CLM-P880

CLM-P880

Curvature-Driven Smoothing in Backpropagation Neural Networks

C. M. Bishop

CULHAM
LABORATORY

AEA TECHNOLOGY

Curvature-Driven Smoothing in Backpropagation Neural Networks

Chris M Bishop
Theory and Computing Division
Culham Laboratory
AEA Technology
(Euratom/UKAEA Fusion Association)
Tel: 0235 - 21840 ext 3502
Fax: 0235 - 463682

Abstract

The standard backpropagation learning algorithm for feedforward networks aims to minimise the mean square error defined over a set of training data. This form of error measure can lead to the problem of over-fitting in which the network stores individual data points from the training set, but fails to generalise satisfactorily for new data points. In this paper we propose a modified error measure which can reduce the tendency to overfit and whose properties can be controlled by a single scalar parameter. The new error measure depends both on the function generated by the network and on its derivatives. A new learning algorithm is derived which can be used to minimise such error measures.

Keywords

Backpropagation, Curvature, Feed-Forward, Generalisation, Hidden unit, Learning algorithm, Neural network, Smoothing.

Acknowledgements

The author would like to thank I F Croall, K D Horton and I G D Strachan for a number of useful discussions relating to this work.

Running title: Curvature-Driven Smoothing

1 Introduction

Feed-forward neural networks, trained by error backpropagation, form one of the most widely used neural network architectures. If the network has hidden neurons, and non-linear activation functions, it can generate a large class of non-linear continuous mappings between multidimensional spaces (Funahashi, 1989; Hornik et al., 1989). The standard learning algorithm (Rumelhart et al., 1986) minimises an error measure which is the sum, over a set of training data, of the squares of errors for the output neurons. This form of error measure can lead to the problem of over-fitting of the data (sometimes called over-generalisation), in which the network 'stores' individual data points, but fails to generalise satisfactorily for inputs not included in the training set. An analogous problem arises when curve-fitting using high order polynomials (Tikhonov and Arsenin, 1977; Denker et al, 1987). Again, this problem is related to the minimisation of a mean-square error measure.

One approach for reducing this effect is to include a sufficiently large number of examples in the training set. This may not always be practical, however, particularly for problems with many degrees of freedom.

A closely related issue concerns the number of neurons in the network. The number of input and output neurons is generally determined by the dimensionality of the data itself. However, there exists no satisfactory theoretical basis for determining the number of hidden units, which must often be decided by trial and error. If the network has too few hidden neurons, the class of functions which it can generate is too restricted and it is unable to achieve the desired accuracy. Increasing the number of hidden neurons can, however, lead to the problem of over-fitting.

In this paper we propose the use of a modified error measure designed to reduce the tendency to over-fit, even for a network with many hidden neurons. To minimise this error measure a new learning algorthim is derived which generalises the standard back-propagation procedure. The modified error measure is described in section 2, and the learning algorithm is derived in section 3. Finally some conclusions are presented in section 4.

2 Curvature-Driven Smoothing

To begin with, consider mappings of a single variable x to a single variable y, so that the network has one input and one output neuron. The network has a feed-forward architecture in which each neuron generates a non-linear function of the weighted sum of its inputs:

$$z_i = f(\sum_j w_{ij} z_j), \tag{1}$$

where z_j is the activation of the j^{th} neuron, w_{ij} is the connection weight between neurons i and j, and the function f is taken to be the standard sigmoid:

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

There also exists a threshold for each neuron. Since, however, these are equivalent to weights from an extra neuron whose output is permanently set to $+1$, they are contained in the formalism of Eq.(1) and we need consider them no further. By construction, the function $y(x)$ will be continuous, single-valued and differentiable.

Suppose we have a set of data points $\{x_p, t_p\}$, $p = 1, ..., P$, where t_p is the target value for y corresponding to $x = x_p$. The standard learning algorithm minimises the error measure

$$E^{\text{S}} = \frac{1}{2}\sum_{\{p\}}(y_p - t_p)^2 \tag{3}$$

where $y_p = y(x_p)$. The use of this error measure can result in the network function $y(x)$ over-fitting the data points, as shown schematically in Figure 1. We now seek to modify the error measure so as to generate smoother functions $y(x)$, as indicated in Figure 2. To achieve this we add to the standard error measure a regularising term which depends on the geometrical properties of the network function $y(x)$:

$$E = E^{\text{S}} + \lambda E^{\text{C}} \tag{4}$$

$$E^{\text{C}} = \frac{1}{2}\int_a^b \kappa^2 dx \tag{5}$$

where κ is the curvature of the line $y = y(x)$, and the interval (a, b) spans the range of values of $\{x_p\}$. Smoother curves will have a small value of E^{C} for a given internal (a, b). Note the use of κ^2 in Eq.(5) rather than κ, since κ carries a sign. Similar regularising functions have been discussed in the context of curve-fitting (Tikhonov and Arsenin, 1977; Denker et al., 1987). In terms of the network function $y(x)$ we can write (Korn and Korn, 1968)

$$E^{\text{C}} = \frac{1}{2}\int_a^b \frac{(y'')^2}{[1+(y')^2]^3} dx \tag{6}$$

where primes denote d/dx. We now replace the integral in Eq.(6) by a sum over a discrete set of points $\{x_n\}$

$$E^{\mathrm{C}} = \frac{1}{2}\sum_{\{n\}} \frac{(y_n'')^2}{[1+(y_n')^2]^3}\Delta x_n \tag{7}$$

where $y_n = y(x_n)$, and $\Delta x_n = x_n - x_{n-1}$. It will often be convenient to choose the points $\{x_n\}$ to coincide with the $\{x_p\}$ since this will make use of values for the neuron activations which need to be calculated anyway for the minimisation of E^{S}. If, however, the data points are too sparse in some regions for sufficient accuracy to be obtained, it is straightforward to include extra values of x. Equally, it may be acceptable to exclude a proportion of the data points from $\{x_n\}$ and so reduce the training time.

For a network with N input and M output neurons, the curvature term in the error measure can be generalised as follows:

$$E^{\mathrm{C}} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\int_{a_i}^{b_i} \kappa_{ij}^2 dx_i, \tag{8}$$

where $y_j(x_1, ..., x_N)$ is the activation of output neuron j. The curvature of y_j with respect to variations in the input x_i is given by

$$\kappa_{ij}^2 = \frac{(\partial^2 y_j/\partial x_i^2)^2}{\left[1+(\partial y_j/\partial x_i)^2\right]^3}. \tag{9}$$

3 Learning Algorithm

The standard mean-square error measure E^{S} is a function of the synaptic weights w_{ij} through the network function $y(x)$. For a given training set $\{x_p, y_p\}$ the error E^{S} can be minimised by gradient descent using the backpropagation training algorithm (Rumelhart et al., 1986). The curvature term E^{C}, however, contains derivatives of $y(x)$ and so the standard backpropagation procedure does not apply. We now derive a generalised form of backpropagation which can be used to minimise error measures containing derivative terms, and which therefore can be applied to the curvature function considered in the previous section.

As usual, the network is trained by gradient descent so that

$$\Delta w_{ij}(m) = -\eta \left(\frac{\partial E^{\mathrm{S}}}{\partial w_{ij}} + \frac{\partial E^{\mathrm{C}}}{\partial w_{ij}} \right) + \mu \Delta w_{ij}(m-1), \tag{10}$$

where m denotes the training step number, η is the learning rate, and μ is the momentum. The derivative $\partial E^{\mathrm{S}}/\partial w_{ij}$ is calculated using the standard backpropagation algorithm. We now seek a procedure for calculating $\partial E^{\mathrm{C}}/\partial w_{ij}$.

From Eq.(7) we have

$$\frac{\partial E^{\mathrm{C}}}{\partial w_{ij}} = \sum_{\{n\}} \left\{ \frac{(y_n'')}{[1+(y_n')^2]^3} \frac{\partial y_n''}{\partial w_{ij}} - \frac{3(y_n'')^2 y'}{[1+(y_n')^2]^4} \frac{\partial y_n'}{\partial w_{ij}} \right\} \Delta x_n. \tag{11}$$

To calculate the partial derivatives in Eq.(11), we note that w_{ij} and x_n are independent variables, and so we can interchange the order of the derivatives:

$$\frac{\partial}{\partial w_{ij}} \left(\frac{dy}{dx} \right) = \frac{d}{dx} \left(\frac{\partial y}{\partial w_{ij}} \right). \tag{12}$$

We now introduce an 'error' term σ_i for each neuron:

$$\frac{\partial y}{\partial w_{ij}} = \sigma_i z_j, \tag{13}$$

$$\sigma_i = \frac{\partial y}{\partial z_i} z_i(1-z_i), \tag{14}$$

where we have used Eq.(1) together with the relation

$$f' = f(1-f). \tag{15}$$

From the chain rule for partial derivatives it follows that

$$\sigma_i = z_i(1-z_i) \sum_k w_{ki} \sigma_k. \tag{16}$$

Using Eqs.(12) and (13) we can write

$$\frac{\partial y'}{\partial w_{ij}} = z_j \frac{d\sigma_i}{dx} + \sigma_i \frac{dz_j}{dx} \tag{17}$$

where

$$\frac{d\sigma_i}{dx} = z_i(1-z_i)\sum_k w_{ki}\frac{d\sigma_k}{dx} + (1-2z_i)\frac{dz_i}{dx}\sum_k w_{ki}\sigma_k. \tag{18}$$

For the output neuron we have

$$\sigma_o = y(1-y), \tag{19}$$

$$\frac{d\sigma_o}{dx} = (1-2y)\frac{dy}{dx}. \tag{20}$$

The partial derivatives z_i' can be calculated during the forward propagation phase using

$$\frac{dz_i}{dx} = z_i(1-z_i)\sum_j w_{ij}\frac{dz_j}{dx}, \tag{21}$$

which follows from Eqs.(1) and (15). The quantities σ_i, $d\sigma_i/dx$ can then be found by backpropagation using the recursive formulae of Eqs.(16) and (18).

Analogous results for the second derivatives are easily obtained:

$$\frac{\partial y''}{\partial w_{ij}} = z_j\frac{d^2\sigma_i}{dx^2} + 2\frac{d\sigma_i}{dx}\frac{dz_j}{dx} + \sigma_i\frac{d^2z_j}{dx^2}, \tag{22}$$

$$\begin{aligned}\frac{d^2\sigma_i}{dx^2} &= z_i(1-z_i)\sum_k w_{ki}\frac{d^2\sigma_k}{dx^2} \\ &+ 2(1-2z_i)\frac{dz_i}{dx}\sum_k w_{ki}\frac{d\sigma_k}{dx} \\ &+ (1-2z_i)\frac{d^2z_i}{dx^2}\sum_k w_{ki}\sigma_k \\ &- 2\left(\frac{dz_i}{dx}\right)^2\sum_k w_{ki}\sigma_k,\end{aligned} \tag{23}$$

$$\frac{d^2\sigma_o}{dx^2} = (1-2y)\frac{d^2y}{dx^2} - 2\left(\frac{dy}{dx}\right)^2, \tag{24}$$

$$\begin{aligned}\frac{d^2z_i}{dx^2} &= z_i(1-z_i)\sum_j w_{ij}\frac{d^2z_j}{dx^2} \\ &+ z_i(1-z_i)(1-2z_i)\left(\sum_j w_{ij}\frac{dz_j}{dx}\right)^2.\end{aligned} \tag{25}$$

We can now summarise the learning algorithm as follows:

(1) Apply inputs $\{x_n\}$ and forward propagate to generate, layer by layer, the neuron activations z_{jn}, y_n using Eqs.(1) and (2), and the various derivatives dz_{jn}/dx_n etc. using Eqs.(21) and (25).

(2) Compute the error for the output neuron using Eqs.(19), (20) and (24) and back-propagate the errors using Eqs.(16), (18) and (23).

(3) Update the weights using Eqs.(10), (11), (17), and (22).

This algorithm readily generalises to networks having several input and ouput neurons, with a corresponding error measure given by Eq.(8), simply by keeping track of the indices labelling each neuron. For each term in the sum in Eq.(8) we have to consider the dependence of the ouput y_j on the input x_i with all other inputs held fixed. The learning algorithm equations derived above can then be applied to the function $y_j(x_i)$. Many of the heuristic procedures which have been developed to speed up conventional backpropagation can also be applied to this algorthim (Cater, 1987; Dahl, 1987; Hush and Salas, 1988; Jacobs, 1988; Stornetta and Huberman, 1987).

4 Summary and Discussion

In this paper we have proposed a new error measure for feed-forward neural networks which is intended to bias the network in favour of smooth solutions and thereby avoid the problem of over-generalisation. A learning algorithm for the minimisation of this error measure has also been described. This algorithm is also applicable to other error measures which are expressible in terms of the (multivariate) network function and its derivatives. For instance, the smoothing functional

$$\tilde{E} = \int \{y^2 + (y')^2\} dx \tag{26}$$

has been found useful in other contexts (Tikhonov and Arsenin, 1977; Farhat and Bai, 1989). In this case the smoothing term contains only first derivatives of $y(x)$, and so the learning algorithm is simplified.

The greater complexity of the learning algorithm described here, compared with the standard backpropagation procedure, will result in a significant increase in training time which in some situations will make this procedure inappropriate. It should be emphasised, however, that one of the great advantages of feed-forward networks, namely their speed of operation once trained, is unaffected. Detailed results from software simulations of networks using this new algorithm will be described in a subsequent publication.

A major outstanding problem with feed-forward networks is the determination of the number of hidden neurons. With the technique described in this paper it should be possible to use a network with a fixed number of neurons and to control the properties of the solution by varying a single scalar parameter λ instead of having to change the network architecture.

References

Cater J P (1987) Successfully Using Peak Learning Rate of 10 (and greater) in Backpropagation Networks with the Heuristic Learning Algorithm. *Proceedings of the IEEE First International Conference on Neural Networks.* San Diego, CA Vol II 645-651.

Dahl E D (1987) Accelerated Learning using the Generalised Delta Rule. *Proceedings of the IEEE First International Conference on Neural Networks.* San Diego, C A Vol II, 523-530.

Denker J, Schwarz D, Wittner B, Solla S, Howard R, Jackel L & Hopfield J (1987) Large Automatic Learning, Rule Extraction and Generalisation *Complex Systems* **1** 877-922.

Farhat N H, and Fai, B (1989) Echo Inversion and Target Shape Estimation by Neuromorphic Processing *Neural Networks* **2** No 2, 117-125.

Funahashi K (1989) On the Approximate Realisation of Continuous Mappings by Neural Networks. *Neural Networks* **2** No 3, 183-192.

Hornik K, Stinchcombe M & White H (1989) Mulitlayer Feedforward Networks are Universal Approxmations *Neural Networks* **2** No 5, 359-366.

Hush D R & Salas J M (1988) Improving the Learning Rate of Backpropagation with the Gradient Reuse Algorithm. *Proceedings of the IEEE International Conference on Neural Networks* San Diego, CA Vol I, 441-446.

Jacobs R A (1988) Increased Rates of Convergence through Learning Rate Adaptation. *Neural Networks* Vol I, No 4 295-307.

Korn G A & Korn T M (1968) *Mathematical Handbook for Scientists and Engineers* 2nd Ed 564.

Rumelhart D E & McClelland J L (1986) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* Vol 1: Foundations. Cambridge, MA MIT Press.

Stornetta W S & Huberman B A (1987) An Improved Three-Layer, Backpropagation Algorithm *Proceedings of the IEEE First International Conference on Neural Networks.* San Diego, CA Vol II, 637-643.

Tikhonov A N & Arsenin V Y (1977) *Solutions of Ill Posed Problems.* New York, Wiley.

Nomenclature

E	total error measure
E^{C}	curvature error term
E^{S}	sum-of-squares error term
f	sigmoid activation function
M	total number of output neurons
m	training step number
N	total number of input neurons
P	total number of training data points
w_{ij}	synaptic weight from neuron j to neuron i
x	network input variable
x_n	input value for nth training pattern
y	network output variable
z_i	activation of neuron i

Greek Symbols

Δx_n	difference in x between training points $n-1$ and n
η	learning rate
κ	curvature of network function
λ	coefficient of cuvature error term
μ	momentum coefficient
σ_i	error for neuron i
σ_o	error for output neuron

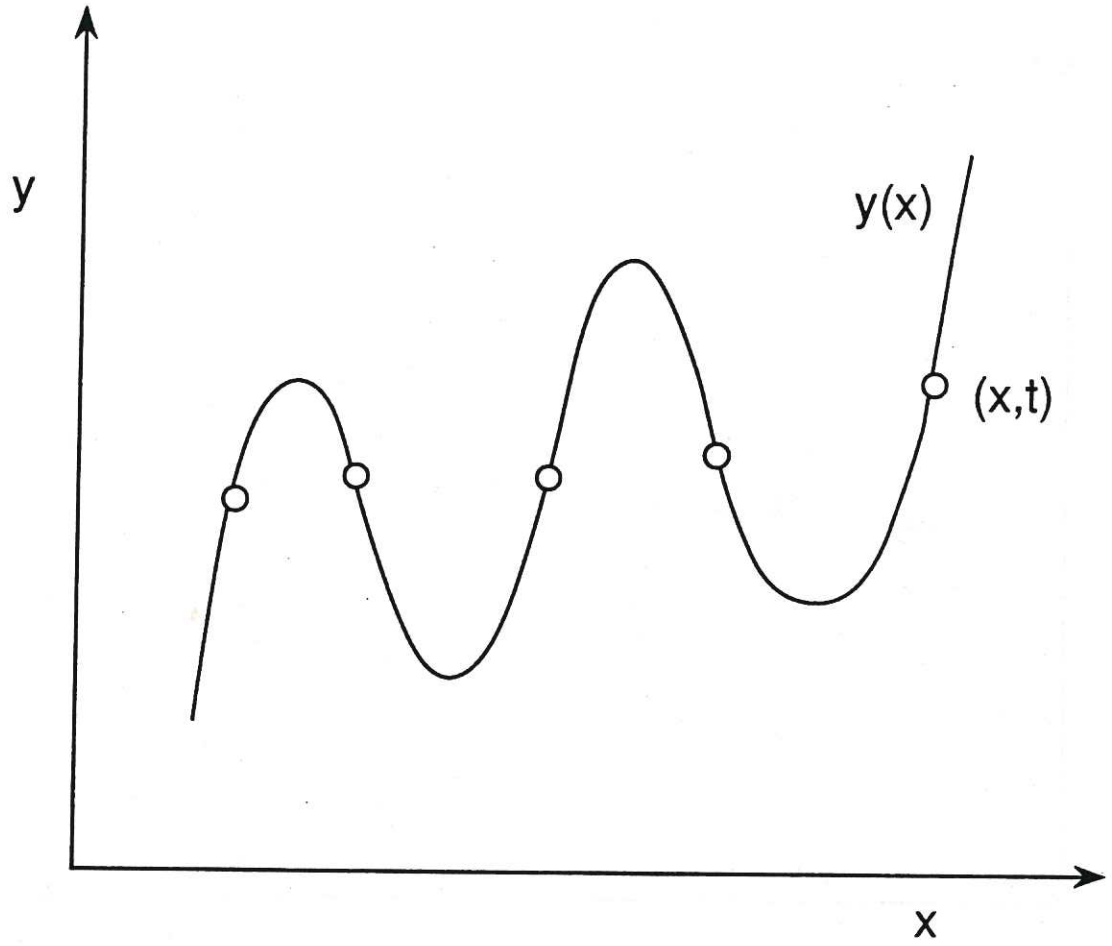

Figure 1. A schematic illustration of a set of data points $\{x_p, t_p\}$, together with an interpolating function $y(x)$ which over-fits the data.

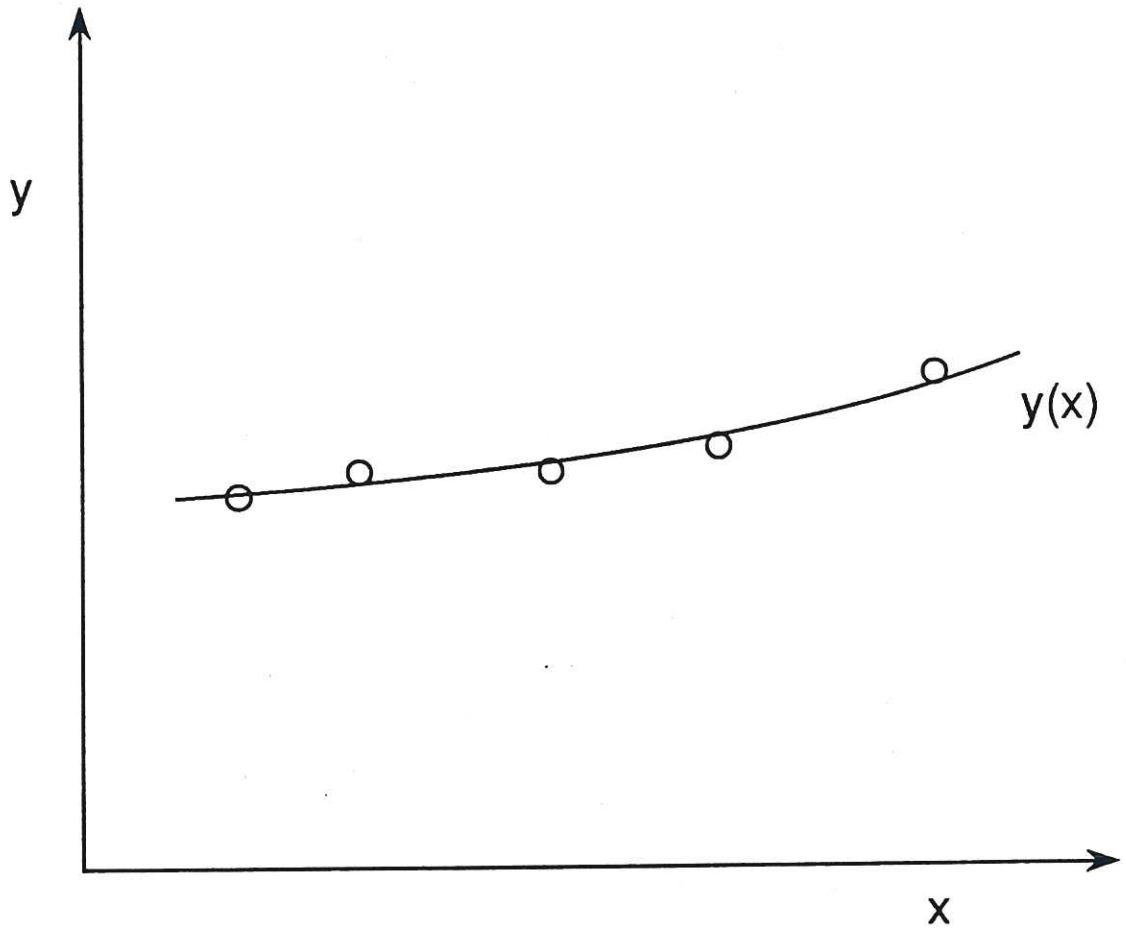

Figure 2. The same data points as in Figure 1, with an interpolating function $y(x)$ which gives a smooth representation of the underlying trend in the data.