Training with Noise

is Equivalent to Tikhonov Regularization

Chris M Bishop

Neural Computing Research Group
Dept. of Computer Science
Aston University
Birmingham, B4 7ET, U.K.

April 1994

Neural Computing Research Group Report NCRG/4290

(Accepted for publication in *Neural Computation*)

Abstract

It is well known that the addition of noise to the input data of a neural network during training can, in some circumstances, lead to significant improvements in generalization performance. Previous work has shown that such training with noise is equivalent to a form of regularization in which an extra term is added to the error function. However, the regularization term, which involves second derivatives of the error function, is not bounded below, and so can lead to difficulties if used directly in a learning algorithm based on error minimization. In this paper we show that, for the purposes of network training, the regularization term can be reduced to a positive definite form which involves only first derivatives of the network mapping. For a sum-of-squares error function, the regularization term belongs to the class of generalized Tikhonov regularizers. Direct minimization of the regularized error function provides a practical alternative to training with noise.

1 Regularization

A feedforward neural network can be regarded as a parameterized non-linear mapping from a d-dimensional input vector $\mathbf{x} = (x_1, \ldots, x_d)$ into a c-dimensional output vector $\mathbf{y} = (y_1, \ldots, y_c)$. Supervised training of the network involves minimization, with respect to the network parameters, of an error function, defined in terms of a set of input vectors $\mathbf{x}$ and corresponding desired (or target) output vectors $\mathbf{t}$. A common choice of error function is the sum-of-squares error of the form

$$
\begin{aligned}
E &= \frac{1}{2} \iint \|\mathbf{y}(\mathbf{x}) - \mathbf{t}\|^2 \, p(\mathbf{x}, \mathbf{t}) \, d\mathbf{x} \, d\mathbf{t} && (1) \\
&= \frac{1}{2} \sum_k \iint \{y_k(\mathbf{x}) - t_k\}^2 \, p(t_k \mid \mathbf{x}) \, p(\mathbf{x}) \, d\mathbf{x} \, dt_k && (2)
\end{aligned}
$$

where $\|\cdots\|$ denotes the Euclidean distance, and k labels the output units. The function $p(\mathbf{x}, \mathbf{t})$ represents the probabilitity density of the data in the joint input-target space, $p(t_k \mid \mathbf{x})$ denotes the conditional density for t_k given the value of $\mathbf{x}$, and $p(\mathbf{x})$ denotes the unconditional density of $\mathbf{x}$. In going from (1) to (2) we have integrated over the variables $t_{j \neq k}$. For a finite discrete data set consisting of n samples labeled by the index q we have

$$
p(\mathbf{x}, \mathbf{t}) = \frac{1}{n} \sum_q \delta(\mathbf{x} - \mathbf{x}^q) \delta(\mathbf{t} - \mathbf{t}^q) \tag{3}
$$

Substituting (3) into (1) gives the sum-of-squares error in the form

$$
E = \frac{1}{2n} \sum_q \|\mathbf{y}(\mathbf{x}^q) - \mathbf{t}^q\|^2 \tag{4}
$$

The results obtained in this paper are derived in the limit of large data sets, and so we shall find it convenient to work with the notation of continuous probability density functions.

One of the central issues in network training is to determine the optimal degree of complexity for the model $y_k(\mathbf{x})$. A model which is too limited will not capture sufficient of the structure in the data, while one which is too complex will model the noise on the data (the phenomenon of overfitting). In either case the performance on new data, that is the ability of the network to generalize, will be poor. The problem can be regarded as one of finding the optimal trade-off between the high bias of a model which is too inflexible and the high variance of a model with too much freedom (Geman et al., 1992).

There are two well-known techniques for controlling the bias and variance of a model, known respectively as *structural stabilization* and *regularization*. The first of these involves making adjustments to the number of free parameters in the model as a way of controlling the number of degrees of freedom. In the case of a feedforward network this is generally done by varying the number of hidden units, or by pruning out individual weights from an initially oversized network. By contrast, the technique of regularization makes use of a relatively flexible model, and then controls the variance by modifying the error function by the addition of a penalty term $\Omega(\mathbf{y})$, so that the total error function becomes

$$
\widetilde{E} = E + \lambda \Omega(\mathbf{y}) \tag{5}
$$

where the parameter λ controls the bias-variance trade-off by affecting the degree to which $\Omega(y)$ influences the minimizing function $\mathbf{y}(\mathbf{x})$. The regularization functional $\Omega(\mathbf{y})$, which is generally expressed in terms of the network function $\mathbf{y}(\mathbf{x})$ and its derivatives, is usually chosen on the basis of some prior knowledge concerning the desired network mapping. For instance, if it is known that the mapping should be smooth, then $\Omega(\mathbf{y})$ may be chosen to be large for functions with large curvature (Bishop, 1991; 1993).

Regularization has been studied extensively in the context of linear models for $\mathbf{y}(\mathbf{x})$. For the case of one input variable x and one output variable y, the class of Tikhonov regularizers takes the form

$$\Omega(y) = \sum_{r=0}^{R} \int_{a}^{b} h_r(x) \left(\frac{d^r y}{dx^r} \right)^2 dx \tag{6}$$

where $h_r \geq 0$ for $r = 0, \ldots, R-1$, and $h_R > 0$. For such regularizers, it can be shown that the linear function $y(x)$ which minimizes the regularized error (5) is unique (Tikhonov and Arsenin, 1977).

2 Training with Noise

There is a third approach to controlling the trade-off of bias against variance, which involves the addition of random noise to the input data during training. This is generally done by adding a random vector onto each input pattern before it is presented to the network, so that, if the patterns are being recycled, a different random vector is added each time. Heuristically, we might expect that the noise will 'smear out' each data point and make it difficult for the network to fit individual data points precisely. Indeed, it has been demonstrated experimentally that training with noise can lead to improvements in network generalization (Sietsma and Dow, 1991). We now explore in detail the relation between training with noise and regularization.

Let the noise on the input vector be described by the random vector $\boldsymbol{\xi}$. The error function when training with noise can then be written in the form

$$\widetilde{E} = \frac{1}{2} \iiint \sum_k \{y_k(\mathbf{x} + \boldsymbol{\xi}) - t_k\}^2 \, p(t_k \mid \mathbf{x}) \, p(\mathbf{x}) \, \widetilde{p}(\boldsymbol{\xi}) d\mathbf{x} \, dt_k \, d\boldsymbol{\xi} \tag{7}$$

where $\widetilde{p}(\boldsymbol{\xi})$ denotes the distribution function of the noise. We now assume that the noise amplitude is small, and expand the network function as a Taylor series in powers of $\boldsymbol{\xi}$ to give

$$y_k(\mathbf{x} + \boldsymbol{\xi}) = y_k(\mathbf{x}) + \sum_i \xi_i \left. \frac{\partial y_k}{\partial x_i} \right|_{\boldsymbol{\xi}=0} + \frac{1}{2} \sum_i \sum_j \xi_i \xi_j \left. \frac{\partial^2 y_k}{\partial x_i \partial x_j} \right|_{\boldsymbol{\xi}=0} + \mathcal{O}(\boldsymbol{\xi}^3) \tag{8}$$

The noise distribution is generally chosen to have zero mean, and to be uncorrelated between different inputs. Thus we have

$$\int \xi_i \widetilde{p}(\boldsymbol{\xi}) \, d\boldsymbol{\xi} = 0 \qquad\qquad \int \xi_i \xi_j \widetilde{p}(\boldsymbol{\xi}) \, d\boldsymbol{\xi} = \eta^2 \delta_{ij} \tag{9}$$

where the parameter η^2 is controlled by the amplitude of the noise. Substituting the Taylor series expansion (8) into the error function (7), and making use of (9) to integrate over the noise distribution, we obtain

$$\widetilde{E} = E + \eta^2 E^R \tag{10}$$

where E is the standard sum-of-squares error defined in (2), and the extra term E^R is given by

$$E^R = \frac{1}{2}\iint \sum_k \sum_i \left\{ \left(\frac{\partial y_k}{\partial x_i}\right)^2 + \frac{1}{2}\{y_k(\mathbf{x}) - t_k\}\frac{\partial^2 y_k}{\partial x_i^2} \right\} p(t_k \mid \mathbf{x})\, p(\mathbf{x})\, d\mathbf{x}\, dt_k \tag{11}$$

This has the form of a regularization term added to the usual sum-of-squares error, with the coefficient of the regularizer determined by the noise variance η^2. This result has been obtained earlier by Webb (1993)[1].

Provided the noise amplitude is small, so that the neglect of higher order terms in the Taylor expansion is valid, the minimization of the sum-of-squares error with noise added to the input data is equivalent to the minimization of the regularized sum-of-squares error, with a regularization term given by (11), without the addition of noise. It should be noted, however, that the second term in the regularization function (11) involves second derivatives of the network function, and so evaluation of the gradients of this error with respect to network weights will be computationally demanding. Furthermore, this term is not positive definite, and so the error function is not, a-priori, bounded below, and is therefore unsuitable for use as the basis of a training algorithm.

We now consider the minimization of the regularized error (10) with respect to the network function $\mathbf{y}(\mathbf{x})$. Our principal result will be to show that the use of the regularization function (11) for network training is equivalent, for small values of the nosie amplitude, to the use of a positive definite regularization function which is of standard Tikhonov form and which involves only *first* derivatives of the network function.

We first define the following conditional averages of the target data

$$\langle t_k \mid \mathbf{x}\rangle \equiv \int t_k\, p(t_k \mid \mathbf{x})\, dt_k \tag{12}$$

$$\langle t_k^2 \mid \mathbf{x}\rangle \equiv \int t_k^2\, p(t_k \mid \mathbf{x})\, dt_k \tag{13}$$

After some simple algebra, we can then write the sum-of-squares error function in (2) in the form

$$\begin{aligned} E &= \frac{1}{2}\sum_k \iint \{y_k(\mathbf{x}) - \langle t_k \mid \mathbf{x}\rangle\}^2\, p(t_k \mid \mathbf{x})\, p(\mathbf{x})\, d\mathbf{x}\, dt_k \\ &+ \frac{1}{2}\sum_k \iint \{\langle t_k^2 \mid \mathbf{x}\rangle - \langle t_k \mid \mathbf{x}\rangle^2\}\, p(t_k \mid \mathbf{x})\, p(\mathbf{x})\, d\mathbf{x}\, dt_k \end{aligned} \tag{14}$$

We note that only the first term in (14) depends on the network mapping $y_k(\mathbf{x})$. The minimum of the error function therefore occurs when the network mapping is given by the conditional average of the target data

$$y_k^{\min}(\mathbf{x}) = \langle t_k \mid \mathbf{x}\rangle \tag{15}$$

[1]A similar analysis was also performed by Matsuoka (1992), but with an inconsistency in the Taylor expansion, which meant that second order terms were treated incorrectly.

This represents the well-known result that the optimal least-squares solution is given by the conditional average of the target data. For interpolation problems it shows that the network will average over intrinsic additive noise on the target data (not to be confused with noise added to the input data as part of training) and hence learn the underlying trend in the data. Similarly, for classifications problems in which the target data uses a $1\!-\!\text{of}\!-\!N$ coding scheme, this results shows that the network outputs can be interpreted as Bayesian posterior probabilities of class membership and so again can be regarded as optimal. Note that (15) represents the global minimum of the error function, and requires that the network model be functionally rich enough that it can be regarded as unbiased. The error function does not vanish at this minimum, however, as there is a residual error given by the second term in equation (14). This residual error represents the mean variance of the target data around its conditional average value.

For the regularized error function given by equation (10) we see that the minimizing function will have the form

$$y_k^{\min}(\mathbf{x}) = \langle t_k \mid \mathbf{x} \rangle + \mathcal{O}(\eta^2) \tag{16}$$

Now consider the second term in equation (11) which depends on the second derivatives of the network function. Making use of the definition in equation (12), we can rewrite this term in the form

$$\frac{1}{4} \iint \sum_k \sum_i \left\{ \{y_k(\mathbf{x}) - \langle t_k \mid \mathbf{x} \rangle\} \frac{\partial^2 y_k}{\partial x_i^2} \right\} p(t_k \mid \mathbf{x})\, p(\mathbf{x})\, d\mathbf{x}\, dt_k \tag{17}$$

Using (16) we see that, to order η^2, this term vanishes at the minimum of the total error function. Thus, only the first term in equation (11) needs to be retained. It should be emphasized that this result is a consequence of the average over the target data. It therefore does not require the individual terms $y_k - t_k$ to be small, only that their (conditional) average over t_k be small.

The minimization of the sum-of-squares error with noise is therefore equivalent (to order η^2) to the minimization of a regularized sum-of-squares error without noise, where the regularizer, given by the first term in equation (14), has the form

$$\widehat{E}^R = \frac{1}{2} \int \sum_k \sum_i \left(\frac{\partial y_k}{\partial x_i} \right)^2 p(\mathbf{x})\, d\mathbf{x} \tag{18}$$

where we have integrated out the t_k variables. Note that the regularization function in equation (18) is not in general equivalent to that given in equation (11). However, the total regularized error in each case is minimized by the same network function $\mathbf{y}(\mathbf{x})$ (and hence by the same set of network weight values). Thus, for the purposes of network training, we can replace the regularization term in equation (11) with the one in equation (18). For a discrete data set, with probability distribution function given by equation (3), this regularization term can be written as

$$\widehat{E}^R = \frac{1}{2n} \sum_q \sum_k \sum_i \left(\frac{\partial y_k^q}{\partial x_i^q} \right)^2 \tag{19}$$

Note that there is nothing in our analysis which is specific to neural networks. The advantage of feedforward networks, as parameterized non-linear models for the function

$\mathbf{y}(\mathbf{x})$, is that their relative flexibility allows them to represent good approximations to the optimal solution given by equation (16).

We can apply a similar analysis in the case of the cross-entropy error function given by

$$E = -\iint \sum_k \{t_k \ln y_k(\mathbf{x}) + (1 - t_k)\ln(1 - y_k(\mathbf{x}))\}\ p(t_k \mid \mathbf{x})\, p(\mathbf{x})\, d\mathbf{x}\, dt_k \tag{20}$$

Using the Taylor expansion (8) as before, we again arrive at a regularized error function of the form

$$\widetilde{E} = E + \eta^2 E^R \tag{21}$$

where the regularizer is given by

$$\begin{aligned} E^R \;=\; & \frac{1}{2}\iint \sum_k \sum_i \left\{ \left[\frac{1}{y_k(1-y_k)} - \frac{(y_k - t_k)(1-2y_k)}{y_k^2(1-y_k)^2} \right] \left(\frac{\partial y_k}{\partial x_i} \right)^2 \right. \\ & \left. + \left[\frac{(y_k - t_k)}{y_k(1-y_k)} \right] \frac{\partial^2 y_k}{\partial x_i^2} \right\} p(t_k \mid \mathbf{x})\, p(\mathbf{x})\, d\mathbf{x}\, dt_k \end{aligned} \tag{22}$$

which involves second derivatives of the network mapping function, and which contains terms which are not positive definite[2]. From equation (20) it follows that the network function which minimizes the regularized error again has the form given in equation (16). Using this result, and following a similar line of argument to that presented for the sum-of-squares error, we see that the second and third terms in (22) vanish. Thus, this regularization function can be simplified to give

$$E^R = \frac{1}{2}\int \sum_k \sum_i \left\{ \frac{1}{y_k(1-y_k)} \left(\frac{\partial y_k}{\partial x_i} \right)^2 \right\} p(\mathbf{x})\, d\mathbf{x} \tag{23}$$

Again, we see that this is now positive definite, and that it only involves first derivatives. Note, however, that it is not of the standard Tikhonov form given in equation (6). For a discrete data set, as described by equation (3), we have

$$E^R = \frac{1}{2}\sum_q \sum_k \sum_i \left\{ \frac{1}{y_k^q(1-y_k^q)} \left(\frac{\partial y_k^q}{\partial x_i^q} \right)^2 \right\} \tag{24}$$

Efficient techniques for evaluating the derivatives of regularization functions such as equations (19) or (24) with respect to the weights in a feedforward network, based on extensions of the standard backpropagation technique, have been described in Bishop (1993). These derivatives can be used as the basis for standard training algorithms such as gradient descent or conjugate gradients. An alternative to training with noise is therefore to minimize the regularized error functions directly.

3 Perturbative Solution

In our analysis we have assumed the noise amplitude to be small. This allows us to find a perturbative solution for the set of neural network weights which minimizes the

[2] When this form of cross-entropy error function is used, it is convenient to take the activation functions of the output units to have the logistic sigmoid form $y(a) = \{1 + e^{-a}\}^{-1}$ where a is the total input to the unit. This has the property that $y'(a) = y(1-y)$ which leads to some simplification of derivatives when they are expressed in terms of a instead of y.

regularized error function, in terms of the weights obtained by minimizing the sum-of-squares error function without regularization (Webb, 1993). Let the unregularized error function be minimized by a weight vector $\mathbf{w}^\star$ so that

$$\left.\frac{\partial E}{\partial w_n}\right|_{\mathbf{w}^\star} = 0 \tag{25}$$

If we write the minimum for the regularized error function in the form $\mathbf{w}^\star + \Delta\mathbf{w}$, then we have

$$0 = \left.\frac{\partial(E + \eta^2 E^R)}{\partial w_n}\right|_{\mathbf{w}^\star+\Delta\mathbf{w}} = \left.\frac{\partial E}{\partial w_n}\right|_{\mathbf{w}^\star} + \sum_m \Delta w_m \left.\frac{\partial^2 E}{\partial w_n \partial w_m}\right|_{\mathbf{w}^\star} + \eta^2 \left.\frac{\partial E^R}{\partial w_n}\right|_{\mathbf{w}^\star} \tag{26}$$

If we consider the discrete regularizer given in equation (19), and make use of equation (25), we obtain an explicit expression for the correction to the weight values in the form

$$\Delta\mathbf{w} = -\eta^2 \mathbf{H}^{-1} \sum_q \sum_k \sum_i \nabla_{\mathbf{w}} \left(\frac{\partial y_k}{\partial x_i^q}\right)^2 \tag{27}$$

where $\mathbf{H}$ is the hessian matrix whose elements are defined by

$$(\mathbf{H})_{nm} = \frac{\partial^2 E}{\partial w_n \partial w_m} \tag{28}$$

A similar result is obtained for the case of the cross-entropy error function. An exact procedure for efficient calculation of the hessian matrix for a network of arbitrary feedforward topology was given in Bishop (1992). Similarly, extended backpropagation algorithms for evaluating derivatives with respect to the weights of the form occuring on the right hand side of (27) were derived in Bishop (1993). The fact that the second derivative terms in (11) can be dropped means that only second derivatives of the network function occur in equation (27), and so the method can be considered for practical implementation. With third derivatives present, the evaluation of the weight corrections would become extremely cumbersome.

4 Summary

We have considered three distinct approaches to controlling the tradeoff between bias and variance, as follows:

1. Minimize a sum-of-squares error function, and add noise to the input data during training;
2. Minimize directly a regularized sum-of-squares error function without adding noise to the input data, where the regularization term is given by equation (19);
3. Minimize a sum-of-squares error without adding noise to the input data, and then compute the corrections to the network weights using equation (27);

(with analogous results for the cross-entropy error function). If the noise amplitude parameter η^2 is small, these three methods are equivalent.

References

Bishop C M, 1991. Improving the generalization properties of radial basis function neural networks, *Neural Computation* **3** 579–588.

Bishop C M, 1992. Exact calculation of the Hessian matrix for the multilayer perceptron, *Neural Computation* **4** 494–501.

Bishop C M, 1993. Curvature-driven smoothing in feedforward networks, *IEEE Transactions on Neural Networks*, **4** 882–884.

Geman S, Bienenstock E and Doursat R, 1992. Neural networks and the bias/variance dilema, *Neural Computation* **4** 1–58.

Matsuoka K, 1992. Noise injection into inputs in back-propagation training, *IEEE Transactions on Systems, Man and Cybernetics* **22** 436–440.

Sietsma J, and Dow R J F, 1991. Creating artificial neural networks that generalize, *Neural Networks* **4** 67–79.

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

Webb A R, 1993. Functional approximation by feed-forward networks: a least-squares approach to generalization, *IEEE Transactions on Neural Networks*, in press.