Similarities of Error Regularization, Sigmoid Gain Scaling, Target Smoothing, and Training with Jitter

Russell Reed, *Member, IEEE*, Robert J. Marks II, *Fellow, IEEE*, and Seho Oh

***Abstract*—The generalization performance of feedforward layered perceptrons can, in many cases, be improved either by smoothing the target via convolution, regularizing the training error with a smoothing constraint, decreasing the gain (i.e., slope) of the sigmoid nonlinearities, or adding noise (i.e., jitter) to the input training data. In certain important cases, the results of these procedures yield highly similar results although at different costs. Training with jitter, for example, requires significantly more computation than sigmoid scaling.**

I. Introduction

CONSIDER a layered perceptron with a single hidden layer and input $\boldsymbol{x}$. Let the data have a target function $t(\boldsymbol{x})$, and let $p_{\boldsymbol{n}}(\boldsymbol{x})$ be a noise probability density function. Then, under conditions detailed in this paper, the following four training procedures yield highly similar results.

1) *Convolutional Target Smoothing*: Replace the target function with the effective target

$$t_{\text{eff}}(\boldsymbol{x}) = \int_{\boldsymbol{y}} t(\boldsymbol{y}) p_{\boldsymbol{n}}(\boldsymbol{x}-\boldsymbol{y})\, d\boldsymbol{y} = t(\boldsymbol{x}) * p_{\boldsymbol{n}}(\boldsymbol{x})$$

where '*' denotes convolution.

2) *Jittered Training Data*: Train with jittered data using $p_{\boldsymbol{n}}(\boldsymbol{x})$ as the jitter density [27].

3) *Sigmoid Scaling*: Train without jitter. After training, scale the sigmoid slopes (or, equivalently, the node weights) [26]. The manner in which the slopes are scaled is dictated by $p_{\boldsymbol{n}}(\boldsymbol{x})$ (see (22)).

4) *Error Regularization*: Train with the regularized error function [25]

$$E + \lambda \left\| \frac{\partial y}{\partial \boldsymbol{x}} \right\|^2$$

where E is the conventional sum-of-squares error, y is network output for the input $\boldsymbol{x}$, and

$$\frac{\partial y}{\partial \boldsymbol{x}} = \left(\frac{\partial y}{\partial x_1}, \frac{\partial y}{\partial x_2}, \cdots, \frac{\partial y}{\partial x_n} \right)$$

is the gradient of $y(\boldsymbol{x})$. The Lagrangian multiplier, λ, is chosen to be equal to the variance of $p_{\boldsymbol{n}}(\boldsymbol{x})$.

The similarity of these four procedures is established analytically and illustrated by examples.

Manuscript received October 28, 1992; revised March 28, 1994. Portions of this paper have appeared in [27] and [26].

The authors are with the Department of Electrical Engineering, FT-10, University of Washington, Seattle, WA 98195 USA.

IEEE Log Number 9409159.

II. Training with Jitter

Many studies of artificial neural networks (e.g., [24], [7], [29], [34], [31], [17], [23], [30], [20]) have noted that adding small amounts of input noise (jitter) to the training data often results in improved generalization and fault tolerance. Holmström and Koistinen [11], [13], [14] have shown that training with jitter is consistent, in that, under appropriate conditions, the resulting error function approaches the true error function as the number of training samples increases and the degree of jitter decreases.

Consider a network trained with noisy input data, $\{\boldsymbol{x} + \boldsymbol{n}, t(\boldsymbol{x})\}$, where $\boldsymbol{n}$ is noise that varies with each presentation. During training, the network sees the target $t(\boldsymbol{x})$ in conjunction with the noisy input $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{n}$. The input $\tilde{\boldsymbol{x}}$ may be produced by various combinations of inputs $\boldsymbol{x}$ and noises $\boldsymbol{n}$ while the target depends only on $\boldsymbol{x}$. Various targets may therefore be associated with the same noisy input $\tilde{\boldsymbol{x}}$. The network, however, can produce only a single output for any given input. For arbitrary noise and input sampling distributions, the effective target for a given input $\tilde{\boldsymbol{x}}$ is the expected value

$$\langle t(\boldsymbol{x}) | \tilde{\boldsymbol{x}} \rangle = \frac{\int_n t(\tilde{\boldsymbol{x}} - \boldsymbol{n}) p_{\boldsymbol{x}}(\tilde{\boldsymbol{x}} - \boldsymbol{n}) p_{\boldsymbol{n}}(\boldsymbol{n})\, d\boldsymbol{n}}{\int_\xi p_{\boldsymbol{x}}(\tilde{\boldsymbol{x}} - \xi)\, p_{\boldsymbol{n}}(\xi)\, d\xi}. \quad (1)$$

In the special case where the sampling distribution is uniform and the standard deviation of the noise is small (relative to the extent of the input domain), the interaction between $p_{\boldsymbol{x}}$ and $p_{\boldsymbol{n}}$ will have little effect in the interior of the domain. In regions where these boundary effects can be ignored, the $p_{\boldsymbol{x}}$ terms cancel out, the denominator integrates to one, and this simplifies to the approximation

$$\langle t(\tilde{\boldsymbol{x}} - \boldsymbol{n}) | \tilde{\boldsymbol{x}} \rangle \approx \int_{\boldsymbol{n}} t(\tilde{\boldsymbol{x}} - \boldsymbol{n}) p_{\boldsymbol{n}}(\boldsymbol{n})\, d\boldsymbol{n} \approx t(\tilde{\boldsymbol{x}}) * p_{\boldsymbol{n}}(\tilde{\boldsymbol{x}}). \quad (2)$$

Thus, in this special case, the effective target when training with jittered input data is approximately equal to the convolution of the original target $t(\boldsymbol{x})$ and the noise density $p_{\boldsymbol{n}}(\boldsymbol{x})$.

The convolution is a smoothing operation in general. If, for example, $t(\boldsymbol{x})$ is a step function and $p_{\boldsymbol{n}}(\boldsymbol{x})$ is Gaussian, then the convolution produces the Gaussian cumulative distribution which is a smooth function similar to the sigmoid. This convolutional property resulting from jittered sampling is studied in [12].

A. Effective Target for Sampled Data

The convolution property holds when training data are continuously and uniformly distributed over the entire input space and the magnitude of the noise is small. In practice, however, we typically only have samples $\{(\boldsymbol{x}_i, t_i)\}$ of the underlying function. In this case, the distribution of $\tilde{\boldsymbol{x}} = \boldsymbol{x}_i + \boldsymbol{n}$ is not uniform and the optimum output function is modified.

Let the training set be $\{(\boldsymbol{x}_i, t_i) | i = 1 \cdots M\}$. During training, we randomly select one of the training pairs with equal probability, add noise to the input, and apply it to the network. Given that the training point is $\boldsymbol{x}_k$ (a randomly selected point from the training set), the probability density of the noisy input $\tilde{\boldsymbol{x}}$ is

$$P[\boldsymbol{x} + \boldsymbol{n} = \tilde{\boldsymbol{x}} | \boldsymbol{x} = \boldsymbol{x}_k] = p_{\boldsymbol{n}}(\tilde{\boldsymbol{x}} - \boldsymbol{x}_k).$$

The training points are selected from the training set with equal probabilities $P[\boldsymbol{x} = \boldsymbol{x}_k] = 1/M$ so the probability density of the input to the network, $\tilde{\boldsymbol{x}}$, is

$$\begin{aligned} P[\tilde{\boldsymbol{x}}] &\equiv P[\boldsymbol{x} + \boldsymbol{n} = \tilde{\boldsymbol{x}}] \\ &= \sum_{k=1}^{M} P[\boldsymbol{x} + \boldsymbol{n} = \tilde{\boldsymbol{x}} | \boldsymbol{x} = \boldsymbol{x}_k] P[\boldsymbol{x} = \boldsymbol{x}_k] \\ &= \frac{1}{M} \sum_{k=1}^{M} p_{\boldsymbol{n}}(\tilde{\boldsymbol{x}} - \boldsymbol{x}_k). \end{aligned} \tag{3}$$

Given that a particular noisy input $\tilde{\boldsymbol{x}}$ is observed, the probability that it is generated by training data $\boldsymbol{x}_k$ plus noise is found by Bayes' rule

$$\begin{aligned} P[\boldsymbol{x} = \boldsymbol{x}_k | \boldsymbol{x} + \boldsymbol{n} = \tilde{\boldsymbol{x}}] &= \frac{P[\boldsymbol{x} + \boldsymbol{n} = \tilde{\boldsymbol{x}} | \boldsymbol{x} = \boldsymbol{x}_k] P[\boldsymbol{x} = \boldsymbol{x}_k]}{P[\boldsymbol{x} + \boldsymbol{n} = \tilde{\boldsymbol{x}}]} \\ &= \frac{p_{\boldsymbol{n}}(\tilde{\boldsymbol{x}} - \boldsymbol{x}_k)(1/M)}{(1/M) \sum_{j=1}^{M} p_{\boldsymbol{n}}(\tilde{\boldsymbol{x}} - \boldsymbol{x}_j)} \\ &= \frac{p_{\boldsymbol{n}}(\tilde{\boldsymbol{x}} - \boldsymbol{x}_k)}{\sum_{j=1}^{M} p_{\boldsymbol{n}}(\tilde{\boldsymbol{x}} - \boldsymbol{x}_j)}. \end{aligned} \tag{4}$$

Let $P(k|\tilde{\boldsymbol{x}})$ denote this probability.

The expected value of the training target, given the noisy training input $\tilde{\boldsymbol{x}}$, is then

$$\begin{aligned} \langle t_{\text{train}}(\tilde{\boldsymbol{x}} - \boldsymbol{n}) | \tilde{\boldsymbol{x}} \rangle &= \sum_i t_i P(i|\tilde{\boldsymbol{x}}) \\ &= \sum_i \frac{t_i p_{\boldsymbol{n}}(\tilde{\boldsymbol{x}} - \boldsymbol{x}_i)}{\sum_k p_{\boldsymbol{n}}(\tilde{\boldsymbol{x}} - \boldsymbol{x}_k)}. \end{aligned} \tag{5}$$

This describes the expected value of the training target given that the input is a noisy version of one of the training samples. As the number of samples approaches ∞, the distribution of the samples approaches $p_{\boldsymbol{x}}$ and (5) becomes a good approximation to (1).

Let $y(\tilde{\boldsymbol{x}})$ be the network output for the input $\tilde{\boldsymbol{x}}$. The expected value of the error while training, given this input, is

$$\tilde{\mathcal{E}} = \sum_{i=1}^{M} (t_i - y(\tilde{\boldsymbol{x}}))^2 P(i|\tilde{\boldsymbol{x}}). \tag{6}$$

Abbreviate $P(i|\tilde{\boldsymbol{x}})$ with P_i and $y(\tilde{\boldsymbol{x}})$ with y. After expanding the square

$$\begin{aligned} \tilde{\mathcal{E}} &= \Big(\sum t_i^2 P_i\Big) - 2y\Big(\sum t_i P_i\Big) + y^2 \sum P_i \\ &= \Big(\sum t_i^2 P_i\Big) - \Big(\sum t_i P_i\Big)^2 + \Big[\Big(\sum t_i P_i\Big)^2 \\ &\quad - 2y\Big(\sum t_i P_i\Big) + y^2 \sum P_i\Big] \\ &= \Big(\sum t_i^2 P_i\Big) - \Big(\sum t_i P_i\Big)^2 + \Big[\Big(\sum_i t_i P_i\Big) - y\Big]^2 \end{aligned}$$

and

$$\frac{\partial \tilde{\mathcal{E}}}{\partial w} = -2\Big[\Big(\sum_i t_i P_i\Big) - y\Big] \frac{\partial y}{\partial w}. \tag{7}$$

In other words, under gradient descent, the system acts as if the target function is $\Sigma_i t_i P_i$, the expected value of the target in (5), given the conditions stated for P_i. Likewise, from (7) the effective error function is

$$E_{\text{eff}} = \Big[\Big(\sum_i t_i P_i\Big) - y\Big]^2 \tag{8}$$

in the sense that $\partial E_{\text{eff}}/w = \partial \tilde{\mathcal{E}}/w$. In contrast to conventional training with discrete data, the effective target function for jittered data is defined for all $\boldsymbol{x}$ and not just at the given training points. The following example illustrates the point.

Example 1: Fig. 1(a) is the Voronoi map of a set of points in two dimensions, the basis for a nearest neighbor classifier. Fig. 1(b) is expression (5), and Fig. 1(c) is convolution of the sampled target function with a Gaussian function. It can be seen that the convolution property smooths the decision surface and removes small features. Note that the zero contours in Fig. 1(b)–(c) coincide.

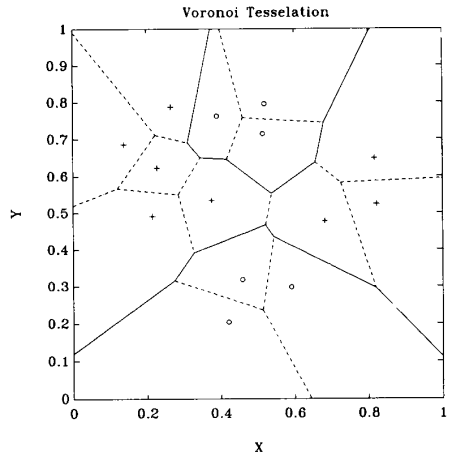
III. Relationship of Error Regularization and Training with Jitter

Regularization is another method used to improve generalization. Regularization often assumes that the target function is smooth and that small changes in the input do not cause large changes in the output. Poggio and Girosi [25], for example, suggest the cost function

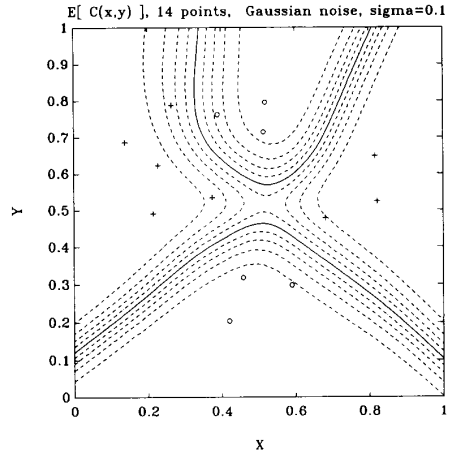
$$\sum_i (y_i - t_i)^2 + \lambda ||Py||^2 \tag{9}$$

where "P is usually a differential operator" and λ balances the trade-off between smoothing and minimizing the error.

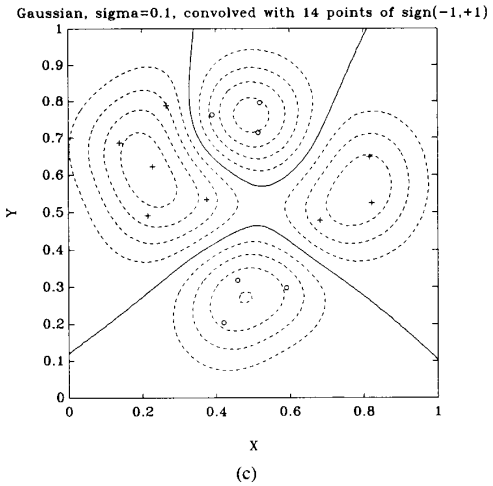
Jittering the inputs while keeping the target fixed embodies this smoothness assumption and results in a similar cost function. That is, we add small amounts of noise to the input



(a)



(b)



(c)

Fig. 1. A 14-point nearest neighbor classification problem. (a) The Voronoi map for the 14 points. (b) The expected value for the classification of x calculated in (5) for a Gaussian noise distribution ($\sigma = 0.1$). (c) The convolution of the training set with the same Gaussian noise. The zero contours of (b) and (c) coincide.

data, assume that the target function does not change much, and minimize

$$\mathcal{E} = \{\langle [t(\boldsymbol{x}) - y(\boldsymbol{x}+\boldsymbol{n})]^2 \rangle\} \tag{10}$$

$$= \{t(\boldsymbol{x})^2 - 2t(\boldsymbol{x})\langle y(\boldsymbol{x}+\boldsymbol{n})\rangle + \langle y(\boldsymbol{x}+\boldsymbol{n})^2\rangle\} \tag{11}$$

where $\{u\}$ indicates the expected value of u over the training patterns and $\langle u \rangle$ indicates the expected value of u over the noise $\boldsymbol{n}$. For small $||\boldsymbol{n}||$, the network output can be approximated by

$$y(\boldsymbol{x}+\boldsymbol{n}) \approx y(\boldsymbol{x}) + \left(\frac{\partial y}{\partial \boldsymbol{x}}\right)^T \boldsymbol{n} + \frac{1}{2}\boldsymbol{n}^T H \boldsymbol{n} \tag{12}$$

where the superscript T denotes the vector transpose and H is the Hessian matrix with element $h_{ij} = \partial^2 y/(\partial x_i \partial x_j)$. For surfaces with small curvature, we can assume H is approximately zero; the complete calculation is given in Appendix A.

Substitution into (11) and dropping the independent variable for brevity gives

$$\mathcal{E} \approx \left\{ \begin{array}{l} t^2 - 2ty - 2t\left(\dfrac{\partial y}{\partial \boldsymbol{x}}\right)^T \langle \boldsymbol{n} \rangle \\ \quad +y^2 + 2y\left(\dfrac{\partial y}{\partial \boldsymbol{x}}\right)^T \langle \boldsymbol{n} \rangle + \left(\dfrac{\partial y}{\partial \boldsymbol{x}}\right)^T \langle \boldsymbol{n}\boldsymbol{n}^T \rangle \left(\dfrac{\partial y}{\partial \boldsymbol{x}}\right) \end{array} \right\}. \tag{13}$$

Assume zero-mean uncorrelated noise with equal variances, $\langle \boldsymbol{n} \rangle = 0$ and $\langle \boldsymbol{n}\boldsymbol{n}^T \rangle = \sigma^2 \boldsymbol{I}$. Then

$$\mathcal{E} \approx \left\{ t^2 - 2ty + y^2 + \sigma^2 \left(\frac{\partial y}{\partial \boldsymbol{x}}\right)^T \left(\frac{\partial y}{\partial \boldsymbol{x}}\right) \right\} \tag{14}$$

$$\approx \{(t-y)^2\} + \sigma^2 \left\{ \left|\left| \frac{\partial y}{\partial \boldsymbol{x}} \right|\right|^2 \right\}. \tag{15}$$

The term $\{(t-y)^2\} = E$ is the conventional unregularized error function and the term $\{||(\partial y/\partial \boldsymbol{x}||^2\}$ is the squared magnitude of the gradient of $y(\boldsymbol{x})$ averaged over the training points.

$\mathcal{E}$ is an approximation to the regularized error function in (9). Like (9), it introduces a term that encourages smooth solutions [33], [4]. Comparison of (15) and (9) shows that σ^2 plays a role similar to λ in the regularization equation, balancing smoothness and error minimization. They differ in that training with jitter minimizes the gradient term at the training points whereas regularization usually seeks to minimize it for all $\boldsymbol{x}$.

Equation (15) shows that, when it can do so without increasing the conventional error, the system minimizes sensitivity to input noise by driving the gradient of the transfer function to zero at the training points. A similar result is derived in [19] and, by analogy with the ridge estimate method of linear regression, in [18]. A system which explicitly calculates and backpropagates similar terms in a multilayer perceptron is described in [6]. A more general approach using the Hessian information is described in [2]–[4].

Example 2: Fig. 2(a) shows the decision boundary formed by an intentionally overtrained 2/50/10/1 feedforward network. With 671 weights, but only 31 training points, the network is very underconstrained and chooses a very nonlinear boundary.

Training with jittered data discourages sharp changes in the response near the training points and so discourages the network from forming overly complex boundaries. Fig. 2(b) shows the same network trained for the same amount of time from the same initial conditions with Gaussian input noise ($\sigma = 0.1$). Despite very long training times, the network shows no effects of overtraining. For reference, Fig. 2(c) shows the expression in (5).

IV. Relationship between Training with Jitter and Sigmoid Scaling

A drawback of training with jitter is that it usually requires the use of a small learning rate and many sample presentations to average over the noise. In this section, we demonstrate that, in certain cases, the expected response of a network driven by a jittered input can be approximated by simply adjusting the sigmoid slopes. This is, of course, generally much faster than averaging over the noise. This result provides justification for gain scaling as a heuristic for improving generalization. Application of this technique to a single-hidden-layer perceptron with a linear output is considered.

A. Linear Output Networks

Consider the function

$$y(\boldsymbol{x}) = \sum_k v_k h_k(\boldsymbol{x}) \tag{16}$$

where

$$h_k(\boldsymbol{x}) = g(w_k^T \boldsymbol{x} - \theta_k) \tag{17}$$

and $g(\cdot)$ is the node nonlinearity. This describes a single-hidden-layer network with a linear output.

With jitter (and the approximations stated for (2)), the expected output for a fixed input $\boldsymbol{x}$ is

$$\begin{aligned} \langle y(\boldsymbol{x}+\boldsymbol{n})\rangle &\approx y(\boldsymbol{x}) \star p_{\boldsymbol{n}}(\boldsymbol{x}) \\ &\approx \sum_k v_k h_k(\boldsymbol{x}) \star p_{\boldsymbol{n}}(\boldsymbol{x}) \\ &\approx \sum_k v_k [h_k(\boldsymbol{x}) * p_{\boldsymbol{n}}(\boldsymbol{x})] \end{aligned} \tag{18}$$

i.e., a linear sum of convolutions of the hidden unit responses with the noise density. The symbol $\star$ denotes correlation, $a(x) \star b(x) = \int_{-\infty}^{+\infty} a(\tau) b(\tau - x)\, d\tau$, and should not be confused with the * convolution operator although correlation and convolution can be interchanged here since we assume that $p_{\boldsymbol{n}}(\boldsymbol{x})$ is symmetric.

In most neural network applications, the nonlinearity is the sigmoid $g(z) = 1/(1+e^{-z})$. If, instead, we use the Gaussian cumulative distribution function (GCDF), which has a very similar shape (see Fig. 3), then the shape of the nonlinearity will be invariant to convolution with a Gaussian input noise

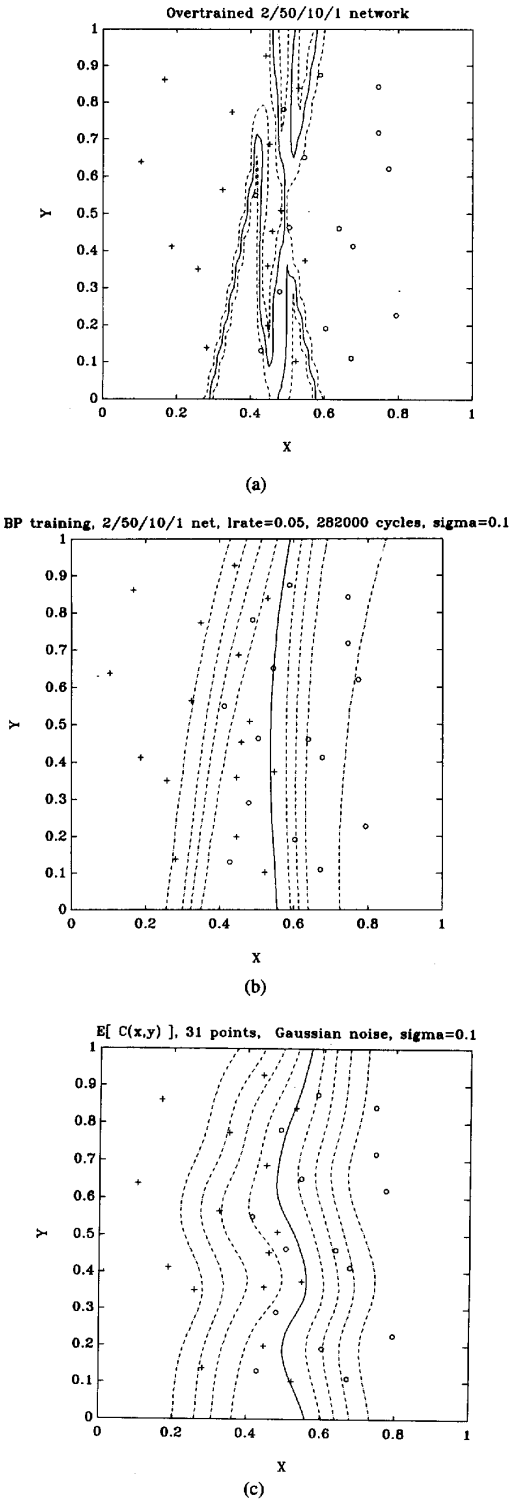


Fig. 2. (a) An intentionally overtrained 2/50/10/1 feed-forward network chooses an overly complex boundary and can be expected to generalize poorly. (b) The same network trained with Gaussian ($\sigma = 0.1$) input noise forms a much smoother boundary and better generalization can be expected. (c) The expression in (5) for the expected value of the target function at an arbitrary point x.

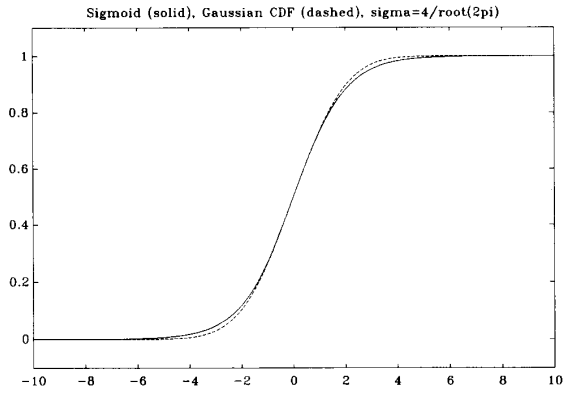


Fig. 3. The conventional sigmoid $1/(1+e^{-x})$ and the Gaussian cumulative distribution (GCDF) function have very similar shapes and give similar results when used as the node nonlinearities. The GCDF is useful in this analysis because it is shape invariant when convolved with a spherical Gaussian noise density.

density. That is, if we assume that the noise is zero-mean Gaussian and spherically distributed in N dimensions

$$p_{\boldsymbol{n}}(\boldsymbol{x}) = \frac{1}{\sigma_1^N (2\pi)^{N/2}} \exp\left(\frac{-||\boldsymbol{x}||^2}{2\sigma_1^2}\right) \tag{19}$$

(where $||\boldsymbol{x}||^2 = \boldsymbol{x}^T\boldsymbol{x}$) and the g nonlinearity is the Gaussian cumulative distribution function

$$g(z) = \int_{-\infty}^{z} \frac{1}{\sigma_2\sqrt{2\pi}} \exp\left(\frac{-\tau^2}{2\sigma_2^2}\right) d\tau \tag{20}$$

then the convolution in (18) can be replaced by a simple scaling operation

$$h_k(\boldsymbol{x}) * p_{\boldsymbol{n}}(\boldsymbol{x}) = g(a_k(w_k^T\boldsymbol{x} - \theta_k)) \tag{21}$$

where a_k is a scaling constant defined below. A derivation is given in Appendix B.

The significance of this is that when equivalence (21) holds, the expected response of the network to input noise approximated by (18) can be computed exactly by simply scaling the hidden unit nonlinearities appropriately; we do not have to go through the time consuming process of estimating the response by averaging over many noisy samples, that is

$$\langle y(\boldsymbol{x}+\boldsymbol{n})\rangle \approx \sum_k v_k g(a_k(w_k^T\boldsymbol{x} - \theta_k)) \tag{22}$$

where the scaling constant a_k depends on the magnitude of the weight vector w_k and the noise variance

$$a_k = \frac{1}{\sqrt{||w_k||^2\sigma_1^2 + 1}}. \tag{23}$$

Note that the bias θ_k is not included in the weight vector and has no role in the computation of a_k. It is, however, scaled by a_k.

This does not say that we can train a network without jitter and then simply scale the sigmoids to compute exactly the network that would result from training with jitter because it does not account for the dynamics of training with random noise, but it does suggest some similarities. A connection can be drawn as follows. If the sample size is large enough and (unjittered) training is successful, then $y(\boldsymbol{x}) = t(\boldsymbol{x}) + \epsilon(\boldsymbol{x})$, where $\epsilon(\boldsymbol{x})$ represents a (hopefully) small residual error. For a fixed $\boldsymbol{x}$, taking the expected value of both sides over added input noise $\boldsymbol{n}$ gives

$$\langle y(\boldsymbol{x}+\boldsymbol{n})\rangle = \langle t(\boldsymbol{x}+\boldsymbol{n})\rangle + \langle \epsilon(\boldsymbol{x}+\boldsymbol{n})\rangle. \tag{24}$$

We can approximate $\langle y(\boldsymbol{x}+\boldsymbol{n})\rangle$ easily by scaling the weights: Let $\tilde{y}(\boldsymbol{x})$ denote the output of the scaled network that computes $y(\boldsymbol{x})$ before scaling; then, for a given input $\boldsymbol{x}$, the scaled network computes (assuming Gaussian input noise) $\tilde{y}(\boldsymbol{x}) \approx \langle y(\boldsymbol{x}+\boldsymbol{n})\rangle$. The term $\langle t(\boldsymbol{x}+\boldsymbol{n})\rangle$ is the effective target function when training with jittered inputs. (That is, when training with jitter, the effective target for a network given the input $\tilde{\boldsymbol{x}} = \boldsymbol{x}+\boldsymbol{n}$ is $\langle t(\tilde{\boldsymbol{x}}-\boldsymbol{n})\rangle$. Assuming a uniform $p_{\boldsymbol{x}}$, a symmetric $p_{\boldsymbol{n}}$, and ignoring boundary effects, the effective target for a network with input $\boldsymbol{x}$ is $t_{\text{eff}}(\boldsymbol{x}) \approx \langle t(\boldsymbol{x}+\boldsymbol{n})\rangle$.) Finally, if the residual error $\epsilon(\boldsymbol{x})$ is a high-frequency signal then it is substantially attenuated by the low-pass filtering (smoothing) effect of convolving with a Gaussian, $\langle \epsilon(\boldsymbol{x}+\boldsymbol{n})\rangle \approx 0$, which leaves

$$\tilde{y}(\boldsymbol{x}) \approx t_{\text{eff}}(\boldsymbol{x}). \tag{25}$$

Of course, if unjittered training is unsuccessful and the remaining error is large, then $\langle \epsilon(\boldsymbol{x}+\boldsymbol{n})\rangle \not\approx 0$, and the approximation will be poor.

Example 3: Fig. 4(a)–(f) verifies this scaling property. Fig. 4(a)–(b) shows the response of a network with two inputs, three GCDF hidden units, and a linear output unit. Fig. 4(c)–(d) shows the average response using spherically distributed Gaussian noise with $\sigma = 0.1$ and averaged over 2000 noisy samples per grid point. Fig. 4(e)–(f) shows the expected response computed by scaling the hidden units. The RMS error (on a 64 × 64 grid) between the averaged noisy response and the scaled expected response is 0.0145. The scaled expected response was computed in a few seconds; the average noisy response required approximately 20 hours on a 20 MHz 386 personal computer.

B. Relation to Weight Decay

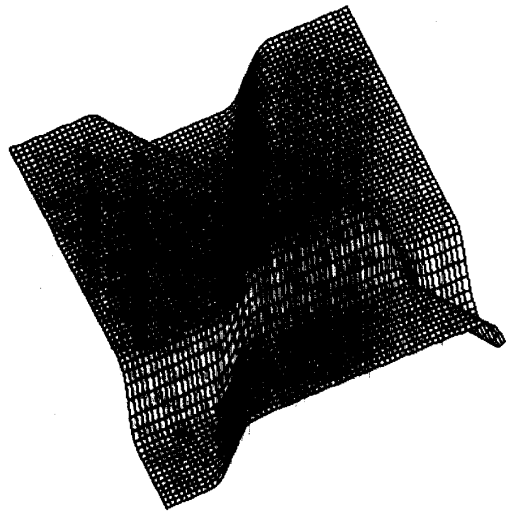
The scaling operation is equivalent to

$$w \rightarrow \frac{w}{\sqrt{||w||^2\sigma_1^2 + 1}}. \tag{26}$$

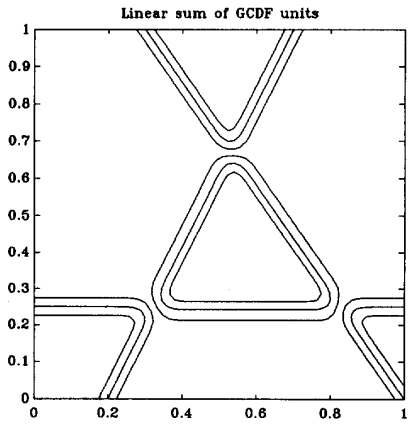
Since the denominator is not less than one, this always reduces the magnitude of w or leaves it unchanged. When $\sigma_1 = 0$ (no input noise), the weights are unchanged. When $\sigma_1 \rightarrow \infty, w \rightarrow 0$. When $||w||\sigma_1$ is small, the scaling has little effect. When $||w||\sigma_1$ is large, the scaling is approximately

$$w \rightarrow \frac{w}{||w||\sigma_1} \tag{27}$$

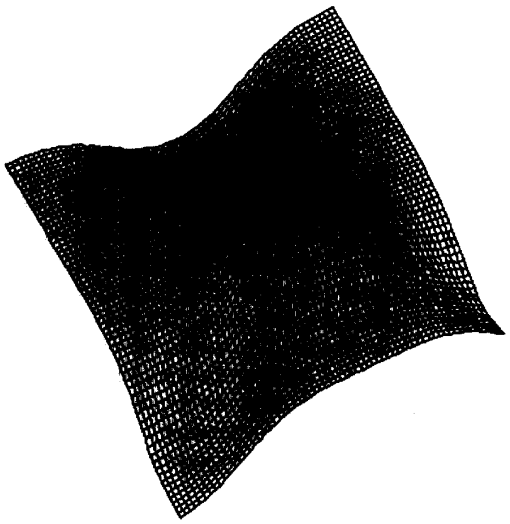
and the magnitude of w is reduced to approximately $1/\sigma_1$. This has some properties similar to weight decay [24], [35], [34], [28], another commonly used heuristic for improving generalization. Weight decay can be considered to be another form of regularization (e.g., [15]). The development of weight decay terms as a result of training single-layer linear perceptrons with input noise is shown in [9].



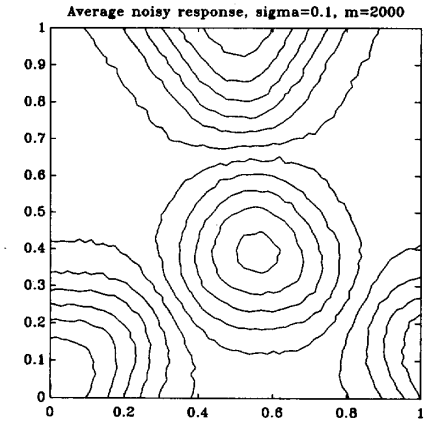
(a)



(b)



(c)



(d)

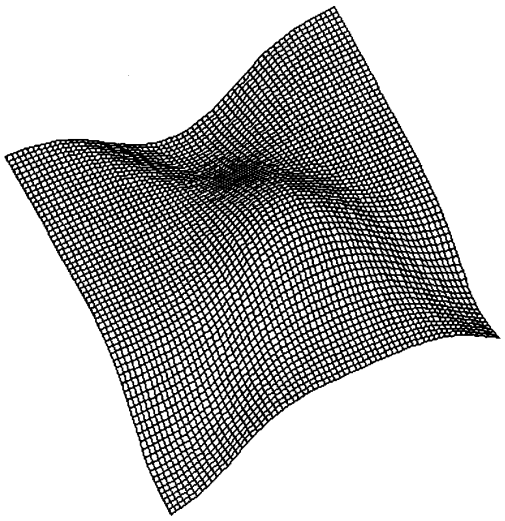
Fig. 4. Example 3. (a) The transfer function of the original network and (b) its contour plot. (c) The average response with additive Gaussian input noise, $\sigma = 0.1$, averaged over 2000 noisy samples per grid point and (d) its contour plot.

V. Extension to General Layered Neural Networks

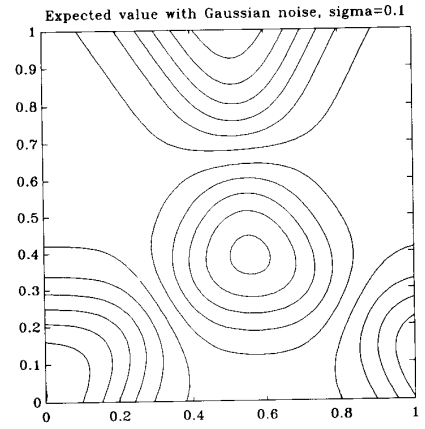
The results relating training with jittered data and error regularization hold for any network. The analysis for gain scaling, however, is valid only for networks with a single hidden layer and a linear output node. More general feedforward networks have multiple layers and nonlinear output nodes. Even though the invariance property does not hold for these networks, these results lend justification to the idea of gain scaling [16], [10] and weight decay as heuristics for improving generalization.

The gain scaling analysis uses a GCDF nonlinearity in place of the usual sigmoid nonlinearity, but these have very similar shapes so this is not an important difference in terms of representation capability. (Differences might be observed in training dynamics, however, because the GCDF has flatter tails.) The precise form of the sigmoid is not important as long as it is monotonic nondecreasing; the usual sigmoid is widely used because its derivative is easily calculated.

The GCDF nonlinearity is used here because it has a convenient shape invariance property under convolution with a Gaussian input noise density. There may be other nonlinearities that, while not having this shape invariance property, are such that their expected response can still be calculated reasonably efficiently using a similar approach. If, for example, $g(x) * p_{\boldsymbol{n}}(x) = h(x)$, the function $h(x)$ may be different in form from $g(x)$, but still reasonably easy to calculate. As a specific example, if $g(x)$ is a step function and $p_{\boldsymbol{n}}(x)$ is uniform (both in one dimension), then $h(x)$ is a semi-linear ramp function: zero for $x < \alpha$, equal to x for $-\alpha \leq x \leq \alpha$, and one for $x > \alpha$. The expected network response can then



(e)



(f)

Fig. 4. (*Continued*) Example 3. (e) The expected response computed by scaling and (f) its contour plot.

be computed as a linear sum of $h(x)$ nonlinearities rather than a linear sum of $g(x)$ nonlinearities. Although different nonlinearities are used in calculating the normal and expected responses, this should, in general, still be much faster than averaging over many presentations of noisy samples.

The scaling results can also be applied to radial basis functions [22], [21], [25] which generally use Gaussian probability density function (PDF) hidden units and a linear output summation. The convolution of two spherical Gaussian PDF's with variances σ_1^2 and σ_2^2 produces a third Gaussian PDF with variance $\sigma_3^2 = \sigma_1^2 + \sigma_2^2$, so the expected response of these networks to noise is easily calculated using similar shape invariant scaling.

VI. Discussion

Training with jitter, error regularization, gain scaling, and weight decay are all methods that have been proposed to improve generalization. Training with small amounts of jitter approaches the generalization problem directly by assuming that slightly different inputs give approximately the same output. If the noise distribution is smooth, the network will interpolate among training points in proportion to a smooth function of the distance to each training point.

With jitter, the effective target function is typically a smoothed version of the actual target. This is approximately equivalent to a smoothing regularization with the noise variance playing the role of the regularization parameter. Training with jitter thus allows regularization within the conventional layered perceptron architecture.

Although large networks generally learn rapidly, they tend to generalize poorly because of insufficient constraints. Training with jitter helps to prevent overfitting by providing additional constraints. The effective target function is a continuous function defined over the entire input space, whereas the original target function may be defined only at the specific training points. This constrains the network and forces it to use any excess degrees of freedom to approximate the smoothed target function rather than forming an arbitrarily complex boundary that just happens to fit the original training data (memorization). Even though the network may be large, it models a simpler system.

The expected effect of jitter can be calculated very efficiently in certain networks by a simple scaling of the sigmoid gains. This suggests the possibility of a post-training step to choose optimum gains based on cross-validation with a test set. This might make it possible to improve the generalization of large networks while retaining the advantage of fast learning.

The problem of choosing an appropriate noise variance has not been addressed here. Holmström and Koistinen [11], [13], [14] suggest several methods based on cross-validation. Considerable research has been done on the problem of selecting an appropriate λ for regularization, especially for linear models (e.g., [8]). Because of the relationship between training with jitter and regularization, the regularization research may be helpful in selecting an appropriate noise level.

Appendix

A. Small-Perturbation Approximation

For small noise amplitudes, the network output $y(\boldsymbol{x} + \boldsymbol{n})$ can be approximated by

$$y(\boldsymbol{x} + \boldsymbol{n}) \approx y(\boldsymbol{x}) + \left(\frac{\partial y}{\partial \boldsymbol{x}}\right)^T \boldsymbol{n} + \frac{1}{2}\boldsymbol{n}^T H \boldsymbol{n} \qquad (28)$$

where H is the Hessian matrix with element $h_{ij} = \partial^2 y/(\partial x_i \partial x_j)$. Assuming an even noise distribution so that $\langle n^k \rangle = 0$ for k odd, one can write

$$\mathcal{E} \approx \{(t-y)^2\} + \sigma^2 \left\{ \left|\left| \frac{\partial y}{\partial \boldsymbol{x}} \right|\right|^2 \right\}$$

$$+\left\{\sigma^2(y-t)Tr(H)+\frac{\sigma^4}{4}Tr(H)^2+\frac{\sigma^4}{2}Tr(H^2)+\frac{m_4-3\sigma^4}{4}\left(\sum_i h_{ii}^2\right)\right\} \quad (29)$$

where m_4 is the fourth moment $\langle n^4\rangle$. Dropping all terms higher than second order in σ gives

$$\mathcal{E}\approx\{(t-y)^2\}+\sigma^2\{(y-t)Tr(H)\}+\sigma^2\left\{\left|\left|\frac{\partial y}{\partial \boldsymbol{x}}\right|\right|^2\right\} \quad (30)$$

and when H is assumed to be zero, this reduces to (15). The Laplacian term, $Tr(H)=\nabla^2 y$, omitted in (15), can be described as an approximate measure of the difference between the average surrounding values and the precise value of the field at a point [5]. The third term in (30) is the first order regularization term in (15).

Training with nonjittered data simply minimizes the error at the training points and puts no constraints on the function at other points. In contrast, training with jitter minimizes the error while also forcing the the approximating function to have small derivatives and a local average that approaches the target in the vicinity of each training point.

B. CDF-PDF Convolution in n Dimensions

The following shows that the convolution of an n-dimensional spherical Gaussian probability density function (PDF) and a Gaussian cumulative distribution function (CDF) results in another Gaussian CDF.

Let $f_1(\boldsymbol{x})$ be a spherical Gaussian PDF in n-dimensions

$$f_1(\boldsymbol{x})=\frac{1}{\sigma_1^n(2\pi)^{n/2}}\exp\left(\frac{-||\boldsymbol{x}||^2}{2\sigma_1^2}\right) \quad (31)$$

and let $F_2(\boldsymbol{x})$ be a Gaussian CDF of the form

$$F_2(\boldsymbol{x})=\int_{-\infty}^{w^T\boldsymbol{x}}\frac{1}{\sqrt{2\pi}}\exp\left(\frac{-\tau^2}{2}\right)d\tau. \quad (32)$$

This can be written as

$$F_2(\boldsymbol{x})=\int_{-\infty}^{\hat{w}^T\boldsymbol{x}}\frac{1}{\sigma_2\sqrt{2\pi}}\exp\left(\frac{-\tau^2}{2\sigma_2^2}\right)d\tau \quad (33)$$

where $\hat{w}=w/||w||$ and $o_2=1/||w||$.

The convolution of F_2 and f_1 is the n-dimensional integral

$$F_2(\boldsymbol{x})*f_1(\boldsymbol{x})=\int_\alpha F_2(\alpha)f_1(\boldsymbol{x}-\alpha)\,d\alpha. \quad (34)$$

Separate $\boldsymbol{x}$ and α into components parallel and orthogonal to $\hat{w}$

$$\begin{aligned}\boldsymbol{x}&=\ell\hat{w}+\gamma\\ \ell&=\hat{w}^T\boldsymbol{x}\\ \hat{w}^T\gamma&=0\\ \alpha&=k\hat{w}+\beta\\ k&=\hat{w}^T\alpha\\ \hat{w}^T\beta&=0\\ ||\boldsymbol{x}-\alpha||^2&=(\ell-k)^2||\hat{w}||+2(\ell-k)\hat{w}^T(\gamma-\beta)+||\gamma-\beta||^2\\ &=(\ell-k)^2+||\gamma-\beta||^2\end{aligned}$$

where ℓ and k are scalars and γ and β are n-dimensional vectors orthogonal to $\hat{w}$. Then

$$\begin{aligned}F_2(\alpha)&=\int_{-\infty}^{\hat{w}^T\alpha}\frac{1}{\sigma_2\sqrt{2\pi}}\exp\left(\frac{-\tau^2}{2\sigma_2^2}\right)d\tau\\ &=\int_{-\infty}^{k}\frac{1}{\sigma_2\sqrt{2\pi}}\exp\left(\frac{-\tau^2}{2\sigma_2^2}\right)d\tau\\ f_1(\boldsymbol{x}-\alpha)&=\frac{1}{\sigma_1^n(2\pi)^{n/2}}\exp\left(\frac{-||\boldsymbol{x}-\alpha||^2}{2\sigma_1^2}\right)\\ &=\frac{1}{\sigma_1^n(2\pi)^{n/2}}\exp\left(\frac{-(\ell-k)^2}{2\sigma_1^2}\right)\exp\left(\frac{-||\gamma-\beta||^2}{2\sigma_1^2}\right)\\ &=\frac{1}{\sigma_1\sqrt{2\pi}}\exp\left(\frac{-(\ell-k)^2}{2\sigma_1^2}\right)\cdot\frac{1}{\sigma_1^{n-1}(2\pi)^{(n-1)/2}}\\ &\quad\cdot\exp\left(\frac{-||\gamma-\beta||^2}{2\sigma_1^2}\right)\end{aligned} \quad (35)$$

and

$$\begin{aligned}F_2(\boldsymbol{x})*f_1(\boldsymbol{x})&=\int_k\int_\beta\int_{-\infty}^{k}\frac{1}{\sigma_2\sqrt{2\pi}}e^{-\tau^2/(2\sigma_2^2)}\,d\tau\\ &\quad\cdot\left(\frac{1}{\sigma_1\sqrt{2\pi}}e^{-(\ell-k)^2/(2\sigma_1^2)}\right)\\ &\quad\times\left(\frac{1}{\sigma_1^{n-1}(2\pi)^{(n-1)/2}}e^{-||\gamma-\beta||^2/(2\sigma_1^2)}\right)\\ &\quad d\beta\,dk\\ &=\int_k\int_{-\infty}^{k}\frac{1}{\sigma_2\sqrt{2\pi}}e^{-\tau^2/(2\sigma_2^2)}\,d\tau\\ &\quad\cdot\left(\frac{1}{\sigma_1\sqrt{2\pi}}e^{-(\ell-k)^2/(2\sigma_1^2)}\right)dk\\ &\quad\times\int_\beta\frac{1}{\sigma_1^{n-1}(2\pi)^{(n-1)/2}}e^{-||\gamma-\beta||^2/(2\sigma_1^2)}\,d\beta\\ &=\int_k\int_{-\infty}^{k}\frac{1}{\sigma_2\sqrt{2\pi}}e^{-\tau^2/(2\sigma_2^2)}\,d\tau\\ &\quad\cdot\left(\frac{1}{\sigma_1\sqrt{2\pi}}e^{-(\ell-k)^2/(2\sigma_1^2)}\right)dk.\end{aligned}$$

Thus $F_2(\boldsymbol{x})*f_1(\boldsymbol{x})$ reduces to a one-dimensional convolution of a Gaussian CDF with standard deviation $\sigma_2=1/||w||$ and

a Gaussian PDF with standard deviation σ_1. It can be shown (see Appendix C) that this is a Gaussian CDF with variance $\sigma_3^2 = \sigma_1^2 + \sigma_2^2$.

Letting Z_a denote the Gaussian CDF function with standard deviation a

$$\begin{aligned} F_2(\boldsymbol{x}) * f_1(\boldsymbol{x}) &= Z_{\sigma_2}(\ell) * g(\ell) \\ &= Z_{\sigma_3}(\ell) \\ &= Z_1\left(\frac{\ell}{\sigma_3}\right) \\ &= Z_1\left(\frac{\hat{w}^T\boldsymbol{x}}{\sqrt{\sigma_1^2 + (1/||w||)^2}}\right) \\ &= Z_1\left(\frac{||w||\hat{w}^T\boldsymbol{x}}{\sqrt{||w||^2\sigma_1^2 + 1}}\right) \\ &= Z_1\left(\frac{w^T\boldsymbol{x}}{\sqrt{||w||^2\sigma_1^2 + 1}}\right) \\ &= F_2\left(\frac{\boldsymbol{x}}{\sqrt{||w||^2\sigma_1^2 + 1}}\right). \end{aligned} \tag{36}$$

Thus, the convolution of a Gaussian CDF and a Gaussian PDF can be computed by a simple scaling of the original CDF.

C. CDF-PDF Convolution in One Dimension

The following [1] demonstrates that the convolution of Gaussian PDF with variance σ_1^2 and a Gaussian CDF with variance σ_2^2 results in a Gaussian CDF with variance $\sigma_3^2 = \sigma_1^2 + \sigma_2^2$. All the functions are one dimensional.

Consider two independent random variables X_1 and X_2 with PDF's f_1 and f_2 and CDF's F_1 and F_2. The random variable $Y = X_1 + X_2$ has the PDF $f_1 * f_2$ and consequently its CDF is $F_1 * f_2 = f_1 * F_2$. Let X_1 and X_2 be zero mean Gaussian, $X_1 \sim N(0, \sigma_1^2)$ and $X_2 \sim N(0, \sigma_2^2)$, then, clearly, $Y \sim N(0, \sigma_1^2 + \sigma_2^2)$ has a Gaussian PDF with variance $\sigma_3^2 = \sigma_1^2 + \sigma_2^2$. Since Y has the CDF $f_1 * F_2$, $f_1 * F_2$ is a Gaussian CDF with zero mean and variance $\sigma_3^2 = \sigma_1^2 + \sigma_2^2$.

ACKNOWLEDGMENT

The authors express their appreciation to the anonymous reviewers for their helpful criticisms and for the proof in Appendix C which is much simpler than the original. The authors also thank Boeing Computer Services and Washington Technology Center.

REFERENCES

[1] Anonymous reviewer, personal communication, 1993.
[2] C. M. Bishop, "Curvature-driven smoothing in backpropagation neural networks," in *Proc. Int. Neural Network Conf.*, Paris, 1990, pp. 749–752.
[3] ——, "Curvature-driven smoothing: A learning algorithm for feed-forward networks," *IEEE Trans. Neural Networks*, vol. 4, no. 5, pp. 882–884, 1993.
[4] ——, "Improving the generalization properties of radial basis function neural networks," *Neural Computation*, vol. 3, no. 4, pp. 579–588, 1991.
[5] H. F. Davis and A. D. Snider, *Introduction to Vector Analysis*4th ed. Boston, MA: Allyn and Bacon, Inc., 1979.
[6] H. Drucker and Y. Le Cun, "Double backpropagation and increasing generalization performance." in *Proc. Int. Joint Conf. Neural Networks*, Seattle, vol. II, pp. 145–150, 1991.
[7] J. L. Elman and D. Zipser, "Learning the hidden structure of speech," *J. Acoust. Soc. Amer.*, vol. 83, no. 4, pp. 1615–1626, 1988.
[8] N. P. Galatsanos and A. K. Katsaggelos, "Methods for choosing the regularization parameter and estimating the noise variance in image restoration and their relation," *IEEE Trans. Image Process.*, vol. 1, no. 3, pp. 322–336, 1992.
[9] J. A. Hertz and A. Krogh, "Statistical dynamics of learning," in *Artificial Neural Networks (ICANN-91)*, T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, Eds. Amsterdam: Elsevier, 1991, vol. 1, pp. 125–131.
[10] M. Hoehfeld and S. E. Fahlman, "Learning with limited numerical precision using the cascade-correlation algorithm," *IEEE Trans. on Neural Networks*, vol. 3, no. 4, pp. 602–611, 1992.
[11] L. Holmström and P. Koistinen, "Using additive noise in back-propagation training," *IEEE Trans. Neural Networks*, vol. 3, no. 1, pp. 24–38, Jan. 1992.
[12] R. J. Marks, II, *Introduction to Shannon Sampling and Interpolation Theory*. New York: Springer-Verlag, 1991.
[13] P. Koistinen and L. Holmström, "Kernel regression and backpropagation training with noise," in *Proc. Int. Joint Conf. Neural Networks (Singapore)*, 1991, pp. 367–372.
[14] ——, "Kernel regression and backpropagation training with noise," in *Advances in Neural Information Processing (4)*, J. E. Moody, S. J. Hanson, and R. P. Lippmann, Eds. 1992, pp. 1035–1039.
[15] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Advances in Neural Information Processing Systems (4)*, J. E. Moody, S. J. Hanson, and R. P. Lippmann, Eds. 1992 pp. 950–957.
[16] J. K. Kruschke, "Creating local and distributed bottlenecks in hidden layers of backpropagation networks," in *Proc. 1988 Connectionist Models Summer School*, D. Touretzky, G. Hinton, and T. Sejnowski, Eds., 1989, pp. 120–126.
[17] A. Linden and J. Kindermann, "Inversion of multilayer nets," in *Proc. Int. Joint Conf. Neural Networks*, Washington D.C., vol. II, 1989, pp. 425–430.
[18] K. Matsuoka, "An approach to generalization problem in backpropagation learning," in *Proc. Int. Neural Network Conf.*, (Paris), vol. 2, pp. 765–768, 1990.
[19] ——, "Noise injection into inputs in backpropagation learning," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 3, 1992, pp. 436–440.
[20] J. I. Minnix, "Fault tolerance of the backpropagation neural network trained on noisy inputs," in *Proc. Int. Joint Conf. Neural Networks (Baltimore)*, vol. I, pp. 847–852, 1992.
[21] J. Moody and C. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, pp. 281–294, 1989.
[22] ——, "Learning with localized receptive fields. In *Proc. 1988 Connectionist Models Summer School*, pp. 133–143, 1988.
[23] S. Oh, R. J. Marks, II, and M. A. El-Sharkawi, "Query based learning in a multilayered perceptron in the presence of data jitter," in *Applications of Neural Networks to Power Systems*, M. A. El-Sharkawi and R. J. Marks, II, Eds. New York: IEEE Press, 1991, pp. 72–75.
[24] D. C. Plaut, S. J. Nowlan, and G. E. Hinton, "Experiments on learning by back propagation", Carnegie-Mellon Univ., Tech. Rep. CMU-CS-86-126, 1986.
[25] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proc. IEEE*, vol. 78, no. 9, pp. 1481–1497, 1990.
[26] R. Reed, R. J. Marks, II, and S. Oh, "An equivalence between sigmoidal gain scaling and training with noisy (jittered) input data," in *Proc. RNNS/IEEE Symp. Neuroinformatics Neurocomputing*, Rostov-on-Don, Russia, 1992.
[27] R. Reed, S. Oh, and R. J. Marks, II, "Regularization using jittered training data," in *Proc. Int. Joint Conf. Neural Networks*, Baltimore, vol. III, 1992, pp. 147–152.
[28] R. D. Reed, "Pruning algorithms—A survey," *IEEE Trans. Neural Networks*, vol. 4, no. 5, pp. 740–747, 1993.
[29] Ph. Refregier and J.-M. Vignolle, "An improved version of the pseudo-inverse solution for classification and neural networks," *Europhysics Lett.*, vol. 10, no. 4, pp. 387–392, 1989.
[30] C. H. Séquin and R. D. Clay, "Fault tolerance in feed-forward artificial neural networks," in *Neural Networks: Concepts, Applications, and Implementations*, P. Antognetti and V. Milutinović, Eds. Englewood Cliffs, NJ: Prentice-Hall, vol. IV, 1991, pp. 111–141.
[31] J. Sietsma and R. J. F. Dow, "Creating artificial neural networks that generalize," *Neural Networks*, vol. 4, no. 1, pp. 67–79, 1991.
[32] ——, "Neural net pruning—Why and how," in *Proc. IEEE Int. Conf. Neural Networks (San Diego)*, vol. I, 1988, pp. 325–333.
[33] A. R. Webb, "Functional approximation by feed-forward networks: A least-squares approach to generalization," R.S.R.E. Memorandum 4453, 1991.
[34] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman, "Generalization

by weight-elimination applied to currency exchange rate prediction," in *Proc. Int. Joint Conf. Neural Networks*, Seattle, vol. I, 1991, pp. 837–841.

[35] ——, "Generalization by weight-elimination with application to forecasting," in *Advances in Neural Information Processing (3)*, R. Lippmann, J. Moody, and D. Touretzky, Eds., 1991, pp. 875–882.

Russell Reed (M'95) received the B.S. and M.S. degrees in electrical engineering from Texas A&M University, College Station, TX, in 1981 and 1986. At present, he is a doctoral candidate in electrical engineering at the University of Washington, Seattle, WA.

From 1986 to 1990 he worked at World Instruments in Longview, Texas, designing microprocessor based instruments and PC software. His research interests include adaptive/learning systems, pattern recognition, and neural networks.

Mr. Reed is a member of the INNS.

Robert J. Marks II (S'76–M'77–SM'83–F'94) received the B.S. and M.S. degrees in electrical engineering from Rose–Hulman Institute of Technology, Terre Haute, IN, in 1972 and 1973, respectively. He received the Ph.D. degree in electrical engineering from Texas Tech University, Lubbock, TX, in 1977.

He is a Professor in the Department of Electrical Engineering at the University of Washington, Seattle.

He was awarded the Outstanding Branch Councilor award in 1982 by IEEE and, in 1984, was presented with an IEEE Centennial Medal. He was named a Distinguished Young Alumnus of Rose-Hulman Institute of Technology in 1992 and, in 1993, was inducted into the Texas Tech Electrical Engineering Academy.

He was Chair of IEEE Neural Networks Committee (1989) and served as the first President of the IEEE Neural Networks Council (1990–1991). In 1992, he was given the honorary title of Charter President. He was named an IEEE Distinguished Lecturer in 1992. He is a Fellow of the Optical Society of America. He was the co-founder and first President of the Puget Sound Section of the Optical Society of America and was elected that organization's first Honorary Member. He is co-founder and current President of Multidimensional Systems Corporation in Lynnwood, Washington and is a founder of Financial Neural Networks, Inc. in Kirkland, WA. He is the Editor-in-Chief of the IEEE TRANSACTIONS ON NEURAL NETWORKS (1992–present) and serves as an Associate Editor of the IEEE TRANSACTIONS ON FUZZY SYSTEMS (1993–present). He serves on the Editorial Board of the *Journal of Intelligent Control, Neurocomputing, and Fuzzy Logic* (1992–present). He was also the topical editor for *Optical Signal Processing and Image Science* for the *Journal of the Optical Society on America—A Neurocomputing* (1989–1992).

Dr. Marks served as North American Liasion for the 1991 Singapore International Joint Conference on Neural Networks (IJCNN), International Chair of the 1992 RNNS/IEEE Symposium on Neuroinformatics and Neurocomputing (Rostov-on-Don, USSR) and Organizational Chair for both the 1993 IEEE Virtual Reality Annual International Symposium (VRAIS) in Seattle and the IEEE-SP International Symposium on Time-Frequency and Time-Scale Analysis (Victoria, BC, 1992). He also served as the Program and Tutorials Chair for the First International Forum on Applications of Neural Networks to Power Systems (Seattle, 1991). He was elected to the Board of Governors of the IEEE Circuits and Systems Society (1993–1996) and was the co-founder and first Chair of the IEEE Circuits & Systems Society Technical Committee on Neural Systems & Applications. He is the General Chair of the 1995 International Symposium on Circuits and Systems, Seattle. He is also the Technical Program Director for the first IEEE World Congress on Computational Intelligence, Orlando, FL, July 1994. Five of his papers have been reproduced in volumes of collections of outstanding papers. He has two US patents in the field of artificial neural networks. He is the author of the book *Introduction to Shannon Sampling and Interpolation Theory* (Springer–Verlag, 1991) and is editor of the companion volume, *Advanced Topics in Shannon Sampling and Interpolation Theory* (Springer–Verlag, 1993).

Seho Oh received the B.S. degree in electronics engineering from Seoul National University and the M.S. degree in electrical engineering from Korea Advanced Institute of Science and Technology, Seoul. He received the Ph.D. degree in electrical engineering from the University of Washington, Seattle, in 1989.

From 1981 through 1986, he was in Central Research Laboratory of Goldstar Company. Currently, he is in Neopath Inc. His research interests are in the area of signal analysis, generalized time frequency analysis, artificial neural networks, fuzzy systems, and pattern recognition.

Dr. Oh is the co-author of over 40 archival and proceedings papers and has been issued two United States patents.