ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

DETECTION OF COMPUTER VIRUSES USING WELM_ FGA_ FBFO

Ankur Singh Bist
* Computer Engineering K.I.E.T, Ghaziabad.

ABSTRACT

Computer viruses are big threat for our society .The expansion of various new viruses of varying forms make the prevention quite tuff .Here we proposed WELM_FGA_ FBFO to detect computer viruses. The proposed method efficiently detects static and nature changing viruses.

KEYWORDS: WELM_FGA_ FBFO, Extreme Learning Machine.

INTRODUCTION

There are various processes that have been used in the direction of classification of computer viruses from normal files that will finally lead to virus detection. Machine learning techniques are widely used in this direction. As statistics says that the attacks of malicious codes are increasing day by day so there is requirement of strong techniques that can be used for their detection. Malicious code designers use lot of techniques that are difficult to analyse and detect. The static methods also seems not to work in the case where every time there are rapid dynamicity from attacker side so now a days main focus is going on towards the methods that are dynamic and are able to detect zero day computer viruses [1].

The rise in the malicious threats like computer viruses activities are required to be handled and observed strongly to make certain defence that can stand as a saviour of security domain. Other types of malware are:

1. Worms
2. Trojan horse
3. Botnets
4. Adware
5. Spyware

The mutating behaviour of metamorphic viruses is due to their adoption of code obfuscation techniques like dead code insertion, Variable Renaming, Break and join transformation. Figure 1 shows the assembly file of the virus code. Computer viruses like polymorphic viruses and metamorphic viruses use more efficient techniques for their evolution so it is required to use strong models for understanding their evolution and then apply detection followed by the process of removal. Code Emulation is one of the strongest ways to analyze computer viruses but the anti-emulation activities made by virus designers are also active [4][5][6][7][8].

Figure 1: Assembly code of Virus File

Extreme Learning Machine

Extreme Learning Machine (ELM) [1] is based on least square solution and was originally proposed for single hidden layer feed-forward network. The advantage of this is, rather than tuning of all the parameters here, hidden neurons

and input weight, are chosen randomly. Here, hidden nodes may be sigmoid additive nodes or Gaussian kernel nodes. Hidden node parameters are randomly taken and output node parameter is calculated using Moore-Penrose inverse. This algorithm is thousand times faster than past learning methods such as, back propagation and also reaches the smallest training error. Due to these features, generalization performance of this algorithm is also good. We have given data (x_p,t_p) for N random different samples where, $x_p=[x_{p1},x_{p2},\ldots\ldots\ldots,x_{pn}]^t \epsilon R^n$ and $t_p=[t_{p1},t_{p2},\ldots\ldots\ldots,t_{pm}] \epsilon R^m$
Activation function g(x) with L hidden nodes is defined as

$$\sum_{p=1}^{L} \beta_p \,.\, g_p(x_q)=\sum_{p=1}^{L} \beta_p \,.\, g(w_p \,.\, x_q+b_p) = o_q \quad \text{for } q=1,2,3,\ldots\ldots\ldots N$$

Where $w_p=[w_{p1},w_{p2},\ldots\ldots\ldots w_{pn}]^t$ weight vector connecting p^{th} hidden node and input nodes.
$\beta_p=[\beta_{p1},\beta_{p2},\ldots\ldots\ldots\ldots\beta_{pm}]^t$ is weight vector connecting p^{th} hidden node and output nodes.
And b_p is the threshold of the p^{th} hidden node. With Activation function g(x) and L hidden nodes SLF's can

approximate these N samples with zero error i.e. $\sum_{q=1}^{L} \|o_q-t_q\|=0$

Thus, we can say that there exist w_p, b_p and β_p such that,

$$\sum_{p=1}^{L} \beta_p \,.\, g(w_p \,.\, x_q+b_p)=t_q \quad \text{for } q=1,2,3\ldots\ldots\ldots N$$

We can also show the above equation as

$$H\beta=T \qquad (1)$$

Where $H(w_1,\ldots\ldots\ldots..w_L,b_1,\ldots\ldots\ldots\ldots b_L,x_1,\ldots\ldots\ldots..x_N)$

$$= \begin{bmatrix} g(w_1x_1+b_1) & \cdots & g(w_Lx_1+b_L) \\ \vdots & \cdots & \vdots \\ g(w_1x_N+b_1) & \cdots & g(w_Lx_N+b_L) \end{bmatrix}_{NxL}$$

$$\beta= \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{Lxm} \quad \text{and } T=\begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{Nxm}$$

When the input weights (w_p) and hidden layer biases (b_p) are determined then output weight β_p is calculated as:

$$\beta = H^{+}T$$

Where H^{+} is the Moore-Penrose generalized inverse of matrix H.

WEIGHTED EXTREME LEARNING MACHINE

Weighted extreme learning machine could be termed as extreme learning machine with weight matrix and regularization parameter. Regularization parameter is used to represent trade-off between maximizing marginal distance and minimizing misclassification error cost/value and, weight matrix is inversely proportional to the normalized input data.

Training samples are given in the form of (x_i, t_i) where, $x_i \epsilon R^n$ and t_i is either -1 or +1 for i=1…….N and n is the number of features in each training sample. Define an NxN diagonal weight matrix for each training sample x_i where, weight assigned to minority class is always greater than weight assigned to majority class. Input weight (w_{pj} for p=1….L,j=1….n) between n input nodes and L hidden layer nodes and bias (b_p for p=1…L) at each hidden layer node are randomly generated between 0 and 1. Hidden layer output is $h_p= w_{pj}x_p + b_p$.Output layer node output is $y_i =h_1(x_i)\beta_1 + h_2(x_i)\,\beta_2 +\ldots.+h_L(x_i)\,\beta_L= \mathbf{h}(x_i)\boldsymbol{\beta}$ and output in vector form is $\mathbf{Y = H\beta}$. Target output is $\mathbf{T} = t_1, t_2,\ldots\ldots\ldots\ldots\ldots,t_N$.Since

it is a binary classifier so there is only one output node. Error vector is **ξ = Y-T = H β-T.** Output weight should be such that the marginal distance is maximum for better generalization performance. Architecture of ELM is shown in fig: 1. So, for maximizing marginal distance and minimizing error vector we can model optimization function as

Minimize: $f = \frac{1}{2}\|\beta\|^2 + \frac{1}{2}CW\sum_{i=1}^{N}\|\xi_i\|^2$

Subject to: $h(x_i)\beta = t_i^T - \xi_i^T$, for i=1,2,……N

Solving this optimization function using langrage multiplier and Karush-Kuhn-Tucker theorem [2] we get required output weight as

When N is small: $\beta = H^T\left(\frac{I}{C} + WHH^T\right)^{-1}WT$

When N is large: $\beta = \left(\frac{I}{C} + H^TWH\right)^{-1}H^TWT$

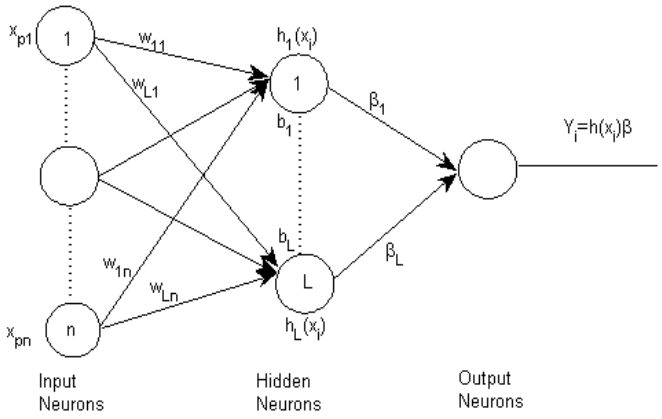

Fig.2. Architecture of Weighted ELM

WEIGHT DETERMINATION IN WEIGHTED ELM

In weighted ELM, two schemes for weighing misclassification are given. Minority misclassification weight is given as inverse of total minority samples and majority misclassification weight is given as inverse of total majority samples.

$$W_{Minority} = \frac{1}{\$(Minority)} \quad and \quad W_{Majority} = \frac{1}{\$(Majority)}$$

Where, $(minority) is the number of samples belongs to minority class and $(majority) is the number of sample belongs to majority class.

In other scheme, minority misclassification weight is given as inverse of number of minority samples multiplied by 1 and majority misclassification weight is given as inverse of majority samples multiplied by 0.618.

$$W_{Minority} = \frac{1}{\$(Minority)} \quad and \quad W_{Majority} = \frac{0.618}{\$(Majority)}$$

Here, minority misclassification weight is always greater than majority misclassification weight which causes proper positioning of classification boundary and hence determines good generalization performance [9] [10].

GENETIC ALGORITHM

GA(η, χ, μ)
//Initialize generation 0:
K: =0;
Pk : =a population of η randomly generated individuals.
//Evaluate Pk :
Compute *fitness* (i) for each i ϵ Pk ;
do
{
//**Create generation** *k+1*
//**1. Copy:**
Select (1- χ) x η members of Pk and insert into Pk+1
//**2.CrossOver:**
Select (χ x η) members of Pk ; pair them up; produce offspring;
//**3.Mutate:**
Select (μ x η) members of Pk+1 ; invert a randomly-selected bit in each;
//**Evaluate** Pk+1 :
Compute *fitness* (i) for each i ϵ Pk ;
//**Increment :**
K:=k+1;
}
While fitness of individuals in Pk is not high enough.
return the fittest individual from Pk;

η is the number of individuals in the population;
χ is the fraction of population replaced by crossover in each iteration;
μ is the mutation rate

BACTERIAL FORAGING ALGORITHM (BFOA)

1. Intialize parameters p, S, Ns ,NC ,NRe,Ned,Ped,C(*i*)=(1,2,3…S),θ^i
2**.** Elimination dispersal-loop: l=l+1
3**.** Reproduction loop: k=k+1
4.Chemotaxis loop: j=j+1
[a] For i=1,2,..S take a chemotactic step as follows for bacterium I as follows
[b] Compute fitness function, J(i,j,k,l).
Let J(i,j,k,l)= J(i,j,k,l)+J_{cc} (θ^i (j,k,l),P(j,k,l)) (i.e. add on the cell to cell attractant-repellant profile to simulate the swarming behavior)
where J_{cc} is defined in (2).
[c] Let J_{last} =J(i,j,k,l) to save this value since we may find a better cost via run.
[d] Tumble: generate a random vector $\Delta(i) \epsilon \Re^p$ with each element Δ_m(i),m=1,2,..p, a random numberon[-1,1].
[e]Move :Let
θ^i (j+1,k,l)= θ^i (j,k,l)+ C(i)$\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$
this result in step size C(i) in the direction of tumble for bacterium i.
[f] Compute (I,j+1,k,l) and let
J(i,j+1,k,l) =J(i,j,k,l)+ J_{cc} (θ^i (j+1,k,l),P(j+1,k,l)).
[g] Swim
i)Let m=0 (Counter for swim length).
ii)While m<N_s (if have not climbed down too long).

- Let m=m+1
- If J(i,j+1,k,l)<J_{last} (if doing better), let J_{last} =J(i,j+1,k,l) and let
  θ^i (j+1,k,l)= θ^i (j,k,l)+ C(i)$\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$
  and use this θ^i (j+1,j,k) to compute the new J(i,j+1,k,l) as we did in [f]

- Else let m=N_s .This is the end of the while statement.

Goto the next bacterium (i+1) if i≠S(i.e. go to [b] to process the next bacterium).
5. If j<N_c ,goto step 4. In this case continue chemotaxis since the life of bacteria is not over.
6.Reproduction:
[a] For all given k and l, and for each i=1,2,..,S ,let
$J^i_{\text{health}} = \sum_{j=1}^{Nc+1} J(i,j,k,l)$
Be the health of the bacterium i(a measure of how many nutrients it got over it's lifetime and how successful it was avoiding noxious substances).Sort bacteria and chemotactic parameters C(i) in order of ascending cost J_{health} (higher cost means lower health).
[b]S_r bacteria with the highest J_{health} values die and the remaining S_r bacteria with the best values split (this process is performed by the copies that are made are placed at the same location as their parent).
7. If k<N_{re} , go to **step 3.** In this case, we have not reached the number of specified reproduction steps , so we start the next generation of the chemotactic loop.
8. Elimination-dispersal: For i=1,2,..,S with probability P_{ed} , eliminate and disperse each bacterium (this keeps the number of bacteria in the population constant).To do this,if a bacterium is eliminated, simply disperse another one to a random location on the optimization domain . If l<N_{ed} ,then go to step 2;otherwise end.
The purpose is to unite both methods (GA & BFO) in the best way possible, and the solution that we proposed is to apply fuzzy logic to manage with the uncertainty in both optimization algorithms.

FGA + FBFO METHOD

The general approach of the proposed FBFO + FGA method can be described as follows:

1. It receives a required function to be optimized.
2. It evaluates the role of both GA and BFO.
3. A main fuzzy system is responsible for receiving values resulting from step 2.
4. The main fuzzy system decides which method to use (GA or BFO).
5. Then, another fuzzy system receives the Error and Derivate of error (DError) as inputs to evaluate if it is necessary to change the parameters in the GA or BFO.
6. There are 3 fuzzy systems. One is for decision making (called main fuzzy), the second one is for changing parameters of the GA (called fuzzyga) and the third fuzzy system is used to change parameters of the BFO (is called fuzzybfo).
7. The main fuzzy system decides in the final step the optimal value for the function introduced in step
8. Repeat the above steps until the termination criterion of the algorithm is met.

PROPOSED WORK

As we have seen that, Weighted ELM is a good approach for classification which overcomes the problem of presence of data with imbalanced class distribution in ELM and also improves the accuracy of ELM but, it is unable to provide optimal solution because, here misclassification cost or weight, which is assigned to binary classes is inversely proportional to the number of samples or we can say that, here, dependency of weight on number of samples exist which restricts search space for finding optimal weights at which good classification accuracy can be achieved.

As we have seen that, no considerable work had been done in this field till date .Therefore, in our proposed weighing scheme, we found weight with the help of FMA + FGA method which, increases search space. Then these weights are assigned directly to t he binary class which eliminates the dependency on number of samples. NGVCK and MWOR kit [3] is taken for generating virus dataset and normal files are taken from windows 7. After applying proposed technique satisfactory results are being achieved.

CONCLUSION

This work proposes WELM_FGA_ FBFO for the detection of computer viruses, which improves the performance of WELM and gives optimal misclassification weights. This paper involves the NGVCK and MWOR kits used for malware generation. These kits are broadly used in research due to their property of generating challenging viruses that is not easy to detect. This study will be helpful for those working in the field of computer virology.

ACKNOWLEDGEMENT

I wish to express my special thanks to all who supported me directly or indirectly in this work.

REFERENCES

[1] Huang G., Zhu Q., Siew C., "Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks. International Joint Conference on Neural Networks", pp. 985–990, 2004.
[2] Fletcher R., "Practical Methods of Optimization: Constrained Optimization. Wiley, New York 2". 2002.
[3] VX Heavens. http://vx.netlux.org/
[4] Bist, Ankur Singh, and Sunita Jalal. "Identification of metamorphic viruses."Advance Computing Conference (IACC), 2014 IEEE International. IEEE, 2014.
[5] Bist, Ankur Singh. "Detection of metamorphic viruses: A survey." Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on. IEEE, 2014.
[6] Bist, Ankur Singh. "Classification and identification of Malicious codes." IJCSE. 2012.
[7] Bist, Ankur Singh. "Fuzzy Logic for Computer Virus Detection." IJESRT, ISSN: 2277-9655.
[8] Bist, Ankur Singh. "Hybrid model for Computer Viruses: an Approach towards Ideal Behavior." International Journal of Computer Applications 45 (2012).
[9] Sharma, Rudranshu, et al. "Genetic Algorithm based Weighted Extreme Learning Machine for binary Imbalance Learning."
[10] Sharma, Rudranshu, and Ankur Singh Bist. "INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY MACHINE LEARNING: A SURVEY."