MATHEMATICAL APPROACH TO DESIGN AN ONLINE MULTILAYER FEED FORWARD NEURAL NETWORK AND ITS MODEL ON LABORATORY WORK

G.GAJENDRAN*
DR.T.VENUGOPAL**

*Research Scholar, SCSVMV University, Kancheepuram, Tamil Nadu, India
** Professor of Mathematics and Director of Research and Publications, SCSVMV University, Kancheepuram, Tamil Nadu, India

ABSTRACT

Research on discrete problems depends on mathematical models. Multilayer feed forward neural network is one of the computational mathematical models that can be performed as a linear or non-linear equation within the problems. The proposed method is a mathematical approach that develops Multilayer feed forward neural network with back-propagation under the learning rate influenced by sum of absolute error to overcome from complexity of a conventional model. The proposed model applied to predict refractive index for binary mixtures of ionic liquids with solvent water or ethanol and obtained results are compared with existing results.

KEYWORDS: Multilayer Feed Forward Neural Network, Back-Propagation, Ionic Liquid, Refractive Index

INTRODUCTION

The conventional back-propagation (BPN) algorithm is an innovative method [1, 4, 7] for training the Multilayer feed forward neural network (MLFFN). Researchers [9, 10] still investigate the training skill required by BPN due to their longer time of convergence. These issues are addressed through an innovative methodological alternative that rejects the options of local minima with the choice of learning rate and learning momentum [1, 7]. [2] proposes the method to initialize the components of weight vectors that involved in feed forward neural network (FNN). This method increases the rate of convergence of FNN by determining the output value of each neuron in the active region. An adaptive learning rate for BPN algorithm defined in [6] and [8]. The learning rate is adjusted with condition based on error measured on validation set in [8] whereas [6] defines the learning rate as the derivative of the sigmoidal value of the error of the layer.

An MLFFN with BPN needs a set of mathematical techniques that exploring a successful learning model from the given problem. Section 2 discusses the proposed techniques. Application of the proposed model on theoretical prediction of refractive examined in section 3.

1 Mathematical approach

In order to improve the performance of conventional MLFFN with BPN method, the objectives of this section are pre – process for designing the problems, controlling the internal representation of MLFFN and post – process that provides information for BPN whether MLFFN learning or stop the learning by analytical method.

2.1 Designing the problem

Designing the problem aims the discrete patterns into normalization and categorization in order to form a set within the range of learning.

2.1.1 Normalizing

Scaling the environment and observation to fall within a specified range [1] is known as normalization. In order to normalize the components of input patterns of the environment and output patterns of their corresponding observation, the real world problem can be arranged as: for a finite positive integer $\mathscr{P}$, there is a couple of sets $X \subset \mathbb{R}^m$ of input patterns and $Y \subset \mathbb{R}^n$ of their corresponding output patterns whose elements are $\boldsymbol{x}^p$ and $\boldsymbol{d}^p$ respectively. Such that the set of discrete examples can be defined as $\boldsymbol{p} = \{(\boldsymbol{x}^p, \boldsymbol{d}^p) \,/\, p = 1, 2, 3, \ldots, \mathscr{P}\}$ and X_i be defined as the set of i^{th} component in $\boldsymbol{x}^p$ or in $\boldsymbol{d}^p$ for every p then the normalizer of $\boldsymbol{p}$ is defined by a function f : $X_i \rightarrow (0, 1)$ such that $f(x) = a + b\frac{x - Min\{X_i\}}{Max\{X_i\} - Min\{X_i\}}$ for $x \in X_i$

2.1.2 Categorization

Real world examples are mostly in the form of discrete values, it is must to categorize the normalized values to perform MLFFN with BPN [1]. The set of normalized examples categorized into two subsets as follows: It is possible to choose an element c $\in \mathbb{R}$, the set of positive real number, the training set X = $\{(\mathbf{x}^p, \mathbf{d}^p) \,/\, p = 1, 2, 3, \ldots, P\}$ can be defined as a subset of the set of normalized examples such that for u $\neq$ v, any pair of elements $(\mathbf{x}^u, \mathbf{d}^u)$ $(\mathbf{x}^v, \mathbf{d}^v)$ in X satisfy the condition $\frac{1}{m+n}[\|x^u - x^v\| + \|d^u - d^v\|] \geq c$ and the testing set $\{(\mathbf{x}^p, \mathbf{d}^p) \,/\, p = P+1, P+2, \ldots, \mathscr{P}\}$ can be defined as a subset of the set of normalized examples such that for any element ($\mathbf{x}^u$, $\mathbf{y}^u$) in that set satisfies the condition $\frac{1}{m+n}\left[\|x^u - x^v\| + \|d^u - d^v\|\right] < \varepsilon$ for atleast one element ($\mathbf{x}^v$, $\mathbf{d}^v$) in X

Categorization of two classes explores that the elements of testing set are a neighborhood of atleast one element of training set so that the trained MLFFN avoids over fitting or under fitting the values in the testing set.

2.2 Controlling the internal representation of MLFF

Internal representation (IR) of an MLFFN is an activity of formally presenting the accumulated information by BPN from the range of given problem to MLFFN. Controlling the IR consists of mapping representation and improving the performance of BPN

2.2.1 Mapping representation

An MLFFN is a group of artificial neurons where each artificial neuron receives input vector from previous neurons and transforms them as an output [3]. Mathematical model of any neuron (fig.1) represents its input and output as a linear combination ip^p of an input vector $(x_1^p, x_2^p, \dots, x_m^p) \in \mathbb{R}^m$ together with weight vector $(w_1, w_2, \dots, w_{m+1}) \in \mathbb{R}^{m+1}$ and activated value [1, 4, 7] of ip^p i.e., op^p, respectively, where

$ip^p = \sum_{i=1}^{m+1} w_i x_i^p$ and $op^p = \frac{1}{1+e^{-ip^p}}$.

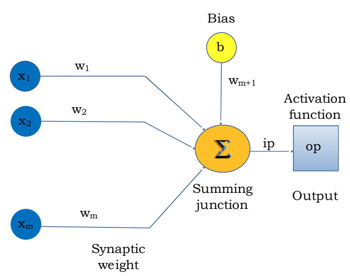

Fig.1: Artificial neuron

The slope of op^p is atmost parallel to ip^p axis and least at every point of the intervals $(-\infty, -3.4)$ and $(3.4, \infty)$, initial learning of MLFFN gets slow. It is important to initializing some neurons in which $ip^p \in [-3.4, 3.4]$. For g = 6.8 and $\forall\, u, v \in \{1, 2, \dots, P\}$, evidently ${}^{P}C_2$ distinct terms $(ip^u - ip^v)^2$ satisfy $(ip^u - ip^v)^2 \leq g^2$

By Cauchy- Schwarz inequality,

$$\left(\sum_{i=1}^{m+1} w_i (x_i^u - x_i^v)\right)^2 \leq \sum_{i=1}^{m+1} (w_i)^2 \sum_{i=1}^{m+1} (x_i^u - x_i^v)^2$$

For $\omega^2 = max\{(w_i)^2 /\ 1 \leq i \leq m+1\}$,

$$\mathcal{A} = max\{\sum_{i=1}^{m+1} (x_i^u - x_i^v)^2 / \forall\, u, v \in \{1, 2, \dots, P\}\}$$

and any real number r > 1, choice of weight vector satisfies the conditions

$\left(\frac{m+1}{r}\right)\omega^2 \leq (m+1)\omega^2$ and $\left(\frac{m+1}{r}\right)\omega^2\mathcal{A} \leq g^2$

explores that $-g\sqrt{\frac{r}{(m+1)\mathcal{A}}} \leq \omega \leq g\sqrt{\frac{r}{(m+1)\mathcal{A}}}$

This procedure is common for all neurons in MLFFN. It explores that an analytical presentation to initialize all weight vectors corresponding to their input vectors determines the MLFFN architecture in the form of mapping or function and is known as mapping representation. The main object of mapping representation is that MLFFN can produce the output value within the interval of target outputs.

2.2.2 Mathematical model of MLFFN

Let X be a compact subset of $[0,\ 1]^m$ whose elements are normalized input patterns and Y be a compact subset of $[0,\ 1]^n$. The mathematical model of MLFFN with one hidden layer and proposed weigh components is a function N: $X \to Y$ such that the output value produced by MLFFN is defined as $\mathrm{N}(\mathbf{x}^p) = \mathbf{y}^p$ for p = 1, 2, 3, …, P

2.2.3 Error

Let D be a compact subset of $[0,1]^n$ whose elements $\boldsymbol{d}^p = \left(d_1^p, d_2^p, \ldots, d_n^p\right)$ are actual output pattern corresponding to an input pattern $\mathbf{x}^p$**.** Since an initial weight vector $\boldsymbol{W}$ in $\mathrm{N}(\mathbf{x}^p)$ is set at random, MLFFN's output value $\mathbf{y}^p$ may not equal to $\mathbf{d}^p$. Therefore an error function of continuous weight vector $\boldsymbol{W}$ is defined as a sum of mean square value [3] of residues $d_k^p - y_k^p$ for all k and p.

i.e., $E(\boldsymbol{W}) = \frac{1}{2}\sum_{p=1}^{P}\sum_{k=1}^{n}\left(d_k^p - y_k^p\right)^2$

2.2.4 Training or learning

Learning [1, 7] is a process by which all weight components of $\boldsymbol{W}$ in MLFFN are updated and MLFFN is embedded to minimizing the error function $E(\boldsymbol{W})$.

2.2.5 Back propagation

In 1986, Rumelhart [4] showed that $E(\boldsymbol{W})$ can be minimized by updating weight components w of $\boldsymbol{W}$**.** General model is to update weight components in the opposite direction of $\frac{\partial E^p(\boldsymbol{W})}{\partial w}$. This learning law is called Backpropagation algorithm which is a supervised training algorithm for MLFFN with online training. In this training, all weight components involved in MLFFN are updated by presenting input pattern and its corresponding output pattern one after one. An error function of $\boldsymbol{W}$ is defined as

$$E^p(W) = \frac{1}{2}\sum_{k=1}^{m}\left(d_k^p - y_k^p\right)^2$$

For all proposed initialized weight components w(0), the weight updated rule for this model is formally given by

$$w(t+1) = w(t) - \eta * \frac{\partial E^p(W)}{\partial w}$$

where t is epoch number and η is learning rate.

2.2.6 Proposed learning rate

The important research work of MLFFN with BPN is to choose the learning rate [6, 8] for BPN to update all weight components to ensure a successful MLFFN. But it is difficult to choose the learning rate, because the primary reason for this is the non-monotonic nature of the process. In order to achieve all weight components to minimize the error function by online learning method, the proposed model for learning rate described as follows:

Let N = {N_1, N_2, N_3, …}be the set of MLFFNs corresponding to epochs, where N_{t+1} is updated network of N_t by BPN. Let ae_{t1} and ae_{t2} be a sum of absolute errors of all patterns produced by N_i before learning and after learning respectively. But due to non-monotonic nature, the MLFFN may diverge. To avoid divergence, suppose $ae_{t1} < ae_{t2}$ then ae_{t2} should not exceed 3. ae_{t1}, otherwise MLFFN stops the learning. Based on this, with initial learning rate $\eta(0)$ close to zero, the proposed learning rate can be represented as

$$\eta(t+1) = \begin{cases} \eta(t) + e\,\dfrac{ae_{t2}}{ae_{t1}} + f, & if\ ae_{t2} \in (0,\ ae_{t1}) \\ \eta(t) - e\,\dfrac{ae_{t2}}{ae_{t1}} - f, & if\ ae_{t2} \in (ae_{t1}, 3.ae_{t1}] \end{cases}$$

otherwise $\eta = 0\ if\ ae_{t2} \in (3.ae_{t1}, \infty)$ or ae_{t1} or ae_{t2} satisfies the stopping rule, where e and f are real numbers close to zero. The proposed learning rate uses the information by comparing learning capability of MLFFN. During learning, it acts as an autonomous and adaptive learning rate to control the non monotonic of MLFFN at most.

2.2.7 **Renormalization**

The actual input patterns and output patterns are normalized for training and testing of MLFFN with BPN, it should be renormalized to correlate with environment and their corresponding observation. The renormalized method for modeling or approximation is given as: Theoretical output defined by substituting the obtained output value by trained MLFFN in the inverting function of normalizer.

2.3 Stopping rule

The sequence of sum of absolute errors produced by $\{N_i\}$ is non monotone. The system can be stopped when it performances best assessment with environments and their corresponding observation and with existing results on those problems.

3 MLFFN model on laboratory work

MLFFN with BPN can learn by examples. Its mapping representation can be estimated if it is able to provide information on the evolution of a real system sufficiently near to those obtained by experiments on the real system. In this chapter, proposed model applied in the examples of chemical problems.

3.1 Chemical industry

The role of chemistry in the advancement of human civilization is very significant, but this achievement has come at the cost of human health and the global environment. It is necessary to develop a mathematical model for precise data prediction in the chemical industry.

3.1.1 Ionic liquid

Ionic liquids (ILs) are room temperature molten salts and exploited as green solvents. Their use as an environmentally friendly alternative to conventional solvents has gained much attention recently in both academic and industrial areas.

3.1.2 Refractive index

Refractive index is an important physical quantity for analysis, because many substances can be easily identified by knowing its value. It is also used for theoretical purposes due to the fact that it gives information about the electronic configuration of the different ions and molecules forming the liquid. Experimental measurements of the refractive index for seven binary mixtures of four ionic liquids ethyl (EMIM), butyl (BMIM), hexyl (HMIM), and octyl (OMIM) of the family 1-alkyl-3-methyl Imidazolium tetrafluoroborate (CnMIM-BF4) with solvents, water or ethanol are given in [10] with respect molecular mass, refractive index and density of pure components and mole fraction of ionic liquid mixture at T = 298.15K

3.2 Description of proposed model

Proposed model uses 60 experimental data [10] with 7 input components (Table 1) which are molecular mass, density, and refractive index of pure component ethyl, butyl , hexyl, or octyl and solvent water or ethanol and corresponding one output refractive index of binary mixture. In order to normalize the input components and output, normalizer uses the values a = 0.2 and b = 0.8

3.2.1 Training set and Testing set

The normalized data categorized into two sets as follows: Any pair of elements in training set X satisfy the condition $\frac{1}{8}[\|x^u - x^v\| + \|y^u - y^v\|] > 0.0125$ and element ($\mathbf{x}^u$, $\mathbf{y}^u$) in testing set satisfies the condition $\frac{1}{8}[\|x^u - x^v\| + \|y^u - y^v\|] \leq 0.0125$ for at least one element ($\mathbf{x}^v$, $\mathbf{y}^v$) in X. The proposed model uses 66.7% (40 training data) of examples in the training set, and rest 33.3% (20 testing data) of examples in the testing set. There are 8 neurons in input layer, one neuron in output layer and 13 hidden units in the architecture of MLFFN.

3.2.2 Weight initialization

The weight vectors between input layer and each neuron in hidden layer [1, 7] are initialized for the training set with $\mathcal{A} = 4.2077$. All weight vectors with their value r are given in table 3. The weight vector between hidden layer and output layer [1, 7] is initialized for training set (table 4) with $\mathcal{A} = 0.9097$ and r = 3.5

3.2.3 Training parameter

The parameter used for this examples are as follows:

- Partial derivative $\frac{\partial E^P(W)}{\partial w}$ for bias between hidden layer and output layer is comprised by the factor 0.005
- Initial learning rate is $\eta = 0.009$
- e = 0.00030045 and f = 0.00098108

3.2.4 Result

With proposed training, MLFFN learned with the training set. The number of epochs is 25193 and at the end of training $E(W)$ = 0.003079. In each epoch, all 60 renormalized data are compared with experimental data and existing data [5]. MLFFN stops learning when it produces the best result compare with result in [5]. Comparison result of each data with their absolute error is presented in table 2 and fig. 2 and 3. Out of 60 data, proposed model performs an average of 0.031% error, Gladstone – Dale model performs an average of 0.14% error and Newton model performs an average of 0.14% [5].

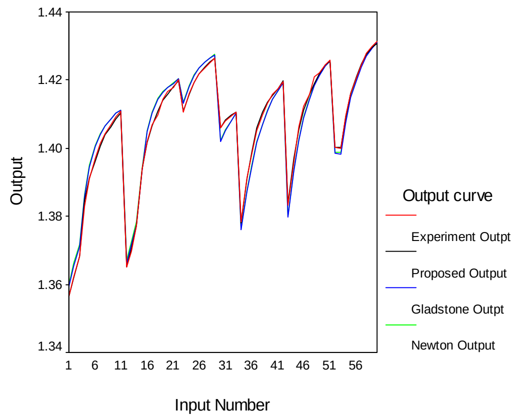

Fig.2 Performance of theoretical result

4 Research methodology

The algorithm of proposed model is executed through the c – Program. The results are compared with statistical tools and presented in table 5. Correlation coefficient shows a reciprocal relation between experimental result and theoretical result, whereas proximity shows the theoretical result close to the experimental result.

5 Conclusion

An MLFFN is mathematical function N from the set of discrete input vectors to the corresponding output vectors with their relational information stored in the weights by minimizing an objective function under the BPN training with a considerable number of epochs. But the algorithm of BPN training heavily depends on the choice of learning rate. The proposed learning rate is an autonomous and adaptive model which uses the learning capability of LFFN. The performance of proposed online model applied on theoretical prediction of refractive index.

References:

1. Haykin S. Neural networks: A comprehensive foundation, IEEE Computer Society Press, 1994.
2. Jim Y.F.Yam, Tommy W.S.Chow, "A weight initialization method for improving training speed in feedforward neural network", pp. 219-232/NC/ELSEVIR, 2000
3. McCulloch.W.S, Pitts.W, " A logical calculus of the ideas immanent in nervous activity", Bulletin of Mathematical Biophysics, vol.5, pp 115-133, 1943.
4. Rilo.E, Domingez-Perez.M, Vila.J, " Easy prediction of the refractive index for binary mixtures of ionic liquids with water or ethanol", Vol.47, pp.219-222, J.Chem Thermodynamics , 2012.
5. Rumelhart.D.E, G.E.Hinton, and R.J.Williams, "Learning Internal Representations by Error Propagation", in Parallel Distributed Processing. Exploration in microstructure of Cognition. Vol. 1: Foundations., MIT Press, 1986.
6. Saeid Iranmanesh(1), M. Amin Mahdavi, "A Differential Adaptive Learning Rate Method for Back-Propagation Neural Networks", vol. 50, World Academy of Science, Engineering and Technology, 2009

7. SanjeevKkulkarni, Gilbert Harman, “An Elementary Introduction to Statistical Learning Theory”, A JOHN WILEY & SONS, INC., PUBLICATION 2011
8. Stefan Duffner, Christophe Garcia, “An Online Backpropagation Algorithm with Validation Error-Based Adaptive Learning Rate” pp. 249-258/LNCS 4668/Springer – Verlag Berlin Heidelberg, 2007
9. Stelios Timotheou, “A novel weight initialization method for the random neural network”, Vol.73, (2009) pp.160–168, Neuro computing , 2009.
10. Zainddin.Z, Mahat.N, Abu Hassan.Y, “Improving the convergence of the backpropagation algorithm using local adaptive techniques”, vol.1 World Academy of Science, Engineering and Technology, 2005

Appendix I

Table 1: 60 data with their input components and corresponding output

Sl. No.	Ionic liquid with solvent	7 input components							Actual output
		Ionic liquid			Solvent			Ionic liquid	
		Molecular mass Kg/Mol	Density kg / m^3	Refractive index of Pure compounds (Experiment)	Molecular mass Kg/Mol	Density kg / m^3	Refractive index of Pure compounds (Experiment)	Mole fraction	Refractive index of mixture
1.	Ethyl with water	0.19798	1279.9	1.4123	0.01801	997.0	1.3325	0.0556	1.3565
2.		0.19798	1279.9	1.4123	0.01801	997.0	1.3325	0.0762	1.3623
3.		0.19798	1279.9	1.4123	0.01801	997.0	1.3325	0.0985	1.3682
4.		0.19798	1279.9	1.4123	0.01801	997.0	1.3325	0.1902	1.3830
5.		0.19798	1279.9	1.4123	0.01801	997.0	1.3325	0.2894	1.3909
6.		0.19798	1279.9	1.4123	0.01801	997.0	1.3325	0.4011	1.3971
7.		0.19798	1279.9	1.4123	0.01801	997.0	1.3325	0.5020	1.4013
8.		0.19798	1279.9	1.4123	0.01801	997.0	1.3325	0.6018	1.4043
9.		0.19798	1279.9	1.4123	0.01801	997.0	1.3325	0.6986	1.4068
10.		0.19798	1279.9	1.4123	0.01801	997.0	1.3325	0.8086	1.4091
11.		0.19798	1279.9	1.4123	0.01801	997.0	1.3325	0.9015	1.4108
12.	Butyl with water	0.22603	1201.2	1.4212	0.01801	997.0	1.3325	0.0559	1.3650
13.		0.22603	1201.2	1.4212	0.01801	997.0	1.3325	0.0708	1.3698
14.		0.22603	1201.2	1.4212	0.01801	997.0	1.3325	0.0899	1.3771
15.		0.22603	1201.2	1.4212	0.01801	997.0	1.3325	0.1745	1.3939
16.		0.22603	1201.2	1.4212	0.01801	997.0	1.3325	0.2963	1.4021
17.		0.22603	1201.2	1.4212	0.01801	997.0	1.3325	0.4067	1.4070
18.		0.22603	1201.2	1.4212	0.01801	997.0	1.3325	0.5304	1.4097
19.		0.22603	1201.2	1.4212	0.01801	997.0	1.3325	0.6326	1.4144
20.		0.22603	1201.2	1.4212	0.01801	997.0	1.3325	0.7065	1.4166
21.		0.22603	1201.2	1.4212	0.01801	997.0	1.3325	0.8003	1.4179
22.		0.22603	1201.2	1.4212	0.01801	997.0	1.3325	0.9005	1.4201
23.	Hexyl with water	0.25409	1145.4	1.4283	0.01801	997.0	1.3325	0.3032	1.4107
24.		0.25409	1145.4	1.4283	0.01801	997.0	1.3325	0.4059	1.4161
25.		0.25409	1145.4	1.4283	0.01801	997.0	1.3325	0.5058	1.4192
26.		0.25409	1145.4	1.4283	0.01801	997.0	1.3325	0.6061	1.4218
27.		0.25409	1145.4	1.4283	0.01801	997.0	1.3325	0.7057	1.4237
28.		0.25409	1145.4	1.4283	0.01801	997.0	1.3325	0.8030	1.4250
29.		0.25409	1145.4	1.4283	0.01801	997.0	1.3325	0.9002	1.4265
30.	Ethyl with ethanol	0.19798	1279.9	1.4123	0.04607	785.8	1.3600	0.6087	1.4060
31.		0.19798	1279.9	1.4123	0.04607	785.8	1.3600	0.7056	1.4079
32.		0.19798	1279.9	1.4123	0.04607	785.8	1.3600	0.8072	1.4094
33.		0.19798	1279.9	1.4123	0.04607	785.8	1.3600	0.9054	1.4107
34.	Butyl with ethanol	0.22603	1201.2	1.4212	0.04607	785.8	1.3600	0.0986	1.3784
35.		0.22603	1201.2	1.4212	0.04607	785.8	1.3600	0.2002	1.3914
36.		0.22603	1201.2	1.4212	0.04607	785.8	1.3600	0.2881	1.3982
37.		0.22603	1201.2	1.4212	0.04607	785.8	1.3600	0.3999	1.4050
38.		0.22603	1201.2	1.4212	0.04607	785.8	1.3600	0.5024	1.4096

39.		0.22603	1201.2	1.4212	0.04607	785.8	1.3600	0.6082	1.4131
40.		0.22603	1201.2	1.4212	0.04607	785.8	1.3600	0.7107	1.4158
41.		0.22603	1201.2	1.4212	0.04607	785.8	1.3600	0.8040	1.4175
42.		0.22603	1201.2	1.4212	0.04607	785.8	1.3600	0.9059	1.4196
43.	Hexyl with ethanol	0.25409	1145.4	1.4283	0.04607	785.8	1.3600	0.0980	1.3833
44.		0.25409	1145.4	1.4283	0.04607	785.8	1.3600	0.1998	1.3959
45.		0.25409	1145.4	1.4283	0.04607	785.8	1.3600	0.3000	1.4062
46.		0.25409	1145.4	1.4283	0.04607	785.8	1.3600	0.3991	1.4123
47.		0.25409	1145.4	1.4283	0.04607	785.8	1.3600	0.5079	1.4157
48.		0.25409	1145.4	1.4283	0.04607	785.8	1.3600	0.6123	1.4210
49.		0.25409	1145.4	1.4283	0.04607	785.8	1.3600	0.7020	1.4222
50.		0.25409	1145.4	1.4283	0.04607	785.8	1.3600	0.8030	1.4243
51.		0.25409	1145.4	1.4283	0.04607	785.8	1.3600	0.8596	1.4258
52.	Octyl with ethanol	0.28214	1104.2	1.4330	0.04607	785.8	1.3600	0.2034	1.4002
53.		0.28214	1104.2	1.4330	0.04607	785.8	1.3600	0.2021	1.4000
54.		0.28214	1104.2	1.4330	0.04607	785.8	1.3600	0.3000	1.4095
55.		0.28214	1104.2	1.4330	0.04607	785.8	1.3600	0.4096	1.4161
56.		0.28214	1104.2	1.4330	0.04607	785.8	1.3600	0.5125	1.4208
57.		0.28214	1104.2	1.4330	0.04607	785.8	1.3600	0.6156	1.4245
58.		0.28214	1104.2	1.4330	0.04607	785.8	1.3600	0.7207	1.4278
59.		0.28214	1104.2	1.4330	0.04607	785.8	1.3600	0.8150	1.4298
60.		0.28214	1104.2	1.4330	0.04607	785.8	1.3600	0.9014	1.4314

Table 2: 60 experimental output and proposed output and existing output with their absolute error

Sl. No.	Experimental output	Proposed output	Absolute error	Gladstone – Dale model output	Absolute error	Newton model output	Absolute error
1	1.3565	1.356336	0.000164	1.3592	0.0027	1.3598	0.0033
2	1.3623	1.361924	0.000376	1.3655	0.0032	1.3661	0.0038
3	1.3682	1.368305	0.000105	1.3711	0.0029	1.3717	0.0035
4	1.383	1.384776	0.001776	1.3858	0.0028	1.3863	0.0033
5	1.3909	1.391339	0.000439	1.3945	0.0036	1.3949	0.004
6	1.3971	1.396292	0.000808	1.4005	0.0034	1.4007	0.0036
7	1.4013	1.400453	0.000847	1.404	0.0027	1.4042	0.0029
8	1.4043	1.40384	0.00046	1.4066	0.0023	1.4067	0.0024
9	1.4068	1.406238	0.000562	1.4085	0.0017	1.4086	0.0018
10	1.4091	1.408466	0.000634	1.4102	0.0011	1.4102	0.0011
11	1.4108	1.410387	0.000413	1.4113	0.0005	1.4113	0.0005
12	1.365	1.36498	0.00002	1.3663	0.0013	1.367	0.002
13	1.3698	1.370522	0.000722	1.3717	0.0019	1.3725	0.0027
14	1.3771	1.377179	0.000079	1.3775	0.0004	1.3782	0.0011
15	1.3939	1.393783	0.000117	1.3935	0.0004	1.3941	0.0002
16	1.4021	1.40169	0.00041	1.4047	0.0026	1.4052	0.0031
17	1.407	1.406556	0.000444	1.4103	0.0033	1.4106	0.0036
18	1.4097	1.41093	0.00123	1.4143	0.0046	1.4145	0.0048
19	1.4144	1.413938	0.000462	1.4165	0.0021	1.4167	0.0023
20	1.4166	1.415884	0.000716	1.4178	0.0012	1.4179	0.0013
21	1.4179	1.418014	0.000114	1.4191	0.0012	1.4192	0.0013
22	1.4201	1.419788	0.000312	1.4203	0.0002	1.4203	0.0002
23	1.4107	1.41092	0.00022	1.4132	0.0025	1.4136	0.0029

24	1.4161	1.415768	0.000332	1.4181	0.002	1.4184	0.0023
25	1.4192	1.419313	0.000113	1.4212	0.002	1.4215	0.0023
26	1.4218	1.421924	0.000124	1.4235	0.0017	1.4236	0.0018
27	1.4237	1.423835	0.000135	1.4252	0.0015	1.4253	0.0016
28	1.425	1.4253	0.0003	1.4264	0.0014	1.4265	0.0015
29	1.4265	1.426565	0.000065	1.4274	0.0009	1.4275	0.001
30	1.406	1.406045	0.000045	1.4021	0.0039	1.4022	0.0038
31	1.4079	1.408392	0.000492	1.4052	0.0027	1.4053	0.0026
32	1.4094	1.40978	0.00038	1.408	0.0014	1.408	0.0014
33	1.4107	1.410681	0.000019	1.4103	0.0004	1.4103	0.0004
34	1.3784	1.378173	0.000227	1.3759	0.0025	1.3762	0.0022
35	1.3914	1.390938	0.000462	1.3873	0.0041	1.3876	0.0038
36	1.3982	1.398667	0.000467	1.3946	0.0036	1.3949	0.0033
37	1.405	1.40613	0.00113	1.4017	0.0033	1.402	0.003
38	1.4096	1.410572	0.000972	1.4068	0.0028	1.407	0.0026
39	1.4131	1.413499	0.000399	1.411	0.0021	1.4112	0.0019
40	1.4158	1.415639	0.000161	1.4143	0.0015	1.4144	0.0014
41	1.4175	1.417513	0.000013	1.4169	0.0006	1.417	0.0005
42	1.4196	1.419722	0.000122	1.4193	0.0003	1.4193	0.0003
43	1.3833	1.383721	0.000421	1.3799	0.0034	1.3802	0.0031
44	1.3959	1.397149	0.001249	1.3932	0.0027	1.3936	0.0023
45	1.4062	1.405496	0.000704	1.4022	0.004	1.4026	0.0036
46	1.4123	1.411253	0.001047	1.4089	0.0034	1.4092	0.0031
47	1.4157	1.415677	0.000023	1.4144	0.0013	1.4146	0.0011
48	1.421	1.419067	0.001933	1.4185	0.0025	1.4187	0.0023
49	1.4222	1.421711	0.000489	1.4214	0.0008	1.4216	0.0006
50	1.4243	1.4244	0.0001	1.4241	0.0002	1.4242	0.0001
51	1.4258	1.425719	0.000081	1.4255	0.0003	1.4255	0.0003
52	1.4002	1.400305	0.000105	1.3984	0.0018	1.3989	0.0013
53	1.4	1.400165	0.000165	1.3983	0.0017	1.3988	0.0012
54	1.4095	1.409102	0.000398	1.4075	0.002	1.408	0.0015
55	1.4161	1.41622	0.00012	1.4149	0.0012	1.4152	0.0009
56	1.4208	1.421084	0.000284	1.4199	0.0009	1.4202	0.0006
57	1.4245	1.424783	0.000283	1.4239	0.0006	1.4241	0.0004
58	1.4278	1.427605	0.000195	1.427	0.0008	1.4272	0.0006
59	1.4298	1.42951	0.00029	1.4294	0.0004	1.4295	0.0003
60	1.4314	1.430897	0.000503	1.4312	0.0002	1.4313	0.0001

Appendix II

Table 3: Initial weight components between input layer and hidden layer

Sl. No	r	Weight between input layer and 12 hidden neurons							
1	3.5	0.721673	0.703157	0.069888	0.905123	0.215487	-1.00255	1.504659	-0.56987
2	3.5	1.970683	1.990196	0.352709	-0.64378	-1.03255	2.012549	-0.54687	1.10257
3	3.3	0.25978	-1.56488	2.125487	0.021549	1.716999	1.18757	1.929468	1.067776
4	3.0	-1.29876	1.98764	1.000458	0.78954	-1.19736	-1.70827	-0.05028	0.293325
5	4.01	-2.34408	-0.41352	-1.77843	-0.79517	-0.25987	1.088975	-0.87594	-0.45214
6	4.0	-1.92774	-0.09176	1.898243	0.75373	1.875462	-0.02155	1.115487	0.55487
7	3.8	-1.36597	0.365847	1.987451	-2.00146	-0.61938	-0.0324	-0.30497	-0.73582
8	3.8	0.598745	-2.15488	1.44874	-0.11549	-1.03571	-1.96296	-0.30497	-0.73582
9	3.6	-1.22549	0.554875	-0.21549	0.425187	-1.80745	-1.25851	-0.24404	0.87402
10	3.61	-2.22378	-0.61499	0.887248	1.648471	0.986549	-1.20155	1.541021	-0.22254
11	3.21	-2.0966	0.155714	1.580071	0.226749	0.986549	-0.02549	1.54872	-0.82546
12	3.2	1.65483	-0.63533	1.098754	-1.1147	-1.26393	-0.48781	-0.6825	-0.5477

Table 4: Initial weight components between hidden layer and output layer

r	Weight between 12 hidden layers and output neurons									
3.5	-0.564875	-0.1125487	-1.265487	-1.021548	-0.2225487	-0.710530265	-2.1185117585166	-0.235987	1.225487	2.014587	1.587461

Appendix III

Table 5: Statistical comparison between experimental and theoretical results

Model	Correlation coefficient with experimental result	Proximity (Euclidean distance with experimental result)
Proposed Output	0.999	0.05
Gladstone Output	0.992	0.17
Newton Output	0.992	0.18

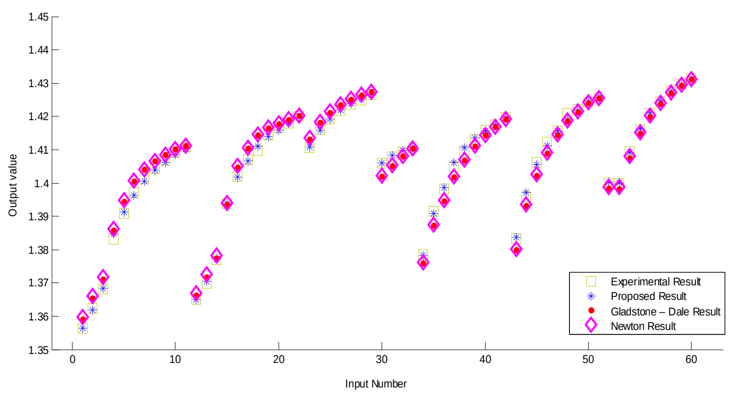

Fig. 3 shows that the proposed result plotted by the symbol * close to experimental result plotted by the symbol □

compare with Gladsone – Dale method potted by the symbol ● and Netwon method plotted by the symbol ◊

Appendix IV

Algorithm for Mathematical approach to design an on line multilayer feed forward neural network

Step.1 Categorize normalized data into training set and testing set

Step2. Initialize all weight components using training set

Step3. Present the experimental data and existing model

Setp4. Perform MLFFN with an element from training set and go to step 6 or if all training elements are presented, go to step 5

Step5. Compare renormalized MLFFN's output with experimental output and existing output. If MLFFN perform well than existing result, MLFFN stops the learning . Otherwise, go to step 4.

Step6. Train MLFFN with BPN along with proposed learning rate and go to Step4.

End.