R. Reed, R.J. Marks II and S.Oh, "An equivalence between sigmoidal gain scaling scaling and training with noisy (jittered) input data", Proceedings of the RNNS/IEEE Symposium on Neuroinformatics and Neurocomputing, (Rostov-on-Don, Russia, October, 1992), pp. 120-127, IEEE

An Equivalence between Sigmoidal Gain Scaling and Training with Noisy (Jittered) Input Data

Russell Reed, Robert J. Marks II, Seho Oh
Dept. of Electrical Engineering, FT-10
University of Washington, Seattle, WA, 98195

Abstract

Training with additive input noise (jitter) is a commonly used heuristic for improving generalization in layered perceptron artificial neural networks. One result of training with jitter is that the effective target function is the convolution of the actual target and the noise density. For many noise densities, this is approximately equivalent to a smoothing regularization. A drawback of training with jitter, in comparison with the unjittered case, is that many more sample presentations are required in order to average over the noise and estimate the expected response. In this paper, we demonstrate that the expected effect of jitter can be computed, in certain cases, by a simple scaling of the sigmoid nonlinearities. Application of this technique to a single–hidden–layer perceptron with a linear output is considered.

1 Introduction

Many studies (e.g., [9,8,5,6]) have noted the benefits of adding small amounts of noise to the inputs during training. Holmström and Koistinen [1,2,3] have shown that the approach is consistent, i.e., 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.

In a previous paper [7], we have shown that training with noisy (jittered) input data has the result that the effective target function is the convolution of the actual target with the noise density (see section A.1). For many noise densities, this is a smoothing operation. Training with jitter is also approximately equivalent to a regularization which favors smooth solutions. Regularization is helpful in improving generalization when the network is underconstrained —when the number of parameters to be determined is much larger than the number of training samples —and helps to prevent overtraining.

A drawback of training with jitter is that many more sample presentations are required in order to average over the noise and estimate the expected response. In this paper, we demonstrate that the expected effect of jitter can be computed, in certain cases, by a simple scaling of the sigmoid nonlinearities. This means that the benefits of training with noise can be obtained without the computational cost of averaging over many noisy samples. These results provide justification for gain scaling as a heuristic for improving generalization.

2 Linear Output Networks

Consider the function

$$y(\mathbf{x}) = \sum_k v_k \, g\left(w_k^T \mathbf{x} - \theta_k\right) \tag{1}$$

where $g(\cdot)$ is a sigmoid nonlinearity (monotonic nondecreasing). This describes a single–hidden–layer network with a linear output.

With jitter, the expected output for a fixed input $\mathbf{x}$ is (see section A.1)

$$\begin{aligned} \langle y(\mathbf{x}+\mathbf{n}) \rangle &= y(\mathbf{x}) \star p_n(\mathbf{x}) \\ &= \sum_k v_k \, g_k(\mathbf{x}) \star p_n(\mathbf{x}) \end{aligned}$$

$$= \sum_k v_k \, g_k(\mathbf{x}) * p_n(\mathbf{x}), \tag{2}$$

i.e., a linear sum of convolutions of the hidden unit responses with the noise density.

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, then the shape of the nonlinearity will be invariant to convolution with a Gaussian input noise density. That is, if we assume that the noise is zero–mean Gaussian and spherically distributed in N dimensions

$$p_n(\mathbf{x}) = \frac{1}{\sigma_1^N (2\pi)^{N/2}} \exp\left(\frac{-\|\mathbf{x}\|^2}{2\sigma_1^2}\right) \tag{3}$$

and the g nonlinearity is the Gaussian cumulative distribution function (GCDF)

$$g(\mathbf{x}) = \int_{-\infty}^{w^T\mathbf{x}-\theta} \frac{1}{\sigma_2\sqrt{2\pi}} \exp\left(\frac{-\tau^2}{2\sigma_2^2}\right) d\tau. \tag{4}$$

then the convolution can be replaced by a simple scaling operation; i.e.,

$$g(w^T\mathbf{x} - \theta) * p_n(\mathbf{x}) = g\left(a\left(w^T\mathbf{x} - \theta\right)\right). \tag{5}$$

A derivation is given in the Appendix (section A.2).

3 Significance

The significance of this is that, if the equivalence (5) holds, then, for the network considered, the expected response of the network to input noise can be computed exactly by simply scaling the hidden unit nonlinearities

$$\langle y(\mathbf{x}+\mathbf{n})\rangle = \sum_k v_k \, g\left(a_k\left(w_k^T\mathbf{x} - \theta_k\right)\right) \tag{6}$$

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

$$a_k = \frac{1}{\sqrt{\|w_k\|^2\sigma_1^2 + 1}}. \tag{7}$$

(Note that the threshold 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.)

Thus, if we have a network of this type that performs a particular function, we can compute its expected response to Gaussian input noise by simply scaling the hidden unit nonlinearities appropriately; we don't have to go through the time–consuming process of estimating the response by averaging over many noisy samples.

Simulation

Figs. 1 and 2 verify this scaling property. 3–D plots are shown in Fig. 1 and contour plots are shown in Fig. 2. Fig. 1(a) shows the response of a network with two inputs, three GCDF hidden units, and a linear output unit. Fig. 1(b) shows the average response using spherically distributed Gaussian noise with $\sigma = 0.1$ and averaged over 2000 noisy samples per grid point. Fig. 1(c) shows the expected response computed by scaling the hidden units. The RMS error (on a 64 x 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 20MHz 386 personal computer.

The scaling operation is equivalent to

$$w \rightarrow \frac{w}{\sqrt{\|w\|^2\sigma_1^2 + 1}} \tag{8}$$

Since the denominator is not less than 1, 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 $\|w\|$ is small, the scaling has little effect. When $\|w\|$ is large, the scaling is approximately

$$w \rightarrow \frac{w}{\|w\|\sigma_1}. \tag{9}$$

This has some properties similar to weight decay, another commonly used heuristic for improving generalization.

4 Extension to General Layered Neural Networks

The network considered here has a single hidden layer and a linear output node; its computational capabilities are limited. More general feed–forward networks have multiple layers and nonlinear output nodes. The invariance property does not hold in this case, but these results lend justification to the idea of gain scaling [4] and weight decay as heuristics for improving generalization. A single–hidden–layer network with a nonlinear output node has qualitative properties similar to the network considered here if the output nonlinearity is not too sharp.

The network considered here uses a GCDF nonlinearity in place of the usual sigmoid nonlinearity, but these have very similar shapes so this is not an important difference. 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 $g(x) * p_n(x) = h(x)$, for example, the function $h(x)$ may be different in form from $g(x)$, but still reasonably easy to calculate. If $g(x)$ is a step function and $p_n(x)$ is uniform (in one dimension), then $h(x)$ is a semi–linear ramp function, for example. The expected network response can then 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.

A Appendix

A.1 Convolution Property

Consider a network trained with noisy input data, $\{\mathbf{x}+\mathbf{n}, t(\mathbf{x})\}$, where $\mathbf{n}$ is noise that varies with each presentation.

During training, the network sees the target $t(\mathbf{x})$ in conjunction with the input $\tilde{\mathbf{x}} = \mathbf{x} + \mathbf{n}$. The input $\tilde{\mathbf{x}}$ can be produced by various combinations of desired inputs $\mathbf{x}$ and noises $\mathbf{n}$, so the net learns to produce $\langle t(\tilde{\mathbf{x}} - \mathbf{n}) \,|\, \tilde{\mathbf{x}} \rangle$. If the training data covers the entire input space, the effective target is the convolution of the clean target $t(\mathbf{x})$ and the noise density $p_n(\mathbf{x})$

$$\begin{aligned} \langle t(\tilde{\mathbf{x}} - \mathbf{n}) \,|\, \tilde{\mathbf{x}} \rangle &= \int_{\mathbf{n}} t(\tilde{\mathbf{x}} - \mathbf{n}) p_n(\mathbf{n})\, d\mathbf{n} \\ &= t(\tilde{\mathbf{x}}) * p_n(\tilde{\mathbf{x}}). \end{aligned} \tag{10}$$

For many centrally distributed noise densities, this is a smoothing operation. If $t(\mathbf{x})$ is a step function and $p_n(\mathbf{x})$ is Gaussian, for example, then $\langle t(\tilde{\mathbf{x}} - \mathbf{n}) \,|\, \tilde{\mathbf{x}} \rangle$ is the Gaussian cumulative distribution, which

is a smooth function similar to the sigmoid. For symmetric noise densities, convolution and correlation are equivalent; i.e., $\langle t(\mathbf{x}-\mathbf{n})\rangle = \langle t(\mathbf{x}+\mathbf{n})\rangle$.

A.2 CDF–PDF Convolution in n Dimensions

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

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

$$f_1(\mathbf{x}) = \frac{1}{\sigma_1^n (2\pi)^{n/2}} \exp\left(\frac{-\|\mathbf{x}\|^2}{2\sigma_1^2}\right) \tag{11}$$

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

$$F_2(\mathbf{x}) = \int_{-\infty}^{w^T\mathbf{x}} \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-\tau^2}{2}\right) d\tau. \tag{12}$$

This can be written as

$$F_2(\mathbf{x}) = \int_{-\infty}^{\hat{w}^T\mathbf{x}} \frac{1}{\sigma_2\sqrt{2\pi}} \exp\left(\frac{-\tau^2}{2\sigma_2^2}\right) d\tau \tag{13}$$

where $\hat{w} = w/\|w\|$ and $\sigma_2 = 1/\|w\|$.

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

$$F_2(\mathbf{x}) * f_1(\mathbf{x}) = \int_\alpha F_2(\alpha) f_1(\mathbf{x}-\alpha)\, d\alpha, \tag{14}$$

but $F_2(\mathbf{x})$ is effectively one–dimensional in that it depends only on the projection of $\mathbf{x}$ onto w. Separate $\mathbf{x}$ and α into components parallel and orthogonal to $\hat{w}$

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

where ℓ and k are scalars and γ and β are $(n-1)$–dimensional vectors orthogonal to $\hat{w}$.

Then

$$\begin{aligned}
F_2(\mathbf{x}) &= \int_{-\infty}^{\hat{w}^T\mathbf{x}} \frac{1}{\sigma_2\sqrt{2\pi}} \exp\left(\frac{-\tau^2}{2\sigma_2^2}\right) d\tau \\
&= \int_{-\infty}^{k} \frac{1}{\sigma_2\sqrt{2\pi}} \exp\left(\frac{-\tau^2}{2\sigma_2^2}\right) d\tau \\
f_1(\mathbf{x}-\alpha) &= \frac{1}{\sigma_1^n(2\pi)^{n/2}} \exp\left(\frac{-\|\mathbf{x}-\alpha\|^2}{2\sigma_1^2}\right) \\
&= \frac{1}{\sigma_1^n(2\pi)^{n/2}} \exp\left(\frac{-(\ell-k)^2}{2\sigma_1^2}\right) \exp\left(\frac{-\|\gamma-\beta\|^2}{2\sigma_1^2}\right) \\
&= \frac{1}{\sigma_1\sqrt{2\pi}} \exp\left(\frac{-(\ell-k)^2}{2\sigma_1^2}\right) \cdot \frac{1}{\sigma_1^{n-1}(2\pi)^{(n-1)/2}} \exp\left(\frac{-\|\gamma-\beta\|^2}{2\sigma_1^2}\right)
\end{aligned} \tag{15}$$

and

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

Thus $F_2(\mathbf{x}) * f_1(\mathbf{x})$ reduces to a one–dimensional convolution of a Gaussian CDF with $\sigma_2 = 1/\|w\|$ and a Gaussian PDF with standard deviation σ_1. It can be shown (see section A.3) that this is a Gaussian CDF with standard deviation $\sigma_3 = \sqrt{\sigma_1^2 + \sigma_2^2}$.

Letting Z_a denote the Gaussian CDF function with standard deviation a,

$$\begin{aligned}
F_2(\mathbf{x}) * f_1(\mathbf{x}) &= Z_{\sigma_3}(\ell) * g(\ell) \\
&= Z_{\sigma_3}(\ell) \\
&= Z_1\left(\frac{\ell}{\sigma_3}\right) \\
&= Z_1\left(\frac{\hat{w}^T\mathbf{x}}{\sqrt{\sigma_1^2 + (1/\|w\|)^2}}\right) \\
&= Z_1\left(\frac{\|w\|\,\hat{w}^T\mathbf{x}}{\sqrt{\|w\|^2\sigma_1^2 + 1}}\right) \\
&= Z_1\left(\frac{w^T\mathbf{x}}{\sqrt{\|w\|^2\sigma_1^2 + 1}}\right) \\
&= F_2\left(\frac{\mathbf{x}}{\sqrt{\|w\|^2\sigma_1^2 + 1}}\right)
\end{aligned} \tag{16}$$

A.3 CDF–PDF Convolution in One Dimension

The following 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.

$f_1(x)$ is the Gaussian PDF

$$f_1(x) = \frac{1}{\sigma_1\sqrt{2\pi}} e^{-x^2/(2\sigma_1^2)} \tag{17}$$

and has the Fourier transform

$$F_1(u) = e^{-(2\pi\sigma_1)^2u^2/2}. \tag{18}$$

(The notation in this section is independent of the other sections. $F_2(u)$ here is different from $F_2(\mathbf{x})$ in the previous section.)

$f_2(x)$ is another Gaussian PDF with variance σ_2^2 and $g_2(x)$ is the CDF

$$g_2(x) = \int_{-\infty}^{x} f_2(\tau)\, d\tau \tag{19}$$

and has the transform

$$\begin{aligned} G_2(u) &= \frac{F_2(u)}{j2\pi u} + \frac{1}{2}F_2(0)\delta(u) \\ &= \frac{e^{-(2\pi\sigma_2)^2u^2/2}}{j2\pi u} + \frac{1}{2}\delta(u). \end{aligned} \tag{20}$$

The transform of the convolution is the product of the transforms

$$\begin{aligned} g_3(x) &= g_2(x) * f_1(x) \\ G_3(u) &= G_2(u)F_1(u) \\ &= \left[\frac{e^{-(2\pi\sigma_2)^2u^2/2}}{j2\pi u} + \frac{1}{2}\delta(u)\right] \cdot e^{-(2\pi\sigma_1)^2u^2/2} \\ &= e^{-(2\pi)^2(\sigma_1^2+\sigma_2^2)u^2/2} + \frac{1}{2}\delta(u) \end{aligned}$$

Taking the inverse transform of $G_3(u)$ gives

$$g_3(x) = \int_{-\infty}^{x} \frac{1}{\sigma_3^2\sqrt{2\pi}} e^{-x^2/(2\sigma_3^2)}\, dx \tag{21}$$

which is another Gaussian CDF with variance $\sigma_3^2 = \sigma_1^2 + \sigma_2^2$.

References

[1] L. Holmström and P. Koistinen. Using additive noise in back-propagation training. *IEEE Transactions on Neural Networks*, 3(1):24–38, Jan. 1992.

[2] P. Koistinen and L. Holmström. Kernel regression and backpropagation training with noise. In *Proceedings of the International Joint Conference on Neural Networks*, pages 367–372, 1991. (Singapore).

[3] P. Koistinen and L. Holmström. Kernel regression and backpropagation training with noise. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing (4)*, pages 1035–1039, 1992.

[4] J. K. Kruschke. Creating local and distributed bottlenecks in hidden layers of back-propagation networks. In D. Touretzky, G. Hinton, and T. Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 120–126, Morgan Kaufmann Publishers, 1988.

[5] A. Linden and J. Kindermann. Inversion of multilayer nets. In *Proceedings of the International Joint Conference on Neural Networks*, page 425, 1989.

[6] J. I. Minnix. Fault tolerance of the backpropagation neural network trained on noisy inputs. In *Proceedings of the International Joint Conference on Neural Networks*, pages 847–852, 1992. vol. I, (Baltimore).

[7] R. Reed, S. Oh, and R. J. Marks II. Regularization using jittered training data. In *Proceedings of the International Joint Conference on Neural Networks*, pages 147–152, 1992. vol. III, (Baltimore).

[8] J. Sietsma and R. J. F. Dow. Creating artificial neural networks that generalize. *Neural Networks*, 4(1):67–69, 1991.

[9] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman. Generalization by weight–elimination applied to currency exchange rate prediction. In *Proceedings of the International Joint Conference on Neural Networks*, page 837, 1991.

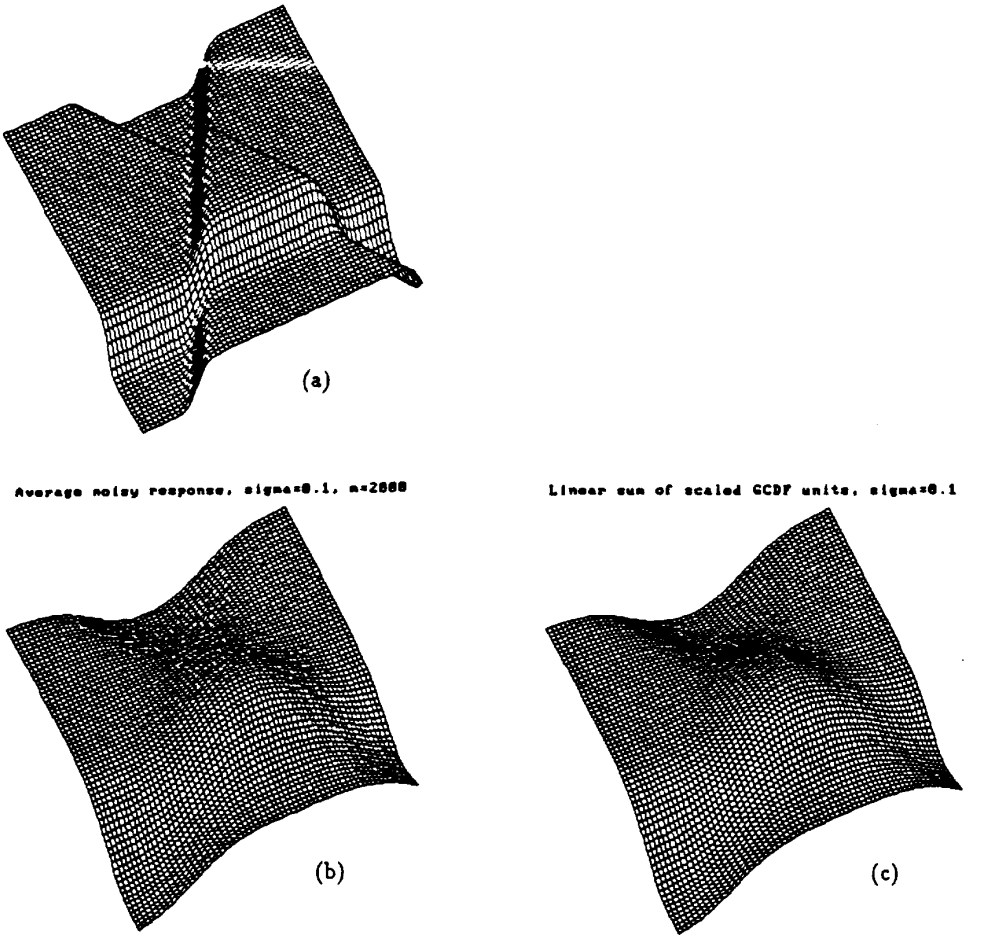
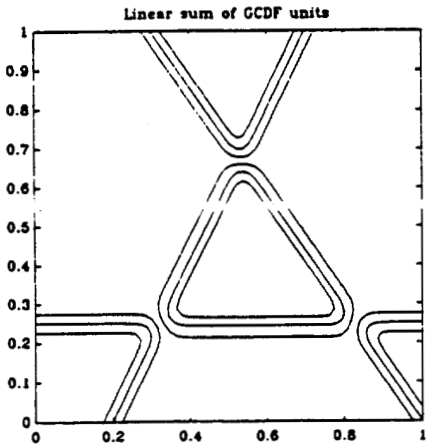
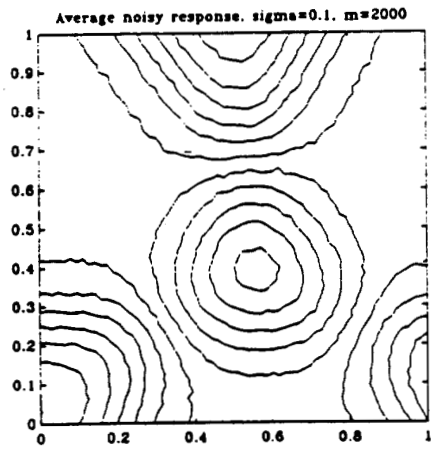


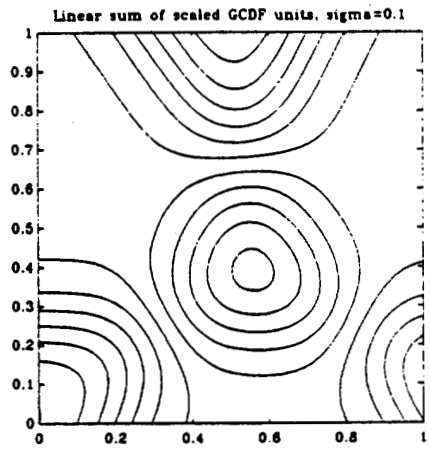
Figure 1: The network transfer function (3–d plots): (a) response of the original network; (b) average response with additive Gaussian input noise, $\sigma = 0.1$, averaged over 2000 noisy samples per grid point; (c) expected response computed by scaling.



(a)



(b)



(c)

Figure 2: The network transfer function (contour plots): (a) response of the original network; (b) average response with additive Gaussian input noise, $\sigma = 0.1$, averaged over 2000 noisy samples per grid point; (c) expected response computed by scaling.