社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2004－ICS－138　(11)
2004／12／4

A Machine Learning Approach to Anti-virus System

HOANG KIEM[†], NGUYEN THANH THUY[††]
and TRUONG MINH NHAT QUANG[†††]

Abstract ─ As the development of Internet nowadays, computer viruses have been a hot story. They more and more seriously infect, destroy and steal data from many computer systems in the world. Therefore, it is necessary to improve the identifying method of an anti-virus system. In this paper, we would like to introduce an intelligent anti-virus system by using Machine Learning Rule-based Approach. The basic tasks of this project consist of modeling knowledge base, forming rule sets to recognize known viruses, diagnosing and discovering interesting rules from virus database using Data Mining algorithms, finding hidden attributes/behaviors and predicting unknown viruses with Genetic Programming tools.

Keywords ─ Machine Learning, Genetic Programming, Data Mining, Computer Virus, Anti-virus.

1. Introduction

As the development of Internet nowadays, computer viruses become very widespread. They more and more seriously infect, destroy and steal data, which directly influence the network security and data safety of many computer systems in the world [5][9][10]. In addition, most current viruses usually encrypt their bodies to hide themselves. Although anti-virus systems (AVs) try to clean them day-by-day, new viruses are more than AVs can handle and infected areas are more than AVs can treat [2][8]. In this situation, AVs have to face problems such as omitting some known viruses, warning dramatically for a pure dataset or unrecognizing certain potential harmful codes.

To improve the identifying method of an AVs, we apply a Machine Learning rule-based approach that involves the basic tasks: modeling knowledge base, forming rule sets to recognize known viruses, diagnosing and discovering interesting rules using Data Mining algorithms, finding hidden attributes and predicting unknown viruses with genetic programming tools.

2. Modeling Knowledge Base

At first, we define virus classes corresponding to widespread viruses such as F(ile)-viruses, B(oot)-viruses, I(nternet Worm)-viruses, T(ext)-viruses, E(mail)-viruses, Ba(ckdoor)-viruses, Tr(ojan Horse)-viruses,… In each diagnose purpose; a class is defined with its particular characteristics. In general, a standard virus class has an object-oriented form:

Object: Virus family identification
Property: Attributes/behavior
Method: Treatment/direction outline

[†] University of Natural Sciences Ho Chi Minh City
[††] Hanoi University of Technology
[†††] Cantho Inservice University

Using this model, we suggest three knowledge bases as follow:

- KB1: examine determined diseases
- KB2: diagnose new determined diseases
- KB3: diagnose unusual diseases

When KB1 is used to treat most known viruses, KB2 is used to detect some variances of known viruses, and KB is used to predict unknown viruses. In this assignment, the AV expert system also requires the decisions in different usage levels: end-users, technicians and system administrators. To lessen human experts' hard work, we propose a model called Knowledge Base Increasing Diagram (Figure 1).

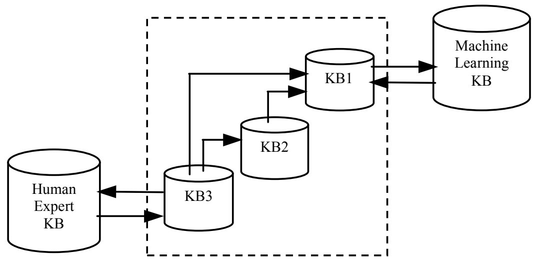


Figure 1: *Knowledge Base Increasing Diagram*

3. Forming Rule Sets

Scanning data to recognize all characteristics of viral codes in ID-virus library, an AVs needs to build the set V_K of K vectors-Virus signatures $V_K = \{v_1, v_2, \ldots, v_k\}$ and to determine the existence of v_i in the diagnosed set S. The conventional rule set has the form:

$$R: p_1 \wedge p_2 \ldots \wedge p_n \Rightarrow q$$

where p_i represents the virus signature/behavior and q is a result/conclusion of the process[7].

The advantage of this formula is clear: AVs can identify almost known viruses from data test.

However, the searching algorithms have been failed when working without virus signatures. So AVs cannot recognize any variance/unknown virus.

Our solution is a composition of a conventional AVs and a Machine Learning (ML) approach. The Knowledge Base Increasing Diagram illustrates to this strategy: all of the users' decisions at each level will be transferred to KB1, the simplest knowledge base that contains all rule sets to recognize all current viruses in the market.

Similar to a Knowledge Discovering in Databases - KDD systems, our system composes the processes: Data Selection, Cleaning, Enrichment, Coding, Data Mining and Reporting. In the most important stage - Data Mining, we will present an advanced forming rule set to recognize variance/unknown viruses from data test.

4. Data Processing as an Examinating Phase

The examination process composes the first three stages of the data processing. They are data selection, cleaning, and enrichment. Depending on KB levels, the correspondent facilities of this system will process data test automatically or manually.

Data selection tools help to pick up data required to test from a huge database. For example, if we want to diagnose Internet worms from the incoming data of a server mail, the data selection process will allocate the addresses of incoming mail folder system. This knowledge is trained manually by a human expert, automatically by default settings or learnedly by observation of using behaviors.

The objective of the cleaning operation is a complete process to prevent the rough/duplication data. In that example, this operation will extract all executable elements from incoming data folder and assign them into an appropriate diagnose space.

Data enrichment is the last operation of the data processing. In fact, it is a data analyzer to discover and multiply some hidden viral codes from data test. It returns a copy of data test as a 'virtual' infected data to be passed to the coding phase. For example, we want to recognize virus class V by applying the rule R_v to search V_i vectors in the observation S. It is a good idea to generate S' from S, and then assign it to another process to diagnose some potential viral codes of involved virus class U.

5. Data Coding as a Part of the Diagnosing Phase

Coding can be an infinitive number of difference codes that are related to any number of different potential patterns we would like to find. In the ML-AVs, a data coding function class called Procedural Code Assignment was built. Using some linear classification methods like NNSRM-Nearest Neighbor Rule-based Structural Risk Minimization, we have gotten an observation space in which a labeled data point is related to a treatment function class.

Suppose a set of diagnosed objects $\{x_1, x_2, x_3,\dots,x_n\}$ is vectors in observation space X. The objective of this stage is analyzing similar characteristics of x_i points to assign them into **Class 1** or **Class 2** - Possibly or impossibly infected by internet worm W respectively. This suggestion is based on the inertia of some special objects, which permits us to decrease the number of data points in huge data tests. In its simplest form, the NN classifier can be formulated as:

$$f_s(x) = y_{i'} \text{ where } \| x_{i'} - x\|_\chi = \min_{x_i \in S}\| x_i - x\|_\chi$$

where S is the reference set of sample data points from classes 1 and 2.

The primary problem with the classical NN rule is the computational and data storage. Given an uncategorized data point x, we need to calculate the distance from x to every data points in the training set [3]. This approach is to minimize the upper bound on the test error $R^{emp}(f)$ in equation

$$R(f) \le R^{emp}(f) + \varepsilon(f, h, \nu)$$

directly by minimizing the confidence $\varepsilon(f,h,\nu)$ while fixing R^{emp} at zero. Classification algorithm is described as follow:

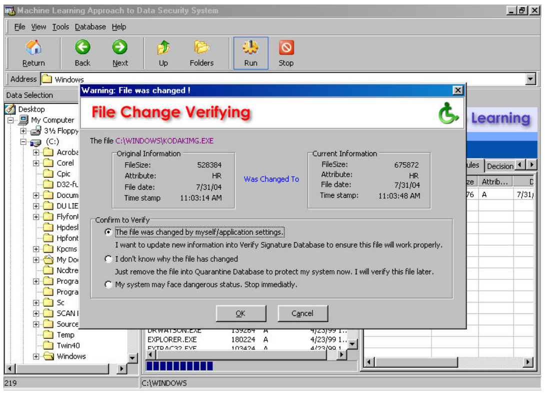
```
setup phase:
  - compute all pair wise distance $\|x_i - x_j\|_\chi$ for all
    $x_i$ in class 1 and $x_j$ in class 2
  - sort these distance in ascending order, $d^{(0)}$,
    $d^{(1)}$,…
training phase:
  - initialize u=1, $S=\{x_i, x_j\}$ where $\|x_i - x_j\|_\chi = d^{(0)}$
  - while $R^{emp}(fs) > 0$, do
        find $x_i$ in class 1 and $x_j$ in class 2 such that
        $\|x_i - x_j\|_\chi = d^{(u)}$
        update $S \leftarrow S \cup \{x_i, x_j\}$
        increment u
```

When all data point are labeled, we pass this process to pre-diagnose stage by eliminating all elements unnecessary to test [3]. It makes the amount of data points decreases and speeding-up the system performance. The algorithm is guaranteed to converge. In the worse case, the reference set will consist of all the training samples; hence the classifier will have zero training error. In this low-cost algorithm, the computation complexity is small.

6. Data Mining as an Enhancement of the Diagnosing Phase

This is the most important stage of an ML-AVs. In general, this stage uses two principal techniques: forward inference and backward inference[7]. When the first inference is used to recognize known virus V by analyzing the rule of the existence v_i code in observation space, the second inference is used to find the symptoms of an infection to ensure that an unknown virus is hiding in the system. On the other hand, the diagnosing of known viruses runs the identification rule sets, and the one of unknown viruses discovers interesting rules from data test. In some cases, both inference techniques are composed to give the final results.

There are many virus types (so we have the same number of virus classes). Each appropriate data mining technique is applied for a virus class. For example, to diagnose an unknown B-virus, we setup a virtual machine with a tree decision algorithm built-in [6]. This model helps to recognize most unknown B-viruses without using ID-virus library. To diagnose another virus like F-classes and I-classes, we apply a reduced algorithm of association rule running on a database named VerisignDB. Its results are also similar to (Figure 2).


Machine Learning Approach to Data Security System

Warning: File was changed !

File Change Verifying

The file C:\WINDOWS\KODAKIMG.EXE

Original Information
FileSize: 528384
Attribute: HR
File date: 7/31/04
Time stamp 11:03:14 AM

Was Changed To

Current Information
FileSize: 675872
Attribute: HR
File date: 7/31/04
Time stamp: 11:03:48 AM

Confirm to Verify
[✓ The file was changed by myself/application settings.]
I want to update new information into Verify Signature Database to ensure this file will work properly.
I don't know why the file has changed
Just remove the file into Quarantine Database to protect my system now. I will verify this file later.
My system may face dangerous status. Stop immediatly.

Figure 2: *The MLA2AES Association Rule Set detected an unknown virus infection behavior*

In our last research, an algorithm based on genetic programming has just applied to recognize some encrypted viruses. In an experiment period, we give two virus attributes that are independent of data format: *Directory Reallocation Addresses* ($A1 \in [0.0, 1.0]$) and *Indexed Section Pages* ($A2 \in [0.0, 1.0]$). They are used as an additional classifying decision. In Figure 3, we can see a graphical example of how cases are assigned to classes. In it the corresponding area graphically represents each rule. The two rules are:

Rule 1: IF ($a \le A1 \le c$) AND ($A2 \le n$) THEN Class 1

Rule 2: IF ($b \le A1 \le d$) AND ($m \le A2 \le p$) THEN Class 2

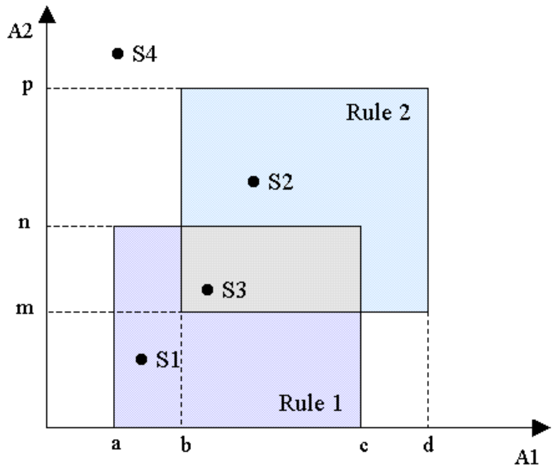


Figure 3: *Examples of assignment of indeterminate cases to only one class*

We have some indeterminate cases:

- If the example does not satisfy any of the found rules, it is assigned to the class from the frontier of which it has the lowest distance.
- If the example satisfies more than one of the found rules, it is assigned to the class from the frontier of which it has the highest distance.

As we can visually evaluate, the sample S1 can be easily seen as belonging to class 1. Likewise the sample S2 can be classified in class 2.
S3 is closer to the frontier of Rule 1 than it is to the frontier of Rule 2, it is more than internal to Rule 2, hence it can be assigned to class 2 rather than to class 1.

Similarity, let us consider S4. It is closer to the frontier of Rule 2 than it is to the frontier of Rule 1, so it can be seen as 'missed' by Rule 2 by a smaller quantity than it is 'missed' by Rule 1. So S4 is assigned to class 2[1].

However, to ensure the exactness of anti-encrypt/anti-polymorphism algorithms, we need to add other virus attributes. When the attributes A_i are increased, the rules are composed by several ANDs, ORs and NOTs, a procedure has been devised which computes the distance of a sample from the frontier of the rule. With genetic programming, we can solve this problem by specifying the individuals as the rule encodes in a tree structure for which limits on the tree depth. The tree nodes are either functions or terminals. The function set consists of the logical connectives {AND, OR, NOT} and the relational operators

{<,≤, =, ≥, >}. The terminals are simply A_i or values in the domain of A_i. The individuals in the initial population are randomly created by selecting from the function or terminal sets. The new elements in the population are generated by means of three operators: *crossover*, *copy* and *mutation*. To *crossover*, two parent individuals are selected and a sub tree is picked on each one. Then crossover swaps the nodes and their relative sub trees from one parent to the other. The *copy* operator simply chooses an individual in the current population and copies it without changes into the new population. The *mutation* operator can be applied to either a function node or a terminal node. A node in the tree is randomly selected. If the chosen node is a terminate it is simply replaced by another terminate.

The population is constituted by the possible class description, that is to say sets of conditions on attributes of the objects to classify. In these cases it is possible to use statistical fitness functions. These functions can be explained as the difference between the actual number of examples for which the rule classifies properly the belonging or not belonging to the class and the number of examples for which there is an incorrect classification whether the conditions or the prediction are not satisfied [1].

7. Experimentation

We have setup a Machine Learning Approach to Anti-virus Expert System-MLA2AES (Figure 4) and test it on 36,178 samples with a data size of 10,000,000 KB. The testing computer has an Intel Celeron 1.2GHz CPU with 256 MB RAM running on Windows XP Professional (Table 1)

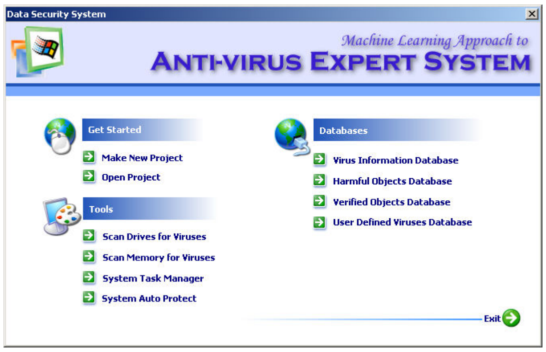


Figure 4: *Machine Learning Approach to Antivirus Expert System*

Table 1: *Experiment of AVs performance*

No.	Applications	Manufactures	Time (s)	Viruses
1	FixSasser Tool	MLA2AES	195	1
2	FxSasser Fixtool	Symantec Corp.	196	1
3	RmNetsky	Grisoft Anti-virus	337	1
4	AntiFunLove	Kaspersky Lab	390	1
5	ClrAV	Kaspersky Lab	728	123
6	AVG 7.0	Grisoft Anti-virus	1025	263
7	Scan for Viruses	MLA2AES	324	650
8	Norton Anti-virus	Symantec Corp.	1095	1565

As we see:

- With minimal virus sample, MLA2AES is as fast as/faster than the others.
- With more virus samples, MLA2AES classifying speed is much better.

In order to evaluate MLA2AES training speed conventionally, we call:

- V_c: the number of virus samples
- T_0: the duration of diagnosing time when $V_c=1$
- T: the duration of diagnosing time when $V_c>1$
- T_1: the average duration of diagnosing time for the appearance of a virus when $V_c>1$
- T_2: the average duration of diagnosing time for an added virus when $V_c>1$
- V_e: the conventional number of virus samples
- C_e: the conventional size of data test (KB)
- T_e: the conventional duration time (s)
- S_e: the conventional diagnosing speed (KB/s)

T_1, T_2, T_e and S_e are calculated as follow:

- $T_1=T / V_c$
- $T_2= (T-T_0)/(V_c-1)$
- $T_e = T + (V_e-V_c)\times T_2$
- $S_e = C_e/T_e$

With a conventional test where V_e=2,000; C_e=10,000,000 KB; we have the result in Table 2.

Table 2: *Conventional experiment of AVs speeds*

AVs	T_0	T_1	T_2	T	T_e	S_e (KB/s)
MLA2 AES	195	0.498	0.1987	324	592.245	16884.9
NAV	196	0.699	0.5748	1095	1345.038	7434.734
AVG	337	3.897	2.6259	1025	5586.188	1790.129
KL	390	5.918	2.7704	728	5928.041	1686.898

We have also the chart of conventional experiment of AVs speeds (Figure 5)

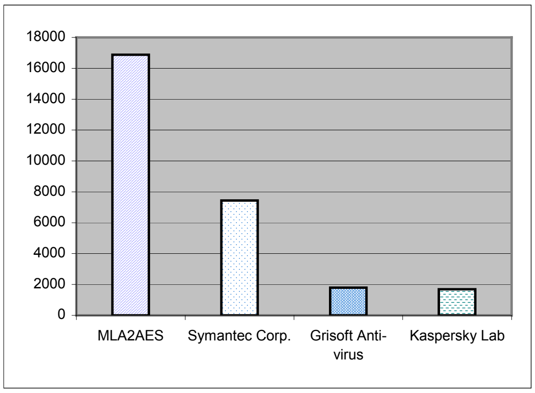


Figure 5: *Chart of conventional experiment of AVs speeds*

8. Conclusion and Future Works

By using several Data Mining Algorithms such as Tree Decision and Association Rule, we have detected some unknown virus infections in our experiment systems. In addition, some hidden variances with different self-extractors of such I-Classes as Lovgate, Nimda, Sobig, Nachi, Bugbear, Yaha, Netsky and Bagle are also detected.

From the experiment results, it is clear that MLA2AES is suitable to diagnose the computer viruses event if working on unknown/variance virus families. In addition, this approach helps to classify data test to eliminate a large amount of unnecessary data points from observation space. So the application performance is better.

In the initiation phase of this project, we have only applied some linear classifier for the classes Internet worm, Backdoor, Trojan horse, Hack tool, Spy ware,… For nonlinear virus classes like Macros virus or Text virus, we have to choose another classification methods. Next, we also have to improve the MLA2AES effectiveness to predict several new virus classes in the future market.

References

1) De Falco, A. Della Cioppa, E. Tarantino. *Discovering interesting classification rules with genetic programming*. Elsevier Science, 30 October 2001.
2) Jeffrey O. Kephart, Gregory B. Sorkin, Morton Swimmer, and Steve R. White. *Blueprint for a Computer Immune System*. Proceedings of the 1997 International Virus Bulletin Conference, San Francisco, California, October, 1997.
3) Karacah B., Rajmanath R., Wesley E. Snyder. *A Comparative of Structural Risk Minimization by Support Vector Machines and Nearest Neighbor Rule*. Elsevier Science, 5 September 2003.
4) Krzysztof J. Cios, Witold Pedrycz, Roman W. Swiniarski. *Data Mining, Methods for Knowledge Discovery*. USA, Kluwer Academic Publishers, 1998
5) Matt Richtel. *Super-Fast Computer Virus Heads Into the Workweek*. New York Times, Technology Section, March 29, 1999.
6) Nguyen Thanh Thuy, Truong Minh Nhat Quang. *A Global Solution to Anti-virus Systems*. The Proceedings of the 1st International Conference on Advanced Communication Technology. 10-12 February 1999, Muju-Korea, 374-377.
7) Nguyen Thanh Thuy, Truong Minh Nhat Quang. *Expert System Approach to Diagnosing and Destroying Unknown Computer Viruses.* The Scientific Conference Proceedings of the 5th ASEAN Science and Technology Week. 10-1998.
8) Steve R. White, Morton Swimmer, Edward J. Pring, William C. Arnold, David M. Chess, John F. Morar. *Anatomy of a Commercial-Grade Immune System*. IBM Thomas J. Watson Research Center, 1999.
9) Steve R. White, Jeffrey O. Kephart, and David M. Chess. *Computer Viruses: A Global Perspective*. Proceedings of the Fifth International Virus Bulletin Conference, Boston, 1995, pages 185-191.
10) Steve R. White, Jeffrey O. Kephart, and David M. Chess. *The Changing Ecology of Computer Viruses*. Proceedings of the Sixth International Virus Bulletin Conference, Brighton, UK, 1996, pp. 189-202.