Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 108C (2017) 1682–1691

Procedia Computer Science

International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland

Collaborative Support Vector Machine for Malware Detection

Kai Zhang[1,2], Chao Li[3*], Yong Wang[1,2*], Xiaobin Zhu[4*], Haiping Wang[2]

[1]*School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China*

[2]*The Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China*

[3] *National Computer Network Emergency Response Technical Team/Coordination Center of China*

[4] *Beijing Technology and Business University, Beijing, China*

lichao@cert.org.cn, brucezhucas@gmail.com

Abstract

Malware has been the primary threat to computer and network for years. Traditionally, supervised learning methods are applied to detect malware. But supervised learning models need a great number of labeled samples to train models beforehand, and it is impractical to label enough malicious code manually. Insufficient training samples yields imperfect detection models and satisfactory detection result could not be obtained as a result. In this paper, we bring out a new algorithm call ColSVM (Collaborative Support Vector Machine) based on semi-supervised learning and independent component analysis. With ColSVM, only a few labeled samples is needed while the detection result keeps in a high level. Besides, we propose a general framework with independent components analysis, with which to reduce the restricted condition of collaborative train. Experiments prove the efficiency of our model finally.

Peer-review under responsibility of the scientific committee of the International Conference on Computational Science

Keywords: malware detection; independent component analysis; semi-supervised learning

1 Introduction

Malware is defined as any type of computer software harmful to computers or networks, which has been posing a serious threat to the global security [1]. What's more, the amount of malware is increasing rapidly in recent years [2][3]. Therefore, detecting malicious code is of great significance and draws attention of experts worldwide in the field of information security.

Traditionally, researchers use supervised learning methods to fulfill the detection of malware, but the disadvantages are obvious. Firstly, it is hard to obtain an excellent model for malware detection in many cases. When supervised learning methods are applied to detect malware, labeled samples are necessary for training a detection model. However, it is impractical to label a large scale of unlabeled

* Corresponding authors.

1877-0509
Peer-review under responsibility of the scientific committee of the International Conference on Computational Science
10.1016/j.procs.2017.05.063

samples as only a handful of related experts are qualified for this work [4]. As a result, the labeled samples for model training are usually insufficient in many cases, yielding that the detection models are imperfect. Secondly, the generalization ability of malware detection models learned by supervised methods is poor. With malicious code increasing rapidly, new variants appears frequently [5][6][7]. But supervised learning is to train a constant classifier with labeled data, and the classifier is not always suitable for detecting new variants in anther dataset, which also affect the result of detection [8][9]. Therefore, we study how to fulfill malware detection in more effective ways.

Thanks to the rapid development of machine learning [10][11][12][13], new methods appear such as active learning [14][15][16] and semi-supervised [17][18][19] learning. These methods combine the advantages of supervised learning and unsupervised learning, as they not only train model with labeled samples like supervised learning but also make full use of unlabeled samples like unsupervised learning [20][21]. In this paper, we bring out a new algorithm，ColSVM, with collaborative training, a method belonging to semi-supervised learning, to detect malware.

The key contributions of ColSVM are--reduces the restricted condition of collaborative training, making it possible to design malware detection model with the same two supervised learning methods; reduces the dependence on labeled samples while the detection result keeps in a high level.

Generally, collaborative training needs to train two different classifiers, so if the feature of samples is multi-views, we could train classifiers from different views. But if the feature is single-view, different supervised learning methods should be applied to guarantee the difference of classifiers. However, when detecting malware, the performance of SVM outperforms competitors' evidently while the feature of malware is single-view. Therefore, how to fulfill malware detection with two same SVM classifiers is a key issue. In this paper, ColSVM preprocesses dataset with ICA(independent components analysis) and dived the feature into two unrelated parts, in which way two SVM classifiers could be applied later. With our model, only a small amount of labeled samples is needed to achieve the family classification of malicious code. Experiments prove the efficiency of ColSVM. Besides, we also discuss about the effect of recommended samples' number towards malware detection in the experiment.

The rest of this paper is organized as follow: In section 2, we introduce the preliminary knowledge. In section 3, we explain the method for designing ColSVM. The experiment is described in section 4, and we make the conclusion in section 5 finally.

2 Preliminary

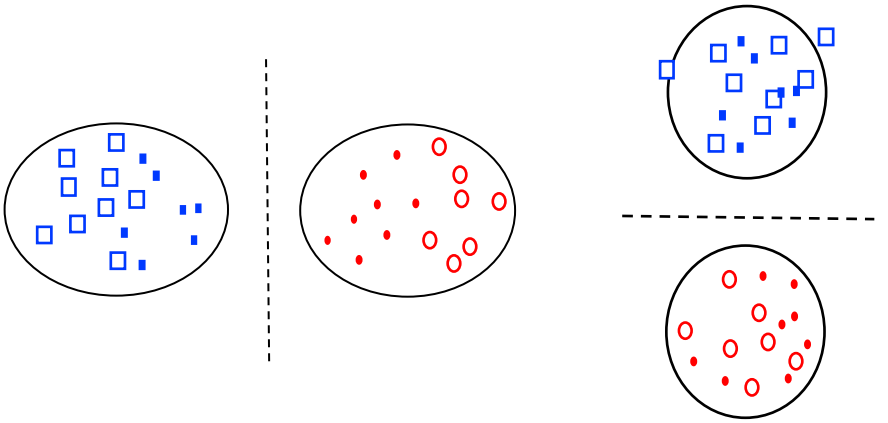
In this section, we will introduce the theories contributing to design ColSVM.

2.1 Collaborative training

Collaborative training is put forward by Blum and Mitchell in 1998 [22]. It runs in the following steps. Firstly, train two separate classifiers with two sub-feature sets respectively. Secondly, each classifier classifies the unlabeled data and extract recommended samples. Next, add the recommended samples to the train set of each other, and train classifiers for the second time. Then we would obtain two better classifiers and obtain the ultimate result with the two new classifiers finally. Collaborative training is highly effective in reducing the dependence on labeled samples. However, the theory is restricted to use in certain conditions which are list below.

- (i) Feature can be split into two sets;
- (ii) Each sub-feature set is sufficient to train a classifier;
- (iii) The two sets are conditionally independent given the class.

For malware detection in this paper, condition (i) and condition (ii) are satisfied, but the feature does not meet condition(iii) due to the mutual relationship among vectors of feature set. However, it is necessary to solve the case and the necessity is presented in figure 1.



(a) view 1 (b) view 2

Fig 1: The two views of Collaborative training

As shown in Figure 1, figure 1(a) shows the distribution of samples in view 1, while figure 1(b) shows in view 2. The red dots, blue dots, circles and squares in the figure are all unlabeled samples to be detected, and we use red dots to represent negative samples while red dots to represent positive ones. Besides, for the convenience of description, we mark the samples in high degrees of confidence in obvious way, where the circles represent negative samples whose confidence are in high degree and squares represent positive ones. If the two views are independent mutually, the distribution of samples are different in the two views. As we can see in the figure, samples in high degrees in view 1 are distributed randomly in view 2. If the two views are not independent, on the contrary, the random distribution could not be guaranteed. In order to describe the necessity clearly, take the extreme case as example—supposing the distributions of samples in two views are completely consistent, we will obtain two same classifiers. The two classifiers' recommended samples are also same in the case and the collaborative train makes no sense. Therefore, two unrelated views are of great importance for collaborative train, and we proposed to preprocess samples with ICA, in which way to meet the condition (iii) and make collaborative train feasible.

2.2 Independent Component Analysis

ICA is a method to find out the hidden factors or components from multidimensional statistical data [23]. It attempts to decompose a multivariate signal into independent non-gaussian signals. From the perspective of linear transformation and linear space, the source signals are non-gaussian and independent from each other, while the observation signal is a linear combination of source signals. The function of ICA is to estimate source signals without knowing both source signals and linear transformation. The basic idea of ICA theory is to extract signals which are as independent as possible from a set of mixed observation signals, and then characterize the other signals with the independent signals [24]. ICA can be described mathematically as follows:

The data are represented by the random vector $\boldsymbol{X} = (x_1, x_2 \dots x_m)^T$ and the components as the random vector $\boldsymbol{S} = (s_1, s_2 \dots s_m)^T$. ICA can be expressed as the liner relationship between $\boldsymbol{X}$ and $\boldsymbol{S}$-- $\boldsymbol{X} = \boldsymbol{AS} = \sum a_i s_i$. Here $A = (a_1, a_2 \dots a_m)$ is the mixing matrix in the formula. Then we could get signal $\boldsymbol{Y}$ with formula $\boldsymbol{Y} = \boldsymbol{WX} = \boldsymbol{WAS}$, where $\boldsymbol{A} = inv(\boldsymbol{W})$.

The task of ICA is to obtain a separation matrix ***W*** through ***X***, making signal ***Y*** the most optimal approximation to ***S***.

2.3 Support Vector Machine

In this paper, we design model with SVM (support vector machine) to detect malicious code. SVM is an excellent machine learning method based on supervised learning [25][26]. It includes two cases--linear separable problem and nonlinear separable problem. In the case of linear separable problem, the train set is define as $\Omega = \{(x_i, y_i)|i = 1,2 \dots N\} \subset R^m \times \{-1,1\}$, ,where $x_i \in R^m$, $y_i \in \{-1,1\}$. Supposing the set is linear separable, then we'll obtain a hyperplane $w^T x + b = 0 (x \in R^m)$, and the formula can be expressed as $y_i(w^T x + \mathrm{b}) \geq 1$,i=1,2…*N*. In the case of nonlinear separable problem, the problem is more complicated and we can't fulfill the classification just by hyper-plane, therefore we do it by hyper-surface instead. The main idea of hyper-surface is to express the training samples in a higher feature space ***H*** , where the training samples will be linear separable.

Here a nonlinear mapping $\Phi: R^m \leftarrow \boldsymbol{H}$ is needed in order to make Ω linear separable. Next we could do classification like processing linear separable problem. The only problem is we need to replace x with Φ(x) and the final function is $(x) = \mathrm{sign}\{\sum_{i-1}^{N} \lambda_i y_i \phi(x_i)^T \phi(x) + b\}$.

3 Design of malware detection model

In this section, we will formally describe our design of ColSVM. Before description, we would express the notation in the first place.

3.1 Notation

There are mainly three kinds of set, i.e. training set ***Tr***, testing set ***Te*** and recommended set ***R***. Each set is composed of two parts, i.e. feature property set and class property set. Then notations in detail are listed in table1.

Symbol	Meaning
Tr	Training set
Te	Testing set
L	Labeled Sample
U	Unlabeled Sample
R	Recommended Sample
Rp	Positive Fake Sample
Pn	Negative Fake Sample
X	Feature Property
Y	Class Property
Xr	Feature Property of Training Dataset
Yr	Class Property of Training Dataset
Xe	Feature Property of Testing Dataset
Ye	Class Property of Testing Dataset
vd	Vector Dimension

Table 1: Main symbols used in the paper

3.2 Description of algorithm ColSVM

In this part we'll express algorithm ColSVM in detail including the method to design algorithm and its execution process.

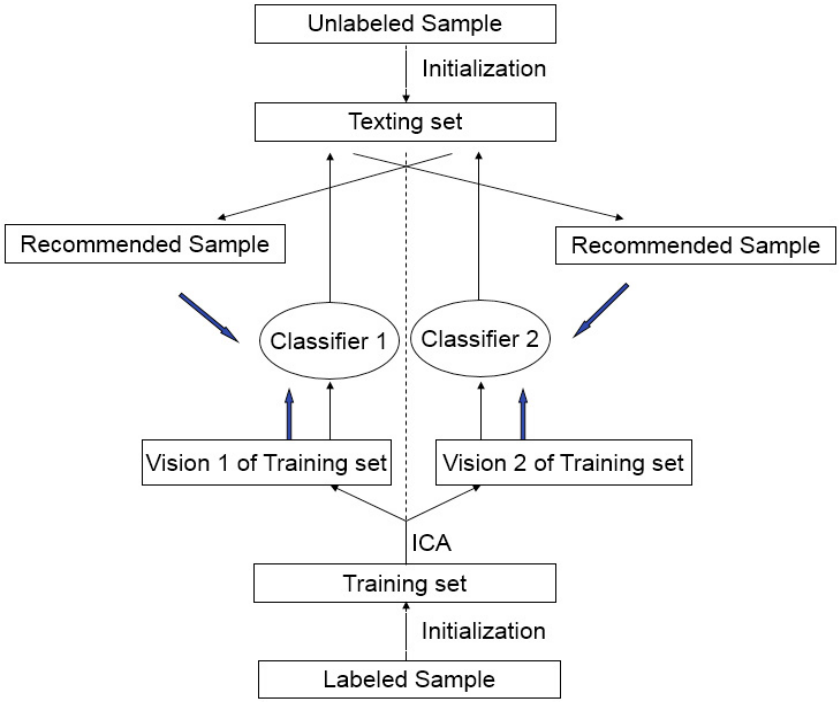


Fig 2: Execution of ColSVM

The realization steps of ColSVM are as follow:

Step1 Independent feature set partition

In order to get two mutually independent sub-feature set of malware, we firstly preprocess labeled dataset ***L*** with ICA and get dataset ***Tr***, and then split ***Tr*** into two sets ***Tr1*** and ***Tr2***. With the help of ICA, the independence of ***Tr1*** and ***Tr2*** is guaranteed. We also handle unlabeled dataset ***U*** with ICA and get testing set ***Te***, and obtain ***Te1*** and ***Te2*** correspondingly;

Step2 Train individual classifiers

We train two classifiers, i.e. classifier *C1* and classifier *C2* by training with ***Tr1*** and ***Tr2***. Although the training set is the same one, the two classifiers are absolutely unrelated as the two sub-feature sets ***Tr1*** and ***Tr2*** are totally independent. Then we test the testing set ***Te*** by using classifier *C1* and classifier *C2* and obtain two different results ***Ye1*** and ***Ye2***;

Step3 Form new training set

The next step concerns how to form new training set by recommended samples. Recommended samples are selected depending on the distance between sample and hyper-plane. Firstly, we sort the results ***Ye1*** and ***Ye2*** based on the distance, and then select the top k to form the recommended dataset ***R1*** and ***R2*** respectively. Finally, we combine ***R1*** with ***L2*** and combine ***R1*** with ***L2*** to form new training set;

Step4 train new classifiers and obtain final result

we train two classifiers with new training set and get classifier C1' and classifier C2'. Then we make a second test towards testing dataset ***Te*** with the two new classifiers and obtain the ultimate results ***Ye1'*** and ***Ye2'***, with which we Compute precision, recall rate, F-measure and accuracy rate at last .

The execution process of algorithm ColSVM is demonstrated clearly in figure 2, and the pseudo-code of algorithm ColSVM is shown as follows.

form feature set ***Tr*** by handling ***L*** with ICA;
split the ***Tr*** into two sets ***Tr1*** and ***Tr2*** ;
For *1*:*N*
Obtain classifier *C1* by training with SVM and ***Tr1***;
Obtain classifier *C2* by training with SVM and ***Tr2***;
Obtain result ***Ye1*** by testing ***Te*** with *C1*;
Obtain result ***Ye2*** by testing **Te** with *C2*;
$\boldsymbol{Temp1} \leftarrow Qsort(1, \boldsymbol{Temp1})$;
$\boldsymbol{Temp2} \leftarrow Qsort(1, \boldsymbol{Temp2})$;
$\boldsymbol{R1} \leftarrow Top(k1, \boldsymbol{Temp2})$;
$\boldsymbol{R2} \leftarrow Top(k2, \boldsymbol{Temp2})$;
$\boldsymbol{Tr1} = (\boldsymbol{L1}, \boldsymbol{R2})$;
$\boldsymbol{Tr1} = (\boldsymbol{L2}, \boldsymbol{R1})$;
END
Obtain result ***Ye1'*** by testing *Te* with *C1*;
Obtain result ***Ye2'*** by testing *Te* with *C2*;
Compute pre,recall,F1,auc with ***Ye1'*** and ***Ye2'***;

Table 2: Pseudocode of ColSVM

4 Experiment Results

In this section, we introduce the malware dataset used in experiments, and then show the performance of our model.

4.1 Dataset

In this paper, the Malware dataset for experiment contains 2415 samples belonging to 8 classes. These malware is the primary threat at present, and we extract them randomly, in which way the number of every class could represent its distribution proportion.

We analyze the executable file of malware and extract key words which are most likely identified as abnormality, and obtain feature of malicious codes by CRC64 unified coding. Specially, we choose 162 key words as the features of malware. Then we use binary variables to form 162-dimensional feature set, and each dimension corresponds to each key word. If a sample includes a certain word, the variable of this dimension is set as 1, otherwise set as 0. The information in detail is listed in table 2.

C_ID	C_NAME	C_NUM
C1	killAV	1047
C2	Trojan/Win32.Agent.dbvl[Downloader]	366
C3	Worm.Win32.Palevo.ayal	462
C4	"\u7279\u5f817"	108
C5	tp2	69
C6	Worm.Win32.Palevo.ayal29	282
C7	"\u7279\u5f8117"	33
C8	gh0st	48

Table 3: Malware Dataset

4.2 Performance evaluation

The performance of a classifier can be quantified with precision, recall, F-measure and accuracy. We use *TP, TN, FP, FN* to represent the number of true positives, true negatives, false positives and false negatives respectively, then we can obtain the four performance indexes precision *P*, recall *R*, F-measure *F*, accuracy *A* as follows.

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + TN}$$

$$F = 2\frac{P * R}{P + R}$$

$$A = \frac{TP + FN}{TP + FP + TN + FN}$$

4.3 Experiment for comparison with traditional method

In this experiment, we would compare our model with traditional method. As introduced in section 2, when detecting malware, SVM could achieve the most satisfactory result among supervised learning methods, so we compare our model with SVM. We use RBF kernel function to design the classification model and set penalty factor as 100 while kernel parameter is set as 0.01. The number of recommended samples is 8, and the number of training samples is the variable in the experiment. As F-measure is the fusion of the two indexes precision and recall, so we use F-measure and accuracy to evaluate the algorithm. The performance of SVM and ColSVM are shown in figure 3~6, and we can see ColSVM's is better than SVM's when the number of recommended samples in training set varies from 20 to 60. The average values of F-measure and accuracy of ColSVM are lifted by 21.06% and 16.71% respectively. The enhancement is especially remarkable when the training set is small, and we can draw conclusion that our model is particularly efficient when labeled samples are insufficient.

4.4 Experiment for discussion of recommended samples' number

In this experiment, we will discuss the effect of recommended samples' number towards malware detection. We also use RBF kernel function to design the classification model and the parameters are set as experiment 1. The training set is composed of 16 labeled samples, and 8 of them are positive ones in order to guarantee every type of malicious code is included. The rest 8 ones are chosen randomly. The number of recommended samples are variable and set as 0,2,4,6,8 respectively. Specially, when the number of recommended samples is 0, ColSVM is equivalent to SVM. Performance is shown further in

figure 7~10, and we depict auxiliary line (solid line) in figures for convenience of comparison, representing the case when number is 0.

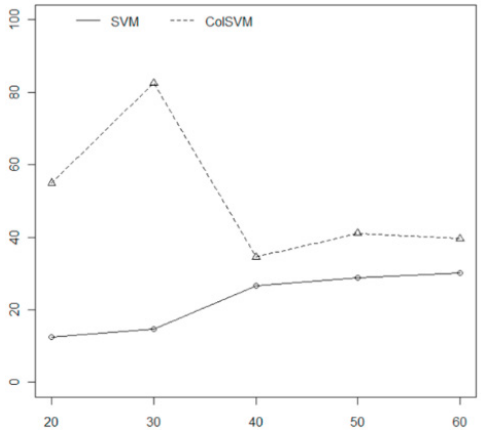


Fig 3: Comparison of recall

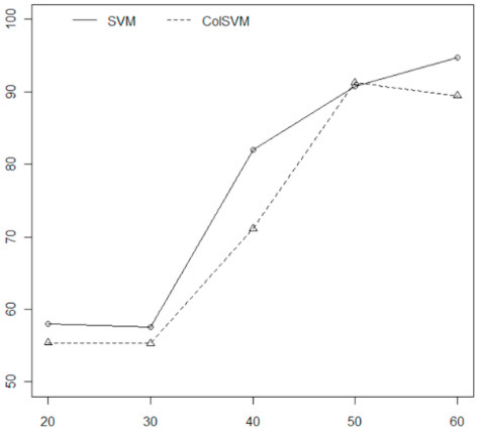


Fig 4: Comparison of precision

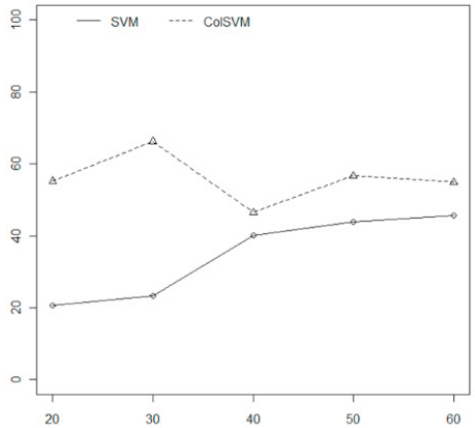


Fig 5: Comparison of F-measure

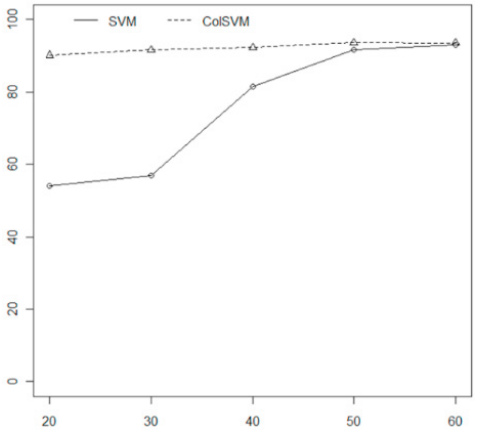


Fig 6: Comparison of accuracy

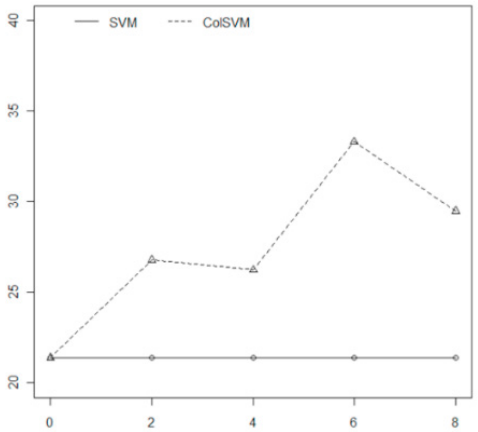


Fig 7: Comparison of recall

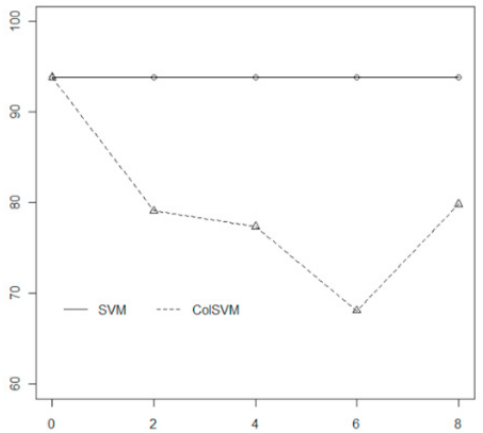


Fig 8: Comparison of precision

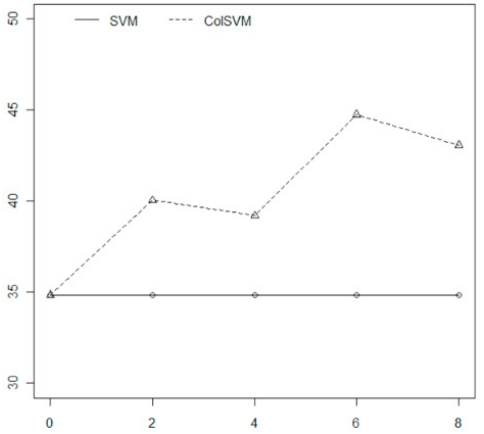


Fig 9: Comparison of F-measure

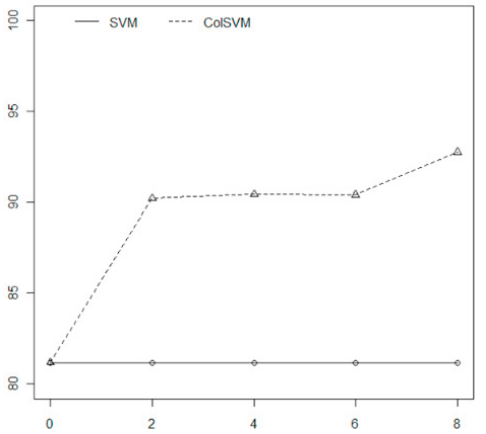


Fig 10: Comparison of accuracy

We use F-measure and accuracy to evaluate the algorithm. The X axis variable in the figures denotes the number of recommended samples. As we can see in the picture, F-measure and accuracy will rise with the increasing of recommended sample number. When the number of recommended samples varies from 2 to 8, the average of F-measure and accuracy of ColSVM will be lifted by 6.35% and 9.79% respectively. The performance of ColSVM is obvious superior to SVM's.

5 Conclusion

Malware detection has become an important topic of research due to the rapid growth of malicious code in recent years. As malicious code detection with supervised learning method requires a large number of labeled samples, it is not practical to handle dataset in a large scale. Therefore, collaborative train, as a kind of semi-supervised learning method has been applied in this paper, and we propose a new algorithm called ColSVM combined with ICA. The validity is proved by experiments lastly.

Future work will be oriented on two main directions. Firstly, we will test our algorithm on larger dataset in order to broaden its' application. Next, we will research on multiple views instead of two views to improve the performance of ColSVM further.

6 Acknowledgment

This work was supported by National Natural Science Foundation of China (Grant 61501457 and 61402023).

References

[1] N.Karampatziakis, J.W.Stokes,A.Thomas, and M.Marinescu. Using file relationships in malwareclassification. *In Proceedings of the Conference on Detection of Intrusions and Malware and Vulnerability Assessment*, 7591:1-20, 2012.

[2] E. E. Papalexakis, T. Dumitras, D. H. P. Chau, B. A. Prakash, and C. Faloutsos. Spatio-temporal mining of software adoption & penetration. *In Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 878-885, 2013.

[3] X. Hu, S. Bhatkar, K. Griffin, and K. G. Shin.Mutantx-s: Scalable malware clustering based onstatic features. *In Proceedings of USENIX AnnualTechnical Conference*, pages 187-198, 2013.

[4] Michael Bailey, Jon Oberheide, Jon Andersen, Z. Morley Mao, Farnam Jahanian, Jose Nazario. Automated classification and analysis of internet malware. *International Workshop on Recent Advances in Intrusion Detection*, pages 178-197, 2007.

[5] X.Y. Zhang, Z. Hou, X. Zhu, G. Wu, S. Wang: Robust malware detection with dual-lane AdaBoost. In Proc. IEEE International Conference on Computer Communications (INFOCOM), pp. 1051-1052, 2016.

[6] J.Z.Kolter, and M.A.Maloof. Learning to detect Learning to detect malicious executables in the wild. *In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, Pages 470-478, 2004.

[7] Santos, I.,Laorden, C, and Bringas, P.G. Collective Classification for Unknown Malware Detection. *In Proceedings of the International Conference on Security and Cryptography*, pp. 251-256, 2011.

[8] X. Zhang, J. Cheng, H. Lu, and S. Ma. Selective sampling based on dynamic certainty propagation for image retrieval. In Proc. International Multimedia Modeling Conference (MMM), pp. 425-435, 2008.

[9] X. Zhang, J. Cheng, H. Lu, and S. Ma. Weighted co-SVM for image retrieval with MVB strategy. In Proce. IEEE International Conference on Image Processing (ICIP), pp. 517-520, 2007

[10] X. Zhang: Preference modeling for personalized retrieval based on browsing history analysis. *IEEJ Transactions on Electrical and Electronic Engineering*, 8(S1), pp. 81-87, 2013.

[11] X.B. Zhu, X. Jin, X.Y. Zhang, C.S. Li, F.G. He, L. Wang: Context-aware local abnormality detection in crowded scene. *Science China Information Sciences (SCIS)*, 58(5), pp. 1-11, 2015.

[12] X. Zhang: Effective search with saliency-based matching and cluster-based browsing. *High Technology Letters*, 19(1):105-109, 2013.

[13] X. Y. Zhang, "Simultaneous Optimization for robust correlation estimation in partially observed social network," *Neurocomputing*, 205, pp. 455–462, 2016.

[14] X. Y. Zhang, S. Wang, and X. Yun, "Bidirectional active learning: a two-way exploration into unlabeled and labeled data set," *IEEE Transactions on Neural Networks and Learning Systems*, 26(12), pp. 3034–3044, 2015.

[15] X. Zhang, "Interactive Patent classification based on multi-classifier fusion and active learning," *Neurocomputing*, 127, pp. 200–205, 2014.

[16] X. Y. Zhang, S. Wang, X. Zhu, X. Yun, G. Wu, and Y. Wang, "Update vs. upgrade: modeling with indeterminate multi-class active learning," *Neurocomputing*, 162, pp. 163–170, 2015.

[17] X. Zhang, C. Xu, J. Cheng, H. Lu, and S. Ma, "Effective annotation and search for video blogs with integration of context and content analysis," *IEEE Transactions on Multimedia*, 11(2), pp. 272–285, 2009.

[18] Zhi-Hua Zhou, De-Chuan Zhan, and Qiang Yang. Semi-supervised learning with very few labeled training examples. *In Proceeding of Twenty-Second AAAI Conference on Artificial Intelligence (AAAI),* pages 675-680, 2007.

[19] K. Zhang, X. Yun, X. Y. Zhang, X. Zhu, C. Li, S. Wang: Weighted hierarchical geographic information description model for social relation estimation. *Neurocomputing*, 216: 554-560, 2016.

[20] X. Zhang, C. Xu, J. Cheng, H. Lu, and S. Ma. Automatic semantic annotation for video blogs. *In Proceedings of IEEE International Conference on Multimedia and Expo*, pages 121-124, 2008.

[21] X. Zhang, J. Cheng, C. Xu, H. Lu, and S. Ma. Multi-view multi-label active learning for image classification. In *Proc. IEEE International Conference on Multimedia and Expo*, pp. 258-261, 2009.

[22] Blum A, and Mitchell T. Combining labeled and unlabeled data with co-training. *In Proceedings of the eleventh annual conference on Computational learning theory*, pages 92-100, 1998.

[23] A. Hyvärinen, and E Oja. Independent Component Analysis. *Algorithms and Applications Neural Networks*, 13(4-5):411-430, 2000.

[24] F. Bach, and M. Jordan. Kernel Independent Component Analysis. *Journal of Machine Learning Research*, 3:1-48, 2002.

[25] X. Zhang: Dynamic batch selective sampling based on version space analysis. *High Technology Letters*, 18(2): 208-213, 2012

[26] https://en.wikipedia.org/wiki/Support_vector_machine