FUNCTION APPROXIMATION WITH MULTILAYERED PERCEPTRONS USING L1 CRITERION

H.C. Ong and S.H. Quah
School of Mathematical Sciences,
Universiti Sains Malaysia, 11800 USM Penang, Malaysia.

Abstract

The least absolute error criterion is more robust and less easily affected by noise compared to the least squares error criterion. The objective of this study is to present an absolute error criterion for the sigmoidal backpropagation rather than the usual least squares error criterion. We use a 2-stage algorithm for non-linear $L1$ optimization by Madsen, Hegelund and Hansen (1991) to obtain the optimum result. This is a combination of a first order method that approximates the solution by successive linear programming and a quasi-Newton method using approximate second order information to solve the problem. To validate the performance and efficiency of the 2-stage L1 algorithm, a comparison is made on the error made in both the 2 stage L1 algorithm and the least squares error algorithm.

Introduction

The $L2$ criterion approach has been commonly used in functional approximation and generalization in the error backpropagation algorithm. We present an absolute error criterion for the sigmoidal backpropagation rather than the usual least squares error criterion. The focus in the study is on the single hidden layer multilayer perceptron but the implementation may be extended to include two or more hidden layers. Hornik, Stinchcombe and White (1989), Cybenko(1989) and Funahashi (1989) showed that it is sufficient to use a single hidden layered MLP in universal approximations. The backpropagation algorithm as a steepest descent approach is too slow for many applications unless some form of acceleration of learning rate or second order information is used. However, the sigmoidal activation function enables the error function to be differentiable. This is why the error function is incorporated into the algorithm by Hald and Madsen (1985) to minimize the sum of absolute values of a set of non-linear functions. This is a combination of a first order method that approximates the solution by successive linear programming and a quasi-Newton method using approximate second order information to solve the system of non-linear equations arising from the necessary first order conditions at a solution. The latter is intended to be used only in the final stage of the iteration but several switches between the two methods may take place.

Since the time of Gauss, it has been generally accepted that $L2$ methods of combining observations by minimizing the sum of squared errors have significant computational advantages over earlier $L1$ methods based on the minimization of the sum of absolute errors advocated by Boscovich, Laplace and others. In neural network applications, the easily differentiable sum of squared errors has made the $L2$ error criterion a natural choice. However, $L1$ methods are known to have significant robustness advantages over $L2$ methods in many applications. Neural network learning with $L1$ criteria ought to make a network behave closer to the actual function to be approximated, regardless of the class of activation functions. This means that the network should be robust to the unexpected, irregular training sequence and there is a possibility of avoiding a local minimum in the convergence. Thus, the convergence rate is improved because the influence of an incorrect sample is reduced.

The Error Backpropagation Function

An MLP with a single hidden layer is shown in Figure 1. There are I neurons in the input layer, J neurons in the hidden layer and K neurons in the output layer. K is normally taken to be one for the case of functional approximation. The interconnection weights from the input to the hidden layer are denoted by $\{w_{ij}\}$ while those from the hidden layer to the output are denoted by $\{u_{jk}\}$. The sigmoidal activation function for the hidden and output layers are $h(-)$ and $g(-)$ respectively. Each exemplar vector $x^{(q)}$ is mapped into an output $z^{(q)}$ from the network as compared to the target output $t^{(q)}$, where $q = 1, 2, \cdots, Q$ is the number of exemplars or training sets.

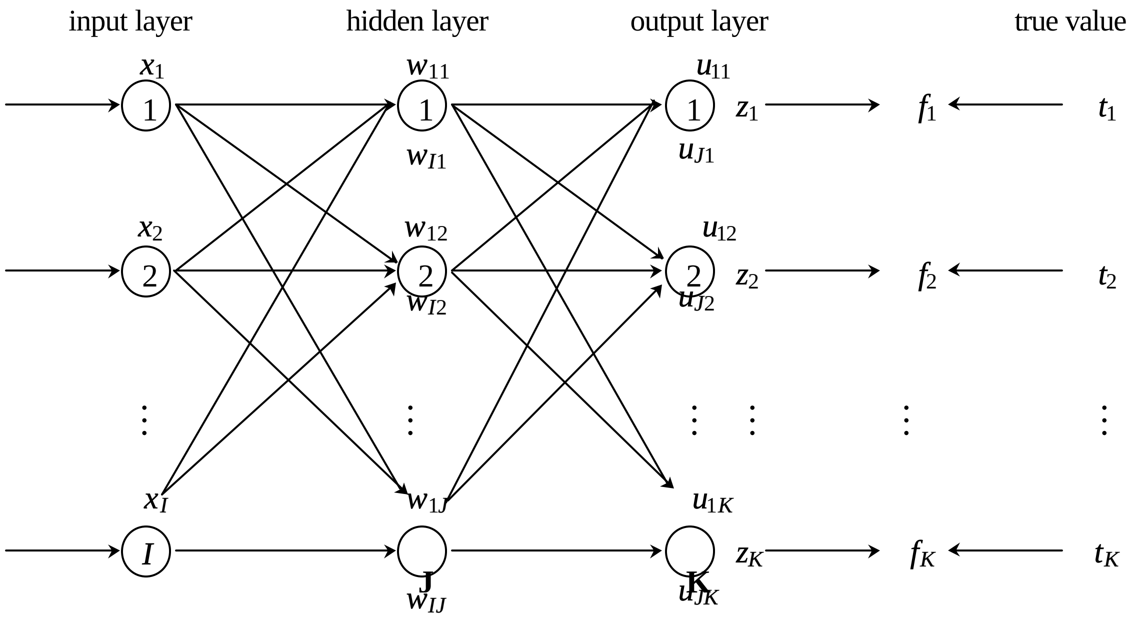


Figure 1 Single hidden layer neural network for training.

Let $r_j = \sum_{i=1}^{I} w_{ij} x_i$ and $y_j = h\left(\sum_{i=1}^{I} w_{ij} x_i\right)$.

Let $s_k = \sum_{j=1}^{J} u_{jk} y_j$ and $z_k = g\left(\sum_{j=1}^{J} u_{jk} y_j\right)$ for $h(r_j) = \dfrac{1}{1+\exp(-a_1 r_j + b_1)}$ and

$g(s_k) = \dfrac{1}{1+\exp(-a_2 s_k + b_2)}$ where a_1, a_2 are rate factors and b_1, b_2 are biases.

The absolute error criterion is given by $F(x) = E(w,u) = \sum_{q=1}^{Q} \left| t^{(q)} - z^{(q)} \right|$ while the error backpropagation function is

$$f^{(q)} = t^{(q)} - z^{(q)} = t_k^{(q)} - g\left(\sum_{j=1}^{J} u_{jk} y_j\right) = t_k^{(q)} - g\left(\sum_{j=1}^{J} u_{jk}\, h\left(\sum_{i=1}^{I} w_{ij} x_i\right)\right).$$

The Two Stage $L1$ Algorithm

The function to be minimized is

$$F(\underline{x}) = E(\underline{w}, \underline{u})$$
$$= \sum_{q=1}^{Q} \left| t^{(q)} - z^{(q)} \right| \text{ where } q = 1, 2, \cdots, Q \text{ are the number of exemplars}$$
$$= \sum_{q=1}^{Q} \left| f_q \right|.$$

Here, the vector of unknown parameters are the weights in the hidden layer, w_{ij}, and weights in the output layer, u_{jk}, in which

$$(\underline{x}) = (\underline{w}, \underline{u}) \in R^N \quad \text{and} \quad N = I \times J + J \times K,$$

where I, J and K are the total number of nodes in the input, hidden and output layers respectively and i, j and k are the corresponding nodes numbers.

In a linear $L1$ problem (the error function f_q linear), two efficient, basically different approaches exist. They are the simplex linear programming method described by Barrodale & Roberts(1978) and the direct descent method of Bartels, Conn & Sinclair (1978). In the case of the nonlinear $L1$ problem, an iterative procedure must be used.

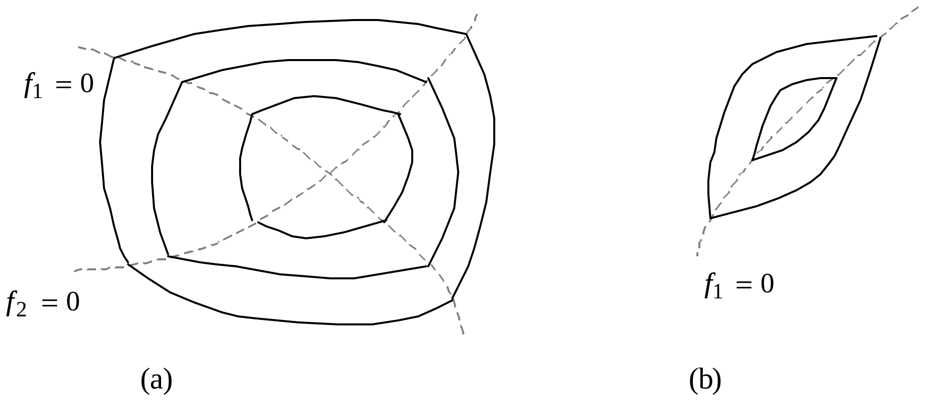


Figure 2 Two dimensional level curves for $L1$ objective function, F.

Notice that F has sharp edges shown as dotted curves in Figure 2 where one of the functions, f equals zero. The optimization method used is that for unconstrained 2-stage minimization of the $L1$ norm of a vector function (Madsen, Hegelund & Hansen, 1991). Figure 2 shows two different two-dimensional unconstrained problems. In Figure 2(a), two functions equal zero at the solution while in the case of Figure 2(b) only one function does so. The solutions are usually situated at an edge, that is, at a point where one or more of the nonlinear functions equal zero. At the edge, the function is non-differentiable and thus ordinary nonlinear programming methods are not efficient for obtaining the solution. However, directional derivatives exist everywhere in any direction and this can be utilized to obtain a fast rate of final convergence.

In Figure 2(a), a quadratic rate of convergence can be obtained by applying Newton's Method to solve the nonlinear equations $f_i = 0,\ i = 1, 2$. This requires only first derivatives. However, in Figure 2(b), the linear approximations do not characterize a minimum at the edge. Some second order information is needed to define the position of the minimum along the edge and, therefore, also obtaining a fast rate of convergence to the minimum. In the algorithm, we approximate the second order information on the basis of gradients which are assumed to be available.

The active set is defined as $Z(\underline{x}) = \{i : f_i(\underline{x}) = 0\}$.

Because of continuity $Z(\underline{y}) \subseteq Z(\underline{x})$ for $\underline{y}$ in a neighbourhood of $\underline{x}$ and, hence, the objective function can be locally split into a differentiable and a non-differentiable part:

$$F(\underline{y}) = \sum_{i \notin Z(\underline{x})} \left| f_i(\underline{y}) \right| + \sum_{i \in Z(\underline{x})} \left| f_i(\underline{y}) \right|.$$

Although F is non-differentiable, the directional derivatives exist at any point $\underline{x}$ for any direction $\underline{e}$. The first order correction in F at $\underline{x}$ is expressed as

$$DF(\underline{x};\ d) = \| \underline{d} \| F'_{\underline{e}}(\underline{x}), \qquad \text{where } \underline{e} = \frac{\underline{d}}{\| \underline{d} \|}.$$

Since the functions are twice differentiable, it follows that, for $\alpha > 0$ sufficiently small, we have $F(\underline{x} + \alpha \underline{x}) = F(\underline{x}) + \alpha F'_{\underline{e}}(\underline{x}) + O(\alpha^2)$, and therefore, we obtain (by a continuity argument) that

$$F(\underline{x} + \underline{d}) = F(\underline{x}) + DF(\underline{x}; d) + O\left(\| \underline{d} \|^2 \right).$$

For small values of $\| \underline{d} \|$, $\Delta F(\underline{x};\ \alpha \underline{d}) = \alpha \Delta F(\underline{x};\ \underline{d}), \quad 0 < \alpha \le 1$. If we linearize each function f at $\underline{x}$, then for small changes of $\| \underline{d} \|$, the linearized $L1$ function (with $Z = Z(\underline{x})$) can be expressed as

$$\begin{aligned} \overline{F}(\underline{x};\ \underline{d}) &= \sum_{i \notin Z} \left| f_i + f_i'(\underline{x})^T \underline{d} \right| + \sum_{i \in Z} \left| f_i(\underline{x}) + f_i'(\underline{x})^T \underline{d} \right| \\ &= F(\underline{x}) + DF(\underline{x};\ \underline{d}) \end{aligned}$$

since $f_i(\underline{x}) = 0,\ i \in Z$ implies $\sum_{i \notin Z} \left| f_i(\underline{x}) \right| = F(\underline{x})$. The signs of the functions are described through

$$s_i(\underline{x}) \equiv \begin{cases} +1, & \text{for } f_i(\underline{x}) > 0 \\ 0, & \text{for } f_i(\underline{x}) = 0 \\ -1, & \text{for } f_i(\underline{x}) < 0. \end{cases}$$

The L_1 Lagrangian function, with active set Z and signs s_i of the inactive functions is

$$L(\underline{x},\ \underline{d}) = \sum_{i \notin Z} s_i f_i(\underline{x}) + \sum_{i \in Z} d_i f_i(\underline{x}),$$

where $\sum_{i \notin Z} s_i f_i(\underline{x})$ is the differential part of the objective function. Thus, a stationary point is characterized by a nonlinear system of equations

$$L'_x(\underline{x},\ \underline{d}) = 0,\ \left| d_i \right| \le 1 \quad \text{and} \quad f_i(\underline{x}) = 0,\ i \in Z \tag{1}$$

where $L'_x(\underline{x}, \underline{d})$ denotes the derivatives with respect to $\underline{x}$ and $d_i = 0$ for $i \notin Z$. Therefore, we consider the above nonlinear system of equations as a set of $(N + s)$ real equations in $(N + s)$ unknowns where s is the number of elements at $Z(\underline{x})$.

The 2-stage L1 optimization algorithm is a combination of a first order method that approximates the solution by successive linear programming and a quasi-Newton method using approximate second order information to solve the system of nonlinear equations arising from the necessary first order conditions at a solution. The latter is intended to be used only in the final stages of the iteration, but several switches between the two methods may take place.

The algorithm to be described consists of **four** parts (Hald & Madsen, 1985): The Stage 1 iteration, the Stage 2 iteration, conditions for switching to Stage 2 and causes for switching back to Stage 1. The iteration always starts in Stage 1.

(i) The **Stage 1 iteration** is a first order algorithm for linearly unconstrained minimax optimization. At the *k*th stage of the iteration, with an approximation, $\underline{x}_k$, of the solution and a local bound, L_k, the gradient information at $\underline{x}_k$ is used to find a better approximation $\underline{x}_{k+1}$. Therefore, we find the increment as a solution $\underline{h}_k$ of the linearized $L1$ problem: Minimize (as a function of $\underline{h}$) $\overline{F}(\underline{x}_k;\underline{h}) = \sum_{i=1}^{m} \left| f_i(\underline{x}_k) + f_i'(\underline{x}_k)^T \underline{h} \right|$ subject to $\|\underline{h}\|_\infty \le L_k$. The linear problem is solved very efficiently using an implementation of the algorithm of Bartels, Conn & Sinclair (1978). The point $(\underline{x}_k + \underline{h}_k)$ is accepted as the next iteration point provided that the decrease in F exceeds a small multiple of the decrease predicted by the linear approximation. Therefore, $\underline{x}_{k+1} = \underline{x}_k + \underline{h}_k$ provided that $F(\underline{x}_k) - F(\underline{x}_k + \underline{h}_k) \ge r(F(\underline{x}_k) - \overline{F}(\underline{x}_k;\ \underline{h}_k))$ with $r = 0.01$. If this inequality does not hold, we let $\underline{x}_{k+1} = \underline{x}_k$.
The choice of L_{k+1} is also based on the inequality with another value of r. If the decrease in F is rather poor we wish the bound to be smaller. If the decrease is very close to the decrease predicted by the linear approximation, we wish the bound to be increased. In fact, if the inequality is not true with $r = 0.25$, we let $L_{k+1} = 0.25\|\underline{h}_k\|_\infty$; if it is true with $r = 0.75$, we let $L_{k+1} = 2\|\underline{h}_k\|_\infty$. In all other cases, we let $L_{k+1} = \|\underline{h}_k\|_\infty$. These rules ensure that L is decreased, when $\underline{x}_{k+1} = \underline{x}_k$.

(ii) **Conditions for switching to Stage 2** is set when the active set (or the solution set of the latest linear problem) have been stabilized or constant for a certain number of consecutive Stage 1 iterations. The Stage 2 iteration is introduced in order to speed up the final rate of convergence in cases where quadratic convergence is not obtained using the stage 1 iteration. It is required that $Z_k = Z_{k-j_1} = ... = Z_{k-j_n}$, that is, the active set must have been constant for $(n+1)$ consecutive stage 1 iterations and also the first order multiplier estimates are in the prescribed range $-1 \le (\underline{d}_k)_i \le 1, i \in Z_k$, where $(\underline{d}_k)_i$ denotes the i th component of $\underline{d}_k$.

(iii) In the **Stage 2 iteration**, a quasi-Newton method with BFGS (Broyden-Fletcher-Shanno-Goldfarb) updating formula is used for solving the stationary point of the objective function. It is assumed that an active set Z approximating the corresponding quantities in the solution has been determined before Stage 2 is entered. Here, the values of $\underline{x}_k$, $\underline{d}_k$, and B_k obtained in Stage 1 are used as starting values. The BFGS formula updates the approximate Hessian matrix containing second order information. The values obtained in Stage 1 are used as starting values in Stage 2. The implementation solves the equation using Gaussian elimination on the whole system.

(iv) The **conditions for switching back to Stage 1** are set up to ensure that if a Stage 2 iteration is started with an improper active set, then a switch back to Stage 1 will take place. It is required that $\left|(\underline{\boldsymbol{d}}_k)_i\right| \le 1, \quad i \in Z_k$ hold for every iteration k and that there is no change in sign for the inactive function during the Stage 2 iteration. $(\underline{\boldsymbol{d}}_k)_i$ denotes the ith component of $\underline{\boldsymbol{d}}_k$. Finally, it is required that the residuals corresponding to the left-hand side of Eq.(1) decrease in every iteration in the following sense:
$\| r(\underline{\boldsymbol{x}}_{k+1}, \ \underline{\boldsymbol{d}}_{k+1}) \| \le \boldsymbol{h} \| r(\underline{\boldsymbol{x}}_k, \underline{\boldsymbol{d}}_k) \|$ where $0 < \boldsymbol{h} < 1$ and $r(\underline{\boldsymbol{x}}_k, \underline{\boldsymbol{d}}_k)$ denotes the left-hand side of Eq.(1) at the kth stage of the iteration.

<u>Simulation Results using 5 Non-linear Functions</u>

The performance of the 2-stage $L1$ error backpropagation algorithm was tested using five non-linear functions. These functions were scaled so that the standard deviation is 1 (for a large regular grid with 2 500 points on $[0,1]^2$), and translated to make the range nonnegative (Hwang *et al.*, 1994). This facilitates performance comparisons across the different functions. The abscissa values $\{(x_{\ell 1}, x_{\ell 2})\}$ were generated as uniform random variates on $[0,1]$ which are independent of each other. We generated 225 points $\{(x_{\ell 1}, x_{\ell 2})\}$ of abscissa values, and used this same set for experiments with all the five functions, thus eliminating an unnecessary variability component in the simulation. In other words, $y_\ell^{(j)} = g^{(j)}(x_{\ell 1}, x_{\ell 2}),$ for $\ell = 1, 2, \cdots, 225$ and $j = 1, \cdots, 5.$

Figure 3, which is generated using MATLAB, gives 3-dimensional perspective plots of the five functions using a meshgrid of the 225 points. The functions are as follows:

(i) *Simple Interaction Function:* $g^{(1)}(x_1, x_2) = 10.391\,((x_1 - .4)\cdot(x_2 - .6) + .36)$.

(ii) *Radial Function :*

$$g^{(2)}(x_1, x_2) = 24.234\,(r^2(.75 - r^2)), r^2 = (x_1 - .5)^2 + (x_2 - .5)^2 .$$

(iii) *Harmonic Function:* $g^{(3)}(x_1, x_2) = 42.659\,((2 + x_1)/20 + \mathrm{Re}(z^5))$,
where $z = x_1 + ix_2 - .5(1+i)$, or equivalently, with $\tilde{x}_1 = x_1 - .5,\ \tilde{x}_2 = x_2 - .5,$

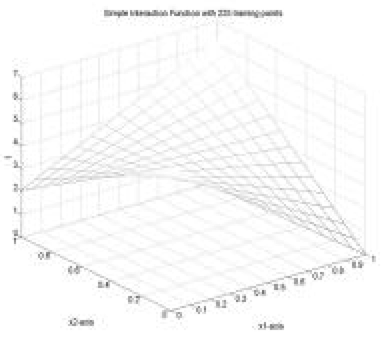
$$g^{(3)}(x_1, x_2) = 42.659\ (.1 + \tilde{x}_1(.05 + \tilde{x}_1^4 - 10\tilde{x}_1^2\,\tilde{x}_2^2 + 5\tilde{x}_2^4)) .$$

(iv) *Additive Function:* $g^{(4)}(x_1, x_2) = 1.3356\,(1.5(1 - x_1) + e^{2x_1 - 1}$

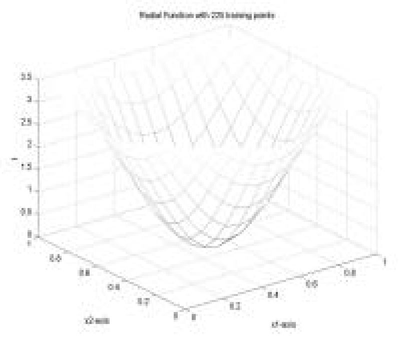
$$\sin(3p(x_1 - .6)^2) + e^{3(x_2 - .5)} \sin(4p(x_2 - .9)^2)$$

(v) *Complicated Interaction Function:*

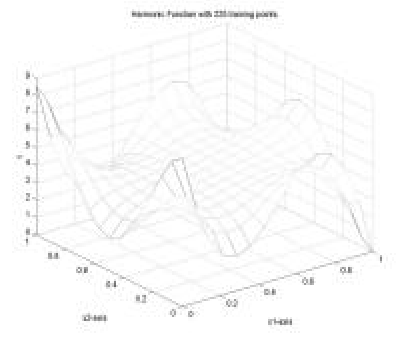
$$g^{(5)}(x_1, x_2) = 1.9\,\left(1.35 + e^{x_1} \sin(13(x_1 - .6)^2)\cdot e^{-x_2} \sin(7x_2)\right).$$



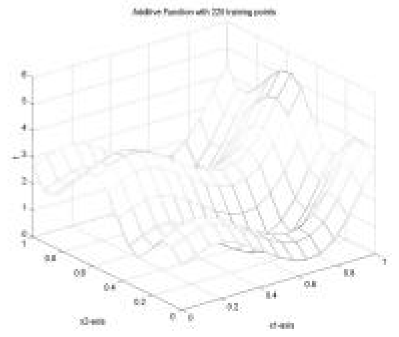
(a) $g^{(1)} =$ simple interaction



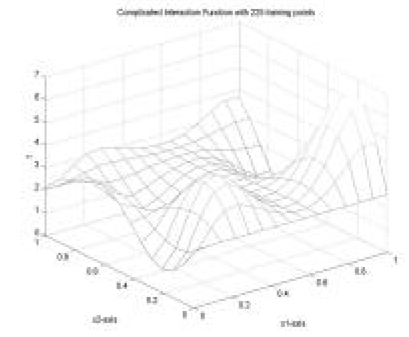
(b) $g^{(2)} =$ radial



(c) $g^{(3)} =$ harmonic



(d) $g^{(4)} =$ additive



(e) $g^{(5)} =$ complicated interaction

Figure 3. Plots of functions $g^{(1)}, \cdots, g^{(5)}$
.

The 225 points generated from the five non-linear functions as shown in Figure 3 are then trained using the error backpropagation function using 8 nodes in a single hidden layer. The same starting points were used for all the functions. The weights or parameters generated are fed back into the error backpropagation function and the resultant output function is then generated to produce the approximated functions.

Table 1 Comparison of accuracy determined by the FVU error measure on two stopping criterions using 8 hidden nodes.

	FVU error measure with stopping criterion	
	$\|h\|_k < 10^{-5}\|x\|_k$	$\|h\|_k < 10^{-8}\|x\|_k$
$g^{(1)}$ = simple interaction	0.00599438	0.00362016
$g^{(2)}$ = radial	0.02570136	0.02157959
$g^{(3)}$ = harmonic	0.55727138	0.54078910
$g^{(4)}$ = additive	0.48704131	0.02072139
$g^{(5)}$ = complicated interaction	0.20382973	0.19519377

As in the paper by Hwang *et al.* (1994), the fraction of variance unexplained (FVU) on the test set is used for the comparison of the accuracy in the simulations of the five non-linear functions. It is defined as

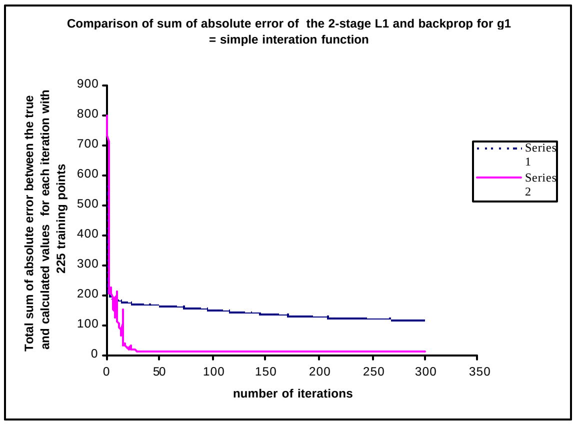
$$\text{FVU} = \frac{\sum_{q=1}^{Q}(\hat{g}(x_q) - g(x_q))^2}{\sum_{q=1}^{Q}(g(x_q) - \bar{g}(x))^2}$$

where $g = g^{(j)}, \quad j = 1, 2, \cdots, 5$ and $\bar{g}(x) = \frac{1}{Q}\sum_{q=1}^{Q} g(x_q).$

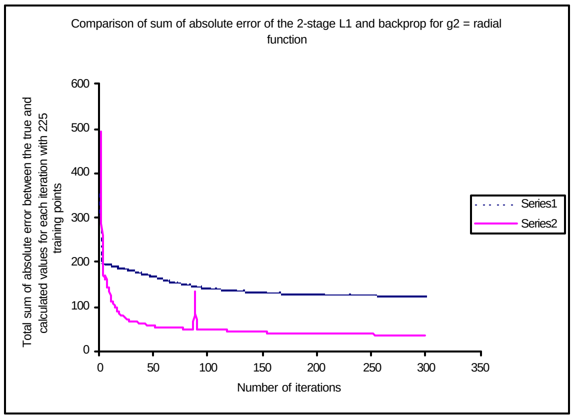
Note that the FVU is proportional to the commonly used mean square error (MSE) and a more intuitive value is its square root. The accuracy is determined by the FVU error measure of the 225 independent test data above. We show in Table 1, a comparison of the FVU error measure with two different stopping criterion on all the five non-linear functions. It is obvious for all the five functions that as the stopping criterion becomes more accurate, the FVU error measure becomes smaller. An obvious result also, is that the harmonic function is the most difficult to approximate in the sense of needing the most neurons for a given approximation error bound.

Comparison Between the Performance of the 2-stage $L1$ Algorithm and the Sum of Least Squares Error Algorithm

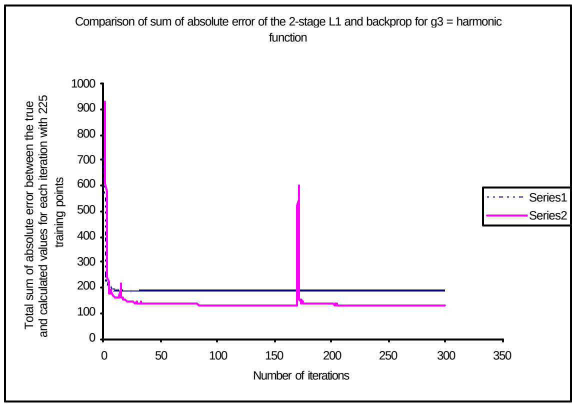
To further validate the performance and efficiency of the 2-stage $L1$ algorithm, a comparison is made between the 2-stage $L1$ algorithm and the sum of least squares error. The total sum of the absolute errors between the true and calculated values for each iteration with 225 training points is made for both the 2-stage $L1$ and the least squares error backpropagation algorithms on the five non-linear functions $g^{(1)}$ to $g^{(5)}$ described. The error values are calculated for the first 300 iterations in all the five functions and a comparison is shown in Figures 4(a)-(e). From the five graphs, it is obvious that the 2-stage $L1$ algorithm (bold line – series 2) outperform the sum of least squares error backpropagation (fine line – series 1). The few spikes in the error is due to the switches between the two stages in the 2-stage $L1$ algorithm.



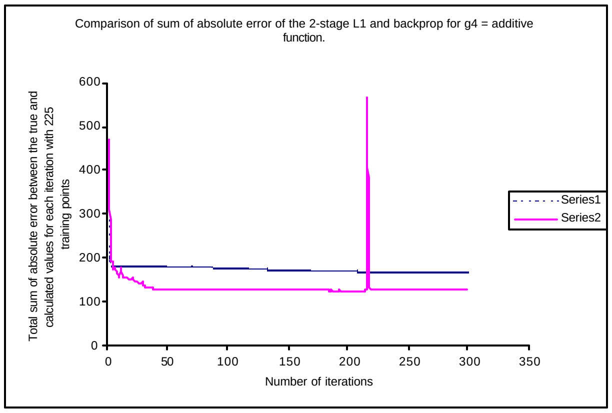
(a) $g^{(1)}$ = simple interaction



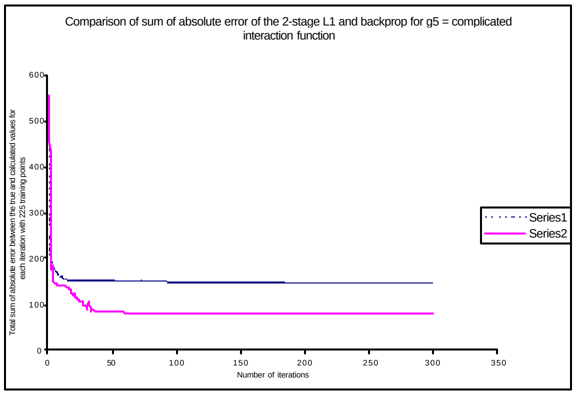
(b) $g^{(2)}$ = radial



(c) $g^{(3)}$ = harmonic



(d) $g^{(4)}$ = additive



(e) $g^{(5)}$ = complicated interaction

Figure 4 Comparison of the sum of absolute errors of the 2-stage $L1$ (bold line – series 2) and least square backpropagation (fine line – series 1) for functions $g^{(1)}$ to $g^{(5)}$ in the first 300 iterations.

Summary

We have reviewed the error backpropagation function and incorporated the function into a 2 stage $L1$ optimization. We have thus successfully implemented an error backpropagation algorithm using the $L1$ criterion and tested the performance on five different nonlinear functions. The work here provides an alternative to the usual $L2$ criterion for the investigation of the properties of other architectures and learning procedures which involve substituting the least square criterion with the least absolute criterion.

The single hidden layer MLP presented here permits generalization and this can be done in a number of ways. The activation function may change from layer to layer (or from unit to unit). We can replace the simple linearity at each unit (i.e., the $\sum_{i=1}^{I} w_{ij} x_i$ and $\sum_{j=1}^{J} u_{jk} y_j$) by some more complicated function of the x_{ij} and y_{jk}. The architecture may be altered to allow for different links between units of different layers (and perhaps also of the same layer). The least absolute $L1$ error criterion is preferable to other Lp criteria $(p > 1)$ when applied to real problems where the observed data used in learning include some error. It provides an alternative to the usual $L2$ criterion for the investigation of the properties of other architectures and learning procedures like, for example, the radial basis function, which involves substituting the least squares criterion with the least absolute criterion.

References

Barrodale, I & Roberts, F.D.K. An efficient algorithm for discrete $L1$ linear approximation with linear constraints. *SIAM Journal on Numerical Analysis* **15**(1978), 603–611.

Bartels, R.H., Conn, A.R. and Sinclair, J.W., Minimization techniques for piecewise differentiable functions: the $L1$ solution to an overdetermined linear system, *SIAM Journal on Numerical Analysis* **15**(1978), 224-241.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems.* **2**(1989), 303-314.

Funahashi, K., On the approximate realization of continuous mappings by neural networks, *Neural Networks* **2**(1989) 183-192.

Hald, J. and Madsen, K., Combined LP and quasi-newton methods for non-linear $L1$ optimization, *SIAM Journal on Numerical Analysis* **22**(1)(1985), 68-80.

Hornik, K., Stinchcombe, M. and White, H., Multilayer feedforward networks are universal approximators, *Neural Networks* **2**(1989), 359-366.

Hwang, J.N., Lay, S.R., Maechler, M., Martin, R.D. and Schimert, J., Regression modeling in backpropagation and projection pursuit learning, *IEEE Transactions on Neural Network.* **5**(3)(1994), 342-353.

Madsen, K., Hegelund, P. and Hansen, P.C., Robust c subroutines for non-linear optimization, Institute for Numerical Analysis, Technical University of Denmark, Report N1-91-03, 1991.