

# Parameter Training in MANET using Artificial Neural Network

**Baisakhi Chatterjee**

Computer Science & Engineering Department, Institute of Engineering and Management, Kolkata, City, 700 091, India  
E-mail: [baisakhi.chatterjee95@gmail.com](mailto:baisakhi.chatterjee95@gmail.com)

**Himadri Nath Saha**

Computer Science & Engineering Department, Institute of Engineering and Management, Kolkata, City, 700 091, India  
E-mail: [himadri@iemcal.com](mailto:himadri@iemcal.com)

Received: 10 June 2019; Accepted: 27 August 2019; Published: 08 September 2019

**Abstract**—The study of convenient methods of information dissemination has been a vital research area for years. Mobile ad hoc networks (MANET) have revolutionized our society due to their self-configuring, infrastructure-less decentralized modes of communication and thus researchers have focused on finding better and better ways to fully utilize the potential of MANETs. The recent advent of modern machine learning techniques has made it possible to apply artificial intelligence to develop better protocols for this purpose. In this paper, we expand our previous work which developed a clustering algorithm that used weight-based parameters to select cluster heads and use Artificial Neural Network to train a model to accurately predict the scale of the weights required for different network topologies.

**Index Terms**—Clustering, MANET, Artificial Intelligence, Artificial Neural Network.

## I. INTRODUCTION

Wireless ad hoc networks [1], also known as Mobile Ad hoc Networks (MANET) are self-configuring infrastructure less networks consisting of mobile nodes that can communicate with each other in order to send and receive messages. Nodes in such a network are autonomous [2] and are free to move in any direction and even join or leave the network thus forming highly dynamic topologies.

MANETs differ from traditional wireless networks due to their decentralized system [3]. In a traditional network, nodes send messages to a base station and the base station, in turn, relays these messages to their destination nodes. Thus, these networks are relatively fixed. In MANETs, however, nodes are free to move in any direction and can communicate directly with other nodes by temporarily established links. Fig.1 is an example of centralized wireless network while Fig. 2 shows an ad hoc network [4].

The advent of internet and wireless networks has made MANETs a popular area of study. Because of their lack of infrastructure, ad hoc networks have found use in a

variety of fields, including environmental monitoring, military communications and disaster relief[5,6]. Due to the largely dynamic nature of MANETs, efficient routing can be difficult and it has been observed that imposing a hierarchy leads to better scaling [7]. A commonly used routing method is clustering. In our previous paper [8] we have proposed a unique algorithm that chooses efficient cluster heads based on degree of connectivity, battery power and packet drop rate. Our goal in this paper is to extend the previous algorithm by training the parameters used.

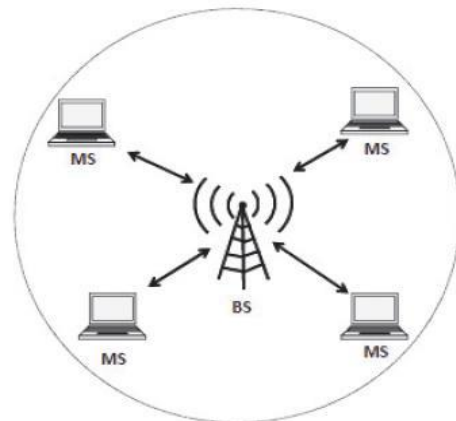


Fig.1. A base station (BS) and its network cell

The proposed model aims to recognize the weights of the factors used in said paper using machine learning, specifically, artificial neural networks (ANN). ANNs are supervised mathematical learning tools inspired by biological neural networks. The network is connected with a vast number of nodes, which are termed as neurons. The connections between the nodes emulate the synaptic connections found in biological entities. In general, ANNs follow the principle of Hebbian Learning [9] which states that the connections between two neurons might be strengthened if the neurons fire simultaneously or, in a more concise way, “cells that fire together, wire together”[10]. Essentially, this means, that neurons which are closely linked with other neurons tend to act on the

same input. Since ANNs can be trained to recognize patterns in data, they are incredibly useful for analysis of mapped information. By exploiting this functionality, we aim to create a secure, robust, energy efficient routing algorithm.

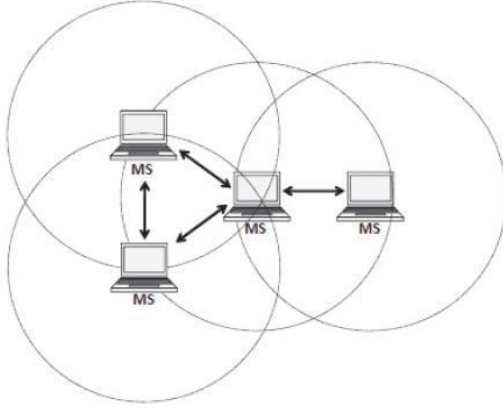


Fig.2. Mobile stations in Ad hoc mode

The rest of this paper is organized as follows: Section II features relevant works. Section III expounds on the principle behind our work. Section IV discusses results obtained and Section V presents our conclusion and discusses future scope of research.

## II. RELATED WORKS

Machine learning has been an integral part of wireless sensor networks since their inception. Machine learning refers to a vast array of methods that help the machine ‘learn’, i.e., progressively improve its performance, to understand and categorize vast amounts of data [11]. The data can either have pre-labeled responses, which is known as supervised learning, or can be unlabeled, called unsupervised learning, where the machine tries to infer meaning from similarities in data points. A number of works have utilized machine learning concepts to develop efficient algorithms for MANETs.

The mobile nature of wireless sensors makes it difficult for researchers to accurately develop schemes for efficient routing, network management, event detection etc. Feng and Chang [12], in their paper, have developed a technique using a hierarchical support vector machine (H-SVM) in order to predict future locations of nodes, as well as, to estimate the amount of noise that will be present in a network so that effective demodulation algorithms can be implemented. The choice of support vector machine was due to it requiring less computation.

A support vector machine can classify n-dimensional data into binary labels by construction of a hyperplane. An H-SVM has multiple levels with a finite number of SVMs [13] on each level that can similarly part the data. Thus, the H-SVM extends the SVM’s ability by being able to classify multi-class data. H-SVM with  $m$  levels has  $2m-1$  SVM classifiers and they are arranged as those  $2m-1$  branch nodes of a binary tree with height  $m$ . According to the authors, H-SVM can classify data in a

distributed manner and use simple network information which reduces computational needs. However, the SVM classification error at each stage is assumed to be the same for this work, which will not be the case in real life data. Thus, this model leaves room for error which accumulates at every stage.

Choi and Hossain [14] use a Hidden Markov Model (HMM)[15] to estimate channel parameters in a cognitive radio (CR) system. The goal of the paper is to effectively identify all parameters affecting a channel so that a secondary user (SU) can take advantage of an idle primary user (PU) without hindering any transmission by the PU. Thus, PU traffic pattern, channel gain from PU to SU, and noise density at SU constitute the parameters. Since the PU is modeled as a Markov process, the underlying states representing PU activity is hidden from the SU and the latter can only sense the results. Bayesian learning is utilized to infer these hidden states. Bayesian learning constitutes of deducing the probability distribution of the target variables which are determined using the input sequence. The EM algorithm is used for this task. While this paper identifies a unique approach, it also highlights that the HMM model is not identifiable in some cases due to which the EM algorithm fails. It is also specific only to HMM models and problems must be represented as such for use which makes it non-versatile.

Donohoo et al. [16] have studied a variety of techniques to analyze data from wireless devices in order to predict energy efficient protocols. The list of explored approaches includes linear discriminant analysis [17], linear logistic regression [18], k-nearest neighbor [19], support vector machines [13] and neural networks [20]. Linear discriminate analysis (LDA) used a Bayesian approach. The input data was considered as random variables of a prior distribution. It proved to be a fast and unbiased algorithm, but did not fare well for data with very different underlying covariant matrices. Similar to LDA, linear logistic regression (LLR) is a technique based on Bayes’ theorem which derives a linear model that directly predicts the output variable based on some input variable. However, LLR, unlike LDA, tries to maximize the likelihood of data from a set of classification samples. It is sensitive easy to interpret but possess challenges when the data is complicated.

Support Vector Machine, as we have already discussed, is a powerful scheme which can use multiple attributes to classify data, but is limited to only binary classifications. K-nearest neighbour is a non-parametric unsupervised learning algorithm. It gives fairly accurate results, but is slow and needs all training data at testing phase leading to both time and memory problems. Thus, both these results do not yield much scope for us. Neural Networks on the other hand, showed great promise, getting results with accuracy up to 90%, showing that they perform well where data is imprecise, as is in case of most wireless networks.

Alnwaimi et al. [21] have applied reinforcement learning models to self-configure/optimize femtocells in a distributed heterogeneous network (HN). The HN is represented as a “layer of closed-access LTE femtocells

(FCs) overlaid upon an LTE radio access network". The FCs, being self-organizing, can sense radio environment and tune their parameters to avoid interference in the network. The model solves resource allocation problems, as well as, interference likelihoods by being able to identify unused slots for spectrum allocation and being able to select sub-channels from the available pool. The authors envisioned that this would lead to better quality of service (QoS).

Gispan et al. [22] uses linear regression to estimate a non-random parameter vector. A distributed network has  $N$  sensors which detect some vector  $v$  that determines a non-random vector  $\theta$ . A linear regression model is used for this analysis. For each sensor  $i$ ,  $x_i = H_i\theta + v_i$ . The data is compressed before being sent to a fusion center which uses the data to compute the value of  $\theta$ . Linear regression is simple, easy to implement, however transferring data to the FC comes at a cost, and while this paper has assumed lossless transmission, that will not always be the case, leading to errors. Moreover, the parameter is computed at one center which makes this algorithm useless for mobile decentralized nodes that are found in MANETs. Such nodes are too dynamic to consider equipping one as a fusion center, not to mention, that back and forth of data transmission would lead to exhausting battery life.

The mobile nature of MANETs make it difficult to obtain efficient routing routes since neighbours continuously keep changing. New nodes are added, old ones removed, or move out of range, and as mobility increases efficiency decreases. To curb this, Cadger et al. [23] devised a novel algorithm that uses machine learning techniques to predict the movement of nodes. The theory was that if the algorithm could predict future locations of mobile nodes, they could plan accordingly and improve overall accuracy. The neural network algorithm was designed and tested using FANN which is an open source C library for creating multilayer perceptron NNs. The authors used 7 input neurons, 15 hidden neurons, 1 hidden layer, and 2 output neurons. Results and data suggest that this algorithm showed considerably positive results and could even be adapted for different parameters. This also suggests that mobility is an important criterion for network topology.

With the recent surge in wireless heterogeneous networks, there has been an increase in utilization of machine learning techniques for finding efficient routes so that nodes can transmit data effectively. Tang et al. [24] have devised a unique approach which deals in controlling traffic in the wireless backbone. Traditional routing protocols do not utilize previously learned data about network congestion to circumvent such problems in the future. Using Convolutional Neural Networks (CNN) the authors aimed to teach the network how to find better routes for their path. The model was evaluated by the authors using extended simulations and was proved to showcase considerably lower delays in delivery time. As previously stated, deep learning mechanism allowed the authors to include varied inputs and allow the model to learn from real world data. This further supports out

theory that in a dynamic mobile environment, neural networks are optimum to train data as required.

In our earlier work [8], we have proposed Acceptability Based Routing Protocol (ABRP) which is a secure, hierarchical, energy efficient routing protocol that aims to select cluster heads based on an acceptability score. Acceptability score was calculated as  $AC = \alpha.D + \beta.P + \gamma.S$  where, AC is Acceptability Score;  $\alpha, \beta, \gamma$  are Weight Factors and D, P and S are the number of neighbours, battery power and security factors respectively. While D, P and S differ for each node, the weight factors are selected based on the estimation of their relative importance for the network. This routing protocol works by dividing groups of nodes into clusters and determining a clusterhead to route messages. If the acceptability score of a clusterhead falls below the threshold value defined in the paper, reclustered would be initiated. The drawback of this algorithm was that the values of the parameters,  $\alpha, \beta, \gamma$  were selected based on their estimated importance. While this algorithm showed good results, we believe it could be further improved by training the parameters using machine learning based on certain input criteria.

### III. THE PROPOSED MODEL

In this section we aim to revisit our original algorithm, Acceptability Based Routing Protocol (ABRP) [8] and extend it to be able to determine the best parameters for cluster selection. The goal of this paper is to predict the weight factors for different networks such that the best possible weight factors will be selected. For this, we take characteristics of the network as the input parameters and use artificial neural networks to train the model.

Since ANNs are supervised learning tools, they are incredibly useful in learning patterns with already mapped data. For this project, we have chosen to implement a multilayer perceptron. An MLP is composed of a vast number of connected units arranged in layers. The first layer is always the input layer while the last layer is always the output layer. The inner layers are referred to as the hidden layers and are responsible for the bulk of the processing. MLPs work well when the goal is to map random features to a set output and has good accuracy and speed of learning [25].

In our model, we selected 5 parameters for input, viz, number of nodes, average number of neighbours, mobility, battery power and packet forward rate. A diagrammatic representation of the artificial neural network created as a result is shown in Fig.3. As we have mentioned in [8], battery power and packet forward rate are important parameters for ensuring a low-cost secure routing protocol. Number of neighbours determine the cluster head which is why this parameter has been included. Mobility and number of nodes determine the kind of network we are dealing with [23] and map well to ANNs. The output layer consists of 8 nodes, each having a different  $\alpha, \beta, \gamma$  combination. Since we know, it is easy to ascertain possible values of  $\alpha, \beta, \gamma$  using combinatorics. The result was 69 possible cases. These cases included 0

values, a sample of which can be seen in Table 1. Since any one weight factor having 0 relevance would mean that said factor would become irrelevant for our network, we restricted our possible outputs to exclude zero values because all factors are essential. Thus, we re-computed the table eliminating all zero values. This has given us 36 possible factors as seen in Fig.4 that we shall now work with.

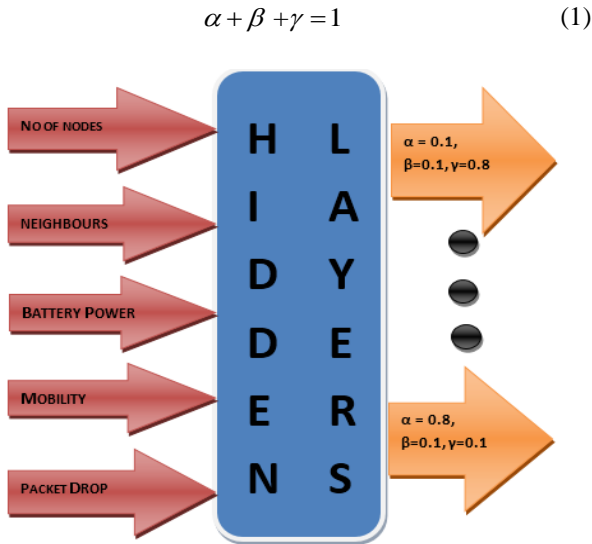


Fig.3. An example of an artificial neural network

Table 1. A sample of possible values of  $\alpha$ ,  $\beta$ ,  $\gamma$  including 0 values.

Serial No	$\alpha$	$\beta$	$\gamma$
1	1	0	0
2	0	0	1
3	0.1	0	0.9
4	0.2	0	0.8
5	0.3	0	0.7
6	0.4	0	0.6
7	0.5	0	0.5
8	0.6	0	0.4
9	0.7	0	0.3
10	0.8	0	0.2

Each node acts as a decentralized processing unit, which, based on its own data activates nodes of the next layer. Traditionally a node is activated by a weighted sum of all its parent nodes. Instead of the binary connections of activation versus no activation we find in biological networks, nodes in an artificial neural network take on probabilistic values. For example, let nodes be denoted by the notation  $a_{ij}$  where  $i$  refers to the layer corresponding to the node while  $j$  refers to the node number. Thus, the first node in the input layer will be named  $a_{00}$ , while the last node in the same layer will be named  $a_{04}$  for this network. The first node in the next layer will similarly be named  $a_{10}$ . The value  $a_{10}$  will be a weighted sum of all the values of its parents, viz,  $a_{00}$ ,

$a_{01}, \dots, a_{0n}$  where  $n$  is the number of nodes in the layer. Since the sum will be greater than one, we use a normalizing function so that the value appears between 0 and 1. In this paper we used ReLU [26] (denoted by  $\sigma$ ) for this purpose since ReLU has been shown to achieve better result than sigmoid [27]. The weights are the values assigned to the arrows linking each nodes. We use the notation  $w_{ijk}$  where the arrow connects node  $j$  of the layer  $i$  to node  $k$  of the layer  $i+1$ . It is by adjusting these weights, that we determine the strength of the connections and how firing of one neuron will activate the next one. Mathematically, in case of node  $a_{10}$ ,

$$a_{10} = \sigma \left( \begin{aligned} &a_{00} \cdot w_{000} + a_{01} \cdot w_{010} + a_{02} \cdot w_{020} \\ &+ a_{03} \cdot w_{030} + a_{04} \cdot w_{040} \end{aligned} \right) \quad (2)$$

Where  $a_{10}$  is a node the zeroth node on the first layer and its value is computed by normalizing the sum of all the values in the previous layer multiplied with their weights. As we can see, for each node,  $w_{ijk}$  may just be written as  $w_j$ . Thus, we can rewrite this equation as

$$a_{10} = \sigma \left( \begin{aligned} &a_{00} \cdot w_0 + a_{01} \cdot w_1 + a_{02} \cdot w_2 \\ &+ a_{03} \cdot w_3 + a_{04} \cdot w_4 \end{aligned} \right) \quad (3)$$

In general,

$$a_{ij} = \sigma \left( \sum_{j=0}^n a_{(i-1)j} \times w_j \right) \quad (4)$$

Where  $a_{ij}$  is the  $j$ th node on  $i$ th layer and  $w_j$  is the weight of the  $j$ th node in the  $i-1$  layer. In this manner all the nodes in the  $i$ th layer are assigned values. This continues for all successive layers and the neurons keep firing till finally the output layer is activated. Initially, the model starts with generic weights and in each successive case the weights are refined to better suit the predetermined result. A cost function determines how close the result was to the actual value. This process occurs multiple times till the results converge and weights in each layer obtain the best possible value to suit the output. In our case each of the 36 output nodes was assigned a probability value. To train our model, we use a simple mean squared error (MSE) as cost function. MSE is attractive because it is simple to use, parameter free and inexpensive to compute. All  $l_p$  norms are valid distance metrics in RN, the  $p = 2$  case, in particular, being the ordinary distance metric in  $N$ -dimensional Euclidean space [28]. Thus, we compute,

$$MSE = \sum_{j=1}^n a_{Lj} \quad (5)$$

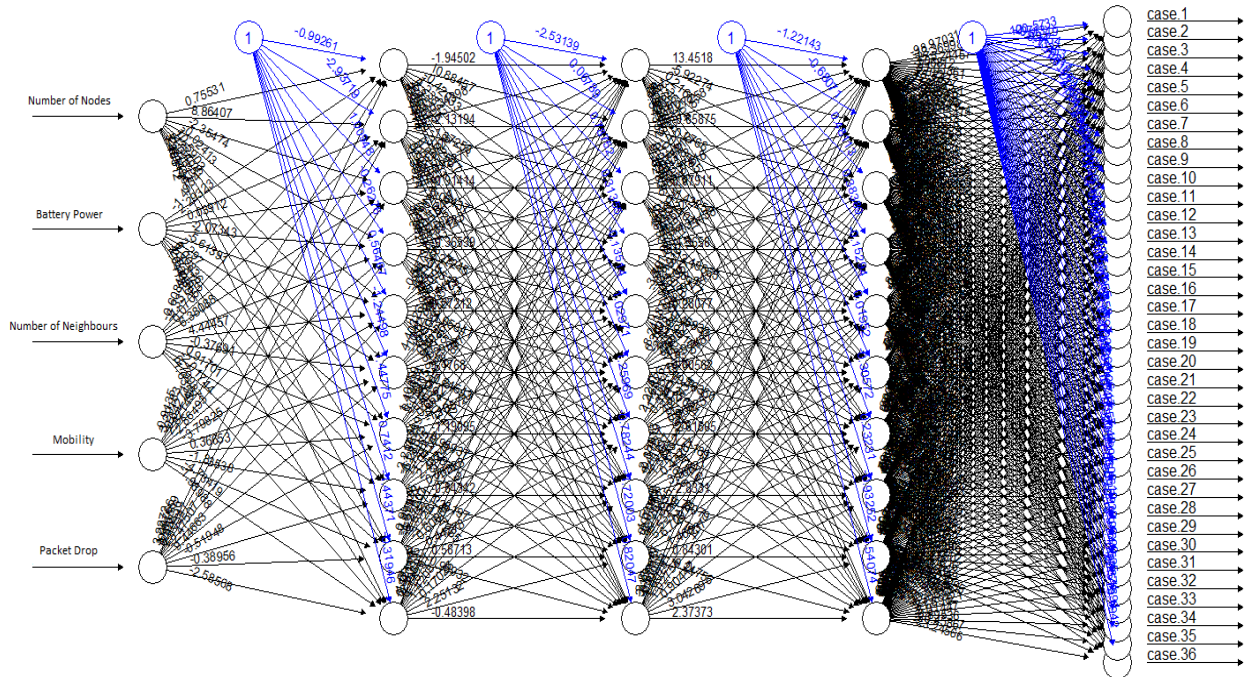


Fig.4. Plot of the neural network formed in R by the algorithm, showing the input parameters, output cases and weights

Where  $L$  refers to the last layer,  $n$  is number of outputs and  $\hat{a}$  is the expected output value. The sum gives us the total error for the network. For example, if we consider only the first four of the 36 possible cases and assume value 3 is the correct answer, we may get something like Table 2 in the first iteration. The error in this scenario would be the sum of all the squares of differences between observed value and expected value.

Table 2. A simple example for calculating MSE

Serial No.	$\alpha$	$\beta$	$\gamma$	Observed Value	Expected Value
1	0.1	0.1	0.8	0.3	0
2	0.1	0.2	0.7	0.2	1
3	0.1	0.3	0.6	0.9	0
4	0.1	0.4	0.5	0.4	0

From this table we can calculate the MSE as,

$$MSE = (0.5 - 0.0)^2 + (0.2 - 0.0)^2 + (0.3 - 0.0)^2 + (0.4 - 0.0)^2 = 0.94 \quad (6)$$

Similarly, we calculate MSE for all possible cases and compute the average. With each new dataset, we then try to minimize this cost function using the gradient descent technique [29]. In an  $n$ -dimensional space consisting of all the weights and biases, by changing the inputs, we minimize the cost function, and thus, we reduce the error, thereby, allowing the machine to learn. This model consisting of all the parameters is then capable of predicting the best weight factors for our network.

We have used the algorithm from our earlier work to simulate test cases. This has been done using Global Mobile Information System Simulator (GloMoSim) 2.03. In a comparison between GloMoSim, NS2, NS3 and

Omnet++, GloMoSim had the best CPU utilization for large scale networks and was almost comparable to NS3 in terms of memory usage and computational speed. However, NS3 is still considered to be very new and widely lacks credibility. Furthermore, GloMoSim has high scalability and supports much larger networks than NS2 or NS3[30, 31]. The combination of all of these factors led us to choose GloMoSim.

Using this simulator, different scenarios were tested which included different cases of  $\alpha$ ,  $\beta$  and  $\gamma$ . One case was marked as best in each scenario. Metrics like packet drop, number of clusters formed, and residual battery power have aided in this choice. This knowledge was then used to pick five parameters to be used as input, viz., Number of Nodes, Average Number of Neighbours, Average Mobility of Nodes, Battery Power of Network and Packet Drop Rate. The network was implemented using neuralnet package in R, having 5 inputs, 36 outputs and 3 hidden layers with 10 neurons each.

Following this, the results predicted by our algorithm was utilized to simulate networks of different sizes to test our results. Once again, GloMoSim was our simulator of choice. The networks we simulated ranged from 10 nodes to 20. For each case we chose the parameters we had previously selected and the new values. In all cases the training was shown to be effective.

#### IV. RESULTS

The data was divided such that 75% was used for training and 25% for testing. Fig.3 visualizes the neural network used to achieve our results. The input nodes are, as can be seen, are the ones previously mentioned and each of the 36 cases are treated as output. In order to further visualize the data, Fig.5 shows a comparison

between real and predicted values that were obtained in the course of our work. The graph thus plotted only shows the cases that were chosen, and not the cases with 0 values in order to keep the plot simple. X-axis

corresponds to the different network configurations test, while Y-axis pertains to the different possible parameter values.

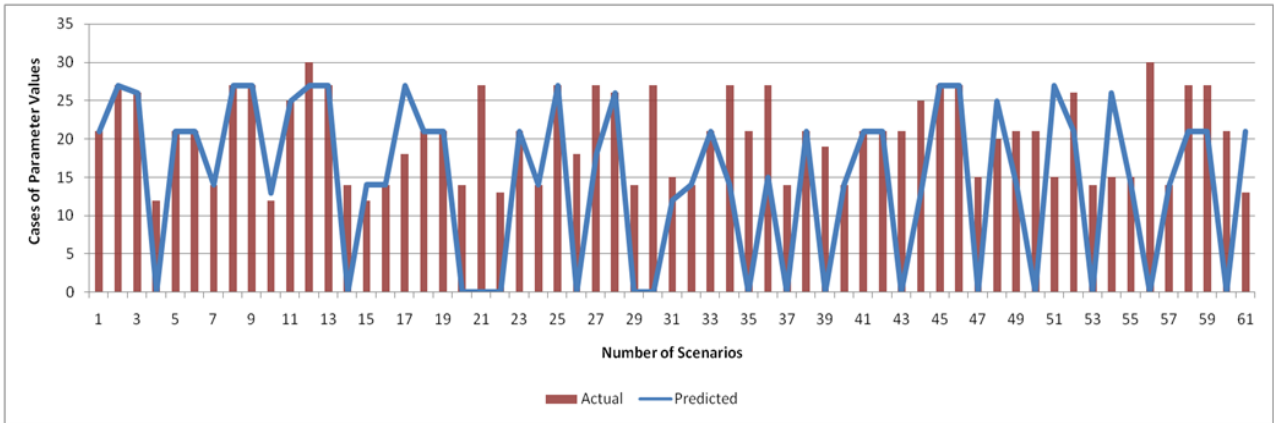


Fig.5. Predicted vs Real Values.

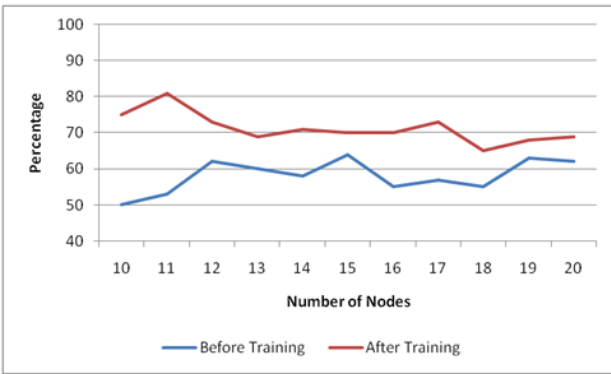


Fig.6. Percentage of battery power in network before data was trained vs. after training

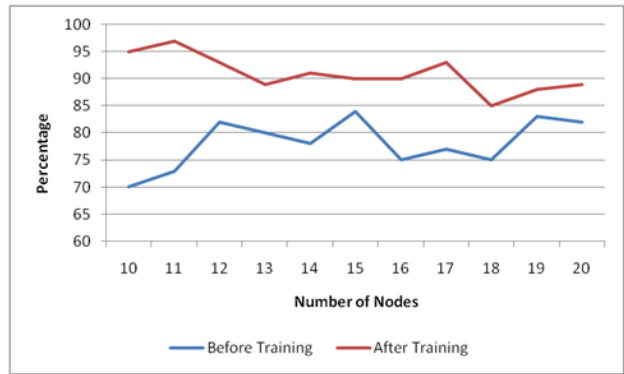


Fig.7. Security factor in network before data was trained vs. after training

A. Accuracy

We denote Accuracy as ACC where

$$ACC = \frac{TP + TN}{P + N} \tag{7}$$

Where TP= true positive, TN=true negative, P=positiveinstances, N=negative instances. Here, basically the termspositive and negative tells us about the researcher’s predictionor what one can say expectation. True and positive refers tothe fact that whether the prediction is same as that of theobservation.

B. Precision

$$Precision = \frac{TP}{TP + TF} \tag{8}$$

Precision is basically the number of actual positives among those which has been reported as positive.

C. Recall

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

Recall is the fraction of positives which has been recognized as such.

To determine this, we construct a simple confusion matrix. A confusion matrix is a n x n matrix that is a useful guide to determine the performance of a classifier by tabulating number of predictions inrelation to actual cases. In our case, we will have a 36 x 36 matrix. True

Positive will be all cases where the network predicted 1 and the answer was 1. True Negative will be cases where the network predicted 0 and the actual label was 0. False Positive is when the prediction was 1 but the actual answer was 0 and False negative occurs when actual answer was 0 but the classifier predicted 1.

Using this, we have found that our model displays a 83.3% accuracy. This means that 83.3% of the predicted data was correct. When given information about the

characteristics and topology of the network, our model could accurately predict the best course in most cases. Precision is 75.7% and Recall is 53.2%. Our high precision value shows that the model can correctly recognize a good number of relevant results. Recall results shows that a significant number of data has been correctly identified and that we have been successful in predicting a large number of results. This is good since it shows that our algorithm can be trained to correctly identify the best parameters for our routing protocol

To further verify our results, we have tested our predicted values using GloMoSim. In order to check energy efficiency, we have compared the percentage of remaining battery power in the node for each network with that of the remaining battery power when untrained parameter values were used. This is shown in Fig.5. Security factor was also compared for both cases. Security factor measures the packet drop rate which is used to determine how efficiently packets are transmitted in the network. Fig.6 shows the results. As can be seen in both cases, our current algorithm shows superior results.

#### V. CONCLUSION

ABRP is a secure, energy-efficient clustering routing protocol which utilizes the infrastructure-less nature of mobile ad hoc networks. The object of this paper was to expand on ABRP and use machine learning techniques, viz., artificial neural networks to train weight parameters -battery power, security factor and degree of neighbours. This algorithm determines the importance of each parameter for a particular type of network in calculating acceptability of a node to become cluster head.

After training, our model was able to analyze input factors like battery power, mobility, packet drop rate, total number of nodes and degree of neighbours in order to correctly predict the best weight parameters for a significant number of cases. The algorithm showed a high rate of success in predicting output values and could accurately identify best weight parameters that would provide the most efficient routing. To determine this, we have further tested the predicted parameters using GloMoSim. In all cases, training the parameters significantly improved the performance of the algorithm proving the efficacy of our training model. Thus we can conclude that supervised learning techniques, especially artificial neural networks, can be utilized to identify the best routing algorithm for a particular network.

An avenue we haven't explored is optimizing the number of hidden layers and the neurons present in each layer in order to better predict data. As is obvious, more number of layers increases processing time, which, as the number of data entries increase, could lead to problems. However, too few layers could lead to underfitting. A number of optimization algorithms have also been proposed [32], but wisdom on the subject is varied. Our future work would include determining the best possible architecture of the neural network for greatest efficiency at least cost.

#### REFERENCES

- [1] C.-K. Toh, *Wireless ATM and Ad-Hoc Networks*. Springer US, 1997.
- [2] H. N. Saha, D. Bhattacharyya, P. K. Banerjee, et al. "Implementation Of Personal Area Network for Secure Routing in MANET by using Low Cost Hardware" *Turkish Journal Of Electrical Engineering & Computer Sciences*, Vol 25, Number 1, pp.235-246, January 2017, SCI E (Thomson Reuters), ESCI and Web of Science (Thomson Reuters), Scopus (Elsevier).
- [3] M. M. Zanjireh and H. Larijani, "A Survey on Centralised and Distributed Clustering Routing Algorithms for WSNs," in 2015 IEEE 81st Vehicular Technology Conference (VTC Spring), 2015 doi: 10.1109/VTCSpring.2015.7145650
- [4] R. Singh, H. N. Saha, D. Bhattacharyya, and P. K. Banerjee, "Administrator and Fidelity Based Secure Routing (AFSR) Protocol in MANET," *Journal of Computing and Information Technology*, vol. 24, no. 1, pp. 31–43, Mar. 2016.
- [5] A. Vasiliou and A. A. Economides, "MANETs for environmental monitoring," 2006 International Telecommunications Symposium, Fortaleza, Ceara, 2006, pp. 813-818. doi: 10.1109/ITS.2006.4433383
- [6] Fajardo, J. T. B., & Oppus, C. M., "A mobile disaster management system using the Android technology", *WSEAS Transactions on Communications*, vol. 9, pp. 343–353, June 2010.
- [7] C. R. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," in *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 7, pp. 1265-1275, Sep 1997. doi: 10.1109/49.622910
- [8] H.N.Saha, A. Chatterjee, B. Chatterjee, A. K. Saha and A. Santra, "Acceptability Based Clustering Routing Protocol in MANET," 2018
- [9] Hebb, D.O. (1961). "Distinctive features of learning in the higher animal". In J. F. Delafresnaye (Ed.). *Brain Mechanisms and Learning*. London: Oxford University Press.
- [10] Hebb, D. O. (1940). "Human Behavior After Extensive Bilateral Removal from the Frontal Lobes". *Archives of Neurology and Psychiatry*. 44 (2): 421–438. doi:10.1001/archneurpsyc.1940.02280080181011
- [11] BA. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers. I," in *Computer Games I*, Springer New York, 1988, pp. 335–365.
- [12] V. Feng and S. Y. Chang, "Determination of Wireless Networks Parameters through Parallel Hierarchical Support Vector Machines," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 3, pp. 505-512, March 2012. doi: 10.1109/TPDS.2011.156
- [13] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt and B. Scholkopf, "Support vector machines," in *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18-28, July-Aug. 1998. doi: 10.1109/5254.708428
- [14] K. W. Choi and E. Hossain, "Estimation of Primary User Parameters in Cognitive Radio Systems via Hidden Markov Model," in *IEEE Transactions on Signal Processing*, vol. 61, no. 3, pp. 782-795, Feb.1, 2013. doi: 10.1109/TSP.2012.2229998
- [15] L. Rabiner and B. Juang, "An introduction to hidden Markov models," in *IEEE ASSP Magazine*, vol. 3, no. 1, pp. 4-16, Jan 1986. doi: 10.1109/MASSP.1986.1165342
- [16] B. K. Donohoo, C. Ohlsen, S. Pasricha, Y. Xiang and C. Anderson, "Context-Aware Energy Enhancements for

- Smart Mobile Devices," in IEEE Transactions on Mobile Computing, vol. 13, no. 8, pp. 1720-1732, Aug. 2014. doi: 10.1109/TMC.2013.94
- [17] P. P. Markopoulos, "Linear Discriminant Analysis with few training data," 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, 2017, pp. 4626-4630. doi: 10.1109/ICASSP.2017.7953033
- [18] T. Haifley, "Linear logistic regression: an introduction," IEEE International Integrated Reliability Workshop Final Report, 2002., Lake Tahoe, CA, USA, 2002, pp. 184-187. doi: 10.1109/IRWS.2002.1194264
- [19] S. Taneja, C. Gupta, K. Goyal and D. Gureja, "An Enhanced K-Nearest Neighbor Algorithm Using Information Gain and Clustering," 2014 Fourth International Conference on Advanced Computing & Communication Technologies, Rohtak, 2014, pp. 325-329. doi: 10.1109/ACCT.2014.22
- [20] M. Mishra and M. Srivastava, "A view of Artificial Neural Network," 2014 International Conference on Advances in Engineering & Technology Research (ICAETR - 2014), Unnao, 2014, pp. 1-3. doi: 10.1109/ICAETR.2014.7012785
- [21] G. Alnwaimi, S. Vahid and K. Moessner, "Dynamic Heterogeneous Learning Games for Opportunistic Access in LTE-Based Macro/Femtocell Deployments," in IEEE Transactions on Wireless Communications, vol. 14, no. 4, pp. 2294-2308, April 2015. doi: 10.1109/TWC.2014.2384510
- [22] L. Gispán, A. Leshem, and Y. Be'ery, "Decentralized estimation of regression coefficients in sensor networks," Digital Signal Processing, vol. 68, pp. 16-23, Sep. 2017. doi: 10.1016/j.dsp.2017.05.005
- [23] F. Cadger, K. Curran, J. Santos, and S. Moffet, "Opportunistic Neighbour Prediction Using an Artificial Neural Network," International Journal of Advanced Pervasive and Ubiquitous Computing, vol. 7, no. 2, pp. 38-50, Apr. 2015.
- [24] F. Tang et al., "On Removing Routing Protocol from Future Wireless Networks: A Real-time Deep Learning Approach for Intelligent Traffic Control," in IEEE Wireless Communications, vol. 25, no. 1, pp. 154-160, February 2018. doi: 10.1109/MWC.2017.1700244
- [25] Kotsiantis, Sotiris. (2007). Supervised Machine Learning: A Review of Classification Techniques. Informatica (Ljubljana). 31.
- [26] Vinod Nair , Geoffrey E. Hinton, Rectified linear units improve restricted boltzmann machines, Proceedings of the 27th International Conference on International Conference on Machine Learning, p.807-814, June 21-24, 2010, Haifa, Israel
- [27] Xavier Glorot, Antoine Bordes, YoshuaBengio, "Deep Sparse Rectifier Neural Networks," Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, PMLR 15:315-323, 2011.
- [28] Zhou Wang and A. C. Bovik, "Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures," IEEE Signal Processing Magazine, vol. 26, no. 1, pp. 98-117, Jan. 2009
- [29] Sebastian Ruder, "An overview of gradient descent optimization algorithms," 2016, arXiv:1609.04747 {refcs.LG}
- [30] Khana, Atta ur Rehman, et al. 'A Performance Comparison of Network Simulators for Wireless Networks'. ArXiv:1307.4129 [Cs], July 2013. arXiv.org, <http://arxiv.org/abs/1307.4129>.
- [31] Kabir, Mohammed & Islam, Syful & Hossain, Md & Hossain, Sazzad. (2014). Detail Comparison of Network Simulators. 10.13140/RG.2.1.3040.9128.
- [32] Foram S. Panchal, Mahesh Panchal, "Review on Methods of Selecting Number of Hidden Nodes in Artificial Neural Network", International Journal of Computer Science and Mobile Computing, Vol.3 Issue.11, November- 2014, pg. 455-464

### Authors' Profiles



**Baisakhi Chatterjee**, has obtained her Bachelor's degree in Computer Science from Institution of Engineering and Management, Kolkata in 2018. She has worked under Prof. Saha during that time. Her main research interests include wireless network.

She has gained experience working as a summer intern in Indian Statistical Institute under the guidance of Prof. Nivedita Das. In this capacity, her research focused detection of communities in large scale networks. Her previous works include research on glycolysation of amino acids and routing protocols of ad hoc networks. She currently works in NRIFintech, India.



**Dr. Himadri Nath Saha** is the Head of the Department of Computer Science & Engineering at Institute of Engineering & Management, Kolkata. He has completed his Bachelor of Engineering from Jadavpur University, his Master of Engineering from Indian Institute of Engineering, Science and Technology(IEST, Shibpur and has obtained his Ph.D. in Engineering from Jadavpur University.

His research interest includes Machine Learning, IOT, Wireless Communication, Mobile Ad-hoc Networks, Network Security, Cryptography and Algorithms. He has several publications in international journals and has collaborated with multiple researchers on various projects. He has also written a textbook on "Database Management System".

Dr. Saha is the "Fellow" of many premiere organisations like Institution of Engineers India(IEI) and The Institution of Electronics and Telecommunications Engineers(IETE), India and senior member of IEEE(USA). He is the Branch Counselor of IEEE Student Branch & Faculty Head of ACM Student Chapter.

**How to cite this paper:** Baisakhi Chatterjee, Himadri Nath Saha, "Parameter Training in MANET using Artificial Neural Network", International Journal of Computer Network and Information Security(IJCNIS), Vol.11, No.9, pp.1-8, 2019.DOI: 10.5815/ijcnis.2019.09.01