree. Each sequence in the sequence tree can be considered as a sequence-extended sequence and an itemset-extended sequence. Then using prune candidate extension, it generates the frequent pattern (itemsets). Space utility in SPAM may not be good, and also it needs to load all data into memory, which will be inefficient (even impractical) for large data. The authors of LAPIN algorithm tried to reduce searching by scan only part of the search space. The key feature of LAPIN is that the last position of item s is the key to judge whether a k-length frequent sequence can grow to be frequent appending it with s or not. But still the support counting is time consumption.

Beside the above mentioned Apriori-based approaches, researchers proposed a variety of algorithms using pattern-growth techniques. In the early years of the 20th century, a pattern-projected sequential pattern mining algorithm named FreeSpan was introduced in [51], which obtains all frequent sequences (itemsets) based on so-called projected pattern growth. Note that a projected data is the set of suffixes w.r.t. a given prefix sequence. Later the most representative algorithm using the pattern-growth strategy, called PrefixSpan, was proposed in [52], which explores prefix-projection in sequential pattern mining. It tests only the prefix sub-sequences, and then projects their corresponding postfix subsequences into the projected sub-databases. By exploring only local frequent sequences (itemsets), sequential patterns can be recursively grown in each projected sub-database. PrefixSpan algorithm mines the complete set of patterns, but greatly reduces the efforts of candidate subsequence generation. Moreover, prefix-projection substantially reduces the size of projected data and leads to efficient processing. Although the projection-based approaches (i.e., FreeSpan, PrefixSpan) can achieve a significant improvement over Apriori-based approaches [53], the projection mechanism still suffers from some drawbacks, while the major cost is caused by constructing projected databases.

In the recent years, some other pattern-growth algorithms have been developed, such as FS-Miner [54], PLWAP [55], etc. Furthermore, the interesting idea of constraint-based techniques has been widely studied, including closed sequential patterns, maximal sequential pattern and top-k sequential patterns, etc. Up to now, some algorithms for mining closed and maximal sequential patterns have been proposed, such as CloSpan [56], ClaSP [57] and CloFAST [58], which try to prune the search space. However, the maximal representation may cause the information loss of support. In simple words, each method of pattern mining algorithms has advantages and disadvantages where the efficiency also could be related to the data type and size.

3. The proposed approach

In this section, we explain in detail the proposed ensemble learning anomaly detection, called fim-RF. Algorithm 3 presents in a very simple way the different steps of the proposed anomaly detection approach.

First of all, we do data pre-processing which involves cleaning the data and removing redundant and unnecessary entries. To do so, we use feature engineering based on a) feature selection, b) feature encoding, c) feature construction and d) feature normalization. The aim of feature selection is to come from many features to a few that are useful, since not all the features are created equally. Those features that are irrelevant to the problem need to be removed. Also, there are some features that will be more important than others to the model accuracy. Furthurmore, some features will be redundant in the context of other features. Feature selection addresses these problems by automatically selecting a

Algorithm 3 Function fim-RF($\mathbf{S}$, $\mathbf{S}_l$, $\mathbf{x}$)

input

$\mathbf{S}$: training data, $\mathbf{S}_l$: labels of data (0:normal, 1:abnormal), $\mathbf{x}$: a test instance

min_{sup}: minimum support threshold in frequent itemset mining

α, β: ensemble learning regularization parameters

τ: classification probability threshold

output

$\mathbf{x}_l$: label of the test instance

do pre-processing

feature selection

feature encoding

feature construction

feature normalization

$\mathbf{x}_{l_{rf}}$, $P_{rf}(\mathbf{x})$ = random_forest_classifier($\mathbf{S}$, $\mathbf{S}_l$, $\mathbf{x}$)

do frequent closed/maximal itemset mining

$P_{nf}(\mathbf{x})$ = probability of $\mathbf{x}$ to be a normal frequent closed/maximal itemset

$P_{af}(\mathbf{x})$ = probability of $\mathbf{x}$ to be a abnormal frequent closed/maximal itemset

$$P(\mathbf{x}) = \frac{\alpha.\Big(\frac{P_{nf}(\mathbf{x}) + (1 - P_{af}(\mathbf{x}))}{2}\Big) + \beta.P_{rf}(\mathbf{x})}{\alpha + \beta}$$

if $P(\mathbf{x}) < \tau$ **then**

$\mathbf{x}_l = 0$ (normal)

else

$\mathbf{x}_l = 1$ (abnormal)

return $\mathbf{x}_l$

subset that are most useful to the problem. Here, we rely on chi-square test, which is a statistical test of independence to determine the dependency of two features. We rank the features and then we choose the top-k features. Feature encoding involves converting the features of the data into numerical data and saving in a machine-readable format, which is essential for many data mining algorithms. Because they require data to consist of purely numerical features. Since packet data consists of both numerical and categorical features we adopt an effective method of converting categorical features into numerical ones. When the categorical feature takes its values in some finite set of categories, one typical conversion method is to adopt binary number representation where we use m binary numbers to represent a m-category feature (one-hot encoding). However, in case the number of categories for each categorical feature is very large the dimension of the input will be potentially intractable. To solve this problem, we use a histogram based encoding to model the distribution of values. First, we encode each categorical value into its integer representation, and then, we evaluate the frequency distribution histogram of numbers. To help detecting network, IP scans and distributed attacks, we create a new feature as the number of distinct IP-sources associated to the IP-destination. Lastly, feature normalization which plays a crucial role in the data pre-processing. Since, without normalization, features with significantly larger values dominate the features with smaller values, we normalize the data in the boundary of [0,1] by:

$$\hat{x}_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{1}$$

After the pre-processing steps, we build our proposed ensemble classifier as follows: We

firstly partition the data according to their application types (such as HTTPWeb, SSH,...) due to the difference behaviour of data in different applications, prior to run the random forest classification algorithm to get the predicted label ($\mathbf{x}_{l_{rf}}$) and the probability $\mathrm{P}_{rf}(\mathbf{x})$ which is the proportion of votes of the trees in the forest for test instance $\mathbf{x}$. In the context of pattern mining, we rely on frequent itemset mining. To do so, we obtain the top-k frequent close (and maximal) itemsets for normal data as well as abnormal (i.e. attack) data using an improved ECLAT algorithm. ECLAT algorithm, originally proposed in [59], is based on the breadth-first search strategy, which adopts the technologies of vertical data format, lattice theory, equivalence classes, intersection and so on. The main strategy steps of ECLAT are as follows: Scan the data to get all frequent k-itemsets, generate candidate (k+1)-itemsets from frequent k-itemsets, then get all frequent (k+1)-itemsets by clipping non-frequent candidate itemsets, and repeat the above steps, until no candidate itemset can be generated. By partitioning list of the set of itemset, we reduce the search space as well as the time of generating candidate itemsets, and speed up the calculation of intersection.

Once, we obtain the frequent closed (and maximal) itemsets for normal and abnormal data, we need to calculate the probability of normal classes of test instance $\mathbf{x}$ as well as the abnormal one. The probability of $\mathbf{x}$ to be a normal frequent closed/maximal itemset, $P_{nf}(\mathbf{x})$, is defined by:

$$P_{nf}(\mathbf{x}) = \frac{S_{\mathbf{x}_{nf}}}{N} \tag{2}$$

where $S_{\mathbf{x}_{nf}}$ is the support number of $\mathbf{x}$ to be a normal frequent itemset, with respect to the minimum support threshold equal to the pre-defined given min_{sup} value, and N is the total number of data points. Similarly, the probability of $\mathbf{x}$ to be an abnormal frequent itemset, $P_{af}(\mathbf{x})$, is defined as:

$$P_{af}(\mathbf{x}) = \frac{S_{\mathbf{x}_{af}}}{N} \tag{3}$$

where, $S_{\mathbf{x}_{af}}$ the support number of $\mathbf{x}$ to be an abnormal frequent itemset.

For each test instance $\mathbf{x}$, different situations can happen, such as a) $\mathbf{x}$ be a normal frequent itemset and not to be an abnormal frequent itemset, b) $\mathbf{x}$ be an abnormal frequent itemset and not to be a normal frequent itemset, and c) $\mathbf{x}$ be a normal frequent itemset and also be an abnormal frequent itemset, The situation 'a' leads that the test instance $\mathbf{x}$ with high probability be normal, situation 'b' leads that the test instance $\mathbf{x}$ with high probability be abnormal (i.e. attack), where in the situation 'c', one need to have confidence in the probabilities $P_{nf}(\mathbf{x})$ and $P_{af}(\mathbf{x})$.

Finally, as we mentioned above in the algorithm 3, the classification probability of the proposed ensemble learning method is defined by:

$$P(\mathbf{x}) = \frac{\alpha.\Big(\frac{P_{nf}(\mathbf{x}) + (1 - P_{af}(\mathbf{x}))}{2}\Big) + \beta.P_{rf}(\mathbf{x})}{\alpha + \beta} \tag{4}$$

where α, β are the ensemble learning regularization parameters. Notice that, one can optimize the parameters using the cross-validation test. In the next section, to have a closer look at the ability of the proposed ensemble learning (fim-RF), we detailed extensive experiments.

4. Experimentation

Here we describe the dataset used to lead our experiments, prior to specify the validation process, and to present the obtained results.

4.1. The benchmark ISCX dataset

In this work, we used the public benchmark ISCX dataset [60], to perform experiments and evaluate the performance of our proposed ensemble learning approach. The dataset includes more than million of the traffic packets with twenty features, where it covers one week of network activities with normal and abnormal (i.e. attack) traffic data. Four primary kinds of network attack (i.e. Brute Force SSH, Infiltrating, HTTP DoS, and DDoS) are conducted with normal traffic. Table 1 describes the ISCX dataset considered in our experiments, with its characteristics: number of normal traffic data, number of attack data and number of features.

appName	# of Normals	# of Attacks	# of Features
HTTPWeb	681151	40351	20
SSH	2585	7305	20
ICMP	7919	295	20
FTP	13181	226	20
DNS	309286	73	20

Table 1: ISCX (flows): data description

As mentioned, as input to the proposed ensemble learning anomaly detection algorithm, we use of the pre-processed data. Since, the normal traffic patterns look very different depending on the application or service, data are classified according to their application layers such as HTTP Web, SSH, FTP, ICMP and so on, which makes it more efficient to build an anomaly detector for each of these application layers.

4.2. Validation process

Here we compare the proposed anomaly detection algorithm (fim-RF) with the state-of-the-art anomaly detection methods (i.e. SVM, 1-NN, NB, MLP, Decision Tree and RF). To evaluate each method, we rely on the commonly used measures for anomaly detection: ACCuracy (ACC), True Positive Rate (TPR) and False Positive Rate (FPR). The accuracy is the proportion of true results (both true positives and true negatives) among the total number of cases examined. True positive rate which is called 'recall', or 'sensitivity' in binary classification, measures the proportion of actual positives that are correctly identified. While recall can be viewed as the probability of detection, false positive rate is the probability of false alarms. The mentioned comparison measures are defined as:

$$\text{ACC} = \frac{TP + TN}{TP + FN + FP + TN},\ \text{TPR} = \frac{TP}{TP + FN},\ \text{FPR} = \frac{FP}{FP + TN}$$

The 'accuracy' and 'true positive rate' lies in [0, 100] in percentage. The higher index, the better the agreement is. In the other side, the lower 'false positive rate' illustrates the better result. Finally, Table 2 presents the classical confusion matrix, where N shows 'normal' data and P illustrates 'abnormal' (or 'attack').

		Predicted class	
		Positive class	Negative class
Actual	Positive class	TP (True Positive)	FN (False Negative)
	Negative class	FP (False Positive)	TN (True Negative)

Table 2: Confusion matrix

Training and testing sets are formed by k-fold cross validation in the ratio of 80% and 20% of the network traffic, respectively. For all the state-of-the-art methods, the parameters are estimated through a standard grid search process, and finally, the results reported hereinafter are averaged after 10 repetitions of the corresponding algorithm.

4.3. Experimental results

In the context of anomaly detection, the 'accuracy', the 'true positive rate' and the 'false positive rate' for each method, and for the various tested protocols, are reported in Tables 3, 4 and 5, respectively. Results in bold correspond to the best assessment values.

appName	SVM	1-NN	Naive Bayes	Decision Tree	Neural Network	RF	fim-RF
HTTPWeb	98.99	99.70	98.04	99.89	99.02	99.88	**99.93**
SSH	99.47	99.90	99.22	99.87	99.89	99.89	**99.98**
ICMP	99.83	99.95	99.90	99.99	99.93	99.99	**100.0**
FTP	99.62	99.95	99.54	99.97	99.94	99.97	**99.98**
DNS	99.98	**99.99**	96.18	99.98	99.98	**99.99**	**99.99**

Table 3: Comparison of 'Accuracy' (in %)

appName	SVM	1-NN	Naive Bayes	Decision Tree	Neural Network	RF	fim-RF
HTTPWeb	98.20	97.47	92.74	99.12	98.75	99.38	**99.72**
SSH	99.78	99.95	99.34	99.92	99.95	99.97	**99.98**
ICMP	97.44	99.74	**100.0**	**100.0**	98.68	**100.0**	**100.0**
FTP	87.20	99.36	99.79	99.36	98.52	99.79	**100.0**
DNS	52.31	86.15	52.31	89.61	18.46	89.23	**98.49**

Table 4: Comparison of 'True Positive Rate' (in %)

appName	SVM	1-NN	Naive Bayes	Decision Tree	Neural Network	RF	fim-RF
HTTPWeb	0.96	0.16	1.64	0.05	0.96	0.08	**0.04**
SSH	1.39	0.23	1.11	0.26	0.26	0.33	**0.03**
ICMP	0.08	0.04	0.10	0.01	0.03	0.01	**0.00**
FTP	0.18	0.04	0.46	**0.02**	0.03	**0.02**	**0.02**
DNS	0.01	3.81	0.01	0.01	0.01	**0.00**	**0.00**

Table 5: Comparison of 'False Positive Rate' (in %)

According to the Tables 3, 4 and 5 the proposed fim-RF algorithm leads to the best 'accuracy', 'true positive rate' and 'false positive rate' results, for all the application layers in comparison with the other methods.

4.3.1. Relationships between the methods and application layers

To compare globally the different anomaly detection approaches, here we rely on a Multiple Correspondence Analysis (MCA), to analyze the *seven* methods (considered as individuals) and *five* application layers (considered as categorical variables). MCA is a data analysis technique for nominal categorical data and can be viewed as an extension of correspondence analysis (CA) which allows one to analyze the pattern of relationships of several categorical dependent variables. It can also incorporate quantitative variables. MCA is concerned with relationships within a set of variables, which usually are homogeneous, and allows the direct representation of individuals as points in geometric space.

To do so, each method is described by a vector ("–", "+","++",...), with as many dimensions as there are application layers, in which the modalities "++", "+" and "–" indicate whether the accuracy, detection rate or false alarm rate of a method on an application layer is respectively highly greater, greater or lower than the mean obtained for that application layer over all the methods. Distinct groups of methods, corresponding to distinct ways to perform on the different application layers, can be distinguished.

From Figure 2, one group (left-bottom) is defined by fim-RF, RF, DT and 1-NN and is opposed to the other methods as it yields the highest accuracy performances (corresponding to modality "++"). In addition, as one can see NB anomaly detection classifier yields the lowest accuracy (corresponding to modality "–"), particulary on DNS, HTTPWeb and SSH.

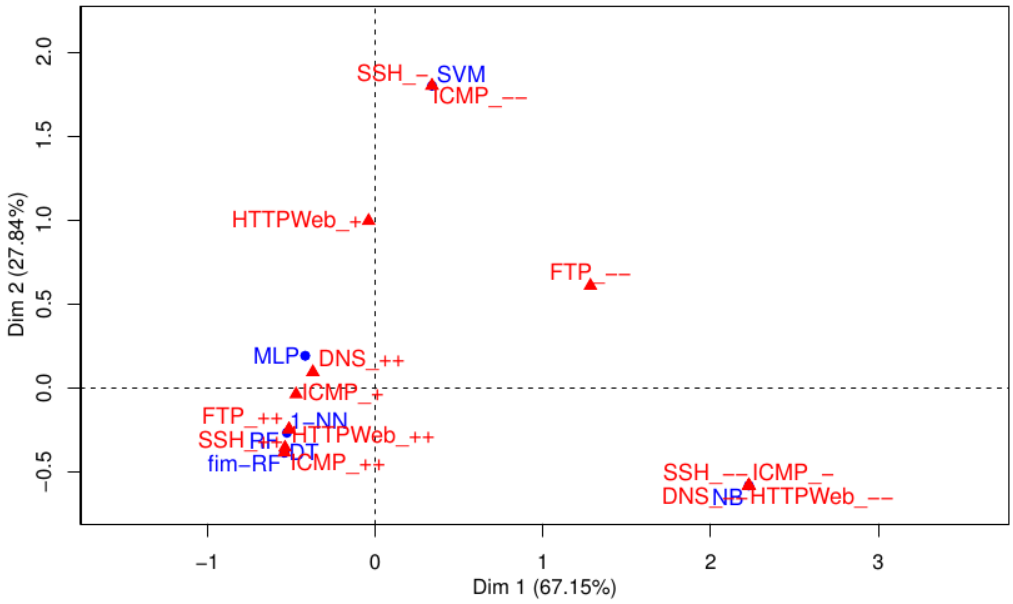

Figure 2: Global comparison of the 'Accuracy'

From Figure 3, similar as the previous figure, one can see that the fim-RF, RF, DT and 1-NN and outperform the other methods as they yield the highest true positive rates (corresponding to modality "++"). Finally, Figure 4 shows that fim-RF, RF and DT have the lowest false positive rate almost for all the application layer, while for instance, NB has the highest false alarm rate for HTTPWeb, FTP, ICMP and SSH, but good results for DNS (low false positive rate).

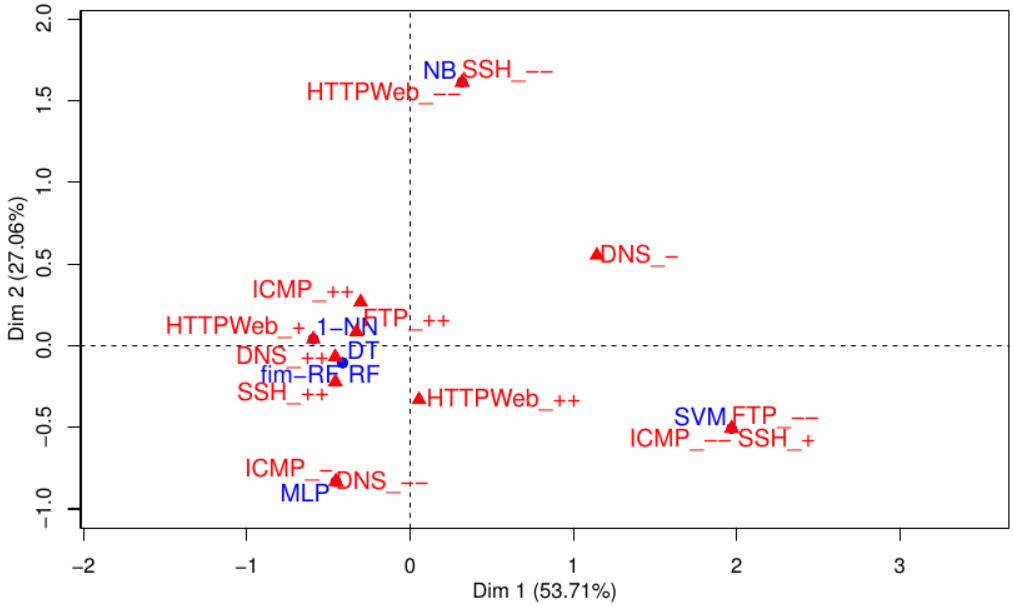

Figure 3: Global comparison of the 'Detection Rate'

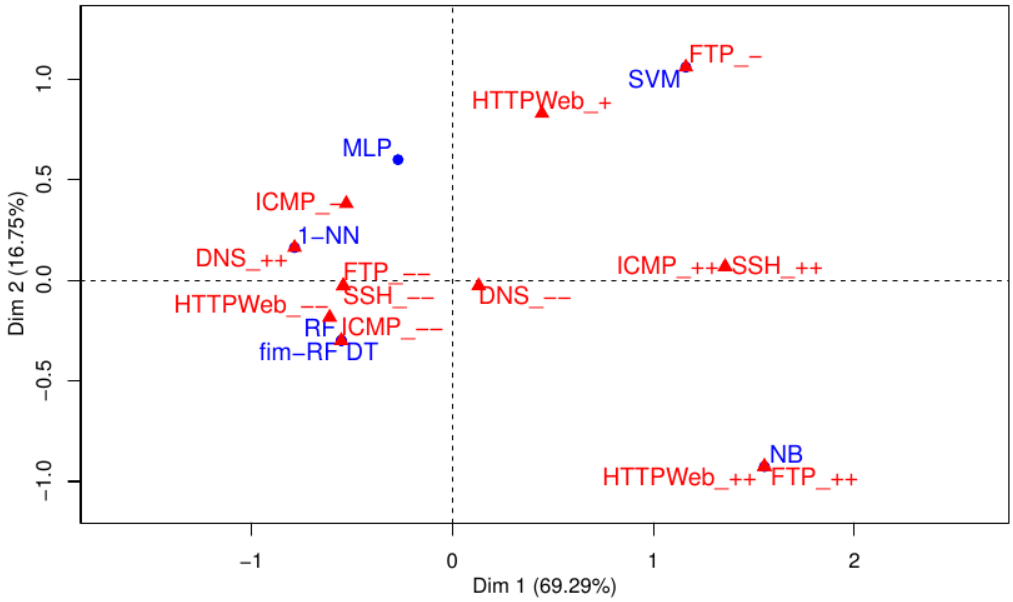

Figure 4: Global comparison of the 'False Alarm Rate'

5. Conclusion

Ensemble learning is a powerful machine learning paradigm which has exhibited apparent advantages in many applications. This paper has proposed an ensemble learning anomaly detection by using a machine learning role-based (i.e. random forest) and a pattern mining-based (frequent closed/maximal itemset) method. The efficiency of the introduced ensemble learning method (fim-RF) is analyzed on a dynamic, scalable and labeled dataset, called ISCX, which is now-days commonly explored for data intrusion benchmarking. The results illustrate that the fim-RF, in overall, outperforms the other state of the art methods (i.e. SVM, 1-NN, NB, MLP, Decision Tree and RF), through the commonly used measures: accuracy, true positive rate and false positive rate.

6. References

[1] V. J. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial Intelligence Review*, vol. 22, pp. 85–126, Oct 2004.

[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, pp. 15:1–15:58, July 2009.

[3] E. Aleskerov, B. Freisleben, and R. B. Rao, "Cardwatch: a neural network based database mining system for credit card fraud detection," in *CIFEr*, 1997.

[4] C. Spence, L. Parra, and P. Sajda, "Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model," in *Proceedings of the IEEE Workshop on Mathematical Methods in Biomedical Image Analysis (MMBIA'01)*, MMBIA '01, (Washington, DC, USA), pp. 3–, IEEE Computer Society, 2001.

[5] B. Liu, S. Nath, R. Govindan, and J. Liu, "DECAF: Detecting and characterizing ad fraud in mobile apps," in *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, (Seattle, WA), pp. 57–70, USENIX Association, 2014.

[6] R. Oentaryo, E.-P. Lim, M. Finegold, D. Lo, F. Zhu, C. Phua, E.-Y. Cheu, G.-E. Yap, K. Sim, M. N. Nguyen, K. Perera, B. Neupane, M. Faisal, Z. Aung, W. L. Woon, W. Chen, D. Patel, and D. Berrar, "Detecting click fraud in online advertising: A data mining approach," *J. Mach. Learn. Res.*, vol. 15, pp. 99–140, Jan. 2014.

[7] R. Fujimaki, T. Yairi, and K. Machida, "An approach to spacecraft anomaly detection problem using kernel feature space," in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, (New York, NY, USA), pp. 401–410, ACM, 2005.

[8] F. E. M.A., "Xli. on discordant observations," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 23, no. 143, pp. 364–375, 1887.

[9] D. E. Denning, "An intrusion-detection model," *IEEE Trans. Softw. Eng.*, vol. 13, pp. 222–232, Feb. 1987.

[10] W. Feng, Q. Zhang, G. Hu, and X. Huang, "Mining network data for intrusion detection through combining svms with ant colony networks," *Future Generation Comp. Syst.*, vol. 37, pp. 127–140, 2014.

[11] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," in *KES*, 2015.

[12] S. Duque and M. N. bin Omar, "Using data mining algorithms for developing a model for intrusion detection system (ids)," *Procedia Computer Science*, vol. 61, pp. 46 – 51, 2015. Complex Adaptive Systems San Jose, CA November 2-4, 2015.

[13] E. M. Knorr, R. T. Ng, and V. Tucakov, "Distance-based outliers: Algorithms and applications," *The VLDB Journal*, vol. 8, pp. 237–253, Feb. 2000.

[14] 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*, pp. 93–104, ACM, 2000.

[15] H.-P. Kriegel, P. Kroger, E. Schubert, and A. Zimek, "Outlier detection in arbitrarily oriented subspaces," in *Proceedings of the 2012 IEEE 12th International Conference on Data Mining*, ICDM '12, (Washington, DC, USA), pp. 379–388, IEEE Computer Society, 2012.

[16] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, pp. 1443–1471, July 2001.

[17] S. Hawkins, H. He, G. J. Williams, and R. A. Baxter, "Outlier detection using replicator neural networks," in *Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery*, DaWaK 2000, (London, UK, UK), pp. 170–180, Springer-Verlag, 2002.

[18] Z. Zhou, "Ensemble learning," in *Encyclopedia of Biometrics*, pp. 411–416, Springer US, 2015.

[19] R. Polikar, "Ensemble learning," in *Ensemble machine learning*, Springer, Boston, MA, 2012.

[20] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *J. Artif. Int. Res.*, vol. 11, pp. 169–198, July 1999.

[21] T. Shon, Y. Kim, C. Lee, and J. Moon, "A machine learning framework for network anomaly detection using svm and ga," *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*, pp. 176–183, 2005.

[22] H. F. Eid, A. Darwish, A. E. Hassanien, and A. Abraham, "Principle components analysis and support vector machine based intrusion detection system," 2010.

[23] I. Levner, "Feature selection and nearest centroid classification for protein mass spectrometry," *BMC Bioinformatics*, vol. 6, p. 68, Mar 2005.

[24] S. Soheily-Khah, *Generalized k-means based clustering for temporal data under time warp*. Theses, Universite Grenoble Alpes, Oct. 2016.

[25] R. Gil-Pita and X. Yao, "Using a genetic algorithm for editing k-nearest neighbor classifiers," in *Intelligent Data Engineering and Automated Learning - IDEAL 2007* (H. Yin, P. Tino, E. Corchado, W. Byrne, and X. Yao, eds.), (Berlin, Heidelberg), pp. 1141–1150, Springer Berlin Heidelberg, 2007.

[26] W. Li, P. Yi, Y. Wu, L. Pan, and J. Li, "A new intrusion detection system based on knn classification algorithm in wireless sensor network," *Journal of Electrical and Computer Engineering*, 2014.

[27] R. Entezari-Maleki, A. Rezaei, and B. Minaei-Bidgoli, "Comparison of classification methods based on the type of attributes and sample size.," *JCIT*, vol. 4, no. 3, pp. 94–102, 2009.

[28] M. Kukreja, S. A. Johnston, and P. Stafford, "Comparative study of classification algorithms for immunosignaturing data," *BMC bioinformatics*, vol. 13, p. 139, 2012.

[29] A. Ashari, I. Paryudi, and A. M. Tjoa, "Performance comparison between nave bayes, decision tree and k-nearest neighbor in searching alternative design in an energy simulation tool," *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 11, 2013.

[30] D. Barbara, N. Wu, and S. Jajodia, "Detecting novel network intrusions using bayes estimators," in *Proc. SIAM Intl. Conf. Data Mining*, 2001.

[31] N. A. Heard, D. J. Weston, K. Platanioti, and D. J. Hand, "Bayesian anomaly detection methods for social networks," *Ann. Appl. Stat.*, vol. 4, pp. 645–662, 06 2010.

[32] C. De Stefano, C. Sansone, and M. Vento, "To reject or not to reject: That is the question-an answer in case of neural classifiers," *Trans. Sys. Man Cyber Part C*, vol. 30, pp. 84–94, Feb. 2000.

[33] A. K. Ghosh and A. Schwartzbard, "A study in using neural networks for anomaly and misuse detection," in *Proceedings of the 8th Conference on USENIX Security Symposium - Volume 8*, SSYM'99, (Berkeley, CA, USA), pp. 12–12, USENIX Association, 1999.

[34] M. F. Augusteijn and B. A. Folkert, "Neural network classification and novelty detection," *International Journal of Remote Sensing*, vol. 23, no. 14, pp. 2891–2902, 2002.

[35] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, pp. 5–32, Oct. 2001.

[36] T. K. Ho, "Random decision forests," in *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*, ICDAR '95, (Washington, DC, USA), pp. 278–, IEEE Computer Society, 1995.

[37] A. Liaw and M. Wiener, "Classification and Regression by randomForest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[38] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *Trans. Sys. Man Cyber Part C*, vol. 38, pp. 649–659, Sept. 2008.

[39] N. B. Saeid SOHEILY-KHAH, Pierre-Francois Marteau, "Intrusion detection in network systems through hybrid supervised and unsupervised machine learning process: A case study on the iscx dataset," *International Conference on Data Intelligence and Security, ICDIS*, 2018.

[40] T. Shi and S. Horvath, "Unsupervised learning with random forest predictors," *Journal of Computational and Graphical Statistics*, vol. 15, no. 1, pp. 118–138, 2006.

[41] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, pp. 3–42, Apr. 2006.

[42] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?," *J. Mach. Learn. Res.*, vol. 15, pp. 3133–3181, Jan. 2014.

[43] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," *SIGMOD Rec.*, vol. 22, pp. 207–216, June 1993.

[44] M. J. Zaki, "Mining maximal and closed frequent itemsets," in *New Generation of Data Mining Applications* (M. Kantardzic and J. Zurada, eds.), ch. 23, pp. 571–598, IEEE/Wiley Press, 2005.

[45] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, (San Francisco, CA, USA), pp. 487–499, Morgan Kaufmann Publishers Inc., 1994.

[46] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," in *Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '96, (London, UK, UK), pp. 3–17, Springer-Verlag, 1996.

[47] M. J. Zaki, "Sequence mining in categorical domains: Incorporating constraints," in *Proceedings of the Ninth International Conference on Information and Knowledge Management*, CIKM '00, (New York, NY, USA), pp. 422–429, ACM, 2000.

[48] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu, "Sequential pattern mining using a bitmap representation," in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, (New York, NY, USA), pp. 429–435, ACM, 2002.

[49] Z. Yang, Y. Wang, and M. Kitsuregawa, "Lapin: Effective sequential pattern mining algorithms by last position induction for dense databases," in *Advances in Databases: Concepts, Systems and Applications* (R. Kotagiri, P. R. Krishna, M. Mohania, and E. Nantajeewarawat, eds.), (Berlin, Heidelberg), pp. 1020–1023, Springer Berlin Heidelberg, 2007.

[50] Z. Yang and M. Kitsuregawa, "Lapin-spam: An improved algorithm for mining sequential pattern.," in *ICDE Workshops*, p. 1222, IEEE Computer Society, 2005.

[51] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu, "Freespan: Frequent pattern-projected sequential pattern mining," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, (New York, NY, USA), pp. 355–359, ACM, 2000.

[52] J. Pei, J. Han, B. Mortazavi-asl, H. Pinto, Q. Chen, U. Dayal, and M. chun Hsu, "Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth," in *17th international conference on data engineering*, pp. 215–224, 2001.

[53] W. Gan, J. C. Lin, P. Fournier-Viger, H. Chao, and P. S. Yu, "A survey of parallel sequential pattern mining," *CoRR*, vol. abs/1805.10515, 2018.

[54] M. El-Sayed, C. Ruiz, and E. A. Rundensteiner, "Fs-miner: Efficient and incremental mining of frequent sequence patterns in web logs," in *Proceedings of the 6th Annual ACM International Workshop on Web Information and Data Management*, WIDM '04, (New York, NY, USA), pp. 128–135, ACM, 2004.

[55] C. I. Ezeife, Y. Lu, and Y. Liu, "Plwap sequential mining: Open source code," in *Proceedings*

of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations, OSDM '05, (New York, NY, USA), pp. 26–35, ACM, 2005.

[56] X. Yan, J. Han, and R. Afshar, "Clospan: Mining closed sequential patterns in large datasets," in *In SDM*, pp. 166–177, 2003.

[57] A. Gomariz, M. Campos, R. Marin, and B. Goethals, "Clasp: An efficient algorithm for mining frequent closed sequences," in *Advances in Knowledge Discovery and Data Mining* (J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, eds.), (Berlin, Heidelberg), pp. 50–61, Springer Berlin Heidelberg, 2013.

[58] F. Fumarola, P. F. Lanotte, M. Ceci, and D. Malerba, "Clofast: Closed sequential pattern mining using sparse and vertical id-lists," *Knowl. Inf. Syst.*, vol. 48, pp. 429–463, Aug. 2016.

[59] M. J. Zaki, "Scalable algorithms for association mining," *IEEE Trans. on Knowl. and Data Eng.*, vol. 12, pp. 372–390, May 2000.

[60] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, pp. 357–374, May 2012.

Authors

Saeid SOHEILY KHAH graduated in computer engineering, and received master degree in artificial intelligence & robotics in 2005. He then received his second master in information analysis and management from Skarbek university in Warsaw.In 2013, he joined to the LIG (Laboratoire d'Informatique de Grenoble) at Université Grenoble Alpes as a doctoral researcher. He successfully defended his dissertation and got his Ph.D in Oct 2016. In Nov 2016, he joined to the IRISA/Expression at Universit Bretagne Sud as a postdoctoral researcher. Lastly, in Oct 2017, he joined Skylads as a research scientist. His research interests are machine learning, data mining, cyber security system, anomaly detection, digital advertising and artificial intelligence.

Yiming WU received his B.S.E.E. degree from Northwestern Polytechnical University, Xian, China, 2011. He received his Ph.D. degree in Electrical Engineering from University of Technology of BelfortMontbliard, Belfort, France, 2016. He joined Skylads as a data scientist in 2018, and his research has addressed topics on machine learning, artificial intelligence and digital advertising.