An Immune Concentration Based Virus Detection Approach Using Particle Swarm Optimization

Wei Wang[1,2], Pengtao Zhang[1,2], and Ying Tan[1,2]

[1] Key Laboratory of Machine Perception, Ministry of Eduction, Peking University
[2] Department of Machine Intelligence, School of Electronics Engineering and Computer Science, Peking University, 100871, P.R. China
ytan@pku.edu.cn

Abstract. This paper proposes an immune concentration based virus detection approach which utilizes a two-element concentration vector to construct the feature. In this approach, 'self' and 'nonself' concentrations are extracted through 'self' and 'nonself' detector libraries, respectively, to form a vector with two elements of concentrations for characterizing the program efficiently and fast. Several classifiers including k-nearest neighbor (KNN), RBF neural network and support vector machine (SVM) with this vector as input are then employed to classify the programs. The selection of detector library determinant and parameters associated with a certain classifier is here considered as an optimization problem aiming at maximizing the accuracy of classification. A clonal particle swarm optimization (CPSO) algorithm is used for this purpose. Experimental results demonstrate that the proposed approach not only has a very much fast speed but also gives around 98% of accuracy under optimum conditions.

Keywords: Immune Concentration, Clonal Particle Swarm Optimization, Virus Detection.

1 Introduction

Computer virus has been considered as an increasingly serious problem while evolving with the rapid development of computer environments, such as operating system, network, etc. There are two main virus detection methods: signature-based method and malicious activity detection[1]. With novel advanced technologies being widely used in manufacturing new viruses, polymorphic viruses can change their signatures while spreading. Heuristic methods which is more sophisticated, like malicious activity detection, can be used to identity unknown viruses. Among those heuristic methods, AIS as a dynamic, adaptive and distributed learning system, protects benign files against virus invasion by distinguishing 'nonself' from 'self'[2]. It is often combined with traditional classification algorithms to construct a virus detection system, including Naïve Bayes, Support Vector Machine (SVM), Artificial Neural Network (ANN), k-nearest neighbor (KNN), and hybrid approaches[3][4].

In this paper, a AIS method is used to generate a two-element concentration vector as the feature vector for virus detection. 'Self' and 'nonself' detector libraries contain the

Y. Tan, Y. Shi, and K.C. Tan (Eds.): ICSI 2010, Part I, LNCS 6145, pp. 347–354, 2010.

bit strings which are utmost representative of benign and virus programs, respectively. Document frequency and information gain of a fixed length fragment are used to decide whether the fragment can be taken as a detector. 'Self' and 'nonself' concentrations are constructed by using 'self' and 'nonself' detectors in the libraries to traverse a program, respectively. Unlike the traditional anti-virus methods, our proposed model helps reduce the feature dimensionality. The feature with these two elements is the input of a classifier, and one binary value is the output. As a result, the detector library determinant and parameters associated with the classifier become the vector that we need to optimize by using a clonal particle swarm optimization (CPSO) algorithm[5]. The optimal vector is the one whose cost function associated with classification is minimum, namely the one make the accuracy of classification maximum.

Comprehensive experiments are conducted on a public virus data set in the pervious works[6][7]. 10-fold cross validation is used to measure the performance. Comparisons on performance are also made among different classifiers including k-nearest neighbor (KNN), RBF neural network and support vector machine (SVM). Experimental results shows that the proposed approach achieves more than 97% detection rate by just using a two-element vector, so outperforms current approach.

The paper is organized as follows. In Section II, algorithmic implementations of our proposed approach are elaborated in detail with CPSO-based optimization process for detector library determinant and parameters associated with a certain classifier. Several experimental results are reported in Section III.

2 Immune Concentration Based Virus Detection Approach

2.1 Overview of Our Proposed Approach

Our proposed approach is mainly divided into three parts. (1)Generate 'self' and 'nonself' detector libraries. (2)Extract the two-element concentration vector of each training sample as the input of a classifier and use CPSO to optimize detector library and parameters associated with the classifier. (3)Several trained classifiers including KNN, RBF neural network and SVM are used to detect the testing samples characterized by optimized concentration vectors. The outline of our proposed approach is described in algorithm 1 and each step in detail is discussed in the following sections.

2.2 Generation of Detector Libraries

The operating principle of generating 'self' detector library and 'nonself' detector library is shown in algorithm 2.

One meaningful computer instruction is 8 or 16 bits normally. Here a fixed length L-bit fragment of binary data which is considered containing appropriate information of functional behaviors is taken as the detector to discriminate virus from benign program. The length L is set not too short to discriminate 'self' and 'nonself' and not too long to make virus-special data hidden in the binary data of files. Intuitively, the fragment that appears most in virus programs while seldom in benign programs is a good representative of a virus. Consequently, a sliding window (size: L bits, the overlap: [L/2] bits)

Algorithm 1. Algorithm for Virus Detection

Generate *'self'* and *'nonself' detector libraries* from training set
The sizes of the libraries are decided by parameter m which corresponds to proportional selection of the *candidate detectors*

for each the sample in training set **do**
Extract the *two-element concentration vector* of each training sample through the two detector libraries as the input of a classifier
end for

Use these *feature vectors* to train a certain classifier
Use *CPSO* to optimize *detector library determinant* m and *classifier-related parameters*

while Algorithm is running **do**
if a program is detected **then**
Characterize the sample by *concentration vector* through trained *'self'* and *'nonself' detector libraries*
Use trained classifier to predict the *label* of the program
end if
end while

is used to count the fragment's document frequency in the virus and benign programs, which can reflect its tendency to be a virus or a benign file. In this paper, L is set to 32.

After counting the document frequency of each fragment, the tendency $T(X)$ of fragment X is defined in formula 1.

$$T(X) = P(X = 1|C_v) - P(X = 1|C_s) \tag{1}$$

$P(X = 1|C_v)$means document frequency of fragment X appears in virus samples of training set;
$P(X = 1|C_s)$means document frequency of fragment X appears in benign samples of training set.

If each fragment is extracted to form a dictionary, the size of this dictionary would be very large. The features that appears in most of files are not relevant to separate these files because all the classes have instances that contain these features. So with the number growth of features, the cost of computing would be increasing but the accuracy may not be improved and even worse. We reduces the number of fragments to generate 'self' and 'nonself' detector libraries based on information gain ratio of each detectors to make the detectors more representative. The process is described in Algorithm 2, in which $m$is a parameters to be adjusted, means proportional selection of all the fragments. The information gain is defined in formula 2.

$$IG(X, C) = \sum_{x\in\{0,1\},c\in\{C_v,C_s\}} P(X = x \wedge C = c) \cdot \log_2 \frac{P(X = x \wedge C = c)}{P(X = x) \cdot P(C = c)} \tag{2}$$

$P(X = 0|C_v)$means document frequency of fragment X which doesn't appear in virus samples of training set;

$P(X = 0|C_s)$means document frequency of fragment X which doesn't appear in benign samples of training set.

Algorithm 2. Algorithm for Generation of Detector Libraries

Initialize *'self'* and *'nonself' detector libraries* as $\varnothing$

while Algorithm is running **do**
  for each *fragment X* in the sample of training set **do**
    Calculate the *tendency* of *fragment X* by formula 1
    Calculate the *information gain* of *fragment X* by formula 2
  end for

  for each *fragment X* in the sample of training set **do**
    if $IG(X) > m$ **then**
      if $T(X) < 0$ **then**
        add *fragment X* into *'self' detector library*
      else
        add *fragment X* into *'nonself' detector library*
      end if
    end if
  end for
end while

Extract $P_s\%$ of fragments to form *'self' detector library* and $P_n\%$ of fragments to form *'nonself' detector library*, P_s, P_n are decided by parameter m

2.3 Construction of Feature Vector

For constructing a feature vector, a sliding window with [L/2] bits overlap is used to get the 'self' concentration and 'nonself' concentration of a program i, which are defined in formula 3,4, respectively.

$$BC_i = \frac{BN_i}{N_i} \tag{3}$$

$$VC_i = \frac{VN_i}{N_i} \tag{4}$$

Where BC_i and VC_i denotes the 'self' and 'nonself' concentration, respectively; BN_i is the number of detectors appearing in both program i and 'self' detector library; VN_i is the number of detectors appearing in both program i and 'nonself' detector library; N_i denotes the number of different L-bit fragments in the program.

2.4 Classification Parameters Selection

The feature constructed by two-element concentrations is the input of a classifier, and one binary value is the output. The generation of 'self' and 'nonself' detector libraries, which in turn determine the two-element concentration vector uniquely, here is an optimization problem. The optimal vector is the one make the accuracy of classification maximum.

Algorithm 3. Algorithm for Feature Construction

For a program to be detected, calculate the number N_i of different *L-bit fragments* using a sliding window with [L/2] bits overlap

Initialize $BN_i = 0, VN_i = 0$

for each different *L-bit fragment* in the program **do**
  if it appears in *'self' detector library* **then**
    BN_i++;
  else if it appears in *'nonself' detector library* **then**
    VN_i++;
  end if
end for

self-concentration=BN_i/N_i
nonself-concentration=VN_i/N_i

The vector that we need to optimize $P^* = \{P_s^*, P_n^*, P_1^*, P_2^*, \cdots, P_m^*\}$ is composed of detector library determinant m, and parameters $P_1^*, P_2^*, \cdots, P_m^*$ associated with a certain classifier. When m is set to different values, P_s^*, P_n^* would take different values, different detector libraries are obtained. So for a program to be characterized, a 'self' concentration which represents its similarity to benign program and a 'nonself' concentration which represents its similarity to virus can be constructed. $P_1^*, P_2^*, \cdots, P_m^*$ are classifier-related parameters which influence the performance of a certain classifier. Different classifiers hold different parameters and lead to different performance. For examples, parameters associated with KNN include number of nearest neighbors and the ways of distance measures. SVM-related parameters that determine the position of optimal hyperplane in feature space, include cost parameter C and kernel parameters. Finally, the trained classifiers are used to classify detecting samples.

The optimal vector is the one whose cost function associated with classification is minimum, namely the one make the accuracy of classification maximum. The cost function $CF(P)$ can be defined as

$$CF(P) = Err(P) \tag{5}$$

where $Err(P)$ is the classification error measured by 10-fold cross validation on the training set.

Input vector include two parts: detector library determinant m, and $P_1^*, P_2^*, \cdots, P_m^*$ these classifier-related parameters. Output is to find a P^*, so that

$$CF(P^*) = Err(P^*) = \min_{\{m, P_1^*, P_2^*, \cdots, P_m^*\}} Err(P) \tag{6}$$

3 Experimental Results

3.1 Data Set

Experiments are conducted on a public virus data set in the pervious works[6][7]. "Cilpku08" data set, which can get from http://www.cil.pku.edu.cn/malware, includes

3547 viruses classified to 685 families based on their properties. we select 915 benign files and 900 virus to test our method. 1815 programs are divided into ten partitions with approximately equal numbers of virus and benign programs. 10-fold cross validation is used to measure the performance.

3.2 Experiments on Different Concentrations

Here, m is a parameters to be adjusted, which means proportional selection of all the fragments. Different 'self' and 'nonself' concentrations correspond to $P_s\%$ of fragments extracted to form 'self' library and $P_n\%$ of fragments extracted to form 'nonself' library. P_s, P_n are decided by parameter m. A linear SVM with default parameters is used as the classifier. m ranges from 10% to 100% with a step 10%. Figure 1 shows the accuracy with different m value.

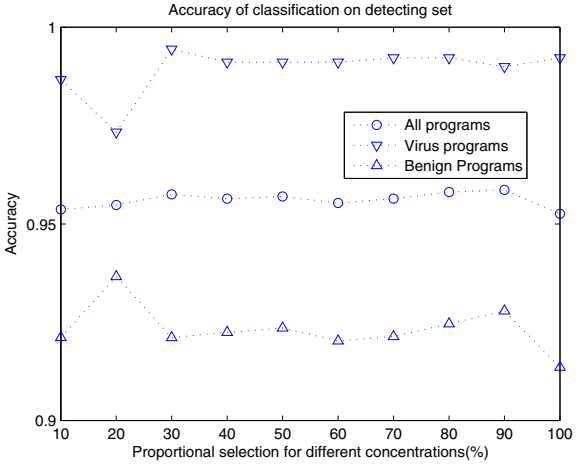

Fig. 1. Accuracy with different concentrations on detecting set by SVM

It is easy to see that different *m* gets almost the same result. In order to get a small detector library, we set *m* as 10%.

3.3 Classification Parameter Optimization

The selection of detector library determinant m and the classifier-related parameters, P_1^*,P_2^*,$\cdots$,P_m^*, is a dynamic optimization process.

Parameters associated with KNN include number of nearest neighbors K and the ways of distance measures, K is optimized in the integer number interval [1, 20], the ways of distance measures are chosen among $euclidean$, $cityblock$, $cosine$, $correlation$. For RBF neural network, the spread is optimized in real number interval [1, 5]. For SVM with linear kernel the cost parameter C is optimized in real number interval [1, 200]. For SVM with RBF kernel the cost parameter C and γ are optimized in real number interval

Table 1. Average detection rates on the detecting set under optimum conditions

	Empirical classification designs			Optimized classification designs		
Methods	$All(\%)$	$Virus(\%)$	$Benign(\%)$	$All(\%)$	$Virus(\%)$	$Benign(\%)$
KNN	95.76	94.00	97.50	98.90	98.78	99.02
RBF NN	97.57	97.56	97.59	97.96	97.78	98.14
SVM(Linear)	95.86	99.00	92.79	96.92	99.22	94.66
SVM(RBF)	95.81	99.44	92.22	98.62	98.00	99.24

[1, 200] and [1, 20], respectively. m is optimized in the integer number interval [5, 100]. The maximum number of generations is set to be 200 as the stop criterion, the number of particles in a swarm is 20.

The randomness of CPSO leads to the performance and obtained parameters vary slightly, therefore the average results of ten independent classes of experiments are used to evaluate tests, which is more reasonable. The average performances of empirical and optimized classification designs are reported in Table 1.

The results show that the optimized classification design resulted in a 2% increase in detection rate compared with the empirical classification design in average. The CPSO-based method has got an obvious advantage.

4 Conclusions

The immune concentration based virus detection approach proposed in this paper took a two-element concentration vector as the virus feature and employed several classical classifiers to detect virus. When parameters were optimized, the accuracy on detecting set reached 98%.

Different from traditional binary data mining methods, our method established an uniform framework for a general and systematical approach of feature construction and reduced the dimensionality to make the training and detecting faster. Also, the proposed feature extraction optimization approach attained better comparable results than the approach with empirical tentative parameters setting. The new method is easier without sacrificing performance.

Acknowledgement

This work was supported by the National High Technology Research and Development Program of China (863 Program), with grant number 2007AAOIZ453, and partially supported by National Natural Science Foundation of China (NSFC), under grant number 60673020 and 60875080. This work was also in part financially supported by the Research Fund for the Doctoral Program of Higher Education (RFDP) in China.

References

1. Henchiri, O., Japkowicz, N., Nathalie, J.: A feature selection and evaluation scheme for computer virus detection. In: Sixth International Conference on Data Mining, pp. 891–895 (2006)

2. Kephart, J.O.: A biologically inspired immune system for computers. In: Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems, pp. 130–139 (1994)
3. Schultz, M.G., Eskin, E., Zadok, F., Stolfo, S.J.: Data mining methods for detection of new malicious executables. Security and Privacy, 38–49 (2001)
4. Wang, J., Deng, P.S., Fan, Y., et al.: Virus detection using data mining techniques. In: IEEE 37th Annual 2003 International Carnahan Conference on Security Technology, pp. 71–76 (2003)
5. Tan, Y., Xiao, Z.: Clonal particle swarm optimization and its applications. In: IEEE Congress on Evolutionary Computation, pp. 2303–2309 (2007)
6. Wang, W., Zhang, P.T., Tan, Y., He, X.G.: A hierarchical artificial immune model for virus detection. In: 2009 International Conference on Computational Intelligence and Security, pp. 1–5 (2009)
7. Chao, R., Tan, Y.: A virus detection system based on artificial immune system. In: 2009 International Conference on Computational Intelligence and Security, pp. 6–10 (2009)
8. Kerchen, P., Lo, R., Crossley, J., et al.: Static analysis virus detection tools for unix systems. In: 13th National Computer Security, pp. 4–9 (1990)
9. Hofmeyr, S.A., Forrest, S., Somayaji, A.: Intrusion detection using sequences of system calls. Journal of Computer Security, 151–180 (1998)
10. Tan, Y., Deng, C., Ruan, G.C.: Concentration based feature construction approach for spam detection. In: International Joint Conference on Neural Networks, pp. 3088–3093 (2009)
11. Deng, P.S., Wang, J., Shieh, W., et al.: Intelligent automatic malicious code signatures extraction. In: IEEE 37th Annual 2003 International Carnahan Conference on Security Technology, pp. 600–603 (2003)
12. Preda, M.D., Christodorescu, M., Jha, S., et al.: A semantics-based approach to malware detection. In: 34th Annual Symposium on Principles of Programming Languages, vol. 42(1), pp. 377–388 (2007)
13. Ruan, G.C., Tan, Y.: A three-layer back-propagation neural network for spam detection using artificial immune concentration. Soft Computing 14, 139–150 (2010)
14. Moskovitch, R., Stopel, D., Feher, C., et al.: Unknown malcode detection via text categorization and the imbalance problem. In: IEEE International Conference on Intelligence and Security Informatics, pp. 156–161 (2008)
15. Drucker, H., Wu, D., Vapnik, V.N.: Support vector machines for spam categorization. IEEE Transactions on Neural Networks 10, 1048–1054 (1999)