# A Semi-supervised Learning Approach for High Dimensional Android Malware Classification

Qiao Shang[1], Ni Li[2,3(✉)], Qi Qi[1(✉)], and Xiao-Wei Lin[1]

[1] School of Computer Science and Technology, Hainan University, Haikou 570228, China
{qqi,lxw0821}@hainanu.edu.cn

[2] School of Mathematics and Statistics, Hainan Normal University, Haikou 571158, China
lini@hainnu.edu.cn

[3] Key Laboratory of Data Science and Intelligence Education of Ministry of Education, Hainan Normal University, Haikou 571158, China

**Abstract.** In this paper, we proposed a semi-supervised learning approach to deal with the task of high dimensional Android malware classification. Our approach includes a random projection method for reducing feature dimensionality which would be more efficient than usual feature selection methods in existing work for the task. We also introduced a new method of SGD-based SVM with adapted sampling, which was based on the insight from the confidence and nearest neighbor clustering analysis of input data. The approach was tested on a real-world competition dataset, and effectiveness of the new method was verified by experimental results. By using the new method, we can even obtain a better classification accuracy than the best score produced in the competition.

**Keywords:** SVM · SGD · Adapted sampling · Classification · Dimensionality reduction · Android malware

## 1 Introduction

The concerns over malware threats on mobile devices have been raised since smart phones become proliferated over last several years. Traditionally signature-based anti-malware software can't predict new threats from malicious applications. Methods based on machine learning have drawn more attention in recent years.

In machine learning-based methods, it usually needs input data to train a classification model, which will predict a mobile software's label, i.e. malicious or not. The input data can be formed by a vector of Application Interfaces (APIs) called within an Application Package (APK) file. Android platform provides tens of thousands of APIs for developers. The number of key APIs ranges from a few hundred to several thousands [2]. A subset of APIs will be called during


J. Cheng et al. (Eds.): CSS 2020, LNCS 12653, pp. 20–31, 2021.
https://doi.org/10.1007/978-3-030-73671-2_3

the runtime of an application. And those API calls within an APK file can be obtained through reverse engineering.

The learning task here is to classify an android APK file's label based on such a feature vector that indicates which APIs have been used within the application. Recent works in the related field tend to use neural networks or deep learning-based models as in [7,8], these types of models usually have a huge number of parameters to train for, and are generally inefficient especially in a high dimensional input space. A simple neural network model was also presented in [17] with a constrained number of input features. Authors in [12] selected 36 features including features of permission, manifest analysis, and domains, and fed its into a list of classifiers implemented in the WEKA software for predicting android malware. The highest accuracy was 93.63% for the binary classification, and it was achieved by Random Forest (RF). As for feature selection, it is generally hard to select an optimal subset of features, whereas approximate methods exist [11]. Another related work as in [3] applied linear Support Vector Machine (SVM) on multi-modal features including information of permissions, categories, description and the API usage features. Their results showed that API calls carried the most essential behavioral information for android malware detection. A weakness of their method is that the linear SVM model was generated from solving a quadratic optimization problem by a Newton-based method, which will make the training process less efficient when there is a large number of training data. Yuki Maruno et al. [9] also proposed a RF classifier with only API-based features for the task of android malware detection. However, RF-based methods usually suffer from high dimensional data, and are often sensitive to results of feature selection. In their method, features were filtered out by applying a simple suffix-aligned rule based on API names, and their predictive result is hard to reproduce.

We propose a semi-supervised learning approach for android malware classification. The classifier is trained by SVM-based supervised learning in accordance with suggestions from the statistical analysis of input data. Briefly, our approach is mainly comprised of the following components:

- SVM model optimized by Stochastic Gradient Descent (SGD).
- Dimensionality reduction on feature vectors by random projection.
- Confidence analysis of training examples, and nearest neighbor clustering analysis of testing examples without disclosure of its labels .
- Improved SGD-based SVM model with adapted sampling distribution of training examples.

We employed the approach on a real world dataset from the 8th International Cybersecurity Data Mining Competition [1]. Experimental results show that the SGD-based SVM with adapted sampling performs better than the one with a uniform sampling distribution. And it can even yield a better predictive accuracy than the competition winner's best result.

The following Sect. 2 presents the proposed approach, and experimental setting and results will be described in Sect. 3. We will conclude in Sect. 4.

## 2 Proposed Approach

The semi-supervised learning approach includes a SGD-based SVM classifying model, a procedure of feature dimensionality reduction, and statistical analysis of training and testing data. The purpose is to make the model training more efficient and effective. The basic idea is to maintain geometric characteristic of transformed feature space when reducing the number of features, and to figure out what labeled examples are more important so that SVM during training will lean on those examples more frequently than others.

First, we introduce the method of feature dimensionality reduction. Second, we go into details about methods of statistical analysis of training and testing data, including ones for confidence and nearest neighbor clustering analysis. Then we will present the algorithm of SGD-based SVM with adapted sampling.

### 2.1 Random Projection

The main method we employed for feature dimensionality reduction is Random Projection (RP) [15]. It is a simple geometric technique for reducing the dimensionality of a set of points in Euclidean space while preserving pairwise distances approximately. The method is especially more suitable for a big data scenario than Singular Value Decomposition (SVD) [6], another widely used method for reducing feature dimensions, because of its efficiency. In our experiments, SVD-based method was also used for comparison with the RP method.

The computation of random projection is indicated by the following formula:

$$S' = S \cdot W^T$$

Where $S$ denotes a training or testing data matrix with row and column dimension as $(m, d)$. The random matrix $W$ consists of $r$ rows and $d$ columns, where $r$ indicates a lower dimensional number. Each element of $W$ is randomly generated from a Gaussian distribution with $\mu = 0$, and $\sigma = 1.0$. $W$ is also divided element-wisely by $\sqrt{d}$ to approximate its rows as orthogonal basis in the $r$ dimensional space. The resulted matrix $S'$ is the transformed lower dimensional data. This random projection method preserves all relative pairwise distances between the input feature vectors with high probability [4].

### 2.2 Confidence Analysis and Nearest Neighbor Clustering

SVM was designed to find out a discriminating hyperplane in high dimensional feature space to separate two classes with low sample complexity, because that the hyperplane was determined only by the example points, also called support vectors, around the discriminated boundary. In many real world applications, the number of support vectors is expected to be much smaller than the total number of training examples, and only those support vectors are relevant for generating the solution model. Previous works [14,16] usually focused on selecting a subset of training examples in order to speed up the SVM training.

The confidence measure was originally introduced in [16], used for evaluating how likely a training example point could be a support vector. Based on these quantities, a training sample set can be reduced by picking those with high confident values. A reduced sample set can make traditional SVM training more efficient.

Intuitively, imaging that drawing a sphere around a training example as large as possible until it covers a data point from a differently labeled class. Then, the more data points (of the same labels with the centered example point) are contained in the sphere, the less likely the centered training example will be located near the discriminating boundary of a hyperplane.

Given a labeled training sample set $X$, for each example $x$, its confidence measure can be deduced by the following steps:

1. Compute a pair-wise Euclidean distance vector containing distances between $x$ and any other data points within $X$.
2. Sort this distance vector in increasing order, and count from the beginning that the number of elements whose labels are the same as $x$'s until meeting a first different label of data example. Denotes this count as $N(x)$.
3. The confidence measure of $x$ is then in inverse proportion to $N(x)$.

Besides the confidence analysis for training examples, we also want to link this measurement with the testing examples. It's easier for SVM to successfully classify data points located far from the discriminating boundary than those adjacent to.

We also employ the K-Nearest-Neighbor (KNN) method [10] to find the adjacent training examples around every testing data point. Statistical analysis of these neighboring training points can gain us insight for the proximity of testing points to the decision boundary. For example, given the experimental data set in Sect. 3, it showed that many of the adjacent training examples around testing points have small $N(x)$ values. This result inspires us to optimize the training effect by increasing the sampling frequency of training examples with high confidence measurement. Notice that this process is done without the disclosure of labels of testing examples.

### 2.3 SGD-Based SVM with Adapted Sampling

#### 2.3.1 SGD-Based SVM

SVM [13] is an algorithm for learning the hypothesis of halfspaces with preference for large margin of data points. Assuming that a training sample set consists of $m$ examples of $(\mathrm{x_i}, y_i)$, where $\mathrm{x_i}$ is a feature vector, and $y_i$ is a corresponding label, then the model parameter w can be obtained by minimizing a regularized empirical loss function (based on the Hinge loss) as followed:

$$\min_{\mathrm{w}} \left( \frac{\lambda}{2} \|\mathrm{w}\|^2 + \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i \langle \mathrm{w}, \mathrm{x_i} \rangle\} \right) \tag{1}$$

where $\lambda$ is a regularization parameter.

The optimization problem can be efficiently solved by the method of SGD [5], even there is a large number of training examples. Basically, it randomly picks up an example to calculate an approximate gradient or a sub-gradient if the gradient did not exist, and then using the gradient to iteratively update the model parameter. The update rule of w can be further rewritten as:

$$\mathrm{w}^{(t+1)} = -\frac{1}{\lambda t}\sum_{j=1}^{t}\mathrm{v_j}\,, \tag{2}$$

where $\mathrm{v_j}$ is a sub-gradient of the loss function at $\mathrm{w}^{(\mathrm{j})}$ by the chosen random example at iteration $j$.

### 2.3.2 Adapted Sampling

Due to the nature of the SGD-based method, its descending route on the geometric surface of the optimization problem is strongly affected by which training example it will select at each of time steps, or even by the order of examples being picked out. These factors are directly influenced by the sampling distribution over training examples. SGD by default would uniformly picked out an example. If the distribution is altered, then it will likely cause the model parameter ending up at a different spot on the geometric surface, which would potentially make an impact on the method's ability of generalization.

The adaptation of sampling distribution is mainly based on results of the confidence analysis of training examples as presented in Sect. 2.2. The smaller value of $N(x)$ an example has, the more confidently it is close to be a support vector. Therefore, for a SVM-based method, high confident examples should be selected more often than others. We also consider the frequency of $N(x)$-indexed examples, and give chances to low confident examples.

Finally, the method of SGD-based SVM with adapted sampling is presented in Algorithm 1.

The algorithm needs a parameter $T$ denoting the total number of iterations, and $\theta$ denoting $-\sum_{j=1}^{t}\mathrm{v_j}$ as in Eq. 2. A major difference here is the adapted sampling distribution for selecting out training examples contrasting with uniformly choosing.

## 3 Experiment

This section first describes the process of data preparation for experiment. It then shows results of data analysis including confidence analysis of training examples, and nearest neighbor clustering analysis of testing and training examples. These results form the inspiration of developing a new method of SGD-based SVM with adapted sampling. Last, it will present experimental results of predictive accuracies given by the new method.

**Algorithm 1.** SGD-Based SVM with Adapted Sampling

**Require:** $T$
initialize: $\theta^{(1)} = 0$
**for** $t = 1 \rightarrow T$ **do**
  $\mathrm{w}^{(t)} = \frac{1}{\lambda t}\theta^{(t)}$
  Select an example $(x_i, y_i)$ randomly from the adapted sampling distribution
  **if** $y_i \langle \mathrm{w}^{(t)}, x_i \rangle < 1$ **then**
    $\theta^{(t+1)} = \theta^{(t)} + y_i x_i$
  **else**
    $\theta^{(t+1)} = \theta^{(t)}$
  **end if**
**end for**
$\overline{\mathrm{w}} = \frac{1}{T} \sum_{t=1}^{T} \mathrm{w}^{(t)}$
**return** $\overline{\mathrm{w}}$

**Table 1.** Dimensions of prepared data

| Data | Dimensions |
|---|---|
| Training | (30897,37107) |
| Training-label | (30897,1) |
| Testing | (30833,37107) |
| Testing-label | (30833,1) |

### 3.1 Data Preparation

The dataset for experiment was originally from a data mining competition (CDMC 2017) [1]. Training data file includes 30897 rows of API identification numbers (IDs). Each row represents an Android application (aka an APK) comprised of a different number of APIs. There are totally 37107 of unique APIs with its name provided. The label file for training data contains the same number of rows, and each row has either a number 1 for labelling malicious class or $-1$ for benign class. Testing data file represents 30833 different applications with the same format. Since the competition had closed, the true labeling of the testing data was also released.

In the setting of our experiment, the original training and testing data were extended to full matrices. Each row of application becomes a full scale of API indicators initialized by zeros. Then, the columns corresponding APIs used by the application were filled up by 1s. Dimensions, numbers of rows by columns, of the prepared datasets are shown in Table 1.

The processed datasets of training and testing were essentially high dimensional and sparse matrices, which posed challenges for training SVM models. The motivation was to turn them into dense matrices with lower number of feature

dimensions. We employed the RP method as introduced in Sect. 2.1 to transform the datasets. The best random matrix was selected out by cross-validation method, and then it was multiplied by both training and testing datasets separately. We set the reduced feature dimension to number 10000 and 1000, respectively. Then dimensionality reduced datasets were used in the following experiment.

### 3.2 Confidence and Nearest Neighbor Clustering Analysis

We computed confidence measurement based on $N(x)$ values introduced in Sect. 2.2 for training data. Figure 1 shows histogram of N(x) values for the 10000-feature set, and Fig. 2 shows the counterpart for the 1000-feature set. The density curve lines were formed by the method of smooth spline regression. These two figures show almost the same pattern that a majority of training examples has relatively small $N(x)$ values (aka highly confident data points), and as the $N(x)$ values get increased, the amount of corresponding examples is rapidly decreased.

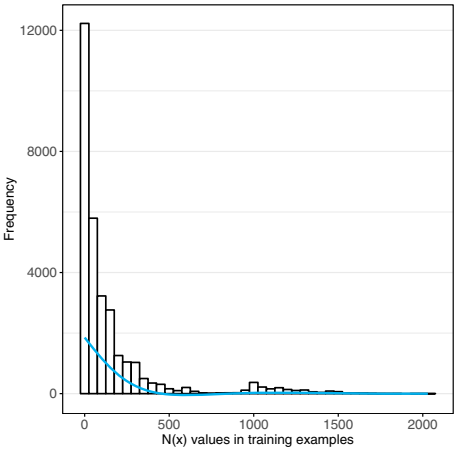


**Fig. 1.** Histogram of $N(x)$ values with the RP:10000 dataset

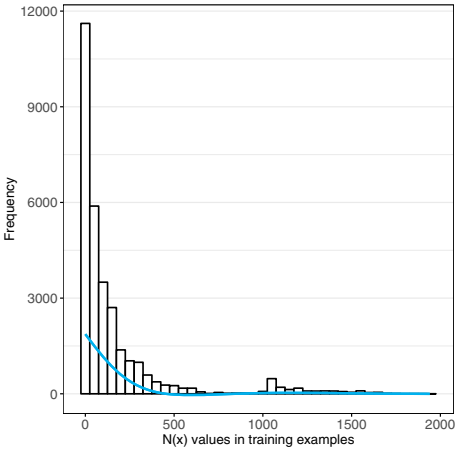


**Fig. 2.** Histogram of $N(x)$ values with the RP:1000 dataset

Next, we used testing examples as centroids to cluster training examples around them on the 10000-feature set. It gathered 10 nearest neighboring training examples for each of testing examples, and computed average $N(x)$ values and Euclidean distances between centroids and neighbored points within every neighborhood. Histogram Fig. 3 shows that many testing examples have a neighborhood of relatively small $N(x)$ values of training examples, and those neighbors are also very close in Euclidean distance to their testing centroids as shown in Fig. 4. Imaging that a large number of testing points are located very closely to the potentially discriminating boundary supported by high confident training points. This phenomenon would wield challenges for SVM-based methods.

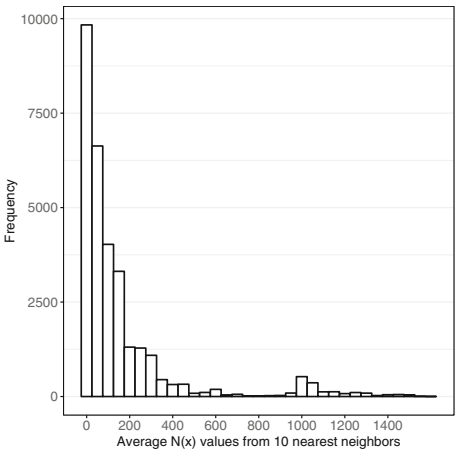


**Fig. 3.** Histogram of average $N(x)$ values

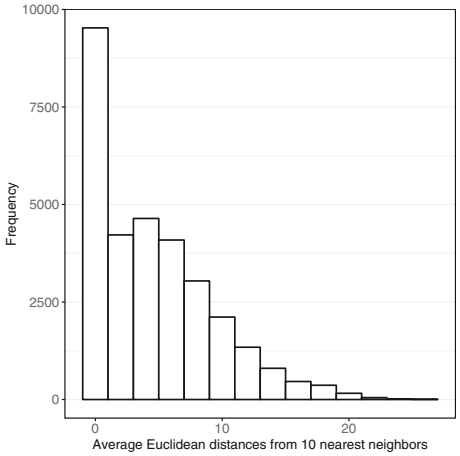


**Fig. 4.** Histogram of average Euclidean distances

Based on observations from previous figures, the statistical characteristic of the training and testing datasets suggests that we should put more weights on low $N(x)$ values of training examples than those of high $N(x)$ values generally during SGD processing, in order to deal with the hard phenomenon.

### 3.3 Prediction of SGD-Based SVM with Adapted Sampling

The statistical analysis of confidence and nearest neighbor clustering of the datasets suggested to alter sampling distribution during SGD processing in order to select high confident training examples more frequently than low confident ones. The adapted sampling distribution was directly converted from a smooth spline regression of frequency data of $N(x)$ values on training examples. The regression curve lines were drawn on Figs. 1 and 2.

Before applying the Algorithm 1, training examples with $N(x)$ values larger than 400 were filtered out. To evaluate the effectiveness of the SGD-based SVM with adapted sampling, we compared it to SGD-based SVM with uniform sampling. Both of models were trained 200 trials, with random shuffle of training examples before each trial. Within every trial, a model was trained with 200 epochs of training set. The $\lambda$ parameter was fixed to a tuned value during these experiments. Predictive performances were measured by the accuracy rate that is the percentage of testing examples being correctly classified.

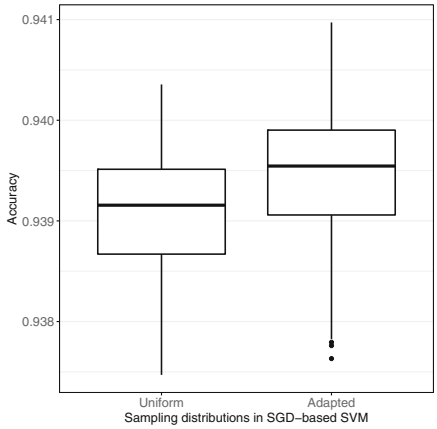


**Fig. 5.** Box plots of accuracy scores with the RP:10000 dataset

We first show the results of the approach applied on the 10000-feature dataset generated by RP (RP:10000). Figure 5 compares box plots of accuracy scores given by SGD-based SVM with uniform and adapted sampling distributions, and each plot was formed by 200 random trials of results. Also, Fig. 6 shows results with the same setting of experiments as in Fig. 5 but on a different 10000-feature dataset generated by a truncated SVD method (SVD:10000). The purpose of comparing with SVD was to validate the RP method of dimensionality reduction. The detailed statistics of the two sets of box plots are displayed in Table 2.

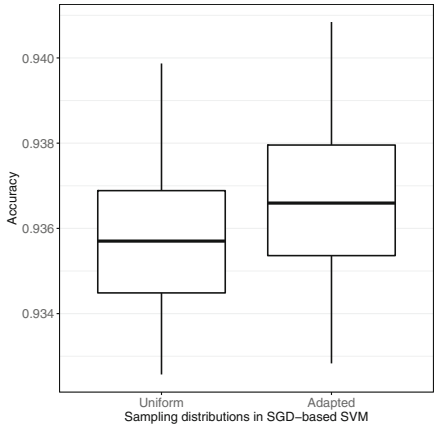


**Fig. 6.** Box plots of accuracy scores with the SVD:10000 dataset

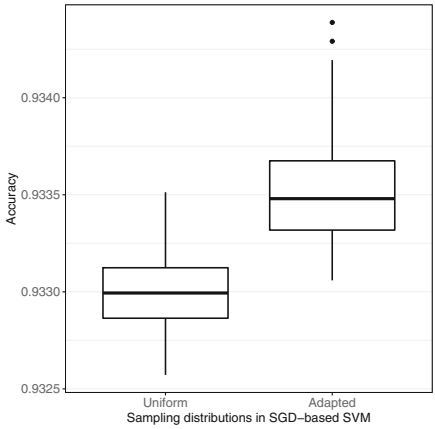


**Fig. 7.** Box plots of accuracy scores with the RP:1000 dataset

**Table 2.** Statistics with the box plots

| | RP:10000 | | SVD:10000 | |
|---|---|---|---|---|
| | Uniform | Adapted | Uniform | Adapted |
| Median | 0.93916 | 0.93955 | 0.93570 | 0.93659 |
| Sd | 0.00064 | 0.00062 | 0.00162 | 0.00162 |
| Max | 0.94036 | 0.94097 | 0.93987 | 0.94084 |

Adapted sampling in SGD-based SVM boosts predictive performance over that with uniform sampling as shown in both of Figs. 5 and 6. With the RP:10000 dataset, It increased median accuracy score by 0.00039, which should not be undervalued because that the increased amount occupies more than 60% of the relatively small value span of the standard deviation (Sd). It also pushed the maximum score from 0.94036 to 0.94097 which already exceeded the championing result 0.9405 of the competition. The same kind of observations of comparing adapted sampling with uniform sampling can be found out on Fig. 6 as well when experimenting with the SVD:10000 dataset. Additionally, these results also show that the RP is a competitive method of dimensionality reduction in practice compared with the SVD-based method. As shown in Table 2, performances on the RP:10000 dataset generally achieved higher scores and lower standard deviation than its counterparts on the SVD:10000 dataset.

We also experimented with a lower number of feature dimension. Specifically, the same setting of experiment was repeated on a 1000-feature dataset transformed by the RP method, and its results are shown on Fig. 7. The advantage of adapted sampling over uniform sampling in SGD-based SVM is more obvious in the picture than previous ones such that the box framed by upper and lower

quartiles of accuracies given by the adapted sampling's is completely located above the box given by the uniform sampling's, although accuracy scores in this setting of feature dimension are relatively smaller than those in the 10000-feature one.

## 4 Conclusion

In this paper, we presented a semi-supervised learning approach for tackling with high-dimensional Android malware classification tasks, which incorporated components such as feature dimensionality reduction, confidence analysis and nearest neighbor clustering analysis of input data. Within the approach, we also introduced a new model of SGD-based SVM with adapted sampling. Experimental results on a real-world competition's dataset show that the performance of our sampling-adapted SGD-based SVM is statistically better than the original SGD-based SVM with uniform sampling. It can even achieve a more accurate result than the champion's score in the competition.

**Acknowledgements.** This work was sponsored in part by Hainan University's Scientific Research Start-Up Fund, and the Ministry of Education of China's Scientific Research Fund for the Returned Overseas Chinese Scholars awarded to the corresponding author. This work was also supported by the National Natural Science Foundation of China under the grants No. 11861030, No.61962017, No.61865005, Natural Science Foundation of Hainan Province projects under the grant No.2019RC176, 2019RC088, Hainan Key R&D Program No. ZDYF2019115, and the National Key Research and Development Program of China under the grant No. 2018YFB1404400. The authors of this work was also supported by the State Key Laboratory of Marine Resource Utilization in the South China Sea at Hainan University, and the Key Laboratory of Big Data and Smart Services of Hainan Province.

## References

1. The 8th international Cybersecurity Data Mining Competition (cdmc2017) (2017). http://www.csmining.org/cdmc2017/
2. Aafer, Y., Du, W., Yin, H.: DroidAPIMiner: mining API-level features for robust malware detection in android. In: Zia, T., Zomaya, A., Varadharajan, V., Mao, M. (eds.) SecureComm 2013. LNICST, vol. 127, pp. 86–103. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-04283-1_6
3. Ban, T., Takahashi, T., Guo, S., Inoue, D., Nakao, K.: Integration of multi-modal features for android malware detection using linear SVM. In: 2016 11th Asia Joint Conference on Information Security (AsiaJCIS), pp. 141–146. https://doi.org/10.1109/AsiaJCIS.2016.29
4. Blum, A., Hopcroft, J.E., Kannan, R.: Foundations of Data Science. Cambridge University Press, New York (2020)
5. Bottou, L.: Large-Scale Machine Learning with Stochastic Gradient Descent, pp. 177–186. Physica-Verlag HD, Heidelberg, USA (2010). https://doi.org/10.1007/978-3-7908-2604-3_16

6. Golub, G.H., Reinsch, C.: Singular value decomposition and least squares solutions. Num. Math. **14**(5), 403–420 (1970)
7. Kang, J., Jang, S., Li, S., Jeong, Y.S., Sung, Y.: Long short-term memory-based malware classification method for information security. Comput. Electr. Eng. **77**, 366–375 (2019). https://doi.org/10.1016/j.compeleceng.2019.06.014. https://www.sciencedirect.com/science/article/pii/S0045790618328167
8. Lin, Q.G., Li, N., Qi, Q., Hu, J.B.: Classification of IoT malware based on convolutional neural network. In: 2020 International Conference on Service Science (ICSS), pp. 51–57 (2020). https://doi.org/10.1109/ICSS50103.2020.00016
9. Maruno, Y., et al.: Solution for the CDMC 2017. Technical Report, Kyoto Women's University (2017)
10. Peterson, L.E.: K-nearest neighbor. Scholarpedia **4**(2), 1883 (2009). https://doi.org/10.4249/scholarpedia.1883
11. Qi, Q., Li, N., Li, W.: Exploration of heuristic-based feature selection on classification problems. In: Chen, G., Shen, H., Chen, M. (eds.) Parallel Architecture, Algorithm and Programming, pp. 95–107. Springer Singapore, Singapore (2017)
12. Sachdeva, S., Jolivot, R., Choensawat, W.: Android malware classification based on mobile security framework. IAENG Int. J. Comput. Sci. **45**(4), 514–522 (2018)
13. Shalev-Shwartz, S., Ben-David, S.: Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, New York, USA (2014)
14. Vapnik, V.: Estimation of Dependences Based on Empirical Data. Springer-Verlag New York, New York, USA (2006)
15. Vempala, S.S.: The random projection method, DIMACS series in discrete mathematics and theoretical computer science. In: DIMACS/AMS, vol. 65 (2004). http://dimacs.rutgers.edu/Volumes/Vol65.html
16. Wang, J., Neskovic, P., Cooper, L.N.: Training data selection for support vector machines. In: Wang, L., Chen, K., Ong, Y.S. (eds.) ICNC 2005. LNCS, vol. 3610, pp. 554–564. Springer, Heidelberg (2005). https://doi.org/10.1007/11539087_71
17. Zhou, Q., Feng, F., Shen, Z., Zhou, R., Hsieh, M.Y., Li, K.C.: A novel approach for mobile malware classification and detection in android systems. Multimedia Tools Appl. **78**(3), 3529–3552 (2019). https://doi.org/10.1007/s11042-018-6498-z