MSML: A Novel Multi-level Semi-supervised Machine Learning Framework for Intrusion Detection System

Haipeng Yao, Danyang Fu, Peiying Zhang, Maozhen Li, Yunjie Liu

Abstract—Intrusion detection technology has received increasing attention in recent years. Many researchers have proposed various intrusion detection systems using machine learning methods. However, there are two noteworthy factors affecting the robustness of the model. One is the severe imbalance of network traffic in different categories, and the other is the non-identical distribution between training set and test set in feature space. This paper presents a multi-level intrusion detection model framework named MSML to address these issues. The MSML framework includes four modules: pure cluster extraction, pattern discovery, fine-grained classification and model updating. In the pure cluster module, we introduce an concept of "pure cluster" and propose a hierarchical semi-supervised k-means algorithm (HSK-means) with an aim to find out all the pure clusters. In the pattern discovery module, we define the "unknown pattern and apply cluster based method aiming to find those unknown patterns. Then a test sample is sentenced to labeled known pattern or unlabeled unknown pattern. The fine-grained classification module can achieves fine-grained classification for those unknown pattern samples. The model updating module provides a mechanism for retraining. KDDCUP99 dataset is applied to evaluate MSML. Experimental results show that MSML is superior to other existing intrusion detection models in terms of overall accuracy, F1 score and unknown pattern recognition capability.

Index Terms—intrusion detection, semi-supervised learning, unknown pattern discovery, class imbalance, non-identical distribution.

1 Introduction

With the rapid development of the Internet, the number of network invasions greatly increases. As a widely-used precautionary measure, intrusion detection has become an important research topic. Machine learning (ML), which can address many nonlinear problems well, has gradually become the mainstream in the field of intrusion detection system. Many existing models based on ML employ supervised learning algorithms to train an intrusion detection classifier using a set of labeled training samples, then use this classifier to classify unlabeled test data.

There are two main problems in network traffic, which affect the robustness of the ML model. Firstly, network traffic has a severe class-imbalance problem. It means that some categories have much more samples than others. This problem is also called elephant traffic and mouse traffic problem. The generated ML model will suffer because the model will be much more suitable for elephant traffic rather than mouse traffic. Secondly, training data and test data can have a non-identical distribution problem. It means that our training data and test data are generated by two different probabilistic distributions. The independent identical distribution is an extremely important premise of statistical machine learning. Non-identical distribution problem can cause a decrease in accuracy rate. Unfortunately, the distribution of network traffic is not static because the uncertainty of users behaviors leads to the variable distribution of network traffic. The high cost of expert system determines that it is not possible to mark a large amount of network traffic in real-time. We usually train the model using historical labeled data.

In this paper, we propose a novel intrusion detection system based on a multi-level semi-supervised machine learning framework (MSML) to tackle with aforementioned problems. The main contributions of this paper are described as follows:

- In order to alleviate the class imbalance problem, we propose a cluster-based under-sampling methods, which is a hierarchical, semi-supervised k-means clustering algorithm.
- For the non-identical distribution problem, a method is proposed to distinguish known and unknown pattern samples in the test set. We can obtain a guaranteed high recognition accuracy for known pattern samples. Also, the unknown pattern samples could be fine-grained classified.

In this paper, the MSML framework is experimented and evaluated on the KDDCUP99 dataset. The results show the proposed framework significantly outperforms our baseline method and many other existing models in the respect of overall accuracy, F1 score and unknown pattern discovery ability. The results also show that when the non-identical distribution problem do not occur, our framework is still suitable and well-performed.

- *Haipeng Yao, Danyang Fu and Yunjie Liu are with Beijing University of Posts and Telecommunications, Beijing, P.R. China, Haidian District Xitucheng Road 10, 100876, Beijing, China. Email: yaohaipeng@bupt.edu.cn.*
- *Peiying Zhang is with the College of Computer & Communication Engineering, China University of Petroleum (East China), Qingdao 266580, P.R. China. He is also with State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications.*
- *Maozhen Li is with the Department of Electronic and Computer Engineering, Brunel University, Uxbridge, Middlesex, UB8 3PH, U.K.*

Manuscript received May 1, 2018; revised xxx 1, 2018.

The rest of this paper is organized as follows. Section 2 introduces the related works. In section 3, we present the details of our proposed MSML framework. Experiments and numerical results are discussed in Section 4. Section 5 is the summarization of this paper.

2 Related Work

Most researches focused on applying machine learning methods in intrusion detection or network traffic classification. These works generally can be summarized as a $1 + N$ problem. First, a supervised learning classifier is built using the labeled training dataset. The labeled training dataset contains one normal category and N intrusion categories. Then, the classifier is used to classify unlabeled test data to distinguish normal data and intrusion data. Many basic supervised learning algorithms such as decision tree [1], [2], [3], support vector machine [4], [5], neural network [6], [7] and ensemble learning [8], were all used in intrusion detection system. In addition, some unsupervised methods were also applied to assist supervised models in some cases. Al-Yaseen et al. [9] applied a hybrid support vector machine method. However, the hybrid support vector machine method was so time-consuming particularly when the training dataset was extremely large; Hence Al-Yaseen et al. proposed a modified k-means algorithm with an aim to compress the size of training dataset. Wang et al. [10] proposed a novel model framework. In their framework, N supervised classifiers were built in N prepared clusters so each classifier met a relatively easy learning problem. Feature selection algorithms or the descending dimension algorithms were also usually used before supervised learning algorithms to increase the efficiency of training [11], [12]. In recent years, deep learning methods such as CNN and SAE have been tried to be applied in intrusion detection system. Wang [13] proposed an end to end model using deep learning method.

For the class imbalance problem, widely-used approaches include under-sampling, overlap-sampling, ensemble learning and cost sensitive learning [14], [15]. In general, under-sampling is based on category. However, when there is an imbalanced problem within samples of one category, random under-sampling could lead to losses of rare pattern samples of the category. Several cluster-based methods [16] were then proposed and applied for the intrusion detection system. In our work, we introduced a concept of ”pure cluster pattern” and proposed a hierarchical semi-supervised k-means algorithm. The proposed algorithm can find all the samples which are in pure cluster patterns with an aim to address the class imbalance problem.

For the non-identical distribution problem, this paper is mainly inspired by the literature [17] and [18]. Erman et al. [17] presented the earliest work that uses semi-supervised learning method to detect unknown traffic. Unknown traffic is the traffic whose category is not in the labeled training samples. Unknown traffic is a kind of non-identical distribution problem. The idea of [17] was to mix labeled traffic with unlabeled traffic and then use k-means clustering. If none of the samples in a cluster had a label, then all samples in this cluster would be labeled with unknown. On the foundation of the work [17], the authors of [18] made their contributions to fine-grained classification for unknown traffic. However, according to their [17] related description of their dataset, we can see that their training set was relatively class-balanced and that the known traffic was identically distributed. However, the non-identical distribution problem results from not only the unknown traffic but also the known traffic. In our paper, we focus on identifying unknown patterns rather than unknown traffic.

There is a method called one-class svm [19] which is a potential way to identify unknown patterns. The conventional svm algorithm aims to solve the two-classification problem, where the selected hyperplane making the support vector farthest from intermediate hyperplane. One-class svm is only used for one category in the training set at a time, while the remaining samples are regarded as others. Then its bounded hyperplane wraps the samples of this category. All the bounds of the hyperplane are actually a wrapper for known pattern samples after using one-class svm for all categories. Moreover, all samples beyond the boundary should be marked as unknown patterns. Al-Yaseen et al. [9] applied this method to build the model. However, their results were highly dependent on the choices of the super-parameters. In addition, it is difficult to find the clear relationship between parameters in [9] accuracy. Some improper parameters even lead to a much worse result than that of using the traditional methods.

3 Proposed Scheme (MSML)

In this section, we introduce our proposed MSML framework, as shown in Figure 1. The data generator process for MSML is also shown in Figure 1.

The aim for data generator process is to generate the required training set and test set for the MSML framework. Due to the semi-supervised property of the MSML, the training set consists of labeled samples and unlabeled samples. The training labeled data was labeled in the past, reflecting the distribution of historical known network traffic. The training unlabeled samples and test samples are all generated by the network traffic generator, which reflects the distribution of current network traffic. The data preprocessing module is devised to do something necessary before training model such as normalization and data cleaning.

The MSML consists of four modules including pure cluster extraction, pattern discovery, fine-grained classification and model updating. The pure cluster extraction module aims to find large and pure clusters. In the pure cluster module, this paper defines an important concept of ”pure cluster pattern” and proposes a hierarchical semi-supervised k-means algorithm (HSK-means), aiming at finding out all the pure clusters. In the pattern discovery module, this paper defined the ”unknown pattern” and applied cluster based method to find those unknown patterns. The fine-grained classification module achieves fine-grained classification for those unknown pattern samples. The model updating module provides a mechanism for retraining. For any test sample, once labeled by one module, will not go on any more; and all test samples will be labeled in pure cluster extraction module, pattern discovery module and fine-grained classification module.

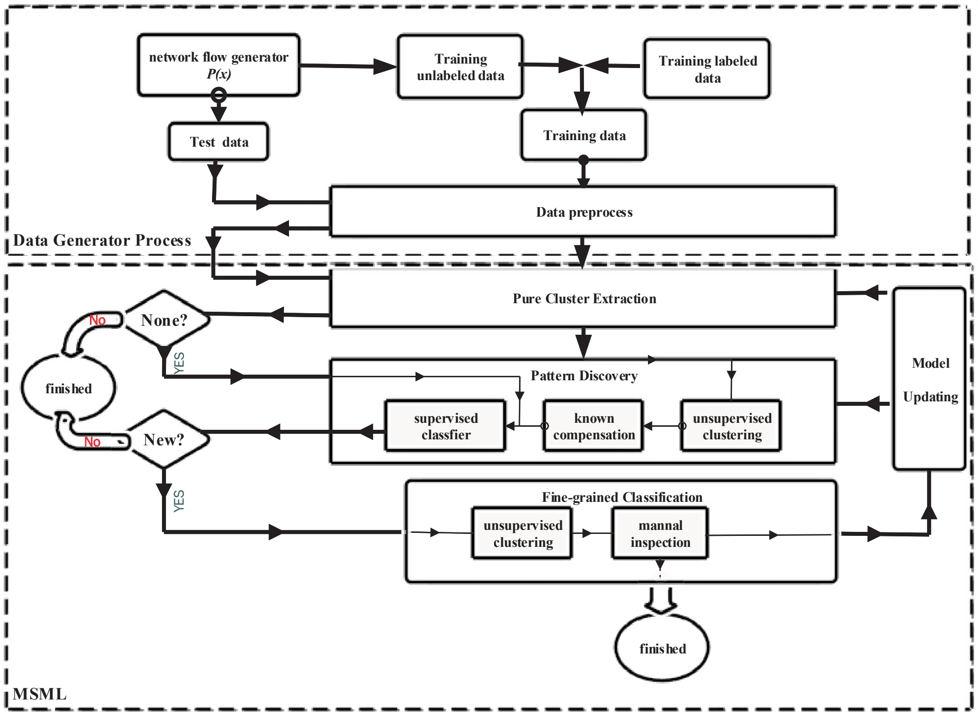


Fig. 1: The MSML framework

3.1 Pure Cluster Extraction(PCE)

The pure cluster can be defined as a cluster where almost all samples have the same category. The potential samples located in this pure cluster can be considered to be the same category as the other samples in this cluster.

Given a training labeled set, $Tr^l = \{l_1, l_2, \cdots, l_N\}$, which consists of N labeled samples and an unlabeled training set, $Tr^u = \{u_1, u_2, \cdots, u_M\}$, which consists of M unlabeled samples. The labeled and unlabeled training samples are merged with an aim to form our training set $Tr = Tr^l \cup Tr^u$. The training set are partitioned into K clusters $\{C_1, C_2, ..., C_K\}$ $(K \leq |Tr|)$ using the K-means clustering method. If the training labeled samples and the training unlabeled samples are identically distributed, we assume that a large cluster C_i which contains a large number of samples will meet the following formula, by the principle of the central limit theorem.

$$\frac{|C_i^l|}{|C_i|} \approx \frac{N}{N+M} \tag{1}$$

where C_i^l is the k-th cluster in the labeled samples. We can conclude that the cluster is not pure, if the left hand side of the equation 1 is significantly smaller than the right hand side. The reason is that the labeled samples only characterize a part of the whole cluster. Consequently, we mark a cluster pure cluster when all the labeled samples of this cluster belong to a same category and the cluster meets the following formula:

$$\frac{|C_i^l|}{|C_i|} \geq \eta * \frac{N}{N+M} \tag{2}$$

$$|C_i| \geq MinPC \tag{3}$$

where $MinPC$ is a parameter reflecting the lower limit size of pure clusters. η is a real number and cannot exceed 1. However, the value of η cannot set to be too small.

We proposed a hierarchical semi-supervised K-means (HSK-means) based on the above mentioned definition of pure cluster. The pseudo-codes of training and test phases of HSK-means algorithm are shown in Algorithm 1. The 1-15 lines in algorithm 1 show the training process of HSK-means. The 16-30 lines in algorithm 1 show the test process of HSK-means. We define a new parameter ArsPC, reflecting the average size of clusters when HSK-means employ the clustering in the whole training set. Obviously, the number of clusters for clustering $K = |Tr| \ / \ ArsPC$. The key point is that if a cluster does not meet formula 2, then we will perform the K-cluster algorithm on this cluster recursively unless this cluster does meet formula 3, as is shown in the 10st line in algorithm 1.

For the training set, any sample in pure clusters can be extracted from the training set. We only preserve those non-pure cluster samples for the training set of the next module. This is actually a under-sampling method based on cluster. For those pure clusters, the sampling rate is zero. For those non-pure clusters, the sampling rate is one. For the test set, all samples in the pure clusters are labeled while all samples in those non-pure cluster are not labeled and are preserved for the test set of the next module.

The smaller the value of parameter $ArsPC$ is, the smaller the value of parameter $MinPC$ is, the bigger the number of pure clusters will become. If all pure clusters cover overmuch samples, the module will leave little remaining training data to next module, thereby falling into overfitting. Hence, the appropriate values of two parameters $ArsPC$ and $MinPC$ are important for the whole MSML framework. It is necessary to adjust these values of the two parameters.

Algorithm 1 MSML-PCE

```
Require: training set $Tr$, test set $Te$
Ensure: new training set $Tr_{pd}$, new test set $Te_{pd}$, labeled test set $Te_{finished}$
 1: function TREECLUSTER($Tr, Tr_{pd} MinPC, ArsPc$)
 2:     Calculate the number of clusters $K \leftarrow max(|Tr|/ArsPC, 2)$
 3:     Perform clustering on $Tr$ to obtain clusters $\{C_1, C_2, ...C_K\}$
 4:     $Tr.childern \leftarrow \{C_1, C_2, ...C_K\}$
 5:     for $i = 1 \rightarrow K$ do
 6:         if the cluster $C_i$ is a pure cluster then
 7:             Label all samples in $C_i$
 8:             Update $Tr_{pd}$ with $Tr_{pd} - C_i$
 9:         else if $|C_i| \geq MinPC$ then
10:             $TreeCluster(C_i, Tr_{pd}, MinPC, ArsPC)$
11:         end if
12:     end for
13: end function
14: $Tr_{pd} \leftarrow Tr, Te_{pd} \leftarrow \{\}, Te_{finished} \leftarrow \{\}$
15: Execute function $TreeCluster(Tr, Tr_{pd}, MinPC, ArsPc)$
16: for $\forall s_i \in Te$ do
17:     $index \leftarrow Tr$
18:     while $index$ has children do
19:         Find the nearest children $C_j$ from $s_i$
20:         $index \leftarrow C_j$
21:         if $index$ has a label $C$ then
22:             Label $s_i$ with $C$
23:             put $s_i$ into $Te_{finished}$
24:             $break$
25:         end if
26:     end while
27:     if $s_i$ has not a label then
28:         put $s_i$ into $Te_{pd}$
29:     end if
30: end for
```

3.2 Pattern Discovery (PD)

Pattern is an abstract concept. In this paper, it refers to a kind of data distribution in the feature space. Pattern can be classified in different ways. For example, pattern can be divided into known pattern and unknown pattern. The known patterns refer to all patterns that exist in the training set, while the unknown patterns refer to patterns that do not exist in the training set. For another example, pattern can also be classified into intrusion pattern and normal pattern. The intrusion patterns refer to patterns of labeled training intrusion samples, while the normal patterns refer to patterns of labeled training normal samples. In a word, pattern is an abstract description about data distribution that we wish to recognize. What we wish to recognize determines the definition of specified pattern. Then we can apply heuristic and machine learning methods to recognize it. For example, the pure cluster introduced in section 3.1 are belonged to a kind of pattern. Then we use our proposed hierarchal semi-supervised K-means algorithms to obtain the required pure clusters.

In this paper, the main criteria that determines a sample to be known or unknown is the pattern of this sample rather than the sample category. Our hypothesis about the real distribution of network traffic is that the majority of known traffic samples come from known patterns while the majority of unknown traffic samples come from unknown patterns. That means, the probability that a known traffic sample is an unknown pattern sample is very close to zero. It is so distinctive to the work in [17], [18] where the criteria is traffic category. From the point of view of patterns, [18] indicated an assumption that unknown traffic was unknown pattern and known traffic was known pattern.

This module has three sub-modules, clustering, known compensation and supervised classifier. The pseudo-codes of training and test phases of the pattern recovery module are shown in Algorithm 2. All training samples which are not located in pure clusters form this modules training set. Similarly, all test samples which are not labeled in previous module form this modules test set.

We applied a semi-supervised algorithm named K-means in the process of clustering. K-means employ a clustering in the whole training set Tr_{pd}. We define a new parameter AsPD to denote the average size of clusters . Obviously, the number of clusters for the clustering can be expressed by $K = |Tr_{pd}| \ / \ AsPD$. If the training labeled samples and the training unlabeled samples are identically distributed, a large cluster, which contains a large number of samples, will meet the equation 1 by the central limit theorem. However, if a cluster does not meet the equation 1 but it meets the following formula, we regard the cluster as a unknown pattern and label all the samples in the cluster as "new".

$$\frac{|C_i^l|}{|C_i|} < \mu * \frac{N}{M} \tag{4}$$

$$|C_i| \geq AsPD \tag{5}$$

where μ is a real number. Its value is suggested to a small number.

The second step is called known compensation. With the decrease of the value of AsPD, both TP (true positive rate) and FP (false positive rate) increase. The reason is that some known patterns are considered as unknown patterns. This can significantly reduce the precision of the unknown patterns and can increase the complexity of the internal structure of the unknown patterns. In order to maximize the increase of TP while minimizing the increase of FP, we make some compensation. The idea is to use other supervised learning algorithms to train the labeled data and calculate the confidence of each unknown patterns. Those patterns whose confidence are large enough will not be considered as unknown patterns any more. We define a confidence minimum threshold. In this paper, we use the softmax regression algorithm to calculate this confidence. Considering the softmax regression is sensible to unbalanced data, this paper presents a cluster-based under-sampling technique applying a under-sampling rate function. The larger the cluster is, the value of the under-sampling rate function is.

The third step is ”supervised classifier”. After the second step, the training set is composed of one normal category, N intrusion categories, and one unknown category. The normal and intrusion categories are all regarded as known patterns. The unknown category represents all the unknown patterns. In this paper, we use a random forest algorithm with an aim to build a supervised classifier. After this step, all the test samples are labeled. We successfully separate all the test samples into known patterns and unknown patterns. If unknown patterns are essential to perform a fine-grained classification, then next model will work.

Algorithm 2 MSML-PD

```
Require: training set $Tr_{pd}$, test set $Te_{pd}$, labeled test set $Te_{finished}$
Ensure: labeled test set $Te_{finished}$, unknown test set $Te_{new}$
 1: $Te_{new} \leftarrow \{\}, \quad Tr^s_{pd} \leftarrow \{\}, \quad Tr_{new} \leftarrow \{\}$
 2: Perform clustering on $Tr_{pd}$ to obtain clusters $\{C_1, C_2, ...C_K\}$
 3: for $i = 1 \rightarrow K$ do
 4:     if $C_i$ is a unknown pattern then
 5:         $Tr_{new} \leftarrow Tr_{new} \bigcup C_i$
 6:     end if
 7:     Generate $C^s_i \subseteq C_i$ using under-sampling rate function $func$
 8:     put $q$ into $Tr^s_{pd}, \ \forall q \in C^s_i$
 9: end for
10: Choose training labeled samples $Tr^l_{pd}$ from $Tr^s_{pd}$
11: Train a softmax classifer using $Tr^l_{pd}$
12: for all $C_i \in Te_{new}$ do
13:     if $\exists$ $category\ C$ & minium probility of $C \geq \alpha$ then
14:         $Tr_{new} \leftarrow Tr_{new} - C_i$
15:     end if
16: end for
17: Choose training labeled samples $Tr^l_{pd}$ from $Tr_{pd}$
18: Combine $Tr^l_{pd}$ with $Te_{new}$ to train a supervised classifer $f$
19: for $\forall s \in Te_{pd}$ do
20:     if $s$ is classified as $new$ by classifer $f$ then
21:         Put $s$ into $Tr_{new}$
22:     else Put $s$ into $Te_{finished}$
23:     end if
24: end for
```

3.3 Fine-grained Classification (FC)

After the processing of pattern discovery module, all the test samples are labeled. However, some samples are labeled as ”new”, which is neither a normal category nor an intrusion category, but a new category. Expert inspection is used to achieve fine-grained classification. We have high confidence to classify these samples correctly with low artificial cost because we have separated a few of unknown patterns from a group of complicated patterns. Algorithm 3 is the pseudo-code of fine-grained classification module. we perform k-means clustering on these unknown pattern samples first. For each cluster, we randomly select several samples (e.g., three samples) for manual inspection. If all the selected samples have the same ground-truth category, then we achieve fine-grained classification, as is shown in 5-9 lines in algorithm 3. The parameter $MaxFC$ indicates the number of allowed attempts.

Algorithm 3 MSML-FC

```
Require: labeled test set $Te_{finished}$, unknown test set $Te_{new}$
Ensure: labeled test set $Te_{finished}$
 1: Perform clustering on $Te_{new}$ to obtain clusters $\{C_1, C_2, ...C_K\}$
 2: for $i = 1 \rightarrow K$ do
 3:     $count \leftarrow 0$
 4:     while $count < MaxFC$ do
 5:         Randomly select $A$ samples from $C_i$
 6:         Expert inspection
 7:         if all the $A$ samples has the same ground-truth category $C$ then
 8:             Label all samples in $C_i$ with $C$; $break$
 9:         end if
10:     end while
11:     if none sample in $C_i$ is labeled then
12:         Label all samples in $C_i$ with $suspious$
13:     end if
14:     put all samples in $C_i$ into $Te_{finished}$
15: end for
```

3.4 Model Updating

In this section we continue to discuss how we can update our model. When the amount of ”new” samples is relatively large, it is possible to train a supervise model based on the samples of these unknown patterns. In this manner, a consecutive sample of the ”new” can be identified as a specific class directly by this model. Only the current distribution of network traffic generator does not change, it is effective to do so.

When the distribution varies due to going through a long period of time, introducing a feedback mechanism is necessary. If a new cluster is pure and have enough samples and the cluster does not overlap with other prepared pure clusters in feature space, then it is time to update the pure cluster extraction module. Otherwise we should update the pattern discovery module. In doing so, the model can be always able to adapt to the new traffic distribution.

4 Evaluation

4.1 Dataset

We choose the KDDCUP99 dataset to evaluate the MSML framework. The KDDCUP99 dataset contains four intrusion categories and one normal category. The four intrusion categories are DOS, R2L, U2R and Probe, respectively. Each of intrusion categories contains several of subcategories. The KDDCUP99 training dataset contains 5 categories and 22 subcategories. The KDDCUP99 test dataset contains 17 subcategories of 4 categories which do not appear in the training dataset.

Inconsistent dataset

In order to evaluate the performance of our MSML framework on non-identical distribution dataset, we construct a dataset named *inconsistent dataset*. The training dataset is composed of two parts including labeled samples and unlabeled samples. We choose a fraction of the KDDCUP99 training dataset as labeled samples due to the fact that using all of the KDDCUP99 training dataset is time-consuming. We randomly select 20 percent of the KDDCUP99 test dataset as unlabeled samples due to the fact that the training unlabeled samples and test samples need to be identically distributed according to our MSML framework. Table 1 shows the composition of the compositive training set of inconsistent dataset.

TABLE 1: Composition of Training Set

Category	Number
Normal	3242
DOS	1026
Probe	7829
U2R	52
R2L	563
Unlabeled	60020

Consistent dataset

In order to evaulate the performance of our MSML framework on identical distribution dataset, we construct a dataset named *consistent dataset*. We divide the KDDCUP99 training dataset into three subsets according to the rate of 20%, 20% and 60%, respectively. These three subsets represent training labeled samples, training unlabeled samples and test samples, respectively.

4.2 Data Pre-process

The KDDCUP99 dataset contains 41 features including nine discrete features and 32 consecutive features. We adopt one-hot and numerical-order methods to deal with discrete features. We employ one-hot when using ANN and K-means due to the fact that numerical-order can bring nonexistent sequence influence. We adopt numeric-order when using other methods because one-hot can greatly increase the data sparsity.

In this paper, 32 consecutive features are normalized. However, the values of some consecutive features (”duration, ”src_bytes”, ”num_root”,_bytes”, ”num_compromised”) show unusual value distribution. For these unusual features, the majority of the values are much smaller than the maximum value, which means 0-1 normalization will make majority of values close to zero. To avoid this situation, we adopt logarithmic normalization processing. Logarithmic normalization does not change the order of the values, but it can significantly reduce the effect of abnormal maximum value.

4.3 Evaluation Criteria

The TP, TN, FP, FN [20] are usually used to evaluate the performance of machine learning model, which can be described by a confusion matrix shown in Table 2.

The Precision, Recall, F1_score and Accuracy [20] are also defined to evaluate the model performance.

$$P = Precision = \frac{TP}{TP + FP} \tag{6}$$

$$R = Recall = \frac{TP}{TP + FN} \tag{7}$$

$$F1\_score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{8}$$

$$Accurary = \frac{TP + TN}{TP + TN + FN + FP} \tag{9}$$

In addition, we define several of evaluation indexes for MSML framework including *Capture rate*, *Coverage rate* and *Coverage capture rate*. After pattern discovery module, *Capture rate* is the proportion of all test samples which are labeled. *Coverage rate* is the proportion of all test samples which are correctly labeled. *Coverage capture rate* is the proportion of test labeled samples which are correctly labeled. We suppose the test set has M samples. After performing the operation of pattern discovery module, we classify B samples, therein, we correctly classify b samples. Apparently, the remaining $M - B$ samples are labeled ”new”, waiting for being fine-grained labeled. The calculation ways of three indexes are shown in Table 3.

4.4 Baseline Model

In order to validate the effectiveness of the MSML framework, we adopt traditional 1+N model as our compared baseline model. Our experiment based on the baseline model is carried out on both inconsistent dataset and consistent dataset.

Inconsistent dataset comparison

We adopt the labeled samples from the training set of inconsistent dataset to train the baseline model, and evaluate the baseline model on the test set of inconsistent dataset. We take use of several supervised learning algorithms, including Naive Bayes, BP neural network, random forests, support vector machines and so on. Experimental results indicate that random forest achieves the highest overall accuracy by 92.46%. Table 4 shows the confusion matrix on the test set.

Furthermore, through the deep analysis of Figure 2, we can conclude that the overall accuracy of the known traffic samples can reach 97.94%. However, the accuracy of unknown traffic samples is only 6.87%. The unknown traffic samples cause a decrease in not only overall accuracy, but also recall rate of all categories, as shown in Figure 2.

Consistent dataset comparison

An experiment with the similar method as on the inconsistent dataset is conducted on the consistent dataset.

TABLE 2: Confusion Matrix

Actual \ Predicted	Positive	Negative
Positive	TP	FN
Negative	FP	TN

TABLE 3: Self-defined Indexes

Index Name	*Capture rate*	*Coverage rate*	*Coverage capture rate*
Index Value	B/M	b/M	b/B

TABLE 4: Confusion Matrix(Baseline, Inconsistent Dataset)

Actual \ Predicted	Normal	DoS	R2L	U2R	Probe
Normal	60255	743	23	13	259
DoS	6203	223364	8	1	277
R2L	15627	0	539	17	6
U2R	134	0	8	30	56
Probe	620	154	2	0	3390

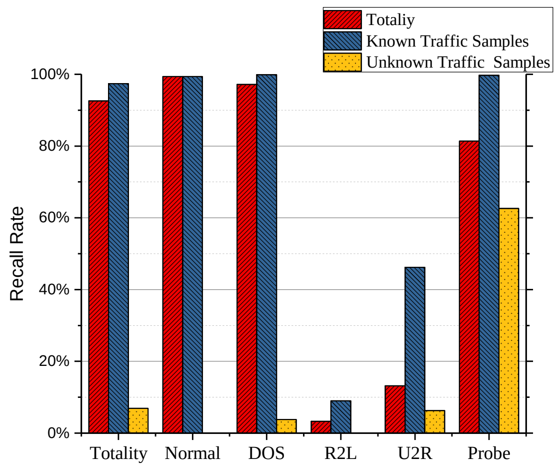


Fig. 2: Recall rate comparison of each category in baseline model

We adopt the labeled samples of the consistent training dataset to train our model. We employ this trained model to evaluate on two test set. One is the whole test set of the consistent dataset. We obtain 99.92% overall accuracy. 99/92%, which is so close to 100%, representing the high recognition capability of known pattern samples. The other is the unknown traffic portion of test set of the inconsistent dataset. We obtain 97.9% accuracy, worse than the value of 99.92%. We attribute the difference to non-identical distribution. Some known traffic samples in the test set of inconsistent dataset actually belong to unknown patterns thereby deteriorating the overall accuracy.

4.5 MSML

In addition to the overall accuracy, this paper claims that the *capture rate* and *coverage capture rate* are also significant. We make the index of *coverage capture rate* have a high value in order to reduce the influence of error classification on intrusion detection. The two indexes of *coverage capture rate* and *capture rate* are often contradictory, due to the fact that the model is apt to consider a portion of known pattern samples as unknown pattern samples. In this way, it can lead to a decrease for the index of capture rate and an increase for the index of coverage capture rate. Meanwhile, it can greatly increase the complexity of the structure of internal feature space for unknown pattern samples and increase the burden on the subsequent fine-grained classification module.

In this paper, there are several parameters which may have an important impact on the above two indexes of coverage capture rate and capture rate. For example, the average cluster size of pure cluster extraction module denoted by $ArsPC$, the lower limit cluster size of pure cluster extraction module denoted by $MinPC$, and average cluster size of pattern discovery module denoted by $AsPD$, these three parameters are of great concern to us. We set different values to these parameters in our experiments on both inconsistent dataset and consistent dataset.

Inconsistent dataset comparison

We conduct some experiments on the inconsistent dataset. The parameter $AsPD$ is set to 20, 50, and 100, respectively. Simulation experiments show that the parameter values have little effect on the results. In our work, we set $AsPD$ to value of 100. Figure 4 and Figure 3 respectively show the trend of *coverage capture rate* and *capture rate* with the change of $ArsPC$ and $MinPC$. In these figures, each circle represents a value of *coverage capture rate* and *capture rate*, the more red the circle is, the bigger the circle is, the higher the circle value becomes.

Figure 3 reflects the relationship between *coverage capture rate* and $ArsPC$, $MinPC$. From the Figure 3, we can observe that the *coverage capture rate* is generally 97% at least. In addition, with the increase of $ArsPC$ and $MinPC$, the *coverage capture rate* has a tendency of sightly increase first then significantly decrease.

Figure 4 reflects the relationship between *capture rate* and $ArsPC$ and $MinPC$. From the Figure 4, we can observe that the capture rate is relatively low when $ArsPC$ and $MinPC$ have lower values. In addition, the overall trend of *capture rate* is increasing with the increase of $ArsPC$ and $MinPC$. Furthermore, there is a certain randomness when the values of $ArsPC$ and $MinPC$ are large, therefore, when the values of $ArsPC$ and $MinPC$ are bigger than a predefined threshold, it is not necessary to continue to increase their values.

Considering both *coverage capture rate* and *capture rate*, it is proper to choose a lower $ArsPC$ and a larger $MinPC$. In this paper, the parameter $ArsPC$ is set to 100 and the parameter $MinPC$ is set to 1500. The accuracy of MSML can reach 96.6%, which has greatly improved compared with

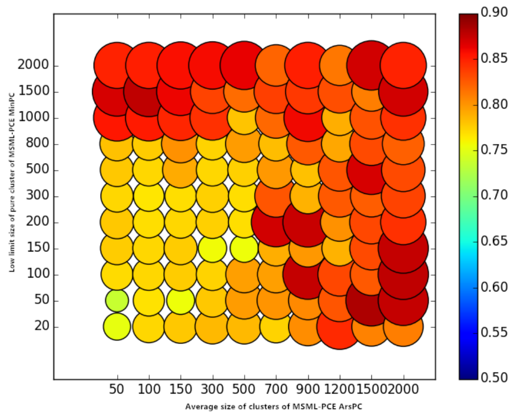


Fig. 3: MSML: coverage capture rate

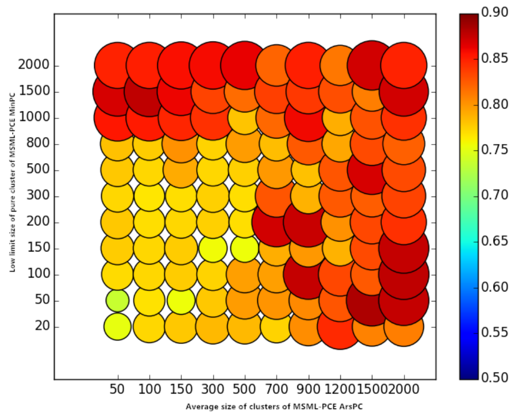


Fig. 4: MSML: capture rate

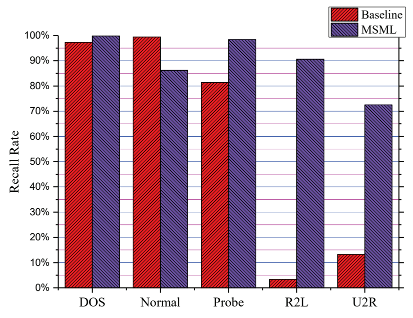


Fig. 5: MSML: Recall

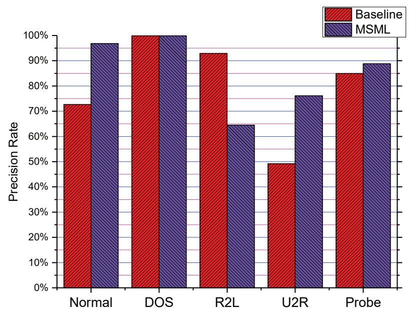


Fig. 6: MSML: Precision

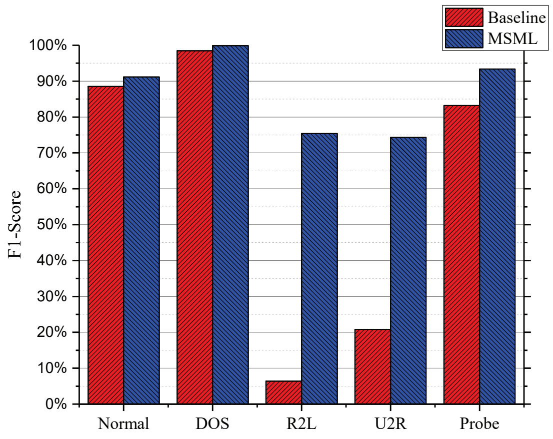


Fig. 7: MSML: F1-score

the baseline model, which can reach 92.5%. It can be seen in Table 5.

With respect to the recall rate, as is shown in Figure 5, the elephant traffic DOS has improved to some extent, while the common traffic has a great improvement. For the mouse traffic, U2R and R2L have particularly notable improvement. The recall rate of U2R and R2L have greatly increased from 13.2% to 72.5%, and 3.3% to 90.6%, respectively. With respect to the precision, as is shown in Figure 6, the precision of DOS remains 99.9%, and the precision of Normal, U2R, and R2L have improved at different degree. With respect to the F1 score, as is shown in Figure 7, the F1 score of all categories improves. The F1 score of Normal, Dos, Probe, R2L and U2R have increased from 0.885 to 0.912, from 0.985 to 0.999, from 0.832 to 0.934, from 0.064 to 0.754 and from 0.208 to 0.743, respectively.

However, we must note that the recall rate of normal and the precision rate of R2L have a descending trend. This situation should be further investigated. Another experiment is conducted. In the model updating module, we find a suspicious cluster, where mixed snmpgetattack and snmpguess of R2L with Normal samples. The number of R2L samples and the number of normal samples is about 53 to 47. Further study [9] find that, in this cluster, samples of R2L and Normal in the feature space are highly analogous so it is difficult to expect them distinguished. To confirm this conjecture , we make all the samples of the suspicious cluster randomly divided into a training set and a test set. We train a supervised classifier using the training set, and evaluate on the test set. Experimental result show that the accuracy of the test set is no higher than 53%. It is illustrated that samples of R2L and normal really cannot be separated in this kind of cluster. We take the principle of priority of the invasion. Hence, all the samples in this cluster are sentenced to R2L. This is different from the baseline experiment, where all of samples are adjudged to Normal. It is the reason

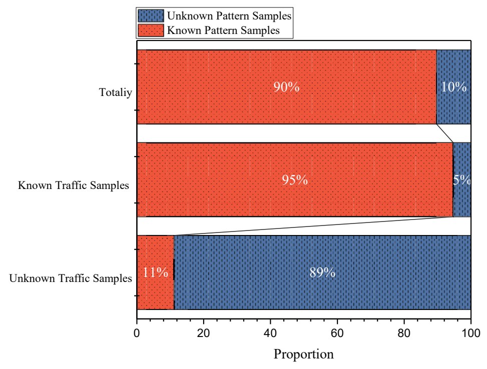


Fig. 8: Pattern in traffic

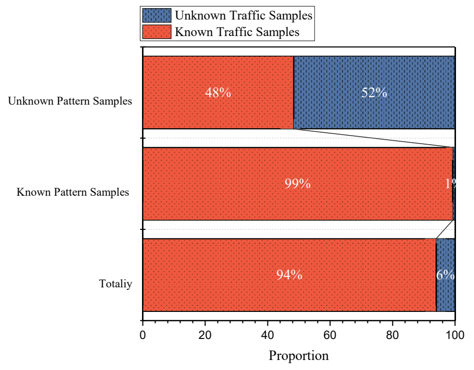


Fig. 9: Traffic in pattern

TABLE 5: Comparison

Method	Normal	DoS	Probe	R2L	U2R	Accuracy
SVM+ELM+MK [9]	98.1	99.5	87.2	21.93	31.39	95.79
NFC [21]	98.2	99.5	84.1	14.1	31.5	N/A
SVM+BIRCH [22]	99.3	99.5	97.5	19.7	28.8	95.7
MOGFIDS [23]	98.4	97.2	88.6	15.8	11.0	93.2
Association rules [24]	**99.5**	96.6	74.8	3.8	1.2	92.4
Baseline	99.4	97.2	81.4	3.3	13.2	92.5
MSML	86.2	**99.8**	**98.4**	**90.6**	**72.5**	**96.6**

why the recall rate of normal and the precision rate of R2L decrease.

Figure 8 and Figure 9 show the relationship between categories and patterns. 8 shows the ratio of known traffic to unknown traffic in the whole test dataset, in the unknown pattern samples, and in the known pattern samples respectively. It can be seen that the unknown traffic ratio of unknown pattern samples to the whole test dataset has a great increment. From Figure we can observe that there are 89 percentage of unknown traffic which is considered as unknown pattern. Meanwhile, 5 percentage of known traffic is considered as unknown pattern. Figure 9 shows the ratio of known pattern samples to unknown pattern samples in the whole test set, in the unknown traffic, and in the known traffic, respectively.

The performance comparison of MSML and other models is illustrated in Table 5. MOGFIDS [23] and baseline models are the most common "1+N" supervised learning models. Both of them have have a good detection rate in Normal and DOS, but the detection rate of U2R and R2L is very bad. Furthermore, the overall accuracy is also at a low level. Association rules [24] is a pure unsupervised learning algorithm combining heuristic rules that can hardly be identified by the rat categories. Both of [9] and [22] apply ways to identify unknown samples. Their detection rate of DOS is particularly high, and the detection rate of other categories is also at a high level. For our MSML, the detection rate of DOS, Probe, U2R and R2L and the overall accuracy are the highest. We has been analyzed that it is the defects of KDDCUP99 test set that lead to a decrease for the detection rate of Normal in MSML. Therefore, we can make the conclusion that MSML-IDS has a strong robustness.

Consistent dataset

The framework of MSML is also employed in the consistent dataset. In this dataset, whatever the values of $ArsPC$, $MinPC$, and $AsPD$ are, we obtain a capture rate of 99.95%. The rate of unknown pattern samples is less than 1/30000, which can be neglected. The experiment indicates that MSML can also be applied to distributed consistent dataset, due to the fact that MSML does not easily classify known pattern samples into unknown patterns. For the difference between training dataset and test dataset, MSML shows a good adaptability.

5 Conclusion

For many intrusion detection systems that are based on machine learning methods, there are two noteworthy factors that affect the robustness of the model: one is the class imbalance problem and the other is non-identical distribution problem. This paper presents a multi-level, semi-supervised machine learning framework(MSML) to address these two problems. The framework can effectively distinguish known pattern samples and unknown pattern samples from the whole dataset. The known pattern samples are ensured high accuracy (99.3%). With introducing a little expert inspection, the MSML framework can achieve fine-grained classification for unknown pattern samples. The overall accuracy of the test set can reach 96.6%. Moreover, the framework can also greatly improve the F1 score of those mouse traffic categories. In our future research, we will consider the optimization of unknown pattern discovery. We will also consider the distributed platforms to accelerate the speed of training the model.

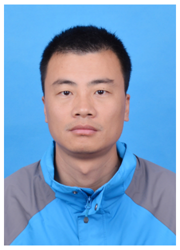
References

[1] X. Li and N. Ye, *Decision tree classifiers for computer intrusion detection*. Nova Science Publishers, Inc., 2003.

[2] T. Abbes, A. Bouhoula, and M. Rusinowitch, "Protocol analysis in intrusion detection using decision tree," in *International Conference on Information Technology: Coding and Computing, 2004. Proceedings. Itcc*, 2006, pp. 404–408.
[3] X. Y. Xiang and F. B. Zhang, "Intrusion detection system model research based on weighted multi-decision tree," *Computer Security*, 2009.
[4] M. V. Kotpalliwar and R. Wajgi, "Classification of attacks using support vector machine (svm) on kddcup'99 ids database," in *International Conference on Communication Systems & Network Technologies*, 2015, pp. 987–990.
[5] S. Singh, J. P. Singh, and G. Shrivastva, "A hybrid artificial immune system for ids based on svm and belief function," in *International Conference on Computing*, 2013, pp. 1–6.
[6] X. Han, "An improved intrusion detection system based on neural network," in *IEEE International Conference on Intelligent Computing and Intelligent Systems*, 2005, pp. 887–890.
[7] X. S. Jiang, X. M. Wei, and Y. S. Geng, "The research of intrusion detection system based on ann on cloud platform," *Applied Mechanics & Materials*, vol. 266, no. 12, pp. 2962–2965, 2013.
[8] S. Mabu, S. Gotoh, M. Obayashi, and T. Kuremoto, "A random-forests-based classifier using class association rules and its application to an intrusion detection system," *Artificial Life & Robotics*, no. 3, pp. 1–7, 2016.
[9] W. L. Al-Yaseen, Z. A. Othman, and M. Z. A. Nazri, "Multi-level hybrid support vector machine and extreme learning machine based on modified k-means for intrusion detection system," *Expert Systems with Applications*, vol. 67, pp. 296–303, 2017.
[10] G. Wang, J. Hao, J. Ma, and L. Huang, "A new approach to intrusion detection using artificial neural networks and fuzzy clustering," *Expert Systems with Applications*, vol. 37, no. 9, pp. 6225–6232, 2010.
[11] J. Visumathi and K. L. Shunmuganathan, "An effective ids for manet using forward feature selection and classification algorithms," *Procedia Engineering*, vol. 38, pp. 2816–2823, 2012.
[12] Amrita and P. Ahmed, *A Hybrid-Based Feature Selection Approach for IDS*. Springer International Publishing, 2014.
[13] Z. Wang, "The applications of deep learning on traffic identification," *BlackHat USA*, 2015.
[14] D. A. Cieslak, N. V. Chawla, and A. Striegel, "Combating imbalance in network intrusion datasets," in *IEEE International Conference on Granular Computing*, 2006, pp. 732–737.
[15] S. M. Gaffer, M. E. Yahia, and K. Ragab, "Genetic fuzzy system for intrusion detection: Analysis of improving of multiclass classification accuracy using kddcup-99 imbalance dataset," in *International Conference on Hybrid Intelligent Systems*, 2013, pp. 318–323.
[16] W. C. Lin, C. F. Tsai, Y. H. Hu, and J. S. Jhang, "Clustering-based undersampling in class-imbalanced data," *Information Sciences*, 2017.
[17] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, "Offline/realtime traffic classification using semi-supervised learning," *Performance Evaluation*, vol. 64, no. 9-12, pp. 1194–1213, 2007.
[18] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, "Robust network traffic classification," *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1257–1270, 2015.
[19] C. C. Chang and C. J. Lin, *LIBSVM: A library for support vector machines*. ACM, 2011.
[20] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *International Conference on Machine Learning*, 2006, pp. 233–240.
[21] L. He, "An improved intrusion detection based on neural network and fuzzy algorithm," *Journal of Networks,9,5(2014-05-08)*, vol. 9, no. 5, 2014.
[22] S. J. Horng, M. Y. Su, Y. H. Chen, T. W. Kao, R. J. Chen, J. L. Lai, and C. D. Perkasa, "A novel intrusion detection system based on hierarchical clustering and support vector machines," *Expert Systems with Applications*, vol. 38, no. 1, pp. 306–313, 2011.
[23] C.-H. Tsang, S. Kwong, and H. Wang, "Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection," *Pattern Recognition*, vol. 40, no. 9, pp. 2373 – 2391, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320306005218
[24] W. Xuren, H. Famei, and X. Rongsheng, "Modeling intrusion detection system by discovering association rule in rough set theory framework," in *International Conference on Computational Intelligence for Modelling, Control & Automation*, 2006, p. 24.

Haipeng Yao is currently an Associate Professor with the School of Information and Communication Engineering, and also with the State Key Laboratory of Networking and Switching Technology in Beijing University of Posts and Telecommunications. His main research interests include future network architecture, big data for networking, the architecture and next generation mobile communication systems.

Danyang Fu is a graduate student in Beijing University of Posts and Telecommunications. He received his bachelors degree in the School of Information and Communication Engineering at Beijing University of Posts and Telecommunications in 2015. His main research interests are in the area of Big Data, and future internet architecture.



Peiying Zhang is a Ph.D. candidate in School of Information and Communication Engineering, and is also with State Key Laboratory of Networking and Switching Technology at Beijing University of Posts and Telecommunications. He is currently an Associate Professor in the College of Computer & Communication Engineering, China University of Petroleum (East China). His research interests include semantic computing, deep learning, network virtualization and future network architecture.

Maozhen Li is currently a Professor in the Department of Electronic and Computer Engineering, Brunel University London. His main research interests include high performance computing, big data analytics, and knowledge and data engineering. He is a Fellow of both BCS and IET.

Yunjie Liu received his B.S degree in technical physics from Peking University, beijing, China, in 1968. He is currently an Academician of China Academy of Engineering, the Chief of the science and technology committee of China Unicom, and the Dean of the School of Information and Communications at Beijing University of Posts and Telecommunications. His current research interests include next generation networks, network architecture and management.