Sediment Loads prediction Using Multilayer Feedforward Neural Networks

A. Taher-shamsi, M. B. Menhaj, R. Ahmadian

ABSTRACT

Sediment transport as a complicated and important phenomenon has attracted a lot of researchers during the last century; however there are some formulae to evaluate sediment loads in aquatic systems. Most of them still face two major problems: firstly, lack of accuracy and secondly, involvement of many parameters which makes them more challenging.

Artificial Neural Networks are known as model-free universal function approximators well suited to deal with real life engineering problems including time series predictions and parameter estimation. In this paper, sediment loads are predicted using two different types of multilayer feedforward neural networks, namely Multi-Layer perceptron (MLP) and Radial Basis Function (RBF). The input variables for both structures are considered to be flow discharge, mean flow depth and width, mean bed material's diameter and water surface slope and the output is sediment discharge. Some different cases have been studied. The results are promising. It has been also observed that mean square prediction errors for the developed MLP is equal to 0.0063 while the devised RBF networks produces much larger mean square errors, namely 0.01260. This indicates that the MLP- load-predictor outperforms the RBF-predictor.

KEYWORDS

Sediment transport, Hydroinformatic, Artificial Neural Networks (ANNs), Multi-Layer perceptron (MLP), Marquadt-Levenberg, Radial Basis Function (RBF)

1. INTRODUCTION

Erosion and sediment transports have been an undetectable part of the aquatic cycle and affected the exploitation of the surface water known as an important water source for miscellaneous uses. Therefore, sediment transport problems have been studied by researchers for a long time. A lot of formulas are available for sediment transport estimation. However, most of the existing sediment transport formulae that indeed estimate sediment transport rate have a restricted range of applicability. Many of them are also based on empirical or semi-empirical methods. Each of these formulas is derived due to different characteristics of fluid, different parameters of flow or variety of bed materials. The flow, fluid and bed material characteristics in which a formula has been derived define the range of applicability that particular formula and the use of any formula beyond its applicability range usually shows dramatic results.

An alternative way to estimate suspended sediment loads is to apply power law, flow discharge and sediment concentration, which is based on stochastic analyses of the measured data at the particular section of the river [1]. Although power law has the privilege of being straightforward, it does not show proper results for transient states and cannot be applied to new streams or those without enough measured data. It is worth mentioning that those formulas which use more parameters cannot guarantee good acceptable results. With attention to the increasing usage of Computational Intelligence (CI) in different areas of science, the CI based methods beside their simplicity and generalizing ability have been used widely in hydraulic, e.g. non-iterative friction factor in pipes [2], flow forecasting [3], optimization of hydraulic

A. Taher-shamsi, Asst. Professor, Department of Civil and Environmental Engineering, Amirkabir University of Technology, Tehran, Iran (Email: atshamsi@aut.ac.ir).

M. B. Menhaj, Professor, Department of Electrical Engineering, Amirkabir University of Technology, Tehran, Iran, (Email: mb.menhaj@aut.ac.ir).

R. Ahmadian, PhD Candidate, Hydroenvironmental Research centre, Cardiff University, UK, (Email: AhmadianR@cf.ac.uk)

river flow modeling [4] and classification of river basin [5].

Furthermore, some researches have used artificial neural networks (ANNs) to predict sediment transport. For instance, Avarideh et. al. employed ANNs to predict sediment in Kor river by the means of river discharge as input data [6], Jain developed an integrated sediment curve using ANNs [7]. Namin et. al. used some fluid and sediment characteristics to predict first bed reference concentration and dimensionless bed load transport parameters and they eventually used these parameters to develop a two dimensional model to predict sediment concentration [8].

This paper extends the method presented earlier by the authors in [9] which used type of modular networks to predict sediment load. Here, two different types of ANNs, namely MLP and RBF, are developed to estimate sediment transport. Different sets of laboratory and field data are used to test the neure based sediment loads predictors. The rest of the paper is organized as follows. Section 2 states the sediment and suspended load problems. In section 3, the neural networks used in the paper are presented. The source of loads date and its characteristics are given in section 4. Section 5 is devoted to present some case studies and discuss the results of simulations. Finally, section 6 concludes the paper.

2. Sediment and Suspended Load

Sediment is fragmental material, primarily formed by physical and chemical disintegration of rocks of the earth's crust. Such particles range in size from large boulders to colloidal size fragments and vary in shape from rounded to angular. They also vary in specific gravity and mineral composition [1]. In accordance to ISO-standards (ISO4363) the bed material load is classified into material moving as bed load where the sediment is in almost continuous contact with the bed, or suspended load in which the sediment maintained in suspension by turbulence for considerable periods of time without contact with the bed [10].

Besides fluid and bed material characteristics, transport capacity of the flow, which is related to some flow parameters, e.g. flow discharge, mean depth and width of the flow and finally the slope of the water surface, is effective on suspended sediment transport.

3. Artificial Neural Networks

ANNs mimic biological information processing mechanisms constructed by electronic processing elements (PEs) connected in a particular fashion. The behavior of the trained ANN depends on the weights, which are also referred to as strength of the connections among PEs [11]. ANNs are counted as model free intelligent dynamic systems which are mainly based on experimental data. While processing the input data, ANNs disclose the hidden rules behind the data and transfer them into the network structure. In other words, ANNs learn general rules by the means of Numerical computation on the input data and for this respect they are called intelligent [12].

ANNs have been applied to the variety of automation problems including adaptive control, optimization, medical diagnosis, decision making as well as information and signal processing including speech processing [11]. Below, two very popular types of feedforward neural networks, Multilayer percerptron (MLP) and radial basis function (RBF) used in the paper are briefly introduced.

3-1 Multilayer Perceptrons

MLP structure is a standard combination of inputs, outputs and linear and/or non-linear units. Fig. 1 depicts a general block diagram representation of the MLP. All processing units' outputs in each layer are connected to every processing unit in next layer. The input layer indeed represents the input vector which must be properly defined for each specific problem. The processing units in hidden layers known as non-linear neurons usually take sigmoid, hyperbolic or any other non linear differentiable type functions as their activation function. Linear neurons are commonly exploited in the output layer to cover the practical range of the underlying problem which is being modeled and most probably to have a better learning speed.

Approximate gradient-desecent method which is the basis of back propagation (BP)-algorithm is one of the simplest methods though it is not a very effective one. Further numerical optimization theory was applied to speed up significantly the convergence of the BP-algorithm. This theory helps us have a rich and robust set of methods that one can apply to multilayer feedforward neural networks to enhance learning rates especially for off-line training the network.

It is worth mentioning that the gradient steepest-descent method considers only the first-order derivative of an performance index or indeed an error function. This helps take into account pattern-by-pattern training if we have no a bunch training data in advance. It is also helpful to take into account higher order derivatives. To see this better, one may use Taylor's series expansion on the error function, $J(\underline{\theta})$ around the current point $\underline{\theta}^0$, we have

$$J(\underline{\theta}) = J(\underline{\theta}^0) + (\Delta\underline{\theta})^T \left(\nabla_{\underline{\theta}} J(\underline{\theta})\right)_{|\underline{\theta}=\underline{\theta}^0} +$$

$$0.5(\Delta\underline{\theta})^T H(\underline{\theta})_{|\underline{\theta}=\underline{\theta}^0} (\Delta\underline{\theta}) + \ldots$$

where $\left(\nabla_{\underline{\theta}} J(\underline{\theta})\right)_{|\underline{\theta}=\underline{\theta}^0}$ is the gradient of the error function with respect to $\underline{\theta}$ evaluated at $\underline{\theta}^0$ and H is called a Hessain matrix and is the second derivative

$$H(\underline{\theta}) \triangleq \nabla^2 J(\underline{\theta}) \Leftrightarrow H_{ij} \triangleq \frac{\partial^2 J}{\partial\theta_i \partial\theta_j}$$

We should note that the parameter vector $\underline{\theta}$ contains all the network weights and biases and $\Delta\underline{\theta} = \underline{\theta} - \underline{\theta}^0$.

To find the minimum of $J(\underline{\theta})$, we set its gradient to zero:

$$\nabla_{\underline{\theta}} J(\underline{\theta}) = \nabla_{\underline{\theta}} J(\underline{\theta}^0) + H(\underline{\theta})_{|\underline{\theta}=\underline{\theta}^0}(\Delta\underline{\theta}) + \ldots = 0$$

Ignore the third and higher-order terms to obtain

$$\underline{\theta} = \underline{\theta}^0 - H^{-1}(\underline{\theta})_{|\underline{\theta}=\underline{\theta}^0} \nabla J(\underline{\theta})_{|\underline{\theta}=\underline{\theta}^0}$$

If we use k to indicate the kth step of training, we can write

$$\underline{\theta}(k+1) = \underline{\theta}(k) - H^{-1}(\underline{\theta}(k))\nabla J(\underline{\theta}(k))$$

This is called Newton's method of parameter adjustment. This method employs the second derivatives in addition to the gradient to calculate the next step direction and step size systematically. Its rate of convergence is of order 2 when it is close to the solution of a convex function [12]. However, in order to converge, it requires a good initial estimate of the parameters (the solution) and furthermore for non-convex functions it may converge to a local minimum or a saddle point while at each iteration it needs huge computations to find the Hessain matrix and its inverse leading to an expensive method in terms of both storage and computation requirements. Therefore, some revised method have been developed. These include the conjugate-gradient method and quasi-Newton technique.

The Levenberg-Marquardt Backpropagation (LMBP) is a type of the quasi-Newton method with the following parameter updating rule:

$$\underline{\theta}(k+1) = \underline{\theta}(k) - \left[\nabla^T J(\underline{\theta}(k))\nabla J(\underline{\theta}(k)) + \mu(k) I\right]^{-1} \nabla J(\underline{\theta}(k))$$

The LMBP is summarized below.

1- Present all inputs to the network and find the corresponding outputs of the network and the errors $\underline{e}^j = \underline{t}^j - \underline{a}^j$ (the error for the jth input pattern, $\underline{t}$ and $\underline{a}$ are respectively the target and the actual output). Determine the sum of squared errors over all inputs, $J(\underline{\theta}) = \sum_{j=1}^{L} (\underline{e}^j)^T \underline{e}^j$.

2- Compute the Jacobian matrix $\nabla J(\underline{\theta}(k))$ and

$$\Delta\underline{\theta}(k) = \left[\nabla^T J(\underline{\theta}(k))\nabla J(\underline{\theta}(k)) + \mu(k) I\right]^{-1} \nabla J(\underline{\theta}(k))$$

3- Re-compute the sum of squared errors using $\underline{\theta}(k) + \Delta\underline{\theta}(k)$. If this is smaller than that compter in step 1, then divide $\mu(k)$ by some factor $\rho > 1$ (e.g., ρ=10) and go back to step 1; this makes the algorithm approach Gauss-Newton, which provides faster convergence. If the new sum of squared errors is not reduced, then multiply $\mu(k)$ by ρ and go to step 2. This makes the algorithm work like the standard BP in which

$$\underline{\theta}(k+1) = \underline{\theta}(k) - \frac{1}{2\mu(k)} \nabla J(\underline{\theta}(k))$$ and

eventually the error function should decrease since a small step in the direction of steepest descent is being taken.

The LMBP appears to be the fastest method for training moderate-sized feedforward networks (up to several hundred weights) [13-14].

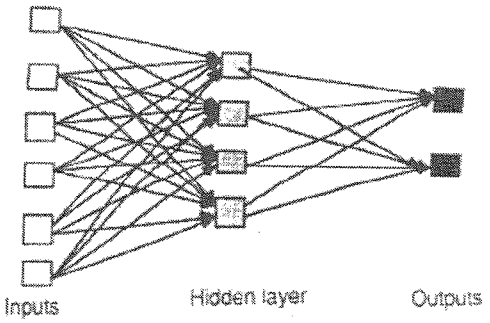

Figure 1: A block diagram representation of a simple feedforward network (MLP)

3-2. Radial Basis Function

A viable alternative to highly nonlinear-in-the-parameters neural networks is the radial basis function [13,15]. It was first introduced in the solution of real multivariable interpolation problems. Like MLP neural networks, RBF networks are suited to applications such as pattern discrimination and classification, pattern recognition, interpolation, prediction and forecasting, and process modeling.

As it can be observed in Fig. 2, RBF networks consist of two layers. The neurons in the first layer do not employ the weighted sum of the external inputs and the sigmoid activation function typically used in MLP. The outputs of the first layer neurons, each of which represents a basis function, are instead computed by the distance between the network input and the center of the basis function. The neuron output decays rapidly to zero as the input moves away from a given center. The second layer of the RBF network is linear and produces a weighted sum of the outputs of the first layer. Because the neurons in this network only respond to inputs that are close to their centers, these neurons will have localized receptive fields in contrast to the MLP networks where the sigmoid function creates a global response. The RBF networks train faster that MLP networks, however require many neurons for higher dimensional input spaces.

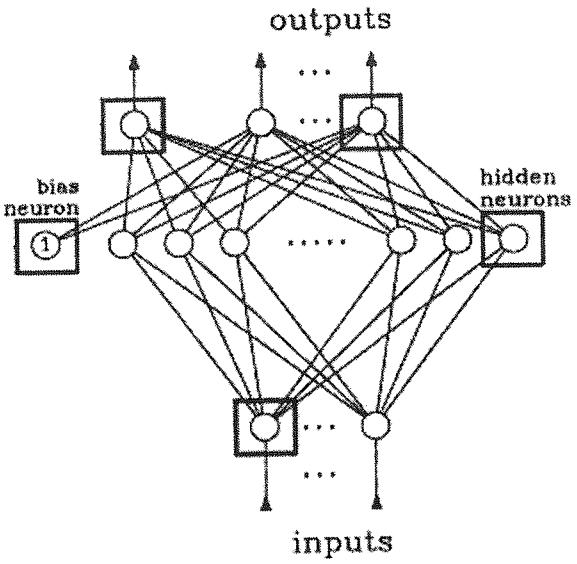

Figure 2: Schematic of Radial Basis Function Network

Unlike the MLP, the hidden neurons in the RBF neywork contain the *Radial Basis Function*, a statistical transformation based on a Gaussian distribution from which the neural network's name is derived. Each hidden neuron takes as its input all the external inputs, p_i s. The hidden neuron has the parameters "centre" and "width" of the basis function. The centre of the basis function is a vector of numbers c_i of the same size as the inputs to the neuron and there is normally a different centre for each neuron in the RBF. The first computation performed by the neuron is to compute the "radial distance", d, between the input vector p_i and the centre of the basis function, typically using Euclidean distance:

$$d = \sqrt{\sum_{i=1}^{n}\left(p_i - c_i\right)^2}$$

The output a is then computed by applying the basis function B to this distance divided by the width w:

$$a = \mathrm{B}\left(\frac{d}{w}\right)$$

The basis function is a curve (typically a Gaussian function as depicted in figure 3) which has a peak at zero distance and which falls to smaller values as the distance from the centre increases. As a result, the neuron gives an output of one when the input is centered, however it reduces as the input becomes more distant from the centre. There are three sets of variables that affect each nodes input to the solution, the position, the width and the output layer weight.

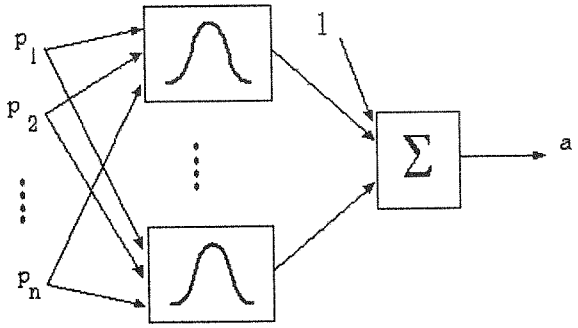

Figure 3. An RBF with one output

Training occurs by adjusting the variables mentioned above to improve the modeling accuracy of the network. The training can be accomplished in a number of ways, from self-organizing techniques similar to SOM's through to supervised techniques as used in feedforward networks which will be briefly formulated below.

The RBF network is designed to perform input-output mapping trained by empirical examples $\left(\underline{p}^q, \underline{t}^q\right), q = 1,2,...,Q$. It is based on the concept of the locally tuned and overlapping receptive field structure studied in the cerebral cortex, the visual cortex as shown in figures 2 and 3.The hidden nodes have normal Gaussin activation function

$$n_j = \frac{R_j\left(\underline{p}\right)}{\sum_{i=1} R_i\left(\underline{p}\right)}$$

where $R_j\left(\underline{p}\right)$ is the receptive field in the input space related to the hidden neuron j. $R_j\left(\underline{p}\right)$ defined below is a region centered on $\underline{c}_j$ (mean as an n-dimensional vector) with a size proportional to σ_j (variance) of the jth Gaussian function.

$$R_j\left(\underline{p}\right) = e^{-\frac{\left|\underline{p}-\underline{c}\right|^2}{2\sigma_j^2}}$$

The output of the RBF network is indeed a weighted sum of the hidden neuron outputs as given by

$$a_i\left(\underline{p}\right) = f_i\left(\sum_{j=1}^{S^1} w_{ij} n_j + b_i\right)$$

where f is the output activation function and b_i is the bias value. In general, f is a linear function meaning that the output nodes are linear and $b_i = 0$.

The main goal of the RBF network is to cover the input space with overlapping receptive fields; it means that for an input vector lying somewhere in the input space, the

receptive fields with centers close to it will be considerably more activated. This in turn leads to have these fields contribute more in the output of the RBF network. In the extreme case if the input lies in the center of the field for a hidden neuron j, and we ignore the overlaps between the fields, then only the hidden neuron j will be activated (called as the winner neuron) and the corresponding weight vector

$$\underline{w}_j = \left[w_{1j}, w_{2j}, ..., w_{S^2 j} \right]^T$$

is selected as the output.

This network is mainly trained by an unsupervised learning in the input layer and supervised learning in the output layer. The weights in the output layer are simply updated

$$\Delta w_{ij} = \alpha \left(t_i - a_i(p) \right) n_j$$

This will minimize the performance index (cost function) defined below if it is averaged over the Q training pairs of data.

$$J\left(w_{ij}; \underline{p}\right) = 0.5 \sum_{q=1}^{Q} \sum_{i=1}^{S^2} \left(t_i^q - \sum_{j=1}^{S^1} w_{ij} f_j \left(\underline{p}^q \right) \right)^2$$

The learning can have another part known as unsupervised part in which the centers and widths $\left(\underline{c}_j, \sigma_j\right), , j = 1, 2, ..., S^1$ are determined through unsupervised learning rules arc such as competitive learning rules, vector quantization method or simply by the following rule:

$$\Delta \underline{c}_w = \alpha \left(\underline{p} - \underline{c}_w \right)$$

where $\underline{c}_w$ is the center of the receptive field which is closest to the input vector and the other centers are kept fixed [16].

Once the field centers have been determined, their widths may be determined through minimization of the following performance function with respect to the width:

$$J\left(\sigma_j\right) = 0.5 \sum_{j=1}^{S^1} \left(\sum_{k=1}^{S^1} R_j\left(\underline{c}_k\right) \left| \frac{\underline{c}_j - \underline{c}_k}{2\sigma_j} - \beta \right| \right)^2$$

where β is the overlap parameter. Doing so makes sure that the field neurons form a smooth and contiguous interpolation over those regions of the input space they are representing. Practically, the widths are usually found by an unsystematic manner (ad hoc technique) such as the mean distance to the β- or first- nearest- neighbors heuristic may be used as:

$$\sigma_j = \left| \frac{\underline{c}_j - \underline{c}_w}{\beta} \right|$$

where $\underline{c}_w$ is the closest vector to $\underline{c}_j$.

The RBF network can be trained by the BP-based learning algorithms too. The goal here will be to minimize globally the error function $J(W)$ given earlier in this section. In this way, the parameters of output layer are still adjusted like before and will also be given below and the output error is then backpropagated to the hidden neurons (receptive fields neurons) to adjust their centers and widths. Based on the chain rule and the BP- rule the RBF learning mechanism can be summarized as:

$$\Delta w_{ij} = \alpha \left(t_i - a_i(p) \right) n_j,$$

$$\Delta \underline{c}_j = \gamma \sum_{i=1}^{S^2} \left(t_i - a_i(p) \right) \frac{\partial a_i(p)}{\partial \underline{c}_j},$$

$$\Delta \sigma_j = \eta \sum_{i=1}^{S^2} \left(t_i - a_i(p) \right) \frac{\partial a_i(p)}{\partial \sigma_j},$$

where the sensitivities $\frac{\partial a_i(p)}{\partial \underline{c}_j}$ and $\frac{\partial a_i(p)}{\partial \sigma_j}$ can be determined using the chain rule on the equations of n_j and a_i. This gives a systematic approach to adjust the centers and widths dynamically. Unfortunately, the RBF network with this learning rule does not learn fast enough.

Another leaning rule for the RBF network is based on the orthogonal least squares learning algorithm [15]. This method selects the centers one by one in a rational manner till an adequate network being constructed and provides the network node-growing capability in which a hidden neuron (receptive node) can be added if necessary to account for a new pattern lies far away from the existing fields.

The RBF method has traditionally been used for strict interpolation in multidimensional space [17-18]. RBF networks are in the Kernel clustering group [12] and require more neurons than standard feedforward networks, but often they can be designed in a fraction of the time to train standard feedforward networks. They work best when many training vectors are available [13]. It has been shown that the RBF networks can approximate an arbitrary function with just one hidden layer and a Gaussian sum density estimator can approximate any probability density function to any desired degree of accuracy [19-20].

4. Source of Data

Valuable information is available from a compendium of solid transport data for mobile boundary channel containing forty-nine sets of data with 5892 records. Each includes different hydraulic and sediment parameters selected from the work of different authors [21]. Some of

these data are measured in field and the rest are laboratory results. Therefore, they represent a wide range of hydraulic conditions of flow. A summary of the data is presented in Table 1. It is evident that each set of data has its own specifications, appropriately selected by the researcher; therefore, there is not a complete similarity between the selected parameters in each of these data sets. Table 2 summarizes data specifications used in the paper. In this study, 5 parameters are used as input data: namely, water discharge, depth, width, slope and sediment mean size. The output is sediment concentration.

5. MLP STRUCTURE

In this section, one or two hidden layers MLP networks are used to predict sediment discharge with Marquardt-Levenberg learning algorithm. Network parameters are set as shown in Table 3. All records are classified randomly in three categories, namely: training, validation and test data sets, with ratios of 70, 20 and 10 percent, respectively. As a result, 4044 of these records were used for training, 1145 for validation of ANNs to prevent over training, and the rest were used for testing the network. Matlab software is employed for the purpose of simulation to create and train the networks based on the whole related parameters as shown in Table 3. Due to the wide range of varieties for each variable, see Table 1, input and output variables were normalized between -1 and +1 according to the following equation:

$$P_n = \frac{2(P - P_{\min})}{P_{\max} - P_{\min}} - 1 \quad (1)$$

In which P_n is the normalized value of parameter P, $P_{\max}$ and $P_{\min}$ are maximum and minimum values of parameter P, respectively.

To achieve the number of hidden layers and number of neurons in each layer, different networks with different neurons in one or two hidden layers are created. A network that consists of one hidden layer with 16 nodes [22] produces the best result. Its result is shown in Table 4 and error percentage for the test data set is illustrated in Figure3. The error percentage is defined as follow:

$$Error(\%) = \frac{T_p - T_r}{T_r} \quad (2)$$

In the above Tp is the predicted value and Tr is the real value of the predicted variable.

As it is shown in Table 4, the mean square errors (MSE) of the training and verification data set are 6.3E-3 and 8.1E-3, respectively. For the test data set, the simulation results are promising while the MSE of predicted values is 7.2E-3 and the correlation coefficient (R2) of ANNs prediction errors is 77.2%.

TABLE1. SUMMARY OF THE DATA

Data Type	Parameter	Unit	Max	Min	Mean	STD
Input	Discharge	m^3/s	660	0.0008778	9935.2	62833
Input	Width	m	188.94	0.1524	99.755	398.68
Input	Depth	m	2.4414	0.0070104	2.1117	6.3761
Input	Slope	-	4	0.01083	0.46488	0.73697
Input	D_{50}	m	0.015	1.30E-5	6.7292	15.948
Output	Sediment Concentration	PHT(*)	19866	0.01	570.86	2740.5

(*)Material by weight of water-sediment expressed in part per 100,000.

Table2. Number of Each Parameter in All Data Set

No.	Discharge	Width	Depth	Slope	D_{50}	Gradation	Specific Gravity	Concentration	Water Temperature	Bed form
6338	×	×	×	×	×		×	×		
4106	×	×	×		×	×	×	×	×	
4874	×	×	×		×		×	×		
6290	×	×	×	×	×		×	×		
1853	×	×	×		×		×	×		×
924	×	×	×	×	×	×	×	×	×	×
3655	×	×	×	×	×	×	×	×	×	

×: measured parameters

TABLE3. MLP NETWORKS PARAMETERS

performance Function	mse
Activation Function in Hidden Layers	tansig
Activation Function in Output Layer	purelin
Max Epochs	2000
Goal	1 e -4
time	5000
mu	1
mu - inc	1.5
mu - dec	0.8
mu - max	1 e 50

As it is depicted in Fig. 3, the majority of the error percentages are less than 5% whereas few local large errors can be observed.

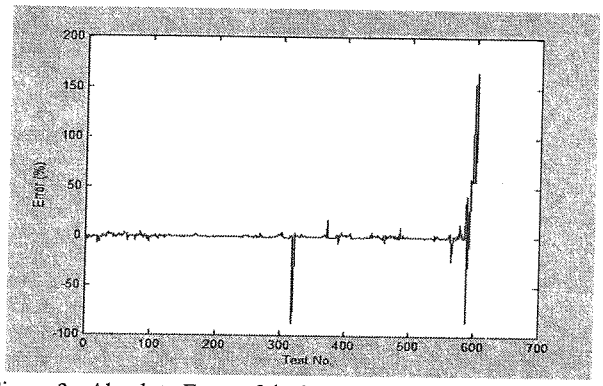

Figure 3: Absolute Error of the MLP Network Predictions on Test Data Set

Structure	MLP	RBF
MSE Normalized Train	0.00629792	0.0126
MSE Normalized Test	0.0072	0.0148
MSE Test	2.60E+06	5.33E+06
STD Normalized Errors	0.0849	0.1212
STD Errors	1.61E+03	2.30E+03

TABLE4. MLP AND RBF NETWORKS RESULTS

6- RBF NETWORKS

Due to the Better results of RBF networks in some cases [12], in this section we used RBF network to predict sediment loads. We also used the same source of data for this part of study. According to the structure of RBF networks all available data was randomly classified in training and testing data set with the ratio of 90%, 5189 records, and 10%, 603 records, respectively. The "*Newrbe*" function in Matlab software is employed to train the networks used in this part. Input and output variables are also normalized analogous to MLP networks with "*premnmx*" function of Matlab. In order to create the RBF network, the *newrbe* function is used. This led to the same number of neurons in its hidden layer as input vectors. This function can produce a network with zero error on training vectors.

The function *newrbe* takes matrices of input vectors P and target vectors T, and a spread constant SPREAD, which is defined by user, for the radial basis layer. So we have a layer of *Radbas* neurons in which each neuron acts as a detector for a different input vector [14]. To achieve the best network, different spreads was used and at last SPREAD = 0.03 gave the best result [22].

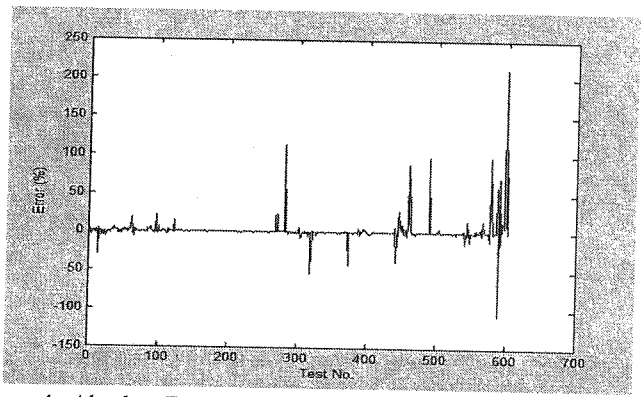

Figure 4: Absolute Error of the RBF Network Predictions on Test Data Set

The results of the RBF network are shown in Table 4 and the corresponding error percentage for the normalized test data is depicted in Fig. 4. This figure obviously shows that the error percentage level for the

RBF network is higher than the MLP network and it is consistent with Table 4. This approves the claim that although the RBF networks in general cannot uite achieve the accuracy of the MLP with BP-based training algorithm, it can be trained faster than the BP-based MLP.

7. CONCLUSIONS

By comparing the results of two networks shown in Table 4, it is concluded that the MLP network has less standard deviations for both normalized and raw data sets besides that raw and normalized mean squared errors of the RBF network is approximately twice the raw and normalized squared errors of the MLP network. These consequences give rise to the fact that sediment loads predicted by the MLP network is more reliable than those of predicted by the RBF network. It should be also mentioned that both methods badly suffer from a few numbers of relatively enormous errors. Evaluating different approaches to remedy this problem, e.g. use of Bad Data Detection algorithm, and other use of different approaches to improve network performance, e.g. data classification, change of input parameters and use of modular networks, are suggested for future work.

REFERENCES

[1] L. C. Van Rijn, "Principles of Sediment Transport in Rivers, Estuaries and Costal Seas," Aqua Publications, The Netherlands, 1993

[2] H. Shayya and S. Sablani, "Artificial neural network for non-iterative calculation of the friction factor in pipe flow," Computers and Electronics in Agriculture , Vol. 21, No. 3, pp. 219–228, 1998

[3] N. T. Danh, H.N. Phien, and A.D. Gupta," Neural network models for flow forecasting," Water SA, Vol. 25, No. 1, pp. 33–39, 1999

[4] N. G. Wright, M. T. Dastorani, P. Goodwin, and C.W. Slaughter, "Using artificial neural networks for optimization of hydraulic river flow modeling results," Proceedings of the International Conference on River Flow, Louvain-al-Neuve, Belgium, 2002, 1–10

[5] B.S. Thandaveswara and N. Sajikumar, "Classification of river basins using artificial neural network," ASCE Journal of Hydrologic Engineering, Vol. 5, No. 3, pp. 290–298, 2000.

[6] F. Avaride, M.E. Banihabib, and A. Taher-shamsi, "Estimation of Sediment load in Rivers using ANN," in proc. 3rd Iranian Hydraulic Conference, Tehran, Iran,2001 (in Persian).

[7] S. K. Jain, "Development of integrated sediment rating curves using artificial neural networks," Journal of Hydraulic Engineering, ASCE, Vol. 127, No. 1, pp. 30–36, 2001.

[8] M. M. Namin and B. Lin, "A 2D Vertical Hydrodynamic and Morphological Model Based on ANNs," Hydroinformatic 2002, Cardiff, UK.

[9] A. Tahershamsi, R. Ahmadian, and M. M. Menhaj, "Study of Categorized Input Data Effects on Sediment Estimation Using Neural Network", 8th International Conference on Fluvial Sedimentology, the Netherlands, August 2005.

[10] ISO4363:2002, Measurement of liquid flow in open channels -- Methods for measurement of characteristics of suspended sediment

[11] L. C. Jain and N. M. Martin, Fusion of Neural Networks, Fuzzy Sets, and Genetic Algorithms, Industrial Applications, London, CRC Press.

[12] D. J. Luenberger, Linear and nonlinear programming, MA: Addison Wiley, 1976.

[13] M. B. Menhaj and N. Seifipour, Application of Computational Intelligence in Control, Tehran, Dr Hesabi Publications, 1377 (in Persian).

[14] Matlab , Release 13 , Help Document

[15] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks," IEEE Transactions on Neural Networks, Vol. 2, No. 2, March 1991, pp. 302-309.

[16] M. B. Menhaj, Fundamental of neural networks, in the series of computational intelligence, Vol. 1, Amirkabir Press, Tehran-Iran,1999.

[17] C. A. Miccelli C. A., "Interpolation of Scattered Data: Distance Matrices and Conditionally Definite Function," Construct. Approx., vol. 2, pp. 11-22, 1986.

[18] M. J. D. Powell, "Radial Basis Function Approximation to Polynominals," Proc. 12th Biennial Numerical Analysis Conf. (Dundee), pp. 223-241, 1987.

[19] E. J. Hartman, J. J. Heller, and J. M. Kowalski, "Layered Neural Networks with Gaussian hidden Units as Universal Approximators," Neural Comput. 2:210-215, 1990.

[20] H. W. Sorenson and D. L. Alspach, "Recursive Bayesian Estimation Using Gaussian Sums," Automatica vo. 7, no 4, pp-465-479, 1971.

[21] A. W. Peterson and R. F. Howells, "A Compendium of Solids Transport Data for Mobile Boundary Channels, "Report No. Hy-1973-ST3, Department of civil Engineering, University of Alberta, Canada, 1973.

[22] R. Ahmadian, "Estimation of sediments in rivers using neural networks", MSc Dissertation, Dep. of Civil and Environmental Engineering, Amirkabir University of Technology, Tehran, Iran, 2005..