Hindawi
Security and Communication Networks
Volume 2020, Article ID 6309243, 12 pages
https://doi.org/10.1155/2020/6309243

Research Article

Incremental Learning for Malware Classification in Small Datasets

Jingmei Li, Di Xue, Weifei Wu, and Jiaxiang Wang

College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China

Correspondence should be addressed to Jingmei Li; lijingmei@hrbeu.edu.cn and Di Xue; dixue@hrbeu.edu.cn

Received 7 October 2019; Accepted 7 January 2020; Published 20 February 2020

Academic Editor: Leandros Maglaras

Information security is an important research area. As a very special yet important case, malware classification plays an important role in information security. In the real world, the malware datasets are open-ended and dynamic, and new malware samples belonging to old classes and new classes are increasing continuously. This requires the malware classification method to enable incremental learning, which can efficiently learn the new knowledge. However, existing works mainly focus on feature engineering with machine learning as a tool. To solve the problem, we present an incremental malware classification framework, named "IMC," which consists of opcode sequence extraction, selection, and incremental learning method. We develop an incremental learning method based on multiclass support vector machine (SVM) as the core component of IMC, named "IMCSVM," which can incrementally improve its classification ability by learning new malware samples. In IMC, IMCSVM adds the new classification planes (if new samples belong to a new class) and updates all old classification planes for new malware samples. As a result, IMC can improve the classification quality of known malware classes by minimizing the prediction error and transfer the old model with known knowledge to classify unknown malware classes. We apply the incremental learning method into malware classification, and the experimental results demonstrate the advantages and effectiveness of IMC.

1. Introduction

Malware (or malicious software) is defined as computer software that is not intended to be executed by a user, has a malicious purpose, and completes a malicious function [1]. Currently, given the popularity and widespread use of the Internet, human malware inspectors can find thousands of malware instances every day, but the emergence of various automated tools has shown that the speed with which malware mutates on the Internet is faster than people realized. New malware samples and new malware classes are increasing continuously. However, traditional machine learning-based malware classifiers have catastrophic forgetting problems [2], which cannot retain learning memory of old classes while classifying new classes. This requires the malware classification with incremental learning, which makes the classifier model improve the classification performance continuously through the learning with self-growth.

Incremental learning [3] is one of the meaningful methods to continuously learn data, which keeps the previous learning results and learns only for the new data. The incremental learning is to improve the classification quality of known and unknown malware classes based on new available training samples. In this paper, we focus on the incremental learning for malware classification. Incremental learning often encounters one challenging problem, that is, stability-plasticity dilemma: retaining the knowledge of the old model and conserving the new knowledge. In this paper, incremental learning is to add new classification planes for new classes to the existing classification model or update the existing classification model, which can find the trade-off between stability and plasticity. To the best of our knowledge, the current research on incremental learning mainly focuses on target recognition [4] and image classification [5]. Few studies have studied the incremental learning of malware classification.

To solve the problem for malware classification, this paper proposes an incremental learning method IMCSVM, which is based on SVM. IMCSVM can realize incremental learning, which learns new knowledge while retaining the knowledge of the old model, no matter whether the new knowledge belongs to old or new classes. For new knowledge

of old classes, IMCSVM constrains the target model to keep it close to the old model. And for new knowledge of new classes, IMCSVM constrains the new classification plane to keep it close to the linear combination of existing classification planes. Based on the proposed IMCSVM, we propose an incremental learning framework for malware classification, named "IMC." First, IMC extracts opcodes from malware samples and transforms them into n-gram opcode sequences. These opcode sequences represent the rich semantic information of malware samples. For malware classification, not all opcode sequences are useful. Therefore, IMC further uses term frequency-inverse document frequency (TF-IDF) to select important opcode sequences. Finally, these selected opcode sequences and the frequency values are inputted into IMCSVM. IMCSVM adjusts existing classification planes for learning new knowledge of old classes and adds new classification planes for learning new knowledge of new classes. In summary, the contributions of our work can be summarized as follows:

(1) We propose an incremental learning method based on SVM, which solves the problem that the stability of the malware classifier is too strong and the plasticity is too poor.

(2) We propose a unified incremental learning method for new knowledge of old classes and new classes, which can adjust existing classification planes and add new classification planes at the same time.

(3) We apply the incremental learning method into malware classification, and the experimental results demonstrate the advantages and effectiveness of IMC.

The remainder of this paper is organized as follows: Section 2 introduces the related work. Section 3 briefly introduces the method architecture of the proposed incremental malware classification framework. Section 4 presents the incremental learning method for the task of malware classification in detail. The experimental results and analysis are presented in Section 5. Finally, Section 6 summarizes this paper and proposes the future work.

2. Related Works

In this section, we present the related work regarding malware classification and incremental learning.

2.1. Malware Classification. So far, malware classification methods mainly focus on feature engineering, which need to extract features from malware or visualization images, for example, API calls [6, 7], system calls [8, 9], and opcode sequences [10, 11]. The malware classification framework in our work is closely related to opcode sequences and SVM. In each instruction obtained by disassembling malware, we make use of the opcodes but ignoring the operands. Opcodes indicate what kind of operation the instruction should perform. For example, Santos et al. [12] propose a malware detection framework, which is based on n-gram opcode sequences. They discuss the accuracy and time results of data-mining classifiers for 1-gram opcodes, 2-gram opcodes, and the combination of 1-gram and 2-gram opcodes. Besides feature selection, they also discuss the advantages and disadvantages of data-mining classifiers, which include decision trees, support vector machines, K-nearest neighbors, and Bayesian networks. Bai et al. [10] incorporate three features, which include byte n-grams, opcode n-grams, and format information, and exploit complementary information of these features to detect the true nature of a program. They evaluated the generalization ability of the proposed method in two datasets. The experimental results show that their method reduces the false alarm rate to 0%. Raphel et al. [13] extract three features, named branch opcodes, unigram opcodes, and bigram opcodes, use Naïve Bayes, term frequency, and document frequency to compute the importance of these features, and then select the significant features by two feature reduction techniques: discriminant feature variance-based approach and Markov blanket. They also incorporate the three features to detect the metamorphic malware samples.

Compared with n-gram opcode sequence selection method, O'Kane et al. [14] develop a lightweight filter to identify feature reduction based on the SVM model and opcode histograms, which run on the virtual machine for dynamic analysis. They use the SVM model to perform an exhaustive search by traversing through all the opcode sequences, which can filter out irrelevant features and reduce the computational overhead required when n-gram opcode sequence analysis is performed on low-level data.

In addition to converting to n-grams, opcode sequences can also be converted to a sequential pattern. Fan et al. [15] use MSPE (malicious sequential pattern extraction) to mine the effective malicious sequential patterns from instruction sequences (opcode sequences), instead of n-gram features of opcode sequences. They introduce the concept of objective-oriented programming in MSPE to learn significant patterns with strong abilities to classify malware and benign files.

Supervised machine learning models need a large amount of labeled malware and benign software. Santos et al. [16] propose a single-class learning method to detect unknown malware. Based on opcode sequences with TF-IDF, they only label malware or benign software and use Rocchio-support vector machines to learn opcode sequence features to identify the true nature of software. And then, they study whether it is better to label malicious or benign software and how the number of labeled samples affects the final accuracy of models.

With the development of deep learning, Yuxin et al. [11] try to use the deep belief network (DBN) to learn n-gram opcode sequences. Compared with traditional machine learning, DBNs can pretrain a multilayer generative model by using unlabeled data, which can better represent the features of malware. They use document frequency and information gain to select 3-gram opcode features.

SVM [17] is a machine learning algorithm based on VC dimension theory and structural risk minimization theory of statistical learning theory. Standard SVM model is mainly used for 2-classification problems to find the best separating hyperplane in the feature space so as to maximize the

interval between positive and negative samples on the training set.

In addition to learning opcode sequences, some researchers also use SVM to learn other characteristics. For example, API sequences extracted from software can be used as the input for SVM [18, 19]. Salehi et al. [20] use support vector machine based on recursive feature elimination to detect activity of software based on API calls and their arguments and return values, in which each software is run in a controlled environment. Nissim et al. [21] dynamically record the activities of computer worms and use the combination of chi-square, gain ratio, and relief as the features of worms. These features are input into SVM to detect computer worms. Ali Mirza et al. [22] use the refined features extracted from the software to create training data by considering each feature as boolean. And then, they integrate SVM and decision trees learning these training data to detect malware. Teng et al. [23] consider objects, roles, agents, and groups of intrusion detection and use SVM-decision trees to learn these extraction features.

2.2. Incremental Learning. The goal of incremental learning is learning new training samples to improve the classification quality. However, incremental learning is more used in target recognition [4] and image classification [5] and less used in the field of information security. For example, Du et al. [24] present the incremental support vector machine learning algorithm based on clustering for intrusion detection. Elham et al. [25] propose an incremental ensemble learning for spam filtering. Myint and Meesad [26] propose an incremental learning algorithm based on support vector machine with the Mahalanobis distance for intrusion prevention, which builds an ellipsoidal kernel instead of a sphere kernel by using the Mahalanobis distance, and the concept of handling the covariance matrix by dividing by zero. Yi et al. [27] develop an improved incremental SVM algorithm for network intrusion detection, which uses a modified kernel function that embeds the mean and mean square difference values of feature attributes and uses the reserved set strategy to solve the oscillation problem of the traditional incremental SVM's follow-up learning process. Noorbehbahani et al. [28] propose a new semisupervised stream classification method for intrusion detection, which is capable of incremental updating using limited labeled data. The method can operate in real time and learn new intrusions by clustering and instance-based learning. Chen et al. [29] develop a classifier using an artificial immune system combined with population-based incremental learning and collaborative filtering for network intrusion detection. The classifier can adapt the weight of key features and adjust antibody hierarchy using mean affinity, so as to improve incremental learning capability.

To the best of our knowledge, only a few incremental learning methods have been applied in the field of malware detection. Nissim et al. [30] present a novel method for malware detection, which can detect new malware and enhance the detection capabilities over time. In this method, the malware is transformed into the vector, which is an input of the detection model based on SVM and active learning. Zhuang et al. [31] propose an incremental learning algorithm for malware detection, which preserves the support vectors obtained by handling old data with SVM.

3. Method Architecture

Figure 1 shows the method architecture of the proposed incremental malware classification (IMC) framework, which consists of three major components: opcode sequence extractor, opcode sequence selector, and incremental multiclass SVM (IMCSVM) classifier, for malware classification. We briefly describe each component below.

(1) Opcode sequence extractor: IMC first extracts opcodes from malware samples and transforms them into n-gram opcode sequences.

(2) Opcode sequence selector: in this component, the TF-IDF algorithm is applied to select the representative opcode sequences.

(3) IMCSVM classifier: in this module, the input malware is transformed into vectors based on the opcode sequence and frequency. Then, the proposed incremental classifier SVM is used to conduct malware classification and update IMC.

The detailed processes of the three components will be presented in Sections 3.1 and 4, respectively.

3.1. Opcode Sequence Feature Extraction and Selection. In order to use machine learning to classify malware, the malware samples must be transformed into the expected input representation. In this paper, n-gram [32] opcode sequences and their frequency were used to represent malware. First, IDA Pro [33] as a disassembly tool was used to extract opcodes from each malware. The logical order for extracting the opcode sequences is the same as that for executing malware. Second, the n-gram ($n = 2$, 3, and 4) opcode sequences were extracted. Figure 2 shows the extraction process of n-gram opcode sequences.

Specifically, malware χ is defined as a set of ordered opcodes o, $\chi = (o_1, o_2, o_3, o_4, \ldots, o_{m-1}, o_m)$, where m is the number of opcodes. An opcode sequence s is defined as an ordered subgroup of opcodes within the malware χ, where $s \subseteq \chi$ and the length of s is 2, 3, and 4. It is made up of ordered opcodes o and $s = (o_i, o_{i+1}) \vee (o_i, o_{i+1}, o_{i+2}) \vee (o_i, o_{i+1}, o_{i+2}, o_{i+3})$.

In our previous study [7], it has been demonstrated that the opcode sequences captured by variable-length n-grams ($n = 2$, 3, 4) are more meaningful than fixed-length n-grams.

In this paper, malware that cannot be disassembled is not within our scope. This malware refers to confused malware that cannot be unpacked by existing anti-unpacking tools, such as UPX, and cannot be disassembled by IDA Pro.

The number of distinct n-grams extracted for the opcode representation was 27849, 309,712, and 1,364,559, for 2-grams, 3-grams, and 4-grams, respectively. There is a lot of redundant information in these features, so some feature

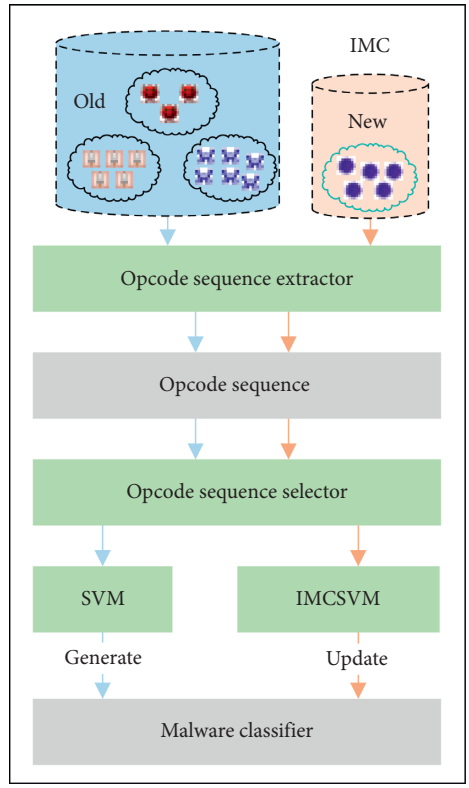

Figure 1: Method architecture of the proposed incremental malware classification framework.

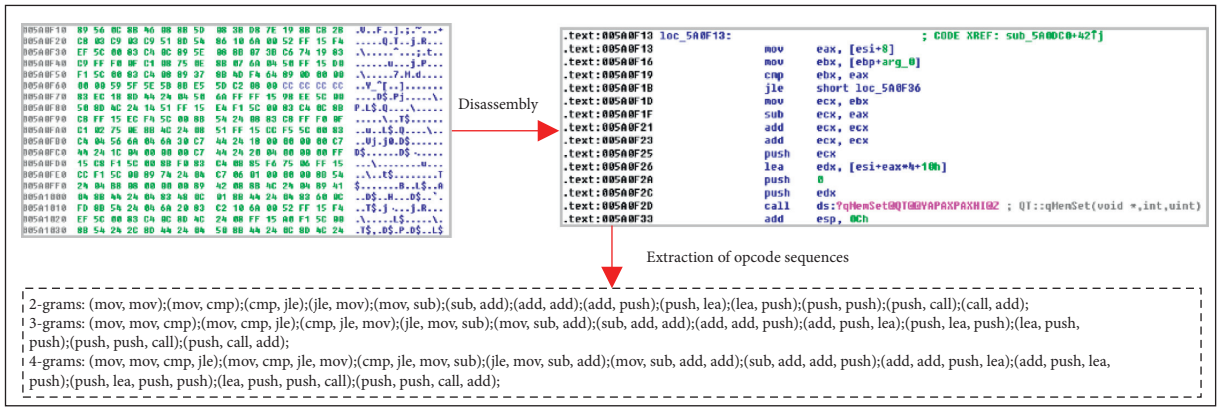

Figure 2: Extraction process of n-gram opcode sequences.

selection methods need to be used to select the most important opcode sequence. In this paper, TF-IDF [34] was used as the weight of each opcode sequence:

$$w_{i,k} = \mathrm{TF-IDF}_{i,k} = \mathrm{TF}_{i,k} \times \mathrm{IDF}_i = \frac{n_i}{\sum_k n_k} \times \log\left(\frac{D}{d_i + 1}\right), \tag{1}$$

where n_i is the number of times the opcode sequence s_i appears in the family k, $\sum_k n_k$ is the total number of opcode sequences in the family k, D is the total number of malware, and d_i is the number of malware containing the opcode sequence n_i.

Based on the TF-IDF of each n-gram feature, the importance of each n-gram feature can be assessed. The top t n-gram features in each malware family were selected according to the descending order of TF-IDF. The weight of each n-gram feature is $w_{i,k}$.

Finally, a vector $\mathbf{v}$ was obtained, which is composed of opcode sequences and their frequencies and given by $\mathbf{v} = \{(s_{i,k}, w_{i,k})\}$, where $s_{i,k}$ is the opcode sequence in the family k and $w_{i,k}$ is the TF-IDF for $s_{i,k}$.

4. Incremental Learning Method

4.1. Incremental Multiclass SVM (IMCSVM). First, we briefly review the standard least-squares SVM (LSSVM) [35] for two classification tasks. The aim of the LSSVM is to find a suitable hyperplane that separates two types of

samples. The hyperplane should satisfy the maximum margin between the support vector and the hyperplane. And the LSSVM can replace the QP problem of SVM by solving linear equations, which greatly facilitates the solution of the Lagrange multiplier α. Given a training sample for two classification tasks,

$$\left\{(x_k, y_k) \,\middle|\, k = 1, \ldots, N, \quad x_k \in \mathbb{R}^n,\ y_k \in \{+1, -1\}\right\}. \tag{2}$$

A standard LSSVM solves the following optimization:

$$\begin{aligned} &\min_{\omega,\mathrm{b},\mathrm{e}} \quad W(\mathbf{w}, \mathbf{b}, \mathbf{e}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{k=1}^{N} e_k^2, \\ &\text{s.t.} \qquad y_k\left[\mathbf{w}^T\phi(x_k) + \mathbf{b}\right] = 1 - e_k, \quad k = 1, \ldots, N. \end{aligned} \tag{3}$$

In this paper, malware classification is a multiclass classification task. Given a training sample,

$$\left\{(x_k, y_k) \,\middle|\, k = 1, \ldots, N, \quad x_k \in \mathbb{R}^n,\ y_k \in \{1, \ldots, M\}\right\}. \tag{4}$$

For a multiclass task, we can minimize the following objective function based on standard LSSVM:

$$\begin{aligned} &\min_{w,\mathrm{b},\mathrm{e}} \quad W(\mathbf{w}, \mathbf{b}, \mathbf{e}) = \frac{1}{2}\sum_{i=1}^{M} w_i^T w_i + C\sum_{i=1}^{M}\sum_{k=1}^{N} e_{ik}^2, \\ &\text{s.t.} \qquad y_k\left[w_i^T\phi(x_k) + b_i\right] = 1 - e_{ik}, \\ &\qquad\qquad\qquad i = 1, \ldots, M,\ k = 1, \ldots, N, \end{aligned} \tag{5}$$

where w_i represents a hyperplane of one-vs-rest, $\phi(x_k)$ represents a kernel function, e_{ik} is a slack variable, C represents the trade-off between the expected risk and the complexity, i is the class number, and k is the sample number. And we use body letters for matrices.

In this paper, the incremental learning method not only needs to learn new data from new classes but also has the ability to learn new data from old classes. Before introducing the proposed incremental learning method, several variables need to be described. D is the given dataset, D_r is the retired dataset, and D_l is the remaining dataset. D_l is also the support vector. The relationship between the three is as follows:

$$D = D_r \cup D_l. \tag{6}$$

D_a is the newly arrived data, and $\widetilde{D}$ is the dataset after adding new data:

$$\widetilde{D} = D_l \cup D_a. \tag{7}$$

When training an SVM classifier, we assume that the training is completed in step s, and the trained classifier has the following parameters:

$$\begin{aligned} \mathbf{w}^s &= \left[w_1^s, w_2^s, \ldots, w_M^s\right], \\ \mathbf{b}^s &= \left[b_1^s, b_2^s, \ldots, b_M^s\right]. \end{aligned} \tag{8}$$

The new data were added in step $(s+1)$, which need to update the following parameters:

$$\begin{aligned} \mathbf{w}^{s+1} &= \left[w_1^{s+1}, w_2^{s+1}, \ldots, w_M^{s+1}\right] \cup w_{M+1}^{s+1}, \\ \mathbf{b}^{s+1} &= \left[b_1^{s+1}, b_2^{s+1}, \ldots, b_M^{s+1}\right] \cup b_{M+1}^{s+1}. \end{aligned} \tag{9}$$

In this paper, the following two aspects are mainly considered in the malware classification task:

(1) Adding the new data from old classes: at this point, D_l and new data are used to adjust the classification plane so that the adjusted plane can describe $\widetilde{D}$.

$$w_i^{s+1} = w_i^s + \Delta\mathbf{w}, \quad i = 1, \ldots, M. \tag{10}$$

(2) Adding the new data from new classes: when a new class is added, not only the classification planes of the old model need to be adjusted, but also a new classification plane should be added to learn the new class. At this point, $\mathbf{w}^{s+1}$ is close to the source plane to make the M-class classifier transfer to an $(M+1)$-class classifier. In addition, the classification task also requires an $(M+1)$-class classifier to protect the old knowledge while learning the new knowledge.

$$\mathbf{w}^{s+1} = \left[w_1^{s+1}, w_2^{s+1}, \ldots, w_M^{s+1}\right] \cup w_{M+1}^{s+1}. \tag{11}$$

Therefore, for the classification task in this paper, the objective function of LSSVM was improved to implement a malware classifier with incremental learning:

$$\begin{aligned} \min_{\mathrm{w},\mathrm{b},\mathrm{e}} W\left(\mathbf{w}^{s+1}, \mathbf{b}^{s+1}, \mathbf{e}^{s+1}\right) &= \frac{1}{2}\underbrace{\sum_{i=1}^{M}\left(w_i^v\right)^T\left(w_i^v\right)}_{\widetilde{V}} + \frac{1}{2} \\ &\quad \cdot \underbrace{\sum_{i=1}^{M}\left(w_i^r\right)^T\left(w_i^r\right)}_{\widetilde{R}} + C^{s+1}\underbrace{\sum_{i=1}^{M+1}\eta_i^{s+1}}_{\widetilde{L}}, \\ w_i^v &= w_i^{s+1} - w_i^s, \\ w_i^r &= w_{M+1}^{s+1} - \varepsilon^{s+1} w_i^s, \\ \eta_i^{s+1} &= \underbrace{\sum_{a\in D_a}\left(e_{ia}^{s+1}\right)^2}_{\widetilde{L}_1} + \lambda^{s+1}\underbrace{\sum_{l\in D_l}\left(e_{il}^{s+1}\right)^2}_{\widetilde{L}_2}, \end{aligned} \tag{12}$$

where w_i^s and w_i^{s+1} represent an old and a new classification plane of one-vs-rest in the M-class classifier, w_{M+1}^{s+1} is a new classification plane that separates the new class and the learned classes, ε^{s+1} is a real number, and λ^{s+1} is the weight of old knowledge.

In the proposed incremental learning method, $\widetilde{V}$, $\widetilde{R}$, and $\widetilde{L}$ have the following meanings:

(1) $\widetilde{V}$ makes the classifier protect the old knowledge while learning the new knowledge.

(2) For the new data from new classes, $\widetilde{R}$ can add the new knowledge of new classes into the old model and transform the old model into the target model.

(3) The loss function is defined as $\widetilde{L}$. Among them, $\widetilde{L}_1$ is the prediction error of minimizing D_a and $\widetilde{L}_2$ is the prediction error of minimizing D_l. λ^{s+1} is a weight, which balances the variation of the prediction error in D_l.

4.2. Updating $\mathbf{w}^{s+1}, \mathbf{b}^{s+1}$, *and* $\mathbf{D}_l$. For equation (12), the Lagrange multiplier method can be used to obtain the dual problem. The Lagrange function of equation (12) can be written as

$$L\left(\mathbf{w}^{s+1}, \mathbf{b}^{s+1}, \mathbf{e}^{s+1}; \boldsymbol{\alpha}^{s+1}\right) = W\left(\mathbf{w}^{s+1}, \mathbf{b}^{s+1}, \mathbf{e}^{s+1}\right) - \sum_{i=1}^{M+1} \sum_{k \in \widetilde{D}} \alpha_{ik}^{s+1} \left\{ y_k \left[\left(w_i^{s+1}\right)^T \phi\left(x_k\right) + b_i^{s+1} \right] - 1 + e_{ik}^{s+1} \right\}, \tag{13}$$

where α is the Lagrange multiplier, $\alpha \geq 0$.

Let the partial derivative of $L(\mathbf{w}^{s+1}, \mathbf{b}^{s+1}, \mathbf{e}^{s+1}; \boldsymbol{\alpha}^{s+1})$ to $\mathbf{w}^{s+1}, \mathbf{b}^{s+1}$, and $\mathbf{e}^{s+1}$ be zero, respectively:

$$\begin{aligned}
&\frac{\partial L}{\partial \mathbf{w}^{s+1}} = 0 \longrightarrow w_i^{s+1} - w_i^s = \sum_{k \in \widetilde{D}} \alpha_{ik}^{s+1} \phi\left(x_k\right), \\
&\frac{\partial L}{\partial \mathbf{b}^{s+1}} = 0 \longrightarrow \sum_{k \in \widetilde{D}} \alpha_{ik}^{s+1} = 0, \\
&\frac{\partial L}{\partial \mathbf{e}^{s+1}} = 0 \longrightarrow 2C^{s+1} \left(\sum_{a \in D_a} e_{ia}^{s+1} + \lambda^{s+1} \sum_{l \in D_l} e_{il}^{s+1} \right) + \sum_{k \in \widetilde{D}} \alpha_{ik}^{s+1} = 0, \\
&\frac{\partial L}{\partial \boldsymbol{\alpha}^{s+1}} = 0 \longrightarrow y_k \left[\left(w_i^{s+1} \phi\left(x_k\right) + b_i^{s+1} \right] - 1 + e_{ik}^{s+1} = 0.
\end{aligned} \tag{14}$$

Furthermore, the kernel function is defined as $\kappa = \varphi(x_{k_1})^T \varphi(x_{k_2})$, and the kernel matrix is defined as $\mathbf{K} = \phi(\mathbf{X})^T \phi(\mathbf{X})$. Thus, the final result of the optimization can be obtained by

$$w_i^{s+1} = w_i^s + \sum_{k \in \widetilde{D}} \mathbf{T}_{ik}^{s+1} \phi\left(x_k\right), \quad i = 1, \ldots, M, \tag{15}$$

$$w_{M+1}^{s+1} = \boldsymbol{\varepsilon}^{s+1} \mathbf{w}^s + \sum_{k \in \widetilde{D}} \mathbf{T}_{(M+1)k}^{s+1} \phi\left(x_k\right), \tag{16}$$

$$\mathbf{b}^{s+1} = \boldsymbol{\Gamma} - \mathbf{O}\mathbf{O}^T \boldsymbol{\varepsilon}^{s+1}, \tag{17}$$

where

$$\mathbf{T}^{s+1} = \mathbf{H} - \mathbf{Z}\mathbf{Z}^T \boldsymbol{\varepsilon}^{s+1},$$

$$\begin{bmatrix} \mathbf{H} & \mathbf{Z} \\ \boldsymbol{\Gamma}^T & \mathbf{O}^T \end{bmatrix} = \begin{bmatrix} \mathbf{K} + \left[\frac{1}{2C^{s+1}} \quad \frac{1}{2C^{s+1}\lambda^{s+1}} \right]^T & \mathbf{1} \\ \mathbf{1} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{Y} & \phi(\mathbf{X})^T \mathbf{w}^s \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{18}$$

where $\mathbf{Y}$ is the label matrix of samples and $\mathbf{T}_{ik}^{s+1}$ is the (i, k) entry of $\mathbf{T}^{s+1}$ in step $(s+1)$.

In IMC, when new data arrive, new data and support vectors of old data are input into IMCSVM. Then, the parameters of IMCSVM are updated by equations (15)–(17) so that IMC can learn new knowledge without forgetting old knowledge.

5. Experimental Results and Analysis

In this section, we first present the experimental setup, which includes the datasets, and the details about the implementation and evaluate criteria. Then, we evaluate the incremental learning ability for the new data from old classes and then for the new data from new classes. Finally, we compare the classification performance between different open-source datasets.

5.1. Data Description. The experimental data in our paper are from VirusShare (https://virusshare.com/) and VX Heavens (https://83.133.184.251/virensimulation.org/index.html). We randomly selected 4 data subsets from VX Heavens, named S1 to S4, and selected 2 data subsets from VirusShare, named S5 and S6. Each data subset contains 20 malware classes, and each class contains 600 malware samples. Any two data subsets do not intersect. In each class of a data subset, we divided 600 malware samples into 500 training samples and 100 testing samples.

5.2. Implementation. In the section, we presented the implementation setup in terms of the following four aspects: (1) the parameter setting of IMCSVM, (2) the baselines of experiments, (3) the design of experiments, and (4) the experiment environment.

Firstly, the parameter setting in the IMCSVM includes C, λ, and the parameter of the RBF kernel. For finding the best parameter values for malware classification, we use 10-fold cross validation in our experiments. These setting parameters are shown in Table 1.

Secondly, for the reason that IMCSVM is an incremental learning method, we compared our model with the five different baselines about incremental learning. For the new data from old classes, we compared the IMC with the All data, RS-ISVM for multiclass classification [27], standard ISVM for multiclass classification [3], and IRotBoost [24]. For the new data from new classes, we compared the IMC with the All data, CIELM [36], MULTIpLE [37], and SVMF-RTST [5]. The All data method trains an SVM model from scratch for multiclass classification using all new and old data when IMC needs to identify a new class.

Thirdly, in the experiment, we used all training samples of the first 5 classes to train a source model and the other classes as the new data from new classes to train incremental learning, and then we used the first 50 samples of all class to train a source model and the other samples as the new data from old classes to train incremental learning.

Lastly, SVM models were created and trained in Python. And our experiments were performed on a PC with Windows 10 v1809, whose configuration is Inter (R) Core i7-7700@

Table 1: Temperature and wildlife count in the three areas covered by the study.

Parameters	Setup
C	[0.001, 100]
λ	$4 \times$ (Number of old data before step $s+1$)/ number of new data at step $s+1$
RBF kernel	$[2^{-13}, 2^{-6}]$

3.6 GHz CPU, NVIDIA GeForce GTX 1080 GPU, 16 GB RAM, and DDR5.

5.3. Evaluate Criteria. We evaluated the classification performance of IMC using precision, recall, $F1$ score, and accuracy. The four evaluation criteria were calculated as

$$\begin{aligned} \text{precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}}, \\ \text{recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}}, \\ F1\ \text{score} &= \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}, \\ \text{accurancy} &= \frac{\text{correctly classified samples}}{\text{total number of samples}}, \end{aligned} \tag{19}$$

where the true positive (TP) is the number of samples belonging to the family k that are correctly labeled to belong to the family k; the false positive (FP) is the number of samples not belonging to the family k that are erroneously labeled to belong to the family k; and the false negative (FN) is the number of samples belonging to the family k that are erroneously labeled to belong to other families.

5.4. The Performance of IMC for the New Data from Old Classes. For evaluating the performance of IMC for the new data from old classes, we used the first 50 samples of all class to train a source model and the model learned 50 malware samples from training samples of every class for each incremental learning. The experimental results are shown in Figures 3 and 4. The change from node n to node $n+1$ can be regarded as one incremental learning process. Node n is the old model for old data from all classes, and node $n+1$ is the new model for old and new data from all classes.

As shown in Figures 3 and 4, we compared IMC with four different incremental learning for the new data from old classes. Since class imbalance is not the goal of this paper, we add the same number of samples to each class in each incremental learning. In Figures 3 and 4, it can be seen that the classification performance of all methods increased with the increase of incremental learning times. After a certain incremental learning, the training samples basically included all the important features of the classes, and after that, the incremental learning did not significantly affect the classification performance. The performance of the All data method, which used All data to retrain the model, was slightly higher than that of IMC. The reason is that the IMC method was based on the old model, which only uses the new data from old classes to adjust the existing classification plane. However, the training time of IMC was significantly lower than that of the All data method. In IMC, the training time for each incremental learning process was basically unchanged because the training samples for each incremental learning process are the same. With the increase of data, the training time of the All data method increased gradually, and the training time was multiplied with the training time of IMC. The performance of IMC was higher than that of other three incremental learning methods. RS-ISVM normalizes the data type by modifying the kernel function, which only modifies the data type and does not optimize the loss balance between new data and old data. IRotBoost uses ensemble learning to learn new data, which only improves the learning performance of new data and ignores the adjustment of old data. Therefore, the classification performance of these two methods is lower than that of IMC. Standard ISVM uses the most basic learning method to adjust the classification plane, and its performance is significantly lower than that of the other four methods.

5.5. The Performance of IMC for the New Data from New Classes. For evaluating the performance of IMC for the new data from new classes, we used all training samples of the first 5 classes to train a source model and the model learned all training samples from one of the other classes for each incremental learning. The experimental results are shown in Figures 5 and 6. The change from node n to node $n+1$ can be regarded as one incremental learning process. Node n is the old model for old classes, and node $n+1$ is the new model for old and new classes.

As shown in Figures 5 and 6, when IMC learns new classes, its performance does not decline significantly, and the learning results of each time tend to be stable. The performance of the All data method, which used All data to retrain the model, was slightly higher than that of IMC. The reason is that the IMC method was based on the old model, which only uses the new data from new classes to adjust the existing classification plane. However, the training time of IMC was significantly lower than that of the All data method. In IMC, the training time for each incremental learning process was basically unchanged because the training samples for each incremental learning process are the same. With the increase of classes, the training time of the All data method increased gradually. Except the All data method, the performance of IMC was higher than that of other three incremental learning methods. This is because the MULTIpLE and SVMF-RTST methods only solve the loss of new data from new classes while ignoring to adjust the old model when learning new data. At this time, the loss of old data from the old model also needs to be adjusted. The CIELM method learns new data from new classes by adding new nodes on the basis of the old model. This method does not consider the association between class nodes. When learning

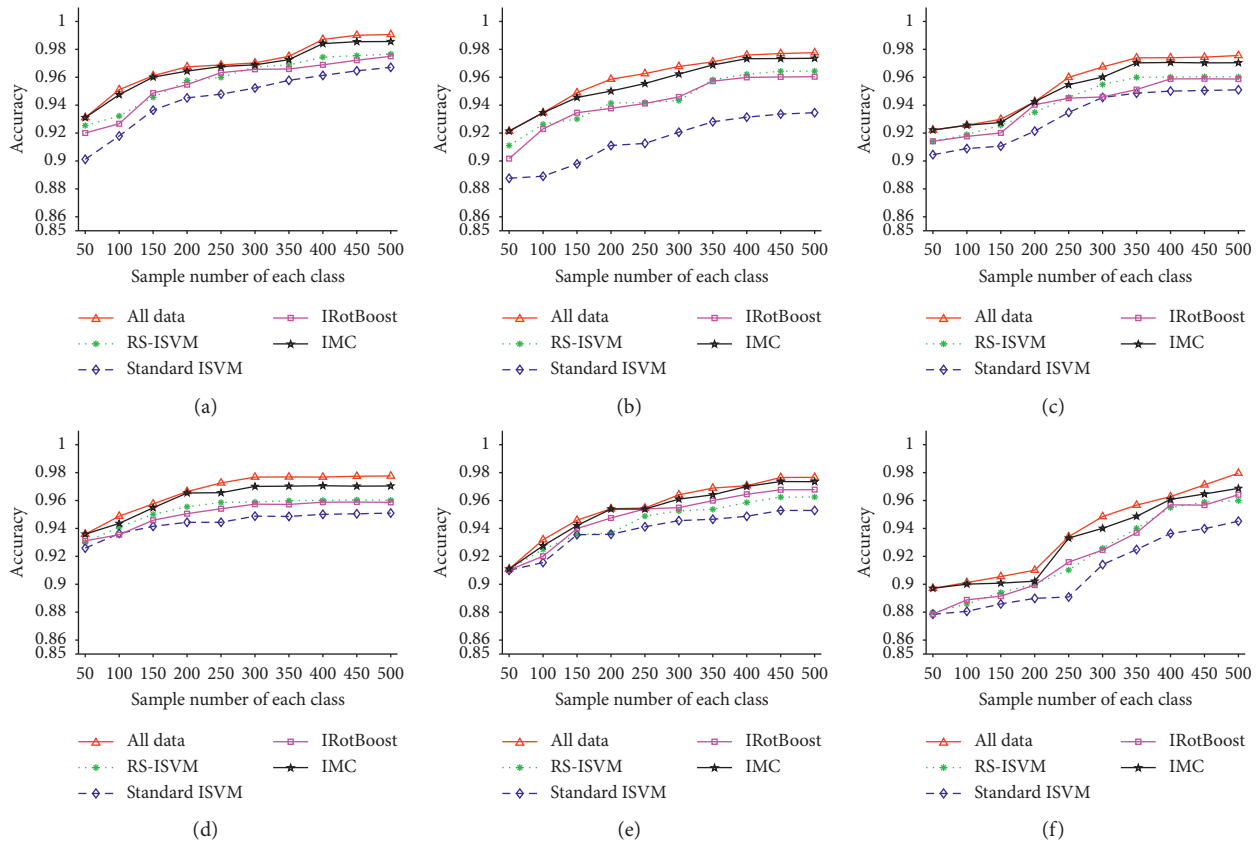

Figure 3: Accuracy comparison of five different incremental learning methods for the new data from old classes. (a) S1. (b) S2. (c) S3. (d) S4. (e) S5. (f) S6.

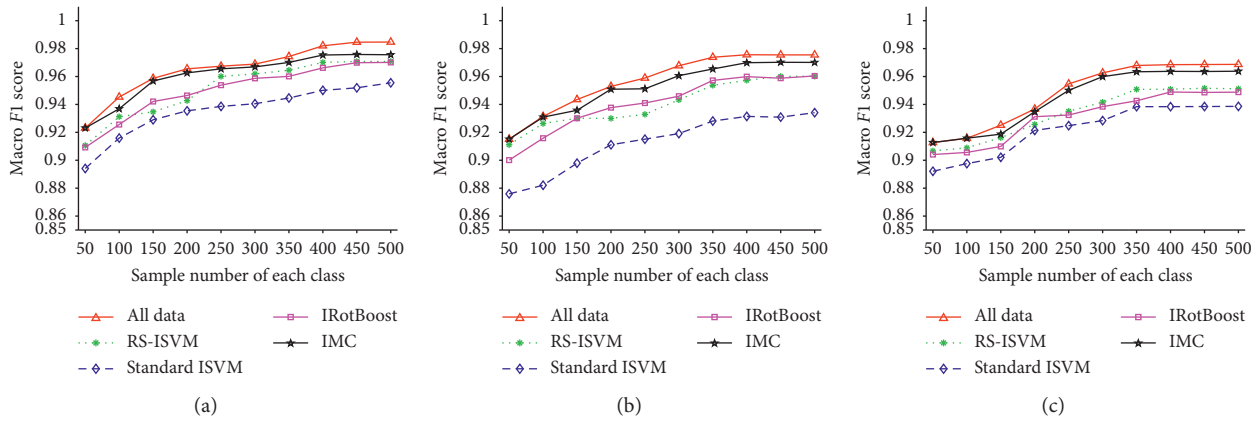

Figure 4: Continued.

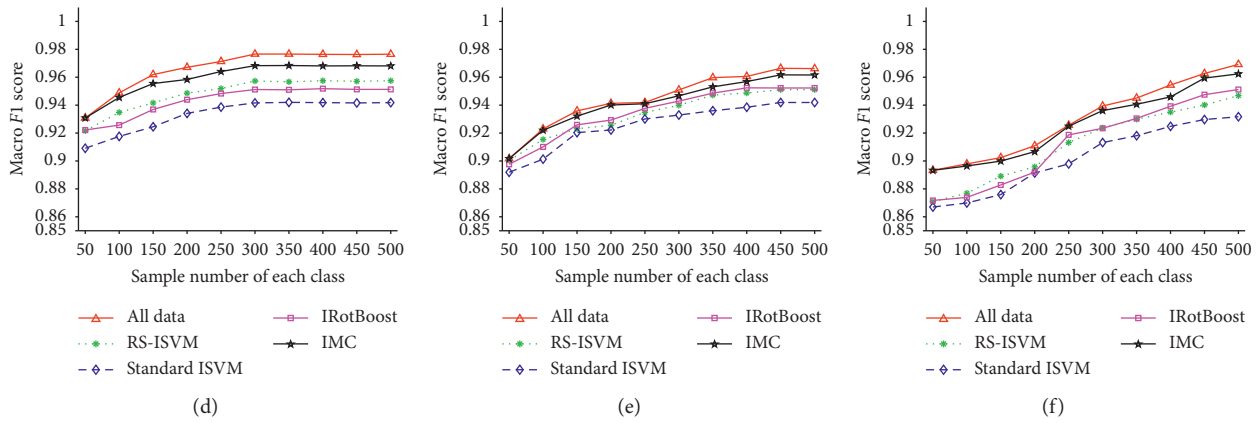

Figure 4: Macro $F1$ score comparison of five different incremental learning methods for the new data from old classes. (a) S1. (b) S2. (c) S3. (d) S4. (e) S5. (f) S6.

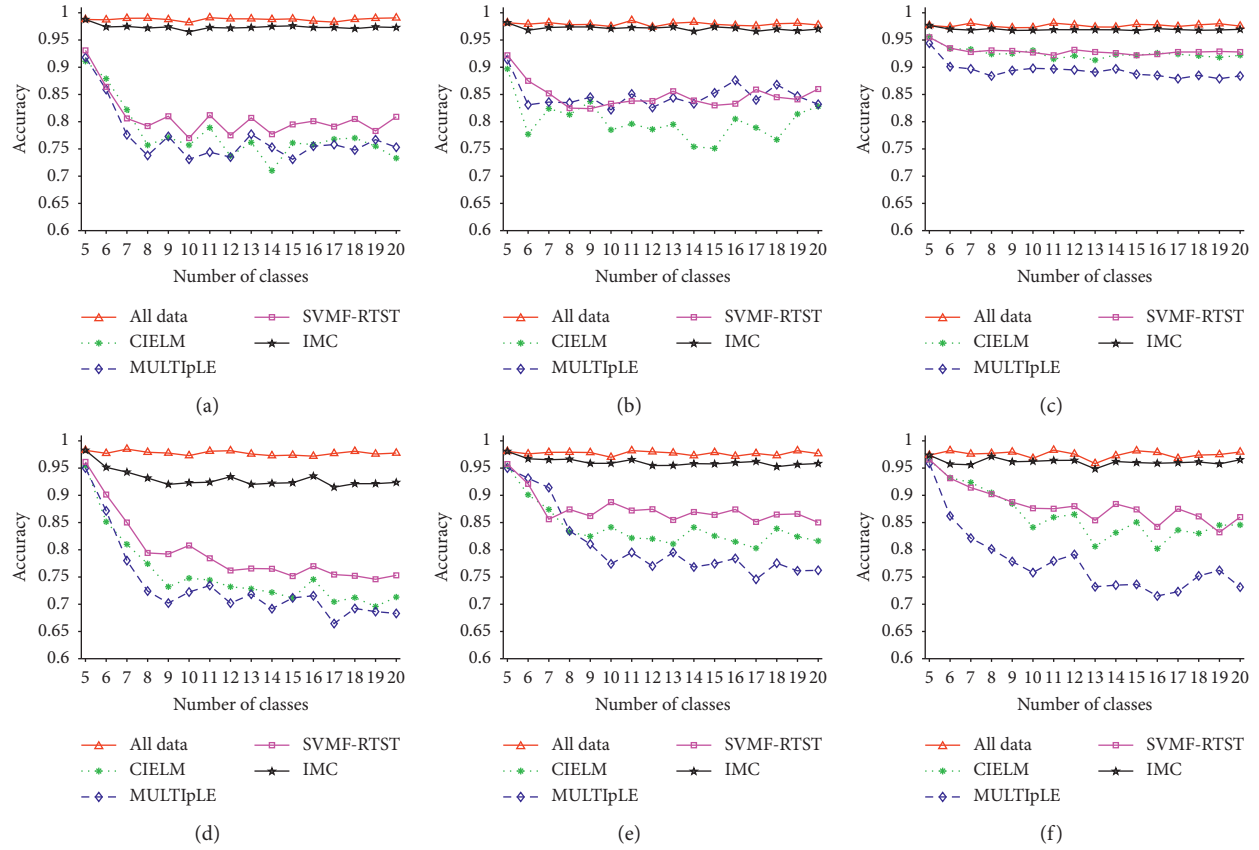

Figure 5: Accuracy comparison of five different incremental learning methods for the new data from new classes. (a) S1. (b) S2. (c) S3. (d) S4. (e) S5. (f) S6.

new data, it needs not only adding new nodes but also adjusting the association between class nodes.

5.6. Comparison between Open-Source Malware Datasets. For evaluating the performance of IMC in different open-source malware datasets, we used two different open-source malware datasets and 6 different data subsets to train IMC. The experimental results are shown in Figure 7. The change from node n to node $n+1$ can be regarded as one incremental learning process. Node n is the old model, and node $n+1$ is the new model. Figure 7(a) shows the process of data incremental learning, and Figure 7(b) shows the process of class incremental learning.

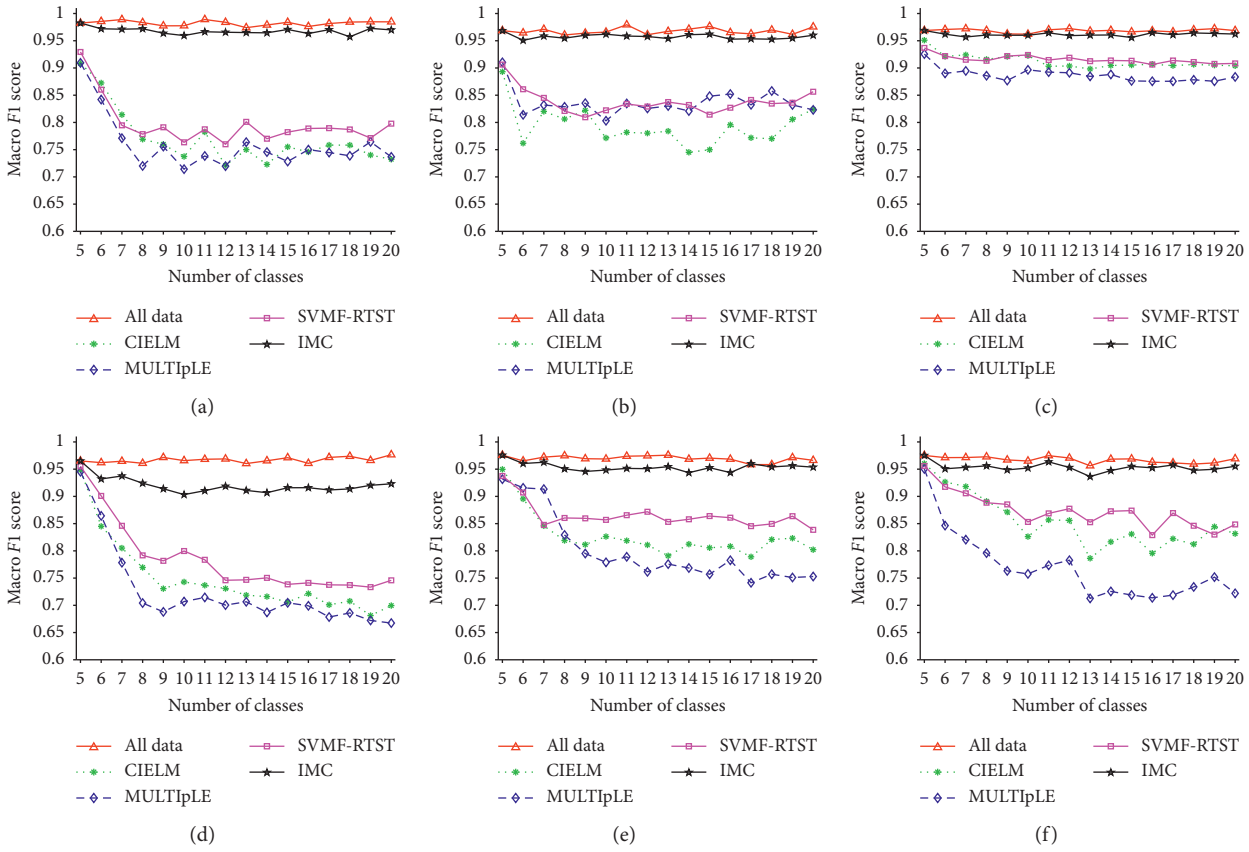

Figure 6: Macro $F1$ score comparison of five different incremental learning methods for the new data from new classes. (a) S1. (b) S2. (c) S3. (d) S4. (e) S5. (f) S6.

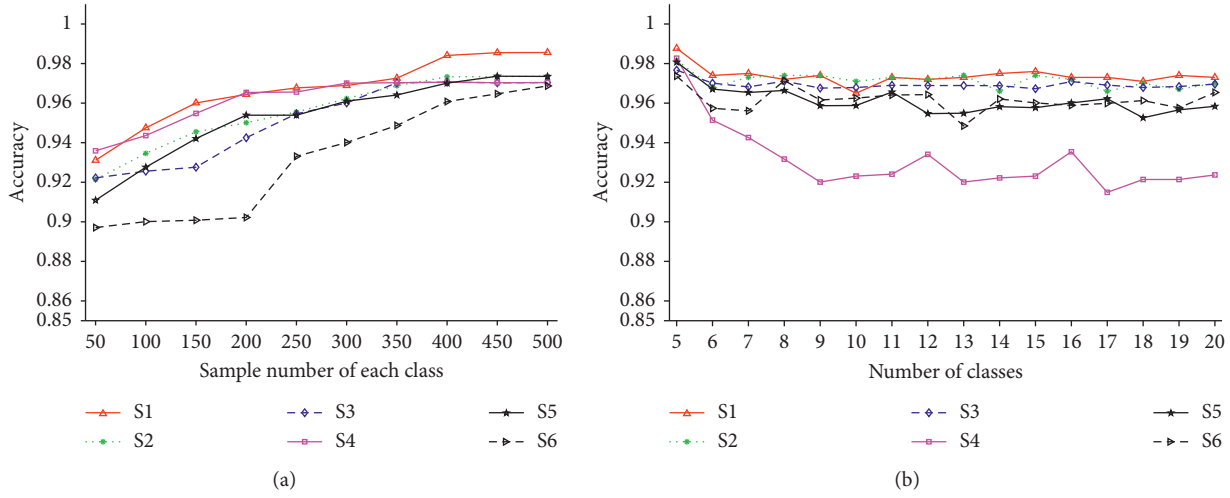

Figure 7: Accuracy comparison of IMC in different open-source malware datasets.

As shown in Figure 7, the change trend of incremental learning accuracy of IMC in different datasets is different, but the overall trend is the same. In other words, in the process of data incremental learning, with the increase of the sample number of each class, the accuracy of IMC gradually increases and tends to be stable because IMC has learned enough effective features; in the process of class incremental learning, with the increase of the number of classes, the

accuracy of IMC gradually decreases and tends to be stable. Even if there are a few fluctuations in the process of becoming stable, the fluctuation range of accuracy is within 2%. As can be seen from Figure 7, IMC has excellent incremental learning ability in different datasets.

6. Conclusion and Future Work

In this paper, we proposed a novel incremental learning method for malware classification, namely, IMC, which is based on LSSVM. IMC is capable of finding a trade-off between stability and plasticity to guarantee that it could learn new class knowledge without forgetting previously acquired old class knowledge. Moreover, IMC can not only learn the new class knowledge by adding new classification planes but also improve the old class knowledge by adjusting old classification planes. The former can learn the new data from new classes, while the latter can learn the new data from old classes. The experimental results demonstrate the advantages and effectiveness of IMC.

As the future work, the proposed method could be applied to large malware datasets; especially when the number of samples from a certain class is less than 1000, the classification performance of the SVM model is poor. So how to develop an incremental learning based on a deep neural network is a valuable study. In addition, sample imbalance and chunk learning should also be considered in incremental learning.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Key Research and Development Plan of China (grant no. 2016YFB0801004).

References

[1] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A survey on malware detection using data mining techniques," *ACM Computing Surveys*, vol. 50, no. 3, pp. 1–40, 2017.

[2] M. Karasuyama and I. Takeuchi, "Multiple incremental decremental learning of support vector machines," *IEEE Transactions on Neural Networks*, vol. 21, no. 7, pp. 1048–1059, 2010.

[3] P. Laskov, C. Gehl, S. Krüger et al., "Incremental support vector learning: analysis, implementation and applications," *Journal of Machine Learning Research*, vol. 7, no. 3, p. 2006, 2006.

[4] C. Tang, W. Li, P. Wang, and L. Wang, "Online human action recognition based on incremental learning of weighted covariance descriptors," *Information Sciences*, vol. 467, pp. 219–237, 2018.

[5] M. Ristin, M. Guillaumin, J. Gall, and L. Van Gool, "Incremental learning of random forests for large-scale image classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 3, pp. 490–503, 2016.

[6] M. Alazab, "Profiling and classifying the behavior of malicious codes," *Journal of Systems & Software*, vol. 100, pp. 91–102, 2014.

[7] D. Xue, J. Li, T. Lv, W. Wu, and J. Wang, "Malware classification using probability scoring and machine learning," *IEEE Access*, vol. 7, pp. 91641–91656, 2019.

[8] G. Liang, J. Pang, Z. Shan, R. Yang, and Y. Chen, "Automatic benchmark generation framework for malware detection," *Security and Communication Networks*, vol. 2018, Article ID 4947695, 8 pages, 2018.

[9] D. Xue, J. Li, W. Wu et al., "Homology analysis of malware based on ensemble learning and multifeatures," *PLoS One*, vol. 14, no. 8, Article ID e0211373, 2019.

[10] J. Bai and J. Wang, "Improving malware detection using multi-view ensemble learning," *Security and Communication Networks*, vol. 9, no. 17, pp. 4227–4241, 2016.

[11] D. Yuxin and Z. Siyi, "Malware detection based on deep learning algorithm," *Neural Computing and Applications*, vol. 31, no. 2, pp. 461–472, 2017.

[12] I. Santos, F. Brezo, X. Ugarte-Pedrero et al., "Opcode sequences as representation of executables for data-mining-based unknown malware detection," *Information Sciences*, vol. 231, pp. 203–216, 2013.

[13] J. Raphel and P. Vinod, "Heterogeneous opcode space for metamorphic malware detection," *Arabian Journal for Science & Engineering*, vol. 42, no. 2, pp. 1–22, 2016.

[14] P. O'Kane, S. Sezer, K. Mclaughlin, and E. G. Im, "SVM training phase reduction using dataset feature filtering for malware detection," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 3, pp. 500–509, 2013.

[15] Y. Fan, Y. Ye, and L. Chen, "Malicious sequential pattern mining for automatic malware detection," *Expert Systems with Applications*, vol. 52, pp. 16–25, 2016.

[16] I. Santos, F. Brezo, B. Sanz, C. Laorden, and P. G. Bringas, "Using opcode sequences in single-class learning to detect unknown malware," *IET Information Security*, vol. 5, no. 4, pp. 220–227, 2011.

[17] R. G. Brereton and G. R. Lloyd, "Support vector machines for classification and regression," *The Analyst*, vol. 135, no. 2, pp. 230–267, 2010.

[18] N. Kawaguchi and K. Omote, "Malware function estimation using API in initial behavior," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E100.A, no. 1, pp. 167–175, 2017.

[19] A. Pektas and T. Acarman, "Malware classification based on API calls and behavior analysis," *IET Information Security*, vol. 12, no. 2, pp. 107–117, 2018.

[20] Z. Salehi, M. A. Sami, and M. Ghiasi, "MAAR: robust features to detect malicious activity based on API calls, their arguments and return values," *Engineering Applications of Artificial Intelligence*, vol. 59, pp. 93–102, 2017.

[21] N. Nissim, R. Moskovitch, L. Rokach, and Y. Elovici, "Detecting unknown computer worm activity via support vector machines and active learning," *Pattern Analysis and Applications*, vol. 15, no. 4, pp. 459–475, 2012.

[22] Q. K. Ali Mirza, I. Awan, and M. Younas, "CloudIntell: an intelligent malware detection system," *Future Generation Computer Systems*, vol. 86, pp. 1042–1053, 2018.

[23] S. Teng, N. Wu, H. Zhu, L. Teng, and W. Zhang, "SVM-DT-Based adaptive and collaborative intrusion detection," *IEEE/*

CAA Journal of Automatica Sinica, vol. 5, no. 1, pp. 108–118, 2018.

[24] H. Du, S. Teng, M. Yang, and Q. Zhu, "Intrusion Detection System Based on Improved Svm Incremental Learning," in *Proceedings of the 2009 International Conference on Artificial Intelligence and Computational Intelligence*, pp. 1–6, Shanghai, China, 2009.

[25] G. Elham and B. Hamid, "Incremental RotBoost algorithm: an application for spam filtering," *Intelligent Data Analysis*, vol. 19, no. 2, pp. 449–468, 2015.

[26] H. O. Myint and P. Meesad, "Incremental Learning Algorithm Based on Support Vector Machine with Mahalanobis Distance (Isvmm) for Intrusion prevention," in *Proceedings of the 2009 6th International Conference on Electrical Engineering/ Electronics, Computer, Telecommunications and Information Technology*, pp. 1–4, Pattaya, Thailand, May 2009.

[27] Y. Yi, J. Wu, and W. Xu, "Incremental SVM based on reserved set for network intrusion detection," *Expert Systems with Applications*, vol. 38, no. 6, pp. 7698–7707, 2011.

[28] F. Noorbehbahani, A. Fanian, R. Mousavi, and H. Hasannejad, "An incremental intrusion detection system using a new semi-supervised stream classification method," *International Journal of Communication Systems*, vol. 30, no. 4, p. e3002, 2015.

[29] M.-H. Chen, P.-C. Chang, and J.-L. Wu, "A population-based incremental learning approach with artificial immune system for network intrusion detection," *Engineering Applications of Artificial Intelligence*, vol. 51, pp. 171–181, 2016.

[30] N. Nissim, R. Moskovitch, L. Rokach, and Y. Elovici, "Novel active learning methods for enhanced pc malware detection in windows os," *Expert Systems with Applications*, vol. 41, no. 13, pp. 5843–5857, 2014.

[31] W. Zhuang, L. Xiao, J. Cui, and W. Zhuang, "Support vector machine based on incremental learning for malware detection," in *Proceedings of the 2015 International Conference on Computer Science and Intelligent Communication*, pp. 204–207, Atlantis Press, Zhengzhou, China, 2015.

[32] P. F. Brown, P. V. Desouza, R. L. Mercer et al., "Class-based *n*-gram models of natural language," *Computational Linguistics*, vol. 18, no. 4, pp. 467–479, 1992.

[33] C. Eagle, *The IDA Pro Book: The Unofficial Guide to the World's Most Popular Disassembler*, No Starch Press, San Francisco, CA, USA, 2011.

[34] G. Salton, *The SMART Retrieval System—Experiments in Automatic Document Processing*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.

[35] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999.

[36] Z. Zhao, Z. Chen, Y. Chen, S. Wang, and H. Wang, "A class incremental extreme learning machine for activity recognition," *Cognitive Computation*, vol. 6, no. 3, pp. 423–431, 2014.

[37] I. Kuzborskij, F. Orabona, and B. Caputo, "From N to $N + 1$: multiclass transfer incremental learning," in *Proceedings of the Computer Vision and Pattern Recognition (CVPR)*, IEEE Portland, OR, USA, June 2013.