International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 10 (2017) pp. 2262-2271
© Research India Publications. http://www.ripublication.com

An ANN based Classification Scheme for QoS Negotiation to Achieve Overall Social Benefit in Cloud Computing

B. Sunil Kamath
Research Scholar, Department of CSE, St Joseph Engineering College, Vamanjoor, Mangalore, India.

Dr. Rio D'Souza
Professor & HOD, Department of CSE, St Joseph Engineering College, Vamanjoor, Mangalore, India.

[1]ORCID : 0000-0003-0994-5580 [2]ORCID-ID is: 0000-0003-0994-5580

Abstract

As there is a rise in the usage of cloud services in the marketplace, so also the issues related to the cloud usage like pricing, SLA monitoring, QoS negotiation and QoS assurance are assuming importance. This work deals with trying to come up with a solution in QoS Negotiation domain. In this work couple of QoS negotiation issues are considered like price and reliability. The set of starting scenarios are created for varying values of reserved, preferred values of price/reliability for buyer and seller. Even the weights are considered as the buyer and seller are going to have opposing preferences. This semi-complex data is considered against four major negotiation strategies namely Simple concession, Tradeoff, Fuzzy based strategy and Bayesian strategy. All the generated scenarios are tested against the above four strategy to record the best strategy for the particular scenario in a log. These recorded values are input to back-propagation neural network to train it to identify the best strategy for any scenario. Once trained the neural network is going to report best strategy for any scenario and that strategy is applied at both buyer and seller end to negotiate the best price for a given reliability. As a future enhancement, still more comprehensive scenario set may be generated and verified with given neural network.

Keywords: Cloud Computing, QoS Negotiation, Game Theory, Fuzzy Logic, Bayesian Learning, Artificial Neural Network.

INTRODUCTION

The 20th century has seen a predominant rise of computers being applied for e-commerce purposes. The decade starting in 2010 has seen the rampant use of cloud technology for storage, compute and network purposes [1]. Cloud technology has some of the features like scalability, multi-tenancy, virtualization, storage, communication and parallel computation. Among these features we focus on cloud storage in this paper which has several issues like price, reliability, availability and response time. Cost effectiveness is one of the features of cloud storage i.e. the cost of renting the cloud storage clearly offsets the cost of investing in on-premises systems which during non-peak traffic time would be idling. One of the feature of cloud storage is once the data is stored in cloud servers, the system allows you to access the data from anywhere in the world at any time as the cloud servers have been positioned in geographically distinct regions. Another feature of cloud storage is its infinite capacity as the systems implements virtualization. Storage servers will be scaled up/down depending on the storage requirements thus giving an impression of infinite capacity. Cloud storage is robust as well as the data uploaded into cloud is replicated in many servers so that if one of the server goes down the data can be rescued from the replicas kept elsewhere.

QoS-negotiation is an important aspect of simple cloud storage [2]. In the current work we have tried to keep cloud storage at the center and tried to elaborate on QoS Negotiation. The main QoS issue of interest to us is reliability. The reliability of a cloud storage which is 70% reliable can be increased by replicating it once i.e. 1-(1-0.70)^2 =91% which is good enough [5]. More reliable systems are required when negotiating for banking applications which cannot tolerate even minor errors in transaction/data loss whereas storage of an online video service need not be that critical with respect to data loss. Clearly type of application determines the reliability requirements of the system.

The automated QoS negotiation [22] that we consider in this work is based on two issue negotiation between buyer (cloud storage consumer) and seller (cloud service provider). The two issues considered are price and reliability. The buyer and seller as in any negotiation [3][4] have conflicting roles to play i.e. price is a higher-is-better attribute for seller whereas it is lower-is-better attribute for buyer. Similarly for reliability the two parties have conflicting roles i.e reliability is lower-is-better attribute for seller whereas it is higher-is-better attribute for buyer. In such situations we need some negotiating strategies to draw a balance between the preferences of opposing parties.

In the earlier work on negotiation, four major strategies for QoS negotiation namely simple concession, tradeoff [5], fuzzy logic based strategy [6] and bayesian learning strategy [7][8] have been implemented. The first two strategies are considering price and reliability as the prime issue for negotiation. Simple concession strategy is based on making concession on price/reliability issue in each round of negotiation by both buyer and seller thereby reducing the total utility of final accepted offer. The Tradeoff on the other hand yields more on its lower-is-better attribute (Reliability) to create a more attractive offer for the opponent

Table I: A sample problem scenario

	Min	**Max**	**Service Provider(seller)**			**Service Consumer(buyer)**		
			Res.value	Pref.value	Weight	Res.value	Pref.value	Weight
Rel.	0%	100%	90%	75%	0.1	80%	95%	0.9
Price	0	0.1	0.06	0.09	0.9	0.08	0.05	0.1

simultaneously demanding more on its higher-is-better attribute (price) with total utility remaining constant. This kind of negotiation is called Pareto optimal negotiation. The third kind of negotiation is Fuzzy Logic based concession in which the price, reliability, acceptance membership functions are written along with rule set. The counter-offer generated by opponent is checked with the fuzzy model to determine acceptance or rejection of the offer. Fuzzy model has separate rule sets and membership functions for buyer and seller. This strategy for negotiation has come in handy for places where tradeoff did not perform well. Finally the conventional bayesian learning technique was extended for reliability [24] and was applied to learn opponent reserve price and reliability in successive negotiation rounds to arrive at a best offer for opponent within a given deadline. This strategy also was found to give better overall utility in some scenarios.

Once the above mentioned four negotiation strategies are in place, we generate starting scenarios by varying the weights. These scenarios generated were fed as input to the four negotiation strategies separately to evaluate the best strategy for the given scenario along with time taken to complete negotiation (round) and this data was recorded in a log. This was repeated for each individual scenario's to create a dataset of hundreds of scenarios each tagged with the best negotiation strategy for that scenario.

After generating dataset in this way, we used the dataset (history) to train a back-propagation neural network unit to classify the scenarios into one of the four classes (Concession, Tradeoff, Fuzzy, Bayesian). After training and retraining the neural network unit on history (dataset), it was ready to classify any given scenario into one of the four classes based on learning received from history. That strategy output by the neural network was used to conduct the bilateral negotiation between buyer and seller to achieve best possible social benefit and overall utility. This current scenario was once again updated into history (log) so that future results of classification were affected by the current scenario classification.

The remainder of the paper is divided into 6 sections. Section II details on the related work. Section III the actual problem scenario. Section IV discusses the Artificial Neural Network (ANN) Section V deals with creation of dataset for training the ANN. Section VI discusses use of ANN for classification of starting scenarios in the dataset Section VII gives the conclusion and future work.

RELATED WORK

Literature that relates to our research work can be categorized as follows: creation of strategies for automatic QoS negotiation and the use of ANN for classification of best possible strategy for an initial starting scenario (during negotiation) from among the list of created negotiation strategies. The former category is revised first.

In game-theoretic negotiation, tradeoff stands out as concession strategy decreases the overall utility but suffers from less than 100% success rate [5][23].

In fuzzy domain, negotiation can happen if the membership functions can be priorly defined for negotiation issues (like price and reliability). Whereas Karim et al. [15] used fuzzy constraints to solve a similar multi-issue negotiation problem by using knowledge base (KB) in e-commerce scenario. In [16] the e-negotiation are considered as distributed fuzzy constraint satisfaction problem where automated agents are going for agreement acceptable to the opponent agent. Many such fuzzy based negotiations can be found in [17][18].

In bayesian domain, Zeng et al. [8] mentions that bayesian learning and updation mechanism can be used for estimating reserve price of opponent in bilateral negotiation. Bayesian learning makes use of set of hypothesis (H_i), a priori knowledge of probability of occurrence of H_i, conditional probability of e given H_i where e is encoded domain knowledge of opponent offers, to arrive at final estimate of reserve price opponent. The work in [12] suggests use of bayesian classifier for reaching an agreement in multi-lateral negotiation scenario based on past experiences. In [13][14] the bayesian Learning procedure is used to estimate opponent Reserve price and deadline and Genetic Algorithm is used to generate proposals in and around the Reserve Price and deadline generated by BL procedure.
Regarding the latter category, ANN has been used in many applications like stock prediction [9], face recognition [19], image classification [20] and automated negotiation [21][22]. We are going to apply it for strategy selection in a bilateral negotiation setting as classifier.

THE PROBLEM SCENARIO

The Table I above depicts the conflicts between service consumer and service provider in terms of two issues reliability and price for cloud procurement scenario. The problem scenario assumed for cloud is storage-as-a-service scenario.

The reliability can range from 0 to 100% (percentage per month) and price from 0 to 0.1 (dollars/GB of data

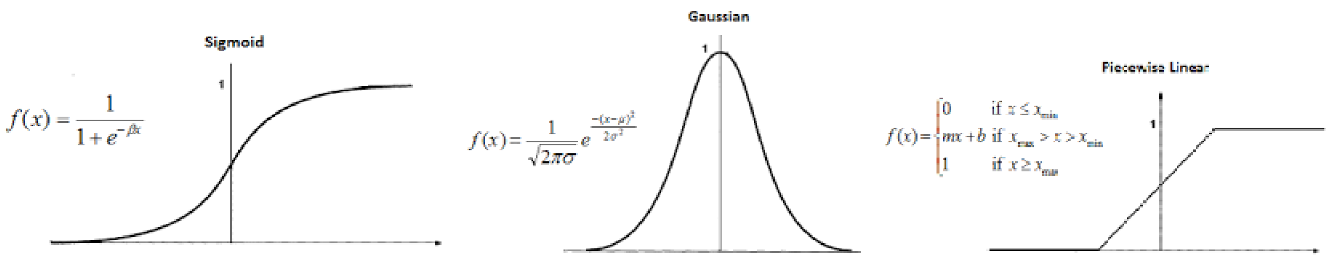

Figure 1: Different types of activation functions

/month). Now consider reliability of service provider which is 75% initially even though provider can provide 90%. Service consumer has a reliability requirement of minimum 80% although it starts negotiation initially at 95%. Clearly there is a preference gap over reliability (75% vs 95%).

Similarly consider price issue. Service Provider seeks to maximize his profit so initially expects 0.09 (can come down to 0.06) whereas Service Consumer is willing to pay only 0.05 initially (can come up to 0.08). Here too there is a conflict over price (0.09 vs 0.05).

Assumption of incomplete information: There are a few assumptions necessary before we negotiate i.e. service consumer and service provider keep their reserved value of reliability and price secret. Service Consumer knows that Service provider cares for storage price more than reliability. Service provider knows that Service Consumer cares more about reliability than price.

Artificial Neural Network (ANN)

Smallest unit of an ANN is an artificial neuron [9]. The working of artificial neuron is similar to biological neurons in human brain. The artificial neuron is programmed in such a way as to closely simulate a biological neuron i.e. the output of the neuron is a function of weighted sum of the inputs and the bias. Suppose there are n inputs designated $v_1,v_2,v_3....v_n$ and let θ be the bias which has a weight of +1 and all other inputs are weighted as $w_1,w_2...w_n$. The weighted sum at the neuron is calculated as follows

$$x = \sum_{i=0}^{n} w_i v_i \text{ --(1)}$$

where $v_0 = \theta$ and $w_0 = +1$ which represents bias for the neuron.

The output of the neuron is a function of this value x. This function usually called activation function can be Sigmoid, Gaussian or Linear.

Accordingly, the output of this neuron is calculated as

$$y = f(x) = f(x) \text{ ----------------------------------(2)}$$

A single neuron by itself is not useful but when interconnected in the form of layers can solve complex problems. The output of one layer of neurons is connected as inputs to another layer to form a neural network.

So the output of a jth neuron in such a network can be calculated as

$$net_j = \sum_{i=1}^{n} w_{ij} x_i + \theta_j \text{ ------------------------------(3)}$$

where x_i is the input from previous layer neuron or from input.

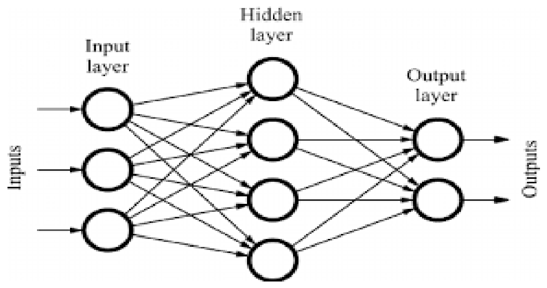

Figure 2: A sample neural network with 1 hidden layer

And output of the individual neuron is

$$o_j = \varphi(net_j) \text{ ------------------------------------(4)}$$

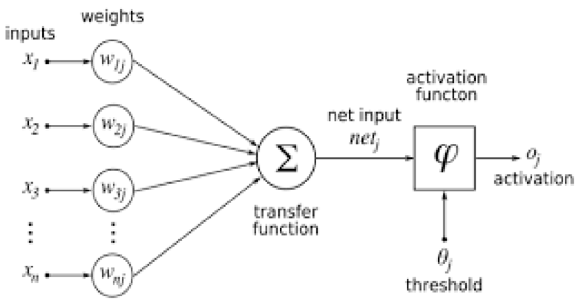

Figure 3: The calculation of output at a neuron in a neural network.

This kind of neural network is called feed-forward neural network as output of previous layer is given as input to the next layer. In this type of network the connections with neurons in the same layer or previous layer is not permitted. The first layer is called input layer. Neurons in this layer just pass on the input value to their output. The next layer is called hidden layer. There can be multiple layers of hidden neurons. In this layer output at each neuron is calculated as per the eq. (3) and (4). Finally the output layer receives its input from output of hidden layer neurons (again using equation (3) and (4)) to decide actual output from Neural Network unit.

The difference between the network output and actual output (in supervised learning) forms the error of the network. The weights of hidden-output layers are adjusted to reduce the error to 0 and the weights of input-hidden layers adjusted to reduce error at the hidden layer. That's why this algorithm is named back-propagation as the errors drive the adjustment of weights between the different layers.

Generation of Datasets for Training ANN

The initial scenario from Table I was taken as base scenario. Then the reserved/preferred reliability are varied by 5% and reserved/preferred price are varied by 10% to get staggered set of base scenarios. Those base scenario's s1-s20 are as listed in Table II. These 20 scenarios are further manipulated by varying the buyer/seller weights on reliability and price by 10%. This gives us 25 new permutations from each of the starting scenarios. That gives us 20 *25 = 500 tuples which is the size of our training set. It can easily be expanded to more number of tuples as a future enhancement.

The generation of dataset involved the following steps. 1) To create strategies for negotiation (namely concession, tradeoff, fuzzy based, bayesian based) with similar parameters as input and output. 2) Run the simple concession, tradeoff, Fuzzy based and Bayesian based strategies on starting scenario say S1 to calculate the total utility for S1 (Refer row S1 in Table III) i.e concession utility column 1.416 , tradeoff utility column 1.575, Fuzzy utility column 1.52 and Bayesian utility column 1.462. 3) Clearly the best strategy is tradeoff (1.575/round 13) which has the highest utility but when we consider the time or number of rounds of negotiation we come up with fuzzy based strategy (1.52/ round 1) as a better strategy for S1. Thus we calculate the-best-strategy for a given scenario by giving say 0.2 weightage for number of rounds of negotiation and say 0.8 weightage for total utility.4) Log the data date/time to the file along with starting scenario, best-strategy for that scenario, the nanotime taken for that scenario per strategy and number of rounds of negotiation for that scenario per strategy. 5) Read the log to create a .csv file consisting of 12 inputs representing a scenario and four outputs representing the classification and 500 rows representing all the scenarios resulting from S1-S20 in Table II. Table III is the outcome of analysis done on the four strategies vs. 20 base Scenarios keeping track of U_S (Seller side Utility), U_B (Buyer side utility), Total utility, negotiated offer (Reliability, Price) and number of rounds of negotiation. A brief discussion is as below.

1) To create simple concession and tradeoff strategy of negotiation, the algorithms from [5] were taken and implemented in java. Then Fuzzy Logic based Concession FLC strategy was devised [6] and implemented in java with support from MATLAB R2015a. Finally Bayesian learning algorithm was used to learn opponent price and reliability to generate an attractive offer for negotiation [7][24].

2) The Negotiation strategies so created were run on individual starting scenarios S1-S20 to get the offers. Then the total utility for the negotiated offer (offer.Reliability, offer.Price) was calculated by adding utility of offer from both buyer as well as seller side. The total utility for Concession, Tradeoff, Fuzzy based and Bayesian based strategies thus calculated were tabulated in table III

3) Clearly the best strategy for the current scenario is calculated based on total utility (say 0.8 weightage) and the number of rounds of negotiation (say 0.2 weightage).

4) Log the data obtained in step 2 and 3 to the file in the following format.

i.e. first line date/time module name that is logging , second line starting scenario (values separated by whitespace), the best-Strategy (say 0 0 1 0 which means Fuzzy based is best strategy), nanotime taken by four individual strategies (like 44563 8874 5495186773 9706240) and the number of rounds taken (7 13 1 12).

Example -

```
Jan 04, 2017 4:32:25 PM cloud.Testing logEverything
INFO: 0.9     0.06    0.75    0.09    0.1     0.9
      0.8     0.08    0.95    0.05    0.9     0.1
      0       0       1       0       44563   88747
       5495186773     9706240 7       13      1
      12
```

To log the data, Slf4j (simple logging façade for java) was used to write the negotiation scenarios along with resulting best strategy into the file.

5) The log file was read and values were populated into a .csv (comma separated values) list for further processing. The datasets were created from logs and it had 12 input columns, 4 output columns. The 12 input columns are as follows: Seller reserved reliability, seller reserved price, seller preferred reliability, seller preferred price, seller reliability weight, seller price weight, buyer reserved reliability, buyer reserved price, buyer preferred reliability, buyer reserved price, buyer reliability weight and buyer price weight. The four output columns represented Concession, Tradeoff, Fuzzy logic based negotiation and Bayesian learning based negotiation with a 1 in the column that has the best total utility/least time among the four and 0 otherwise. Approximately 500 rows were created in the dataset. All the input columns were normalized to be between (0-1). The formula used for normalization of both issues price and reliability was

$$x_{norm} = \frac{x-x_{min}}{x_{max}-x_{min}} \text{-----------------------------} (5)$$

As observed from Table II reliability is already in the normalized form. Only price was normalized using this formulae.

Table II: Negotiation Scenarios S1-S20

Scenario		Seller Reserve Value	Seller Prefer Value	Seller Weights	Buyer Reserve Value	Buyer Prefer Value	Buyer Weights
S1	Rel.	0.90	0.75	0.1	0.80	0.95	0.9
	Price	0.06	0.09	0.9	0.08	0.05	0.1
S2	Rel.	0.85	0.70	0.1	0.75	0.90	0.9
	Price	0.05	0.08	0.9	0.07	0.04	0.1
S3	Rel.	0.80	0.65	0.1	0.70	0.85	0.9
	Price	0.04	0.07	0.9	0.06	0.03	0.1
S4	Rel.	0.75	0.60	0.1	0.65	0.80	0.9
	Price	0.03	0.06	0.9	0.05	0.02	0.1
S5	Rel.	0.70	0.55	0.1	0.60	0.75	0.9
	Price	0.02	0.05	0.9	0.04	0.01	0.1
S6	Rel.	0.90	0.75	0.1	0.80	0.95	0.9
	Price	0.02	0.05	0.9	0.04	0.01	0.1
S7	Rel.	0.85	0.70	0.1	0.75	0.90	0.9
	Price	0.03	0.06	0.9	0.05	0.02	0.1
S8	Rel.	0.80	0.65	0.1	0.70	0.85	0.9
	Price	0.04	0.07	0.9	0.06	0.03	0.1
S9	Rel.	0.75	0.60	0.1	0.65	0.80	0.9
	Price	0.05	0.08	0.9	0.07	0.04	0.1
S10	Rel.	0.70	0.55	0.1	0.60	0.75	0.9
	Price	0.06	0.09	0.9	0.08	0.05	0.1
S11	Rel.	0.90	0.80	0.1	0.85	0.95	0.9
	Price	0.07	0.09	0.9	0.08	0.06	0.1
S12	Rel.	0.85	0.75	0.1	0.80	0.90	0.9
	Price	0.06	0.08	0.9	0.07	0.05	0.1
S13	Rel.	0.80	0.70	0.1	0.75	0.85	0.9
	Price	0.05	0.07	0.9	0.06	0.04	0.1
S14	Rel.	0.75	0.65	0.1	0.70	0.80	0.9
	Price	0.04	0.06	0.9	0.05	0.03	0.1
S15	Rel.	0.70	0.60	0.1	0.65	0.75	0.9
	Price	0.03	0.05	0.9	0.04	0.02	0.1
S16	Rel.	0.90	0.80	0.1	0.85	0.95	0.9
	Price	0.03	0.05	0.9	0.04	0.02	0.1
S17	Rel.	0.85	0.75	0.1	0.80	0.90	0.9
	Price	0.04	0.06	0.9	0.05	0.03	0.1
S18	Rel.	0.80	0.70	0.1	0.75	0.85	0.9
	Price	0.05	0.07	0.9	0.06	0.04	0.1
S19	Rel.	0.75	0.65	0.1	0.70	0.80	0.9
	Price	0.06	0.08	0.9	0.07	0.05	0.1
S20	Rel.	0.70	0.60	0.1	0.65	0.75	0.9
	Price	0.07	0.09	0.9	0.08	0.06	0.1

Table III: Comparison of final offers generated and total utility for Concession, Tradeoff, Fuzzy and Bayesian strategies

Scenario	Concession				Tradeoff				Fuzzy				Bayesian			
	Offer		Utility		Offer		Utility		Offer		Utility		Offer		Utility	
	Rel	Pri.	U_S	U_B	Rel.	Pri.	U_S	U_B	Rel.	Pri.	U_S	U_B	Rel	Pri	U_S	U_B
S1	.80	.07	1.416		.81	.09	1.575		.75	.09	1.52		.82	.07	1.462	
	S→B/7[#1]		.668	.748	S→B/13		.835	.740	S→B/1		.835	.685	B→S/12		.699	.763
S2	.76	.06	1.32		.91	.05	1.346		.88	.05	1.348		.77	.07	1.342	
	S→ B/6		.600	.720	B→S/12		.476	.870	B→S/4		.504	.844	B→S/12		.614	.728
S3	.72	.06	1.224		.72	.07	1.342		.83	.05	1.271		.80	.05	1.240	
	S→ B/6		.532	.692	S→B/11		.665	.677	B→S/6		.472	.798	S→B/12		.469	.777
S4	.68	.05	1.128		.81	.03	1.106		.77	.05	1.221		,79	.04	1.159	
	S→B/6		.464	.664	B→S/10		.306	.800	B→S/12		.477	.774	S→B/12		.385	.774
S5	.64	.04	1.032		.62	.05	1.1		.72	.04	1.131		.99	.03	1.236	
	S→B/6		.396	.636	S→B/9		.495	.605	B→S/21		.426	.705	S→B/14		.274	.962
S6	.76	.03	1.032		.97	.02	1.16		.92	.04	1.291		.82	.04	1.142	
	B→S/7		.276	.756	B→S/10		.215	.945	S→B/21		.406	.885	B→S/12		.339	.803
S7	.76	.05	1.192		.91	.03	1.186		.87	.05	1.301		.85	.04	1.204	
	S→B/6		.456	.736	B→S/10		.296	.890	B→S/12		.467	.834	S→B/12		.379	.824
S8	.72	.06	1.224		.72	.07	1.342		.83	.05	1.271		.80	.05	1.24	
	S→B/6		.532	.692	S→B/11		.665	.677	B→S/6		.472	.798	S→B/12		.469	.771
S9	.68	.06	1.256		.68	.08	1.391		.71	.08	1.381		.67	.07	1.262	
	S→B/6		.608	.648	S→B/11		.760	.631	S→B/7		.723	.657	B→S/12		.624	.638
S10	.64	.07	1.288		.62	.09	1.420		.70	.09	1.446		.99	,07	1.568	
	S→B/6		.684	.604	S→B/9		.855	.565	S→B/13		.799	.647	S→B/14		.647	.921
S11	.86	.06	1.392		.8	.09	1.56		.80	.09	1.56		.78	.07	1.44	
	S→B/8		.581	.811	S→B/2		.83	.73	S→B/1		.830	.730	B→S/12		.716	.723
S12	.80	.06	1.352		.75	.079	1.44		.75	.08	1.44		.85	.07	1.409	
	S→B/6		.596	.756	S→B/2		.745	.695	S→B/1		.745	.695	S→B/14		.610	.798
S13	.76	.06	1.256		.70	.07	1.32		.84	.05	1.281		.80	.06	1.284	
	S→B/6		.528	.728	S→B/2		.660	.66	B→S/4		.476	.804	S→B/14		.518	.766
S14	.72	.05	1.160		.65	.06	1.2		.78	.05	1.233		.63	.05	1.07	
	S→B/6		.460	.700	S→B/2		.575	.625	B→S/10		.484	.749	B→S/12		.461	.618
S15	.68	.04	1.064		.60	.05	1.08		.72	.05	1.14		.88	.04	1.187	
	S→B/6		.392	.672	S→B/2		.490	.59	B→S/20		.440	.700	S→B/14		.33	.855
S16	.76	.04	1.096		.96	.03	1.226		.92	.05	1.3		.90	.04	1.203	
	B→S/7		.348	.748	B→S/10		.291	.935	B→S/20		.420	.880	S→B/14		.329	.874
S17	.80	.05	1.224		.92	.04	1.284		.88	.05	1.313		.81	.05	1.228	
	B→S/6		.452	.772	B→S/12		.404	.880	B→S/10		.473	.839	B→S/12		.442	.785
S18	.76	.06	1.256		.86	.05	1.306		.84	.05	1.281		.80	.06	1.285	
	S→B/6		.528	.728	B→S/12		.481	.825	B→S/4		.476	.804	S→B/14		.518	.767
S19	.72	.06	1.288		.72	.08	1.422		.71	.08	1.393		.82	.07	1.388	
	S→B/6		.604	.684	S→B/11		.755	.667	S→B/5		.734	.658	S→B/14		.612	.775
S20	.68	.07	1.32		.68	.09	1.47		.71	.09	1.46		.91	.08	1.54	
	S→B/6		.68	.64	S→B/11		.85	.621	S→B/11		.809	.650	S→B/14		.698	.845

#1 interpret S→B/7 as buyer accepts the offer from seller at the 7th round of negotiation.

Use of ANN for Classification of Starting Scenarios in the Dataset.

A. Use of Neural Network Unit for Classification

The neural network unit is used as classifier in our work [10]. We have used the ANN from Neuroph Studio [11] to create and train a class of ANN called Multi-Layer Perceptron (MLP). The network was configured with 12 neurons in the input layer, 200 neurons in hidden layer, 4 neurons in the output layer. Training set developed had 16 columns in the .csv file format. The training set was converted from .csv format to DataSet object. We have used java jdk1.7 for programming the neural network and for devising all the negotiation strategies and for classification as well. The experiments were conducted on an Intel ® Core™ i5 – 6200U CPU running @ 2.4 GHz speed. System had 8 GB of RAM and it was a 64-bit operating system on a x64 processor.

The neural network so formed was trained on the DataSet with a learning rate of 0.2 and a momentum of 0.7 along with a max Iteration limit of 2000 with a transfer function type as Sigmoid. The network so obtained was once again retrained with same learning rate 0.2 with a max iteration of 1000 on the same DataSet. The network so trained displayed the following classification results.

1) The MLP that we designed showed that while training a MLP with hidden layer size as 200 on a 500 tuple dataset showed gradual reduction in TNE(Total Network Error) from 1.7 upto 0.2 after 1640 iterations (figure 4. shown below). Also the area under ROC curve (figure 5) is nearer to 0.5 which means less accurate classification.

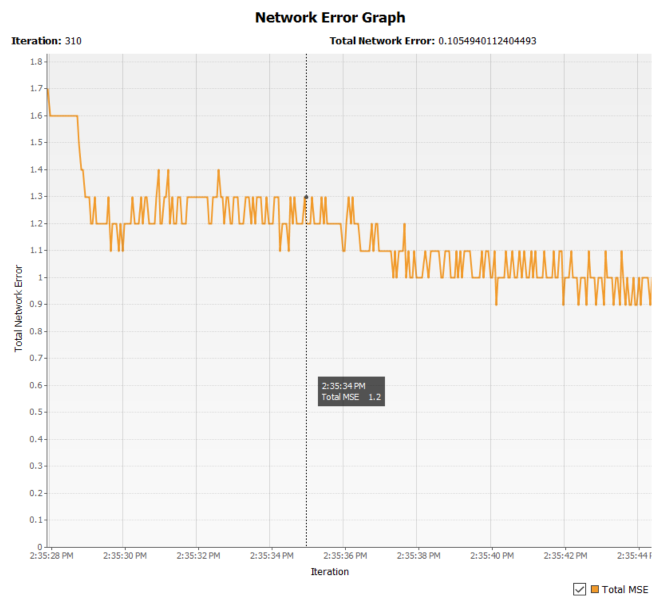

Figure 4: Multilayer perceptron training process

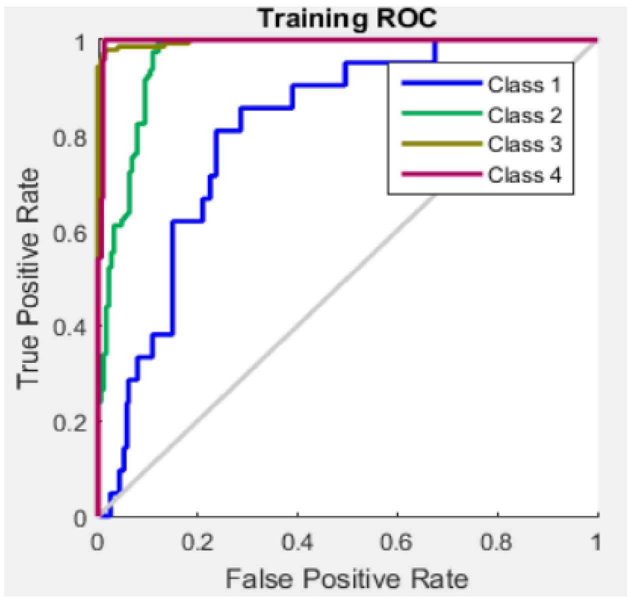

Figure 5: Multilayer perceptron ROC after training once

Further this trained MLP was further retrained with same learning rate of 0.2 and momentum of 0.7 with same 500 tuple dataset. The Total Network Error dipped to 0.1(Refer figure 6) which meant a TMSE of 0.0185 over the same 500 tuple DataSet. Also the area under ROC curve is 1(figure 7) which means a perfect accuracy. This meant the identification of the right strategy for negotiation for any given scenario.

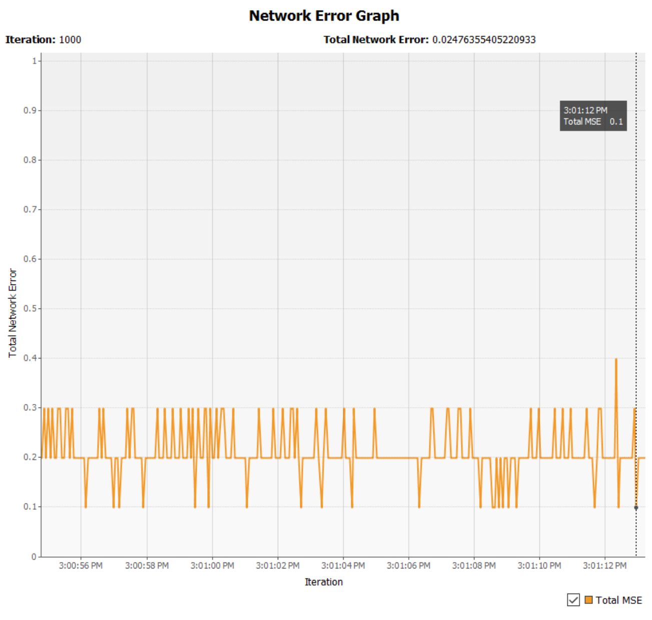

Figure 6: Multilayer perceptron retraining process

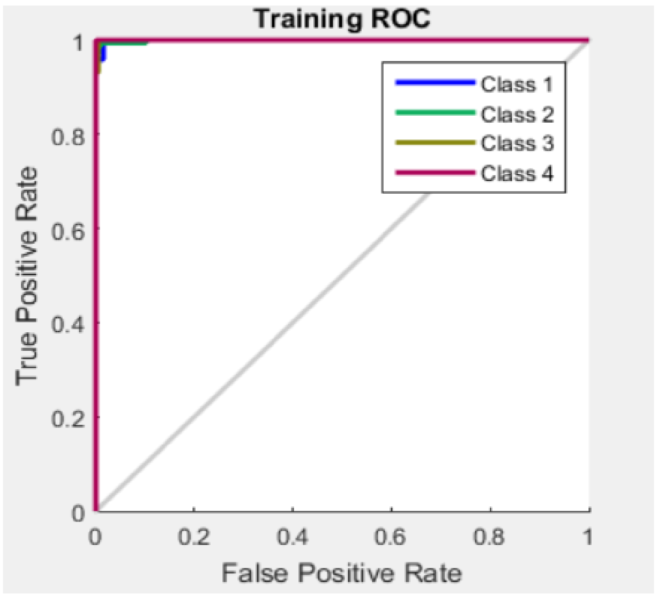

Figure 7: Multilayer perceptron ROC after retraining

2) Once the current scenario classification and identification of right strategy is done, its data like right strategy (0 0 1 0 for say S1 tuple) will be loaded/appended into the history (or log file). This leads to dynamic training set for the MLP under consideration which can give better decisions in future based on the learning received from the past history.

B. Deciding the Number of Neurons in the Hidden Layer of an ANN

The number of samples in the training set is taken as 500 for all cases in this section. The following observation can be made from figure 8 i.e. as we increase the number of neurons in the hidden layer the total mean squared error decreases. On the other hand the time taken while training (i.e. data labels on the chart) to reach smaller error values also decreases with increase in number of hidden neurons. For example for 50 hidden neurons it is 2500 iterations, whereas for 200 hidden neurons it is 1200 iterations. Therefore we can conclude that to reduce the time of training (proportional to number of iterations) we need to choose a higher value for number of neurons in the hidden layer. Hence the number of neurons in hidden layer is taken as 200 neurons in all our experiments as too many or too few hidden neurons means over fitting or under fitting of the training set.

The other observation is the difference between the two curves indicate that there is a drastic reduction in TMSE when the network is retrained with all the parameters like learning rate, momentum, Max error kept the same.

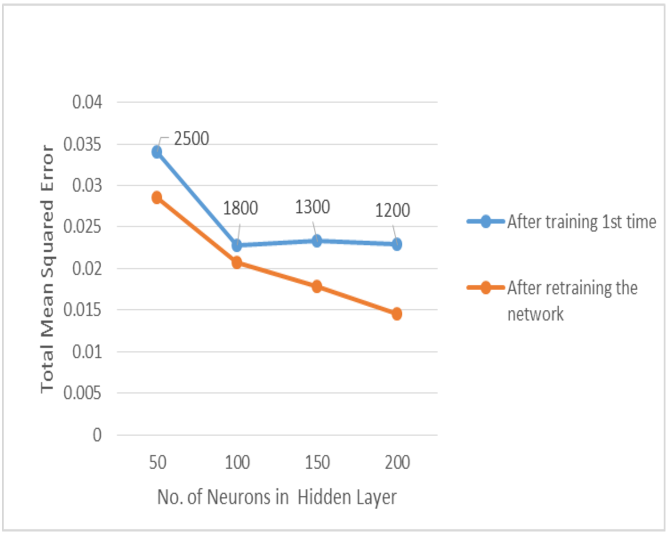

Figure 8: Errors vs. Number of Neurons in Hidden Layer

C. Deciding the Learning Rate and Momentum of ANN

Figure 9 and Figure 10 depicts the behavior of neural network for different datasets. The one in Figure 9 is for dataset with 125 samples and Figure 10 represents the dataset with 500 samples. While the former one has 150 neurons in the hidden layer, the later one has 200 neurons in hidden layer. The data labels on this curve represent the number of iterations required before TMSE settles to a low value at that particular learning rate. It is clear from figure 9 that learning rate of 0.5(MSE 0.07288) is moderate with respect to speed of classification (30 epoch) whereas learning rate of 0.2(MSE 0.2266) is poor with respect to speed of classification (290 epoch). A learning rate of 0.8(MSE 0.3997) is excellent with respect to speed of classification (5 epoch).

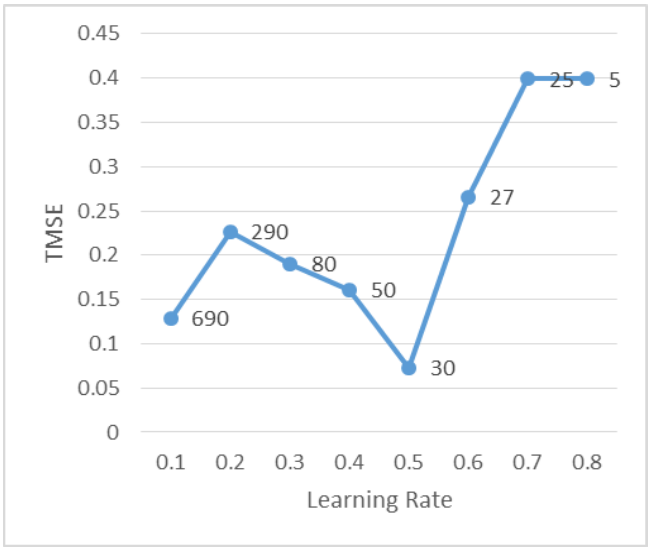

Figure 9: Learning Rate vs. Total Mean Squared Error (dataset with 125 samples)

Similarly in figure 10, the learning rate of 0.2(MSE 0.0229) is found to be ideal. At moderate learning rate the network tends to generalize well and avoid over fitting. In our work, we have heuristically chosen a learning rate of 0.2 with a momentum of 0.7.

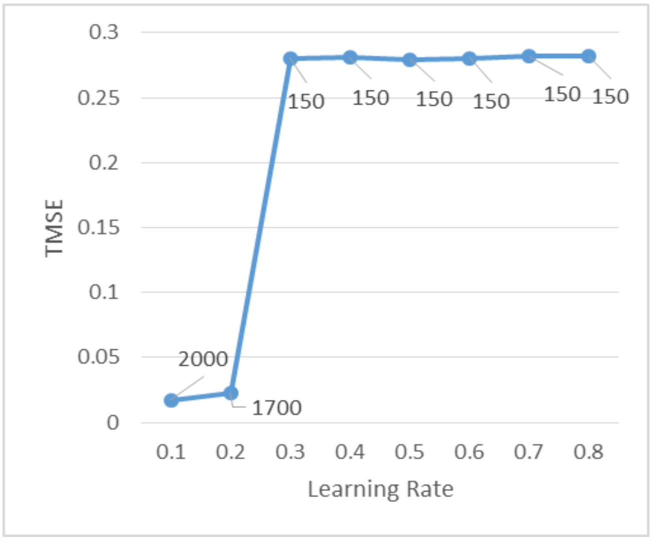

Figure 10: Learning Rate vs. Total Mean Squared Error (dataset with 500 samples)

CONCLUSION AND FUTURE WORK

The main problem area of cloud computing that of QoS-Assurance was identified initially, then a solution to the problem by QoS-Negotiation (with two issues reliability and price) was attempted in this work. Firstly four new negotiation strategies were developed that of concession, tradeoff, Fuzzy based and Bayesian based and were implemented in java. Secondly their efficiency with respect to each other was compared using standard starting scenarios. The one with the best utility-cum-time outcome was tagged as the best-strategy for that particular scenario. Thirdly creation of a dataset for training the neural network to identify the right strategy in future was created. Fourthly neural network was trained over the dataset created which was then deployed in a cloud context to identify the right strategy. As a future work the dataset used to train can be further enhanced by suggesting different starting scenarios and extending them. Also genetic programming may be used to enhance the Bayesian learning technique of negotiation.

REFERENCES

[1] Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation computer systems. 2009 Jun 30;25(6):599-616.

[2] Freitas AL, Parlavantzas N, Pazat JL. An integrated approach for specifying and enforcing slas for cloud services. InCloud Computing (CLOUD), 2012 IEEE 5th International Conference on 2012 Jun 24 (pp. 376-383). IEEE.

[3] Faratin P, Sierra C, Jennings NR. Negotiation decision functions for autonomous agents. Robotics and Autonomous Systems. 1998 Sep 30;24(3):159-82.

[4] Jennings NR, Faratin P, Lomuscio AR, Parsons S, Wooldridge MJ, Sierra C. Automated negotiation: prospects, methods and challenges. Group Decision and Negotiation. 2001 Mar 1;10(2):199-215.

[5] Zheng X, Martin P, Brohman K. Cloud service negotiation: Concession vs. tradeoff approaches. InProceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012) 2012 May 13 (pp. 515-522). IEEE Computer Society.

[6] Wang X, Shen X, Georganas ND. A fuzzy logic based intelligent negotiation agent (FINA) in ecommerce. In2006 Canadian Conference on Electrical and Computer Engineering 2006 May (pp. 276-279). IEEE.

[7] Gwak J, Sim KM. Bayesian learning based negotiation agents for supporting negotiation with incomplete information. InProceedings of the international multiconference of engineers and computer scientists 2011 (Vol. 1, pp. 163-168).

[8] Zeng D, Sycara K. Bayesian learning in negotiation. International Journal of Human-Computer Studies. 1998 Jan 1;48(1):125-141.

[9] Kar A. Stock Prediction using Artificial Neural Networks. Dept of Computer Science and Engineering, IIT Kanpur.

[10] Zhang S, Ye S, Makedon F, Ford J. A hybrid negotiation strategy mechanism in an automated negotiation system. In Proceedings of the 5th ACM conference on Electronic commerce 2004 May 17 (pp. 256-257). ACM.

[11] Neuroph Studio documentation. http://neuroph.sourceforge.net/documentation.html. Date accessed 01/01/2017

[12] Bui HH, Venkatesh S, Kieronska D. Learning other agents' preferences in multi-agent negotiation using the bayesian classifier. International Journal of Cooperative Information Systems. 1999 Dec;8(04):275-93.

[13] Sim KM, Guo Y, Shi B. Adaptive bargaining agents that negotiate optimally and rapidly. In Evolutionary Computation, 2007. CEC 2007. IEEE Congress on 2007 Sep 25 (pp. 1007-1014). IEEE.

[14] Sim KM, Guo Y, Shi B. BLGAN: Bayesian learning and genetic algorithm for supporting negotiation with incomplete information. IEEE Transactions on

Systems, Man, and Cybernetics, Part B (Cybernetics). 2009 Feb;39(1):198-211.

[15] Karim MS, Pierluissi J. Fuzzy Driven Multi-issue Agent Negotiation on Electronic Marketplace. In Advances in Computing and Information Technology 2013 (pp. 239-248). Springer Berlin Heidelberg.

[16] Kowalczyk R. Fuzzy e-negotiation agents. Soft Computing. 2002 Aug 1;6(5):337-47.

[17] Lee HS, Yeh CH. Fuzzy multi-criteria decision making based on fuzzy preference relation. In Knowledge-Based Intelligent Information and Engineering Systems 2008 (pp. 980-985). Springer Berlin/Heidelberg.

[18] Lee RS. iJADE Negotiator—An Intelligent Fuzzy Agent-Based Negotiation System for Internet Shopping. Fuzzy-Neuro Approach to Agent Applications: From the AI Perspective to Modern Ontology. 2006:287-318.

[19] Lawrence S, Giles CL, Tsoi AC, Back AD. Face recognition: A convolutional neural-network approach. IEEE transactions on neural networks. 1997 Jan;8(1):98-113.

[20] Hai TS, Thuy NT. Image classification using support vector machine and artificial neural network. International Journal of Information Technology and Computer Science (IJITCS). 2012 May 2;4(5):32.

[21] Carbonneau R, Kersten GE, Vahidov R. Predicting opponent's moves in electronic negotiations using neural networks. Expert Systems with Applications. 2008 Feb 29;34(2):1266-73.

[22] Beam C, Segev A. Automated negotiations: A survey of the state of the art. Wirtschaftsinformatik. 1997 May;39(3):263-8.

[23] Faratin P, Sierra C, Jennings NR. Using similarity criteria to make issue trade-offs in automated negotiations. artificial Intelligence. 2002 Dec 31;142(2):205-37.

[24] Kamath BS, D'Souza R. Extending bayesian learning based negotiation technique to a two issue bilateral negotiation in cloud. International Journal of Computer Engineering & Technology. 2017 Feb 27;8(1):51-59.