Credit Scoring Model Based on Back Propagation Neural Network Using Various Activation and Error Function

Mulhim Al Doori and Bassam Beyrouti

American University in Dubai, Computer College

Abstract
The Back Propagation algorithm of Neural Networks is a widely used learning technique for training a multi layered perceptron network. The algorithm applies error propagation from outputs to inputs and gradually fine tunes the network weights to minimize the sum of error using the gradient descent technique. Activation functions are employed at each neuron level to provide non-linearity to the network. In this paper, an attempt has been made to assess and compare the results using a combination of activation and error functions applied differently on the hidden and output layers of the network. Sigmoid, Hyperbolic Tangent and Gaussian are the activation functions under study. Furthermore, error functions such as the Mean Squared Error, Huber, and the Complex Sine-Hyperbolic have been considered.
Key words:
Artificial Neural Network, Back Propagation Algorithm, Activation Function, Error Function

1 Introduction

Due to the fact that credit industry has prospered in the last decade, credit assessment of loan application becomes even more critical to Banks and lenders. Credit scoring is a predictive model used to classify a new applicant as good, a customer who is likely to repay financial obligation and thus to accept the application, or bad, a customer who has high possibility of defaulting on loan payments and thus to reject the application due to the incurred costs and profit loss. Behavioral Scoring is another type of credit evaluation which supervises existing customers and decides whether to increase their credit limit. [4] [6]

Previously, creditworthiness was determined by a set of credit analysts who evaluated the customer loan application. Analysts based their judgments on the customer application details, such as time at address, current employment, residential status, spouse’s employments, and number of children and dependents. Other details were requested from the Credit Bureau such as, existing bank accounts, credit cards, number of inquiries on the applicant from other agencies, number of loan defaults and reported bankruptcies, and fraud reports. [3]

With the increased demand on credit loans and the limited number of credit experts, lenders require an efficient and accurate automated credit scoring model. Therefore, numerous attempts have been introduced. Because of the significant number of customer portfolios, a slight enhancement in credit scoring system could lead to loss reduction, future savings, faster processing, and a closer behavioral study on the existing customers. [19] [20]. In order to compare the effect of applying different activation functions, the Australian credit Dataset is utilized. [1]

Bicer et al. showed that Bayesian credit scoring model outperforms Logistic Regression classification models. [2] Chen and Li selected two credit scoring data sets to evaluate the accuracy of their proposed hybrid classifier using the K-Nearest Neighbor, Support Vector Machine, Back-Propagation Network, and the Extreme Learning Machine on a selection of data features. Results show that the F-score is the best selection approach for features selection combined with the KNN and SVM classifiers in the Australian and German data sets respectively. [3] Chuang and Huang proposed two-stage credit scoring models. In the first stage, applicants are grouped into accepted and rejected groups. The second stage retrieves some of the initially rejected good applicants to conditional acceptance. The model recovers potentially misclassified applicants and increase financial revenues. [4] Gangal et al. proposed that results can be improved by selecting a proper error function, namely the Huber Error Function, to minimize the error rate and expedite the weight update rate. [5] Gao et al. proposed Structure Tuning Particle Swarm Optimization (SPSO) which deleted redundant connections between neurons to optimize the structure of the neural network and generated a compact network version. [6] Heiet obtained results showing that Markov-FS model is slightly better than the Markov model by saving data collection, entry and processing times. [7] Hsieh and Hung proved that the relatively large variation within a data set may affect the performance of ensemble classifier over the classifications based on data reduction technique. [8] Hu and Ansell applied five credit scoring models; Naïve Bayes, Logistic Regression, Recursive Partition, Artificial Neural Network, and the Sequential Minimal Optimization on the US retail market. [9] Karlik and Olgac showed that Hyperbolic Tangent Function (tanh) has better performance when applied on both hidden and output network layers. [10] Khashman demonstrated that the selection of an

Manuscript received March 5, 2014
Manuscript revised March 20, 2014

appropriate-to-validation data ratio may affect the neural network performance. In addition, Single-Hidden Layer Neural Network (SHNN) model outperformed the Double-Hidden Layer Neural Network (DHNN) when applied to the credit scoring data set. [11] Lee and Chen proposed a two-stage hybrid credit scoring model which combined Artificial Neural Networks with Multivariate Adaptive Regression Splines (MARS). Results showed that the model has significant performance increase compared to discriminant analysis, logistic regression, artificial neural networks and MARS. [12] Marcano-Cedeno et al. presented an innovative approach inspired by the neuron's biological property of meta-plasticity. The Artificial Meta-Plasticity implementation on Multi-layer Perceptron AMMLP model trained by Back-Propagation algorithm has shown superior results compared to the traditional MLP. [13] Siami et al. proposed a hybrid mining model which combined three classifiers, Artificial Neural Network, Support Vector Machine, and Naïve Bayesian Networks. In order to improve the model accuracy, a majority voting technique is used through implementation to make the most likely decision. [14] Sibi et al. proved that although the selection of a proper activation function is extremely important, factors such as learning rate, momentum, network size, and the number of hidden neurons are more vital for an efficient network training. [15] Sentiono et al. pruned a Neural Network by removing unnecessary weights by the Input and the Hidden Layers using a novel pruning approach. [16] Tong et al. demonstrated that General Regression Neural Network has the best credit scoring model among LDA, LR, Quadratic Discriminant Analysis, and Back Propagation Neural Network (BPNN). [17] Tsai compared the performance of the Support Vector Machine with a Multi-Layer Perceptron Network as the benchmark classifier. In order to obtain fair financial decisions, at least two data sets should be used. Changing the training-to-validation data set ratio does not yield to significant performance changes. The results showed that MLP's performance is superior to the SVM's in financial decision making. [18] Wu proposed a data preprocessing technique augmented with a Bayesian Network based on Tree Augmented Naïve Bayes search algorithm might enhance credit scoring decision making. [19] Zhou et al. applied Area Under Receiver Operating Characteristics Curve (AUC) maximization on two credit scoring data sets based on Support Vector Machines. Results show that AUC has better performance than that of the Linear Regression, Decision Tree, Neural Network, and K-Nearest Neighbor. [20]

2 Artificial Neural Networks

Artificial Neural Networks are mathematical representations inspired by the human brain nerve cells and their communication and processing information techniques. The Neural Network is ideally composed of three layers, the input layer, the hidden layer, and the output layer. The input layer consists of input nodes which represent the system's variable. The hidden layer consists of nodes which facilitate the flow of information from the input to the output layers. The flow is controlled by weight factors associated with each connector. The output layer consists of nodes which represent the system's classification decision. The value of the output nodes are compared with cutoffs to determine the output and classify each case. The weight adjustment is known as training. The training process consists of running input values over the network with predefined classification output nodes. This process runs until the weight values are minimized to an error function. Testing samples are used to verify the performance of the trained network. In the context of credit scoring, numerous studies have proven that Neural Network perform remarkably better than any other statistical approach, such as logistic regression or discriminant analysis. [9]

2.1.1 Activation Functions

Neural networks are characterized by a processing element with numerous synaptic weighted connections and a single output determined by a given relationship. The signal flow is considered to be unidirectional. Each activation function is characterized by its shape, output range, and derivative function. In order to serve the purpose of this paper, activation functions are selected based on their popularity and performance in the context of credit scoring.

The Sigmoid Function

The Sigmoid function is a commonly used "S" shape differentiable activation function in training Neural Networks. Sigmoid function is the most advantageous activation function used in Neural Networks trained with back propagation algorithm with a binary output. Since it can be easily differentiated, the function minimizes the computational cost during training phase. The Sigmoid function produces outputs between 0 and 1. The function is represented by equation (1):

$$y = \frac{1}{1+e^{-2c.x}} \quad (1)$$

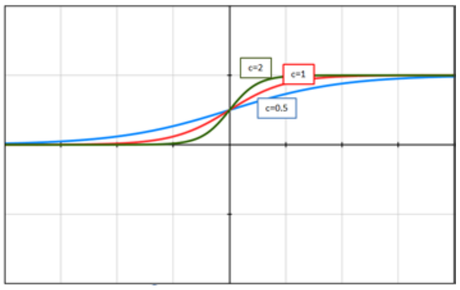


Figure 1. Sigmoid Function

The Hyperbolic Tangent Function

The hyperbolic Tangent Function, also known as Sigmoid Symmetric Activation Function, one of the most used activation functions, bounds the output between -1 and +1. The function is defined as follows:

$$y = \tanh(c.x) \tag{2}$$

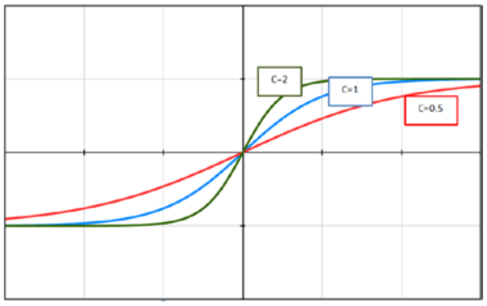


Figure 2. Hyperbolic Tangent Function

The function is also known as Sigmoid Symmetric Function.

The Gaussian Symmetric Function

The Gaussian Symmetric Function is mainly used to fine tune the output of the activation function. The function is defined by:

$$y = e^{-c^2x^2} \tag{3}$$

The output is limited between 0 and +1 as shown below

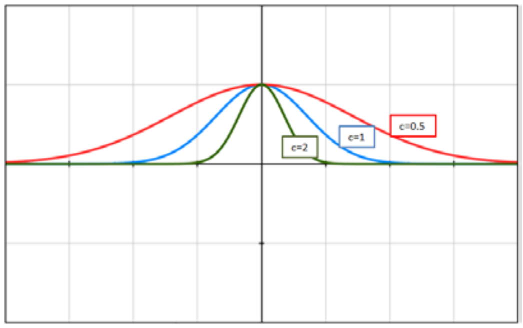


Figure 3. Gaussian Symmetric Function

2.2 Error Functions

The weights of the Back Propagation learning algorithm are initialized with random variables. The neuron outputs are calculated using these weights. Error is measured between the actual and the desired outputs. This error is back propagated. New weights are recalculated and thus neuron outputs are re-evaluated. This process is iterated until the error is minimized to a defined value. We have applied the following error functions to evaluate their respective performance.

Mean Squared Error Function

The Mean Squared Error function is defined by:

$$e = \frac{1}{2}\sum(t-a)^2 \tag{4}$$

where t: is the desired target, and a is the actual output

The Huber Error Function

The Huber Error Function is used to minimize error values due to network training with noisy data

$$h(e) = \begin{cases} \frac{1}{2}.(a-t)^2, & |a-t| < cx < 0 \\ c.|a-t| - \frac{1}{2}.c, & |a-t| \geq cx \geq 0 \end{cases} \tag{5}$$

The Complex Hyperbolic Sine Function

The Complex Hyperbolic Sine Function is defined as follows:

$$f(x) = \sinh(|x|) \tag{6}$$

2.3 Hidden-to-Output Layer Weight Update

In this section, a thorough mathematical derivation is carried out to update the network weights:

$$\omega_{jk} = \omega_{jk} + \Delta\omega_{jk} \tag{7}$$

where: $\Delta\omega_{jk} = -\alpha.\delta_k.a_j + \beta.\Delta\omega_{jk}(t-1)$

Applying the chain rule for partial derivatives:

$$\Delta\omega_{jk} = -\frac{\partial E}{\partial\omega_{jk}} = -\alpha.\underbrace{\left(\frac{\partial E}{\partial a_k}\right).\left(\frac{\partial a_k}{\partial net_k}\right)}_{\delta_k}.\underbrace{\left(\frac{\partial net_k}{\partial\omega_{jk}}\right)}_{a_k} + \beta.\Delta\omega_{jk}(t-1) \tag{8}$$

where α: Learning Rate, net: Node Input, a: Node Output, β: Momentum Constant, j: Node at the Hidden Layer, k: Node at the Output Layer, t: the previous value of the weight

$\left(\frac{\partial E}{\partial a_k}\right)$: error change with respect to the output node

$\left(\frac{\partial a_k}{\partial net_k}\right)$: partial derivative of the output node with respect to the input

$\left(\frac{\partial net_k}{\partial\omega_{jk}}\right)$: change of the input with respect to the weights

$\Delta\omega_{jk}(t-1)$: is the previous value of the $\Delta\omega_{jk}$

In order to calculate $\Delta\omega_{jk}$, Table 1 shows all the possible combinations of the Activation and Error function applied on the output layer

Table 1. Matrix of Equations

		δ_k				
Learning Rate α	X	**Error Function** $\left(\frac{\partial E}{\partial a_k}\right) =$	X	**Activation Function** $\left(\frac{\partial a_k}{\partial net_k}\right) =$	X	$\left(\frac{\partial net_k}{\partial \omega_{kj}}\right) = a_k$
		1. MSE: $(t_k - a_k)$		**1. Sigmoid:** $a_k.(1 - a_k)$		
		2. Huber: $\begin{cases} (t_k - a_k)\ , & \lvert t - a \rvert < cx < 0 \\ c.\frac{(t_k - a_k)}{\lvert t_k - a \rvert}\ , & \lvert t - a \rvert \geq cx \geq 0 \end{cases}$		**2. Tanh:** $1 - a_k^2$		
		3. Sinh: $(t_k - a_k).\cosh(\lvert(t_k - a_k)\rvert) / \lvert(t_k - a_k)\rvert$		**3. Gaussian:** $-2.a_k.c^2.e^{-c^2.a_k^2}$		

2.4 Input-to-Hidden Layer Weight Update

$$\omega_{ij} = \omega_{ij} + \Delta\omega_{ij} \qquad (9)$$

$$\Delta\omega_{ij} = -\alpha.\delta_j.a_i + \beta.\Delta\omega_{ij}(t-1) \qquad (10)$$

$$\Delta\omega_{ij} = -\frac{\partial E}{\partial \omega_{ij}} = -\alpha.\left(\frac{\partial E}{\partial a_j}\right).\left(\frac{\partial a_j}{\partial net_j}\right).\left(\frac{\partial net_j}{\partial \omega_{ij}}\right) + \beta.\Delta\omega_{ij}(t-1)$$

$$\Delta\omega_{ij} = -\frac{\partial E}{\partial \omega_{ij}} = -\alpha.\underbrace{\sum\left(\underbrace{\left(\frac{\partial E}{\partial a_k}\right).\left(\frac{\partial a_k}{\partial net_k}\right)}_{\delta_k}.\underbrace{\left(\frac{\partial net_k}{\partial a_j}\right)}_{\omega_{jk}}\right).\left(\frac{\partial a_j}{\partial net_j}\right)}_{\delta_j}.\underbrace{\left(\frac{\partial net_j}{\partial \omega_{ij}}\right)}_{a_i} + \beta.\Delta\omega_{ij}(t-1) \qquad (11)$$

$\left(\frac{\partial a_j}{\partial net_j}\right)$: derivative of the activation function at the Hidden Layer

2.5 The Australian Credit Data Set

The Australian Credit Data Set [18] is available online through the UCI Machine Learning Repository. The data set is composed of 14 attributes out of which 6 are numerical and 8 are categorical. In addition, one binary attribute is used for classification purposes. The data set has 690 instances of creditworthy applicants in which 307 are classified as good, and 383 as bad. The data set is divided into 3 subsets, the Training Set (60%), the Generalization Set (20%), and the Validation Set (20%). The data set has been normalized to value between 0 and 1 by finding the maximum value of each attribute and dividing it by each value of the 690 instances of the same attribute.

Table 2. Part of the normalized Australian Credit Data Set

Attribute	Data Type	Instance 1	Instance 2	Instance 3	Maximum Value
A1	Binary	1	0	0	1
A2	Continuous	22.08	22.67	29.58	80.25
A3	Continuous	11.46	7	1.75	28
A4	Categorical	2	2	1	3
A5	Categorical	4	8	4	14
A6	Categorical	4	4	4	9
A7	Continuous	1.585	0.165	1.25	28.5
A8	Binary	0	0	0	1
A9	Binary	0	0	0	1
A10	Continuous	0	0	0	67
A11	Binary	1	0	1	1
A12	Categorical	2	2	2	3
A13	Continuous	100	160	280	2000
A14	Continuous	1213	1	1	100001
A15	Binary	0	0	0	1

2.6 Simulations and Results

Simulations have taken into consideration a mixture of activation functions applied on the hidden and output layers of the neural network. In addition, different error functions have been utilized to determine the speed of convergence of the network weights.

A C^{++} simulator has been designed to test the credit scoring model. The simulator has the following components:

- Data Reader Component which loads a comma delimited file and creates three data set; the training set, the generalization set, and the testing set
- Training Component which forwards all the weight to the output layer
- Back Propagation Component which adjusts the weights based on the error encountered

The simulations have been run using a Neural Network with fixed numbers of 10 hidden nodes and momentum of 0.2. The learning rate has discrete values at 0.01, 0.03, 0.05, 0.1, 0.3, 0.5, 0.7, and 0.9. The number of epochs under consideration has values of 1000, 1500, and 2500. Each iteration has been simulated 40 times and the average values have been recorded.

Figure 4 is a snapshot of the Neural Network model. The simulator starts with randomly initializing the network weights to calculate the output. Each iteration generates outputs such as the epoch number, the training accuracy percentage, the error rate, the generalization accuracy percentage, its error rate, and finally the validation accuracy percentage and its corresponding error rate.

The following figures show the validation accuracy and the error rate with respect to the number of epochs and the learning rate for different combination of activation and error functions. The decision of selecting which network structure is ideal depends on:

- Minimizing the number of epochs to reduce processing time
- Achieving the highest classification accuracy

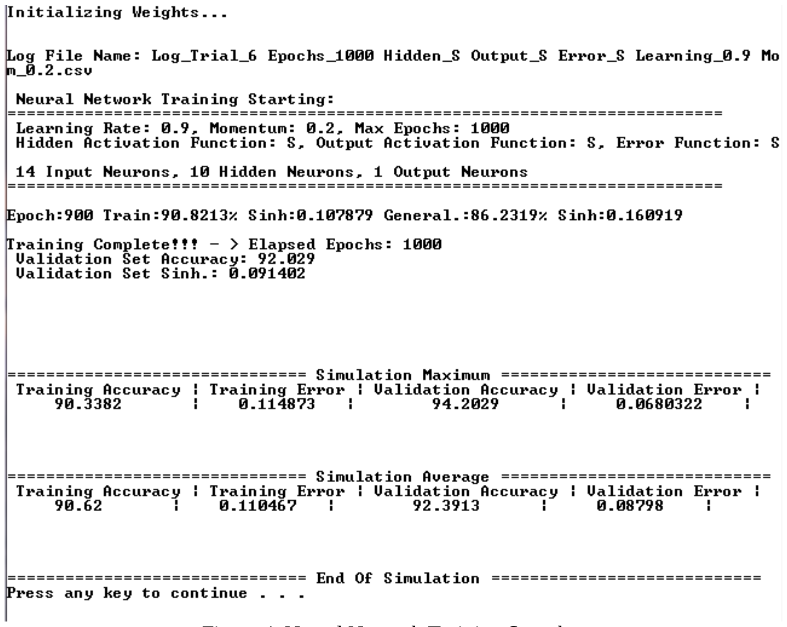
Figure 5 shows the results of applying the Sigmoid Activation Function at the Hidden and Output Layers using three error functions.

Figure 6 shows the results of applying the Tanh Activation Function at the Hidden Layer and Sigmoid Function at the Output's using three error functions.

Figure 7 shows the results of applying the Tanh Activation Function at the Hidden and Output Layers using three error functions.

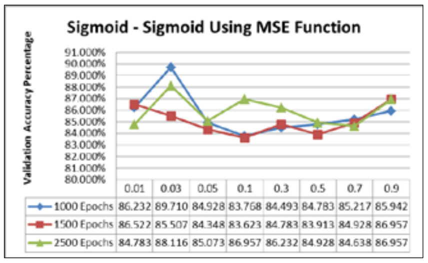
Figure 8 shows the results of applying the Gaussian Activation Function Applied at the Hidden Layer and Sigmoid Function at the Output's using three error functions.

Figure 9 shows the results of applying the Gaussian Activation Function at the Hidden and Output Layers using three error functions.



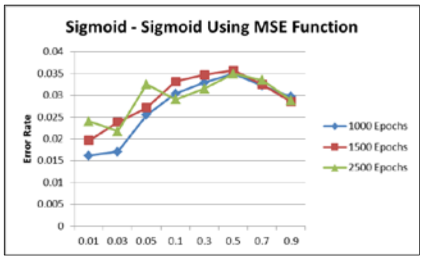
```
Initializing Weights...

Log File Name: Log_Trial_6 Epochs_1000 Hidden_S Output_S Error_S Learning_0.9 Mo
m_0.2.csv

 Neural Network Training Starting:
======================================================================
 Learning Rate: 0.9, Momentum: 0.2, Max Epochs: 1000
 Hidden Activation Function: S, Output Activation Function: S, Error Function: S

 14 Input Neurons, 10 Hidden Neurons, 1 Output Neurons
======================================================================

Epoch:900 Train:90.8213% Sinh:0.107879 General.:86.2319% Sinh:0.160919

Training Complete!!! - > Elapsed Epochs: 1000
 Validation Set Accuracy: 92.029
 Validation Set Sinh.: 0.091402




============================== Simulation Maximum ============================
 Training Accuracy | Training Error | Validation Accuracy | Validation Error |
     90.3382       |    0.114873    |        94.2029       |     0.0680322    |



============================== Simulation Average ============================
 Training Accuracy | Training Error | Validation Accuracy | Validation Error |
     90.62        |    0.110467    |        92.3913       |     0.08798    |



============================== End Of Simulation ============================
Press any key to continue . . .
```

Figure 4. Neural Network Training Snapshot

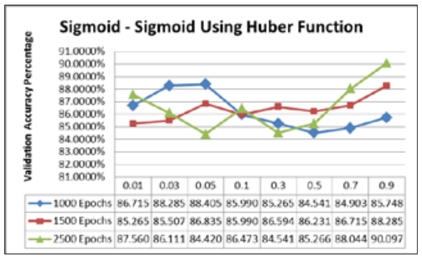


	0.01	0.03	0.05	0.1	0.3	0.5	0.7	0.9
1000 Epochs	86.232	89.710	84.928	83.768	84.493	84.783	85.217	85.942
1500 Epochs	86.522	85.507	84.348	83.623	84.783	83.913	84.928	86.957
2500 Epochs	84.783	88.116	85.073	86.957	86.232	84.928	84.638	86.957

(a)

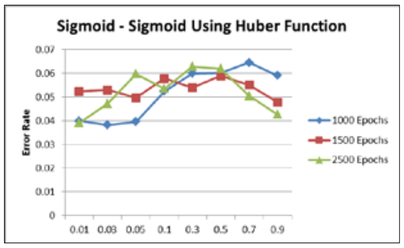


(b)

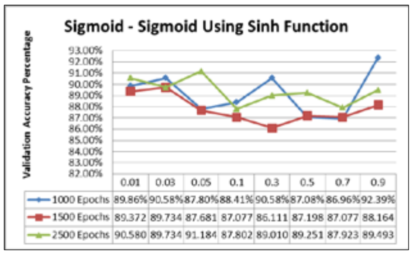


	0.01	0.03	0.05	0.1	0.3	0.5	0.7	0.9
1000 Epochs	86.715	88.285	88.405	85.990	85.265	84.541	84.903	85.748
1500 Epochs	85.265	85.507	86.835	85.990	86.594	86.231	86.715	88.285
2500 Epochs	87.560	86.111	84.420	86.473	84.541	85.266	88.044	90.097

(c)

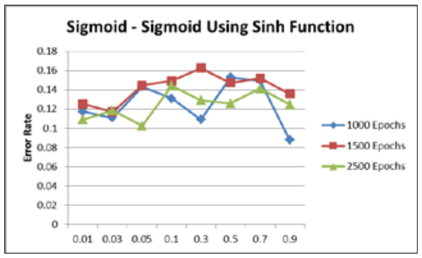


(d)



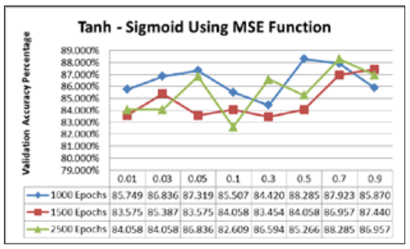
	0.01	0.03	0.05	0.1	0.3	0.5	0.7	0.9
1000 Epochs	89.86%	90.58%	87.80%	88.41%	90.58%	87.08%	86.96%	92.39%
1500 Epochs	89.372	89.734	87.681	87.077	86.111	87.198	87.077	88.164
2500 Epochs	90.580	89.734	91.184	87.802	89.010	89.251	87.923	89.493

(e)



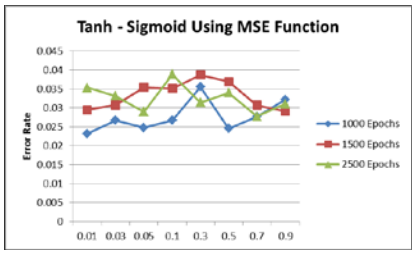
(f)

Figure 5. Sigmoid Activation Function Applied at the Hidden and Output Layers

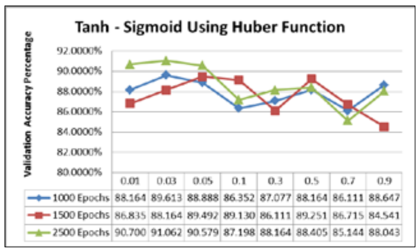


	0.01	0.03	0.05	0.1	0.3	0.5	0.7	0.9
1000 Epochs	85.749	86.836	87.319	85.507	84.420	88.285	87.923	85.870
1500 Epochs	83.575	85.387	83.575	84.058	83.454	84.058	86.957	87.440
2500 Epochs	84.058	84.058	86.836	82.609	86.594	85.266	88.285	86.957

(a)

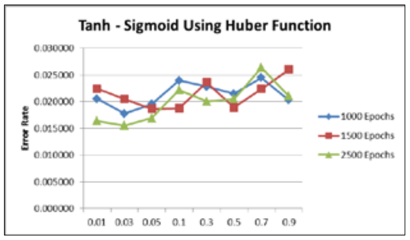


(b)

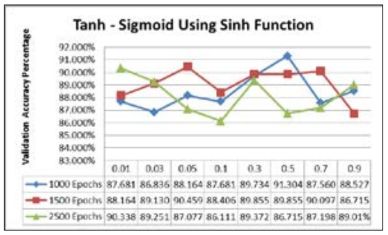


	0.01	0.03	0.05	0.1	0.3	0.5	0.7	0.9
1000 Epochs	88.164	89.613	88.888	86.352	87.077	88.164	86.111	88.647
1500 Epochs	86.835	88.164	89.492	89.130	86.111	89.251	86.715	84.541
2500 Epochs	90.700	91.062	90.579	87.198	88.164	88.405	85.144	88.043

(c)

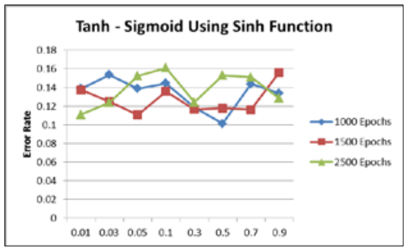


(d)



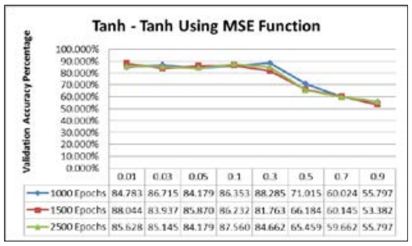
	0.01	0.03	0.05	0.1	0.3	0.5	0.7	0.9
1000 Epochs	87.681	86.836	88.164	87.681	89.734	91.304	87.560	88.527
1500 Epochs	88.164	89.130	90.459	88.406	89.855	89.855	90.097	86.715
2500 Epochs	90.338	89.251	87.077	86.111	89.372	86.715	87.198	89.01%

(e)



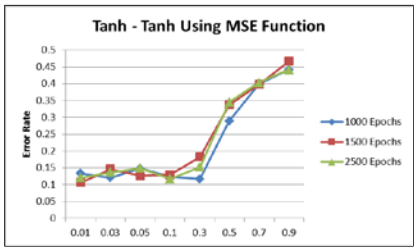
(f)

Figure 6. Tanh Activation Function Applied at the Hidden Layer and Sigmoid Function at the Output's

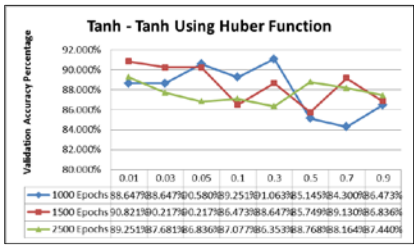


	0.01	0.03	0.05	0.1	0.3	0.5	0.7	0.9
1000 Epochs	84.783	86.715	84.179	86.353	88.285	71.015	60.024	55.797
1500 Epochs	88.044	83.937	85.870	86.232	81.763	66.184	60.145	53.382
2500 Epochs	85.628	85.145	84.179	87.560	84.662	65.459	59.662	55.797

(a)

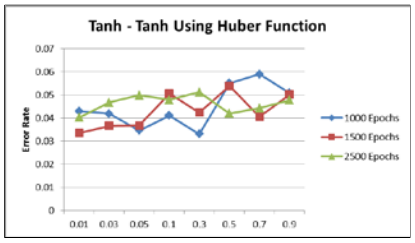


(b)

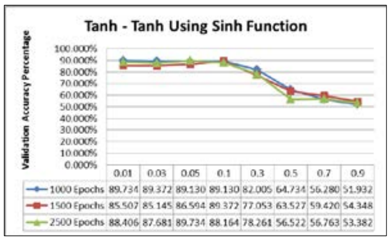


	0.01	0.03	0.05	0.1	0.3	0.5	0.7	0.9
1000 Epochs	88.647%	88.647%	90.580%	89.251%	91.063%	85.145%	84.300%	86.473%
1500 Epochs	90.821%	90.217%	90.217%	86.473%	88.647%	85.749%	89.130%	86.836%
2500 Epochs	89.251%	87.681%	86.836%	87.077%	86.353%	88.768%	88.164%	87.440%

(c)

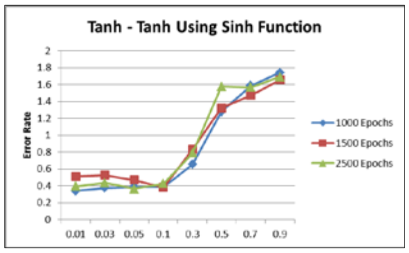


(d)



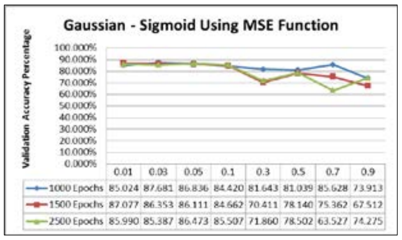
	0.01	0.03	0.05	0.1	0.3	0.5	0.7	0.9
1000 Epochs	89.734	89.372	89.130	89.130	82.005	64.734	56.280	51.932
1500 Epochs	85.507	85.145	86.594	89.372	77.053	63.527	59.420	54.348
2500 Epochs	88.406	87.681	89.734	88.164	78.261	56.522	56.763	53.382

(e)



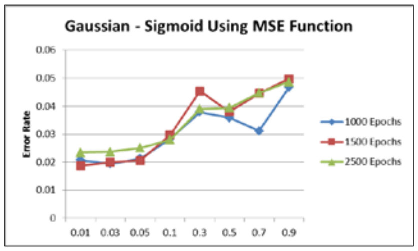
(f)

Figure 7. Tanh Activation Function Applied at the Hidden and Output Layers

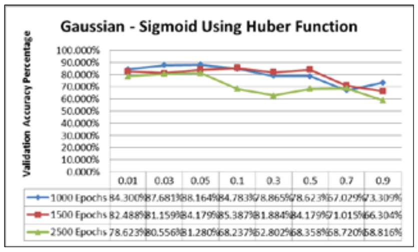


	0.01	0.03	0.05	0.1	0.3	0.5	0.7	0.9
1000 Epochs	85.024	87.681	86.836	84.420	81.643	81.039	85.628	73.913
1500 Epochs	87.077	86.353	86.111	84.662	70.411	78.140	75.362	67.512
2500 Epochs	85.990	85.387	86.473	85.507	71.860	78.502	63.527	74.275

(a)

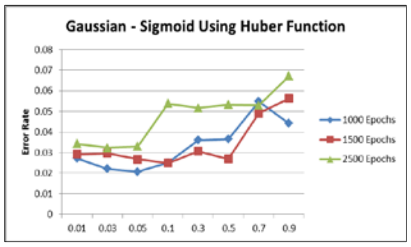


(b)

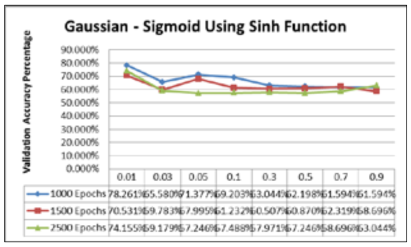


	0.01	0.03	0.05	0.1	0.3	0.5	0.7	0.9
1000 Epochs	84.300%	87.681%	88.164%	84.783%	78.865%	78.623%	67.029%	73.309%
1500 Epochs	82.488%	81.159%	84.179%	85.387%	81.884%	84.179%	71.015%	66.304%
2500 Epochs	78.623%	80.556%	81.280%	68.237%	62.802%	68.358%	68.720%	58.816%

(c)

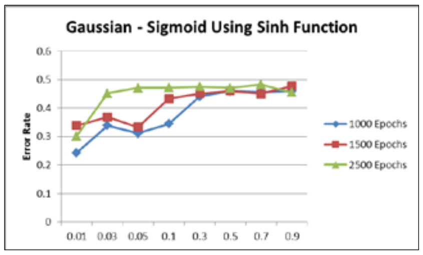


(d)



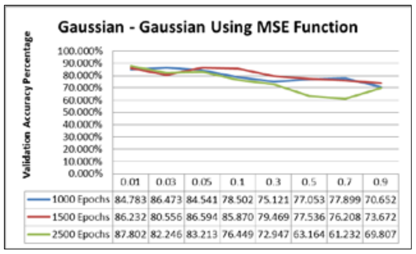
	0.01	0.03	0.05	0.1	0.3	0.5	0.7	0.9
1000 Epochs	78.261%	65.580%	71.377%	69.203%	63.044%	62.198%	61.594%	61.594%
1500 Epochs	70.531%	59.783%	67.995%	61.232%	60.507%	60.870%	62.319%	58.696%
2500 Epochs	74.155%	59.179%	57.246%	57.488%	57.971%	57.246%	58.696%	63.044%

(e)



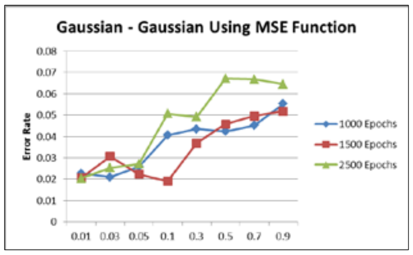
(f)

Figure 8. Gaussian Activation Function Applied at the Hidden Layer and Sigmoid Function at the Output’s

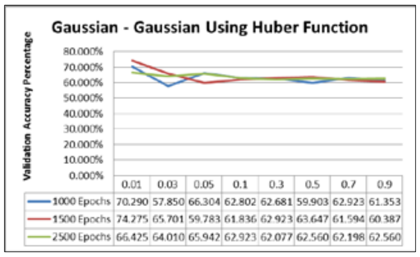


	0.01	0.03	0.05	0.1	0.3	0.5	0.7	0.9
1000 Epochs	84.783	86.473	84.541	78.502	75.121	77.053	77.899	70.652
1500 Epochs	86.232	80.556	86.594	85.870	79.469	77.536	76.208	73.672
2500 Epochs	87.802	82.246	83.213	76.449	72.947	63.164	61.232	69.807

(a)

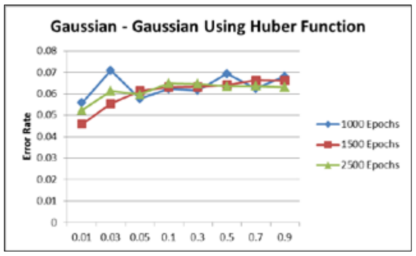


(b)

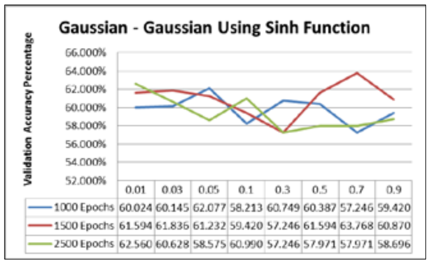


	0.01	0.03	0.05	0.1	0.3	0.5	0.7	0.9
1000 Epochs	70.290	57.850	66.304	62.802	62.681	59.903	62.923	61.353
1500 Epochs	74.275	65.701	59.783	61.836	62.923	63.647	61.594	60.387
2500 Epochs	66.425	64.010	65.942	62.923	62.077	62.560	62.198	62.560

(c)

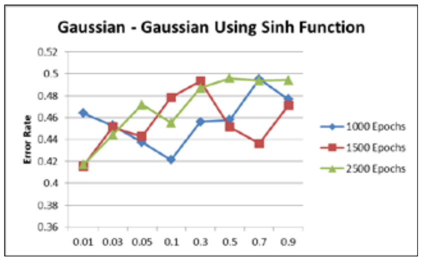


(d)



	0.01	0.03	0.05	0.1	0.3	0.5	0.7	0.9
1000 Epochs	60.024	60.145	62.077	58.213	60.749	60.387	57.246	59.420
1500 Epochs	61.594	61.836	61.232	59.420	57.246	61.594	63.768	60.870
2500 Epochs	62.560	60.628	58.575	60.990	57.246	57.971	57.971	58.696

(e)



(f)

Figure 9. Gaussian Activation Function Applied at the Hidden and Output Layers

2.7 Conclusion

In this paper, three conventional monotonic and differentiable activation and error functions are under study. These popular activation functions are Sigmoid, Hyperbolic Tangent, and Gaussian Symmetric. Functions such as Mean Squared, Huber, and Complex Hyperbolic Sine are the error functions used at the neural network output layer.

The paper demonstrates the importance of selecting proper activation and error functions in neural networks. In addition, learning rate, momentum, and the training-to-validation data set ratio are vital factors to achieve accurate scoring results.

Tables 3, 4, and 5 summarize the results of the above charts by listing the highest accuracy percentage and the error rate per network structure.

Table 3. MSE Function

Activation Function	Epoch	Learning Rate	Accuracy	Error
Sigmoid – Sigmoid	1000	0.03	89.710%	0.0170831
Tanh – Sigmoid	1000	0.5	88.285%	0.0245681
Tanh - Tanh	1500	0.01	88.044%	0.106577
Gaussian - Sigmoid	1000	0.03	87.681%	0.0194372
Gaussian - Gaussian	2500	0.01	87.802%	0.0204707

Table 4. Huber Error Function

Activation Function	Epoch	Learning Rate	Accuracy	Error
Sigmoid – Sigmoid	1000	0.03	88.2850%	0.0382455
Tanh – Sigmoid	2500	0.03	91.0628%	0.015508
Tanh - Tanh	1000	0.3	91.0628%	0.0331084
Gaussian - Sigmoid	1000	0.05	88.164%	0.0206397
Gaussian - Gaussian	1500	0.01	74.275%	0.045943

Table 5. Sinh Error Function

Activation Function	Epoch	Learning Rate	Accuracy	Error
Sigmoid – Sigmoid	1000	0.9	92.39%	0.08798
Tanh – Sigmoid	1000	0.5	91.304%	0.101463
Tanh - Tanh	1000	0.01	89.734%	0.341206
Gaussian - Sigmoid	1000	0.01	78.261%	0.242955
Gaussian - Gaussian	1500	0.7	63.768%	0.436494

It has been observed that the nature of the error function plays a significant role in selecting the appropriate activation functions. Table 4 shows that when applying Huber Error function, both (Tanh-Sigmoid) and (Tanh-Tanh) produce high accuracy percentages. Similarly, Table 5 demonstrates that when applying Sinh error function, (Sigmoid – Sigmoid) and (Tanh – Sigmoid) generate accurate scoring results.

In other words, experimental results demonstrate that the neural network computed satisfactory results when Sigmoid – Sigmoid - Sinh Combination of activation and error function is used for hidden and output layers.

When properly and sufficiently trained, applying appropriate activation and error functions, Neural Network performs remarkably better than any other statistical approach, such as logistic regression or discriminant analysis.

References

[1] Bache, K. & Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[2] Biçer, I., Sevis, D., & Bilgiç, T. (2010). Bayesian credit Scoring Model with Integration of Expert Knowledge and Customer Data. In *International Conference 24th ini EURO Conference "Continuous Optimization and Information Technologies in the FFinancial Sector"(MEC EurOPT 2010)* (pp. 324-329).

[3] Chen, F. L., & Li, F. C. (2010). Comparison of the hybrid credit scoring models based on various classifiers. *International Journal of Intelligent Information Technologies (IJIIT)*, *6*(3), 56-74.

[4] Chuang, C. L., & Huang, S. T. (2011). A hybrid neural network approach for credit scoring. *Expert Systems*, *28*(2), 185-196.

[5] Gangal, A. S., Kalra, P. K., & Chauhan, D. S. (2007). Performance evaluation of complex valued neural networks using various error functions. *Enformatika*,*23*, 27-32.

[6] Gao, L., Zhou, C., Gao, H. B., & Shi, Y. R. (2006). Credit scoring model based on neural network with particle swarm optimization. In *Advances in Natural Computation* (pp. 76-79). Springer Berlin Heidelberg.

[7] Heiat, A. (2011). Modeling Consumer Credit Scoring Through Bayes Network.*World*, *1*(3), 132-141.

[8] Hsieh, N. C., & Hung, L. P. (2010). A data driven ensemble classifier for credit scoring analysis. *Expert Systems with Applications*, *37*(1), 534-545.

[9] Hu, Y. C., & Ansell, J. (2007). Measuring retail company performance using credit scoring techniques. *European Journal of Operational Research*, *183*(3), 1595-1606.

[10] Karlik, B., & Olgac, A. V. (2011). Performance analysis of various activation functions in generalized MLP architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, *1*(4), 111-122.

[11] Khashman, A. (2009). A neural network model for credit risk evaluation.*International Journal of Neural Systems*, *19*(04), 285-294.

[12] Lee, T. S., & Chen, I. F. (2005). A two-stage hybrid credit scoring model using artificial neural networks and multivariate adaptive regression splines. *Expert Systems with Applications*, *28*(4), 743-752..

[13] Marcano-Cedeno, A., Marin-De-La-Barcena, A., Jimenez-Trillo, J., Pinuela, J. A., & Andina, D. (2011). Artificial metaplasticity neural network applied to credit scoring. *International journal of neural systems*, *21*(04), 311-317.

[14] Siami, M., Gholamian, M. R., & Nasiri, R. (2011). A Hybrid mining model based on Artificial Neural Networks, support Vector Machine and Bayesian for credit scoring. *5th Symposium on Advances in Science & Technology, 12-17 May 2011*. Iran:Mashhad

[15] SIBI, P., JONES, S. A., & SIDDARTH, P. (2013). ANALYSIS OF DIFFERENT ACTIVATION FUNCTIONS USING BACK PROPAGATION NEURAL NETWORKS. *Journal of Theoretical and Applied Information Technology*, *47*(3).

[16] Setiono, R., Baesens, B., & Mues, C. (2011). Rule extraction from minimal neural networks for credit card screening. *International Journal of Neural Systems*, *21*(04), 265-276.

[17] Tong, L. I., Yang, C. H., & Yu, H. P. Credit rating analysis using general regression neural network.

[18] Tsai, C. F. (2008). Financial decision support using neural networks and support vector machines. *Expert Systems*, *25*(4), 380-393.

[19] Wu, W. W. (2011). Improving classification accuracy and causal knowledge for better credit decisions. *International Journal of Neural Systems*, *21*(04), 297-309.

[20] Zhou, L., Lai, K. K., & Yen, J. (2009). Credit scoring models with AUC maximization based on weighted SVM. *International journal of information technology & decision making*, *8*(04), 677-696.