Communicated by David Rumelhart

Generalizing Smoothness Constraints from Discrete Samples

Chuanyi Ji
Robert R. Snapp*
Demetri Psaltis
Department of Electrical Engineering, California Institute of Technology, Pasadena, CA 91125 USA

We study how certain smoothness constraints, for example, piecewise continuity, can be generalized from a discrete set of analog-valued data, by modifying the error backpropagation, learning algorithm. Numerical simulations demonstrate that by imposing two heuristic objectives — (1) reducing the number of hidden units, and (2) minimizing the magnitudes of the weights in the network — during the learning process, one obtains a network with a response function that smoothly interpolates between the training data.

1 Introduction

Extensive numerical simulations have demonstrated the utility of error backpropagation (BP) (Rumelhart et al. 1986; Werbos 1974; Sejnowski and Rosenberg 1987) for training a multilayer neural network to learn a given set of input–output associations. The issue of generalization — training a network to respond *reasonably* to input data not present in the training set — is usually addressed by overconstraining the network. Recently, a lower bound on the number of training samples required for generalization by a feedforward network with a fixed number of hidden units has been asymptotically estimated using the saturation property of Vapnik's and Chervonenkis's growth function (Baum and Haussler 1989; Vapnik and Chervonenkis 1968). How to match a network's size and architecture to a given training set so that the response generalizes well is still, however, an unsolved problem.

In addition to accurately replicating the training data, the network response function — the values of the output units as a function of the network inputs — should reflect any constraints that may govern the training problem. Obtaining these constraints directly from a finite set of data is not possible, because many different rules can produce the given

*Current address: Department of Computer Science and Electrical Engineering, Votey Building, University of Vermont, Burlington, VT 05405 USA.

set. Thus, supplementary information is required. This information is generally problem dependent and often eludes a concise formulation.

In this paper, we explore an approach to this problem suitable for analogue association problems subject to certain smoothness constraints. This includes the important class of pattern association problems that originate from a natural setting, and hence are subject to constraints resulting from the continuous nature of physical law. These implicit smoothness constraints have been successfully exploited in other contexts. For example, to infer the locations and orientations of physical surfaces from a set of visual inputs, it is important to assume that these surfaces are piecewise continuous (Marr 1981). Smoothness constraints are often used to "regularize" other ill-posed inverse problems that arise in early vision (Poggio and Koch 1985) and in other fields (Tikhonov and Arsenin 1977). For this broad class of problems, the problem of generalization reduces to that of finding a network with a response function that interpolates smoothly between the training samples. As a simple example, we consider a set of points taken from the graph of a continuous real valued function of one variable and a feedforward network with a single input unit, a layer of N hidden sigmoidal units, followed by a single *linear* output unit. If N and the initial magnitudes of the network weights are chosen too large, the network obtained under the BP algorithm usually has a very irregular response function (cf. Figs. 1 and 2).

One way of obtaining a network with a smooth response function is to add a measure of smoothness to the standard BP error function as a perturbation. For example, one might average an absolute measure of the local curvature of the response function over the expected distribution of network inputs. As this requires integrating over a fine mesh embedded in the input space, more computation may be required than is practical. Instead, we will follow a more intuitive path. In particular we will modify the standard BP learning rule in such a way as to (1) reduce the number of hidden units in the network, and (2) minimize the magnitudes of the network's weights. As it tends to overconstrain the network, the first objective parallels that of polynomial regression, where one seeks the lowest order polynomial that reliably fits a given set of data (Akaike 1977). Furthermore, this objective will often eliminate spurious local extrema in the response function. The second objective is designed to avoid unnecessarily abrupt transitions in the response function. This follows from observing that the gradient of the sigmoidal function $f(\mathbf{w}^t\mathbf{x}-\theta)$ with respect to the input vector, $\mathbf{x}$, is proportional to the weight vector, $\mathbf{w}$.

This intuitive approach was partially inspired by a recent algorithm designed to reduce the number of weights in a network during the training phase (Rumelhart 1988). Here, one auxiliary term, $\sum_i \sigma(w_i)$, was added to the standard BP error function, the summation being performed over all of the weights and thresholds in the network. The terms of the summation, $\sigma(w) = w^2/(1 + w^2)$, measure the magnitudes of the weights

relative to unity. Thus, this summation is a rough measure of the number of "significant" weights in the network; adding it to the error function biases the algorithm toward architectures that use the least number of significant weights. The combined energy function is then minimized by steepest descent. After a certain training criterion is reached, weights with magnitudes falling below a critical threshold can be removed from the network by clamping their values to zero. Although this algorithm reduces the number of weights, it does not effectively reduce the number of hidden units, as architectures with fewer hidden units, but the same number of weights, are not favored.

In contrast to the above, we add two terms to the BP learning rule. The first is designed to remove as many hidden units as possible, while maintaining an acceptable level of error in the response function over the training data. For this to succeed, the units must be operating near their transition regime. The second term is designed to satisfy this requirement as well as minimize the magnitudes of the weights. These modifications to BP are detailed in Section 2. In Section 3, we present the results of several numerical simulations that demonstrate their effectiveness. In the first set of simulations we show that a network, beginning with a large number of hidden units, can be reduced in size to one having a response function that smoothly interpolates between the training data points. In the second set, we construct training data from a network with an "unknown" number of hidden units, and show that the algorithm can be used to infer the architecture of the unknown network with a high probability. Finally, we present our conclusions in Section 4.

2 The Network Reduction Algorithm

We consider the problem of training a feedforward network having a single input unit, one layer of N hidden sigmoidal units, and a single linear output unit, to smoothly interpolate between the M ordered pairs, $\{(x^\pi, y^\pi)\colon\ \pi = 1, \ldots, M\}$, of a given training set. (Here, y^π is the desired output value when the network input is set to x^π.) We assume that the number of hidden units has been initially estimated to be larger than necessary. Let u_i and v_i denote the input and output weights of the ith hidden unit, and θ_i, its threshold value. The response function of the network then has the form $g(x; \mathbf{w}, \boldsymbol{\theta}) = \sum_{i=1}^{N} v_i f(u_i x - \theta_i)$, where, for notational convenience, we let $\mathbf{w} = (u_1, \ldots, u_N, v_1, \ldots, v_N)^t$, and $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_N)^t$. The sigmoidal function is usually taken to be a modified hyperbolic tangent, that is, $f(x) = 1/(1 + e^{-x})$. Under BP, one attempts to find weight and threshold values that minimize the standard error function,

$$\mathcal{E}_0(\mathbf{w}, \boldsymbol{\theta}) = \sum_{\pi=1}^{M} [g(x^\pi; \mathbf{w}, \boldsymbol{\theta}) - y^\pi]^2$$

by gradient descent (Rumelhart et al. 1986; Werbos 1974).

For the architecture described above, we define a hidden unit to be *significant* if it is coupled to both the input and output units with weights of a significantly large magnitude, that is, greater than one. Thus, the quantity,

$$S_i = \sigma(u_i)\sigma(v_i)$$

can be viewed as a measure of the significance of the ith hidden unit, where, as before, $\sigma(w) = w^2/(1 + w^2)$. Following the first objective of the previous section, we desire to favor those architectures that require the fewest number of significant hidden units.

If the given training set does not fully constrain the given network architecture, then there is, in general, a degenerate set of solutions over which $\mathcal{E}_0(\mathbf{w}, \boldsymbol{\theta})$ is acceptably small. Following our first objective, we add a term proportional to

$$\mathcal{E}_1(\mathbf{w}) = \sum_{i=1}^{N} \sum_{j=1}^{i-1} S_i S_j$$

to the standard error function. This biases the algorithm toward those solutions that require the fewest number of significant hidden units. From its definition, $\mathcal{E}_1$ achieves a minimum value of zero if no more than one hidden unit has a nonzero significance, and approaches its upper bound of $N(N-1)/2$ as the magnitudes of all weights increase without bound. After applying the gradient descent algorithm, we obtain the learning rule,

$$w_i^{n+1} = w_i^n - \eta \frac{\partial \mathcal{E}_0}{\partial w_i}(\mathbf{w}^n, \boldsymbol{\theta}^n) - \lambda \frac{\partial \mathcal{E}_1}{\partial w_i}(\mathbf{w}^n) \tag{2.1}$$

$$\theta_i^{n+1} = \theta_i^n - \eta \frac{\partial \mathcal{E}_0}{\partial \theta_i}(\mathbf{w}^n, \boldsymbol{\theta}^n) \tag{2.2}$$

where, η and λ are learning rate parameters. Note that the last term in equation 2.1 couples the dynamics of the weights so that, for example, increasing the significance of the kth hidden unit, respectively increases the decay rate of every weight associated with the other hidden units. Also note that this term is proportional to $\sigma'(w_i) = 2w_i/(1 + w_i^2)^2$, which becomes insignificant for large enough $|w_i|$. This will help stabilize the dominant weights, but will also, in part, necessitate the second objective stated in the previous section.

Because of possible conflicts between the two gradients in equation 2.1, spurious equilibria may exist. It is thus helpful to include the auxiliary term only after the network has learned the training set to a sufficient degree. Consequently, we let $\lambda = \lambda(\mathcal{E}_0) = \lambda_0 \exp(-\beta \mathcal{E}_0)$, where β^{-1} defines a characteristic standard error: the value of $\mathcal{E}_0$ below which the auxiliary term comes into play. Note that the resulting learning rule no longer follows the direction of steepest descent of the combined error function, however, the desired objective is ultimately obtained.

We attain our second objective of reducing the weight magnitudes by subtracting an amount proportional to $\tanh(w_i)$ from the right-hand side of equation 2.1. Although other choices are possible, this one has shown to be effective in practice. Unlike the weight reduction scheme (Rumelhart 1988) discussed in the introduction, our method preferentially reduces the larger weights in the network. We also apply this term to the threshold's update rule, because, in our examples of interest, we are interested in the region of input space around the origin.

We thus obtain the network reduction algorithm,

$$w_i^{n+1} = w_i^n - \eta \frac{\partial \mathcal{E}_0}{\partial w_i}(\mathbf{w}^n, \boldsymbol{\theta}^n) - \lambda \frac{\partial \mathcal{E}_1}{\partial w_i}(\mathbf{w}^n) - \mu \tanh(w_i^n) \tag{2.3}$$

$$\theta_i^{n+1} = \theta_i^n - \eta \frac{\partial \mathcal{E}_0}{\partial \theta_i}(\mathbf{w}^n, \boldsymbol{\theta}^n) - \mu \tanh(\theta_i^n) \tag{2.4}$$

As before, we gate the influence of the new term on how well the network is learning the training set. In this case, it appears helpful to reduce this term gradually with time. In particular, we let $\mu = \mu_0 |\mathcal{E}_0(\mathbf{w}^n, \boldsymbol{\theta}^n) - \mathcal{E}_0(\mathbf{w}^{n-1}, \boldsymbol{\theta}^{n-1})|$.

Once an acceptable level of performance is reached, any weight with magnitude below a certain level is removed from the network. When a hidden unit is connected to the rest of the network with only "removed" weights, then the unit is discarded. Thus, as is desired, the number of hidden units is reduced.

Finally, we mention that this algorithm can be extended to other architectures. For example, for a network having K inputs and L outputs, one may let $S_i = \sum_{k=1}^{K} \sum_{l=1}^{L} \sigma(v_{ki})\sigma(w_{il})$, where, v_{ki} is the value of the weight connecting the kth input to the ith hidden unit, and w_{il}, is that of the weight connecting the ith hidden unit with the lth output.

3 Simulation Results

The first simulation demonstrates that the modified learning rule reduces the number of hidden units and results in a smooth response function. The training set of the first run consists of 9 equally spaced data points taken from the graph of the function, $\phi(x) = e^{-(x-1)^2} + e^{-(x+1)^2}$, over the domain $[-2, 2]$. We begin with the network described in Section 2, with $N = 20$; the 40 weights and 20 thresholds are randomly initialized from a uniform distribution over the interval $[-25, 25]$. With learning parameters set to $\eta \approx 5 \times 10^{-3}$, $\beta = 0.1$, $\lambda_0 = 6.5 \times 10^{-3}$, and $\mu_0 = 5 \times 10^{-4}$, the network is trained by applying the learning rules in equations 2.3 and 2.4. After the value of $\mathcal{E}_0$ falls below the value 0.05 any weight with a magnitude less than 0.1 is set to zero. The resulting network uses only 5 of the 20 available hidden units. At this point, the nominal increase in $\mathcal{E}_0$ resulting from eliminating the weaker weights is corrected by training the reduced network with the standard BP algorithm for a

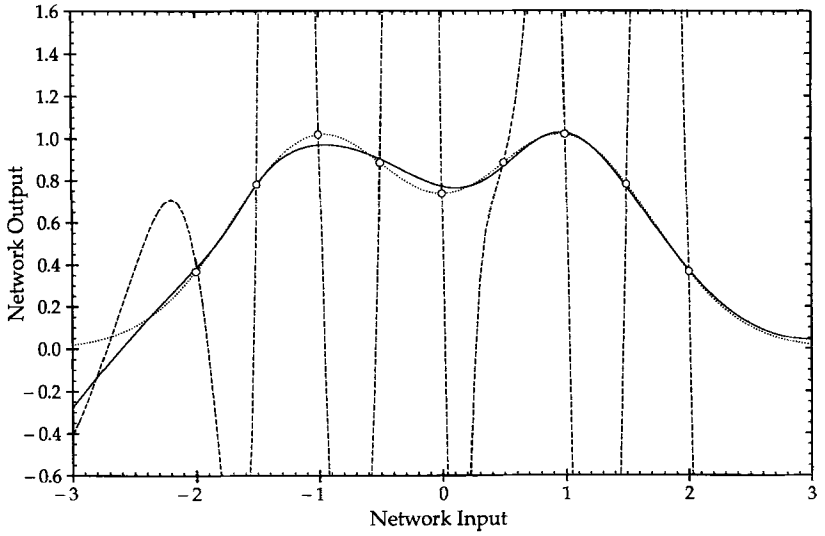

Figure 1: The solid curve indicates the output response of the network with 1 input, 20 hidden units, and 1 output, trained by the network reduction algorithm on the 9 training points indicated by circular markers. The broken curve indicates a response function obtained by training the same network with the same data using BP. The dotted curve indicates the graph of $\phi(x)$ from which the training points were selected.

few additional iterations. In Figure 1, the response function of the network obtained by this procedure is compared against one obtained by BP (*i.e.*, $\lambda_0 = \mu_0 = 0$) with the same initial conditions. Note that the response function obtained by the network reduction algorithm smoothly interpolates between the 9 training points and possesses the same number of local extrema as $\phi(x)$. This cannot be said for the response function generated by BP, which oscillates wildly with minimum and maximum values, -17.5 and 10.9, over the input domain $[-3, 3]$. A more quantitative comparison can be made by averaging the root-mean-square (RMS) deviation between each response function and $\phi(x)$ over the interval $[-2, 2]$, viz.

$$\mathcal{E}_{\text{RMS}} = \frac{1}{2}\left\{\int_{-2}^{2} [g(x; \mathbf{w}, \boldsymbol{\theta}) - \phi(x)]^2 \, dx\right\}^{1/2}.$$

(Note that $\mathcal{E}_{\text{RMS}}$ is a random variable, as the particular determination of the response function, g, depends on the initial values of the weights and thresholds, which are set at random.) For this instance of the net-

work reduction algorithm, we obtain $\mathcal{E}_{\text{RMS}} = 1.15 \times 10^{-2}$, while for BP, $\mathcal{E}_{\text{RMS}} = 1.71$.

If the algorithm does yield a network that generalizes well, then the size of the network should not depend critically on the number of training samples used. This necessarily assumes that the data in each set faithfully represent the significant features of the training problem. Therefore, in a second run, the network is trained with the same initial state, but with a training set containing 17 equally spaced samples taken from the graph of the same function. Again the algorithm reduces the network to 5 significant hidden units, with $\mathcal{E}_{\text{RMS}} = 8.81 \times 10^{-3}$. Training the network under BP with this data yields a network with $\mathcal{E}_{\text{RMS}} = 4.73 \times 10^{-2}$. The response functions of these two networks are displayed in Figure 2.

Next, we explore an "inverse network" problem: To what extent can one use this algorithm to infer the architecture of a feedforward neural network from only a finite sample of its response function? A network containing one input, a layer of two hidden units, and a single linear output is chosen, and a training set of 17 sample points from its response function is generated. Then, an ensemble of 50 new networks, each

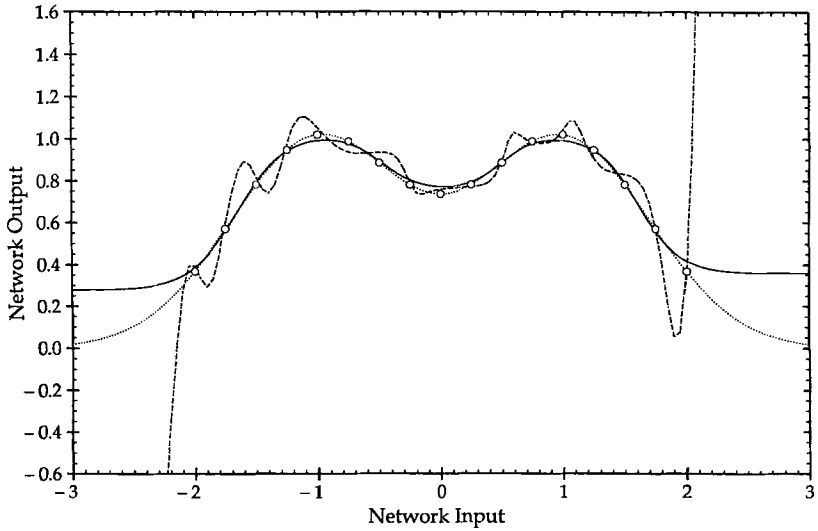

Figure 2: Same as in Figure 1, but with 17 training points. At the left end of the displayed interval, the network response obtained from BP (the broken curve) drops off scale to -5. At the right end, it quickly climbs to 20, and then saturates.

Hidden units (n)	Frequency (ν_n)	$E(\mathcal{E}_{\text{RMS}}\|n)$
2	22	0.0275
3	9	0.0263
4	6	0.0405
5	6	0.0401
6	3	0.0656
7	4	0.0700

Table 1: Simulation Results for the "Inverse Network" Problem. Of a total of 50 runs, the center column shows the number of times a network with n significant hidden units was obtained. The right column equals the average $\mathcal{E}_{\text{RMS}}$ — computed over the interval $[-2, 2]$ — over the ν_n runs ending with n significant hidden units.

containing 10 hidden units, is trained on the data set with the network reduction algorithm. The results of these 50 simulations are summarized in Table 1. Note that 22 times out of 50 the algorithm finds a network of minimal size. It is encouraging that the response functions with the least average RMS error, measured over the interval $[-2, 2]$, come from networks having two or three hidden neurons. In Figure 3, we show how the response functions of the network regress toward that of the "concealed" network as the number of hidden units used decreases. For comparison, 10 networks trained by BP alone yield an average RMS error of 0.461 without any apparent reduction in the number of significant hidden units.

4 Concluding Remarks

In the above we have shown that by adding suitably chosen terms to the BP learning rule, desirable global properties in the network's response function can be obtained. In particular, the BP algorithm was tailored to prefer networks having smoother response functions. From the simulations it is apparent that this behavior is attained at the cost of a slower convergence rate. In the first simulation, where $\phi(x)$ was approximated using a nine point training set, 50,000 iterations were required by the network reduction algorithm, but only 300 were required by BP. This discrepancy was, however, reduced when the networks were trained from the 17 point set: the network reduction algorithm needed 650,000 iterations, and BP, 150,000. Approximately one-half of the 50 "inverse network" simulations required more than $800,000$ iterations. This is only slightly larger than the 400,000 iterations typically needed to train

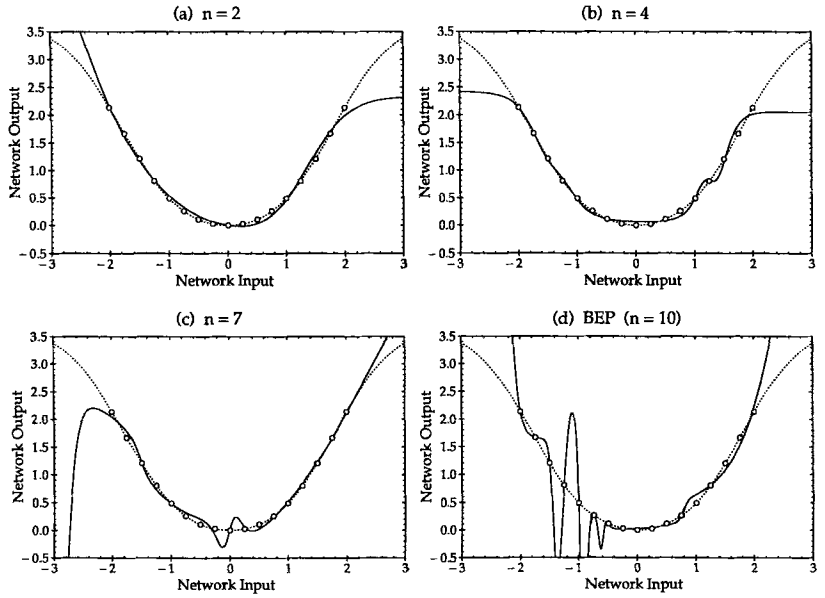

Figure 3: Response functions obtained for the "inverse network" problem are displayed as solid curves, that of the "concealed" network as a dotted curve, and the training points as circular markers. Graphs (a)–(c) reflect the median outcomes — the networks resulting in the median values of $\mathcal{E}_{\text{RMS}}$ — for the sets of trials resulting in n = 2, 4, and 7 significant hidden units, respectively. Graph (d) reflects the median outcome of 10 trials using BP, all of which resulted in 10 significant hidden units. Values of $\mathcal{E}_{\text{RMS}}$ were computed over the input interval $[-2, 2]$; for the response functions displayed in graphs (a)–(d), $\mathcal{E}_{\text{RMS}}$ equals 2.87×10^{-2}, 3.35×10^{-2}, 4.49×10^{-2}, and 0.291, respectively.

a network with 10 hidden units, with the same 17 samples drawn from the "concealed" network, using BP. As BP typically requires a greater number of iterations to learn a given training set on a smaller network, and, as network reduction attempts to select the smallest component of the given network that is just capable of learning the training set, it is not surprising that the network reduction algorithm converges at a slower rate. By a similar argument, we conjecture that network reduction is more effective than training a small network with BP, while gradually adding new hidden units until the network has learned the training set. For example, BP learned the "inverse network" problem within 800,000 iterations in only one of 10 trials, on a network containing two hidden

units, when the weights were initialized randomly from a uniform distribution over the interval $[-25, 25]$. Note, however, that in this case, the convergence rate can be greatly accelerated by chosing initial weight values with smaller magnitudes.

Acknowledgments

We would like to thank our colleagues, Mr. Mark Neifeld, Dr. Jeff Yu, Dr. Fai Mok, and Mr. Alan Yamamura for helpful discussions. This work was supported by the JPL Director's Discretionary Fund and in part by DARPA and the Air Force Office of Scientific Research.

References

Akaike, H. 1977. On entropy maximization principle. In *Applications of Statistics*, P. R. Krishnaiah, ed., pp. 27–41. North-Holland, Amsterdam.

Baum, E. B., and Haussler, D. 1989. What size net gives valid generalization? *Neural Comp.* **1**, 151–160.

Marr, D. 1981. *Vision*, p. 114. W. H. Freeman, San Francisco.

Poggio, T., and Koch, C. 1985. Ill-posed problems in early vision: From computational theory to analogue networks. *Proc. R. Soc. London B* **226**, 303–323.

Rumelhart, D.E. 1988. Presentation given at Hewelett-Packard, Seminar on Neural Networks.

Rumelhart, D.E., Hinton, G.E., and Williams, R.J. 1986. Learning internal representations by error propagation. In *Parallel Distributed Processing*, D.E. Rumelhart and J.L. McClelland, eds., pp. 318–364. MIT Press, Cambridge, MA.

Sejnowski, T.J., and Rosenberg, C.R. 1987. Parallel networks that learn to pronounce English text. *Complex Syst.* **1**, 145–168.

Tikhonov, A.N., and Arsenin, V.Y. 1977. *Solutions of Ill-Posed Problems*. Winston, Washington, D.C.

Vapnik, V.N., and Chervonenkis, A. Ya. 1968. On the uniform convergence of relative frequencies of events to their probabilities. *Dokl. Acad. Nauk SSSR* **181**(4), 781; English translation in *Theory Prob. Appl.* **XVI**, 264–280.

Werbos, P. 1974. Ph. D. thesis, Harvard University.

Received 15 August 1989; accepted 20 February 1990.