Mathematical Exploration of B2C Electronic Commerce architecture via Back propagation Network Learning Algorithm

Riktesh Srivastava

Associate Professor, Information Systems
Skyline University College, Sharjah, UAE
riktesh.srivastava@gmail.com

Abstract

The present study assesses the technique of back propagation neural networks to appraise the average response time of a B2C Electronic Commerce architecture. In order to delineate the response time, diverse array of user requests were engaged per unit time. Furthermore, engagement of Back Propagation Network Learning (BPNL) algorithm is used to summarize the average response time and augment the enactment of the system. The comprehensive study does the comparative investigation to express the average response time for ANN enabled and without-ANN-enabled algorithm. The objective was to plaid whether ANN enabled algorithm had any bearing on the overall performance of the system. For BPNL algorithm, learning of the responses for the user requests were steered for 7 repetitions and then thorough phases were accomplished to assess the response time. After each iteration, error rates were dogged and then feed forward and back propagation algorithm were used to improve the performance. The experimentation will find its prominence in imminent B2C Electronic Commerce system project and employment and will convey the outline for such investigation. Finally, the study expands the meticulous inferences of the study.

Keywords: B2C, Electronic Commerce architecture, BPNL Algorithm, ANN

1. Introduction

The determination of response time for B2C EC architecture is primarily a mathematical problem. Artificial neural networks are biologically stimulated classification algorithms that entails of an input layer of nodes, one or more hidden layers and an output layer. Each node in a layer has one corresponding node in the next layer, thus spawning the stacking effect (Shrivastava & Singh, 2011). Back propagation Network Learning Algorithm (BPNL) is one of the prevalent structures amid artificial neural networks which are extensively used to elucidate

complex problems by modeling complex input-output relationships (Karimi, Menhaj, & Saboori, 2010).

The preliminary BPNL algorithm was suggested by Rumelhart, Hinton, and Williams (1988) and since then became prominent learning algorithms for ANN. BPNL uses gradient-decent search procedure to alter the connection weights. The structure of a BPNL algorithm is revealed in Figure 2. The output of each neuron is the accumulation of the numbers of neurons of the previous level multiplied by its corresponding weights. The input values are converted into output signals with the calculations of activation functions (Hajmeer & Basheer, 2003). BPNL algorithm has been extensively and efficaciously functional in varied applications, such as pattern recognition, location selection and performance evaluations.

Press (1997) and Yao (2004) remarks on the diffusion of electronic commerce architecture with ANN for operative formation of the systems. ANN is the choice for such diffusion as it does not necessitate any expectations about the distribution of data. Hecht-Nielsen (1990) premeditated the mathematical analysis of such a diffusion. Research also exhibited that flexibility and generalization are two most commanding facets of ANN modeling involving BPNL. Sarle (1995), and Wieland and Leighton (1987) directed that if ANN models are instigated appropriately in an Electronic Commerce architecture, they are proficient of modeling complex patterns in data, and they can be pooled with other models to further mend the performance.

Concerning such a diffusion of ANN and Electronic Commerce architecture, experimentation conducted in the study expounds the relative stochastic exploration to appraise the response time of the B2C Electronic Commerce architecture with and without ANN enabled BPNL algorithm. For BPNL algorithm, 7 recapitulations (training) were steered to train the entreaties about users probe. BPNL algorithm was developed using Java programming language and employs both feed forward and back propagation approaches to amend the weights accordingly.

Complete paper is alienated into 6 sections. Section 2 interprets the B2C Electronic Commerce architecture used for the study. Section 3 confirms the mathematical exploration of

gaging the response time without encompassing BPNL algorithm. Section 4 references the mathematical valuations for the BPNL algorithm. Section 5 mentions the result investigation with and without implementing BPNL algorithm. Section 6 explicates the supposition and impending work to be steered.

2. B2C Electronic Commerce architecture

The proposed B2C Electronic Commerce architecture is an extension to the system of Client Server Computing. In the architecture, the chore of Web Server is to yield requests from client and handover it to Application Server. The requests are further conveyed to Database Servers. So the client using the application is not fretful with the complexities of the Business logic and is presented the complete web application with supplementary service. Thus, the architecture offers a momentous workload shift.

In the proposed architecture, the Web server has to no longer do the entire profound lifting when it comes to running applications. The Application Server and Database Server(s), hold the impediments of the architecture. Also, the hardware and software demands on the user's side dwindle and the web server only executes the architectures interface software. The comprehensive architecture is depicted in Figure 1.

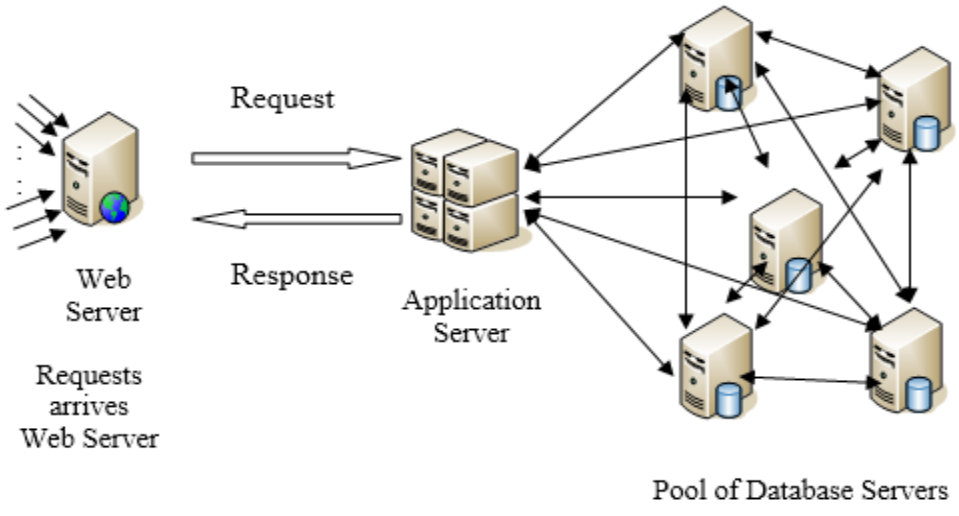

Figure 1 B2C Electronic Commerce architecture

As illustrated in Figure 1, all the clients requests is being established at the Web Server. Once the requests are received at Web Server, it gets transported to the Application Server for further dispensation. Application Server spawns the business logic and then requests get conveyed to the pool of database servers. The study illustrates the employment of the BPNL algorithm between Application Server and Pool of Database Servers. During the research it was witnessed that the foremost time of user entreaties was sandwiched between these two servers. The search of the data in the pool of database server postponements the response time. Employing and continuously training the requests reduces the response time, enhancing the overall system performance.

3. Response time of B2C architecture estimations without Neural Networks

As indicated in Figure 1, the requests need to be passed through the three different types of server. Based on the assumption the total response time is based on the time at each of the server.

$$\text{Total Response time} = tWS + tAS + tDBS \quad -(1)$$

However, while conducting the experiment, it was observed that the requests take the maximum time at the DBS. Upon further investigation, it was observed that the requests take twice the time at the DBS than at WS and AS together. Keeping the fact in view, equation 1 can be described as

$$\text{Total Response time} = tWS + tAS + 2(tWS + tAS) \quad -(2)$$

$$\text{Total Response time} = 3*tWS + 3*tAS \quad -(3)$$

4. Back Propagation Request Learning (BPNL) Architecture

Figure 2 represents the architecture of a simple Neural Networks. As portrayed in the Figure, we have one hidden layer, which is associated to the node in output layer. There is customarily some weights associated with every connection. As depicted in Figure 1, at the input layer, we get the requests from the client, which is usually the raw information. This raw information is fed into the network and gets transferred to the hidden layer, as depicted in Figure 2. Hidden layer accepts data from the input layer. It uses input values and modifies them with

some weight value, this new value is then send to the output layer but it will also be adjusted by some weight from connection amongst hidden and output layer. Output layer process information received from the hidden layer and produces the output. This output is then processed by activation function.

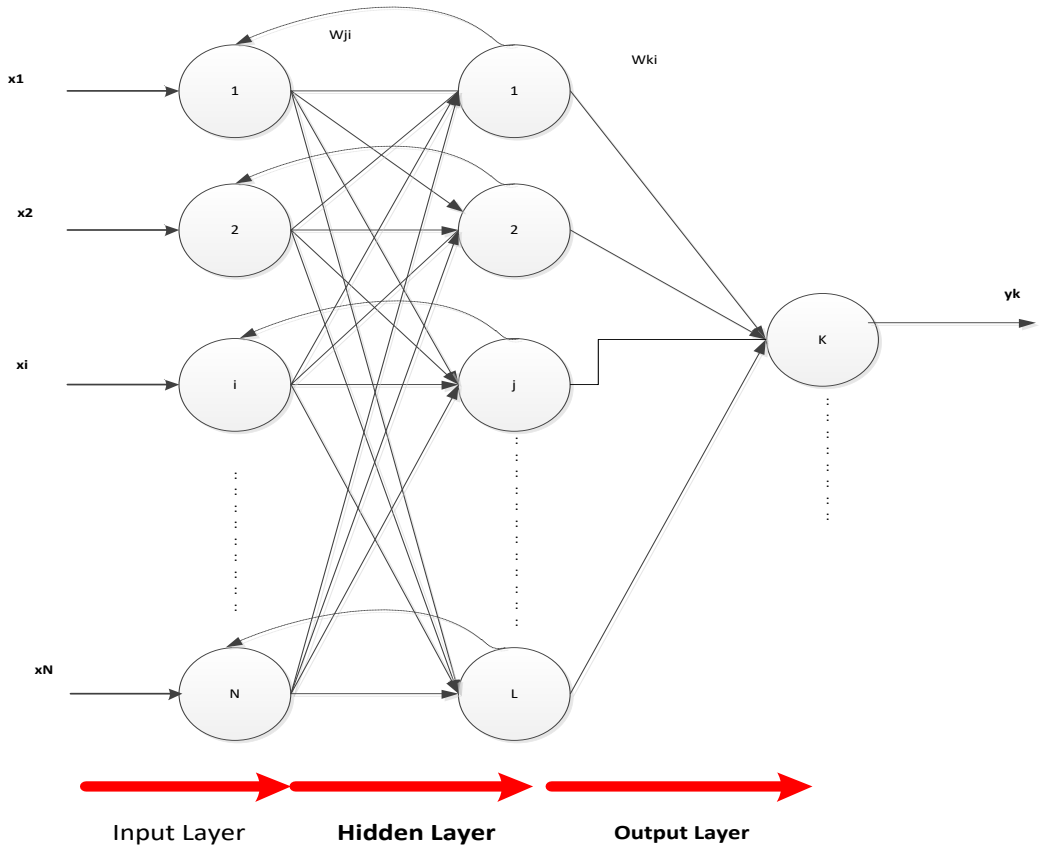

Figure 2 Neural Network approach for scheduling requests

4.1 Mathematical Evaluation of BPNL Algorithm

The BPNL founds its base on the study piloted by Rojas (2005), which claims that the complete algorithm should be broken into four stages. After selecting the weights of the network randomly, the BPNLA is used to compute the necessary corrections. The algorithm can be decomposed in the following four steps:

1. *Feed-forward computation*

2. *Back propagation to the output layer*
3. *Back propagation to the hidden layer*
4. *Weight updates*

The algorithm is clogged when the assessment of the error function has become adequately insignificant. This is very rough and rudimentary assumption for BPNL algorithm. There are some variation but BPNL algorithm based on Rojas (2005) elucidation seems to be fairly precise and easy to follow. The last step, weight updates is happening throughout the algorithm.

BPNL algorithm is being assessed based on the number of requests incoming at the Web Server. The purpose of the algorithm is to accomplish fast response time, after instigating the BPNL algorithm amid Application Server and Database Server. Employment of BPNL algorithm for Figure 1 is depicted in Figure 3 as given below:

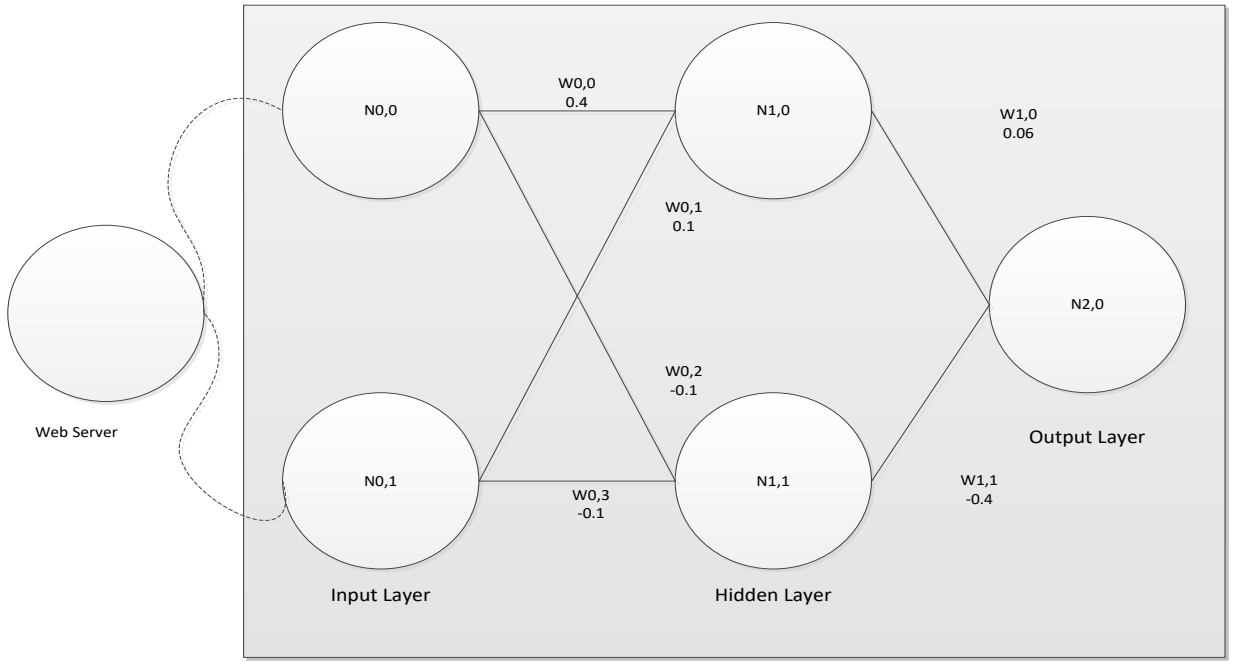

Figure 3 Implementation of BPNL Algorithm

As displayed in Figure 3, the values of weights are taken randomly and will be transformed during BPNL iterations. The sigmoid function formula is $f(x) = \frac{1.0}{1.0+\exp(-x)}$, with Learning rate, β=0.45 and Momentum term, α=0.9.

4.2 BPNL Feed Forward computation

Based on sigmoid function formula, $f(x) = \frac{1.0}{1.0+\exp(-x)}$, the feed forward computation for the Hidden and Output Layers are

$$N1,0 = f(x1) = f(w0,0 * n0,0 + w0,1 * n0,1) = f(0.4 + 0.1) = f(0.5) = 0.622459$$

$$N1,1 = f(x2) = f(w0,2 * n0,0 + w0,3 * n0,1) = f(0.4 + 0.1) = f(-0.2) = 0.450166$$

$$N2,0 = f(x3) = f(w1,0 * n1,0 + w1,1 * n1,1) = f(0.06 * 0.622459 + (-0.4) * 0.450166)$$
$$= f(-0.1427188) = 0.464381$$

Computation of N2,0 completes the BPNL feed forward computation.

4.3 BPNL Back propagation to the output Layer

This step gages the error at the node N2,0. Since, the study boards to get the fast response time, for every search from the web server should give the meticulous result, at the first search at the Database server. In other words, the output should be always be exact$100\% \cong 1$, for every search. The value of N2,0, calculated above is 0.464381.

Error calculation for N2,0

$$N2,0_{Error} = n2,0 * (1 - n2,0) * (N2,0_{Desired} - N2,0)$$
$$= 0.464381(1 - 0.464381) * (1 - 0.464381) = 0.133225$$

Once, error is computed, it will be used for backward propagation and weights. Error is broadcasted from the output layer to the hidden layer first, for which learning rate and momentum is recycled in the equation. So, the weights W1,0 and W1,1 are updated first.

$$\Delta W1,0 = \beta * N2,0_{Error} * n1,0 = 0.45 * 0.133225 * 0.622459 = 0.037317$$

Based on the above-mentioned calculation, the value of $W1,0_{New}$ is as follows:

$$W1,0_{New} = w1,0_{Old} + \Delta W1,0 + \big(\alpha * \Delta(t-1)\big) = 0.06 + 0.037317 + 0.9 * 0 = 0.097137$$

$$\Delta W1,1 = \beta * N2,0_{Error} * n1,1 = 0.45 * 0.133225 * 0.450166 = 0.026988$$

$$W1,1_{New} = w1,1_{Old} + \Delta W1,1 + \big(\alpha * \Delta(t-1)\big) = -0.4 + 0.026988 = -0.373012$$

$\Delta(t-1)$ portrays any preceding delta change of weights, which is always used after the first iteration. As there is no previous delta value as of now, it is placed as 0. For the next iteration, the value of $\Delta(t-1)$ will be updated accordingly.

4.4 BPNL Back propagation to the hidden Layer

This step is used to evaluate the errors propagated from the hidden layer to the input layer.

$$N1,0_{Error} = N2,0_{Error} * W1,0_{New} = 0.133225 * 0.097317 = 0.012965$$

$$N1,1_{Error} = N2,0_{Error} * W1,1_{New} = 0.133225 * (-0.373012) = -0.049706$$

Once the error for hidden layer is calculated, weights between input and hidden layers can be updated.

$$\Delta W0,0 = \beta * N1,0_{Error} * n0,0 = 0.45 * 0.012965 = 0.005834$$

$$\Delta W0,1 = \beta * N1,0_{Error} * n0,1 = 0.45 * 0.012965 * 1 = 0.005834$$

$$\Delta W0,2 = \beta * N1,1_{Error} * n0,0 = 0.45 * -0.049706 * 1 = -0.022368$$

$$\Delta W0,3 = \beta * N1,1_{Error} * n0,0 = 0.45 * -0.049706 * 1 = -0.022368$$

Thus, we can now calculate the new weights between input and hidden layer, as indicated below:

$$W0,0_{New} = W0,0_{Old} + \Delta W0,0 + \big(\alpha * \Delta(t-1)\big) = 0.4 + 0.005834 + 0.9 * 0 = 0.405834$$

$$W0,1_{New} = W0,1_{Old} + \Delta W0,1 + \big(\alpha * \Delta(t-1)\big) = 0.1 + 0.005834 + 0 = 0.105834$$

$$W0,2_{New} = W0,2_{Old} + \Delta W0,2 + \big(\alpha * \Delta(t-1)\big) = -0.1 + -0.022368 + 0 = -0.122368$$

$$W0,3_{New} = W0,3_{Old} + \Delta W0,3 + \big(\alpha * \Delta(t-1)\big) = -0.1 + -0.022368 + 0 = -0.122368$$

4.5 Final Weight updates (Iterations)

Above mentioned three steps of BPNL algorithm is the first pass of the comprehensive algorithm. The intention is to obtain the maximum accuracy, so the results are searched at the first instance from Database server. The study conducted by dspguide.com (Smith, 2001), illustrates that the number of iterations can be any number between ten to ten thousands. For the study, seven iterations were followed, to get the result.

The second pass using new weights to check if the error has decreased.

$$N1,0 = f(x1) = f(w0,0 * n0,0 + w0,1 * n0,1) = f(0.406 + 0.1) = f(0.506)$$
$$= 0.623868314$$

$$N1,1 = f(x2) = f(w0,2 * n0,0 + w0,3 * n0,1) = f(-0.122 - 0.122) = f(-0.244)$$
$$= 0.43930085$$

$$N2,0 = f(x3) = f(w1,0 * n1,0 + w1,1 * n1,1) = f(0.097 * 0.623868314 + (-0.373))$$
$$= f(-0.103343991) = 0.474186972$$

Calculating $N2,0$ completes the forward pass and now for error calculation of $N2,0$ node.

$$N2,0_{Error} = n2,0 * (1 - n2,0) * (N2,0_{Desired} - N2,0)$$
$$= 0.474186972(1 - 0.474186972) * (1 - 0.474186972) = 0.131102901$$

It can be easily checked that the error has decreased and response time is decreased. Next section depicts the error for 7 iterations of BPNL algorithm.

4.6 Error evaluation

Table 1 given below depicts the $N2{,}0_{Error}$ for next 7 iterations. It can be observed from the table that the error rate has significantly decreased on the repeated iterations.

Iterations	$\boldsymbol{N2,0_{Error}}$
1	0.133225
2	0.131102
3	0.129762
4	0.114131
5	0.109769
6	0.104476
7	0.101634

Table 1 $N2{,}0_{Error}\ for\ 7\ iterations$

5. Result Analysis: With and Without Implementing BPNL algorithm

The comprehensive BPNL algorithm is stated in this section, and as indicated adjusts the weights of the user's requests using sigmoid function. However the computational exertion obligatory for finding the correct combination of weights increases substantively when more strictures and more complicated topologies are deliberated. In the algorithm, sigmoid functions were used to shrink the overall response time. This method is not only more general than the usual analytical derivations, but also much easier to follow. It also shows how the algorithm can be efficiently implemented in B2C Electronic Commerce architecture where the number of requests arriving at web server is random in nature.

5.1 Generic BPNL Algorithm

1. Initialize weights for the request type (small random numbers)
2. For each training of user requests
3. Repeat until weights convergence or till a required number of epochs are completed

i. Receive requests as it will be extracted from various queues
ii. Propagate the error backward from output layer to hidden and input layer.
iii. Calculate new weights in accordance with BPNL algorithm.

4. Replace old weights new weights as taken from training algorithm
   i. After every ' t' time units
   ii. Measure performance of each requests
   iii. Repeat until performance falls below a threshold level(Δ) else go to Step iv.
   iv. Set activation of input unit. Inputs to input layer will be actual packets that are to be scheduled.
   v. Compute output of hidden and output layer using sigmoid activation function
   vi. Output will be fed to weight decider module which will calculate the required change in weights of the queues.

5.2 Algorithm Output

The extensive aftermath of the research piloted is portrayed in Table 2 below:

Number of requests	Response Time without BPNL Algorithm	Response Time with BPNL Algorithm (Training 1)	Response Time with BPNL Algorithm (Training 2)	Response Time with BPNL Algorithm (Training 3)	Response Time with BPNL Algorithm (Training 4)	Response Time with BPNL Algorithm (Training 5)	Response Time with BPNL Algorithm (Training 6)	Response Time with BPNL Algorithm (Training 7)
25	100.65	88.572	81.48624	76.5970656	75.63960228	72.60645423	69.95631865	61.56156041
50	104.71	92.1448	84.773216	79.68682304	78.69073775	75.53523917	72.77820294	64.04481859
75	107.63	94.7144	87.137248	81.90901312	80.88515046	77.64165592	74.80773548	65.83080722
100	109.44	96.3072	88.602624	83.28646656	82.24538573	78.94734576	76.06576764	66.93787552
125	117.21	103.1448	94.893216	89.19962304	88.08462775	84.55243418	81.46627033	71.69031789
150	123.84	108.9792	100.260864	94.24521216	93.06714701	89.33515441	86.07442128	75.74549072
175	149.01	131.1288	120.638496	113.4001862	111.9826839	107.4921783	103.5687138	91.14046813
200	161.54	142.1552	130.782784	122.935817	121.3991192	116.5310146	112.2776325	98.80431663
225	177.05	155.804	143.33968	134.7392992	133.055058	127.7195501	123.0577866	108.2908522
250	181.61	159.8168	147.031456	138.2095686	136.481949	131.0090229	126.2271935	111.0799303
275	190.09	167.2792	153.896864	144.6630522	142.854764	137.126288	132.1211785	116.266637
300	221.32	194.7616	179.180672	168.4298317	166.3244588	159.654848	153.827446	135.3681525

Table 2 Outcomes of the Experiment

The result analysis is illustrated in Figure 4 below:

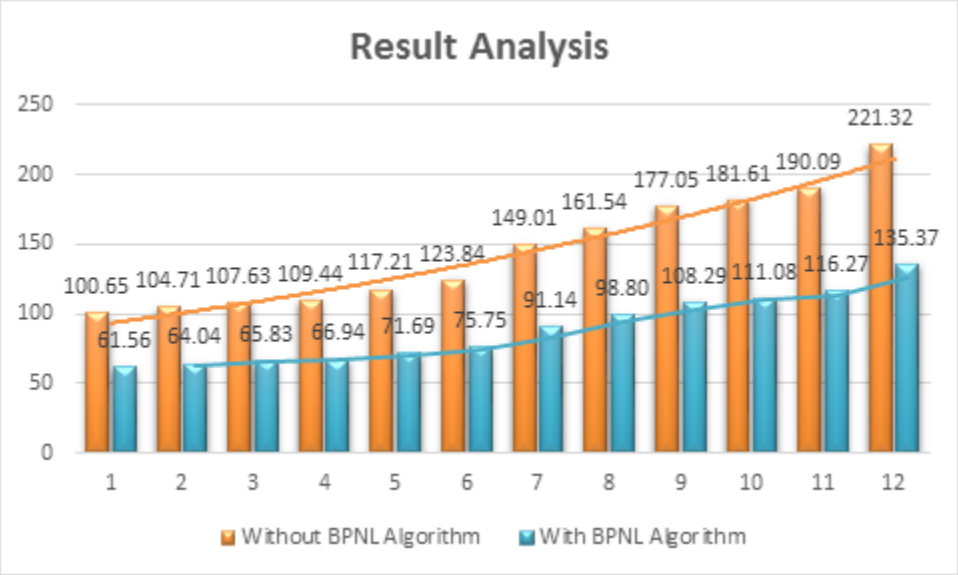

Figure 4 Result Analysis of the experiment

6. Conclusion and Future Work

As evidently specified in the research that employment of BPNL in B2C Electronic Commerce architecture profoundly upsurges the systems performance. Figure 4 of section 5 represents the outcomes of the investigation accompanied without BPNL algorithm and after 7th training using BPNL. Originally, it was pragmatic that BPNL algorithm does not have any influence on the results, however, incessant training has an inclusive influence on the system. The study exhibits that the algorithm gives the enhanced consequences with concentrated 300 requests being acknowledged per unit time. The proposed BPNL algorithm was based on Ergodic condition and permanence was maintained and scrutinized throughout the implementation of the experimentation. The study is premeditated to be demeanor even advance, when elevated amount of requests being acknowledged and Service time being assessed accordingly. BPNL algorithm tangled only 7 autonomous training sets, however, to get a precise aftermath of the query, the training had to be continual in assortment of 10-1000 iterations, which is premeditated for the advance version of the system implementation.

Reference

Hajmeer, MN, & Basheer, IA. (2003). A hybrid Bayesian–neural network approach for probabilistic modeling of bacterial growth/no-growth interface. *International journal of food microbiology, 82*(3), 233-243.

Hecht-Nielsen, R. (1990). *Neurocomputing*: Reading, MA: Addison-Wesley.

Karimi, B., Menhaj, M. B., & Saboori, I. . (2010). Multilayer feed forward neural networks for controlling de-centralized large-scale non-affine nonlinear systems with guaranteed stability. *International Journal of Innovative Computing, Information and Control, 6*(11), 4825-4841.

Press, Larry. (1997). Tracking the global diffusion on the Internet. *Communications of the ACM, 40*(11), 11-17.

Rojas, Raúl. (2005). *Neutral Networks: A Systematic Introduction*: Springer.

Rumelhart, David E, Hinton, Geoffrey E, & Williams, Ronald J. (1988). *Learning representations by back-propagating errors*: MIT Press, Cambridge, MA, USA.

Sarle, Warren S. (1995). *Stopped training and other remedies for overfitting.* Paper presented at the Proceedings of the 27th Symposium on the Interface of Computing Science and Statisfi (.'. _\'. pp. 352-360. Interface Foundation of North America, Fairfax Station. VA, USA.

Shrivastava, Saurabh, & Singh, Manu Pratap. (2011). Performance evaluation of feed-forward neural network with soft computing techniques for hand written English alphabets. *Applied Soft Computing, 11*(1), 1156-1182.

Smith, Steven W. (2001). Introduction to Neural Networks. from http://www.dspguide.com/CH26.PDF,

Wieland, As, & Leighton, R. (1987). *Geometric analysis of neural network capabilities.* Paper presented at the Proceedings of the Second IEEE International Conference on Nerual Networks.

Yao, Jingtao. (2004). Ecommerce Adoption of Insurance Companies in New Zealand. *J. Electron. Commerce Res., 5*(1), 54-61.