Comparison of Different Neural Networks Performances on Motorboat Datasets

M. Fatih Amasyalı[1], Mert Bal[2], Uğur B. Çelebi,
Serkan Ekinci[3], and U. Kaşif Boyacı[4]

[1] Yıldız Technical University, Computer Engineering Department
İstanbul, Turkey
mfatih@ce.yildiz.edu.tr
[2] Yıldız Technical University, Mathematical Engineering Department
İstanbul, Turkey
mertbal@yildiz.edu.tr
[3] Yıldız Technical University, Naval Architecture Department
İstanbul, Turkey
ucelebi@yildiz.edu.tr, ekinci@yildiz.edu.tr
[4] National Research Institute of Electronics and Cryptology ,TÜBİTAK, UEKAE
Gebze, Kocaeli, Turkey
ukasif@uekae.tubitak.gov.tr

Abstract. Calculation of the required engine power and displacement takes an important place in the initial design of motorboats. Recently, several calculation methods with fewer parameters and with a possible gain of time compared to classical methods have been proposed. This study introduces a novel calculation method based on neural networks. The method requires less data input and hence is more easily applicable than classical methods. In this study several different neural network methods have been conducted on data sets which have principal parameters of motorboats and the respective performances have been presented. From the results obtained, displacement and engine power prediction for motor boats can be used at a suitable level for ship building industry.

1 Introduction

At the predesign stage, the necessary information to calculate ship displacement and engine power is usually obtained by testing similar ship models in the towing tank. These data are converted into characteristic curves, tables and empiric equations via principal dimensions of ship. Moreover, with the developing computer technology, today it is also possible to calculate engine power and displacement using specific computer programs for the dimensions of the ship as inputs.

Recently, calculating design parameters with neural network is found to be much better than the traditional calculation methods in terms of time and costs. In the literature there are some applications of neural networks into naval architecture: neural network applications in naval architecture and marine engineering

A. Levi et al. (Eds.): ISCIS 2006, LNCS 4263, pp. 212–220, 2006.

[1], design of a robust neural network structure for determining initial stability particulars of fishing vessels [2], modeling and simulation of ship main engine based on neural network [3], determination of approximate main engine power for chemical cargo ships using radial basis function neural network [4] can be given as examples.

In this study, different neural network models are applied for determining the engine power and displacement based on principal parameters of current motorboats which have a length of 8 to 25 meters. Input data for the model are used ship length, breadth, draught and speed as principal parameters. System output is the engine power and displacement to be predicted and then the results of these methods are compared to examine which method give better results.

A motorboat's profile and upper view are given below in Figure 1. Main dimensions and related parameters in the illustrations are defined below.

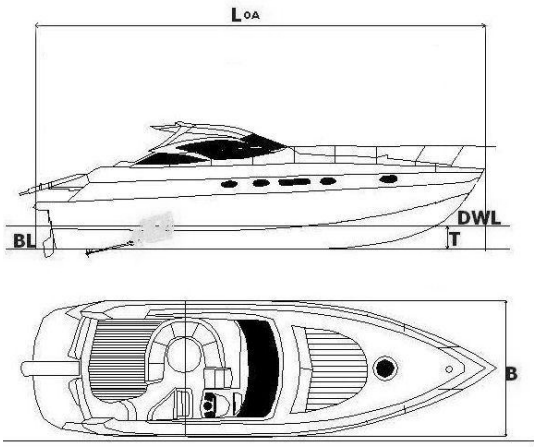

Fig. 1. A motorboat's profile and upper view

(BL) Baseline: The horizontal line parallel to the design waterline (DWL), which cuts the midship section at the lowest point of ship. The vertical heights are usually measured from the baseline.

(LOA) Length Overall: The total length of the ship from one end to the other, including bow and stern overhangs.

(T) Draught: The vertical distance from the waterline at any point on the hull to the bottom of the ship.

(B) Breadth: The distance from the inside of plating on one side to a similar point on the other side measured at the broadest part of the ship.

(V) Ship speed (Knot): The distance in miles taken in an hour.

(∇) Water Displacement (m^3): The water displacement equals the volume of the part of the ship below the waterline including the shell plating, propeller and rudder.

(Δ) Displacement (ton): The displacement is weight of the volume of water displaced by the ship.

$$\Delta(\text{ton}) = \nabla(\text{m}^3)\gamma(\text{ton/m}^3) \tag{1}$$

2 Neural Network Models

In this study different Neural Network architecture and learning algorithms were used. Their basic explanations are given in this section.

2.1 Gradient Descent Back Propagation with Momentum (GDM)

Gradient Descent Back Propagation with Momentum is a generalization of GD introducing the concept of "momentum" in order to avoid trapped in a local minimum. The new weight formula is as follows:

$$\boldsymbol{w_{k+1}} = \boldsymbol{w_k} - \alpha \boldsymbol{g_k} + \lambda \boldsymbol{w_{k-1}} \tag{2}$$

where λ is momentum constant, a number between 0 and 1. If the momentum is 0, then as in GD, the weight change depends on the gradient.

2.2 Gradient Descent with Adaptive Learning Rate Back Propagation (GDA)

Gradient Descent with Adaptive Learning Rate Back Propagation (GDA) is a learning function which updates the weights and bias values according to the gradient descent with adaptive learning rate. At each step, the learning rate is either increased by the factor 1.05 if the performance decreases toward the goal, or decreased by factor 0.7 if the performance grows by more than the factor 1.05 and the performance increasing change is not applied.

2.3 Levenberg Marquardt Method

Levenberg-Marquardt method is one of the best known curve-fitting algorithms. The method works better than the Gauss-Newton method even if the starting point is very far off the desired minimum. To better illustrate the advantage of LM method over Newton method, consider a multilayer feedforward neural network.

The Newton method calculates the weight function recurrently as:

$$\boldsymbol{w_{n+1}} = \boldsymbol{w_n} - [H\psi(\boldsymbol{w_n})]^{-1}\nabla\psi(\boldsymbol{w_n}) \tag{3}$$

where $\psi : \mathbb{R}^K \rightarrow \mathbb{R}$ is the corresponding error function

$$\psi(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{e}(\boldsymbol{w})^T\boldsymbol{e}(\boldsymbol{w}) \tag{4}$$

The exact Hessian matrix calculation is too expensive and the condition that Hessian matrix should be positive definitive is too restrictive.

To overcome these difficulties, LM method proposes a positive-definite approximation $H\psi(\boldsymbol{w}) \approx \boldsymbol{J}(\boldsymbol{w})^T\boldsymbol{J}(\boldsymbol{w}) + \mu\boldsymbol{I}$ where

$$\boldsymbol{J}(\boldsymbol{w}) = \begin{bmatrix} \frac{\partial e_{11}(\boldsymbol{w})}{\partial w_1} & \frac{\partial e_{11}(\boldsymbol{w})}{\partial w_2} & \cdots & \frac{\partial e_{11}(\boldsymbol{w})}{\partial w_K} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_{L1}(\boldsymbol{w})}{\partial w_1} & \frac{\partial e_{L1}(\boldsymbol{w})}{\partial w_2} & \cdots & \frac{\partial e_{L1}(\boldsymbol{w})}{\partial w_K} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_{1Q}(\boldsymbol{w})}{\partial w_1} & \frac{\partial e_{1Q}(\boldsymbol{w})}{\partial w_2} & \cdots & \frac{\partial e_{1Q}(\boldsymbol{w})}{\partial w_K} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_{LQ}(\boldsymbol{w})}{\partial w_1} & \frac{\partial e_{LQ}(\boldsymbol{w})}{\partial w_2} & \cdots & \frac{\partial e_{LQ}(\boldsymbol{w})}{\partial w_K} \end{bmatrix} \tag{5}$$

is the Jacobian matrix, $\mu > 0$,$\boldsymbol{I}$ is an identity matrix of size $K \times K$.

The LM method weight recurrence is then [8]

$$\boldsymbol{w_{n+1}} = \boldsymbol{w_n} - [\boldsymbol{J}(\boldsymbol{w_n})^T\boldsymbol{J}(\boldsymbol{w_n}) + \mu_n\boldsymbol{I}]^{-1}\nabla\psi(\boldsymbol{w_n}) \tag{6}$$

So, each iteration requires the calculation of the inverse of $\boldsymbol{J}(\boldsymbol{w_n})^T\boldsymbol{J}(\boldsymbol{w_n}) + \mu_n\boldsymbol{I}$ which explains the only drawback of the LM method compared to the Newton method, namely the relative sloth if the starting parameters and the functions are "nice".

2.4 Radial Basis Function (RBF) Neural Network

The Radial Basis Function networks have a simple architecture comprising an input layer, a single (usually Gaussian) hidden layer and an output layer, the nodes of which are expressed as a linear combination of the outputs of the hidden layer nodes. For a one-node output layer, the global input-output relationship of an RBF Neural Network can be expressed as a linear combination of K basis function as follows:

$$f(\boldsymbol{x}) = \sum_{k=1}^{K} w_k\phi_k(\boldsymbol{x}) \tag{7}$$

where $\boldsymbol{x} = [x_1, x_2, \ldots, x_M]^T$ is the M-dimensional input vector, w_k are the weighting coefficients of the linear combination, and $\phi_k(\boldsymbol{x})$ represents the response of the k^{th} neuron of the hidden layer. Typically, the basis function $\phi_k(\boldsymbol{x})$ are assumed to be Gaussian shaped with scale factor σ_k; their values decrease monotonically with the distance between the input vector $\boldsymbol{x}$ and the center of each function $\boldsymbol{c_k} = [c_{k1}, c_{k2}, \ldots, c_{km}]^T$ [7].

$$\phi_k(\boldsymbol{x}) = exp(-\frac{\parallel \boldsymbol{x} - \boldsymbol{c_k} \parallel^2}{\sigma^2}) \tag{8}$$

2.5 General Regression Neural Network (GRNN)

Suggested by Specht D.F., the General Regression Neural Network given enough data can solve any smooth approximation. The method is simple but suffers from

high estimation cost. GRNN can be considered as a normalized RBF network. The GRNN consists of an input layer (where the inputs are applied), a hidden layer and a linear output layer. The hidden layer is usually expressed by a multivariate Gaussian function ϕ with an appropriate mean and an auto variance matrix as follows:

$$\phi_k[\boldsymbol{x}] = exp[-(\boldsymbol{x} - \boldsymbol{v_k^x})^T(\boldsymbol{x} - \boldsymbol{v_k^x})/(2\sigma^2)] \tag{9}$$

When $\boldsymbol{v_k^x}$ are the corresponding clusters for the inputs, $\boldsymbol{v_k^y}$ are the corresponding clusters for the outputs obtained by applying clustering technique of the input/output data that produces K cluster centers. $\boldsymbol{v_k^y}$ is defined as

$$\boldsymbol{v_k^y} = \sum_{y(p)\in \text{ cluster k}} y(p) \tag{10}$$

N_k is the number of input data in the cluster center k , and

$$d(\boldsymbol{x}, \boldsymbol{v_k^x}) = (\boldsymbol{x} - \boldsymbol{v_k^x})^T(\boldsymbol{x} - \boldsymbol{v_k^x}) \tag{11}$$

with

$$\boldsymbol{v_k^x} = \sum_{\boldsymbol{x(p)}\in \text{ cluster k}} \boldsymbol{x(p)} \tag{12}$$

The outputs of the hidden layer nodes are multiplied with appropriate interconnection weights to produce the output of the GRNN [6]. The weight for the hidden node k (i.e.,w_k) is equal to

$$w_k = \frac{\boldsymbol{v_k^y}}{\sum_{k=1}^{K} N_k exp[-\frac{d(\boldsymbol{x}, \boldsymbol{v_k^x})^2}{2\sigma^2}]} \tag{13}$$

3 Experimental Results

In this study, engine power and displacement are modeled by using different neural network architectures and training algorithms. Both of problems are function approximation. Many different parameters are used for each architecture and training algorithms. Function approximators's abbreviations and explanations are given at Table 1.

Training and test sets are composed of 92 and 22 samples respectively. The problems share same 4 inputs but differ from the outputs. The inputs are B, L, V and T for each problem. In the first problem the output is displacement while the engine power is the output of second problem. Samples have 4 inputs and 1 input. For each problem, all possible combinations of inputs are given to neural network models to be able to see the prediction abilities of inputs. Training data sets of four single inputs, two and three inputs each with six, and single four input pairs are obtained.

Each train/test set pairs (datasets) are used in 17 different networks varies architecture, algorithms and parameters. In other words 255 (15*17) training and testing process is done for each function approximation problem.

Table 1. Function approximators

Abbreviation	Architecture parameters	Architecture and Learning algorithms
LM-1	No hidden layer (Linear Model)	Multi Layer Perceptron
LM-10-1	Hidden layer with 10 neuron	trained by
LM-20-1	Hidden layer with 20 neuron	Levenberg-Marquardt
LM-100-1	Hidden layer with 100 neuron	optimization
GDA-1	No hidden layer (Linear Model)	Multi Layer Perceptron
GDA-10-1	Hidden layer with 10 neuron	trained by
GDA-20-1	Hidden layer with 20 neuron	gradient descent with
GDA-100-1	Hidden layer with 100 neuron	adaptive learning rate
GDM-1	No hidden layer (Linear Model)	Multi Layer Perceptron
GDM-10-1	Hidden layer with 10 neuron	trained by
GDM-20-1	Hidden layer with 20 neuron	gradient descent
GDM-100-1	Hidden layer with 100 neuron	with momentum
GRNN	Spread = 1.0	Generalized Regression Neural Network
RBF-10	Spread =10	
RBF-20	Spread =20	Radial Basis Function
RBF-50	Spread =50	Neural Network
RBF-100	Spread =100	

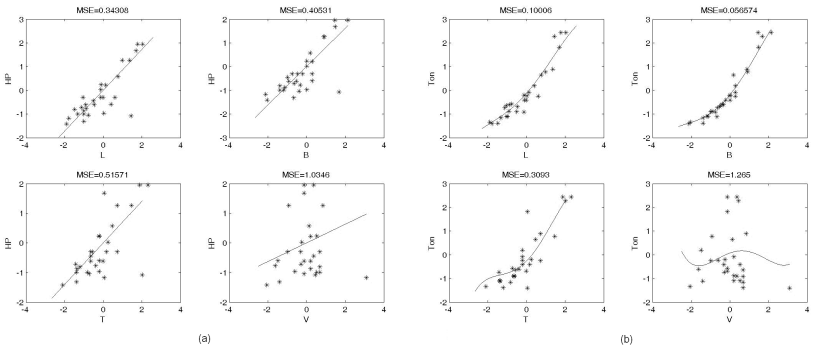

Fig. 2. The stars are test samples while the line is approximated function (a) engine power with GDA linear model (b) displacement with RBF network spread=50

In Figures, Mean Square Error's are given at the top of each experiment. In Figure 2.a, GDA linear models and their engine power approximations are given for each single input. The predicted function of displacement with RBF network in Figure 2.b. X and Y axis show the single input and the output of the functions respectively.

In Figure 3, GDM and RBF models and their engine power function approximations are given for the same dataset with two inputs.

In Table 2, 3, and 4, engine power and displacement prediction experiments are summarized. For each number of inputs, the best feature, the best regressor, the best experiment, the unreliable regressor(s), mean success order and obser-

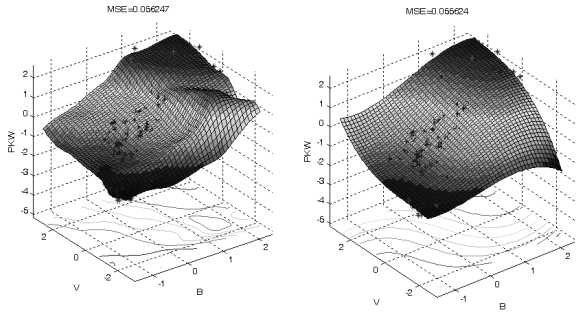

Fig. 3. GDM 20 neuron at the hidden layer, RBF 50. Blue and red stars are train and test sets respectively. The surface is approximated engine power function.

Table 2. Summary of Predictions

Engine power	The Best feature(s)	The average best regressor	The best experiment (Model : Feature(s) : MSE)
1 input	B	LM-1,GDA-1, GDM-1	GDM-10-1 : L : 0.2981
2 inputs	L-B	LM-1, GDM-1	GDM-20 : L-V : 0.255
3 inputs	L-B-V	LM-1, GDM-1	GDA-10 : L-T-V : 0.3068
All (4) inputs	-	GRNN	GDA-1 : All : 0.3075
Displacement			
1 input	B	RBF-50, RBF-100	RBF-10 : B : 0.0556
2 inputs	L-B	RBF-100	RBF-10 : L-B : 0.0497
3 inputs	L-B-V	GDA-1, GDM-1	GDA-10 : L-B-V : 0.0418
All (4) inputs	-	GDM-100	GDM-100 : All : 0.0575

Table 3. Summary of Predictions (continued)

Engine power	Unreliable regressor(s)	Success order
1 input	LM	GRNN>RBF>GDA>GDM>>LM
2 inputs	LM, RBF	GRNN>GDM>GDA>RBF>>LM
3 inputs	LM, GDA, RBF	GRNN>GDM>>LM>GDA>RBF
All (4) inputs	LM, RBF	GRNN>GDM>GDA>>RBF>LM
Displacement		
1 input	LM	RBF>GDM>GRNN>GDA>>LM
2 inputs	LM, RBF	GDM>GRNN>GDA>RBF>>LM
3 inputs	LM, GDA, RBF	GDM>GRNN>LM>GDA>RBF
All (4) inputs	GDA, RBF	GDM>GRNN>LM>>RBF>GDA

vations are given. The best features, the best regressors and success orders are obtained from average values of experiments having different number of inputs. For the explanations of regressor abbreviations see Table 1.

Table 4. Prediction Observations

Number of Inputs	**Engine Power Observations**
1 input	LM, GDA and GDM have similar characteristics. MSE increases with the number of neurons in the hidden layer. LM gives unreliable results when the number of neurons in the hidden layer is above 20. LM, GDA and GDM have very similar performance when linear model is used (no hidden layer). The effect of RBF's spread value over the performance is very few.
2 inputs	LM is very sensitive to the number of hidden neurons. LM and GDM have similar MSEs with linear model. The spread value of radial basis functions in RBF is effective in the performance.
3 inputs	LM and GDM's linear models have similar performances and they are the best regressors for 3 inputs. In RBF, the MSEs decrease while spread value increases.
All (4) inputs	GDA and GDM have similar characteristics. MSE increases with the number of neurons in the hidden layer. LM, GDA and GDM have very similar performance when linear model is used. In RBF, the MSEs decrease while spread value increases. RBF has not robust results when the spread value is 10 or 20.
Number of Inputs	**Displacement Observations**
1 input	The effect of RBF's spread value over the performance is very few. LM, GDA and GDM give similar MSEs when linear model is used.
2 inputs	LM and GDM give similar MSEs when linear model is used. The spread value of radial basis functions in RBF is effective in the performance.
3 inputs	LM and GDM give similar MSEs when linear model is used. In RBF, the MSEs decrease while spread value increases.
All (4) inputs	In GDM, MSEs decreases while the number of neurons in hidden layer increase. LM and GDM give similar MSEs when linear model is used. In RBF, the MSEs decrease while spread value increases. RBF has not robust results when the spread value is 10 or 20.

4 Conclusion

In naval engineering, in order to calculate the engine power and the water displacement, classical methods use several parameters [5] , some of which require tedious work and long experiments. This study aims to estimate the engine power and displacement by the help of only easily accessible parameters. As estimation models, several ANN techniques are used where the samples have 4 inputs and only one output. In this study, two analysis have been conducted to find:

1. The most effective input(s)to determine the output
2. The most successful ANN architecture and/or training algorithm

For the first one, using each possible input combinations 15 different training and test sets have been created. For the second analysis, 17 regressors comprising different architecture, algorithm and parameters in modeling have been used. Some important results have been summarized as follows:

- GDA and RBF give most accurate estimations in some situations but, they have also some unreliable results.
- GDM is the best choice for these two datasets because of high performance and always reliable results.
- When the number of hidden neurons increase LM's results get unreliable.
- GRNN is the best choice for engine power prediction.
- RBF and GRNN's approximated functions are smoother than GDA, GDM and LM's ones.
- When the number of features increased, unreliable results appears more.
- For both of datasets, prediction abilities of inputs are ordered as "B>L>V>T" when they are used alone.
- Engine power is better modeled with linear functions than the non-linear ones.
- Displacement is better modeled than engine power with neural networks. The minimum MSEs of displacement and engine power are 0.042 and 0.255 respectively.

References

1. Raya, T., Gokarna, R.P., Shaa, O.P. :Neural network applications in naval architecture and marine engineering. Artificial Intelligence in Engineering, 10(3), pp.213-226 (1996)
2. Alkan A.D., Gulez K., Yilmaz H. :Design of a robust neural network structure for determining initial stability particulars of fishing vessels, Ocean Engineering (31), pp.761-777 (2004)
3. Xiao J., Wong X., Boa M. :The modeling and simulation of ship main engine based on Neural Network, Shanghai Maritime Universty, Proceedings of the 4th World Congress on Intelligent Control and Automation, (2002)
4. Kapanoglu B., Çelebi U.B., Ekinci S., Yıldırım T. :Determination Of Approximate Main Engine Power For Chemical Cargo Ships Using Radial Basis Function Neural Network, Turksh Naval Academy, Journal of Naval Science and Engineering 2(2), pp:115-115 (2004)
5. Baykal R., Dikili A.C. :Ship Resistance and Engine Power, I.T.U, ISBN 975-561-196-7, Istanbul (in Turkish) (2002)
6. Popescu I., Kanatas A., Constantinou P., Nafornita I. :Applications of General Regression Neural Networks for Path Loss Prediction, Proc. of International Workshop "Trends and Recent Achievements in Information Technology", Cluj Napoca (2002)
7. Corsini G., Diani M., Grasso R., De Martino M., Mantero P., Serpico S.B. :Radial Basis Function and Multilayer Perceptron Neural Networks for Sea Water Optically Active Parameter Estimation in Case 2 Waters: A Comparison, International Journal Remote Sensing 24(20) (2003)
8. Chen Y., Wilamowski B.M. :TREAT: A Trust-Region-based Error-Aggregated Training Algorithm for Neural Network, Proc. IEEE International Joint Conference on Neural Networks-IJCNN 2002, Honolulu Hawaii (2002)