ISSN 2320 - 2602
Volume 2, No.11, November 2013
International Journal of Advances in Computer Science and Technology
Available Online at http://warse.org/pdfs/2013/ijacst032112013.pdf

Agent-Based Automated Negotiation System for Handling Vague Data Using Artificial Neural Network with Adaptive Back Propagation Algorithm

Samuel King Opoku
Computer Science Department,
Kumasi Polytechnic, Ghana.
Samuelk.opoku@kpoly.edu.gh, Samuel.k.opoku@gmail.com

ABSTRACT

An auction market is a market in which buyers enter competitive bids and sellers enter competitive offers at the same time. Many negotiation systems have been proposed that rely on adequate and precise information provided by the negotiation parties for making their decisions. This paper provides mechanisms for addressing inadequate or missing information in a negotiation environment between a seller and many buyers which depends on multiple issues to make a decision. The system employs artificial neural network with adaptive momentum back propagation mechanism to determine whether a buyer should be selected among possible bid winners. The system then uses simple decision controls to determine the overall bid winner. The system was implemented using Java

Key words: Artificial Neural Network, Auction Market, Automated Negotiation, Back Propagation, E-Agent System, E-commerce System, Intelligent Negotiation System.

1. INTRODUCTION

An auction market is a market in which buyers enter competitive bids and sellers enter competitive offers at the same time. A seller lists an item at auction and potential buyers bid on the auction through negotiation [1]. Negotiation is therefore a means for agents to communicate and compromise for reaching mutually beneficial agreements. It is an important conflict management technique by which a joint decision is made by two or more parties [1], [2]. Two basic components are important when designing an automated negotiation system: the negotiation protocol and the negotiation process model [3]-[5]. The negotiation protocol is a set of rules which governs software processing, distributed decision making and communication tasks, It imposes constraints on activities through the specification of permissible inputs, assumptions and actions [4]-[6]. Negotiation process concerns itself with the issues over which agreement must be reached and the models which are employed to act in line with the negotiation protocol in order to achieve the negotiation objectives [7].

In a team negotiation system, three different elements have to be specified. These are negotiation protocol with the opponent, the negotiation strategy used by the opponent and the intra-team negotiation strategy followed by team members in order to decide the actions to perform during the negotiation process [8]. A typical type of team negotiation system is one-to-many negotiation in which one agent can adopt different negotiation strategies with different trading partners [9], [10]. A problem with the general one-to-many negotiation mechanism is that during negotiation, no matter how long an agent has to wait and how many proposals have been received, the agent cannot propose until it has received proposals from all its trading partners [10]-[12]. In actual negotiation environments, agents may have different negotiation strategies, reasoning mechanisms, preferences, constraints and communication time which affects systems flexibility and operations [12], [13].

Many negotiation systems have been proposed. A flexible mechanism for one-to-many negotiation agent which focused on single-issue or single-attribute negotiation was proposed in [14]. In [15], a model was designed that sought to make offers semi-autonomously and were based on human negotiation. Regarding the model for evaluating offers, in [16]-[18], models were designed using simple additive utility scheme. Other designs employed in [19] and [20] were based on game theory or mathematical programming approaches. A proposed algorithm to learn about one's negotiation counterpart and then use that knowledge to obtain a better outcome was designed in [21] for modulating the course of the negotiation on the receipt of additional information. Other such algorithms as Bayesian learning with subjective probability [22] and machine learning approach [23] had also been proposed. These algorithms require historical data and can therefore be a disadvantage when parties are new to each other in the negotiation environment [24]. The accuracy of these learning mechanisms can be improved through nonlinear optimization techniques [23]-[25]

This paper provides mechanisms for addressing inadequate or missing information in a negotiation environment between a seller and many buyers which depends on multiple issues to make a decision. The system employs artificial neural network with adaptive momentum back propagation mechanisms to determine whether a buyer should be selected among possible bid winners. The system was implemented using Java

An Artificial Neural Network (ANN) is a mathematical model that tries to simulate the structure and functionalities of biological neural networks [26]. The basic building block of every ANN is the artificial neuron. Artificial neuron is a simple mathematical model (function) and it has three simple sets of rules: multiplication, summation and activation [27]. At the entrance of an artificial neuron, the inputs are weighted by multiplying them by values called weights. The middle section of the artificial neuron is the sum function that works on all the weighted inputs and a bias. A bias is a neuron that has an output value of 1. The output value is used as inputs to other neurons in the network. At the exit of the artificial neuron, the sum of previously weighted inputs and the bias is passed through an activation function (also called transfer function) [26], [27].

Figure 1 illustrates the working principle of artificial neuron network

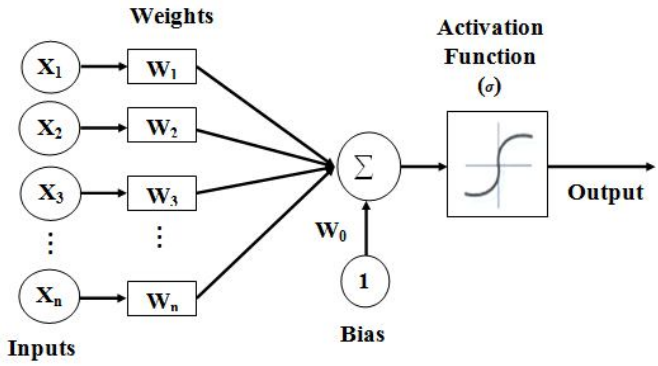

Figure 1: Working Principle of an Artificial Neuron

The following mathematical expression is obtained from Figure 1 above: $\boldsymbol{Y(k) = \sigma\left(\sum_{i=0}^{n-1} W_i(k).X_i(k) + b \right)}$ where

- $X_i(k)$ is input in discrete time k where $\forall i,\ 0 \leq i < n$
- $W_i(k)$ is weight in discrete time k where $\forall i, 0 \leq i < n$
- b is the bias,
- σ is a transfer function,
- Y(k) is output value in discrete time k.

The Transfer function defines the properties of artificial neuron and can be any mathematical function [27]. It is chosen on the basis of the problem that artificial neural network needs to solve. A typical set of transfer functions include [27]-[29]:

- **Step Function**: It is a binary function that has only two possible output values (e.g. 0 and 1). This means if input value meets specific threshold the output value results in 1 and if specific threshold is not met, the result is 0. When this type of transfer function is used in artificial neuron, the artificial neuron is called perceptron. Perceptron is used for solving classification problems. Perceptron is usually found in the output layer of an artificial neural networks

- **Linear Function**: The artificial neural network simply performs linear transformation over the sum of weighted inputs and bias. Such an artificial neuron is commonly used in the input layer of artificial neural networks

- **Non-linear Functions**: The most common non-linear function is the sigmoid function. It is a continuous differentiable function making it easy to calculate derivates for weight updates. The major limitation of the sigmoid function, in some circumstances, is the occurrence of local minima in error function.

In practice, single artificial neuron has almost no usefulness in solving real-life problems [30] but a network of artificial neurons arranged in layers is capable of solving complex real-life problems in a non-linear, distributed, parallel and local way [27], [30]. In a multilayer feed-forward neural network, information flows from inputs to outputs in only one direction [27]. Figure 2 depicts the architecture of a feed-forward neural network

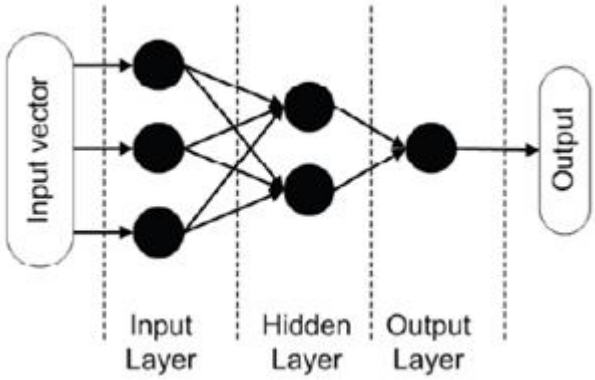

Figure 2: Architecture of a Feed-forward Neural Network

ANN solves problems through learning. A learning algorithm is an adaptive method by which a network of computing units self-organizes to implement the desired behavior by presenting some examples of the desired input output mapping to the network [27]. A correction step is

executed iteratively until the network learns to produce the desired response. The learning algorithm can either be supervised or unsupervised [30]. Unsupervised learning is used when, for a given input, the exact numerical output a network should produce is unknown. The network must organize itself in order to be able to associate clusters with units [26], [27].

Supervised learning denotes a method in which some input vectors are collected and presented to the network. The output computed by the network is observed and the deviation from the expected answer is measured. The weights are corrected according to the magnitude of the error in the way defined by the learning algorithm. Supervised learning is further divided into methods which use reinforcement or error correction [27]. Reinforcement learning is used when after each presentation of an input-output example the network is determined whether it produced the desired result or not. Thus the output of reinforcement learning is either true or false [26], [30]. With error correction learning, the weights are corrected using the error and the input vector. This aims at minimizing or eliminating the error in a single correction step [26], [27], [30].

Back propagation training algorithm is an error-correction learning rule [31], [32]. The algorithm consists of two passes through the different layers of the network, mainly, a forward pass and a backward pass. In the forward pass, input vector is applied to the network. The weighted input vector is propagated through the network from one layer to another [33]. Finally, a set of outputs is produced as the actual response of the network. During the forward pass the synaptic weights of the network are all fixed.

The backward pass is used to adjust the values of the weights so that the error is reduced at each step in accordance with the error-correction rule. The actual response of the network is subtracted from a desired target response to produce an error signal. This error signal is then propagated backward through the network. The weights are adjusted so as to make the actual response of the network move closer to the desired response [29], [33].

2. SYSTEM PROTOCOL DESIGN

This system implements one-to-many negotiation mechanism such that one seller auctions a product and many buyers compete for the product through bidding. Each buyer initiates a thread in the main system. The buyers provide their particulars and decide the amount they are willing to pay.

Figure 3 shows a general overview of the negotiation protocol:

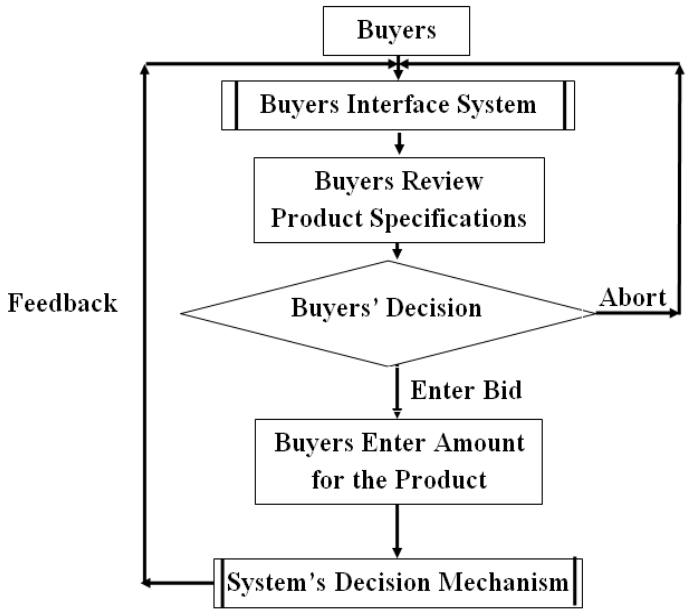

Figure 3: Overview of Negotiation Protocol

The following are the assumptions employed in the implementation of the system:

- Every data required by the system can be normalized into 0's and 1's
- There is only one product of a particular specification for which many buyers enter into competition for it.
- Only one of the winning bidders is selected as a winner.

The issues involved in this work are the brand of the product the buyer requires, the mailing address the buyer wants to receive the product when s/he is announced the winner, the level of complexity of the product the buyer desires and the amount the buyer is willing to pay for the product. The decision requires all the four parameters. The System Decision mechanism is the reasoning model of the negotiation system.

3. SYSTEM ARCHITECTURE

The system is divided into four main sections, namely, Buyers' interface, Normalizer, Neural Network Predictor and Decider. The Buyers' interface is the first interface encountered by potential buyers. The Normalizer converts the users' input into 0's and 1's that can be used by the Neural Network Predictor. Based on the weight values previously obtained through training, the Neural Network Predictor computes whether the buyer should be accepted as a potential bidder or be rejected. The Decider upon the information obtained from the Neural Network Predictor chooses the winner. Figure 4 illustrates how the various sections are inter-related:

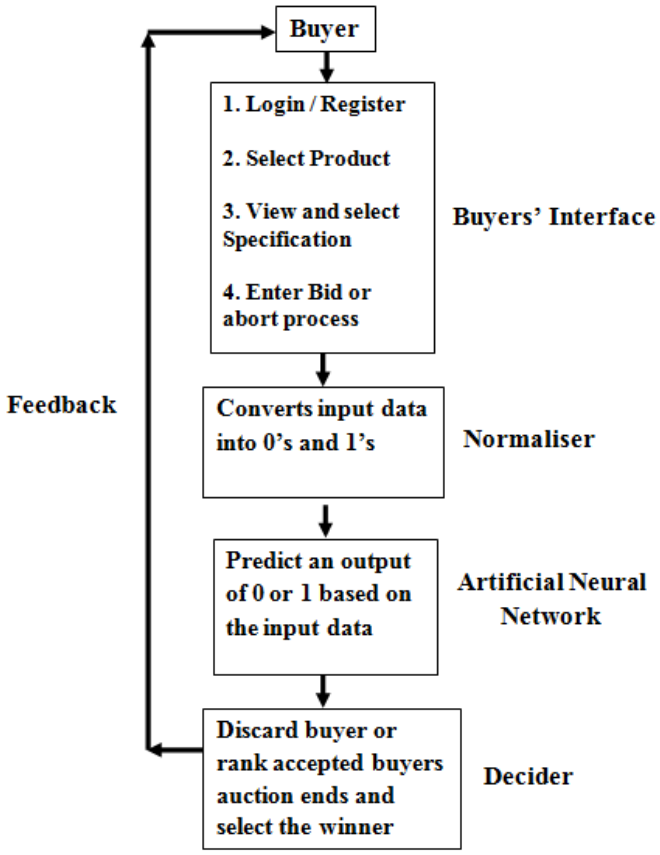

Figure 4: Overview of System Architecture

The feedback in the figure above represents two situations:

- In the first situation, a buyer receives "Rejection" or "Conditionally accepted". If the prediction generated by the artificial neural network which is passed to the Decider is 0, the buyer is rejected otherwise the buyer is conditionally accepted
- In the second situation, all the conditionally accepted buyers are ranked and the first buyer is selected as the winner of the auction

3.1 Buyers' Interface Architecture

This architecture is the only architecture with a graphical user interface implementation. The architecture allows potential bidders to interact with the negotiation system. Buyers are expected to login successfully before conducting any transaction. New buyers are given the opportunity to create account by providing their personal information which will be used later by the system. The information entered by the buyer is kept in a database.

The following algorithm is used to capture buyers' information:

```
00 For each of the input controls
01    If (control is sensitive and buyer ignores data entry) then
02       Prompt the buyer
03    Else if (control is less sensitive and buyer ignores) then
04       Assume that the buyer does not know
05       Use the worst case scenario
06    End if
07 Next for
```

Sensitive data include full name, user name and password. The user name and password are use to authenticate the user (or bidder). Less sensitive data includes but not limited to date of birth.

After a successful login, users are given another interface where they select the product they want to bid for. They also provide information regarding the product specification and the mailing address if desired. The product specifications include brand and other features which together determine the complexity level of the product. The amount the user is willing to pay is also entered by the user. Users can also abort their bidding process at this stage.

3.2 Normalization Architecture

This architecture links the buyers' architecture and the neural network architecture. It converts the necessary data required for making decision to 0's and 1's based on the threshold set by the negotiation system.

Let

M_i be the input parameters required by the system such that $\forall M_i \in \{\text{Brand}, \text{Mailing Address}, \text{Complexity level}, \text{Amount}\}$ and $\forall i \in \{0,1,2,3\}$

T_i be the threshold set for the input parameters such that $\forall i \in \{0,1,2,3\}$

X_i be the normalized data obtained from the input data such that $\forall X_i \in \{0,1\}$ and $\forall i \in \{0,1,2,3\}$

Thus

$X_i = 1$ when $M_i \geq T_i$ or $M_i \in \{T_i\}$, otherwise $X_i = 0$

Table 1 summarizes the various categories of the input parameters

Table 1: Criteria for Input Normalization

Input Parameters	$X_i = 0$	$X_i = 1$
Brand	New	Used
Mailing Address	Far	Near
Complexity Level	Above normal	Normal
Amount	Small	Normal or above

The threshold values are kept in a database. It is updated periodically to reflect the current market values and the goal

of the auction. The major goal of the negotiation system is to maximize project for suppliers. The normalized data is passed to the trained neural network architecture. When the data for a particular input parameter is not provided or the precise data is not known, the system does not stop abruptly. However, the worst case value is used and the normalized value is usually 0.

3.3 Neural Network Architecture

The neural network is made up of three layers, namely, input layer, hidden layer and output layer. The input layer consists of four neurons with each representing an input parameter. The first neuron represents the brand of the product, the second neuron represents the mailing address, the third neuron represents the complexity level and the fourth neuron represents the bid amount. A fifth neuron is added with a bias value of 1

The hidden layer is made up of one layer with (N+1) neurons. N represents the number of neurons which is determined during neural network training. The $(N+1)^{th}$ neuron is added with a bias value of 1. The output layer consists of only one neuron. Thus there are (N+7) neurons in the neural network architecture with N representing the number of neurons in the hidden layer and two neurons representing bias neurons.

Figure 5 depicts the architecture of the artificial neural network obtained at the end of the training

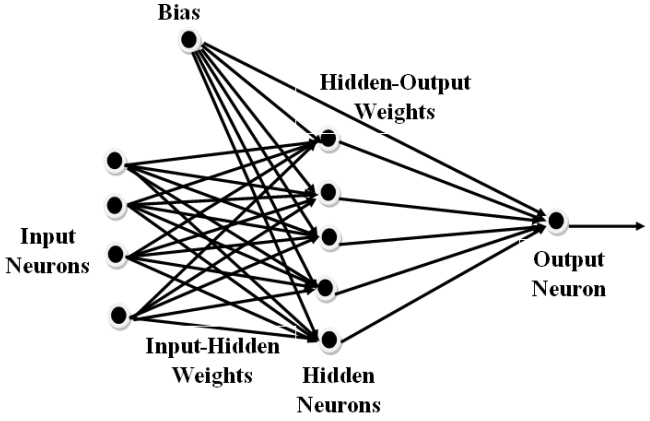

Figure 5: Negotiation Neural Network Architecture

Since all the neurons accept and output either 0 or 1, the sigmoid function was used as an activation function (f) with the formula $f(x) = \frac{1}{1+e^{-x}}$

Let

$X_i, \forall i \in \{0,1,2,3,4\}$ be the values for the neurons in the input layer such that $X_i \in \{0,1\}, \forall i \in \{0,1,2,3\}$ and $X_4 = 1$ representing the bias

$H_j, \forall j \in \{0,1,2,3,\dots,N\}$ be the values of the neurons in the hidden layer such that $H_j \in \{0,1\}, \ \forall j \in \{0,\dots,N-1\}$ and $H_N = 1$ representing the bias

$Y_k, \forall k = 0$ be the value for the neuron in the output layer such that $Y_k \in \{0,1\}$.

$W_{i,j}$ be the weight between the neuron X_i in the input layer and the neuron H_j in the hidden layer such that $\forall W_{i,j}: |W_{i,j}| \leq 1, \forall i \in \{0,1,2,3,4\}$ and $\forall j \in \{0,1,2,\dots,N\}$.

Thus $H_j, \forall j \in \{0,1,2,\dots,N-1\}$ is obtained by the formula

$$H_j = \left\{ \frac{1}{1 + e^{-(\sum_{i=0}^{4}(X_i \cdot W_{i,j}))}} \right\} \in \mathbb{R}^+$$

$W_{j,k}$ be the weight between the neuron H_j in the hidden layer and the neuron Y_k in the output layer such that $\forall W_{j,k}: |W_{j,k}| \leq 1, \forall j \in \{0,1,2,\dots,N\}$ and $\forall k = 0$.

Thus the output value of the neuron in the output layer is obtained by the formula

$$Y_0 = \text{floor}\left\{ \frac{1}{1 + e^{-\left(\sum_{j=0}^{N}(H_j \cdot W_{j,k})\right)}} + 0.5 \right\} \in \mathbb{Z}^+ \in \{0,1\}$$

3.4 Decider Architecture

The Decider communicates with the buyer as to the update of the auction process. The input parameters of the Decider are: the normalized input data, the actual amount, the predicted output of the artificial neural network. When the predicted output of the artificial neural network is 0, the buyer is automatically rejected otherwise the buyer is conditionally accepted.

At the end of the auction, all the conditionally accepted candidates are ranked based on the algorithm described below:

Let

Amt be the actual amount that the buyer entered during the bidding process

$X_i, \forall i \in \{0,1,2,3,4\}$ be the values for the neurons in the input layer such that $X_i \in \{0,1\}, \forall i \in \{0,1,2,3\}$

m denotes the number of conditionally accepted buyers then

$$Rank^m = \frac{Amt^m}{\sum_{i=0}^{3} X_i{}^m} \text{ such that } \forall m > 0, \ Rank^m \in \mathbb{R}^+$$

The buyer with the minimum rank value is the winner since the greater the number of 1's in the normalized data, the more profit the supplier gains. If two or more buyers have the same

rank values then the following conditions are employed to select the winner:

1. The buyer with highest value of $\sum_{i=0}^{3} X_i$
2. If more than one buyer has the highest value of $\sum_{i=0}^{3} X_i$ then the buyer with the highest Amt is selected
3. If no single buyer is selected in the steps above, then the buyer who entered the bid first is selected. Thus First Come, First Serve principle is applied.
4. If multiple winning buyers entered the auction process at the same time and previous steps are not enough to select one winner among them, their names are arranged and sorted in ascending other using their first and last names whiles middle names are ignored. The winner becomes the person whose name comes first.
5. If all the above steps fail to select a single winner, then the user names of the buyers are used. Since the user names are unique to all the users of the negotiation environment, the user names are sorted in ascending order and the winner becomes the person whose user name comes first.

4. IMPLEMENTATION OF THE NETWORK

The network was implemented as software using Java. The network was trained, tested and validated in iterative procedure to finally determine appropriate epoch, number of neurons in the hidden layer, learning rate and initial momentum.

4.1 Training Procedure and Algorithms

Supervised learning method was employed in adjusting the weights. There were only sixteen set of data that can be used to train the network. Given the size of the neural network, the sixteen sets of data produced under-fitting results when used once. Thus in training the network, the whole set of data was used iteratively.

Two mechanisms for adjusting the weights were first employed and then the better mechanism was selected. The mechanisms were batch weight updating mechanism and incremental weight updating mechanism. With batch weight updating mechanism, the weight correction terms were accumulated until the end of a complete iteration (or epoch). A single weight adjustment was made based on the average of the errors obtained in the epoch. It was observed that training usually converged to local minimum

The incremental weight updating mechanism adjusted the weight after each training data set was presented to the network. It was however, realized that the network skewed toward most recent patterns in the cycle. Thus the incremental weight updating mechanism was used in adjusting the weights with the training set arranged judiciously without allowing a particular pattern to follow one another. Table 2 shows the arranged input values and their corresponding target values used in training the network.

Table 2: Training Data Set

Input Values				Target Output
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

The algorithm used in training the network is shown below:

```
00 Assign small positive random values to weights
01 For epochs from 1 to limit assigned by user
02    Read the first set of input data
03    Compute the values for each hidden neurons
04    Apply the activation function on values obtained
05    Compute the value of the output neuron
06    Apply the activation function
07    Compute the error, E= Target – predicted output
08    Use the error to update the weights
09    Read the next set of input data
10    If ((End of data) and (number of epochs reached or
                  |E|<0.00001)) then
11          Terminate training
12    Else
13          Go to Step 03
14    End if
15 Next For
```

4.2 Computational Models

This section describes the various mathematical models that were employed in designing and training the network.

Let

δ_k be the error of the network at the output layer

δ_j be the error of the neuron in the hidden layer

Y'_k be the output of the network during training

Y_k be the target or the expected output of the network

α be the momentum rate

η be the learning rate

θ_j be the output of a neuron in the hidden layer

X_i be the input values of the neurons in the input layer

Then the error of the output neuron is obtained by

$$\delta_k = Y'_k(1 - Y'_k)(Y_k - Y'_k)$$

The term $Y'_k(1 - Y'_k)$ is necessary in the computation because of the sigmoid function.

Updating the weights between the hidden layer neurons and the output neuron in the output layer at any particular period, t is given by:

$$W_{j,k}(t) = W_{j,k} + \eta . \delta_k . \theta_j + \alpha . \Delta W_{j,k}(t-1)$$

Considering the weights between the input layer and the hidden layer, the errors of the neurons in the hidden layer are computed as

$$\delta_j = \theta_j(1 - \theta_j)(\delta_k . W_{j,k})$$

Thus the weights are computed as

$$W_{i,j}(t) = W_{i,j} + \eta . \delta_j . X_i + \alpha . \Delta W_{i,j}(t-1)$$

4.3 Testing and Validating the Neural Network

Figure 6 depicts the interface for entering training parameters

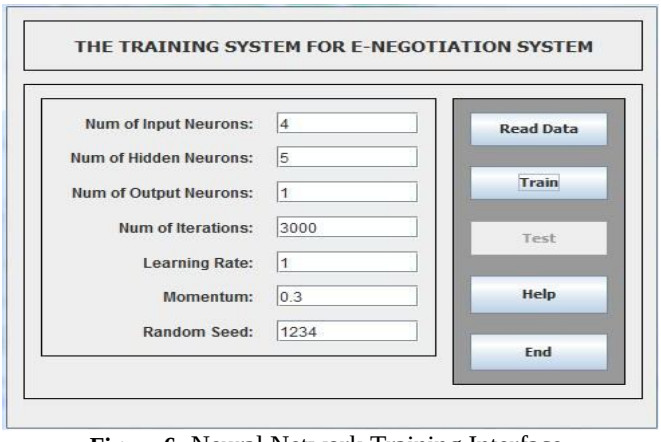

Figure 6: Neural Network Training Interface

When the Read Data button is clicked, an open dialog is displayed that allows the user to select the text file that contains the training set and the target set. The training button is then enabled.

When training is completed, the Test button is enabled. Similarly, when the Test button is clicked, an open dialog is displayed that allows the user to select the text file containing the test data with the corresponding target set.

A typical test data used is shown in Table 3.

Table 3: Testing Data Set

Input Values				Target Output
0	0	0	1	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	1	1
1	0	1	0	0
1	1	0	0	0
1	1	0	1	1

Figure 7 displayed the test result obtained when the network was fully trained

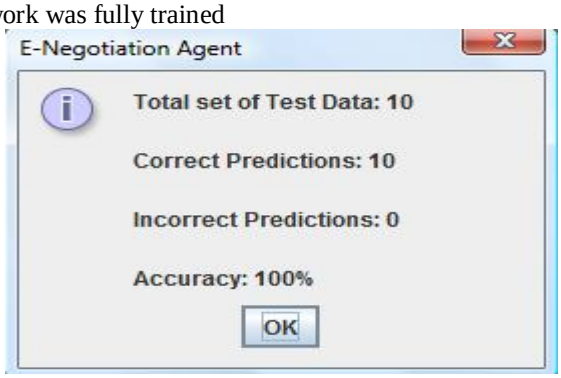

Figure 7: Testing Result

After applying numerous training activities, the best results were obtained when the hidden layer contains five neurons with about 3000 epochs. The weight values are stored in a database and they are used to determine whether a buyer is accepted among winning bidders. Table 4 shows the weights obtained at the end of the training

Table 4: Weight Values of the Neurons

		Hidden Neurons					Bias
		0	1	2	3	4	
Input	0	0.0293	0.0903	0.0715	0.0084	0.0328	
	1	0.0592	0.0306	0.0235	0.0894	0.0968	
	2	0.0187	0.0860	0.0901	0.0519	0.0518	
	3	0.0475	0.0414	0.0214	0.0414	0.0707	
	Bias	0.0258	0.0923	0.0041	0.0794	0.0514	
Output		0.0221	0.0672	0.0393	0.0182	0.0734	0.0259

6. CONCLUSION

Automated negotiation system has been implemented which does not rely on accurate or precise data to make decision with the following strength:

- With the incremental weight updating mechanism employed in the network training, the neural network is able to handle all sorts of input data whiles ensuring maximum correct prediction
- The normalization process ensures that the system streamlines all vague data before it is used in decision making

The system on the other hand is limited by the following:

- Data captured is always normalized wasting system processing time and resources
- Bidders are not prompted to check whether less sensitive data is entered or selected in error. Users, therefore, have no second chance for correcting their errors. When bidders realize their errors, they are made to start the product specification selection process again.

REFERENCES

1. G. E. Kersten, **The science and Engineering of E-negotiation: Review of the emerging field**, *INR05/02, University of Concordia*, 2002
2. S. Kraus, **Strategic Negotiation in Multi-agent Environments**, *The MIT Press*, 2001
3. S. Fatima, M. Wooldridge and N. R. Jennings, **An Agenda Based Framework for Multi-Issue Negotiation**, *Artificial Intelligence*, Volume 152, 2004, pp 1-45
4. A. Brun and A.P.Staudacher, **Negotiation-driven Supply Chain Co-ordination for Small and Medium Enterprises**", *Proceedings of the ECAI Workshop on agent technologies and their application scenarios in logistics edited by I.J.Timm*, 2000
5. A. R. Lomuscio, M. Wooldridge and N. R. Jennigs, **A Classification Scheme for Negotiation in Electronic Commerce**, *International Journal of Group Decision and Negotiation*, 12(1), pp. 31-56, 2003.
6. G. E. Kersten, S. E. Strecker and K. P. Law, **Protocols for Electronic Negotiation Systems: Theoretical Foundations and Design Issues**, *INR 06/04, Interneg Research paper, Canada*, 2004
7. I. Ioannidis and A. Grama, **An Efficient Protocol for Yao's Millionaires' Problem**, *36th Hawaii International Conference on System Sciences (HICSS'03)*, 2003
8. V. Sanchez-Anguix, V. Julian, V. Botti, A. Garcia-Fornes, **Reaching Unanimous Agreements Within Agent-Based Negotiation Teams With Linear and Monotonic Utility Functions,** *IEEE Transactions on Systems, Man, and Cybernetics*, Volume: 42, Issue: 3, Page(s): 778 - 792, 2012
9. J. B. Kim and A. Segev, **A Framework for Dynamic e-Business Negotiation Processes**, *Proceedings of IEEE Int. Conf. E-Commerce*, pp. 24–27, Jan. 2003.
10. I. Rahwan, R. Kowalczyk, and H. H. Pham, **Intelligent Agents for Automated One-to-Many e-Commerce Negotiation,** *Aust. Comput. Sci. Commun.*, Vol. 24, No. 1, pp. 197–204, Feb. 2002.
11. T. D. Nguyen and N. R. Jennings, **Concurrent Bi-lateral Negotiation in Agent Systems**, *In Proc. 4th DEXA Workshop e-Negotiations*, pp. 839–844, 2003.
12. R. Kowalczyk and V. Bui, **On constraint-based reasoning in negotiation agents**, *In Proc. Agent-Mediated E-Commerce*, Vol. III, pp. 31–46, 2003.
13. C. Li, J. Giampapa, and K. Sycara, **Bilateral negotiation decisions with uncertain dynamic outside options**, *In Proc. 1st IEEE Int. Workshop Electron. Contract*, pp. 54–61, 2004.
14. B. An, K. M. Sim, L. G. Tang, S. Q. Li, D. J. Cheng, **Continuous-Time Negotiation Mechanism for Software Agents-Systems**, *IEEE Transactions on Man, and Cybernetics*, Volume: 36 , Issue: 6 Page(s): 1261 - 1272, 2006
15. R. Axelrod, **The Evolution of Cooperation**, *New York: Basic Books,*1984
16. T. Bui, J. Yen, J. Hu, and S. Sankaran, **A Multi-Attribute Negotiation Support Systems with Market Signaling for Electronic Markets**, *Group Decision Negotiation*, Vol. 10, pp. 515–537, 2001
17. N. Karacapilidis and P. Moraitis, **Building an Agent-Mediated Electronic Commerce System with Decision Analysis Features**, *Decision Support Syst.*, Vol. 32, pp. 53–69, 2001
18. Y. Yuan, J. Rose, and N. Archer, **A Web-Based Negotiation Support System**, *Electron. Markets*, Vol. 8, pp. 13–17, 1998
19. S. Talluri, **A Buyer-Seller Game Model for Selection and Negotiation of Purchasing Bids**, *Eur. J. Oper. Res.*, Vol. 143, pp. 171–180, 2002
20. S. Talluri, R. Narasimhan, and S. Viswanathan, **Information Technologies for Procurement Decisions: A Decision Support Systems for Multi-Attribute e-Reverse Auctions**," *Int. J. Prod. Res*., Vol. 45, pp. 2615–2628, 2007
21. R. P. Sundarraj, X. Shi, **Optimization-Based Methods for Improving the Accuracy and Outcome of Learning in Electronic Procurement Negotiations**, *IEEE Transactions on Engineering Management*, Volume: 36 , Issue: 99, Page(s): 1 - 13, 2011
22. D. Zeng andK. Sycara, **Bayesian learning in Negotiation**, *Int. J. Human-Comput. Studies*, Vol. 48, no. 1, pp. 125–141, 1998.
23. R. Coehoorn and N. Jennings, **Learning an opponent's preference to Make Effective Multi-Issue Negotiation Trade-Offs**, *In Proc. Int. Conf. Electronic Commerce*, pp. 59–68, 2004
24. J. Li, J. Huai, C. Hu, and Y. Zhu, **A Secure Collaboration Service for Dynamic Virtual Organizations**, *Inf. Sci*., Vol. 180, pp. 3086–3107, 2010

25. S. Paurobally, P. J. Turner, and N. R. Jennings, **Automating Negotiation for m-Services**, *IEEE Trans. Syst., Man, Cybern*. Vol. 33, No. 6, pp. 709–724, Nov. 2003
26. K. Gurney, **An Introduction to Neural Networks**, Routledge, ISBN 1-85728-673-1 London, 1997
27. A. Krenker, J. Bester, A. Kos, **Introduction to the Artificial Neural Networks, Artificial Neural Networks - Methodological Advances and Biomedical Applications**, ISBN: 978-953-307-243-2, InTech, 2011
28. A. Krenker, M. Volk, U. Sedlar, J. Bester, A. Kos, **Bidirectional Artificial Neural Networks for Mobile Phone Fraud Detection**, *ETRI Journal.*, Vol. 31, No. 1, pp. 92-94, Feb. 2009.
29. R. Rojas, **Neural Networks: A Systematic Introduction**, Springer, ISBN 3-540-60505-3, 1996.
30. H. Wakuya, K. Shida, **Bi-Directionalization of Neural Computing Architecture for Time Series Prediction**. *Proceedings of International Joint Conference on Neural Networks*, pp. 2098–2103, 2001
31. T. M. Hagan, H. B. Demuth, M. Beale, **Neural Network Design**, China Machine Press, 2002.
32. Thiang, Handry Khoswanto, Rendy Pangaldus, **Artificial Neural Network with Steepest Descent Backpropagation Training Algorithm for Modeling Inverse Kinematics of Manipulator**, *World Academy of Sciences, Engineering and Technology*, 2009
33. Himavathi, Anitha, Muthuramalingam, **Feed forward Neural Network Implementation in FPGA Using Layer Multiplexing for Effective Resource Utilization**, *IEEE Transactions on Neural Networks*, 2007.