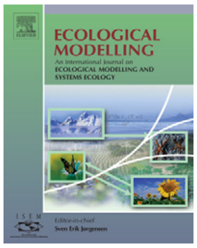
available at www.sciencedirect.com

journal homepage: www.elsevier.com/locate/ecolmodel



Neuroet: An easy-to-use artificial neural network for ecological and biological modeling

Peter A. Noble[*], *Erik H. Tribou*

Civil and Environmental Engineering, University of Washington, Seattle, WA 98195, United States

ARTICLE INFO

Article history:
Received 26 October 2004
Received in revised form 1 June 2005
Accepted 8 June 2005
Published on line 18 December 2006

Keywords:
Neural network
Ecological Modelling

ABSTRACT

Neuroet is an easy-to-use artificial neural network (NN) package designed to assist with determining relationships among variables in complex ecological and biological systems. The package, which is available for download from the web site http://noble.ce.washington.edu, features a procedure to optimize the architecture of NNs by adjusting the number of neurons in the hidden layer, and a novel procedure to identify the input variable, or combinations of input variables, that is/are important for predicting outputs. The package also includes a method to extract equations defining relationships among the data (independent of the NN package). The performance of Neuroet was assessed using benchmark standards for NNs. An example of the program's utility is provided using an environmental data set.

1. Introduction

Artificial neural networks (NNs) are useful tools for recognizing patterns in complex, nonlinear data such as those associated with ecological and biological data as demonstrated by the articles published in Ecological Modelling 120 (2–3) and 146 (1–3), and the Journal of Microbiological Methods 43 (1–2). They are particularly advantageous over conventional (linear-based) statistical methods because NNs can deal with the inherent variability associated with biological data, and therefore, provide better recognition of patterns in data and make better predictions of response variables (when provided with a set of input variables) than conventional methods.

NNs are constructed from computer programs and consist of networks of neurons that receive information from inputs or other neurons, make independent computations, and pass their outputs to other neurons in the network. Each neuron in a network is an independent processing element composed of weights (that are used to weigh the value of data received), a bias term (that prevents divisions by zero), and a transfer function (that passes the value of the neuron forward to the next neuron in the network). Supervised NNs are 'trained' with a set of known inputs and outputs and 'learn' relationships based on examples provided in training data. During training, they adjust the values of weights and biases by minimizing error between outputs and target values, and then adjusting the values of weights and biases using an error function. The adjusted weights and biases can then be used to recalculate the output (forward propagation). The iterative process of adjusting (and readjusting) the weights and biases is referred to as error back-propagation (Bishop, 1995; Rumelhart et al., 1986). In theory, error back-propagation (followed by forward propagation) continues until the global error minimum is attained. However, in practice, NNs often get 'stuck' in local error minima. A variety of training algorithms can be used to ensure that a NN reaches its global error minimum. Conjugate gradient and Levenberg-

[*] *Corresponding author at*: 201 More Hall, Civil and Environmental Engineering, University of Washington, Seattle, WA 98195, United States. Tel.: +1 206 685 7583; fax: +1 206 685 3836.
E-mail address: panoble@washington.edu (P.A. Noble).

0304-3800/$ – see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.ecolmodel.2005.06.013

Marquardt, for example, are sophisticated training algorithms that optimally adjust weights and biases so that the global error between outputs and targets is minimized after several iterations. Once a NN is properly trained, the adjusted weights can then be used to generate a model that potentially provides information on the functional relations among variables.

Despite the general acceptance of NNs as statistical tools for analyzing ecological data, they are still regarded as 'black boxes' because the contribution of input variables to predict output variables is often difficult to disentangle within a network (Olden and Jackson, 2002). Establishing the contributions of input variables to outputs (i.e. response variables) is needed to gain an understanding of the underlying relationships driving ecological and biological processes. Sensitivity analysis is frequently used to determine the contribution of individual input variables to predict outputs (e.g. Urakawa et al., 2002; Noble et al., 2000; Scardi and Harding, 1999). Briefly, this analysis involves (i) training a NN, (ii) extracting the equation defining the relationship between input and output variables, (iii) varying the value of each input variable (in the equation) while holding all other variables constant, and measuring the change in the value of the output. The sensitivity of an input variable to an output variable is expressed as the relative change (i.e. the average slope or range) because the change is relative to the other inputs used to train the NN. The drawback of sensitivity analysis is that correlated and/or noisy input variables often produce inconsistent results—even for NNs trained with the same data set. Variability among NNs is due to the value of weights and biases that are randomly generated at the start of each training run and modified during the course of training. An alternative and robust approach that provides insight into causal relationships among individual input variables (as well as among groups of input variables) and output variables, would be highly desirable, since it would facilitate the development of broad hypotheses focused on understanding nonlinear ecological and biological phenomena. Moreover, an approach that provides a 'simple' way to extract mathematical equations relating input variables to output variables would be also highly desirable because it would lead to the incorporation of the equations in other applications (e.g. MS Excel, C++ programs) – independent of the NN software – illuminating the 'black box'.

The objective of this study was to develop and demonstrate the utility of a neural network (NN) package designed to assist scientists in determining relationships among variables in complex ecological and biological data sets. Specifically, we (i) developed a procedure to automatically determine the optimal number of hidden neurons needed to train a NN, (ii) developed a novel approach to determine the predictive importance of combinations of input variables within a complex data set, and (iii) established a simple procedure for building equations of trained NNs. To this end, we report on the development and rigorous testing of a new NN package called Neuroet (Tribou and Noble, 2004). Example files of real environmental data that can be used to test Neuroet and to demonstrate the procedures outlined in this document are available for download at ftp://EcoMod:EcoMod@128.95.45.41.

2. Fundamentals of neural network computing: calculating the value of an output from a single neuron, calculating the error between output and target values, and adjusting the weights and bias term

The smallest unit of a NN is a single neuron. A neuron is typically composed of weights, a bias term, and a transfer function (Fig. 1). Neurons receive input data, multiply it by a weigh, and pass the information (forward) to other neurons in a network. For example, consider a single neuron that receives two inputs (input #1 has a value of 0.3 and input #2 has a value of 0.2). Each input has a corresponding weight. In our example the neuron will have two weights (wgt #1 is connected to input #1 and has a value of 0.7, wgt #2 is connected to input #2 and has a value of −0.8), a bias term (has a value of −0.5), and a transfer function (Log-Sigmoid) (Fig. 1). The equation defining the value of the output for this single neuron is:

$$\begin{aligned}\text{Output} &= \text{Log-Sigmoid}((\text{input\#1} \times \text{wgt\#1}) \\ &\quad + (\text{input\#2} \times \text{wgt\#2}) + \text{bias})\end{aligned}$$

Incorporating values of the inputs, weights, and the bias term into the equation below, we obtain an output value of 0.39:

$$\text{Output} = \frac{1}{(1 + e^{(-1\times(((0.3\times0.7)+(0.2\times-0.8))+(-0.5)))})}$$

If the actual value of a target is equal to 0.50, the error between the target value and the output would be equal to 0.11 (i.e., 0.5 − 0.39 = 0.11). When NNs are trained, the weights and bias terms of neurons are readjusted individually by an error function. We can simulate training by adjusting wgt #1 to 0.9 and wgt #2 to −0.2. The value of the output would then be equal to 0.43. By adjusting the weights, the prediction of the output has improved by a value of 0.07—the neuron has learned! This 'simple' example provides information on how to calculate the output of a single neuron. NNs consist of networks of neurons that are configured in different ways—depending on the complexity of the data. Calculating the outputs of multi-connected neurons in a network drastically increases the complexity of the equations.

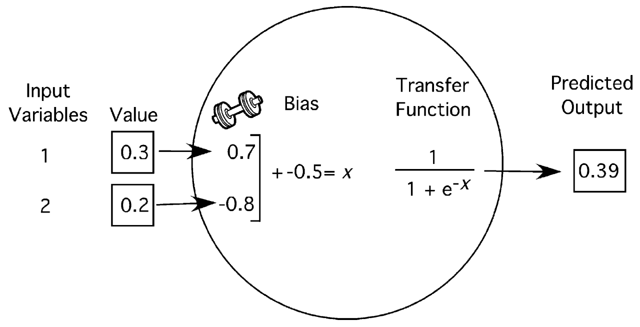


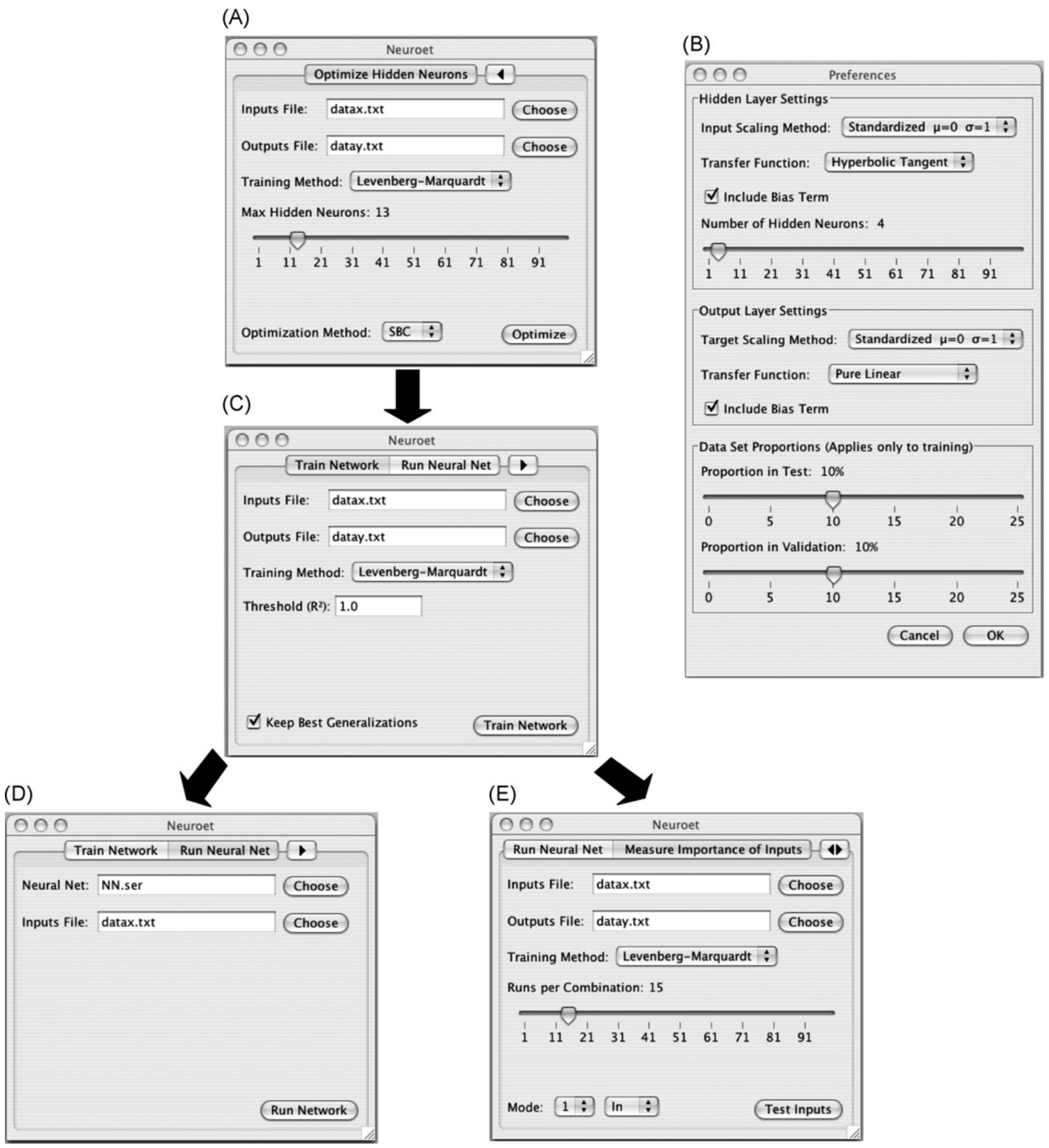
Fig. 1 – The relationship between input variables and the output of a single neuron (represented by the circle). Shown are the weights, bias, and transfer function.

NN packages automatically calculate the values of outputs in complex networks as well as train, test, and validate themselves.

More introductory information on the fundamentals of neural computing are available in the following articles: Basheer and Hajmeer (2000), Reed and Marks (1999), Principe et al. (2000), Hagan et al. (1996), and Bishop (1995). Section 3 provides information on how to: (i) optimize the architecture of NNs by adjusting the number of hidden neurons, (ii) train, test, and validate a NN, (iii) extract the equation defining the relationship between input variables and an output variable of a trained NN, and (iv) determine which inputs, or combination of inputs, are most important for predicting outputs. Section 4 outlines how we rigorously assessed the performance of Neuroet by comparing Neuroet results to those obtained by Sarle (1999).

3. The Neuroet package

Neuroet is an easy-to-use NN package that provides analytical tools to: (i) make predictions, (ii) determine equations between input and output variables, and (iii) determine the predictive importance of input variables. In cases (i) and (ii), it is necessary to optimize the number of hidden neurons before training; otherwise, the NN might not be able to find patterns in the data. Too many hidden neurons affect NN performance by over-fitting (i.e. memorizing) the data, resulting in poor predictions for test and validation data sets—too few hidden neurons affect NN performance by under-fitting the data. Neuroet also provides an automated procedure to determine the optimum number of hidden neurons. The user interface for this procedure is shown in Fig. 2A. Once the number of hidden



(A) Inputs File: datax.txt
Outputs File: datay.txt
Training Method: Levenberg–Marquardt
Max Hidden Neurons: 13
Optimization Method: SBC

(B) Input Scaling Method: Standardized μ=0 σ=1
Transfer Function: Hyperbolic Tangent
☑ Include Bias Term
Number of Hidden Neurons: 4
Target Scaling Method: Standardized μ=0 σ=1
Transfer Function: Pure Linear
☑ Include Bias Term
Proportion in Test: 10%
Proportion in Validation: 10%

(C) Inputs File: datax.txt
Outputs File: datay.txt
Training Method: Levenberg–Marquardt
Threshold (R^2): 1.0
☑ Keep Best Generalizations

(D) Neural Net: NN.ser
Inputs File: datax.txt

(E) Inputs File: datax.txt
Outputs File: datay.txt
Training Method: Levenberg–Marquardt
Runs per Combination: 15
Mode: 1 In

Fig. 2 – Overall structure of Neuroet. (A) Panel for optimizing the number of hidden neurons; (B) Panel for setting preferences (e.g. the number of hidden neurons and the proportion of the data used for testing and validation); (C) Panel for training the neural network; (D) Panel for testing the trained neural network; (E) Panel for measuring the importance of inputs.

neurons is determined, the user must set the number of hidden neurons in the Preference panel (Fig. 2B). The Preference panel sets the architecture and parameters for training and testing the NNs. The user interfaces for training and testing the NN are shown in Fig. 2C and D, respectively. Determining the predictive importance of input variables is accomplished using the interface shown in Fig. 2E.

3.1. *Optimizing the architecture*

The architecture of a NN is determined, in part, by the number of input and output variables. Neurons that receive input variables from a data set are referred to as hidden neurons, while neurons receiving outputs from hidden neurons are referred to as output neurons. Fig. 3 shows the orientation of the data files relative to Neuroet. Neuroet requires two data files: one file containing the input variables that will be used to predict output variables and the other containing the corresponding output variables. File *x* contains rows of tab-delimited columns of data with each column having a separate heading (e.g. Input#1) and each row comprising of a single record. While file *x* serves as the input file for Neuroet, file *y* serves as the file containing output data that the NN will use to learn the patterns. Note the file *y* also contains a heading for the column and that each record in file *x* has a corresponding record in file *y*. Given that the number of input variables is determined by the data set being trained, and the output is limited to one, the user must specify the number of hidden neurons to define the final architecture of a NN. It is essential that the number of hidden neurons be optimized in order to prevent under- or over-fitting of the data that often leads to incorrect predictions and/or poor results. The optimal number of hidden neurons depends on: (i) the number of input and output variables, (ii) the number of training records, (iii) the amount of noise in the output variables, (iv) the complexity of the relationship between input and output variables, and (v) the type of transfer functions.

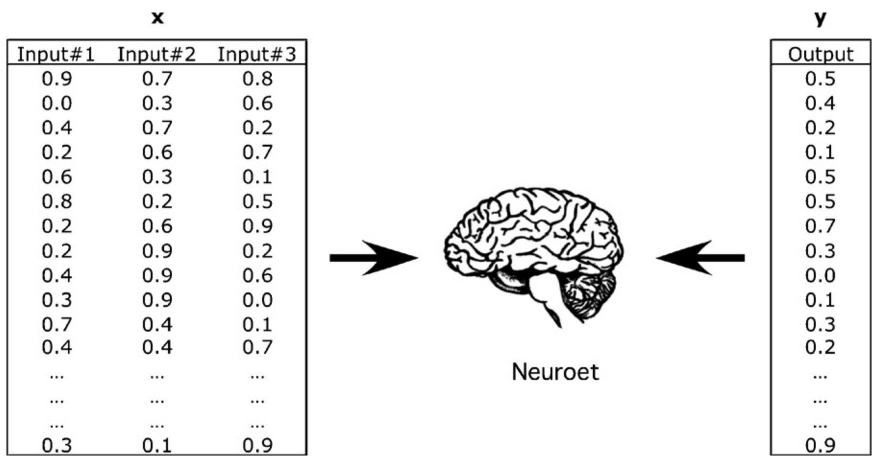
Since there are no explicit rules to estimate the optimal number of hidden neurons (Sarle, 1999), the optimal number of hidden neurons was determined by training sets of NNs containing 1 to *n* number of hidden neurons and calculating the median generalization estimator score for each set of *m* NNs (e.g. $m = 13$ in Fig. 2A). Two generalization estimators were used: Schwarz's Bayesian criterion (SBC), and corrected Akaike's Information Criterion (AICc). The following equations were used to calculate the SBC (Schwarz, 1978) and AICc scores (Hurvich and Tsai, 1989):

$$\mathrm{SBC} = (n)\ \log\left(\frac{\mathrm{SSE}}{n}\right) + (p)\ \log(n)$$

$$\mathrm{AICc} = (n)\ \log\left(\frac{\mathrm{SSE}}{n}\right) + \frac{n+p}{1-(p+2)/n}$$

where *n* represents the number of training records, and *p* represents the number of weights and biases. These estimators were calculated by determining the sum of squares errors (SSE) for each of the 13 NNs, discarding NNs that had SSE lower than the 25th percentile (rounded up), and calculating the median estimator from the remaining NNs. NNs with low SSE were removed for the analysis because we assumed that they did not reach the global error minimum. The optimum number of hidden neurons was determined by the set of NNs yielding the lowest median estimator score, indicating the NN models that best 'fit' the data.

Input and output data files used to demonstrate how to determine the number of hidden neurons are: data1x.txt and data1y.txt, respectively. These files represent daily water samples collected from Oyster Landing, South Carolina, USA and were extracted from the Belle W. Baruch Institute archives (http://links.baruch.sc.edu/Data/LTERDWSintroPage.htm). Details on variable names and interpretation of the biological significant relationships are not provided because they would distract from the objectives of the study. Several different options are available for determining the number of hidden neurons using Neuroet. Details on the available options for Neuroet and step-by-step instructions for new users are available at http://noble.ce.washington.edu/Neuroet. The following settings were used: scaling methods, standardize $\mu = 0$, $s = 1$; transfer function for input neurons, hyperbolic tangent, and transfer function for the output neuron, pure linear; training method, Levenberg-Marquardt; maximum number of hidden neurons was set to 13; Optimization method was set to SBC (e.g. Fig. 2A). Explicit details on the procedures to optimize the number of hidden neu-



Input#1	Input#2	Input#3
0.9	0.7	0.8
0.0	0.3	0.6
0.4	0.7	0.2
0.2	0.6	0.7
0.6	0.3	0.1
0.8	0.2	0.5
0.2	0.6	0.9
0.2	0.9	0.2
0.4	0.9	0.6
0.3	0.9	0.0
0.7	0.4	0.1
0.4	0.4	0.7
...	...	...
...	...	...
...	...	...
0.3	0.1	0.9

Output
0.5
0.4
0.2
0.1
0.5
0.5
0.7
0.3
0.0
0.1
0.3
0.2
...
...
...
0.9

Fig. 3 – Neuroet establishes the relationship of input data in file *x* to output in file *y*.

Table 1 – Rank order of the optimal number of hidden neurons for the demonstration data set based on SBC score

Number of hidden neurons	SBC score	AICc score
4	−12,002	−10,717
3	−12,000	−10,637
5	−11,964	−10,757
6	−11,942	−10,813
2	−11,941	−10,499
1	−11,873	−10,351
7	−11,812	−10,759
8	−11,803	−10,826
9	−11,715	−10,813
10	−11,641	−10,814
11	−11,517	−10,763
12	−11,488	−10,807
13	−11,406	−10,798

Table 2 – R-squared values of observed versus predicted outputs for training, testing, and validation data sets using the demonstration data set

Rank	R-square for file		
	Training	Testing	Validation
1	0.86	0.78	0.78
2	0.86	0.78	0.77
3	0.86	0.78	0.77
4	0.86	0.78	0.77
5	0.86	0.78	0.77
6	0.86	0.78	0.77
7	0.86	0.78	0.77
8	0.86	0.78	0.77
9	0.86	0.78	0.79
10	0.86	0.78	0.77
Last training	0.86	0.77	0.77

Rank order was based on the overall R-squared values.

rons using Neuroet are available in the documentation at http://noble.ce.washington.edu/Neuroet.

Table 1 shows the relationship between the number of hidden neurons and the median generalization estimator scores for the demonstration data set. The optimum number of hidden neurons for this data set was four since NNs trained with four hidden neurons yielded the lowest median generalization estimator score. Note that when this experiment was repeated (using the same data set), five hidden neurons were found to be the lowest median generalization estimator score. Subtle differences in the optimal number of hidden neurons that are estimated by Neuroet occur often and an approach to determine whether or not one score provides a better estimate of the number of hidden neurons required for training a NN than another score is shown in Section 4.

3.2. *Training, testing, and validation*

Once the optimal number of hidden neurons is known, it is possible to train, test, and validate the NN without over- or under-fitting the data (e.g. Fig. 2C). The same input and output files (e.g. data1x.txt and data1y.txt) were used to demonstrate how to train, test, and validate a NN. The settings for Neuroet for this exercise are the following: scaling methods, standardize $\mu = 0$, $s = 1$; transfer function for input neurons, hyperbolic tangent, and transfer function for the output neuron, pure linear; training method, Levenberg-Marquardt; the number of hidden neurons was set to four; the proportion of the data used for training the NN was 80% and 20% of the remaining data was used for testing and validation of the NN (e.g. Fig. 2B and C).

Table 2 shows the R-squared values (observed versus predicted output values) for iterations that occurred just before training was stopped. For this data set, training data (e.g. 0.86) provided better predictions for output variables than testing (e.g. 0.78) and validation (e.g. 0.77) data sets. Subtle differences in R-squared values are functions of the data randomly selected for training, testing, and validating the NN—they change every time a NN is trained. The 'Last Training' results indicate the R-squared values of the training, testing, and validation files when training was stopped.

Fig. 4 shows the relationship between predicted and actual values using data from the 'Last Training' results. Note that the NN explained approximately 85% of the variability in the data.

3.3. *Measuring the predictive importance of input variables*

A novel analytical approach was developed for identifying a variable, or combination of variables (up to five variables), that significantly contribute to predicting outputs. The approach is based on training multiple NNs (15×) using n number of input variables ($n = 1, 2, 3, \ldots n$), two hidden neurons, and one output neuron, and determining which input variable, or combination of input variables, yields the lowest median generalization estimator score (Fig. 5; see Fig. 2E for user interface). Input variables having low scores contributed most to predicting outputs.

Table 3 is a summary of mean R-squared values (i.e. predict versus actual output values) and median generalized estima-

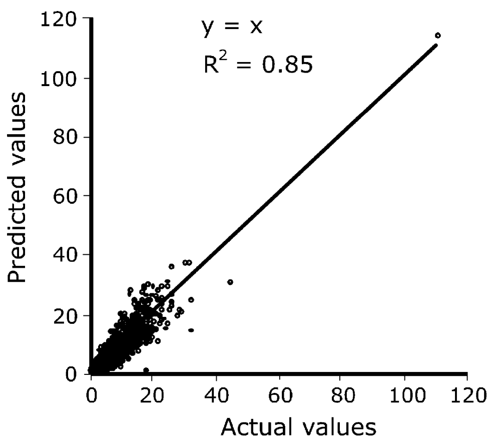


Fig. 4 – Predicted vs. actual output values of a trained NN. Removal of the extreme outliner (value ∼114) and retraining the NN had no significant effect on the equation or the R-squared value.

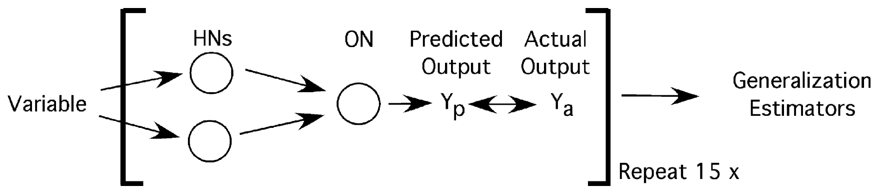


Fig. 5 – Scheme for identifying variable, or combination of variables, that are important for predicting outputs. Fifteen NNs were trained. Only two hidden neurons are used. The variables yielding the lowest median generalization estimator scores were important for predicting the output. HNs, hidden neurons; ON, output neuron; Y_p, predicted output, Y_a, actual output.

tor scores for each input variable and is based on the same data used previously (e.g. data1x.txt and data1y.txt). Note that rank order of the two median generalized estimator scores are similar, indicating that one score does not appear to be superior to the other. Variable #1 had the lowest median generalization estimator score, accounting for approximately 70% of the variability for predicting the output variable. In this data set, there was a clear difference in the predictive importance of input variables with variable#1 and #12 contributing more to predicting output values than variables # 6, #10, and #11 (which each accounted for less than 10% of the variability for predicting the output).

One of the difficulties of comparing input variables in Table 3 is determining whether or not one variable provided a significantly greater contribution to predicting outputs than another. For example, in Table 3, Variable #12 appears to be only marginally more important than Variable #13. Is this difference statistically significant? Is there a threshold score that distinguishes important variables from those that are not?

To address these questions, two approaches were used: (i) variables comprised of entirely random data were added to input data and the modified data set was then re-analyzed, and (ii) the probability that one variable, or combinations of variables, (i.e. rank 1) was more likely correct than another (i.e. rank 2) was calculated using an information theory approach (Motulsky and Christopoulos, 2002). We anticipated that variables containing random data would not contribute to predicting outputs and therefore would have a high median generalization estimator score. We also anticipated the probability that one variable was ranked higher than another would vary subtly between differently trained NNs.

The probability (P) that one NN model was more likely correct than another was determined using the following equation:

$$P = \frac{e^{-0.5\ \Delta AIC_c}}{1 + e^{-0.5\ \Delta AIC_c}}$$

where

$$\Delta AIC_c = AIC_{c2} - AIC_{c1}$$

The evidence ratio (E) provides information on how many times one model was more important than another. The evidence ratio (E) can be calculated using the following equation:

$$E = \frac{P_{model1}}{P_{model2}} = \frac{1}{e^{-0.5\ \Delta AIC_c}}$$

The same equations were used for determining P and E values for SBC.

Comparisons of different variables by P and E scores obtained using the AICc score (Table 4) indicated that 8 of 13 variables were predictively more important than those of lower rank. For example, Variable #1 was 9×10^{122} times more likely to be a much better model than Variable #12. Similarly, Variable #12 was a better model than Variable #13, and so on. Variable #4 was only marginally better than Variable #2, indicating that all variables ranked below Variable 3 were not predictively important as independent variables. Two variables consisting of random numbers were ranked 14 and 15

Table 3 – Predictive importance of variables ranked by their SBC scores

Rank	Inputs included	R-squared	SBC score	AICc score
1	Var#1	0.70	−1817	−279
2	Var#12	0.57	−1251	287
3	Var#13	0.52	−1069	469
4	Var#7	0.48	−951	587
5	Var#8	0.47	−921	617
6	Var#9	0.44	−851	688
7	Var#5	0.28	−424	1115
8	Var#3	0.17	−214	1324
9	Var#4	0.12	−115	1423
10	Var#2	0.12	−114	1425
11	Var#11	0.08	−31	1507
12	Var#6	0.07	−19	1519
13	Var#10	0.05	14	1552

Table 4 – Predictive importance of variables ranked by their AICc score

Rank	Inputs included	AICc score	Probability	Evidence ratio
1	Var#1	−279	0.000	8.84E+122
2	Var#12	287	0.000	3.32E+39
3	Var#13	469	0.000	3.87E+25
4	Var#7	587	0.000	4.11E+06
5	Var#8	617	0.000	1.83E+15
6	Var#9	688	0.000	4.83E+92
7	Var#5	1115	0.000	3.76E+45
8	Var#3	1324	0.000	3.12E+21
9	Var#4	1423	0.340	1.94E+00
10	Var#2	1425		
11	Var#11	1507		
12	Var#6	1519		
13	Var#10	1552		

Table 5 – Predictive importance of pairs of variables ranked by their SBC score

Rank	Inputs included	*R*-squared	SBC score	AICc score
1	Var#1 and Var#12	0.81	−2623	−1064
2	Var#1 and Var#13	0.72	−1999	−439
3	Var#1 and Var#11	0.71	−1927	−367
4	Var#12 and Var#13	0.71	−1920	−361
5	Var#1 and Var#5	0.70	−1862	−303
6	Var#1 and Var#2	0.70	−1858	−299
7	Var#1 and Var#8	0.70	−1845	−285
8	Var#1 and Var#10	0.70	−1845	−285
9	Var#1 and Var#6	0.70	−1844	−284
10	Var#1 and Var#3	0.69	−1834	−275
11	Var#1 and Var#9	0.69	−1833	−273
12	Var#1 and Var#4	0.69	−1831	−271
13	Var#1 and Random_1	0.69	−1824	−264
14	Var#1 and Random_2	0.69	−1821	−261

and are not shown in Table 4. These findings indicate (i) that variables #1 through #13 were predictively more important than variables containing random numbers, and (ii) that incorporation of variables containing random numbers into the input data for NN analysis validated our approach by distinguishing predictively important variables from those that are not, and (iii) that Variables #1, #3, #5, #7, #8, #9, #12, and #13 were important for predicting the output based their *P* and *E* scores.

The predictive importance of pairs of variables is shown in Table 5. Variables #1 and #12 were predictively more important than the other pairs of variables, as anticipated from the results in Tables 3 and 4. Variables containing random numbers (i.e. Ranks 13 and 14) set the lower limit of the possible 105 pairs considered predictively important. The remaining 91 pairs were not shown in Table 5.

We also investigated the predictive importance of combinations of three variables. Variables #1, #12, and #13 had the lowest median generalization estimator score followed by variables #1, #11, and #12 (data not shown). Rank 1 had $P = 0.00$ and $E = 5.5 \times 10^5$ over Rank 2, indicating that the NN models composed of Variables #1, #12, and #13 were significantly more likely to be correct than those based on Variables #1, #11, and #12. Since Rank 2 had a $P = 0.45$ over other ranks, we concluded that Rank 1 provided the best combination of variables to predict the output. We retrained the NN using these variables as inputs to determine the equation describing the relationship between Variable #1, #12, and #13, and the output variable. In the next section, we will extract the equation representing the relationship between these input variables and the output variable.

3.4. *Extracting equations from trained NNs*

Now that we have established that Variables #1, #12, and #13 are important variables for predicting the output, we will demonstrate how to extract the equation describing the relationship between these three variables and the output. Our tasks will be to determine the optimal number of hidden neurons for the three input variables, to train the NN with the optimal number of hidden neurons, to extract the weight and biases, and to built the equation in a spreadsheet (i.e. MS Excel).

Input and output data files used to train, test, and validate the NN were: data2x.txt and data2y.txt, respectively. The optimal number of hidden neurons was determined to be 2. Neuroet settings used for training were the following: scaling methods, standard linear [0,1]; transfer function for input neurons, Log-Sigmoid, and transfer function for the output neuron, pure linear; training method, Levenberg-Marquardt; the number of hidden neurons was set to 2; The proportion of the data used for training the NN was 80% and 20% of the remaining data was used for testing and validation of the NN.

The prediction of one record (i.e. one sampling event) is a function of the three inputs, two hidden neuron, and the output neuron (Fig. 6). Each input and output value is scaled to the maximum and minimum values of the variable, defined by the equation:

$$\mathrm{SI}_{j,i} = \frac{I_{j,i} - \min(I_j)}{\max(I_j) - \min(I_j)}$$

where $\mathrm{SI}_{j,i}$ is the scaled value of input *I* of record *i* and variable *j*, and $\max(I_j)$ is the maximum value of variable *j*, and $\min(I_j)$ is the minimum value of variable *j*.

The value of each hidden neuron (HN) is defined by the equation:

$$\mathrm{HN}_k = \frac{1}{1 + \mathrm{e}^{-(\sum_j \mathrm{SI}_j w_j + b_k)}}$$

where HN_k is the value of *k* hidden neuron, *w* is the weight of variable *j* and *b* is the bias term of *k*th hidden neuron.

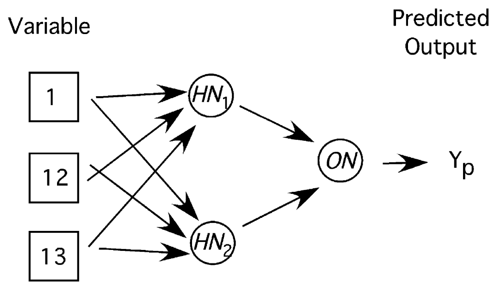


Fig. 6 – Scheme showing the relationship between variables, hidden neurons and predicted outputs. HN, hidden neurons; ON, output neuron; Y_p predicted output.

The value of the output neuron (ON) is defined by the equation:

$$\mathrm{ON} = a\left(\sum_k \mathrm{HN}_k w_k + b_{\mathrm{ON}}\right)$$

where ON is the value of output neuron, a the transfer function which in this case = 1, w the weight of kth HN and b is the bias term of ON.

The prediction (Y) is defined by:

$$Y = \mathrm{ON}(\max(y) - \min(y)) + \min(y)$$

where max(y) is the maximum value of output variable y, and min(y) is the minimum value of output variable y. The value of ON was rescaled because the output variable y was scaled prior to training the NN.

A downloadable spreadsheet version of this equation is called NNexample.xls. Fig. 7 shows a comparison between actual and predicted output values. Variable#1, #12, and #13 accounted for 83% of the variability in the data while the remaining 11 variables in the original data set (data1x.txt) accounted for less than 3% of the total variability (e.g. Fig. 4).

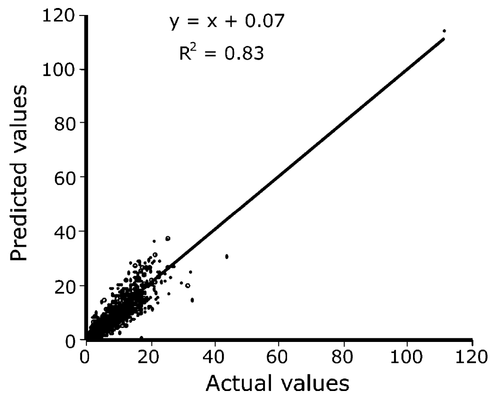


Fig. 7 – Predicted vs. actual values of a trained NN. The predicted values were derived from weights and biases extracted from a trained NN. Variables #1, #12 and #13 were used as input variables. Compare these results to those obtained using 13 input variables and the same output variable (Fig. 4). Variables #1, #12 and #13 accounted for approximately 83% of the variability in the trained NN. The remaining 11 variables of the original data set accounted for less than 3% of the total variability. Removal of the extreme outliner (values ~114) and retraining the NN had no significant effect on the equation or the *R*-squared value.

4. Benchmark testing of Neuroet

To thoroughly assess the performance of Neuroet, we compared results obtained by Neuroet to those obtained by Sarle (1999) using the Donoho-Johnstone (Dojo) neural network benchmarks. The Dojo functions have one input (x), and one output (y), and are noiseless and highly nonlinear (Fig. 8) and represent four functions that have very different patterns.

Sarle (1999) investigated the effects of noise on the number of hidden neurons needed to model four Dojo functions using a NN (Fig. 8). Three amounts of noise were added to the Dojo functions (low, medium, and high; Table 6). The four Dojo functions and four levels of noise (control, low, medium, and high) served as benchmark data for testing NNs. The Dojo files are available for download from the Sarle (1999) web site.

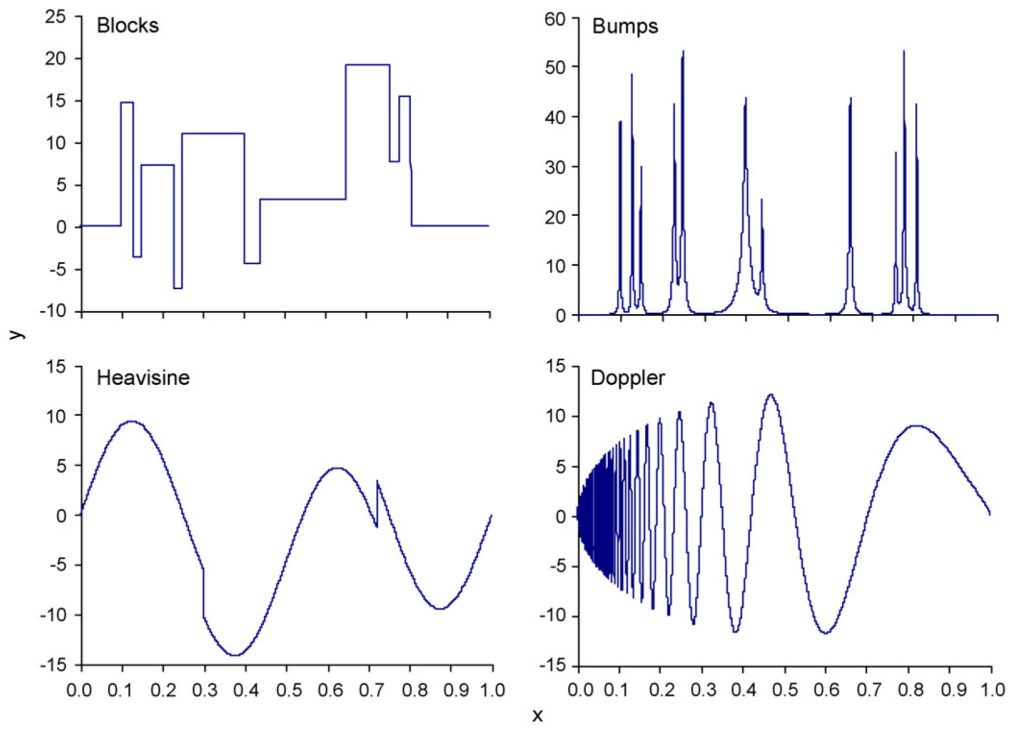


Fig. 8 – Four Donoho-Johnston benchmarks used to determine the optimal number of hidden neurons. The functions are shown in the form of *f*(*y*) = *x*. Noise has not been added to the benchmarks.

Table 6 – Noise level add to different data sets (adapted from Sarle, 1999)

Noise level	Standard deviation	Signal to noise ratio
None	0	–
Low	1	7.00
Medium	$\sqrt{5}$	3.13
High	5	1.40

Table 6 shows the standard deviation and signal to noise ratios added to each of the four functions by Sarle (1999).

Sarle (1999) trained NNs to predict output *y*, given input *x*. The NNs were trained under the following conditions: Levenberg-Marquardt training methods, hyperbolic tangent transfer function was used for hidden neurons, and pure linear transfer function was used for the output neuron, training was terminated when the average error gradient was $\leq 1 \times 10^{-8}$, or when for 10 consecutive iterations, the relative improvement in the error function was $\leq 1 \times 10^{-12}$. The criterion used to assess NN performance was based on generalization estimator scores (e.g. SBC, AICc). NNs that yielded the lowest median generalization estimator scores were considered to have the optimal number of hidden neurons.

To assess the performance of Neuroet, we followed Sarle's protocols exactly with the following exceptions: (i) extensive preliminary runs from multiple random starts were not conducted. Presumably, the reason Sarle did this was to discard algorithms that were 'stuck' in local error optima. Neuroet calculates the sum of squares (SSE) for each NN and those having SSE that falls into the lower 25th percentile (rounded up) were discarded. The median *R*-squared value, and the SBC and AICc scores were calculated from the remaining NN models. NNs with the lowest SBC and AICc scores were considered to have the optimal number of hidden neurons for predicting the output. Our complete analysis was repeated three times. We assumed that it was not necessary to conduct the extensive preliminary runs from multiple random starts because Neuroet automatically does this; (ii) we limited our analyses to a maximum of 35 hidden neurons. Sarle used 100 hidden neurons for his upper limit. Since the maximum number of optimal hidden neurons found by Sarle was 34, we set the upper limit of hidden neurons to be examined to 35, conserving valuable computation time and resources; (iii) we trained 15 independent NNs to determine the generalization scores while Sarle used 50 selected NNs. As mentioned above (i), we used an alternative approach; (iv) we standardized input and output data to have an average of zero and a standard deviation of one, prior to training the NNs. Preliminary studies revealed that scaling the four Dojo functions affected median generalization estimator scores (despite the perception in the literature that doing so should have no affect on the results). We experimentally determined that standardizing the data yielded consistent results, mirroring those obtained by Sarle (1999).

Once the optimal number of hidden neurons was determined, *R*-squared values between predicted and actual outputs were calculated for training, testing, and validation data sets. For all analyses, 80% of the data was used for training the NN, and 20% for NN testing and validation. To ensure that the results were consistent, we repeated the analyses ten times and used the average and standard deviation of *R*-squared values as an indicator for assessing NN performance (e.g. training $\times$ testing $\times$ validation $\times$ 10 runs = 30 runs). In theory, NNs having an optimal number of hidden neurons should yield consistent results. Comparison of the mean *R*-squared value (and standard deviation) of ten independently trained NNs should provide information on how well NNs recognized patterns in data and whether or not pattern recognition ability is consistent among independently trained NNs.

4.1. *Results of blocks, bumps, heavisine and Doppler analyses*

4.1.1. *Blocks*

Neuroet consistently yielded results that were close to those obtained by Sarle (1999). Sarle (1999) estimated that the number of optimal hidden neurons to be 11–21, while Neuroet estimated the number of hidden neurons to be 11–14 (Table 7).

Twelve hidden neurons appeared to be optimal for all Blocks data sets, regardless of the amount of noise in the data sets. Mean *R*-squared values of predicted and actual output values varied by noise level (Table 8), with the control (no noise) yielding a value of 0.96 ± 0.05, and the high noise Blocks data yielding a value of 0.62 ± 0.03. Comparisons of the *R*-squared values for training, testing, and validation data sets revealed similar *R*-squared values (data not shown), indicating that the NNs were not memorizing the data.

Sarle (1999) found that NNs having 11 hidden neurons was optimal for all noise Blocks data. Using the low noise Blocks data, we found that the mean *R*-squared value (and standard deviation) of 10 independently trained NNs having 11 hidden neurons to be lower and more variable (0.91 ± 0.05), than NN having 13 hidden neurons (0.95 ± 0.03; Table 8). In a repeated experiment, Sarle (1999) found that 21 hidden neurons to be optimum for training NNs using the low noise Blocks data. Results from a one-tailed *t*-test revealed that 13 hidden neurons provided higher *R*-squared values than those using 21 hidden neurons ($P < 0.004$; $n = 30$) (Table 8). There were no significant differences in the *R*-squared values for the medium nor high Blocks data sets, indicating that number of hidden neurons predicted by Neuroet and Sarle (1999) yielded similar results.

Hence, we conclude that for the Blocks data sets, Neuroet yielded results that are similar to those obtained by Sarle (1999). Increasing noise had no apparent effect on the estimated number of hidden neurons, though *R*-squared values decreased substantially—but not standard deviations.

4.1.2. *Bumps*

Neuroet determined the optimal number of hidden neurons to be slightly lower than those determined by Sarle (1999) (Table 7). Sarle (1999) determined that the number of optimal hidden neurons to be 22–34, while Neuroet determined the number of hidden neurons to be 22–29. Mean *R*-squared values varied by noise level (Table 8), with the control (no noise) data set yielding the highest value of 0.85 ± 0.07, and high noise data set yielding a value of 0.51 ± 0.08. With exception to low Bumps, overall comparisons of mean *R*-squared value (and standard deviations) of 10 independently trained NNs using Sarle (1999) values were not substantially different

Table 7 – Comparison of the calculated number of hidden neurons by study

Data set	Noise level	Criterion	Calculated number of hidden neurons by study	
			Neuroet[a]	Sarle (1999)[b]
Blocks	None	SBC	12, 12, 13	–
		AICc	12, 12, [13,14][c]	–
	Low	SBC	12, 12, 13	11, 11
		AICc	12, 12, 13	11, 21
	Medium	SBC	12, 12, 12	11, 13
		AICc	12, [12,13, 14], 12	11, 21
	High	SBC	11, 12, 12	11, 11
		AICc	11, [12,13, 14], 12	12, 16
Bumps	None	SBC	23, 23, 24	–
		AICc	23, 23, 24	–
	Low	SBC	22, 26, 28	32, 33
		AICc	26, 27,28	33, 33
	Medium	SBC	24, [24,25], 28	31, 33
		AICc	[25,26, 28], 24, 28	31, 34
	High	SBC	23, 24, 24	22, 26
		AICc	23, 24, 29	33, 33
Heavisine	None	SBC	6, 11, 12	–
		AICc	6, 11, 12	–
	Low	SBC	7, [10,12], 10	6, 8
		AICc	7, 10, 12	6, 9
	Medium	SBC	7, [5,8], 9	7, 7
		AICc	7, 8, [8,9, 10, 11]	7, 13
	High	SBC	3, 3, 5	4, 4
		AICc	[6,7], [6,7, 8], 9	8, 14
Doppler	None	SBC	15, 15, 16	–
		AICc	15, 16, 19	–
	Low	SBC	17, 17, 18	15, 30
		AICc	17, [17,20], 25	19, 31
	Medium	SBC	14, 17 21	12, 12
		AICc	[14,15], 17, 21	12, 31
	High	SBC	12, 14, [9,16]	8, 12
		AICc	14, 16, 22	9, 27

[a] Based on three experiments.
[b] Based on two experiments.
[c] Probability based on information theory indicated no difference.

that those obtained from Neuroet (Table 8). One-tailed t-test results revealed that Neuroet yielded significantly higher R-squared values using 28 hidden neurons than Sarle using 33 ($P < 0.001$; $n = 30$). Hence, we conclude that with exception to the low Bumps data where Neuroet results were higher R-squared values than those obtained by Sarle, similar results were obtained by Neuroet and Sarle (1999).

4.1.3. *Heavisine*

Sarle (1999) determined that the number of optimal hidden neurons to range from 6 to 14 while Neuroet determined the number of hidden neurons to range from 3 to 12 (Table 7). For the low noise Heavisine data set, R-square values of NN containing the optimal numbers of hidden neurons determined by Sarle (1999) and Neuroet were within the same range, indicating no differences. For the medium noise Heavisine data set, Sarle (1999) found that 7–13 hidden neurons to be optimum. R-squared values obtained using 7 and 13 hidden neurons (Table 8) were not significantly different, indicating that more neurons did not improve predictability. However, one-way Student t-test revealed that 9 hidden neurons (predicted by Neuroet) provided significantly higher R-squared values than using 7 or 13 hidden neurons ($P < 0.001$). For the high Heavisine data, there were no significant differences in the results obtained by Sarle or Neuroet. With exception to medium Heavisine data where Neuroet provided better pre-

Table 8 – Comparison of the *R*-squared values using the estimated number of hidden neurons by study

Data set	Noise	This study				Sarle's study (1999)			
		Number of hidden neurons	*R*-squared values[a] (X ± S.D.)	Number of hidden neurons	*R*-squared values[a] (X ± S.D.)	Number of hidden neurons	*R*-squared values (X ± S.D.)	Number of hidden neurons	*R*-squared values (X ± S.D.)
Blocks	None	12	0.96 ± 0.05	–[b]		–	–	–	
	Low	12	0.92 ± 0.05	13	0.95 ± 0.03	21	0.92 ± 0.05	11	0.91 ± 0.05
	Medium	12	0.83 ± 0.05	14	0.86 ± 0.03	11	0.87 ± 0.08	21	0.86 ± 0.03
	High	12	0.62 ± 0.03	13	0.62 ± 0.02	16	0.62 ± 0.03	–	–
Bumps	None	23	0.83 ± 0.12	–	–			–	–
	Low	27	0.82 ± 0.12	28	0.85 ± 0.07	32	0.76 ± 0.13	33	0.78 ± 0.14
	Medium	28	0.75 ± 0.10	26	0.79 ± 0.05	31	0.68 ± 0.08	33	0.80 ± 0.06
	High	24	0.50 ± 0.08	23	0.51 ± 0.08	33	0.45 ± 0.04	26	0.46 ± 0.12
Heavisine	None	6	0.99 ± 0.00	–	–			–	–
	Low	7	0.98 ± 0.01	12	0.98 ± 0.00	6	0.98 ± 0.00	9	0.98 ± 0.00
	Medium	11	0.91 ± 0.01	9	0.91 ± 0.01	7	0.90 ± 0.01	13	0.90 ± 0.02
	High	3	0.62 ± 0.04	6	0.67 ± 0.02	9	0.65 ± 0.01	14	0.67 ± 0.03
Doppler	None	15	0.92 ± 0.02	–	–			–	–
	Low	17	0.93 ± 0.01	25	0.92 ± 0.02	15	0.93 ± 0.01	30	0.93 ± 0.01
	Medium	17	0.86 ± 0.02	21	0.85 ± 0.01	12	0.84 ± 0.14	31	0.83 ± 0.03
	High	16	0.58 ± 0.03	22	0.60 ± 0.01	9	0.57 ± 0.04	21	0.58 ± 0.04

Two optimal number of hidden neurons (from Table 7) that yielded the highest *R*-squared values are shown for each study.

[a] Means were calculated from *R*-squared values of predicted and actual outputs obtained from training, testing and validation data sets of 10 independent NNs.

[b] Not determined.

dictions, there were no significant differences in the number of optimal hidden neurons selected by Sarle (1999) and Neuroet.

4.1.4. *Doppler*

There were subtle differences in the number of optimal hidden neurons determined by Neuroet and Sarle (1999). Sarle (1999) estimated that the number of optimal hidden neurons to range from 8 to 31, while Neuroet estimated the number of hidden neurons to range from 12 to 25 (Table 7). There were no significant differences in R-square values for low or high Doppler data sets (Table 8). However, a one-way Student t-test indicated that Neuroet provided higher R-square values than those obtained using the optimum number of hidden neurons provided by Sarle (1999) ($P < 0.001$).

5. Summary

Experiments aimed at determining the importance of inputs to predict outputs using an example data set revealed that three of 13 possible variables were important for predicting the output variable. The weights and biases extracted from a NN trained to relate three input variables to an output variable was used to build an equation relating input variables to outputs. The R-squared values of predicted versus actual outputs revealed that the equation accounted for most of the variability between input and output variables. Benchmark testing of Neuroet revealed that the procedure to automatically determine the optimal number of hidden neurons yielded accurate results. Moreover, predictions of optimized NNs were similar to (and sometimes better than) those obtained by Sarle (1999).

Acknowledgements

We thank Alex Pozhitkov, Heidi Gough, Travis Krick, and Seana Davidson for their insightful comments during the preparation of the manuscript. Funds were generously provided by the National Science Foundation (DEB 0196203), the Environmental Protection Agency (CEER-GOM- R-82945801), and the National Institute for Health (U01 DE 14955-02) to PAN.

REFERENCES

Basheer, I.A., Hajmeer, M., 2000. Artficial neural networks: fundamentals, computing, design, and application. J. Microbiol. Meth. 43, 3–31.

Bishop, C.M., 1995. Neural Networks for Pattern Recognition. Oxford University Press, London, UK.

Hagan, M.T., Demuth, H.B., Beale, M., 1996. Neural Network Design. PWS Publishing Company, Boston, MA.

Hurvich, C.M., Tsai, C.L., 1989. Regression and time series model selection in small samples. Biometrika 76, 297–307.

Motulsky, H., Christopoulos, A., 2002. Fitting Models to Biological Data using Linear and Nonlinear Regression: A Practical Guide to Curve Fitting. GraphPad Software Inc.

Noble, P.A., Almeida, J.S., Lovell, C.R., 2000. Application of neural computing methods for interpreting phospholipid fatty acid profiles from natural microbial communities. Appl. Environ. Microbiol. 66, 694–699.

Olden, J.D., Jackson, D.A., 2002. Illuminating the "black box"; a randomization approach for understanding variable contributions in artificial neural networks. Ecol. Model. 154, 135–150.

Principe, J.C., Euliano, N.R., Lefebvre, W.C., 2000. Neural and Adaptive Systems. John Wiley and Sons Inc., New York, NY.

Reed, R.D., Marks II, R.J., 1999. Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks. Massachusetts Institute of Technology, MA.

Rumelhart, D.E., Hutton, G.E., Williams, R.J., 1986. Learning representations by back-propagation errors. Nature 323, 533–536.

Sarle, 1999. Donoho-Johnstone Benchmarks: Neural Net Results. ftp://ftp.sas.com/pub/neural/dojo/dojo.html.

Scardi, M., Harding, L.W., 1999. Developing an empirical model of phytoplankton primary production: a neural network case study. Ecol. Model. 120, 213–223.

Schwarz, G., 1978. Estimating the dimension of a model. Ann. Stat. 6, 461–464.

Tribou, E., Noble, P.A., 2004. Neuroet: A Simple Artificial Neural Network for Scientists. University of Washington, Seattle, WA, http://noble.ce.washington.edu/Neuronet.htm.

Urakawa, H., Noble, P.A., ElFantroussi, S., Kelly, J.J., Stahl, D.A., 2002. Single-base pair discrimination of terminal mismatches by using oligonucleotide microarrays and neural network analyses. Appl. Environ. Microbiol. 68, 235–244.