SOME EXPERIENCES WITH NEURAL NETWORKS

Lynda L. Ballou

Dallas E. Johnson

Some Experiences with Neural Networks

Lynda L. Ballou and Dallas E. Johnson
Department of Statistics
Kansas State University
Manhattan, Kansas

Abstract

This paper gives a brief overview of artificial neural networks which may be used to model data similar to the kind where one usually considers regression models. Many practitioners believe that neural networks perform better than regression models for prediction purposes. Some simulations were performed using three different neural net programs, namely Braincel, Ripley's S+ program, and Nychka's S+ program. These simulations reveal some interesting aspects of neural net programs which should be of interest to anyone considering the use of neural net programs to model continuous data.

1. A Brief Introduction to Neural Networks

An artificial neural network, usually referred to as a neural network, has no widely recognized definition. A descriptive definition is: "Neural networks consist of many simple neurons or processors (real or simulated) that have densely parallel interconnections. The processors communicate across connections in terms of "activations" and "inhibitions" — signals that excite or inhibit responses by connected processors — rather than with symbols or messages that have high-level meanings." (Kinoshita and Palevsky, 1987). In the remainder of this section a brief introduction to neural networks is described. For a more detailed description see Bishop (1995).

Figure 1 describes a simplistic neural network, this neural network could be referred to as a neural network with no hidden layer. The inputs or the independent variables are directly connected to the output or predicted value of the dependent variable, the connections are via weights that a neural net program will develop. The assumptions for a neural network are similar to the assumptions required when modeling data with a linear regression model; that is, that a dependent variable is a function of the independent variables, $x_1, \ldots, x_k$, and random error. Many neural networks, but not all, assume that the random errors are independently and identically distributed with mean zero and a common variance. The input variables can be either quantitative or qualitative and are assumed to be uncorrelated. It is assumed that all important relationships between the independent variables have been recognized and dealt with before fitting the data with a neural network program. A variable x_o that is always equal to one is always included and is used as an intercept that corrects for bias in the fitted model. In neural network terminology, it is customary to refer to the observed value of the dependent variable a the target value and denote it, t, and the prediction for the neural net by y. The objective in modeling data with neural networks is to get y "close" to t for all data points. The output or the predicted value, y, is some function of the independent variables. If a linear model is the choice

for the neural net then

$$y(\mathbf{x}) = \sum_{i=0}^{n} w_i x_i, \quad \text{where } x_0 = 1 \text{ and } w_i \text{ are unknown weights.}$$

If y is not some linear combination of the independent variables but believed to be some other function of the independent variables then a more general neural network assumes that

$$y(\mathbf{x}) = g\left(\sum_{i=0}^{n} w_i x_i \right),$$

where $g(\cdot)$ is called the activation function. The activation function could be a binary function, a linear function or some sigmoidal function. The weights in the model are determined so that an error function is minimized. The error function is dependent on the computer program being used. Suppose that the observed data are given by $(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_N, t_N)$. One commonly used error function is the least squares error function

$$E(\mathbf{w}) = \sum_{n=1}^{N} [y(\mathbf{x}_n; \mathbf{w}) - t_n]^2,$$

where $y(\mathbf{x}_n, \mathbf{w})$ is the predicted value of the dependent variable for the n^{th} observation. Another possible error function is the Minkowski error function

$$E(\mathbf{w}) = \sum_{n=1}^{N} |y(\mathbf{x}_n; \mathbf{w}) - t_n|^R, \quad \text{for some } R>0.$$

If the activation function being considered is linear and the error function is the least squares error function then the error function is minimized by $\mathbf{w} = (\mathbf{X}`\mathbf{X})^{-1}\mathbf{X}`\mathbf{t}$. In this case, one gets the usual multiple regression solution. If $g(\cdot)$, the activation function, is not linear or if the error function is not the least squares error function, then the weights must be determined through an iterative process.

Figure 2 describes a neural network with one hidden layer. It is interesting to note that existing literature about neural networks vary in the manner in which they label neural network structures. For instance, some authors count the input layer, so that Figure 2 would be called a 2-layer network while other authors also count the output layer and call it a 3-layer network. Warren Sarle (URL ftp://ftp.sas.com/pub/neural/FAQ.html) has suggested that only the hidden layers within a neural network should be counted. This is the convention followed in this paper. Consequently in this paper, Figure 2 describes a 1-layer network.

The neural network displayed in Figure 2 is a series of input variables connected to the two output variables by a network of weights and nodes within the hidden layer. The assumptions about the model are similar to the assumptions for Figure 1. That is, it is assumed that the each dependent variable is a function of some hidden layer variables which are functions of the independent variables plus some random error. Again x_0 is always equal to one. Each node within the hidden layer except the node z_0 (which will correct for bias in the hidden layer) is connected to the independent variables, $x_1, \ldots, x_n$. The outputs or the predicted values of the dependent variable are functions of the weighted values of the values produced at the nodes in the hidden layer, and the values at the nodes

are weighted values of the independent variables. All weights are determined so that an error function is minimized. This process is done by various algorithms depending upon whether the activation functions are differentiable. Note that neural networks can have multiple hidden layers, if needed, to provide a better fit to the data.

2. Motivation

The data that are the motivation for this paper was supplied by a client. The client was trying to predict total biomass with ten independent variables. The independent variables measured are incident radiation, reflectance at 8 different wave lengths (460 nm, 510 nm, 560 nm, 610 nm (red), 660 nm, 710 nm, 760 nm, and 810 nm (nir)), and the normalized difference vegetation index (NDVI = $(r_{810} - r_{610})/(r_{810}+r_{610})$ where r_w is the reflectance measurement at the wave length, w). The client used a neural network program called Braincel on these data and was impressed with the predictions for total biomass that were generated from the neural net. Braincel is a program developed by Promised Land Technologies (http://promland.com/).

In typical applications of neural networks the data set is divided into two pieces; one portion of the data is called the training data set and the other portion is called the test data set. The neural network program uses the training data set to develop a model, then the model is used to predict the dependent variable for the test data set and an error function is evaluated. This evaluation of the error function for the test data set is the typical method of determining the "goodness of fit" for the model. Following the guidelines in a typical application, this data was randomly divided into two parts, the training data set had 387 observations and the test data had 164. Using the training data, a neural network was developed. In order to assess goodness of fit, a graph of the observed total biomass was plotted against the total biomass predicted by the neural network and the coefficient of determination was calculated, see Figure 3. One might ask how this would compare to a typical multiple regression analysis. A simple multiple regression analysis on the independent variables was performed using a model that only contained linear terms. Stepwise regression was used to select terms for the final model. The terms in the final model were all significant at a 0.15 level or lower. Figure 4 shows the observed total biomass plotted against the predicted total biomass from the resulting linear regression model. A comparison of the plots in Figures 3 and 4 clearly indicates that the neural network did a much better job of fitting the data then the regression model. In order to improve the fit of a regression model, a quadratic model was then used. This model contained the squared and cross-product terms as well as linear terms. The terms in the final model were those that were significant at the 0.15 level or lower using stepwise regression enforcing the inclusion of all linear terms. Figure 5 shows a graph of the observed total biomass plotted against the predicted total biomass from the resulting quadratic regression model. Once again it appears that the neural network also outperforms the quadratic regression model. The predicted values for total biomass for the test data set were also calculated for each of these models: the Braincel model, the linear regression model and the quadratic regression model. Although they are not shown here, graphs of the predicted biomass were plotted against actual biomass for each of the three models using the test data. The patterns seen in Figures 3, 4, and 5 also appear in the test data with $R^2 = 0.6111$ for the Braincel model, $R^2 = 0.4208$ for the linear regression model, and $R^2 = 0.5440$ for the quadratic regression model. Again the neural network seems to out perform the regression models.

The above graphs motivated an interest in neural networks; they also raised two interesting

questions that need answering. First, could a lack-of-fit test, or equivalently, a test for model adequacy, be developed that does not depend upon test data? If so, then this test could help determine an appropriate number of hidden layers and the number of nodes within each layer that would need to be used in the neural network. Another question is can prediction intervals be determined to measure the reliability of a prediction from a neural network? These questions are not answered here, but seeking answers to these questions led to some experiences that are being reported.

One of the basic assumptions in using neural networks is that the input variables are uncorrelated with one another. This assumption is generally ignored in much of the literature. To see if the results from Braincel could be improved by using uncorrelated input variables, a principal component analysis was performed on the ten original input variables and the resulting ten principal components were used as input variables in Braincel to create a new neural network. Figure 6 shows a graph of the actual biomass plotted against the predicted biomass from Braincel using the principal components as inputs. While the value of R^2 is slightly smaller in Figure 6 then Figure 3, the plot in Figure 6 may be slightly more appealing to many data analysts.

3. Some Experiences with Braincel, Ripley's S+ Program, and Nychka's S+ Program

Consider a case where data that is being modeled is of the form

$$t_{ij} = f(\mathbf{x}_i;\beta) + \epsilon_{ij} \quad \text{for } i = 1,2,\ldots,m \text{ and } j = 1,2,\ldots,n$$

where $\epsilon_{ij} \sim$ i.i.d. $N(0, \sigma^2)$ and for each of m different values of the $\mathbf{x}_i$'s there are n values of y called true replicates. When there are true replicates, the typical test for model adequacy is performed by partitioning the residual sum of squares (SSR) from a fitted model into two parts. The first part is the sum of squares due to pure error (SSPE) which can be used to estimate σ^2 and the second part is the sum of squares due to lack-of-fit (SSLF). The test for model adequacy in those case when $f(\mathbf{x}_i, \beta) = \mathbf{x}_i'\beta$, Graybill (1976) is then done by calculating the test statistic

$$F = \frac{SSLF/(m-p)}{SSPE/m(n-1)}$$

where p is the rank of $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2,\ldots,\mathbf{x}_m]'$, $SSPE = \sum_{i=1}^{m} (t_{ij} - \bar{t}_{i\cdot})^2$, and $SSLF = \sum_{i=1}^{m} (\bar{t}_{i\cdot} - y_i)^2$. Model adequacy is rejected when $F > F(\alpha, m\text{-}p, m(n\text{-}1))$. In these cases, one says there is significant lack-of-fit.

To develop a test for adequacy of a neural network, it seems appropriate to begin with the case where there are true replicates as described above. For neural networks, a similar partitioning of the residual sum of squares can be done, however, questions that need to be answered are: What is p? and What is the distribution of the test statistic F? To help to determine p, data were simulated in the following way:

$$t_{ij} = f(x_{1i},x_{2i}) + \sigma\epsilon_{ij} \quad \text{for } i = 1,2,\ldots,25 \text{ and } j = 1,2$$

where $\epsilon_{ij} \sim$ i.i.d. $N(0,1)$, $x_{ji} \sim$ i.i.d. $N(0,1)$, and x_{1i}, x_{2i} and ϵ_{ij} are independent. In the first set of

simulations several linear and quadratic functions of x_{1i} and x_{2i} were used. In addition, several different values of σ were considered. Table 1 provides the expected values of the various sums of squares for different generating models when a correct regression model is used to fit the data.
A question of interest is what are the expectations of these sums of squares when the data are fit with a neural net? In order to gain some insight into the answer to this question, ten different sets of data

$$t_{ij} = 2x_{1i} + 1.60x_{2i} + 1.25x_{1i}^2 + 0.75x_{2i}^2 + 3x_{1i}x_{2i} + \epsilon_{ij} \quad \text{for } i = 1,2,\ldots,25 \text{ and } j = 1,2$$

were simulated. For each case, a neural net was fit to each data set using Braincel, and the sum of squares residuals (SSR) was calculated for each set. SSR was then partitioned into SSPE and SSLF = SSR - SSPE. Note that sum of squares residual and sum of squares lack-of-fit depend on the neural net, but sum of squares pure error does not. Visual examinations of the resulting sum of squares were unrevealing in terms of helping to determine an appropriate value of p. The sum of squares seemed to change dramatically from simulation to simulation for any given case. Sometimes the sum of squares residual were reasonably close to $(n-p^*)\sigma^2$ (where p^* = the number of weights created by the neural network) as one might hope, but at other times they were much too large to seem reasonable. Consequently, instead of answering the question about what p should be in creating F, the simulations generated more questions. One would be how does the number of nodes and the number of hidden layers relate to p, if at all? In order to gain a better understanding of what was occurring when using Braincel to model simulated data, it was decided that there was a need for a much larger simulation study. The data simulated for this study were generated by;where $\epsilon_{ij} \sim$ i.i.d. N(0,1), $x_{ji} \sim$i.i.d. N(0,1), and x_{1i}, x_{2i}, and ϵ_{ij} are independent.

Braincel is an add-on to Microsoft Excel, it is relatively easy to use in most situations but for a large simulation study it became apparent that a different program was needed. Ripley's S+ Program is a S-Plus function that is available on a disk in *Modern Applied Statistics with S-Plus* (1994) by William N. Venables and Brian D. Ripley.

Ripley's S-Plus program was used to develop neural nets for each of 1000 data sets. The neural networks each have two input variables and one hidden layer with two nodes. Note that the neural network determines estimates for 9 parameters. After the neural net was fit for each data set, the sum of squares residual was calculated. Figure 7 shows a histogram for the SSR except that all SSR > 600 were put into the last category. This graph has the general shape of a χ^2 which would seem reasonable, but the mean of 130.45 was considerably higher than the expected mean of the SSR for a quadratic model which is 44. In an effort to understand this problem, one of the data sets was selected and fit by a neural net 1000 times. The graph for the SSR for this repeated fitting is displayed in Figure 8. Examination of Figure 8 shows that about 25% of the time the SSR are about 48 (a somewhat reasonable value) while approximately 75% of the time the value is between 90 and 105 (an unreasonable value). This graph indicates that this Ripley's S+ program is performing poorly in terms of fitting a neural net to this data set.

After much consideration another program that fits neural networks was tried. Nychka's S+ program is a S-Plus function created by Douglas Nychka, Department of Statistics North Carolina State University. This function is available from the authors. (To obtain this function in UNIX follow these steps: mkdir funfits, cd funfits, ftp ftp.eos.ncsu.edu, cd/pub/statistics/stattools/pub/funfits, get funfits.tar.Z, quit, uncompress funfits.tar.Z, tar -xvf funfits.tar, follow the instructions in the README file.)

Nychka's S-Plus program was used to develop neural net models for each of the 1000 data sets simulated for Ripley's program. Again each neural network had two input variables and one hidden layer with two nodes. After the neural net was fit for each data set, the sum of squares residual was calculated. Figure 9 shows a histogram for the SSR. This graph also has the general shape of a χ^2 distribution and while the mean of the SSR is 54.2962 which is higher than the expected sum of squares for the residual for a quadratic model which is 44, it is at least reasonable. In an effort to determine reliability of this program, the data set that was selected for Figure 8 was fit with Nychka's neural net 1000 times. The results of these repeated fittings are shown in Figure 10. Figure 10 indicates that the program is producing consistent results since the SSR have a minimum value of 48.1173 and a maximum value of 48.1461. This graph also indicates that this program is performing much more consistently than Ripley S+ program.

In an final effort to compare programs, Braincel was used to fit neural nets to each of the 1000 simulated data sets. Again the neural networks have two input variables with two nodes in the hidden layer. Figure 11 shows the SSR which has the general shape of a χ^2 distribution but the mean of 95.5309 is higher than the expected sum of squares residual for a quadratic model which is 44. This figure indicates that Braincel is performing better than Ripley's S+ program but not as well as Nychka's S+ program.

Table 1. Expectations of sum of squares under different models when the correct model is fit.

	Model	
Sum of Squares	Linear	Quadratic
SSR	$47\sigma^2$	$44\sigma^2$
SSPE	$25\sigma^2$	$25\sigma^2$
SSLF	$22\sigma^2$	$19\sigma^2$

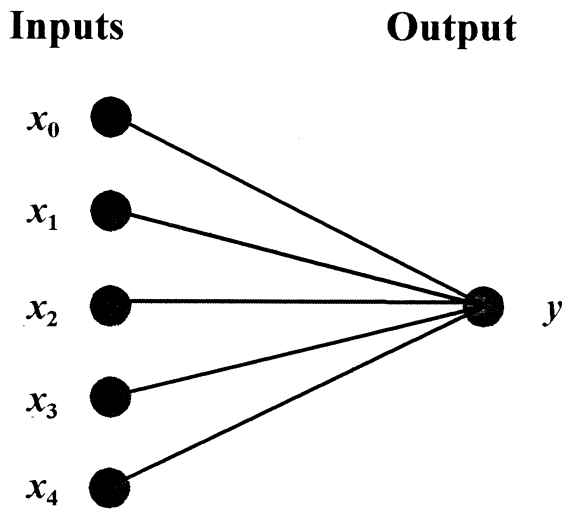

Figure 1. Neural Net with No Hidden Layer

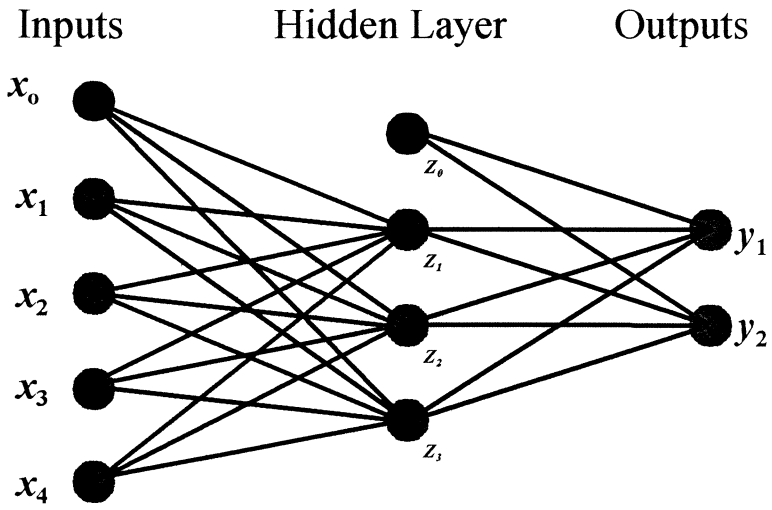

Figure 2. Neural Net with One Hidden Layer

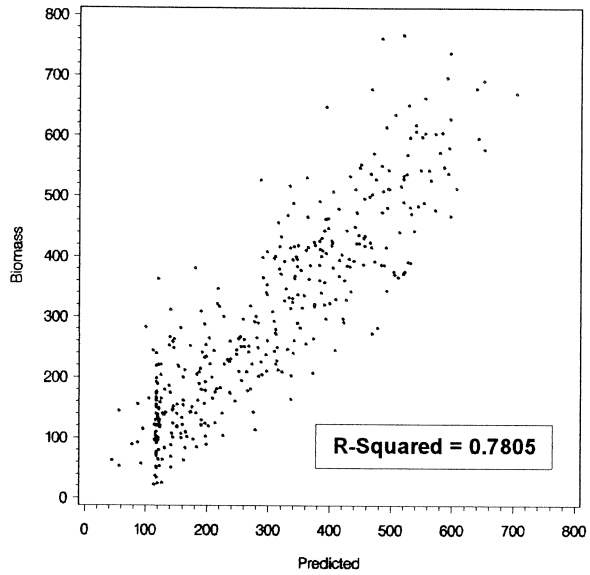

Figure 3. Actual Biomass vs. Predicted Biomass from Braincel

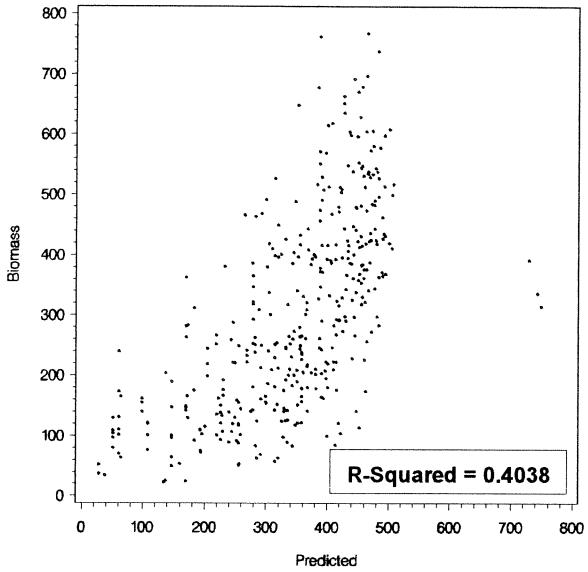

Figure 4. Actual Biomass vs. Predicted Biomass from Linear Regression

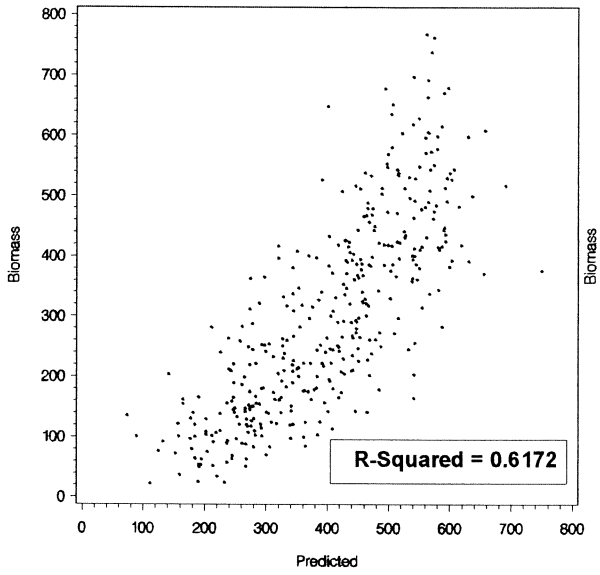

Figure 5. Actual Biomass vs. Predicted Biomass from Quadratic Regression

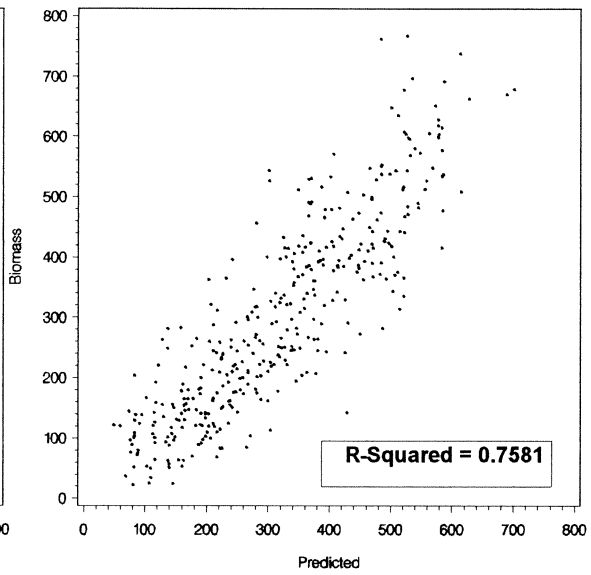

Figure 6. Actual Biomass vs. Predicted Biomass Using Braincel on Principal Components

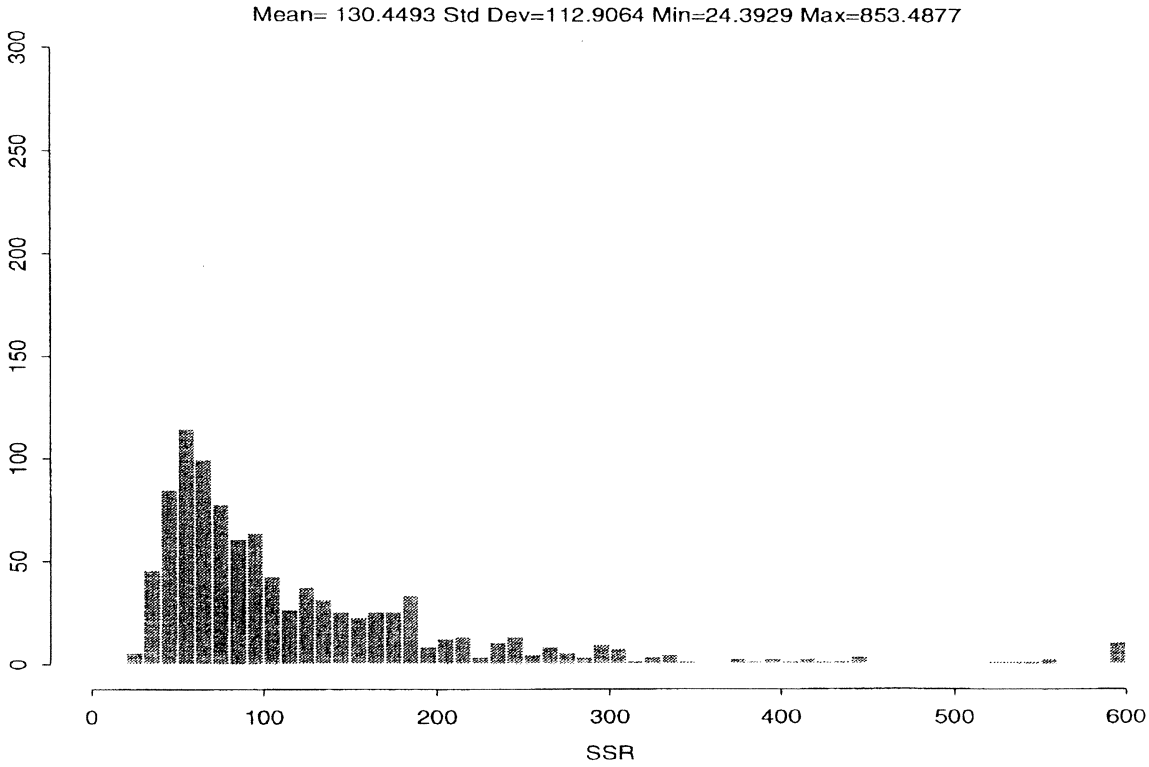

Figure 7. Sum of Squares Residuals for Ripley's S+ Program

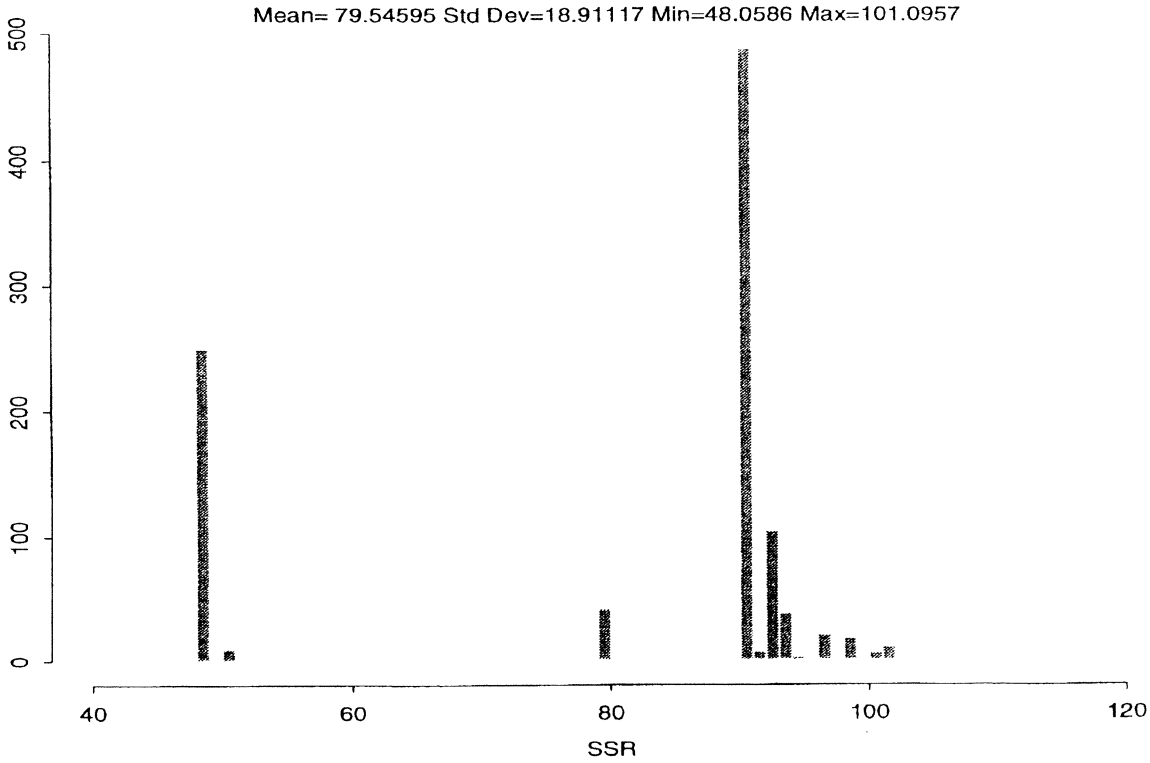

Figure 8. SSR for Data Set Fit with Ripley's S+ Program 1000 Times

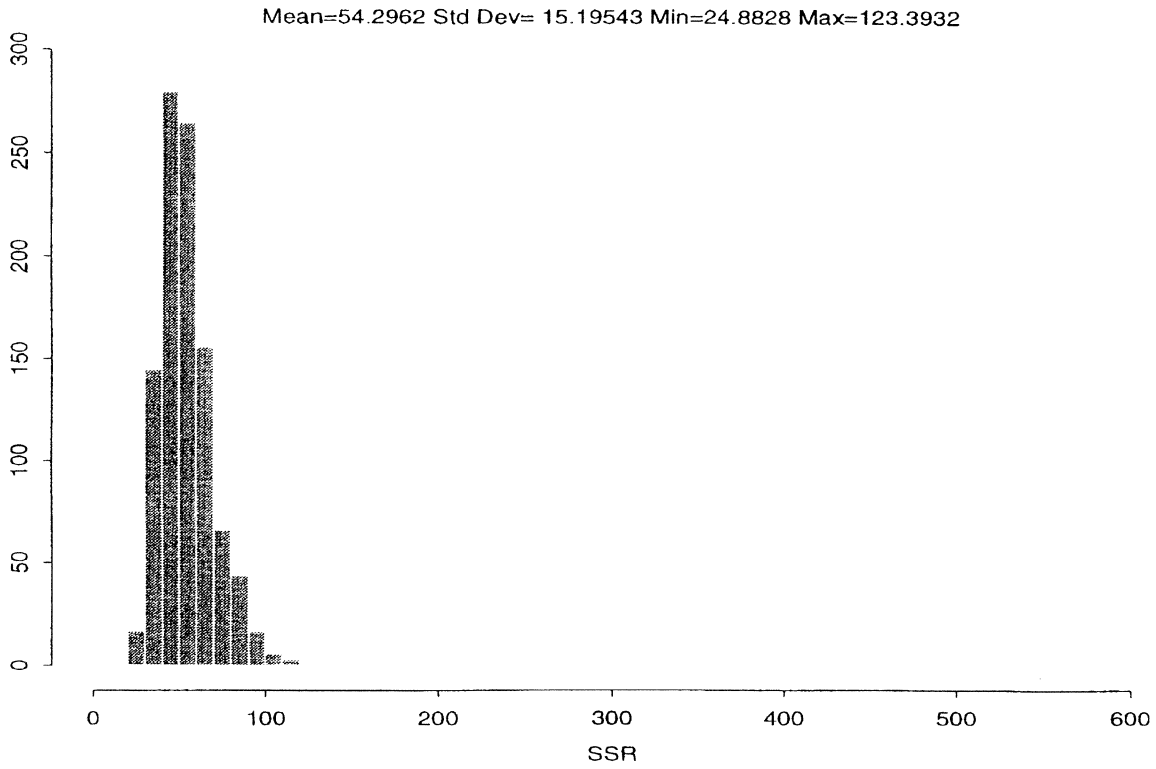

Figure 9. Sum of Squares Residual for Nychka's S+ Porgram

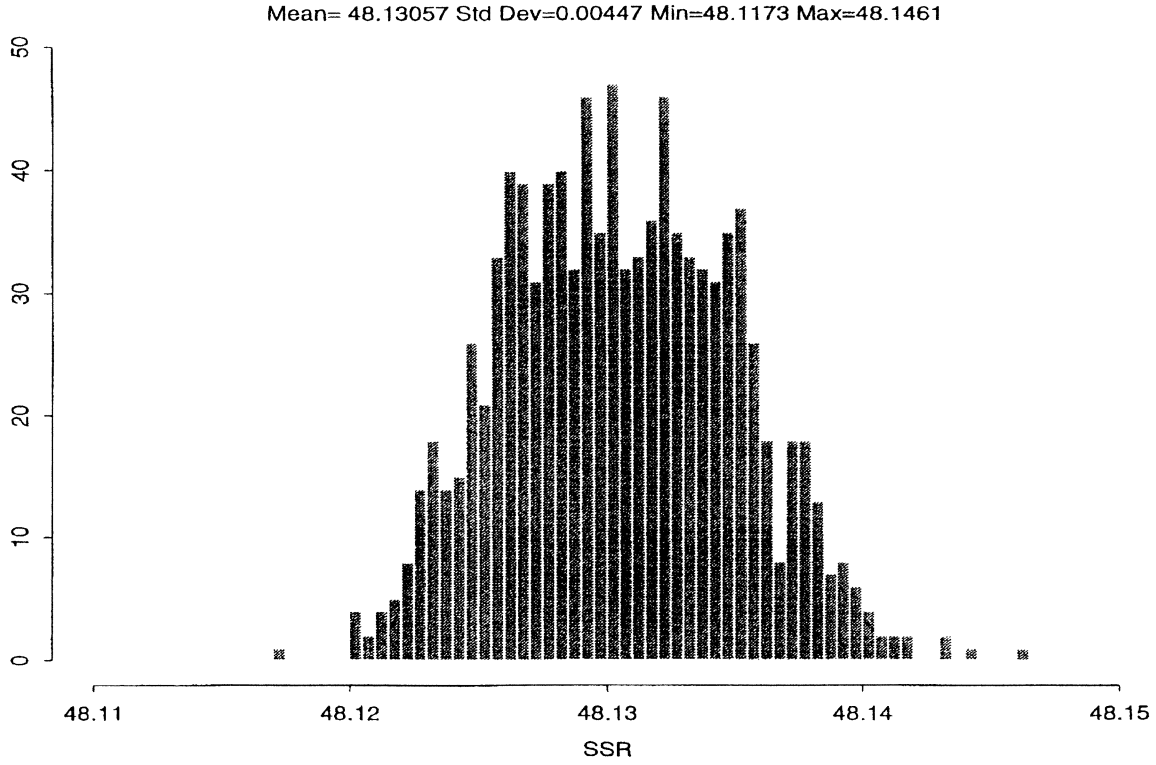

Figure 10. SSR for Data Set Fit with Nychka's S+ Program 1000 Times

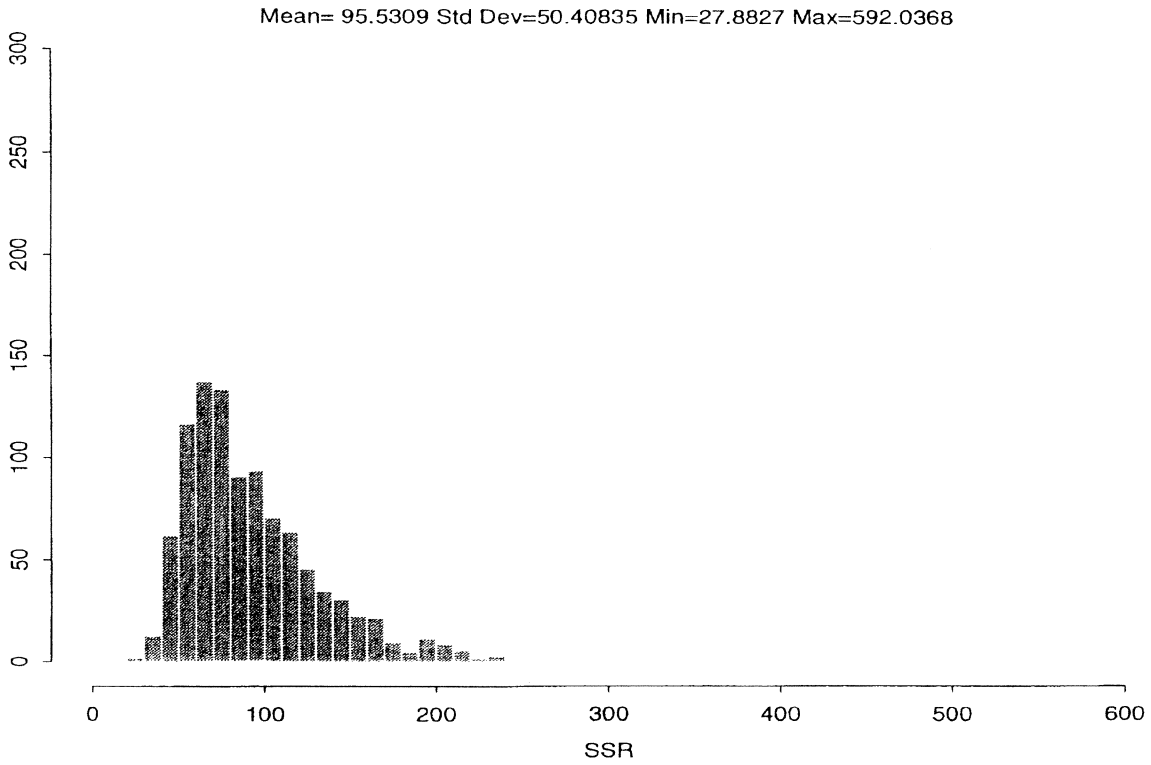

Figure 11. Sum of Squares Residual for Braincel

4. Summary

While neural networks appear to produce very interesting results in terms of their ability to accurately predict continuous data, one should use caution when selecting a neural network program or believing the results. One should always keep the type of data that is being modeled in mind. There are many neural network programs available and those described in this paper are but a few. Before any program is used, its reliability should be verified. That is, one needs to see if the iterative process consistently provides equivalent models by reanalyzing the data many times. Research is presently being conducted that will help data analysts determine the adequacy of the fit of a neural network. The results will be reported at a later time.

References

Bishop, Christopher M. *Neural Networks for Pattern Recognition*. Oxford, NY: Oxford University Press; 1995

Graybill, Franklin A. *Theory and Application of Linear Models*. Belmont, California: Wadsworth Publishing Company, Inc.; 1976

Kinoshita, June and Palevsky, Nicholas G. "Computing with Neural Networks", *High Technology*.

May 1987, 24-31

Venables W. N. and Ripley B. D. *Modern Applied Statistics with S-Plus*. New York: Springer-Verlag New York, Inc.; 1994