Comparison of a Supervised Trained Neural Network Classifier and a Supervised Trained Aggregation Function Classifier

Alexandre Croix, Thibault Debatty, Wim Mees

Royal Military Academy
Brussels, Belgium
Email: a.croix@cylab.be, t.debatty@cylab.be, w.mees@cylab.be

Abstract—**In this paper, we compare the efficiency of two binary classifiers. The first one uses the Weighted Ordered Weighted Averaging (WOWA) aggregation function whose coefficients are learned thanks to a genetic algorithm. The second is based on an artificial neural network trained by a backpropagation algorithm. They are trained to be used in a multi-criteria decision system. These kind of multi-criteria system are more and more common in the cyber-defence field. In this work, we compare the performance of these two classifiers by using two criteria: Area Under the Curve of a Receiver Operating Characteristics (ROC) curve and the Area Under the Curve of a Precision-Recall (P-R) curve. This second criterion is more adapted for imbalanced dataset what is often the case in the cyber-security field. We perform a complete parameter study of these classifiers to optimize their performance. The dataset used for this work is a pool of Hypertext Preprocessor (PHP) files analyzed by a multi-agent PHP webshell detector. We obtain different good results, especially for neural networks and highlights the advantage of the genetic algorithm method that allows a physical interpretation of the result.**

Keywords–Machine learning; neural network; aggregation functions; webshell.

I. Introduction

In the machine learning field, these last years, one method appears to be better than other ones: *neural network*, and particularly *deep learning*. The principle of this kind of algorithm is known for a long time, but the usage increased these last years for two main reasons: new neural network structures were discovered, and the hardware is now powerful enough to produce good results in an acceptable time.

Sometimes, without a good analysis, a neural network is used to solve a problem. But in some cases, it is not the best choice for autonomous learning. It is sometimes interesting to compare the performance of different learning algorithms to solve a problem.

In this paper, we focus on a specific task: the training of two classifiers whose objective is to distinguish if a PHP file, previously analyzed, is a webshell or a harmless PHP file. These two classifiers are: (i) an aggregation function in which the parameters are learned by a genetic algorithm and (ii) a neural network trained by the backpropagation algorithm. These classifiers are chosen instead of others (e.g., decision tree, Support Vector Machines (SVM)...) to compare the performance of a classifier structure very used in practice and an classifier using an aggregation function that is not widely known.

In order to be classified, the PHP files were analyzed by a PHP multi-agent detector composed of 5 different modules. Each agent produces a score between 0 and 1 that is used as inputs for the two classifiers.

We performed a parametric study on both of the classifiers to determine the set of parameters that produces the best result. The dataset used for this work contains 23,415 PHP files where 1,833 [1] are actual PHP webshells.

Rest of the paper is arranged as follow: Section II explains which aggregation function is used for the classification and describes the structure of the genetic algorithm that learns parameters. Section III describes the structure of a neural network, and more specifically the neural network uses in this work. In Section IV, we present our comparison methodology and the results obtained. We conclude and we talk about some way to continue this work in Section V.

II. WOWA aggregation function and genetic algorithm structure

A. WOWA operator

The WOWA operator, WOWA for Weighted Ordered Weighted Averaging, is an aggregation function introduced by Viçen Torra in 1996[2]. This operator generalizes the Weighted Mean (WM) and the Ordered Weighted Averaging (OWA) and allows to merge a set of numerical data in a single result. To aggregate numerical data, WOWA uses two weighting vectors: one for the weighted mean (w) and the other for the OWA operator (p). The weighted mean, weights data in agreement with their sources and OWA gives importance to the data according to their scores.

WOWA combines the advantages of Weighted Mean and OWA operator. In the other hand, WOWA is more complex because it requires two parameters for each data source.

$$WOWA = f(a_1, a_2, ...a_n, w_1, w_2, ...w_n, p_1, p_2, ...p_n) \quad (1)$$

where

- a_i are data sources
- w_i are WM weights
- p_i are OWA weights

This operator allows good accurate results and can be tuned with more parameters than usual aggregation functions. Despite this, this operator is only rarely used in practice.

Learning aggregation operator weights from training dataset is an optimization problem[3]. The optimization algorithm minimizes (or maximizes) a cost function to try to find the global minimum (or maximum) of the solution surface. According to the literature, the algorithm selected to optimize the different weights of the aggregation function is a genetic algorithm[4].

B. Genetic Algorithm structure

A Genetic Algorithm is an evolutive process that maintains a population of chromosomes (potential solutions). Each chromosome is composed of several characteristics called ”genes”. In this work, a ”gene” is a single weight and a ”chromosome” is an element composed of two weight vectors (w and p). The weight vectors contain several ”genes” whose the sum is equal to 1.

The algorithm has five main steps represented in Figure 1.

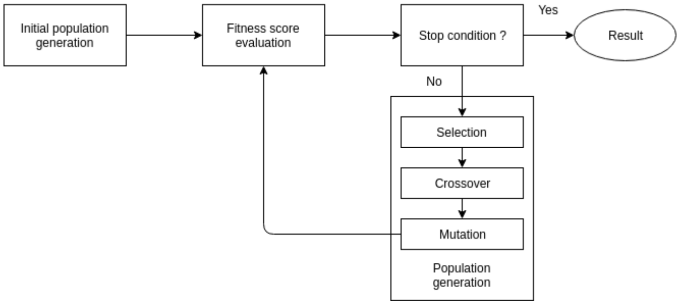

Figure 1. Structure of a Genetic Algorithm

- **Initial population generation**: a set of potential solutions is randomly generated. All ”genes” (weights) are random values between 0 and 1. Then, the weights vectors are normalized.
- **Fitness score evaluation**: all population elements are evaluated by a fitness function.
- **Selection**: according to their fitness score, some elements are selected to be used in the next generation $t + 1$.
- **Crossover**: the selected elements from the previous generation are combined two-by-two to generate new *chromosomes* for the current generation. These new elements keep some characteristics from their parents.
- **Mutation**: each element in the new population has a probability to be mutated. Concretely, a random gene is selected and replaced by another random value. The mutation is very important to avoid converging too fast to a local minimum. The mutation allows to ”jump” to another location in the space of solutions and can discover better results.

This process is repeated until it reaches a termination condition. That can be a sufficient accuracy, a slow convergence since some generations or a fixed number of generation. A complete description of these steps is available in [5].

III. Neural network structure and backpropagation algorithm

An artificial neural network is a computing system inspired by the biological neural networks that constitute animal brains. An Artificial Neural Network is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron receives a signal, processes it and can transmit the result to other neurons connected to it. Figure 2 represents the structure of an Artificial Neural Network. It contains three neurons as inputs, five in the hidden layer and two for the output.

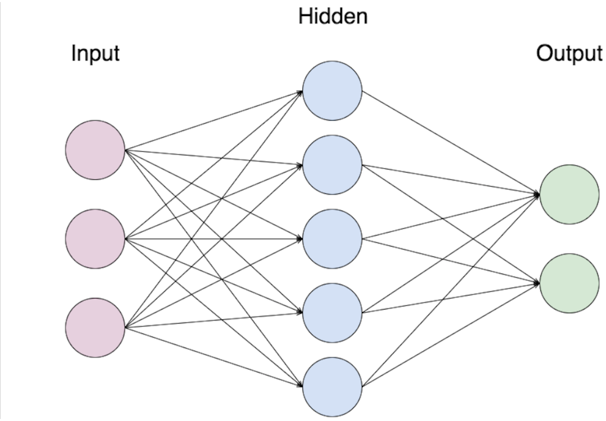

Figure 2. Structure of an Artificial Neural Network

Except for the input layer, a neuron receives several signals (usually real numbers) from the previous layer. These signals are weighted by coefficients and added. Then, the neuron produces an output signal following an activation function that it transmits to the next layer. Figure 3 represents the functioning of an artificial neuron.

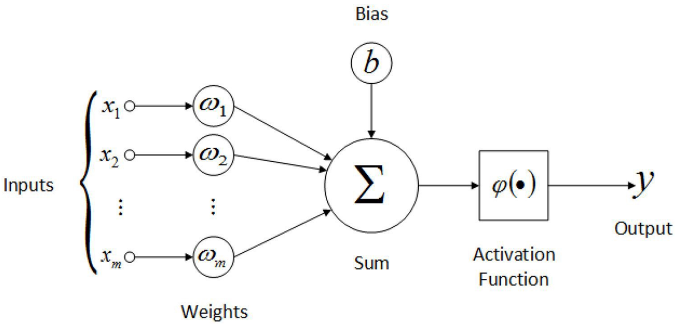

Figure 3. Representation of the functioning of an artificial neuron

In a mathematical point of view, a neuron works following:

$$y = \phi(\sum(x_1\omega_1 + x_2\omega_2 + ... + x_n\omega_n + b)) \tag{2}$$

A *deep* neural network is an artificial neural network with several hidden layers and is very efficient for image recognition or voice recognition there are some big datasets with a lot of features as input (pixels, etc.).

In our problem, it is very difficult to find real PHP webshells and we have only five inputs for each analyzed file. A classical neural network is more suitable for this kind of classification. Our neural network is made of three layers: a five neurons input layer (for the five agents in the webshell detector), a hidden layer and a two neurons output layer (webshell or harmless file).

The job of the algorithm is to find a set of internal parameters (edge's weight and bias) that perform well against some performance measure. It is the cost function. The process is iterative, the convergence occurs over multiple discrete steps that improved internal parameters.

Each step involves using the model with the current set of internal parameters to make predictions on some samples, comparing the predictions to the real expected outcomes, calculating the error, and using the error to update the internal model parameters. This update procedure is the backpropagation (backward propagation) algorithm.

Backpropagation aims to minimize the cost function by modifying the network's weights. The level of adjustment is determined by the gradients of the cost function with respect to those parameters[6]. We do not describe the mathematical principle of the backpropagation algorithm, it is relatively complex and it is not the purpose of this work.

IV. Parametric Study and Evaluation

A parametric study has been performed on the classifiers in order to optimize their performances before comparison. The training dataset used for this work is composed of 23,415 PHP files and contains 1,833 PHP webshells [1].

To evaluate the performance of our classifiers, we use the $k-fold$ cross-validation[7] method (k equals to 10 is known to produce good results). It consists of separating the dataset in k folds, performing the training part on $k-1$ folds and testing on the last fold. This operation is repeated k times by changing the fold used for the evaluation. All these k intermediate results are meant to obtain a general result.

The number of PHP webshell in the dataset is small. With a random fold generation, there is a high probability to obtain very different repartitions. To avoid this issue, each fold is generated with a fixed number of webshells.

The number of webshells in a fold is quite smaller than the number of regular PHP files. To increase the penalty of not detecting a webshell, we artificially increase the number of webshells in the learning dataset. Concretely, we duplicate several times each score related to a webshell.

A. *Performance evaluation methodology*

To evaluate the efficiency of a binary classifier, we used two measurements: (i) the AUC of a ROC curve and (ii) the AUC of P-R curve.

1) Roc curve: The ROC curve is a classical tool to evaluate the performance of a binary classifier[8]. The ROC is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. It is created by plotting the true positive rate (true detection) against the false positive rate (false alarm) as shown in Figure 4

The Area Under the Curve gives a score that allows us to easily evaluate and compare the performance of several classifiers. Closer to 1 the AUC is, better is the classifier.

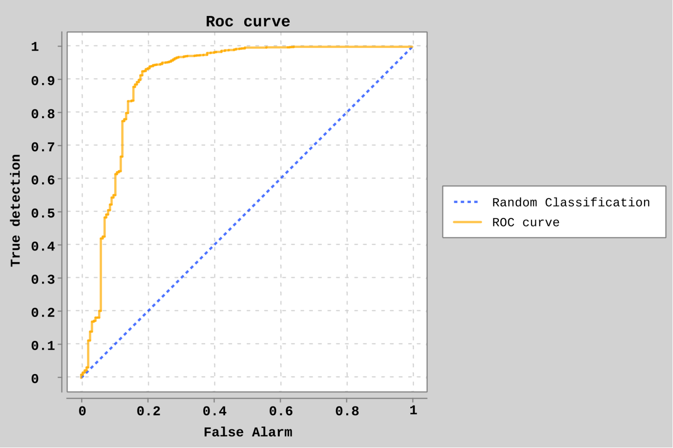

Figure 4. Example of a ROC curve

The dotted blue line represents the behaviour of a random classifier. If the ROC curve is under the curve, that means it is less efficient than a random classification.

2) Precision-Recall curve: A Precision-Recall curve (P-R curve) is, as the ROC curve, a graphical tool to evaluate the efficiency of a binary classifier. It is created by plotting the Recall (X-axis) against the Precision (Y-axis) as shown in Figure 5. Concretely, the precision and the recall are computed for several threshold values. This tool is more informative than the ROC curve for imbalanced dataset [9][10]. Indeed, the P-R curve focuses on the minority class, whereas the ROC curve covers both classes.

Like for the ROC curve, it is very difficult to compare visually different graphs, the AUC solves this issue and allows us an easy comparison. In our dataset, the ratio webshell-harmless file is smaller than 0.1. The evaluation of efficiency by using a P-R curve is perfectly adapted in this situation.

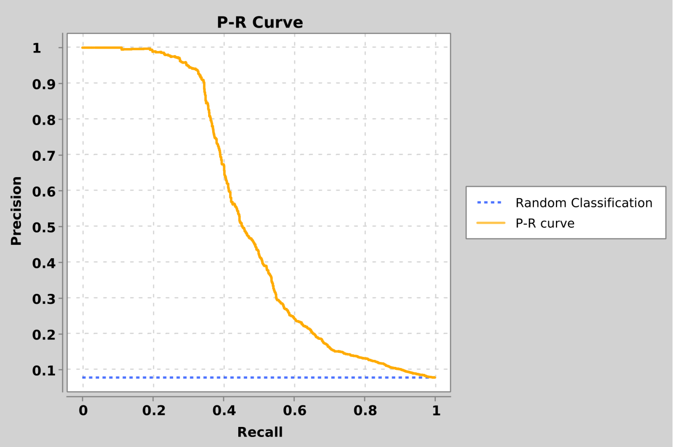

Figure 5. Example of a Precision-Recall curve

The dotted blue line on the graph represents the behaviour of a random classification. A P-R curve under this line is less efficient than a random classifier. The value of this line is $y = \frac{Webshell}{normal\_files}$

B. *Genetic Algorithm parametric study*

A genetic algorithm has several parameters that can be tuned to optimize results. In our parametric study we tested the following parameters:

- **Population size**: number of elements in each generation of the population. We varied this parameter between 40 and 200 by step of 10.
- **Crossover rate**: percentage of the population kept to set up the next generation. We varied this parameter between 5 and 95 by step of 5.
- **Mutation rate**: probability that a gene is randomly mutated. We varied this parameter between 5 and 95 by step of 5.
- **Generation number**: number of generations before stopping the algorithm. The way the algorithm is built, the best element of a generation is equal or better than the best element of the previous generation. Increasing the number of generation can only improve the performance of the classifier, but the improvement is not really significant for generation higher than 200. We fixed the generation number parameter to 200.
- **Fitness score evaluation**: method used to determine the fitness score of a chromosome. Two fitness criteria are implemented in our algorithm: (i) the distance criterion and (ii) the AUC criteria.

To determine the best parameters combination, we tested each parameter independently of the others. It is an approximation. Indeed, it is highly improbable the parameters are completely non-correlated. However, the parametric study shows the importance of the different parameters. We tested all the values of *population size*, *crossover rate* and *mutation rate* with the distance fitness score evaluation and the AUC fitness score evaluation. For each parameter, we perform a 10-folds cross-validation and kept the parameter values that produce the best AUC ROC result and the best P-R AUC value.

a) Distance fitness score: For each chromosome in the population, the WOWA function is computed on all examples of the dataset. The obtained results are substracted to the results given in the training dataset. All these differences are added to obtain a total distance that is the fitness score of the chromosome.

b) AUC fitness score: For each population element, the WOWA function is also computed on all examples of the dataset. Then, these results are used to obtain the ROC. The AUC of this curve is the fitness score of the chromosome.

1) Parametric study results: The first thing we note is the AUC fitness score produces better scores than the other fitness score for all values of all parameters. Intuitively it is logical. This criterion tries to optimize the AUC of a ROC curve that is also a performance criterion of the classifier.

Table I shows, for each parameter, the value that produces the best ROC AUC and the result associated with it. All these parameter values will be used in combination as a new model in the next section.

Table II shows, for each parameter, the value that produces the best P-R AUC and the result associated with it. All these parameter values will be used in combination as a new model in the next section.

TABLE I. Genetic algorithm parameter values that produce the best results in the parametric study for the ROC criterion

Parameter name	Parameter value	ROC result
Population size	75	0.88114
Crossover rate	40	0.88117
Mutation rate	20	0.88125
Fitness function	AUC	0.88125

TABLE II. Genetic algorithm parameter values that produce the best results in the parametric study for the P-R criterion

Parameter name	Parameter value	P-R result
Population size	130	0.73502
Crossover rate	30	0.73612
Mutation rate	5	0.73401
Fitness function	AUC	0.73612

C. *Neural Network parametric study*

As for the genetic algorithm, neural networks can be tuned by modifying some parameters. It is a very difficult point to design correctly a network with the right parameters. The set of basic network parameters are called *Hyperparameters*. In this work, we tuned the most common ones:

- **Activation function**: function used by neurons for the activation. The activation function manages the value of the neuron output. The signals of the previous layer are weighted by the internal parameters, added together and this result is used as input for the activation function. In this work, we used the most common activation functions: tanh, ReLu, sigmoid. Figure 6 represents these three functions.
- **Neurons number**: number of neurons in the hidden layer. Intuitively, we can guess that more neurons are in the hidden layer, more accurate will be the classification. In practice, it is more complicated. Too many neurons can produce an overfitting phenomenon. We varied the number of neurons between 5 and 50 for each activation function.
- **Learning rate**: parameter that controls how much the model change in response to the estimated error each time the model weights are updated. Choosing the learning rate is challenging as a value too small may result in a long training process that could get stuck, whereas a value too large may result in learning a sub-optimal set of weights too fast or an unstable training process. We varied this parameter between 0.1 and 0.95 by step of 0.05, between 0.01 and 0.009 by step of 0.01 and between 0.001 and 0.009 by step of 0.01 for each of the three activation functions.
- **Batch size**: size of the batch. During the training, the dataset elements passed through the network one after one. The batch size is the number of elements that pass through the network before tuning the weights of the network. We varied the batch size between 1000 and 2000 by step of 200 for each of the three activation functions. It was not possible to use a batch size smaller than 1000 in Google Colaboratory.
- **Epoch number**: number of time the entire dataset is passed through the network. A high value of this parameter usually increases the efficiency of the learning but also the time needed. We varied this parameter

between 100 and 350 by step of 50. As for the batch size, it was not possible to use epochs number bigger than 350 in Google Colaboratory

As explained in Section I, the learning of an Artificial Neural Network is clearly faster with GPU. To train our model in a reasonable time, we used the Jupyter Notebook on Google Colaboratory platform. Unfortunately, this platform has some time restriction and does not provide a sufficient infrastructure and to test the whole range of parameters we wanted to test.

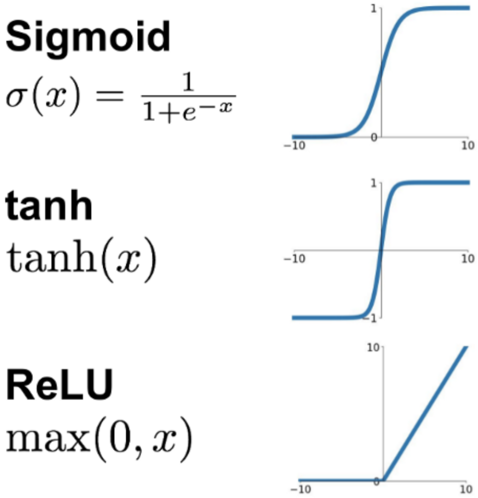

Figure 6. Equations and curves of activation functions

1) Parameters study results: As for the Genetic Algorithm classifier and the fitness function, the activation function in the neural network classifier that produces the best results for all values of all parameters is the ReLu function. Table III and IV show the values of the parameters that produce the best results for the two evaluation criteria.

TABLE III. Neural network parameter values that produce the best results in the parametric study for the ROC criterion

Parameter name	Parameter value	ROC result
Neurons number	38	0.93761
Learning rate	0.04	0.93979
Batch size	2000	0.92746
Epochs number	350	0.94989
Activation function	ReLu	0.94989

Table III shows, for each parameter, the value that produces the best ROC AUC and the result associated with it. All these neural network parameter values will be used in combination as a new model in the next section.

TABLE IV. Neural network parameter values that produce the best results in the parametric study for the P-R criterion

Parameter name	Parameter value	P-R result
Neurons number	38	0.80854
Learning rate	0.05	0.81336
Batch size	2000	0.78883
Epochs number	350	0.83951
Activation function	ReLu	0.83951

Table IV shows, for each parameter, the value that produces the best P-R AUC and the result associated with it. All these neural network parameter values will be used in combination as a new model in the next section.

D. Comparison of the classifiers

For each classifier (genetic algorithm or neural network), we selected the set of parameters that produces the best ROC AUC and the best P-R AUC. With these four models, we ran 10 times a 10-folds cross-validation and we meant the results to minimize the variance.

Table V shows the results for all the 4 classifiers with the corespondent evaluation criterion.

TABLE V. Results of a 10-10-cross validation of the classifiers

Classifier	ROC	P-R
Genetic Algorithm	0.900598	0.745871
Neural Network	0.946812	0.812567

The parameter values of these four models are given in the Tables I, II, III and IV in Section IV.

Figures 7, 8 and 9 represent some example curves obtained during the 10-folds cross-validation.

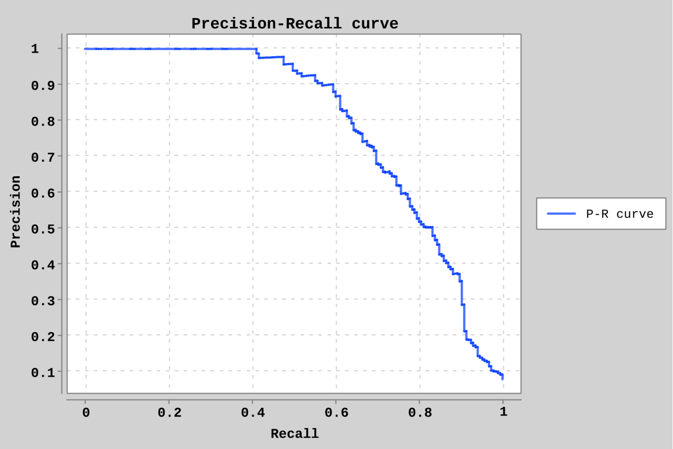

Figure 7. Representation of a Precision-Recall curve for a Neural Network. Parameters: Neurons Number: 46 - Learning Rate: 0.01 - Batch Size: 2000 - Epochs Number: 100 - Activation Function: ReLu

V. Conclusions and Future Works

In this work, we show that a classification based on an artificial neural network trained by the backpropagation algorithm gives better results than a classification using an aggregation function trained by a genetic algorithm.

The neural network is more efficient on the two performance criteria used in this work: ROC AUC and P-R AUC. Moreover, because of the restrictions applied to Google Colaboratory, we were not able to test a significant amount of values for the *batch size* and *epochs number* parameters. It is logical to assume that the results would be even better with a much larger *number of epochs*.

However, it is important to note that the training of a neural network is very slow on CPUs. To obtain results in a reasonable time, it is essential to equip yourself with GPU which is expensive.

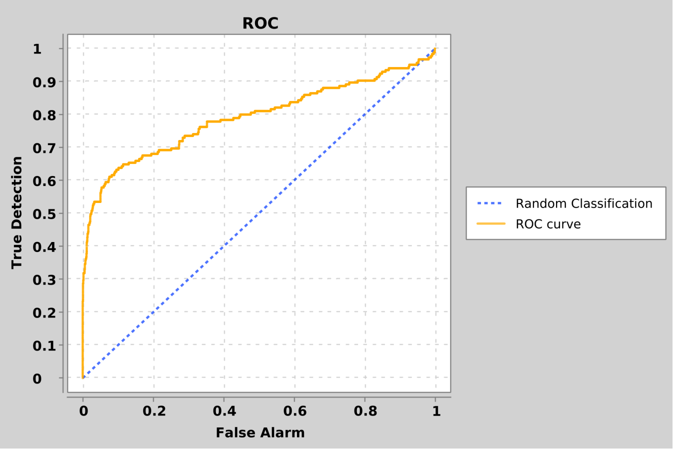

Figure 8. Representation of a Roc curve for a Genetic Algorithm. Parameters: Population Size: 125 - Crossover Rate: 60 - Mutation Rate: 25 - Fitness score: AUC

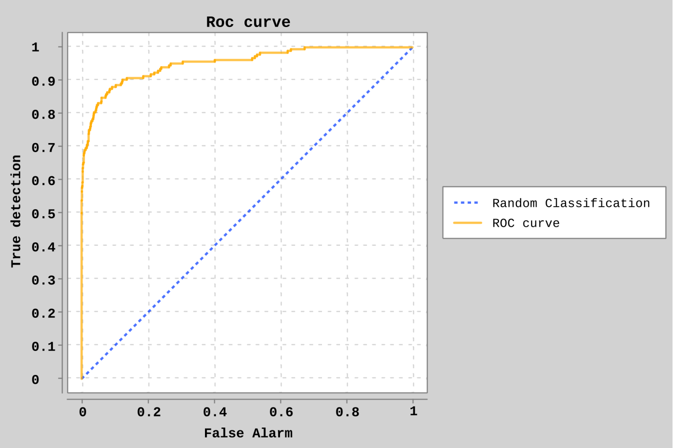

Figure 9. Representation of a Roc curve for a Neural Network. Parameters: Neuron number : 40 - Learning Rate : 0.06 - Batch Size : 2000 - Epoch number : 350 - Activation function : ReLu

We noted that for the genetic algorithm, the *AUC* criteria produces always better results than the *distance* criteria. On the neural network, we noted the activation function that gives the best results, is always the *ReLu* function.

During our work, we noted that the parameters of the genetic algorithm have only a very small influence on the result. It may depend on the dataset used, however, a parametric study seems to be much less important for the genetic algorithm than to train a neural network.

Another important point is about the interpretation of the internal parameters after the training. On an artificial neural network, it is difficult to interpret the meaning of the network parameters (weight, bias, ...). A neural networks is often used as a ”black-box”. The trained aggregation function, on the other hand, gives interesting information about the modules that are aggregated. It highlights the modules with high importance and, mostly, the less important. This could point out that a module is inefficient and improve it in priority.

For example, a typical w weights vector obtained by the genetic algorithm with the dataset used in this work is $w = [0.4407, 0.532, 0.0204, 0.0027, 0.0042]$. We note easily that the most important agent is the second one: the webshell signature analyzer. On the other side, the least effective agent is the obfuscation detector in the fourth position.

This work could be improved in several points. In the first time, it should be interesting to perform a bigger parametric study on the neural network classifier. We think it is possible to increase the efficiency of the results obtained with a better infrastructure to train the model.

It seems interesting to determine the correlation between the different parameters of the algorithms. Indeed, we performed our parametric study by tuning independently each parameters. If we knew the correlations between the different parameters, we would probably be able to determine a better parameters combination.

The algorithms designed for these works are made to be easily used in another project. It could be interesting to test these classifiers on different types of data to determine if the parameters are independent of the dataset or not.

References

[1] Z.-H. Lv, H.-B. Yan, and R. Mei, “Automatic and accurate detection of webshell based on convolutional neural network,” in Cyber Security, X. Yun, W. Wen, B. Lang, H. Yan, L. Ding, J. Li, and Y. Zhou, Eds. Singapore: Springer Singapore, 2019, pp. 73–85.

[2] V. Torra, “Weighted owa operators for synthesis of information,” vol. 2, 10 1996, pp. 966 – 971 vol.2.

[3] V. Torra, “On the learning of weights in some aggregation operators: the weigthed mean and owa operators,” 1999, URL: http://digital.csic.es/handle/10261/2244 [accessed: 2020-08-19].

[4] D. Nettleton and V. Torra, “A comparison of active set method and genetic algorithm approaches for learning weighting vectors in some aggregation operators,” International Journal of Intelligent Systems, vol. 16, 09 2001, pp. 1069–1083.

[5] A. Croix, T. Debatty, and W. Mees, “Training a multi-criteria decision system and application to the detection of php webshells,” in 2019 International Conference on Military Communications and Information Systems (ICMCIS), May 2019, pp. 1–8.

[6] M. Nielsen, “Neural networks and deep learning - chap 2 : How the backpropagation algorithm works,” 12 2019, URL: http://neuralnetworksanddeeplearning.com/chap2.html [accessed: 2020-08-19].

[7] I. Witten and E. Frank, Data Mining Practical Machine Learning Tools And Techniques, 01 2005, vol. 11.

[8] T. Fawcett, “Roc graphs: Notes and practical considerations for researchers,” Machine Learning, vol. 31, 01 2004, pp. 1–38.

[9] T. Saito and M. Rehmsmeier, “The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets,” PLOS ONE, vol. 10, no. 3, 03 2015, pp. 1–21, URL: https://doi.org/10.1371/journal.pone.0118432 [accessed: 2020-08-19].

[10] J. Keilwagen, I. Grosse, and J. Grau, “Area under precision-recall curves for weighted and unweighted data,” PLOS ONE, vol. 9, no. 3, 03 2014, pp. 1–13, URL: https://doi.org/10.1371/journal.pone.0092209 [accessed: 2020-08-19].