World Applied Sciences Journal 5 (5): 546-552, 2008
ISSN 1818-4952

Multilayer Perceptron Neural Network (MLPs) For Analyzing the Properties of Jordan Oil Shale

[1]*Jamal M. Nazzal,* [2]*Ibrahim M. El-Emary and* [3]*Salam A. Najim*

[1,3]Al Ahliyya Amman University, P.O. Box 19328, Amman, Jordan
[2]King Abdulaziz University, P.O. Box 18388, Jeddah, King Saudi Arabia

Abstract: In this paper, we introduce the multilayer preceptron neural network and describe how it can be used for function approximation. The back propagation algorithm (including its variants) is the principle procedure for training multilayer perceptrons. Car must be taken when training perceptron network to ensure that they do not over fit the training data and then fail to generalize well in new situations. So the main purpose of this paper lies in studying the effect of changing both the No of hidden layers of MLPs and the No of processing elements that exist in the hidden layers on the analyzed properties of Jordan Oil Shale. After constructing such a MLP and changing the number of hidden layers, we found that with increasing the number of processing elements in the hidden layers. We reach to an optional output results w.r.t the experimental one or the analytical formulated one. This obtained output is completely matched with the main concepts of theoretical visions of ANN.

Key words: MLP . El-lojjin oil . sigmoid . perceptron . DM-1 . Dm-2

INTRODUCTION

A geochemical analysis of El-Lajjun oil shall in Jordan was carried out [1]. It was found that El-Lajjun oil shale consists of the following group: Organic matter, biogenic calcite and apatite, detrital clay minerals and quartz. The calorific values of 100 samples were determined. The effect of bore depth, calcium carbonate, organic carbon and sulfur content on the calorific values was studied [1]. The results were well correlated by some empirical formula given by [1]:

$$\text{Calorific value}=352.44\,(CaCo_3)^{-0.066}(S)^{0.257}(C_{org})^{1.143} \quad (1)$$

With correlation coefficient of 0.983 and with an average standard error of 2.63%. The current resurgence of interest in ANNs is largely due to the emergence of powerful new methods as well as to the availability of computational power suitable fir simulation. The field is particularly exciting today because ANN algorithms and architectures can be implemented in VLSI technology for real time applications [2]. Currently two principal streams can be identified in ANN research. Researches in the first stream are concerned with modeling the brain and thereby explain its cognitive behavior. On the other hand, the primary aim of researches in the second streams is to construct useful computers for real world problems of pattern recognition by drawing these principles. In this paper we want to give a complete description to neural networks and their application in analyzing and studying the properties of Jordan oil shale as a recent technique. At the same time, we would like to investigate the effect of changing the number of processing elements that exist in the hidden layer of the ANN on the forecasted parameters. To check the effectiveness of the described ANN, a comparative study was done between the estimated parameters by ANN and that one that is obtained through experiment or calculated using the formula given in Eg.1.

In this paper, we concentrate on the most common neural network architecture called the multilayer perceptron MLPs. For the purpose of this paper, we will look at neural network as function approximators [3]. As shown in Fig. 1, we have some unknown function that we wish to approximate. We want to adjust the parameters of the network so that it will produce the same response as the unknown function, if the same input is applied to both systems.

For our application, the unknown function may correspond to a system (oil shale) we are trying to control as a result of analysis, in which case the neural network will be the identified plant model. The unknowing function could also represent the inverse of a system (oil shale) we are trying to control and analyze, in which case the neural network can be used to implement the controller. This paper is organized from six sections. In section two, we describe the

Corresponding Author: Dr. Jamal M. Nazzal, Al Ahliyya Amman University, P.O. Box 19328, Amman, Jordan

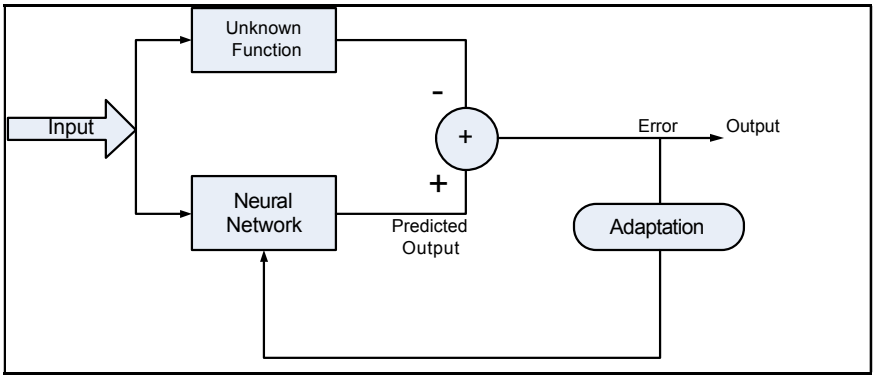

Fig. 1: Neural network as function approximator

neuron model. Section three was devoted to Multilayer Perception Architecture. Section five discusses a lot of obtained results. Finally section six presents the results, conclusion and future works done by others. Section four was dedicated to describe the operation of Multilayer Perception Neural Networks.

NEURON MODEL

The multilayer perception neural network is built up of simple components. In the beginning, we will describe a single input neuron which will then be extended to multiple inputs. Next, we will stack these neurons together to produce layers [4]. Finally, the layers are cascaded together to form the network.

Single-input neuron: A single-input neuron is shown in Fig. 2. The scalar input *p* is multiplied by the scalar weight *W* to form *Wp*, one of the terms that is sent to the summer. The other input, 1, is multiplied by a bias *b* and then passed to the summer. The summer output n often referred to as the net input, goes into a transfer function f which produces the scalar neuron output a (sometimes "activation function" is used rather than transfer function and offset rather than bias).

From Fig. 2, both w and b are both adjustable scalar parameters of the neuron. Typically the transfer function is chosen by the designer and then the parameters w and b will be adjusted by some learning rule so that the neuron input/output relationship meet some specific goal. The transfer function in Fig. 2 may be a linear or nonlinear function of n. A particular transfer function is chosen to satisfy some specification of the problem that the neuron is attempting to solve. One of the most commonly used functions is the log-sigmoid transfer function, which is shown in Fig. 3 [4].

This transfer function takes the input (which may have any value between plus and minus infinity) and

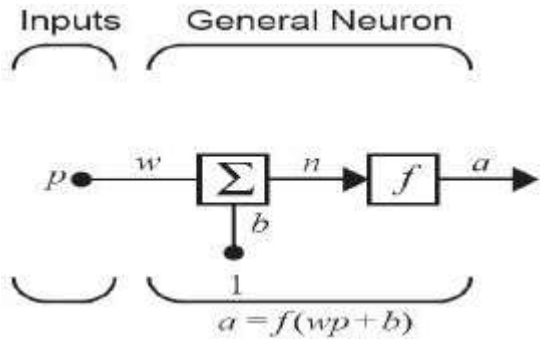

Fig. 2: Single input neuron [4]

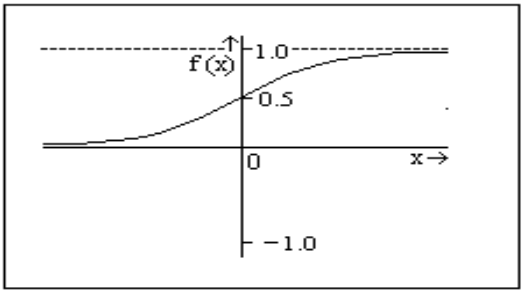

Fig. 3: Log-sigmiod transfer function

squashes the output into the range 0 to 1 according to the expression:

$$a = \frac{1}{1 + e^{-n}} \qquad (2)$$

The log-sigmoid transfer function is commonly used in multi-layer networks that are trained using the back propagation algorithm.

Multiple-input neuron: Typically, a neuron has more than one input. A neuron with R inputs is shown in Fig. 4. The individual inputs $p_1, p_2, \ldots\ldots, p_g$ are each

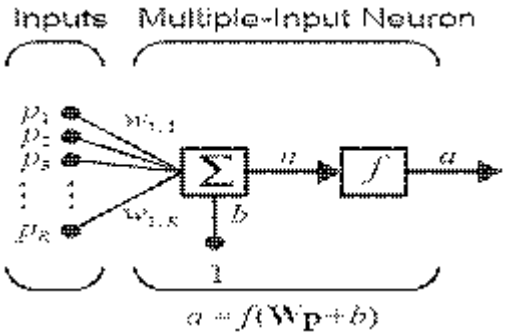

Fig. 4: Multiple-input neuron

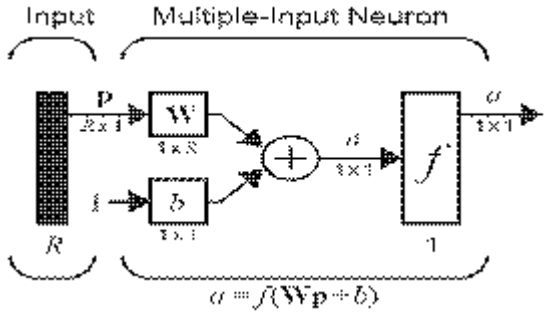

Fig. 5: Neuron with R inputs, abbreviated notations

weighted by corresponding elements $W_{1,1}$ $W_{1,2}$,....$W_{1,R}$ of the weight matrix W.

The neuron has a bias b, which is summed with the weight inputs to form the net input *n*:

$$n = W_{1,1}p_1 + W_{1,2}p_2 + ... + W_{1,R}p_R + b \quad (3)$$

This expression can be written in matrix form as:

$$n = Wp + b \quad (4)$$

Where the matrix W for the single neuron case has only one row. Now the neuron output can be written as:

$$a = f(Wp + b) \quad (5)$$

A particular convention in assigning the indices of the elements of the weight matrix has been adopted [4].

The first index indicates the particular neuron destination for the weight. The second index indicates the source of the signal fed to the neuron. Thus, the indices in $W_{1,2}$ say that this weight represents the connection to the first (and only) neuron from the second source [4]. A multiple-input neuron using abbreviated notation is shown in Fig. 5.

As shown in Fig. 5, the input vector p is represented by the solid vertical bar at left. The dimensions of p are displayed below the variable as *Rx1*, indicating that the input is a single vector of R elements. These inputs go to the weight matrix W, which has R columns but only one row in this single neuron case.

A constant 1 enters the neuron as an input and is multiplied by a scalar bias *b*. The net input to the transfer function *f* is *n*, which is the sum of the bias *b* and the product *Wp*. The neuron's output is a scalar in this case. If there exit more than one neuron, the network output would be a vector.

MULTILAYER PERCEPTRON NETWORK ARCHITECTURES

Commonly one neuron, even with many inputs, may not be sufficient. We might need five or ten, operating parallel, in what we will call a “layer". This concept of a layer is discussed below.

A layer of neurons: A single-layer network of S is shown in Fig. 6. Note that each of the R inputs is connected to each of the neurons and that the weight matrix now has *s* rows.

The layer includes the weight matrix, the summers, the bias vector *b*, the transfer function boxes and the output vector *a*. Each element of the input vector p is connected to each neuron through the weight matrix W. Each neuron has a bias b_i, a summer, a transfer function *f* and an output a_i. Taken together, the outputs form the output vector *a*. It is common for the number of inputs to a layer to be different from the number of neurons (i.e. R≠S). The input vector elements enter the network through the weight matrix W:

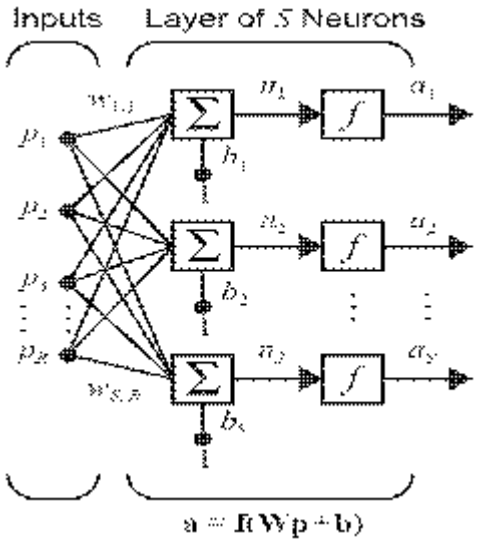

Fig. 6: Layer of S neurons

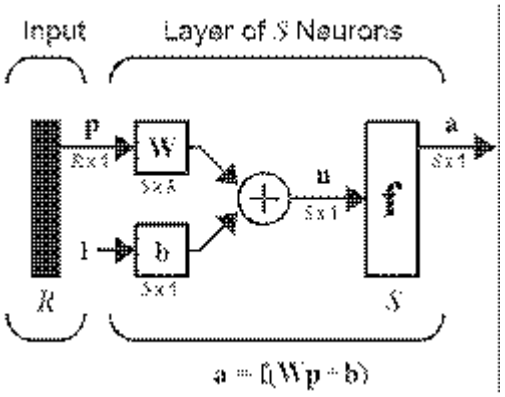

Fig. 7: Layer of S neurons, abbreviated notation

$$W = \begin{bmatrix} W_{1,1} & \cdots & W_{1,R} \\ \vdots & \ddots & \vdots \\ W_{s,1} & \cdots & W_{S,R} \end{bmatrix} \quad (6)$$

The row indices of the elements of matrix W indicate the destination neuron associated with that weight, while the column indices indicate the source of the input for that weight. Thus, the indices in $W_{3,2}$ say that this weight represents the connection to the third neuron from the second source. The S-neuron, R-input, one-layer network also can be drawn in abbreviated notation as shown in Fig. 7.

Here again, the symbols below the variables tell that for this layer, p is a vector of length R, W is an S*R matrix and *a* and *b* are vectors of length S.

As defined previously, the layer includes the weight matrix, the summation and multiplication operations, the bias vector b, the transfer function boxes and the output vector.

Multiple layers of neurons: Now consider a network with several layers which has been implemented in this paper for the purpose of analyzing Jordan oil shale properties. In this network each layer has its own weight matrix *W*, its own bias vector *b*, a net input vector *n* and an output vector *a*. Some additional notation should be introduced to distinguish between these layers. Superscripts are used to identify these layers. The number of the layer as a superscript is appended to the names for each of these variables. Thus, the weight matrix for the second layer is written as W^2. This notation is used in the three-layer network shown in Fig. 8.

As shown in Fig. 8, there are *R* inputs, S^1 neurons in the first layer, S^2 neurons in the second layer, etc. The output of layers one and two are the inputs for layers two and three. Thus layer 2 can be viewed as a one-layer network with $R=S^1$ inputs, $S= S^2$ neurons and an S^1*S^2 weight matrix W^2. The input to layer 2 is a^1 and the output is a^2. A layer whose output is the network output is called an output layer. The other layers are called hidden layers. The network shown in Fig. 8 has an output layer (layer3) and two hidden layers (layers 1 and 2)

STRUCTURE AND OPERATION OF MULTILAYER PERCEPTRON NEURAL NETWORK (MLP)

MLP neural networks consist if units arranged in layers [5]. Each layer is composed of nodes and in the fully connected network considered here each node connects to every node in subsequent layers. Each MLP is composed of a minimum of three layers consisting of an input layer, one or more hidden layers and an output layer. The input layer distributes the inputs to subsequent layers. Input nodes have liner activation functions and no thresholds. Each hidden unit node and each output node have thresholds associated with them in addition to the weights. The hidden unit nodes have nonlinear activation functions and the outputs have linear activation functions. Hence, each signal feeding into anode in a subsequent layer has the original input multiplied by a weight with a threshold added and then

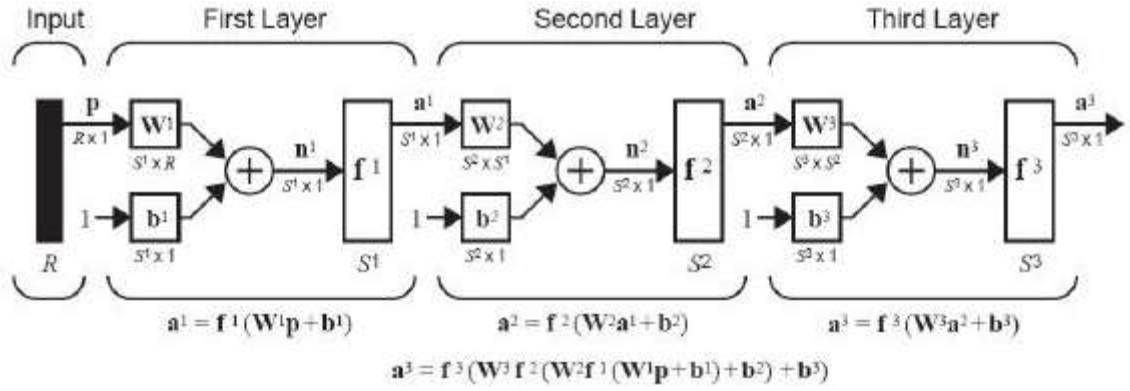

Fig. 8: Three layer network

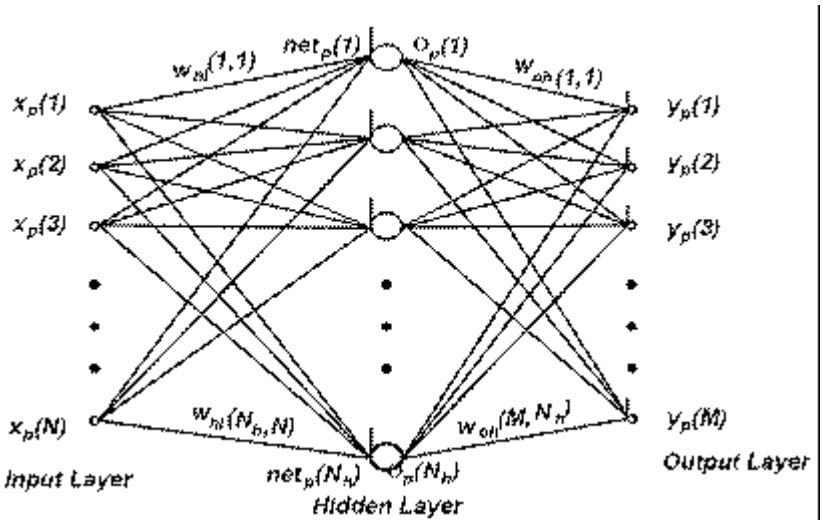

Fig. 9: Typical three layer multilayer perceptron neural network

is passed through an activation function that may be linear or nonlinear (hidden units). A typical three layer network is shown in Fig. 9. Only three layer MLPs will be considered in this work since these networks have been shown to approximate any continuous function [6-8]. For the actual three layers MLP, all of the inputs are also connected directly to all of the outputs [5].

The training data consist of a set N_V training patterns (x_p,t_p) where P represents the pattern number. In Fig. 9, X_P corresponds to the N-dimensional input vector of the Pth training pattern and Y_P corresponds to the M-dimensional output vector from the trained network for the P_{th} pattern. For ease of notation and analysis, threshold on hidden units and output units are handled by assigning the value of one to an augmented vector component denoted by $X_p (N+1)$. The output and input units have linear activations. The input to the J_{th} hidden unit, $net_P (j)$ is expressed [5] as:

$$\text{net}_p(j) = \sum_{k=1}^{N+1} W_{hi}(j,k) X_p(k) \quad 1 \le j \le N_h \tag{7}$$

With the output activation for the P_{th} training pattern, $O_p (j)$, being expressed by:

$$O_p(j) = f(\text{net}_p(j)) \tag{8}$$

The nonlinear activation is typically chosen to be the sigmoid function

$$f(\text{net}_p(j)) = \frac{1}{1+e^{-\text{net}_p(j)}} \tag{9}$$

In (7) and (8), the N input units are represented by the index K and W_{hi} (J,K) denotes the weight connecting the K^{th} input unit to the J^{th} hidden unit.

The overall performance of the MLP is measured by the mean square error (MSE) expressed by :

$$E = \frac{1}{N}\sum_{p=1}^{Nv} E_p = \frac{1}{N}\sum_{p=1}^{Nv}\sum_{i=1}^{M}[t_p(i) - y_p(i)]^2 \tag{10}$$

Where:

$$Ep = \sum_{i=0}^{M}[t_p(i) - y_p(i)]^2 \tag{11}$$

E_p Corresponds to the error for the P^{th} pattern and t_p is the desired output for the P^{th} pattern. This is also allows the calculation of the mapping error for the i^{th} output unit to be expressed by:

$$E_i = \frac{1}{N_v}\sum_{p=1}^{M}[t_p(i) - y_p(i)]^2 \tag{12}$$

with the i^{th} output for the P^{th} training pattern expressed by:

$$Y_p(i) = \sum_{k=1}^{N+1} W_{oi}(i,k) X_p(k) + \sum_{j=1}^{N_h} W_{oi}(i,j).O_p(j) \tag{13}$$

In (13), $W_{oi}(i,k)$ represents the weight from the input nodes to the output nodes and W_{oh} (i,j) represents the weight from the hidden nodes to the output nodes.

CONCLUSION

When investigators design neural networks for the application presented in this paper, there are many ways used to investigate the effects of network structure which refers to the specification of network size (i.e. number of hidden units) when the number of inputs and

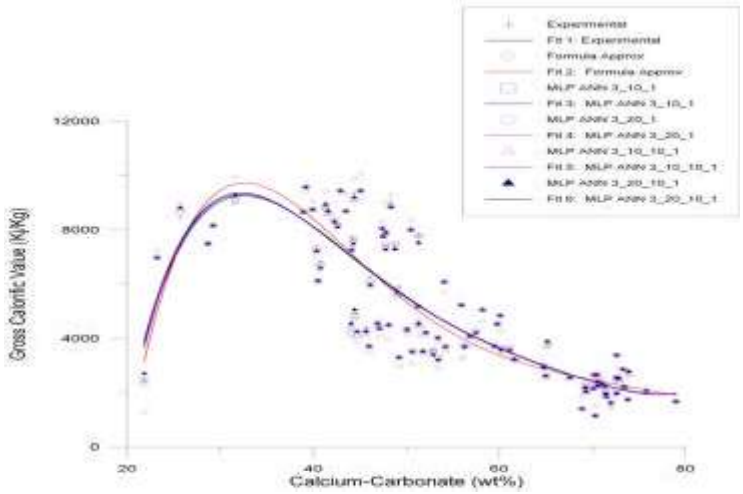

Fig. 10: CaCo3 versus gross calorific value

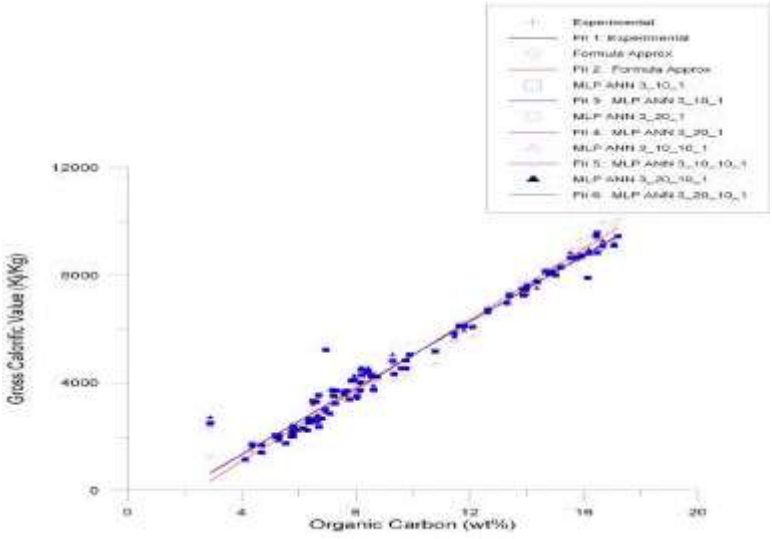

Fig. 11: Corg versus gross calorific value

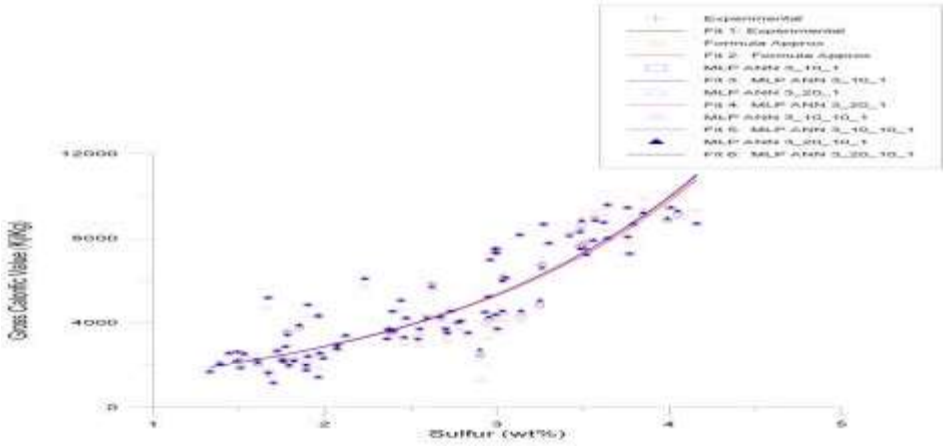

Fig. 12: S versus gross calorific value

Table 1: Results summary

Performance	MLP ANN 3:10:1	MLP ANN 3:20:1	MLP ANN 3:10:10:1	MLP ANN 3:20:10:1
MSE	120517.0498	94695.29502	105564.1689	96255.51725
NMSE	0.019660591	0.01544815	0.017221248	0.015702678
MAE	239.4225634	213.4408001	220.7435148	209.1266576
Min Abs Error	7.123826454	1.171097513	8.106955387	1.71756787
Max Abs Error	1427.779547	1184.631573	1375.296117	1307.388839
r	0.990186314	0.992293629	0.991392243	0.992167131

outputs are fixed (5) which in turn affects on the o/p predicted parameters. A well organized one maybe based on designing a set of different size networks in an ordinary fashion, each with one or more hidden units than the previous one. This approach can be designated as design methodology one (DM-1). Alternatively, the second one is based on designing different size networks in no particular order. This approach can be designated as design methodology two (DM-2).

These two design approaches are significantly different and the approach chosen often is based on the goals set for the network design and the associated performance. In general, the more thorough approach often used for DM-l may take more time to develop the selected network since we may be interested in trying to achieve a trade-off between network performance and network size. However, the DM-1 approach used in this work may produce a superior design since the design can be pursued until we will be satisfied that future increases in network size produces diminishing returns in terms of decreasing training time or testing errors. When we design MLP network to analyze the properties of Jordan oil shale, we reach to the following facts: MLP with a lot of hidden layers (3) performs more better than with one or two hidden layer specially regarding the output performance parameters. Also, when we compare between the o/p of MLP and the mathematical formula, we found that our o/p performance parameter are best suited with the experimental. As a future work, we recommend to use hybrid approaches, which are a combination of ANN with other techniques like expert systems, Fuzzy logic and Genetic Algorithm (GA) to make such analysis of Jordan oil Shale properties.

REFERENCES

1. Anabtawi, M.Z. and jamal M. Nazzal, 1994. Effect of composition of el-lajjun oil shale on its calorific value. Journal of Testing and Evaluation, pp: 175-178.
2. Nitin Malik, 2005. Artificial Neural Networks and their Applications. National conference on Unear thing Technological Developments & their transfer for serving Masses, GLA ITM, Mathura, Mathura, India.
3. Martin, T. Hagan and Howard B. Demuth, Neural Networks for control, www.uldaho.edu
4. Haykin, S., 1999. Neural Networks: A Comprehensive Foundation, 2nd Edn., New Jersey: Prentice-Hall.
5. Walter, H. Delashmit and Michael T. Manry, 2005. Recent Developments in Multilayer Perceptron Neural Networks. Proceedings of the 7^{h} Annual Memphis Area Engineering and Science Conference, MAESC 2005.
6. Hornik, K., M. Stinchcombe and H., White, 1989. Multilayer feed forward Networks are Universal Approximators, Neural Networks, 2 (5): 35g, 366.
7. Hornik, K., M. Stinchombe and H. White, 1990. Universal Approximation of an unknown Mapping and its Derivatives Using Multilayer Feed forward Networks. Neural Networks, 3 (5): 551-560.
8. Cybenko, G., 1989. Approximation by superposition of a sigmoidal function. Mathematics of control, Signal and Systems, 2 (4): 303-314.