"Least Squares Fitting" Using Artificial Neural Networks

YARON DANON and MARK J. EMBRECHTS
Department of Nuclear Engineering & Engineering Physics,
Rensselaer Polytechnic Institute, Troy, NY 12180-3590

INTRODUCTION

When feedforward backpropagation neural nets are used with continuous numbers **(X)** in the input layer and continuous numbers **(Y)** in the output layer, the trained neural net can be viewed as a function that fits the **(X,Y)** data. The learning algorithm minimizes the sum of the squares of the differences between the output of the network and the given data points:

$$error = \sum_{p}^{p_{max}} \sum_{k}^{k_{max}} \left(o_k^p - t_k^p\right)^2, \qquad (1)$$

where o_k^p and t_k^p are the output value and training pattern value of the neural net for pattern p and output neuron k.

This process has a similar minimization objective to that used in the well known least squares fitting method (LSF), where we normally would have k = 1 for the least squares curve fit. The difference between fitting data points with a neural net and fitting with the LSF technique is that with a neural net the fitted function is represented by the network and does not have to be explicitly defined as in the LSF method. The learning process changes the internal parameters (weights) of the network such that the neural net can represent the given set of points in the best way by minimizing the error function.

In this paper the results from a backpropagation fit to various continuous functions will be presented, showing properties of neural network fitted functions. This technique is then used to fit the neutron time dependent background for neutron time of flight cross section measurements, where the theoretical shape of the curve is unknown. The distinct advantages of using a neural net over the LSF method will be discussed for this application.

NETWORK SIZE AND FITTING QUALITY

When training a feedforward net with error propagation for curve-fitting purposes, it is generally desirable to keep the number of neurons in the hidden layer(s) as small as possible, while still maintaining a relatively small value for the error after training (e.g. see Sietsma and Dow [1991]). To illustrate the effect of the size of the network on the quality of a curve-fit we will try to approximate the function

$$Y(X) = \frac{1}{2}e^{-0.8X} + noise, \qquad (2)$$

with a neural net and compare the results with the LSF method. The first step was to generate a set of 11 data points from this equation with up ±20% white noise. A small data base (11 points) was chosen for this example in order to clearly illustrate the effect of the network size on the quality of the curve fit. Various nets were trained (Table I), and training was halted when the error would no longer decrease. The momentum parameter was initially set to 0.79 and increased to 0.9 during the learning stage, while the learning

parameter was initially set to 0.3 and increased to 0.8 at the last stages of the learning process. The training started with 1% noise on the input data and the noise was gradually reduced to 0%.

TABLE I. Error after training for a data set generated by Eq. (2) as a function of the network architecture

Net size	Number of weights	RMS Error
1-1	2	0.0006406
1-2-1	7	0.0004706
1-2-1	10	0.0000456
1-3-3-1	22	0.0000433
1-10-1	31	0.0000000
1-10-10-1	46	0.0000000

From the results of this experiment (Table I) it can be noticed that the error after training can be reduced to zero by increasing the size of the network. The first network (no hidden layer, but with bias node) has two degrees of freedom. This is the same number of degrees of freedom as an exponential least squares fitting method, but the interpolation performance of the neural net (line with short dashes in Fig. 1) is clearly inferior to LSF (solid line in Fig. 1). The feedforward neural network architectures of Table I have between 2 to 46 degrees of freedom. When the number of weights is between 22 and 31, a zero error can be obtained after training, but the network just memorizes the data set and shows a poor interpolation capability (Fig. 1). The objective of training a neural net for curve fitting is certainly not to achieve a zero error after training. As a matter of fact, the networks with poor generalization after training to zero error would show a better performance if the training were halted after a certain finite error.

Deciding which exactly is the best curve fit is in part arbitrary, and depends on the criterion one wants to accept as a "best fit". Reasonable approximations can be generated with standard LSF techniques, but keep in mind here that an exponential shape for the curve fit was an a priori assumption. The neural network approaches can provide acceptable curve fits without any a priori assumptions regarding the shape of the function. However, a different network architectures yield very different curve fits. If just a few test data are available it is hard to obtain good statistics for the performance of a trained net from test data that were held back during the training process. Deciding on the right network architecture for an acceptable curve fit (or deciding when to stop training to prevent overtraining for the larger networks) remains an art in that case.

From the curves in Fig. 1 the network with zero error (1-10-1) gives the poorest results, as indicated by the oscillating nature of the approximation. The networks which still resulted in a finite error after training provided a better approximation, but there are still large discrepancies for $1.5 < X < 3.0$. This can partially be attributed to the fact that it is not such a simple matter to describe an exponential function by sigmoid functions.

When fitting with the least squares method the "goodness" of the fit is normally taken as the "chi-square" value. For LSF this is given by

$$\chi^2 = \sum_{i=1}^{i_{max}} \left[f(x_i) - y_i \right]^2, \tag{3}$$

where $f(X_i)$ is the fitting function, and Y_i are the given data points. Obviously, when fitting data with a neural network, minimizing Eq. [3] would not be the objective anymore. As illustrated in Table I and Fig. 1, the larger networks can be (over)trained to make χ^2 equal to zero, but generalization is rather poor in that case.

PRESENTING DATA TO A NEURAL NET

In a feedforward neural net several inputs are summed, a transfer function operation is performed, and the neuron provides just one output value. A typical transfer function is the sigmoid function

$$f(X) = \frac{1}{1+e^{-X}} . \tag{4}$$

The sigmoid function is bounded between zero and unity, and the same bounds apply for the output value of the neural net if a sigmoid function is used for the output neurons. Neural nets use bias nodes that connect to each neuron. The weights associated with the bias connections are obtained by the training process just as the regular weights. The effect of the bias is to shift the transfer function along the X-axis, as shown in Fig. 2. This shift will make the transfer function perform efficiently over a larger range of input values for a particular neuron by changing the threshold value for a particular learning process.

A different parameter that can be tweaked during the learning process is the temperature. A sigmoid with a temperature has a transfer function given by

$$f(X,T) = \frac{1}{1+e^{-X/T}} . \tag{5}$$

The actual effect of varying the temperature, T, is to change the slope of the transfer function as shown in Figure 3. The use of a low temperature can bring the shape of the sigmoid function to a signum function. The use of a high temperature (especially in the last layer of the network) allows the inputs to receive higher weights without sending the output immediately into saturation (zero or unity).

The independent variable, X, of the sigmoid function can exceed unity. This excess cannot be too large however, because large values in the exponent will normally cause an overflow error. These limitations on the size of the input and output variables of the artificial neural network require preprocessing of the data (see e.g. Thibault [1991]). Often taking the logarithm of large input or output variables and/or a normalization scaling between zero and unity is suggested. Normalizing the input and output parameters might cause an actual increase in the output error for a network trained to a certain precision when the errors are calculated for the inverse normalization results. (For example, if the normalization was done by taking the logarithm of the output neuron and the network was trained to an RMS error ε^2. If we assumed a normalized output X to have an error ε the unnormalized output would have a value of e^X with an error of εe^X.) Changing the temperature can help the network still accept larger input values without the necessity for normalization.

EXAMPLES OF CURVE FITTING WITH NEURAL NETS

A second curve fitting example was performed for the function

$$f(X) = \frac{\sin(4\pi X)}{4\pi X} + \text{noise}(X) \quad 0<X<1 \tag{6}$$

This function without noise was first approximated by a neural network with two hidden layers and 5 nodes in each hidden layer (a 1-5-5-1 network). The RMS error after training was 0.0000247, and the overall fit is shown in Fig. 4. The fit provides very accurate interpolations.

In a second step noise was added to the function according to

$$\text{noise}(X) = 0.2 \text{ X } \text{rnd}[-1,1] \tag{7}$$

The objective in this case is to obtain a curve fit that will do data smoothing and that yields a plat similar to Fig. 4. Again a (1-5-5-1) neural network was trained by error backpropagation to a RMS error of 0.00035. This net has an excellent fitting performance (Fig. 5) and the net can even trace the shape of the function in the X region where the noise is dominant ($0.7 < X < 1$). When the noise becomes too large compared to the function values the fitted curve flattens out. The results from Fig. 5 clearly show that a feedforward neural net trained with error backpropagation can be successfully employed for the smoothing and interpolation of noisy functions.

DETERMINING THE BACKGROUND FROM TOF MEASUREMENTS

The purpose of this section is to illustrate the elegance of non-parametric regression analysis with neural nets to experimentally observed data for which there is no a priori knowledge of the overall shape of the curve. In this application it is the aim to estimate background data for total neutron cross section measurements. These measurements were performed with the time of flight (TOF) method at the linear accelerator (LINAC) at RPI. The background counts are counts superimposed on the "good signal by off energy neutrons. The background count rate is estimated and then subtracted from the measurements. For these TOF measurements neutrons emitted from a tantalum target are flying through a 25 meter evacuated flight tube and hitting a neutron detector at the end of the flight tube. During their path, the neutrons are also passing through a thin sample of material whose cross section is being measured. Some of the neutrons will be removed from the beam by this thin sample, and by comparing a measurement with and without sample, the total neutron cross section can be calculated. The measurements consist of tallying the number of neutrons arriving at the detector as function of their arrival times. The time of flight is a measure for the energy of these neutrons.

The background is time dependent and depends on several parameters such as the flight tube material and geometry, and the detector and neutron emitting target geometry. The accuracy of the measurements could be limited by the accuracy by which the time dependent background can be measures. The background is estimated from a separate experiment in the same laboratory setup, where filters are introduced in the beam. The background measuring experiments described here lead to background estimates for 4 thicknesses of gold samples (with a thickness of 2.5, 5, 10 and 20 mils respectively) in the beam. For each thickness there are 6 background measurements, each one corresponding with a given neutron time of flight. It is now desired to interpolate the background for about 5,000 different

time of flight channels. There is no theoretical expression for the dependency of the background with the TOF, and a neural network will therefore be an excellent choice to try to correlate the measured data. For this experiment it is useful to extrapolate to an open beam case (0 mil thickness gold sample) at the same time, because these extrapolations can actually be compared with experimental values. For this interpolation/extrapolation curve fitting application a (2-4-2) feedforward artificial neural net was trained by error backpropagation. The input variables to this net are the time of flight and the gold sample thickness. The outputs from the net are the experimentally measured background and the corresponding background error. The network was trained to a RMS error of 0.0000653.

The results from this interpolation/extrapolation exercise are summarized in Fig. 6. Only the two extreme background measurements for the thickest (20 mil) and the thinnest (2.5 mild) sample are shown. The error estimate on the measurements is represented for the 20 mil sample by error bars. The background extrapolated to zero by the neural net is represented by the solid line. Not shown on this plot are the extrapolated error values for the background. The neural net solution gives results similar to those obtained from different standard curve fitting techniques that are traditionally used in our laboratory. The neural net solution is more elegant, however, because of its ease of application and the fact that no a priori assumptions about the shape of the background have to be made. When comparing the neural net solution with tradition curve fitting approaches for this application, the neural net results generally lead to more convincing data fits as well.

CONCLUSIONS

Feedforward neural networks trained with error backpropagation lend themselves very well to curve fitting applications and have several similarities with the least squares fitting methods in statistics. The neural net techniques offer a distinct advantage because of the elegance by which the technique can be implemented, and because no a priori assumptions about the shape of the fitting functions have to be made. The advantages of nonparametric regression analysis by neural nets over traditional LSF techniques become even more distinct when several independent variables come into play, wherein the case of fitting a nonlinear function (e.g., f (**X**,**Y**)) with LSF techniques is more numerically complicated and the neural network solution is the same as shown in this paper.

Feedforward neural nets are powerful methods for data smoothing, but proper care has to be taken while selecting the number of hidden layers, and the number of neurons in the hidden layers of feedforward nets. A large number of neurons and a small training error will lead to "memorization effects", and the interpolation predictions will be inaccurate. This difficulty could be alleviated by avoiding overtraining, or by selecting fewer nodes for the hidden layers.

REFERENCES

1. Jocelyn Sietsma and Robert J.F. Dow, "Creating Artificial Neural Networks That Generalize", Neural Networks, Vol. 4, pp. 67-69 (1991).
2. Jules Thibault and Bernard P.A. Grandjean, "A Neural Network Methodology for Heat Transfer Data Analysis", Int. J. Heat Mass Transfer, Vol. 34, No. 8, pp. 2063-2070 (1991).

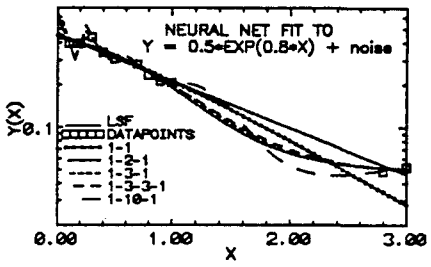

FIG 1. Curve fits for neural networks and LSF

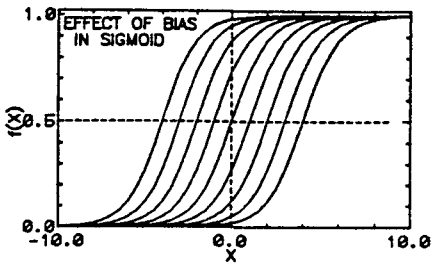

FIG 2. Bias in sigmoid transfer functions

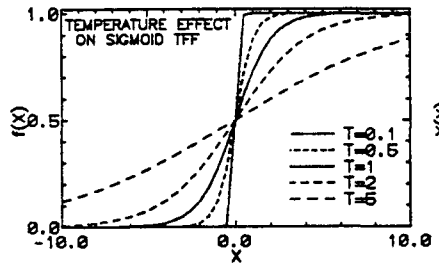

FIG 3. Effect of temperature on sigmoid transfer function

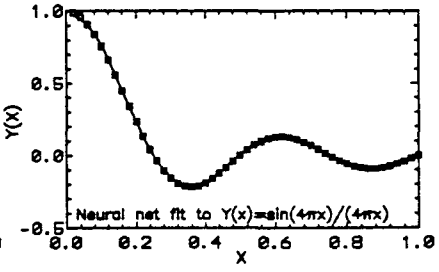

FIG 4. 1-5-5-1 neural net fit for $f(x) = \frac{\sin(4\pi x)}{4\pi x}$

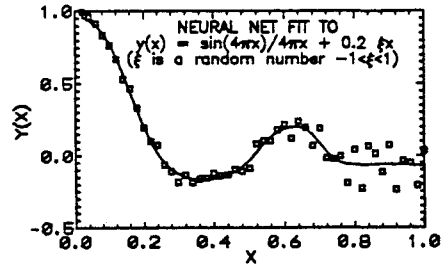

FIG 5. Neural net fit for $f(x) = \frac{\sin(4\pi x)}{4\pi x}$ + noise

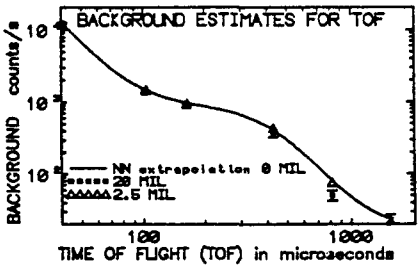

FIG 6. Estimating background for TOF measurements