A Dynamic Self-Structuring Neural Network

Fadi Thabtah
Nelson Marlborough Institute of Technology
NewZeland.
Fadi.Fayez@nmit.ac.nz

Rami M. Mohammad
School of Computing and Engineering
University of Huddersfield
Huddersfield,UK
rami.mohammad@hud.ac.uk

Lee McCluskey
School of Computing and Engineering
University of Huddersfield
Huddersfield, UK
t.l.mccluskey@hud.ac.uk

Abstract—Creating a neural network based classification model is commonly accomplished using the trial and error technique. However, the trial and error structuring method have several difficulties such as time and availability of experts. In this article, an algorithm that simplifies structuring neural network classification models has been proposed. The algorithm aims at creating a large enough structure to learn models from the training dataset that can be generalised well on the testing dataset. Our algorithm dynamically tunes the structure parameters during the training phase aiming to derive accurate non-overfitting classifiers. The proposed algorithm has been applied to phishing websites classification problem and it shows competitive results with respect to various evaluation measures such as Harmonic Mean (F1-score), precision, accuracy, etc.

Keywords- Classification, Neural Network, Phishing, constructive, pruning.

I. INTRODUCTION

Artificial Neural Network (ANN) proved its merits in several classification domains [1]. Nevertheless, one downside of creating any neural network (NN) based classification model is that it is difficult to interpret its results, and it is considered as a black-box. This characteristic has successfully applied in different security domains such as the work done in [2], [3], and [4].

Yet, we believe that this particular obstacle will add a positive edge to the domains where the results are more important than the understanding of how the model works, such as predicting phishing websites. Although selecting a suitable number of hidden neurons and determining the value of some parameters, i.e. learning rate, momentum value and epoch size showed to be crucial when creating a NN model [5] there is no clear procedure for determining such parameters, and most model designers rely on trial and error. However, the trial and error technique involves a painstaking process for many real world problems. In addition, the trial and error technique is a time-consuming process. A poorly structured NN model may cause the model to underfit the training dataset [6]. On the other hand, exaggeration in restructuring the system to suit every single item in the training dataset may cause the system to be overfitted [7]. One possible solution to avoid the overfitting problem is by restructuring the NN model in terms of tuning some parameters, adding new neurons to the hidden layer or sometimes adding a new layer to the network. A NN with a small number of hidden neurons may not have a satisfactory representational power to model the complexity and diversity inherent in the data. On the other hand, networks with too many hidden neurons could overfit the data. However, at a certain stage the model can no more be improved, therefore, the structuring process should be terminated. Hence, an acceptable error rate should be specified when creating any NN model, which itself is considered a problem since it is difficult to determine the acceptable error rate a priori [6]. For instance, the model designer may set the acceptable error rate to a value that is unreachable which causes the model to stick in local minima [1] or sometimes the model designer may set the acceptable error rate to a value that can further be improved.

Overall, we believe that automating the structuring process of NN models is a timely issue and it might displace some of the burden from the system designer. However, automating structuring NN models does not merely means adding new neuron(s) to the hidden layer because the more hidden neurons does not necessarily mean that the accuracy will improve [8]. Hence, it is important to improve the NN performance by updating several parameters such as the learning rate before adding a new neuron to the hidden layer. Sadly, setting the learning rate value is also a trial and error process. Although several studies have been made to come up with the best NN structure, the optimal learning rate value is still concealed.

II. NEURAL NETWORK STRUCTURING APPROACHES

The traditional trial and error method is commonly used in structuring ANNs. However, several attempts have been devoted to automate NN structuring process such as constructive and pruning [9].

Constructive Algorithms

This approach starts with a simple NN structure, i.e. one hidden layered NN with a single neuron in the hidden layer [9] and recursively new parameters i.e. hidden layers, hidden neurons, and connections are added to the initial structure until reaching a satisfactory result. After each addition, the entire network or only the recently added parameter is retrained. Constructive approach is relatively easy for inexperienced users because they are normally asked to specify few initial parameters, for example the number of neurons in the input layer and epoch size and then new

parameters are added to the network. This approach is computationally efficient because it searches for small structures first [9]. However, constructive approach has some hurdles that should carefully be addressed. For example, the user has to decide when to add a new parameter, when to stop the addition process, and when to stop training and produces the network [10].

Pruning Algorithms

The pruning approach starts with an oversized structure, i.e. a multi hidden layered NN with a large number of hidden neurons in each hidden layer. Later on, some parameters, i.e. connections, hidden neurons, and hidden layers are removed from the network. After each training process, the user removes some parameters from the network and the new structure is retrained so that the remaining parameters can compensate the functions played by the removed parameters [9]. If the network performance improved, the user removes more parameters and retrains the network again. However, if the network performance does not improved, the user restores what have been deleted and tries to remove other parameters. This process is repeated recursively until achieving the final network. Normally, only one parameter is removed in each pruning phase [9]. In general, this approach is a time consuming process. In addition, the user does not know a priori how big the initial NN structure should be for a specific problem.

III. An Improved Self-Structuring Neural Network Algorithm

The pseudocode of proposed improved Self-Structuring Neural Network (iSSNN) algorithm is shown in Figure 1 and it works as follows:

1- Parameter Settings Phase

This phase involves several activities as follows:

A. The number of neurons in the input layer is set to the number of the features offered in the training dataset *(line 1 in Figure 1)*.

B. The number of neurons in the output layer is set to one because iSSNN aims to create binary classification models *(line 2)*. In other words, we will create a single neuron that might hold two possible values.

C. The number of neurons in the hidden layer is to be determined by the algorithm. Constructive algorithms normally start with one hidden neuron [9]. At the beginning, the algorithm creates the simplest NN structure that consists of only one neuron in the hidden layer *(line 3)*. This neuron is connected to all neurons in the input and output layers.

Figure 1 iSSNN pseudocode

D. Small non-zero random values will be assigned for each connection weight *(line 4)*.

E. The learning rate value commonly ranges from ≈0 to ≈1 [1]. Following the default value of WEKA [11], the learning rate and momentum value are assumed 0.3 and 0.2 respectively *(line 5) and (line 6)*. However, the learning rate value will be adjusted several times during the network training process.

```
Input
      Upload the minimal dataset and divide it to Training, Testing and Validation
      Integer T_k specifying the number of epochs
      Maximum number of possible hidden neurons
Output
      Optimal number of neurons in the hidden layer
      Optimal learning rate value
      Connection weights between input, hidden and output layers
Initialize
1     Number of neurons in the input layer = number of input features;
2     Number of neurons in the output layer = 1
3     Number of neurons in the hidden layer = 1;
4     Weights = random numbers ranging from -0.5 to +0.5;
5     Learning rate LR= 0.3;
6     momentum value = 0.2
7     Desired Error DER = 50%;
Start
8     Train the network to find the calculated error-rate CER;
9      If CER < DER then {
10  A: Set DER= CER;
11    Train the network;
12    Update the LR after each training epoch;
13    Check the early stopping condition;
14     If (DER Achieved && iteration< T_k) then
15          Go to A;
16     Else {
17          Add a new neuron to the hidden layer;
18          Train the network;
19            If (DER Achieved && iteration< T_k)
20              Go to A;
21            Else
22              Produce the Network;
23          }
24    Else (No Network can be produced)
End
```

Fig. 1 iSSNN pseudocode

F. The initial desired error-rate *(DER)* is set to 50% *(line 7)*. This value will be used to assess if the algorithm can find a possible solution.

G. The model designer specifies the maximum number of epochs.

H. The model designer sets the maximum number of allowed hidden neurons.

I. The training dataset is divided into training, testing, and validation datasets. The training dataset will be used to learn the model and update weights; the testing dataset will be used to assess the overall performance of the derived classifier. However, the validation dataset plays an important role in producing the final model as we will see later in the Training Phase.

2- Warming-up Training Phase

In this phase *(line 8–9)*, the algorithm decides whether to proceed with creating a new classifier or not. The algorithm computes the calculated error-rate (*CER*), aiming to determine what the *DER* to be achieved in the next training phase is. In other words, the *CER* is the *DER* to be achieved in the next training session. Hence, the algorithm trains the network and finds the *CER*. If the algorithm finds a *CER* less than the initial *DER* before reaching the maximum number of epochs, then the algorithm assumes that the current structure can further be improved; hence, the algorithm resets the epoch, and assumes that the *DER* to be achieved in the next training phase is the *CER*. Then the algorithm moves to the Training Phase. On the other hand, if the *CER* is bigger than the *DER,* then the algorithm will be terminated *(line 24)*. The *CER* is equivalent to Mean Square Error (*MSE)* and is calculated as per equation 1. Where A_k is the

predicted value for instance *k;* and D_k is the real value associated with instance k in a training dataset having n examples.

$$CER = \frac{1}{n}\sum_{k=0}^{n}(A_k - D_k)^2 \qquad (1)$$

3- Training Phase

In this phase *(line 10-13)*, the iSSNN algorithm continues training the network until the *CER* is less than the *DER* or the maximum number of epochs is reached or achieving the early stopping condition. Each training epoch starts by updating the learning rate based on the *CER* achieved in the previous epoch. One of the simplest methods for updating the learning rate is the bold driver method [12]. After each training epoch, the algorithm compares the *CER* at time *t* with the *CER* at time *t-1* and if the error has decreased, the learning rate is slightly increased by a specific ratio φ in order to accelerate the error-rate reduction process and converge quickly to the possible solution. In this case, the weights are updated. On the other hand, if the error has increased or has not changed, the iSSNN will heavily decreases the learning rate by φ' because we might be approaching one possible solution and we need to slow down and study the possible solution more deeply. In this case, the weights are not updated. Commonly, φ and φ' are set to 0.1 and 0.5 respectively [6]. The reason that φ is smaller than φ' is because we do not want to make a big step that causes the algorithm to converge from the possible solution. However, as soon as the algorithm approaches a possible solution we need to examine that solution more deeply; therefore the learning rate is heavily decreased.

Normally, the trial and error based NN structuring approach assumes that the number of neurons in the hidden layer and learning rate are fixed values that are not changed during the training phase. However, the iSSNN algorithm follows a different approach in determining the number of neurons in the hidden layer since iSSNN algorithm leaves the network expansion as a last option and keeps pushing on the learning rate to improve the NN performance as much as possible before adding a new neuron. After each training epoch, the iSSNN algorithm calculates the error on the validation dataset. When the error on the validation dataset starts to increase, that mean the model has begun to overfit the data, and the training should halt. Nevertheless, the validation dataset may have several local minima; thus, if we stop training at the first increase, we may lose some points that achieve better results because the error-rate may decrease again at some further points. Therefore, the iSSNN algorithm tracks the error on the validation dataset. If the lastly achieved error is less than the minimum achieved error, that means the generalisation ability of the model is improved; thus the algorithm saves the weights and continues training the network. On the other hand, if the lastly achieved error is bigger than the minimum achieved error, the algorithm continues the training process without saving the weights.

However, if the lastly achieved error is bigger than the minimum achieved error with a specific threshold α, then the algorithm terminates the training process (early stopping); since we assume that exceeding that threshold value might mean that the model diverges from the ideal solution and is difficult to converge back again. A recent study on early stopping [13] finds that the early stopping is triggered if α=50%. Equation 2 clarifies how the iSSNN algorithm handles the early stopping. Where, ε is the lastly achieved error, and ε' is the minimum error.

$$IF\begin{cases} \varepsilon < \varepsilon' \xrightarrow{weights\ Saved} Continue\ training \\ (\varepsilon > \varepsilon')and(\varepsilon < [(1+\alpha)*\varepsilon']) \xrightarrow{weights\ not\ Saved} Continue\ training \\ Otherwize \rightarrow Training\ terminated \end{cases} \qquad (2)$$

4- Improvement Phase

In the training phase, if the iSSNN obtains a *CER* less than the *DER* before reaching the maximum epoch, this could be an indication that the model can further be improved without adding any neurons to the hidden layer *(line 14-15)*. Therefore, the iSSNN maintains the learning rate and weights achieved so far; resets the epoch, assumes that the *DER* to be achieved in the next training phase is the *CER,* and goes back to the Training Phase. Otherwise, the iSSNN goes to Adding a New Neuron Phase *(line 16)*.

If the iSSNN cannot achieve the *DER* and reaches either the maximum allowed epoch or the early stopping condition. In this case, we assume that the current structure has been squeezed to the limit and the network's ability of processing the information is insufficient. Therefore, the algorithm will add a new neuron to the hidden layer *(line 17)*. The algorithm connects the new neuron to all input and output neurons, assigns small non-zero value to its weight, maintains the learning rate and weights achieved so far, and resets the epoch. Yet, adding a new neuron to the hidden layer does not mean that this neuron is permanently added into the network, we must first assess whether the new neuron improves the network performance or not. Hence, the algorithm continues training the network *(line 18)*.

If (after adding a new neuron) the *DER* is achieved, then the algorithm approves adding the new neuron, maintains the learning rate and weights, resets the epoch, assumes that the *DER* to be achieved in the next training phase is the *CER*, and goes back to the Training Phase *(line 19-20)*. Otherwise, the algorithm moves to Producing the Final Network Phase *(line 21)*. The main concern in this phase is that the number of hidden neurons can freely evolve resulting in a complicated structure. Thus, the algorithm allows the system designer to set the maximum number of hidden neurons.

5- Producing the Final Network Phase

If adding a new neuron to the network does not improve the network performance, then the iSSNN removes the lastly added neuron, resets the learning rate and the weights as they were before adding the new neuron, terminates the training process, and the final network is produced *(line 22)*. The most obvious network parameters to evolve in the iSSNN algorithm are the learning rate, the weights, and the number of hidden neurons. TANH activation function has been used for the input layer, whereas, the bipolar hard limit activation function has been used for the hidden layer.

IV. Evaluating the iSSNN on Phishing DataSet

several experiments will be accomplished to evaluate the applicability of the iSSNN algorithm to phishing websites classification problem. The experimental evaluation compares the iSSNN algorithm with Decision Tree (C4.5), Bayesian Network (BN), Logistic Regression (LR), and the traditional Feed Forward Neural Network (FFNN) algorithm implemented in WEKA [11]. FFNN algorithm assumes that the number of neurons in the input layers equals the number of attributes in the training dataset, whereas the number of neurons in the output layer equals the number of classes. The number of neurons in the hidden layer is the average number of neurons in the layers and is calculated as per equation (3).

$$\#\text{hidden neurons} = \frac{\#\text{input neurons}+\text{number of output neurons}}{\#\text{ of layers}} \quad (3)$$

Other algorithms were selected since they use different strategies in producing classification models. For all these algorithms, we used the default parameter settings of WEKA [11]. Whereas, for iSSNN algorithm, two input values should be entered from the system designer, i.e. number of epochs and maximum number of possible hidden neurons. There is no rule of thumb in which one can decide on these values [1]. Therefore, following some recent studies which employ NN to create classification models in different domains [14] [15] [16] , we set the maximum number of possible neurons to 10. Yet, these studies utilise different epoch sizes and the most commonly used epoch size values are 100, 200, 500, and 1000. Four sub-experiments will be conducted, in which the maximum number of possible hidden neurons is 10, and epoch size has been set to 100, 200, 500, and 1000 for experiments 1, 2, 3, and 4 respectively. The iSSNN algorithm has been implemented in Java. All experiments were conducted in a system with CPU Pentium Intel® CoreTM i5-2430M @ 2.40 GHz, RAM 4.00 GB. The platform is Windows 7 64-bit Operating System.

V. Training Datasets

We have used the well-known phishing websites training dataset from the University of California Irvine repository (UCI) [17]. Table 1 shows the description of the training dataset, i.e. number of attributes, number of instances, and class distribution.

Table 1 UCI dataset

Number of attributes	Number of Instances	Class Distribution	
		Phishing	Legitimate
30	11055	44%	56%

The dataset was collected recently by one of the authors of this article and published in UCI repository. Most of the dataset's attributes are binary (0, 1) or ternary (0,1,-1). The dataset is categorized under classification in data mining since there is class label added (target attribute) that has two possible values (Phishy -1, Legitimate 1).

More details on the features names, types, possible values and descriptions are given in [17].

VI. Validation Technique

The iSSNN algorithm splits the training dataset into training, testing and validation datasets. The hold-out validation technique is used in our experiments. Thus, the dataset will be divided into 80% for training and 20% for testing. Moreover, when creating the iSSNN classifiers the training datasets will be further divided into 80% for training and 20% for validation.

VII. Evaluation Metrics

Four classification possibilities have been employed in our experiments as per confusion matrix shown in Table 2.

Table 2 Confusion Matrix

Actual Value	Predicted Value		
		Positive	***Negative***
	Positive	*TP*	*FN*
	Negative	*FP*	*TN*

Where True Positive (TP) is the number of legitimate websites correctly classified as legitimate, False Negative (FN) is the number of legitimate websites incorrectly classified as phishing, False Positive (FP) is the number of phishing websites incorrectly classified as legitimate and True Negative (TN) is the number of phishing websites correctly classified as phishing. Following previous studies related to phishing classification [18], [7], [19], [8], [20], [21], [22] and [23] we use a set of evaluation metrics that can be derived from the confusion matrix shown in Table 2. These evaluation metrics are as follows:

1. Precision (P): the rate of correctly classified legitimate websites in relation to all instances that are classified as legitimate and is calculated as per the equation 4.
$$\mathrm{P}\,\frac{\mathrm{TP}}{\mathrm{TP+FP}} \quad (4)$$
2. Recall (R): is equivalent to TPR (Sensitivity).
3. F1-score (Harmonic Mean): is the weighted average of P and R. F1-score takes both FP and FN into account and is calculated as per equation 5. This metric weights R and P equally, and a good classifier will maximize both P and R simultaneously. Thus, moderately good performance on both will be favoured over good performance on one and poor performance on the other.
$$F1 = \frac{2\,P\,R}{P+R} \quad (5)$$
4. Accuracy (ACC): the overall rate of correctly classified legitimate and phishing websites in relation to the total number of instances in the testing data set and is calculated as per equation 6.
$$\mathrm{Acc} = \frac{\mathrm{TP+TN}}{\mathrm{TP+FP+TN+FN}} \quad (6)$$

VIII. Experimental Results

Three experiments have been done with the aim of evaluating the SSNN algorithm and compare the results with other DM classification algorithms. Information Gain, Chi-Square and Gain Ratio have been used in experiments 1, 2 and 3 respectively. The selection for these methods is because they are commonly used for feature selection in the domain of

phishing websites classification [24], [25], [26], [22], [23]. The results are shown in Tables 3, 4, and 5.

Table 3 Experimental results when using Information Gain

		Number of selected features											
	Algorithm	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	Average
Accuracy	C4.5	92.12	92.25	91.73	92.25	91.60	94.17	92.25	92.37	92.12	92.12	91.99	92.27
	BN	90.31	91.09	91.47	91.21	91.73	92.67	91.86	91.99	92.38	91.86	91.99	91.69
	LR	90.31	91.60	91.34	91.09	91.60	93.26	91.73	91.6	91.86	91.99	91.99	91.67
	FFNN	91.47	92.64	92.37	91.99	92.51	94.03	93.02	93.02	93.93	92.76	94.32	92.91
	SSNN-100	91.73	92.76	91.99	92.37	92.89	94.66	92.76	91.99	93.02	93.02	94.32	92.86
	SSNN-200	92.37	92.76	91.73	92.51	92.51	94.52	93.15	92.25	93.28	92.89	93.41	92.85
	SSNN-500	92.25	92.89	92.37	92.25	91.99	94.17	93.02	93.67	93.41	93.28	94.41	93.06
	SSNN-1000	91.99	92.63	91.99	92.25	92.25	94.39	93.15	93.15	93.41	93.54	94.06	92.98
	Average	91.57	92.33	91.87	91.99	92.14	93.98	92.62	92.51	92.93	92.68	93.31	92.54
F1-score	C4.5	91.50	91.60	91.10	92.40	90.90	93.40	91.50	91.60	91.40	91.40	91.20	91.64
	BN	89.10	90.00	90.50	90.30	90.80	91.70	91.00	91.10	91.60	91.00	91.10	90.75
	LR	89.10	90.70	90.40	90.10	90.70	92.40	90.90	90.80	91.10	91.30	91.30	90.80
	FFNN	90.50	91.70	91.50	91.00	91.70	93.40	92.20	92.30	93.40	92.00	93.80	92.14
	SSNN-100	90.90	91.90	90.90	91.40	92.20	94.10	91.90	91.10	92.30	92.50	93.80	92.09
	SSNN-200	91.70	91.90	90.70	91.60	91.70	93.90	92.40	91.40	92.60	92.20	92.80	92.08
	SSNN-500	91.60	92.00	91.60	91.40	91.20	93.50	92.30	93.10	93.40	92.50	92.70	92.30
	SSNN-1000	91.20	91.70	91.00	91.30	91.50	93.80	92.40	92.50	92.80	92.90	93.50	92.24
	Average	90.70	91.44	90.96	91.19	91.34	93.28	91.83	91.74	92.33	91.98	92.53	91.75
Recall	C4.5	92.70	92.10	92.10	89.90	91.00	91.20	90.40	89.90	90.70	90.70	90.70	91.04
	BN	85.70	87.60	88.50	88.50	88.80	89.60	89.60	89.60	90.20	89.30	89.60	88.82
	LR	85.70	89.30	88.80	88.50	89.30	90.60	90.20	90.20	90.70	91.00	91.00	89.57
	FFNN	88.80	88.80	89.30	87.90	90.20	93.60	89.90	90.70	92.70	91.00	93.80	90.61
	SSNN-100	89.90	88.80	87.40	88.20	90.70	93.70	89.30	89.00	90.40	93.30	94.10	90.44
	SSNN-200	91.90	88.80	87.60	88.80	90.20	93.60	90.20	89.30	91.00	91.60	92.10	90.46
	SSNN-500	92.40	88.80	89.90	89.60	90.20	92.40	91.00	92.40	94.40	90.20	91.00	91.12
	SSNN-1000	89.90	88.80	87.90	88.80	90.20	92.40	90.70	90.80	91.90	92.10	93.00	90.59
	Average	89.63	89.13	88.94	88.78	90.08	92.14	90.16	90.24	91.50	91.15	91.91	90.33
Pecision	C4.5	90.40	91.10	90.10	93.00	90.80	95.70	92.50	93.30	92.00	92.00	91.80	92.06
	BN	92.70	92.60	92.60	92.10	92.90	93.90	92.50	92.70	93.00	92.70	92.70	92.76
	LR	92.70	92.20	92.10	91.80	92.20	94.30	91.70	91.50	92.50	91.50	91.50	92.18
	FFNN	92.40	95.50	93.80	94.30	93.30	93.20	94.70	93.90	93.35	93.10	93.80	93.76
	SSNN-100	92.00	95.20	94.80	94.90	93.60	94.50	94.60	93.20	93.20	91.70	93.40	93.74
	SSNN-200	91.60	95.20	94.00	94.60	92.30	94.30	94.70	93.50	94.20	92.90	93.40	93.70
	SSNN-500	90.90	94.90	93.30	93.40	92.20	94.60	93.60	93.70	94.40	95.30	94.50	93.71
	SSNN-1000	92.50	94.90	94.30	93.60	92.40	94.20	94.20	93.70	93.70	93.70	94.00	93.75
	Average	91.90	93.95	93.13	93.46	92.46	94.34	93.56	93.19	93.29	92.86	93.14	93.21

Table 4 Experimental results when using Chi-Square

		Number of selected features											
	Algorithm	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	Average
Accuracy	C4.5	92.12	92.25	91.73	92.25	91.60	91.73	92.25	92.64	92.12	92.12	91.99	92.07
	BN	90.31	91.09	91.47	91.21	91.73	91.60	91.86	92.12	92.38	91.86	91.99	91.60
	LR	90.31	91.60	91.34	91.09	91.60	91.60	91.73	92.25	91.86	91.99	91.99	91.58
	FFNN	89.02	93.02	92.37	92.51	91.99	92.76	93.15	93.93	93.54	93.67	93.02	92.63
	SSNN-100	91.99	92.76	92.37	91.73	92.64	92.38	93.28	93.28	93.67	93.67	93.66	92.86
	SSNN-200	91.99	92.64	92.37	92.37	92.76	92.25	93.67	93.92	93.67	93.67	93.02	92.94
	SSNN-500	92.62	92.64	92.37	92.51	91.86	92.40	93.90	93.15	93.67	93.67	93.54	92.94
	SSNN-1000	92.24	92.76	92.51	93.02	92.25	92.25	93.67	93.28	94.06	92.57	93.02	92.88
	Average	91.33	92.35	92.07	92.09	92.05	92.12	92.94	93.07	93.12	92.90	92.78	92.44
F1-score	C4.5	91.50	91.60	91.10	91.40	90.90	90.90	91.50	92.00	91.40	91.00	91.20	91.32
	BN	89.10	90.00	90.50	90.30	90.80	90.70	91.00	91.30	91.60	91.00	91.10	90.67
	LR	89.10	90.70	90.40	90.10	90.70	90.70	90.90	91.50	91.10	91.30	91.30	90.71
	FFNN	88.80	92.20	91.50	91.60	91.20	91.90	92.50	93.30	92.80	92.90	92.20	91.90
	SSNN-100	91.40	91.90	91.60	90.90	91.90	91.50	92.60	92.60	93.00	93.00	93.00	92.13
	SSNN-200	91.40	91.70	91.60	91.60	92.00	91.30	93.00	92.30	92.90	93.00	92.20	92.09
	SSNN-500	91.70	91.70	91.40	91.70	91.10	91.50	93.20	92.40	93.00	93.00	92.80	92.14
	SSNN-1000	91.60	91.90	91.70	92.30	91.50	91.30	93.00	92.60	93.30	92.00	92.20	92.13
	Average	90.58	91.46	91.23	91.24	91.26	91.23	92.21	92.25	92.39	92.15	92.00	91.64
Recall	C4.5	92.70	92.10	92.10	89.90	91.00	89.30	90.40	91.90	90.70	89.30	90.70	90.92
	BN	85.70	87.60	88.50	88.50	88.80	89.00	89.60	89.90	90.20	89.30	89.60	88.79
	LR	85.70	89.30	88.80	88.50	89.30	89.30	90.20	91.00	90.70	91.00	91.00	89.53
	FFNN	89.00	89.30	89.30	89.00	90.20	89.00	91.60	91.30	91.00	90.70	89.90	90.03
	SSNN-100	92.40	89.00	90.40	89.30	90.40	89.30	91.00	91.90	91.10	91.10	91.10	90.64
	SSNN-200	92.40	88.80	89.90	89.90	91.00	88.80	92.10	91.60	90.70	92.30	89.90	90.67
	SSNN-500	91.90	89.00	89.80	89.60	90.40	89.80	91.90	91.30	91.90	91.30	90.70	90.69
	SSNN-1000	92.40	89.00	89.90	91.00	90.70	88.50	91.90	91.30	91.60	89.90	89.90	90.55
	Average	90.28	89.26	89.84	89.46	90.23	89.13	91.09	91.28	90.99	90.61	90.35	90.23
Pecision	C4.5	90.40	91.10	90.10	93.00	90.80	92.40	92.50	92.10	92.00	92.00	91.80	91.65
	BN	92.70	92.60	92.60	92.10	92.90	92.40	92.50	92.80	93.00	92.70	92.70	92.64
	LR	92.70	92.20	92.10	91.80	92.20	92.20	91.70	92.00	91.50	91.50	91.50	91.95
	FFNN	90.30	95.20	93.80	94.30	92.20	94.90	93.40	94.30	94.70	93.10	94.70	93.72
	SSNN-100	90.40	94.90	92.80	92.40	93.30	93.80	94.20	93.40	94.00	94.00	94.00	93.38
	SSNN-200	90.40	94.90	93.30	93.30	93.10	94.00	94.00	95.00	95.30	93.80	93.70	93.71
	SSNN-500	91.60	93.90	93.90	93.40	91.60	94.40	94.40	93.90	94.20	94.80	94.50	93.69
	SSNN-1000	90.90	94.90	93.60	93.60	92.30	94.30	94.20	93.90	95.30	93.90	93.70	93.69
	Average	91.18	93.71	92.78	92.99	92.30	93.55	93.36	93.43	93.75	93.23	93.33	93.05

Table 5 Experimental results when using Gain Ratio

		Number of selected features											
	Algorithm	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	Average
Accuracy	C4.5	92.12	92.12	92.12	91.60	91.60	91.99	92.25	92.12	92.24	92.24	92.25	92.06
	BN	90.31	91.21	91.09	91.47	91.73	92.12	91.86	91.86	91.99	91.86	92.25	91.61
	LR	90.31	91.86	91.86	92.12	91.60	91.86	91.73	91.86	91.86	91.86	91.86	91.71
	FFNN	91.47	91.99	89.92	91.09	92.51	92.76	92.76	92.64	92.64	92.89	93.54	92.20
	SSNN-100	91.99	92.24	91.37	91.09	93.02	92.89	93.54	92.41	93.15	93.02	92.89	92.51
	SSNN-200	91.99	92.12	90.95	90.70	92.38	93.54	93.02	92.41	93.67	93.67	93.80	92.57
	SSNN-500	92.25	92.37	90.05	91.09	91.99	93.15	93.28	93.28	93.67	93.67	93.67	92.59
	SSNN-1000	92.25	92.37	90.17	91.09	92.63	92.78	93.28	93.28	93.67	93.28	93.67	92.59
	Average	91.59	91.99	90.94	91.31	92.18	92.64	92.63	92.37	92.75	92.81	92.89	92.23
F1-score	C4.5	91.50	91.50	91.50	91.00	90.90	91.30	91.50	91.30	91.40	91.40	91.50	91.35
	BN	89.10	90.20	90.10	90.50	90.80	91.30	91.00	91.00	91.10	91.00	91.40	90.68
	LR	89.10	91.10	91.10	91.30	90.70	91.10	90.90	91.10	91.10	91.10	91.10	90.88
	FFNN	90.50	91.10	88.90	90.20	91.70	92.10	91.90	91.80	91.90	92.20	92.90	91.38
	SSNN-100	91.40	91.60	91.00	90.20	92.30	92.30	92.90	91.70	92.50	92.40	91.89	91.84
	SSNN-200	91.40	91.50	90.20	89.70	91.60	92.90	92.30	91.70	93.00	93.10	92.80	91.84
	SSNN-500	91.60	91.70	89.00	90.20	91.10	92.50	92.60	92.60	93.00	93.00	93.00	91.85
	SSNN-1000	91.60	91.70	89.10	90.20	91.80	92.20	92.60	92.60	92.90	92.60	92.90	91.84
	Average	90.66	91.24	90.11	90.44	91.36	91.96	91.87	91.60	92.00	92.10	92.08	91.46
Recall	C4.5	92.40	92.10	90.70	88.80	90.70	92.70	91.30	90.60	91.30	91.10	92.20	91.26
	BN	85.70	88.20	88.50	88.50	88.80	89.60	89.60	89.60	89.60	89.60	89.60	88.85
	LR	85.70	90.20	90.20	89.90	89.30	90.40	90.20	90.20	90.40	90.40	90.20	89.74
	FFNN	88.80	88.80	87.40	88.80	90.20	92.90	89.30	89.60	91.00	91.00	91.60	89.95
	SSNN-100	92.70	92.70	92.70	92.10	91.00	91.00	90.40	90.40	89.90	89.90	90.20	91.18
	SSNN-200	92.40	91.90	90.40	87.90	90.70	92.10	91.00	90.60	91.30	92.40	92.10	91.16
	SSNN-500	92.40	91.90	87.60	88.80	90.20	91.90	90.70	91.00	91.30	91.90	91.30	90.82
	SSNN-1000	92.40	91.90	87.60	88.80	89.90	91.60	90.70	91.00	90.40	91.00	91.60	90.63
	Average	90.01	90.96	89.39	89.26	90.10	91.53	90.36	90.29	90.65	90.91	91.10	90.45
Pecision	C4.5	90.40	90.40	90.40	89.90	90.80	91.50	92.50	92.30	93.00	93.00	92.80	91.55
	BN	92.70	92.40	91.80	92.60	92.90	93.00	92.50	92.50	92.70	92.50	93.30	92.63
	LR	92.70	92.00	92.00	92.80	92.20	91.70	91.70	92.50	91.70	91.70	92.00	92.09
	FFNN	92.40	93.50	90.40	91.60	93.30	92.40	94.60	94.10	92.80	93.40	94.20	92.97
	SSNN-100	90.40	91.10	91.20	91.60	93.90	91.90	94.50	92.90	93.70	92.70	93.40	92.48
	SSNN-200	90.40	91.10	89.90	91.50	92.60	93.70	93.60	92.90	94.80	93.70	94.00	92.56
	SSNN-500	90.90	91.60	90.40	91.60	93.00	93.10	94.40	94.20	94.80	94.20	94.80	93.00
	SSNN-1000	90.90	91.60	90.70	91.60	93.80	92.90	94.40	94.20	95.50	94.20	93.05	92.99
	Average	91.41	91.73	90.85	91.66	92.81	92.53	93.40	93.06	93.63	93.18	93.44	92.53

From the results, we can see that the iSSNN outperforms the considered classification algorithms in most cases, particularly when the epoch size is set to 500. For instance, the average F1-score produced from the iSSNN when using the Information Gain is higher than that produced from C4.5, BN, LR, and FFNN with margins of 0.66, 1.55%, 1.50%, and 0.16% respectively when the epoch size is set to 500. In addition, the average F1-score produced from the iSSNN when using the Chi-Square is also beats C4.5, BN, LR, and FFNN with margins of 0.82%, 1.46%, 1.43%, and 0.24% respectively when the epoch size is set to 500. Again, when using the same epoch size, the average F1-score produced from the iSSNN algorithm when using the Gain Ration outperforms C4.5, BN, LR, and FFNN with margins of 0.50%, 1.16%, 0.96%, and 0.46% respectively.

Overall, the high F1-score yielded from the iSSNN reflects that the algorithm is able to derive classifiers that produce good FP and FN rates. That can be attributed due to the well-structured NN classifiers derived from the iSSNN algorithm as a result to the good training procedure employed in the algorithm. In general, the F1-score produced when using different feature selection methods reflects that the NN based algorithms derive better classifiers than other considered classification algorithms when applied to phishing datasets in the sense that the second best result achieved in all experiments was from the FFNN. However, the highest F1-score produced from the iSSNN was when using the Information Gain for feature selection at 92.30%. This value

is higher than the values produced from Chi-Square and Gain Ration with margins of 0.16% and 0.45% respectively.

In terms of average accuracy, and when the epoch size is set to 500, the iSSNN outperforms C4.5, BN, LR, and FFNN with margins of 0.79%, 1.38%, 1.39%, and 0.15% respectively when using the Information Gain. In addition, the average accuracy produced from the iSSNN algorithm when using the Chi-square outperforms C4.5, BN, LR, and FFNN with margins of 0.87%, 1.34%, 1.36%, and 0.30% respectively. Further, the iSSNN algorithm beats C4.5, BN, LR, and FFNN with margins of 0.53%, 0.97%, 0.88% and 0.39% respectively when using the Gain Ration. Overall, the high accuracies produced from the iSSNN when using different feature selection methods are good sign that the training procedure in the iSSNN algorithm is able to produce well-structured NN classifiers. Yet, the highest average accuracy produced from the iSSNN was when the Information Gain has been used for feature selection at 93.06%. This value bypasses the results achieved from Chi-Square and Gain Ration with margins of 0.12% and 0.47% respectively.

In terms of average Recall, we find that the iSSNN algorithm has been defeated two times from the C4.5 when using Chi-Square and Gain Ration with margins of 0.23% and 0.45% respectively when the epoch size is set to 500. However, when using the Information Gain, we find that the iSSNN outperforms the C4.5 with a margin of 0.08%. This difference is relatively small. However, a good classification model is the model that is able to maximize both Precision and Recall simultaneously. Yet, from the results, we find that although the average Recall produced from C4.5 beats the iSSNN algorithm when using Chi-Square and Gain Ration, the iSSNN algorithm outperforms the C4.5 in terms of average Precision with 2.04% and 1.45% when using Chi-Square and Gain Ration respectively when the epoch size is set to 500. Such results confirm that the iSSNN algorithm is able to derive classifiers that show a moderately good performance on both Precision and Recall. The same scenario is also happens with FFNN, since the FFNN produced higher precision than the iSSNN with margins of 0.03% when using Chi-Square when the epoch size is set to 500. Yet, when using the same epoch size, the iSSNN produced higher Recalls than the FFNN with margins of 0.66% and 0.87% when using Chi-Square and Gain Ration respectively.

Overall, the training procedure utilised when deriving NN classifiers using the iSSNN algorithm has proven to be effective in creating well-structured models in terms of number of hidden neurons and weights space.

IX. SUMMARY

In this article we proposed an improved self-structuring neural network algorithm that simplifies structuring NN classifiers. several experiments have been accomplished to evaluates the applicability of the iSSNN on phishing websites data set. Three feature selection methods have been used in order to evaluate these methods and their effect on the performance of the iSSNN and other considered classification algorithms. The results show that the iSSNN algorithm outperformed the considered classification algorithms in most cases. The classifiers produced from the iSSNN have been shown to produce a moderately good performance on both Precision and Recall. However, the experimental results revealed that the Information Gain is more effective than other feature selection methods in improving the performance of the SSNN and other considered classification algorithms. In general, the experimental results show that the iSSNN algorithm is able to produce good NN classifiers.

References

[1] I. Basheer and M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application.," *Journal of Microbiological Methods.,* vol. 43, no. 1, pp. 3-31, 2000.

[2] M. Arvandi, S. Wu and A. Sadeghian, "On the use of recurrent neural networks to design symmetric ciphers," *Computational Intelligence Magazine, IEEE,* vol. 3, no. 2, pp. 42-53, 2008.

[3] K. M. Alallayah, W. AbdElwahed, M. Amin and A. H. Alhamami, "Attack of Against Simplified Data Encryption Standard Cipher System Using Neural Networks," *Journal of Computer Science,* vol. 6, no. 1, pp. 29-35, 2010.

[4] M. Al-Ubaidy, "Black-box attack using neuro-identifier," *Cryptologia (Taylor & Francis),* vol. 28, no. 4, pp. 358-372, 2004.

[5] I. Basheer and M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application," *Journal of Microbiological Methods.,* vol. 43, no. 1, pp. 3-31, 2000.

[6] S. Duffner and C. Garcia, "An Online Backpropagation Algorithm with Validation Error-Based Adaptive Learning Rate," in *Artificial Neural Networks – ICANN 2007*, Porto, Portugal, 2007.

[7] R. M. Mohammad, F. Thabtah and L. McCluskey, "Predicting phishing websites based on self-structuring neural network," *Neural Computing and Applications,* vol. 25, no. 2, pp. 443-458, 2013-B.

[8] R. M. Mohammad, F. Thabtah and L. McCluskey, "Predicting Phishing Websites using Neural Network trained with Back-Propagation," in *ICAI*, Las Vigas, 2013-C.

[9] M. Islam, A. Sattar, F. Amin, X. Yao and K. Murase, "A New Adaptive Merging and Growing Algorithm for Designing Artificial Neural Networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics,* vol. 39, no. 3, pp. 705-722, 2009.

[10] T.-Y. Kwok and D.-Y. Yeung, "Constructive algorithms for structure learning in feedforward neural networks for regression problems," *IEEE Transactions on Neural Networks,* vol. 8, no. 3, pp. 630-645, 1997.

[11] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. H. Witten, "Waikato Environment for Knowledge Analysis," University of Waikato, 2011. [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/. [Accessed 20 December 2011].

[12] R. Battiti, "Accelerated Backpropagation Learning: Two Optimization Methods," *Complex Systems,* vol. 3, no. 4, pp. 331-342, 1989.

[13] M. Riley, J. Karl and T. Chris, "A Study of Early Stopping, Ensembling, and Patchworking for Cascade Correlation Neural Networks," *IAENG International Journal of Applied Mathematics,* vol. 40, no. 4, pp. 307-316, 2010.

[14] V. Kesari, L. K. Verma and P. Tripathi, "Image Classification using Backpropagation Algorithm," *Journal of Computer Science,* vol. 1, no. 2, 2014.

[15] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen and T. Sainath, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine,* vol. 29, no. 6, pp. 82-97, 2012.

[16] S. Madhusmita, S. K. Dash, S. Dash and A. Mohapatra, "An approach for iris plant classification using neural network," *International Journal on Soft Computing ,* vol. 3, no. 1, 2012.

[17] R. M. Mohammad, L. McCluskey and F. Thabtah, "Phishing Websites Data Set," University of California, Irvine, School of Information and Computer Sciences, 2015. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Phishing+Websites. [Accessed 26 March 2015].

[18] R. M. Mohammad, F. Thabtah and L. McCluskey, "An assessment of features related to phishing websites using an automated technique," in *International Conference for Internet Technology And Secured Transactions, 2012*, London, 2012 A.

[19] R. M. Mohammad, F. Thabtah and L. McCluskey, "Intelligent Rule based Phishing Websites Classification," *IET Information Security,* vol. 8, no. 3, pp. 153-160, July 2013-A.

[20] Y. Zhang, J. Hong and L. Cranor, "CANTINA: A Content-Based Approach to Detect Phishing Web Sites.," in *The 16th World Wide Web Conference. WWW '07*, Banff, AB, Canada., 2007.

[21] M. Aburrous, M. A. Hossain, K. Dahal and F. Thabtah, "Intelligent phishing detection system for e-banking using fuzzy data mining," *Expert Systems with Applications: An International Journal,* vol. 37, no. 12, pp. 7913-7921, December 2010 C.

[22] N. Abdelhamid, A. Ayesh and F. Thabtah, "Phishing detection based Associative Classification data mining," *Expert Systems with Applications,* vol. 41, no. 13, p. 5948–5959, 2014.

[23] H. Y. Mansour and H. A. Alshihri, "Adapting associative classification for detecting phishing websites," in *The First Summit on Countering Cyber Crimes*, Riyadh, 2015.

[24] Y. Pan and X. Ding, "Anomaly Based Web Phishing Page Detection," in *The 22nd Annual Computer Security Applications Conference (ACSAC)*, Miami Beach, Florida, USA, 2006.

[25] J. Ma, L. K. Saul, S. Savage and G. M. Voelker, "Identifying suspicious URLs: an application of large-scale online learning," in *The 26th Annual International Conference on Machine Learning*, Montreal, Canada, 2009.

[26] M. Aburrous, M. A. Hossain, K. Dahal and F. Thabtah, "Predicting Phishing Websites using Classification Mining Techniques," in *The Seventh International Conference on Information Technology*, Las Vegas, Nevada, USA, 2010 B.