MISUSE DETECTION
A Neural Network vs. A Genetic Algorithm Approach

Pedro A. Diaz-Gomez and Dean F. Hougen
Robotics, Evolution, Adaptation, and Learning Laboratory (REAL Lab)
School of Computer Science, University of Oklahoma
Norman, OK, USA

Keywords: Misuse detection, genetic algorithms, neural networks, false negative, false positive.

Abstract: Misuse detection can be addressed as an optimization problem, where the problem is to find an array of possible intrusions x that maximizes a function $f(\cdot)$ subject to a constraint r imposed by a user's actions performed on a computer. This position paper presents and compares two ways of finding x, in audit data, by using neural networks and genetic algorithms.

1 INTRODUCTION

The misuse detection problem is the problem of finding intrusions of known types. Knowing the intrusion types in advance, the problem is to find instances of them in audited data.

The misuse detection problem can be formulated as: Given the observation vector $OV \in \mathbb{Z}^+$, and the Attack-Event matrix $AE \in \mathbb{Z}^{mn}$ of known intrusion types, find the best parameter vector $x \in \{0,1\}$ such that $r_i(x) = (AE * x)_i - OV_i \leq 0$, for all $0 \leq i \leq m$, where x_j are independent variables for all $0 \leq j \leq n$ (m is the number of event types to consider and n is the number of intrusions to check). The *best* x is the one that minimizes the length of the residual $r(x)$, i.e., we are facing a linear least squares problem, that can be solved with different methods. However, one can look at the problem as a linear constrained optimization problem, where a Neural Network (NN) can be proposed to solve it.

Mé (1998) addressed the misused detection problem as the problem to find the $x \in \{0,1\}$ vector that maximizes the function $f(x) = w^T \cdot x$, subject to the constraint $r_i(x) = (AE * x)_i - OV_i \leq 0$, for all $0 \leq i \leq m$, where w is a weighting vector that allows for more importance to be assigned to finding some intrusions, $AE \in \mathbb{Z}^+$ is the matrix where columns are known intrusions types and rows are the events necessary for intrusions of those types to be carried out, and $OV \in \mathbb{Z}^+$ is the filtered audit data to be analyzed. The linear problem, where the coefficients of the solution $x \in \mathbb{Z}^+$ are in $\{0,1\}$ can be polynomially reduced to the zero-one integer programming problem that is *NP*-Complete (Mé, 1993). Mé (1998) proposes the use of a Genetic Algorithm (GA) to solve it because of the capability of the GA to work on different subsets of possible solutions, however, some of those subsets could be exclusive (Mé, 1998; Diaz-Gomez and Hougen, 2006; Diaz-Gomez and Hougen, 2005b), making the problem harder to solve.

How the problem is addressed can reveal different methods to solve it. Some methods require more computation time and/or space than others, and some give better quality solutions than others. This positional paper presents two approaches, a NN and a GA, to solve approximately the misuse detection problem and their computational complexities are compared.

2 NEURAL NETWORKS FOR OPTIMIZATION

Neural networks have been widely used to solve optimization problems (Ham and Kostanic, 2001) and, as was addressed in Section 1, the misuse detection problem can be seen as an optimization problem where we want to maximize $f(x) = w^T \cdot x$, subject to $r_i(x) = AE_{i1}x_1 + AE_{i2}x_2 + ... + AE_{in}x_n - OV_i \leq 0$ for $i = 1,2,...,m$, $x_1 \geq 0$, $x_2 \geq 0, ...,$ $x_n \geq 0$, where x_i are

A. Diaz-Gomez P. and F. Hougen D. (2007).
MISUSE DETECTION - A Neural Network vs. A Genetic Algorithm Approach.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - AIDSS*, pages 459-462
DOI: 10.5220/0002410904590462

Table 1: Intrusions x_i found by a neural network.

21	48	69	96	117	144	165	192	213	240	261	288	309	336	357	384	405	432	453	480	501
528	549	576	597	624	645	672	693	720	741	768	789	816	837	864	885	912	933	960	981	-

independent variables and w is the weighting vector.

In order to solve this linear problem with inequality constraints, Ham and Kostanic (2001) propose the use of a NN with the recursive equation of motion

$$x_j(k+1) = \begin{cases} x_j(k) - \mu_j \left\{ w_j + K \sum_{i=1}^{m} r_i(x) AE_{ij} \right\} \\ \qquad \text{if } x_j(k+1) \geq 0, \\ 0 \qquad \text{if } x_j(k+1) < 0 \end{cases} \tag{1}$$

where μ_j is the learning rate, K is a positive parameter, and k is the iteration step.

We set the following parameters: $x_j(0) = 0$ for all j, $\mu_j = \mu_0/(log(1+k))$ with $\mu_0 = 0.005$ (Ham and Kostanic, 2001), $w_j = 1\ \forall j$, $K = 1$, OV—that is used in $r_i(x)$—as in Table 2, AE corresponds to the $m*n$ matrix in which columns are intrusions, $m = 28$, $n = 1,008$, and the NN stops if $\mu_j < 0.00001$ or if the number of iterations is $h = 6,000$.

The net found 41 out of 108 possible intrusions (see Table 1) and had no false positives. Some convergence values for iterations until 600 are shown in Figure 1. The last μ_j was $\mu_{6000} = 0.00057473$.

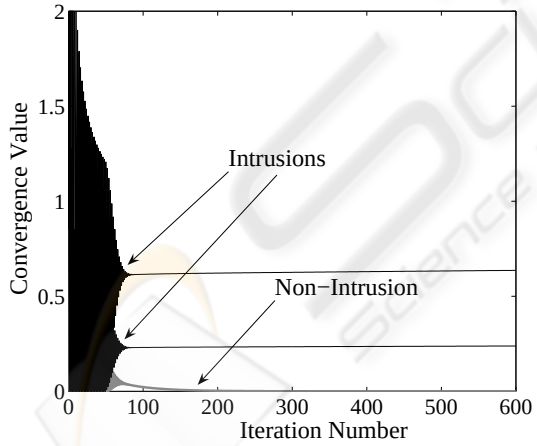

Figure 1: Intrusions type 48 and 21 found by a neural network. At iteration 6,000 the convergence values were $x_{48} = 0.6426$, $x_{21} = 0.241$ and non intrusion $x_{917} = 3.5889e-34$.

Table 2 shows an example of an OV vector and the result of $AE*x-OV$ that shows that the neural network found an x vector that does not violates the constraint. The solution x is such that $x_i \geq 0,\ \forall i$. For example, looking at entries in the AE matrix, intrusions $j = 21 + mod(0,48)$ have values $AE_{26,21+mod(0,48)} = 3$ and there were 21 of those; intrusions $j = 48 + mod(0,48)$ have values $AE_{26,48+mod(0,48)} = 8$ and there were 20 of those; they give a total of activity for entry 26 equal to $3*21*0.241+8*20*0.6426 = 118$ that is exactly OV_{26}—see Figure1 and Table 2.

It should be emphasized that if the initial conditions change, for example if $x_j(0) = 1$ for all j, then the algorithm converges to a second solution. It finds all possible solutions (108), but, in this case the solution violates the constraint and it gives 399 false positives.

3 GENETIC ALGORITHMS FOR OPTIMIZATION

A GA starts with an initial population $P_0 \in \{0,1\}^{sl}$ of possible solutions usually generated randomly, where s is the population size and l is the length of each possible solution. The algorithm iterates g times (generations) through all the individuals in the population looking for fittest individuals x that are artificially selected to mate and give origin possibly to new offspring, according to a fitness function $f(\cdot)$. We follow the guidelines of Mé (1998) but use a different fitness function that tries to avoid false alarms while finding to a maximum number of intrusions, and we use an operator called the *union operator* (Diaz-Gomez and Hougen, 2006). While the GA iterates, the union operator stores all possible solutions (local maximums) and checks if a new one violates the constraint $(AE*x)_i \leq OV_i,\ \forall i$. If a new intrusion is found that does not violate the constraint then it is added to a set S_1; if it violates the constraint, then it is added to a set S_2, such that the entire set of possible intrusions is $S = S_1 \cup S_2$.

The fitness functions uses the partial derivative with respect to x of the Energy Function as in Ham and Kostanic (2001). That is equated to zero in order to obtain a critical x giving for each component j, $\sum_{i=1}^{m} r_i(x) * AE_{ij} = -\frac{w_j}{K}$, which is satisfied by the Equation $\sum_{i=1}^{m} r_i(x) \leq 0$ that is used as a penalty when $r_i(x) > 0$ (Diaz-Gomez and Hougen, 2005a; Diaz-Gomez and Hougen, 2006):

Table 2: Event type, vector of observations OV and constraint comparison using solution x—shown in Table 1—which does not violate the constraint.

Event #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
OV	0	0	0	0	0	0	0	1	0	0	5	0	0	0	0	1	0	25	0	13	0	0	0	2	0	118	315	0
$AE*x-OV$	0	0	0	0	0	0	0	-1	0	0	-5	0	0	0	0	-1	0	-25	0	-13	0	0	0	-2	0	0	-315	0

Table 3: A Subset of intrusions S_2 that violates the constraint with subset S_1 found by a GA.

S_1	21	69	155	157	192	213	261	309	336	384	453	480	501	528	576	683	693	741	768	789	837	864	933	960	1008
S_2	48	96	117	144	165	240	288	357	405	432	549	597	623	644	815	884	911	980	-	-	-	-	-	-	-

$$f(x) = \frac{\sum_{i=1}^{m}(AE*x)_i - \sum_{i=1}^{m} max\{0, r_i(x)\}}{\sum_{i=1}^{m}(AE*x)_i} \quad (2)$$

The parameter settings are: a population size of $s = 1,000$, the fitness function as is Equation 2, one point crossover with probability 60%, a mutation rate of 2.4% per chromosome and a total of $g = 20,000$ generations (Diaz-Gomez and Hougen, 2006). The same AE matrix and OV vector were used as for the case of the NN.

The GA was run 30 times with the same set of parameters just defined, and it found on average 69.3 intrusions ($std = 19.4$). The set of solutions found S was disaggregated by the GA as it was running, i.e., the GA used has the capability to separate the two disjoint sets S_1 and S_2 (Diaz-Gomez and Hougen, 2005b)—see Table 3.

4 NEURAL NETWORK VS. GENETIC ALGORITHM APPROACH

The first topic that we are going to address is how the algorithms presented here, distinguish an intrusion of a non intrusion and the second one is the computational complexity of each algorithm.

4.1 Intrusions vs. Non-Intrusions

For the GA it is clear that a 1 in x_i means a possible intrusion i occurred and a 0 means non-intrusion. For the NN, if x_i converges to a value > 0 then we consider a possible occurrence of an intrusion. However, there is not an exact threshold for the NN to distinguish an intrusion from a non intrusion, as in the GA case. In order to reinforce this fact, we performed tests again, with the same set of parameters defined in section 2 but the vector of observations OV was changed (OV') as in Table 4. The solution of the NN was the same in section 2—see Table 1—but the values of convergence of the intrusions ($x_{48} = 0.0163$, $x_{21} = 0.0061$) and non intrusion changed ($x_{917} = 0$). The NN was looking at each variable (intrusion) x_i independently, as it is expected to do in accordance with the conditions of this paradigm—the most that concerns here is the convergence of $x_i \geq 0$ and that x_i's are independent. To the contrary, the GA looks the possible solution x, with all its components x_i, together, i.e., if there is a possible solution x which violates the constraint, then, x is penalized accordingly—see Equation 2. The set of x_i's are evaluated by the GA at the same time and the algorithm chooses them by looking for the best of those sets.

As NN does not have the capability to look at exclusive sets of intrusions ($S_1 \cap S_2 = \emptyset$), because x_j are independent for $0 \leq j \leq n$, an iterative process that receives as input the output of the NN—i.e., Table 1—and analyzes violations of constraint using $x_j \in \{0,1\}$ can be used as a second phase. This process looks at each row of the AE matrix for columns corresponding to the positions of the NN solution x where x_j is considered a possible intrusion. The output is a subset of Table 1 given in Table 5 as S_1. This time we obtain 12 intrusions type $21 + mod(0,48)$—see Section 2—and 10 intrusions type $48 + mod(0,48)$, which gives us a total of $3*12*1+8*10*1 = 116$ which clearly does not violate the constraint (i.e. $116 \leq OV_{26} = 118$). More than this 22, will begin to violate the constraint—see S_2 in Table 5.

4.2 Computational Complexity

The NN needs to calculate the constraint, i.e., $AE*x-OV$ which has a cost of $m*n$, it adjusts x and, as the algorithm iterates h times, it gives an estimated computational complexity of $O(mnh)$. The GA needs to calculate the constraint for each individual in the population that has a cost of $m*n$ per individual, i.e., with s individuals it gives $m*n*s$ per generation, and as the algorithm iterates g generations, it gives a total computational complexity of $O(mnsg)$ (Diaz-Gomez and Hougen, 2007). So the GA computational com-

Table 4: Vector of observations OV'. Same Solution x—shown in Table 1—which does not violate the constraint.

OV'	0	0	0	0	0	0	1	40	0	0	5	0	0	0	0	1	0	25	0	13	0	0	0	2	0	3	30	0
$AE*x-OV'$	0	0	0	0	0	0	-6	-40	0	0	-5	0	0	0	0	-1	0	-25	0	-13	0	0	0	-2	0	0	-30	0

Table 5: Second Phase. A Subset of intrusions S_2 that violates constraint with subset S_1 found by iterative process.

S_1	21	48	69	96	117	144	165	192	213	240	261	288	309	336	357	384	405	432	453	480	501	549
S_2	528	576	597	624	645	672	693	720	741	768	789	816	837	864	885	912	933	960	981	-	-	-

plexity is higher by $O(sg/h)$.

The space complexity for the NN can be considered as $O(nm)$ because it needs to store the AE matrix, and the OV and x vectors. The GA, besides previous structures, needs to store the population that is of order $O(sl)$. So the GA space complexity is higher in $O(sl)$ than the NN space complexity.

5 CONCLUSIONS AND FUTURE WORK

Two paradigms were tested with the misuse detection problem in audit trail files. As some intrusions share the same types of events, the possible solution x is such that some x_i are dependent, which makes the genetic algorithm paradigm more suited for solving this problem. However, the quality of the solution obtained with the GA has a higher computational complexity cost of $O(sg/h)$—population size by the ratio of number of generations over the NN iterations—and space complexity cost of $O(sl)$—population size by length of x—with respect to the NN.

The GA has the advantage of discriminating an intrusion from a non-intrusion as the solution of the problem is encoded as 1 (intrusion) and 0 (non-intrusion). As the range of values of x_i for the NN are such that $x_i \geq 0$ the values of intrusions are input dependent—depending on the observed vector OV. However, at least for this test set, non-intrusions are variables x_i that converge to 0 or to values ≈ 0 when in the initial conditions x is zero.

For the test set defined in this paper, there were no false positives, except if we consider the NN without the second phase (see Section 4) or if the initial conditions change—see Section 2. For the false negative side, if we look at the two sets S_1 and S_2—see Section 3, the GA has in average (over 30 runs) of 39.14% false negatives, and the NN has 60.95%. However, the set S_2 can have exclusive intrusions, so the process can continue until we get a set of mutually exclusive subsets whose union is S (Diaz-Gomez and Hougen, 2007).

In order to improve the false negative ratio of the GA, it is possible that by increasing the population size ($s > 1{,}000$) the ratio is going to decrease; however, it is possible that the number of generations g should be considered too, independently or in conjunction with the population size. For the case of the NN, it is a more challenging problem to try to diminish the false negative ratio. After the convergence of all x_i's there is no improvement in the solution x, if the number of iterations h is higher.

REFERENCES

Diaz-Gomez, P. A. and Hougen, D. F. (2005a). Analysis and mathematical justification of a fitness function used in an intrusion detection system. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1591–1592.

Diaz-Gomez, P. A. and Hougen, D. F. (2005b). Improved off-line intrusion detection using a genetic algorithm. In *Proceedings of the 7th International Conference on Enterprise Information Systems*, pages 66–73.

Diaz-Gomez, P. A. and Hougen, D. F. (2006). A genetic algorithm approach for doing misuse detection in audit trail files. In *Proceedings of the CIC-2006 International Conference on Computing*, pages 329–335.

Diaz-Gomez, P. A. and Hougen, D. F. (2007). Misuse detection: An iterative process vs. a genetic algorithm approach. In *Proceedings of the 9th International Conference on Enterprise Information Systems*.

Ham, F. M. and Kostanic, I. (2001). *Principles of Neurocomputing for Science & Engineering*. Mc Graw Hill.

Mé, L. (1993). Security audit trail analysis using genetic algorithms. In *Proceedings of the 12th. International Conference on Computer Safety, Reliability, and Security*, pages 329–340.

Mé, L. (1998). GASSATA, a genetic algorithm as an alternative tool for security audit trail analysis. In *Proceedings of the First International Workshop on the Recent Advances in Intrusion Detection*.