Fast curve fitting using neural networks

C. M. Bishop
AEA Technology, Harwell Laboratory, Oxon OX11 0RA, United Kingdom

C. M. Roach
AEA Technology, Culham Laboratory, (Euratom/UKAEA Fusion Association) Oxon OX14 3DB, United Kingdom

(Received 4 February 1992; accepted for publication 22 June 1992)

Neural networks provide a new tool for the fast solution of repetitive nonlinear curve fitting problems. In this article we introduce the concept of a neural network, and we show how such networks can be used for fitting functional forms to experimental data. The neural network algorithm is typically much faster than conventional iterative approaches. In addition, further substantial improvements in speed can be obtained by using special purpose hardware implementations of the network, thus making the technique suitable for use in fast real-time applications. The basic concepts are illustrated using a simple example from fusion research, involving the determination of spectral line parameters from measurements of B IV impurity radiation in the COMPASS-*C* tokamak.

I. INTRODUCTION

The fitting of parametrized functional forms through sets of experimental data (curve fitting) is a widespread problem in data analysis. The optimal values for the parameters are conventionally found by minimizing an error measure, often taken to be the sum of the squares of the errors between the observed data values and those predicted by the function. If the functional form is linearly dependent on the parameters (e.g., a polynomial), then the minimization problem is linear and is easily solved. In many cases, however, it is necessary to consider functional forms which depend nonlinearly on the unknown parameters. The minimization procedure in such cases generally involves an iterative algorithm starting from some initial guess. Such iterative methods are computationally intensive, and hence slow, and for complex problems the need for a suitable initial guess can require human intervention to ensure convergence to the correct solution. For applications involving high volumes of data, or for real-time applications, there is considerable interest in techniques which can automate the curve fitting process and which can operate at high speed.

In this article we use neural networks to provide a novel technique for determining the optimal parameter values of the function directly from the raw data. This approach is much faster than iterative methods and does not require an initial guess for the solution. Furthermore, for real-time applications, it is possible to implement the network in special purpose hardware, thereby exploiting the intrinsically parallel nature of neural networks and hence giving very high processing speeds.

In Sec. II we give a brief overview of neural networks, and we then describe in more detail a particular class of network known as the multilayer perceptron. The neural network approach to curve fitting, based on the multilayer perceptron, is illustrated in Sec. III using a simple example from nuclear fusion research involving measurements of the B IV impurity radiation spectrum at the COMPASS-*C* tokamak. The problem in this case is to determine the width and location of a single spectral line from measurements made at a number of wavelengths. We show how the network is able to predict the width and location directly from the spectral data in a single step process. This example is chosen both for its simplicity, and because it illustrates the major points which must be addressed in applying the technique to more sophisticated problems.[1,2] Section IV contains a brief summary and a discussion of the advantages and disadvantages of the neural network approach.

II. THE MULTILAYER PERCEPTRON

The study of neural networks has undergone a revival in recent years, in part due to the development of powerful new algorithms. Neural networks are analog computational systems whose structure is inspired by studies of the brain. Many different architectures of neural network have been developed to tackle a variety of problems, and research in this area continues at a rapid pace. For an introductory review of neural networks, see Ref. 3. In this section we give a brief overview of a relatively simple but very widely used network type known as the multilayer perceptron (MLP). This class of network will form the basis of our approach to curve fitting described in Sec. III.

An MLP consists of a network of units (also known as processing elements, neurons, or nodes) as illustrated in Fig. 1. Each unit is shown as a circle in the diagram, and the lines connecting them are known as weights or links. The network can be thought of as describing an analytic mapping between a set of real-valued input variables x_m $(m=1,...,M)$ and a set of real-valued output variables y_n $(n=1,...,N)$. The input variables are applied to the M input units at the left of the diagram ($M=4$ and $N=2$ in the case of Fig. 1). These variables are multiplied by a matrix of parameters w_{lm} $(l=1,...,L;\ m=1,...,M)$ corresponding to the first layer of links. Here L is the number of units in the middle, or hidden, layer ($L=3$ in the example shown

 0034-6748/92/104450-07$02.00

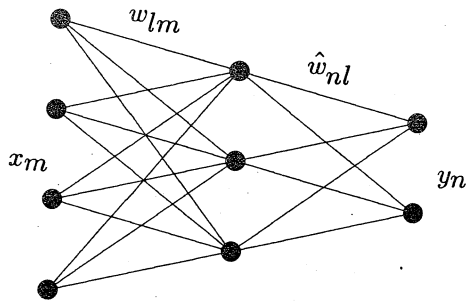

FIG. 1. A MLP with 1 hidden layer.

in Fig. 1). This results in a vector of inputs to the units in the middle layer. Each component of this vector is then transformed by a nonlinear function $f(\cdot)$ so that the outputs of the middle layer units can be written

$$z_l = f\left(\sum_{m=1}^{M} w_{lm}x_m + \theta_l\right) \quad (l=1,...,L), \tag{1}$$

where θ_l is an offset (or threshold). The function $f(\cdot)$ is generally chosen to be the sigmoid defined by

$$f(x) \equiv \frac{1}{1+e^{-x}} - \frac{1}{2}. \tag{2}$$

The outputs from the hidden layer units are now multiplied by a second matrix of parameters $\hat{w}_{nl}$ $(n=1,...,N;$ $l=1,...,L)$, and offsets $\hat{\theta}_n$ are added to the components of the resulting vector to generate the network outputs:

$$y_n = \sum_{l=1}^{L} \hat{w}_{nl}z_l + \hat{\theta}_n, \quad (n=1,...,N). \tag{3}$$

Combining Eqs. (1) and (3) we see that the entire network corresponds to a mapping from inputs x_m to outputs y_n which is specified by the analytic function

$$y_n(x_1,...,x_M) = \sum_{l=1}^{L} \hat{w}_{nl} f\left(\sum_{m=1}^{M} w_{lm}x_m + \theta_l\right) + \hat{\theta}_n, \tag{4}$$

where $f(\cdot)$ is defined by Eq. (2). This mapping is parametrized by the quantities w_{lm}, θ_l, $\hat{w}_{nl}$, and $\hat{\theta}_n$, and can readily be extended to include more than one hidden layer.

The functional form in Eq. (4) appears somewhat arbitrary. Its importance, however, stems from two crucial properties:

(1) For suitable choices of the parameters w_{lm}, θ_l, $\hat{w}_{nl}$, and $\hat{\theta}_n$ the MLP mapping with a single hidden layer can approximate, with arbitrary accuracy, any given nonlinear multivariate mapping (subject to some mild restrictions) provided the number L of middle layer units is sufficiently large. A formal discussion of this property can be found in Ref. 4.

(2) Given a set of P exemplar vector pairs $\{x_m^p, y_n^p\}$ $p=1,...,P$ characterizing a particular mapping, there exist procedures, based on the technique known as error backpropagation, for determining appropriate values for the parameters w_{lm}, θ_m, $\hat{w}_{nl}$, and $\hat{\theta}_n$ so that the network function in Eq. (4) approximates the required mapping. (In the neural network terminology these are referred to as training or learning algorithms).

The training of an MLP involves choosing a set of weights to optimize the accuracy of the mapping and is a computationally intensive process. A large database of examples (independently determined) is used to teach the neural network to perform the mapping. Training algorithms generally aim to minimize an error function E_{net} defined to be the sum of squares of the errors between the output vector $\mathbf{y}(\mathbf{x}^p)$ of the network (for given input vector $\mathbf{x}^p$) and the corresponding target vector $\mathbf{y}^p$, summed over all exemplars p:

$$E_{\text{net}} = \sum_{p=1}^{P} \sum_{n=1}^{N} [y_n(\mathbf{x}^p) - y_n^p]^2. \tag{5}$$

Thus E_{net} is a function of the w's and θ's. (Note that other definitions of the error measure can also be used). There exists an efficient technique for evaluating the derivatives of E_{net} with respect to these parameters. Knowledge of the derivatives allows E_{net} to be minimized using a number of standard optimization procedures. Gradient descent is the simplest method, and when combined with the derivative algorithm the resulting training procedure is known as backpropagation. A detailed account of the backpropagation procedure can be found in Ref. 5. More powerful optimization algorithms exist, and in this article we adopt a limited-memory quasi-Newton method, which is considerably faster than gradient descent.

If the network is to generalize to new data, it is important that the training data spans the input space where the mapping is to be applied. Neural networks are good at interpolating between training examples, but poor at extrapolating beyond the training experience.

The number L of hidden units is a parameter of the neural network model. The optimum number of hidden units depends both on the specific problem being solved and on the size of the available data set. There exist a variety of procedures for chosing L, of which the simplest is a cross validation technique using independent training and test sets. Networks having various numbers of hidden units are trained using the training set and their performances are measured by calculating the mean square error with respect to the test set. The network having the smallest test error is then chosen.

III. AN EXAMPLE: FITTING A SPECTRAL LINE

Curve fitting involves optimizing a parametrized functional form to fit a set of data points. The parametrized function may depend on an arbitrary number of variables, but in this article we consider only functions of one variable λ. It would be straightforward to extend the methods discussed to deal with higher dimensional problems. The basic problem in curve fitting is to find the N parameter values $\alpha_1,...,\alpha_N$, associated with the functional form $F(\lambda;\alpha_1,...,\alpha_N)$, which best represent the underlying trends in a set of M data points $(\lambda_1,x_1),...,(\lambda_M,x_M)$. For the repetitive curve fitting problems which are addressed here,

the values of λ_m, corresponding to the data values x_m, are considered to be fixed.

When the functional form $F(\lambda;\alpha_1,...,\alpha_N)$ is nonlinearly dependent on the parameters, the conventional approaches to finding the optimal parameter values are iterative. Least-squares fitting is one such technique, and this involves minimizing with respect to $\alpha_1,...,\alpha_N$ an error measure $E_{\rm ls}$, where

$$E_{\rm ls}=\sum_{m=1}^{M}[x_m-F(\lambda_m;\alpha_1,...,\alpha_N)]^2. \quad (6)$$

$E_{\rm ls}$ measures the discrepancy between the functional form and the experimental data. The conventional method for finding the minimum error is to adjust the parameters using an iterative algorithm such as steepest descent or conjugate gradients. While in many cases such an approach is satisfactory, there are two potential drawbacks: iterative algorithms are usually computationally intensive, and an appropriate initial guess is required.

The neural network approach to curve fitting described here involves the use of a multilayer perceptron to form a direct mapping between the measured data $x_1,...,x_M$ and the optimal parameter values $\alpha_1,...,\alpha_N$. A set of examples of the mapping must be generated independently in order to train the network, and a typical procedure for achieving this is illustrated below. The N target outputs of the network for the $p^{\rm th}$ example y_n^p are set equal to the best fit parameters α_n^p as determined by a method like least squares. Meanwhile the M corresponding network inputs for the pth pattern are set to the measurement values x_m^p. The training of the network is itself a slow iterative process involving the minimization of the error measure $E_{\rm net}$ obtained from Eq. (5):

$$E_{\rm net}=\sum_{p=1}^{P}\sum_{n=1}^{N}[y_n(\mathbf{x}^p)-\alpha_n^p]^2. \quad (7)$$

$E_{\rm net}$ (which is different from the least-squares error measure $E_{\rm ls}$) is minimized with respect to the weights in the network. Once trained the network can very rapidly perform the curve fitting operation on new data. As was shown in Sec. II, the network mapping involves the simple operations of a matrix multiplication, a nonlinear transformation and a second matrix multiplication, and these can typically be performed much more rapidly than the corresponding iterative algorithm.

Optimization problems are slow, and the presence of local minima leads to the need for human intervention. Algorithms such as simulated annealing can circumvent this problem, but such techniques consume even more computer resources than the iterative method. Conventional curve fitting approaches require an optimization for every set of data points to be fitted. This contrasts with the neural network approach where only one optimization has to be performed in order to solve a whole class of fitting problems. The class of problems soluble using a single neural network is limited to those where the same type of data is to be fitted with a particular functional form. Neural networks provide a very attractive approach for repetitive curve fitting problems, which are particularly prevalent in the analysis of experimental data.

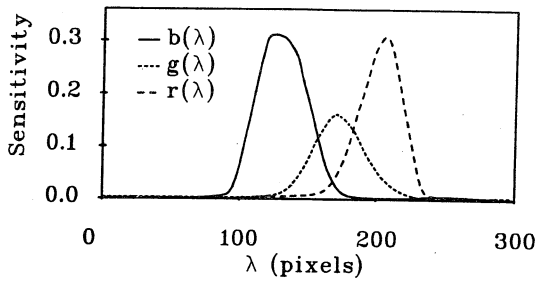

FIG. 2. Instrument response functions for the *B*, *G*, and *R* probes as functions of wavelength (λ). The appropriate range of wavelengths is known, but we adopt the pixel number scale shown in the figure.

In the application of the neural network as described above, we have assumed that the λ_m are fixed. While this is true for the COMPASS spectroscopy example we are about to describe, in some problems the λ_m may vary. The restriction to fixed λ_m can be circumvented by adding the λ_m values as additional inputs to the neural network, but there has been no need to pursue this method here.

The simple example described in this section illustrates the principles of using neural networks for curve fitting. The functional form is a simple Gaussian which depends on three parameters corresponding to width, height, and location. Extensions to more complex problems are straightforward as can be seen in Ref. 1 where an application to the analysis of charge exchange recombination spectra on the JET tokamak, involving up to 11 overlapping spectral lines, is outlined.

A. COMPASS spectroscopy data

The tokamak concept is currently the favored magnetic confinement system for research into producing controlled nuclear fusion.[6] Inside the tokamak heavy isotopes of hydrogen are ionized to form a plasma; and if the ions are heated to a sufficient temperature fusion reactions will ensue. COMPASS is a tokamak experiment at Culham Laboratory and in this article we are concerned with the reconstruction of some COMPASS plasma parameters from ultraviolet (UV) spectroscopy measurements. The COMPASS spectrometer is sensitive to a very clean part of the UV wavelength region which contains one spectral line coming from B IV impurity ions. The spectrometer is used principally for plasma rotation measurements and has been described in more detail in Ref. 7. Essentially the spectrometer has three probes which measure light intensity over three different wavelength regions. The instrument response functions [$b(\lambda)$, $g(\lambda)$, and $r(\lambda)$] are plotted in Fig. 2.

The measurements from each probe, called *B*, *G*, and *R* for "blue," "green," and "red," respectively, are given in terms of the actual spectrum from the plasma $F(\lambda)$ by

$$B=\int_0^\infty F(\lambda)b(\lambda)d\lambda,$$

$$G=\int_0^\infty F(\lambda)g(\lambda)d\lambda, \tag{8}$$

$$R=\int_0^\infty F(\lambda)r(\lambda)d\lambda.$$

Given $F(\lambda)$, the forward problem of computing B, G, and R is straightforward, but the task on hand is the more difficult inverse problem of computing a representation for $F(\lambda)$ from the measurements.

The spectrum is supposedly a single Gaussian line from B IV, and this hypothesis has been checked using a multichannel spectrometer. Thus three measurements suffice to fix the amplitude A, position P, and width W of the Gaussian. The analytic form of the spectrum is defined:

$$F(\lambda)=Ae^{-(\lambda-P)^2/W^2}. \tag{9}$$

Any noise in the measurements causes errors in the reconstructed parameters, and it is clear that having more measurements would reduce this problem. As it is straightforward to compute B, G, and R given $F(\lambda)$, an iterative least-squares technique can be applied. In this approach the Gaussian parameters are first guessed, and then adjusted iteratively so as to minimize the sum of the squares of the differences between the B, G, and R measurements and the corresponding values computed from the current estimates of the Gaussian parameters. This method is slow, and so the technique currently used in COMPASS involves calculating B, G, and R for various positions, widths, and amplitudes of a Gaussian representation of $F(\lambda)$ and interpolating between grid points. While this interpolation method is fast and can achieve high accuracy, it would be an awkward method to implement were extra probes to be added. Hence the interest in trying an intrinsically more adaptable method.

B. Neural network approach

For the problem on hand, a mapping is required to transform the B, G, and R measurements to estimates of A, P, and W, the amplitude, position, and width of the B IV line. It is clear that if P and W are fixed then the measurements are linearly proportional to A. Given the linear invariance of the problem with respect to A, there is no merit in calculating A using a neural network. The normalization of the three measurements is redundant information as far as extracting P and W is concerned. Thus, normalizing the measurements, we are left with two independent quantities which we choose to call $\hat{B}$ and $\hat{R}$ (blueness and redness) given by

$$\hat{B}=B/(R+G+B),$$

$$\hat{R}=R/(R+G+B). \tag{10}$$

A $2-L-2$ MLP can then be trained to map $\hat{B},\hat{R}\rightarrow P,W$. The number L of hidden units is a free parameter and has been optimized using the cross validation procedure described above.

In the present application there are three independent measurements available from the spectrometer, and we have chosen to fit these with a Gaussian functional form which has three degrees of freedom (corresponding to the parameters A, P, and W). Independent data from a 20 channel interferometer have been used to confirm that the spectrum is indeed approximately Gaussian. For most curve fitting applications, however, it is usual to choose the number of degrees of freedom in the functional form to be smaller than the number of independent measurements. This results in an overdetermined problem which is less sensitive to noise on the data.

For simplicity we have chosen to train the network using synthetic data obtained by generating a set of Gaussians with known values of A, P, and W and then evaluating the corresponding values of R, G, and B using (8). An alternative approach is to use real data for which the values of R, G, and B are known and then to use a standard least squares procedure to obtain corresponding values for A, P, and W. For more complex problems than the one considered here it is preferable to use real data for network training since it is difficult to model the detailed features and statistical properties of real data sufficiently accurately.

The training data set has been chosen to fill uniformly the cuboid region of output space:

$$0.03<A<0.15,$$

$$130.0<P<300.0,$$

$$10.0<W<100.0,$$

with rejection of the subregion of this space where $P>230.0+1.5\times W$. Elimination of this subspace was necessary to prevent problems arising from two different points in the output space corresponding to the same point in the input space. The reason for this uniqueness problem is linked with the bump in the high wavelength tails of the blue and green instrument response functions (see Fig. 2). The ambiguous regions were identified by first training a network over the cuboid subregion of the output space defined above. The quality of the neural network fits was found to be very poor in two regions of the P, W plane, because of the ambiguity phenomenon. Elimination of one of these two regions cured the problem, and it was decided to reject the region corresponding to the larger values of P since this was less likely to be encountered in the experiment. This procedure requires human intervention, but only once prior to the training of the neural network.

It is a simple matter then randomly to pick Gaussian parameter values in the space defined, and to use Eqs. (8) and (10) and the measured instrument functions to compute the $\hat{B}$ and $\hat{R}$ values. 1000 artificial examples of the mapping were generated and 900 of those were randomly selected as a training set, leaving 100 as test examples with which to measure the generalization accuracy of the neural network mapping.

The quality of the neural network results can be quantified both using simulated data by assessing how well it fits

TABLE I. Results from MLPs with different numbers of hidden units. E_{train} and E_{test} are the averages of the fractional rms training and test errors in P and W relative to the spreads of the parameters. ΔP and ΔW are the rms test errors in position and width, respectively. These results were mostly obtained after 6000 iterations of the quasi-Newton optimization algorithm. 10 000 iterations were used in the training of the networks with 50 and 60 hidden units.

No. Hidden Units	10	20	30	40	50	60
E_{train}	0.148	0.094	0.064	0.061	0.039	0.046
E_{test}	0.177	0.109	0.083	0.082	0.045	0.067
ΔP	6.94	4.11	3.25	2.84	1.98	2.96
ΔW	5.04	3.17	2.39	2.47	1.20	1.78

a random set of test examples (independent of the training set), and using real data by comparing with the least-squares technique.

C. Performance on simulated data

A number of training runs have been performed using different numbers of hidden units, and the results are given in Table I. In the runs the network with 50 hidden units has achieved the best performance, and so we present the corresponding test data results in more detail. Figure 3 shows the error distributions from the neural network, with the sample containing all 1000 simulated patterns (i.e., the training set is included here). The plots are quite encouraging, in that there are no very badly reconstructed patterns, and the error distribution is reasonably uniform throughout the space. There is some evidence of a slight systematic misfitting of wide spectra. Figure 4 shows three example test spectra (solid curves) with the neural network fits superimposed (dashed curves).

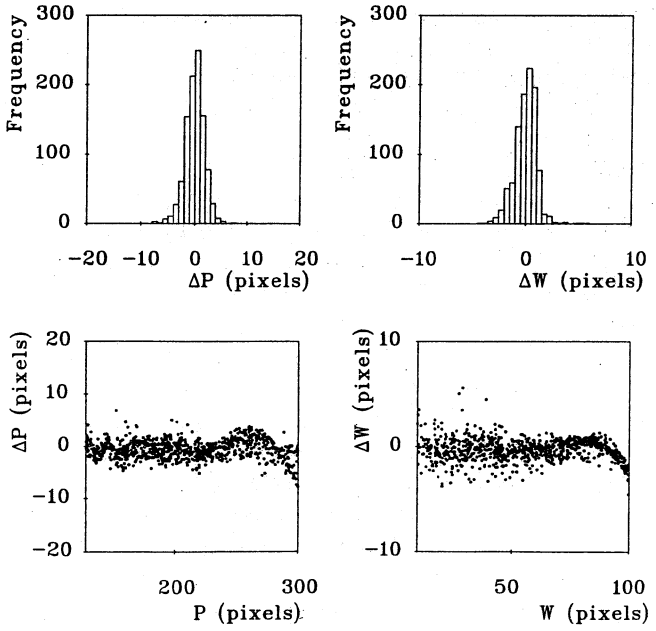

FIG. 3. Error distributions in the position and width. The scatter plots show how the errors vary with postion and width.

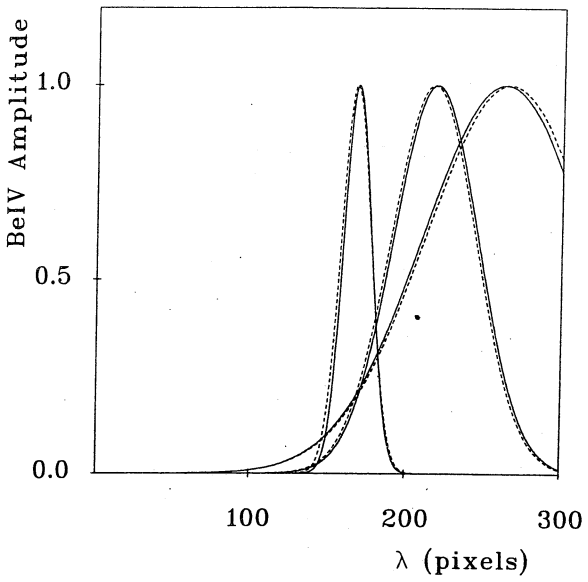

FIG. 4. Three example test spectra (solid) with the neural network fits superimposed (dashed).

D. Performance on experimental data

The network with 50 hidden units, as trained above, has also been tested using real data from the COMPASS-C experiment. Figure 5 shows the experimental measurements $\hat{B}$ and $\hat{R}$ as functions of time, and Fig. 6 shows the least squares and neural network fits to P and W for shot number 6142. The data are noisy, and while this affects the physics results extracted from the data, it has no bearing on our assessment of the network performance. The sudden jump in P in the figure is associated with a change in the plasma rotation due to a process known as mode locking.[8] The spectrometer diagnostic was built primarily to

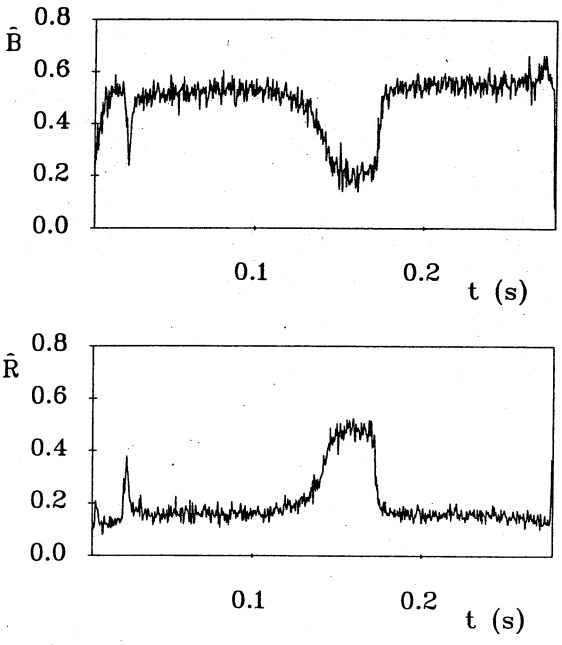

FIG. 5. $\hat{B}$ and $\hat{R}$ as functions of time (t).

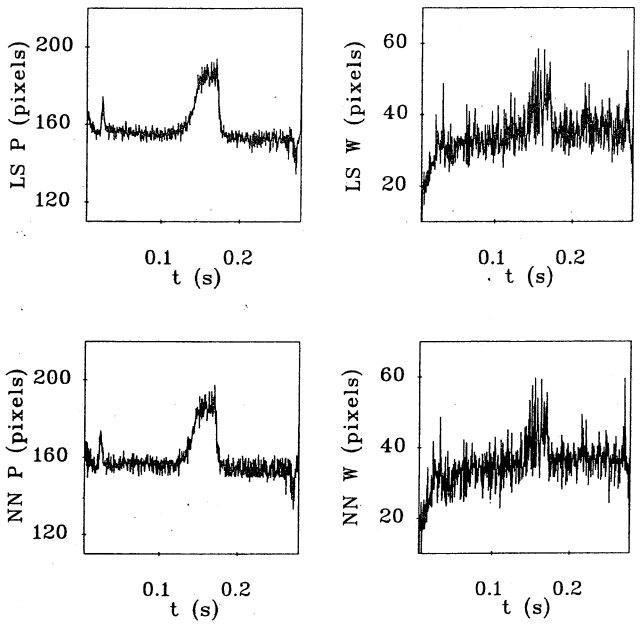

FIG. 6. Least-squares (LS) and neural network (NN) fits to P and W as functions of time.

study this process. In fact B, G, and R measurements are made every 40 μs but the data in the figure have been averaged over 400 μs. The neural network results are just perceptibly different from the least-squares fits, and this is interpreted as being due to errors in the neural network fit. Given the level of precision in the data this is not a significant problem. The rms discrepancies between the neural network and the least squares fit results for P and W, as measured on this real data set, are 2.6 and 1.8 pixels, respectively, slightly higher than the test errors measured using simulated data. This increase is probably due to the distribution of the real data being significantly different from the distribution of the simulated data.

It is important in running a neural network with real data that there should be some means of flagging examples where the neural network fit is poor. In this particular problem it would be straightforward to implement such a scheme. Given the neural network fit for P and W it is straightforward (and fast) to compute $\hat{B}$ and $\hat{R}$ using Eqs. (8) and (10) with the best fit form for $F(\lambda)$, and then to compare these calculated values with those values computed directly from the actual measurements. If there are large discrepancies then the pattern could be flagged as BAD, and the data could then be analyzed using another method. One reason for a poor network fit would be that no similar patterns were included in the training set, and so this situation could be remedied by retraining the network every so often, adding the BAD patterns to the training set.

IV. DISCUSSION

The problem of optimizing the parameters of a given functional form to fit experimental data points is frequently encountered in data analysis. In many cases there is considerable interest in nonlinear curve fitting methods which are fast and automatic. In this article we have shown that multilayer perceptron neural networks can provide a direct mapping from the measured data onto the parameter values associated with the best fit function. While the training of such networks is computationally intensive, the trained networks can process new data very rapidly. In particular they are typically very much faster than the more conventional iterative techniques.

For real-time applications, further substantial improvements in processing speed can be achieved by implementing the network in special purpose hardware. At the simplest level this could consist of a pipelined digital processor, such as the Intel i860, which can implement the basic vector-matrix multiplications of the neural network in a very efficient form. Genuine hardware implementations, however, take full advantage of the highly parallel nature of neural network processing, and use individual hardware elements for each processing unit in the network. An example of a hybrid digital-analog network implementation for the processing of tokamak data in real-time is described in Ref. 9.

The accuracy expected of a neural network solution is typically somewhat less than that obtained from a least-squares iterative approach. In many cases this reduction in accuracy is of little consequence, but even when the highest accuracy is desired, the neural network solution can be used as an initial guess for the standard iterative approach. This use of an educated initial guess allows the iterative method to converge in fewer iterations. Such a combined neural-iterative system can also resolve another difficulty of the purely iterative method which is the problem of "convergence" to the wrong solution if the initial guess is not sufficiently accurate. The network initial guess should be accurate enough to allow the use of the iterative method without human supervision.

A second potential difficulty with the neural network approach is the requirement that the data presented to the trained network should be statistically similar to that which formed the training set. In this sense the network generalizes by "interpolating" within the range of the input data, while the network is not expected to give reliable results if substantially novel input data are used. Again, in many applications a suitable training set will be available and this is unlikely to represent a problem. In situations where novel data may be expected to occur it may be possible to provide a cross check on the quality of the network output. This was possible in the example given in Sec. III by using the network output to recompute the expected values of the observed data (values of $\hat{R}$ and $\hat{B}$ for the example in Sec. III). If these are in good agreement with the experimental values then the network outputs can be regarded as reliable. Note that since the evaluation of the expected values of the data variables is often a fast process, the additional computational overhead in doing this cross check is usually small.

It should also be noted that for any given input vector, the network will produce a unique output vector, so that the network can represent one-to-one and many-to-one mappings, but it cannot represent multivalued mappings.

It is therefore important that the training data represents a functional mapping, and in the example in Sec. III this issue had to be addressed, and was solved by excluding a region of the output data space.

The effect of errors Δx_k in the data points x_k is one important topic which has not so far been raised in this article. These errors lead to uncertainties Δy_i in the fit parameters y_i:

$$\Delta y_i = \sum_{k=1}^{M} \frac{\partial y_i}{\partial x_k} \Delta x_k . \tag{11}$$

Squaring and averaging this equation over a number of measurements of the same quantities gives a relationship between the error matrix of the outputs $\langle \Delta y_i \Delta y_j \rangle$ and the error matrix associated with the measurements $\langle \Delta x_k \Delta x_l \rangle$:

$$\langle \Delta y_i \Delta y_j \rangle = \sum_{k=1}^{M} \sum_{l=1}^{M} \frac{\partial y_i}{\partial x_k} \frac{\partial y_j}{\partial x_l} \langle \Delta x_k \Delta x_l \rangle . \tag{12}$$

The partial derivatives of y_i with respect to x_k are easily obtained for the neural network mapping using the same error backpropagation procedure which is used in the network training.[3] The mapping is nonlinear, and so these partial derivatives must be recomputed whenever new data are presented to the network. Thus, given the measurements' error matrix, it is straightforward to calculate the error matrix for the parameters. It should be added that the neural network mapping itself contributes an additional error (hopefully small) which should be added in quadrature to the errors obtained from Eq. (12).

Finally, it should be emphasized that the techniques described in this article can be applied to a wide range of curve fitting problems, including much more complex situations than the single Gaussian example described in this article. Further applications of neural networks in the field of tokamak data processing are described in Refs. 1, 2, 9, and 10.

ACKNOWLEDGMENTS

It is a pleasure to thank Paddy Corolan for bringing the COMPASS spectroscopy problem to our attention and Ben Sidle for supplying the data.

[1] C. M. Bishop, C. M. Roach, M. G. von Hellerman, and P. R. Thomas (in preparation).
[2] C. M. Bishop, I. G. D. Strachan, J. O'Rourke, G. P. Maddison, and P. R. Thomas (in preparation).
[3] C. M. Bishop, Rev. Sci. Instrum. (to be published).
[4] K. Hornik, M. Stinchcombe, and H. White, Neural Networks **2**, 359 (1989).
[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams *Parallel Distributed Processing* (MIT, Cambridge, 1986), Vol. 1, Chap. 8.
[6] H. P. Furth, Scientific Am. **241**, 39 (1979).
[7] P. G. Carolan, C. A. Bunting and R. A. Bamford, from Proceedings of IOP Plasma Physics Conference at the University of Essex, Colchester, 1991.
[8] M. F. F. Nave and J. A. Wesson, Nucl. Fusion **30**, 2575 (1990).
[9] C. M. Bishop, P. M. Cox, P. S. Haynes, C. M. Roach, M. E. U. Smith, T. N. Todd, and D. L. Trotman *Proceedings of 'NCM'91: Applications of Neural Networks* (Springer, 1992, to be published).
[10] J. B. Lister and H. Schnurrenberger, Nuclear Fusion **31**, 1291 (1991).