Noise Injection into Inputs in Back-Propagation Learning

Kiyotoshi Matsuoka

Abstract— **Back-propagation can be considered a nonlinear regression technique, allowing a nonlinear neural network to acquire an input/output (I/O) association using a limited number of samples chosen from a population of input and output patterns. A crucial problem on back-propagation is its generalization capability. A network successfully trained for given samples is not guaranteed to provide desired associations for untrained inputs as well. Concerning this problem some authors showed experimentally that the generalization capability could remarkably be enhanced by training the network with noise injected inputs. This paper mathematically explains why and how the noise injection to inputs has such an effect.**

I. Introduction

BACK-PROPAGATION [1] may be the most widely used method among various learning methods for neural networks. The method can be considered a kind of nonlinear regression technique, allowing a nonlinear, layered network to acquire an input/output (I/O) relation using a set of sample patterns. During the learning process, the strengths of connections between neural units are iteratively modified so that an error between the actual and desired outputs of the network will be minimized.

A crucial problem on back-propagation is its generalization capability. Usually, training patterns used for learning are only a limited number of samples picked up from a population of input and output patterns. There is no guarantee that the network successfully trained on the given samples provides desired I/O associations for untrained patterns as well. Concerning this problem some authors showed experimentally that adding some noise to input patterns in back-propagation learning remarkably enhanced the generalization capability of the resultant networks [2], [3]. This paper mathematically discusses the reason why the noise injection into input has such an effect.

II. The Standard Back-Propagation

The neural network considered in this paper is the familiar layered network with no recurrent connection. The constituent units (neurons) in it, labelled as $1, \cdots, N$, are grouped into three types: input units $\{1, \cdots, N_I\}$, hidden units $\{N_I+1, \cdots, N_I+N_H\}$, and output units $\{N_I+N_H, \cdots, N_I+N_H+N_O\}$ $(N = N_I+N_H+N_O)$. Input unit i receives external input s_i $(i = 1, \cdots, N_I)$ to the network and transmits it to other units, so output x_i of unit i is given by

$$x_i = s_i \quad (i = 1, \cdots, N_I). \tag{1}$$

The output of hidden or output unit i is given by

$$x_i = g\left(\sum_j w_{ij}x_j + \theta_i\right) \quad (i = N_I+1, \cdots, N) \tag{2}$$

where w_{ij} represents the strength of the connection from unit j to unit i, and θ_i is a bias; $g(\cdot)$ is a nonlinear function that specifies the output of each unit in terms of its total input. The following logistic function is usually employed for it:

$$g(z) = 1/\{1+\exp(-z)\}. \tag{3}$$

The hidden units have no direct interaction with the external world while the output units provide the network's output y_i $(i = 1, \cdots, N_0)$ to the external:

$$y_i = x_{N_I+N_H+i}. \tag{4}$$

We write the overall input–output mapping of the network as

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{W}, \boldsymbol{\theta}; \boldsymbol{s}) \tag{5}$$

or more simply $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{s})$, where $\boldsymbol{s} = [s_1, \cdots, s_{N_1}]^T$, $\boldsymbol{y} = [y_1, \cdots, y_{N_O}]^T$, $\boldsymbol{W} = [w_{ij}]$, $\boldsymbol{\theta} = [\theta_{N_I+1}, \cdots, \theta_N]^T$, and $\boldsymbol{f} = [f_1, \cdots, f_{N_O}]^T$.

Let $\hat{\boldsymbol{s}}^{(m)}$ and $\hat{\boldsymbol{y}}^{(m)}$ $(m = 1, \cdots, M)$ be a given set of pairs of input and output to be learned. In order to obtain a network that produces output $\hat{\boldsymbol{y}}^{(m)}$ for input $\hat{\boldsymbol{s}}^{(m)}$, we determine the values of $\boldsymbol{W}$ and $\boldsymbol{\theta}$ such that they will minimize the following error function:

$$Q(\boldsymbol{W}, \boldsymbol{\theta}) = \sum_m \left|\hat{\boldsymbol{y}}^{(m)} - \boldsymbol{f}\left(\hat{\boldsymbol{s}}^{(m)}\right)\right|^2 \tag{6}$$

where $|\cdot|$ denotes the Euclidean norm of a vector. A simple way for minimizing Q is to give arbitrary values to $\boldsymbol{W}$ and $\boldsymbol{\theta}$ initially and then to iteratively modify them in the opposite direction of the gradient of the error function, i.e., the modification of $\boldsymbol{W}$ and $\boldsymbol{\theta}$ should be

$$\Delta w_{ij} = -\eta\, \partial Q/\partial w_{ij} \qquad \Delta\theta_i = -\eta\, \partial Q/\partial\theta_i \tag{7}$$

where η is a small positive constant. The calculations of $\partial Q/\partial W_{ij}$ and $\partial Q/\partial\theta_i$ can be performed efficiently by the well-known back-propagation algorithm [1].

If the previous iterative calculations have attained the minimum of Q, the resultant network might be optimal for the

Manuscript received June 30, 1990; revised May 22, 1991.

The author is with the Division of Control Engineering, Kyushu Institute of Technology, Tobata, Kitakyushu, 804 Japan.

IEEE Log Number 9104118.

sample patterns used in the learning. However, there is no guarantee that the network is also capable to provide reasonable associations for untrained patterns because the training patterns are usually only a small number of samples picked up from a population of an infinite number of patterns. For the network to have some generalization ability for unpresented patterns we introduce in the next section a new performance function to be minimized.

III. Noise Injection to Inputs

It is an ill-posed question how to optimize $\boldsymbol{W}$ and θ for every possible pattern because we rarely have information about unpresented patterns. So, in order to define the problem specifically we need to make some assumption on the property of the network we want to build. The principle proposed here is:

> The output of the network after learning should be as insensitive as possible to input variation, as long as the error (6) is within a reasonable bound.

An essentially equivalent statement can be seen in [4]: "Similar inputs (should) tend to produce similar outputs even if they are not trained before." Obviously, these claims presuppose some smoothness of the mapping to be realized by the network.

We formulate the above principle mathematically as follows. Let $\boldsymbol{s}^{(m)}$ be an input pattern similar to one training input $\hat{\boldsymbol{s}}^{(m)}$, that is

$$\boldsymbol{s}^{(m)} = \hat{\boldsymbol{s}}^{(m)} + \boldsymbol{d} \tag{8}$$

where $\boldsymbol{d} = [d_1, \cdots, d_{N_I}]^T$ and d_i are small random variables. Then, the variation of the network's output due to the disturbance, $\boldsymbol{d}$, is

$$\begin{aligned}\delta \boldsymbol{y}^{(m)} &= \boldsymbol{f}\left(\hat{\boldsymbol{s}}^{(m)} + \boldsymbol{d}\right) - \boldsymbol{f}\left(\hat{\boldsymbol{s}}^{(m)}\right) \\ &\sim \left\{\partial \boldsymbol{f}\left(\hat{\boldsymbol{s}}^{(m)}\right)\Big/\partial \boldsymbol{s}\right\}\boldsymbol{d}\end{aligned} \tag{9}$$

where $\partial \boldsymbol{f}/\partial \boldsymbol{s} = [\partial f_i/\partial s_j]$. We here assume that $\boldsymbol{d}$ is a random vector independent of $\hat{\boldsymbol{s}}^{(m)}$ with the following mean and variance:

$$\langle \boldsymbol{d} \rangle = \boldsymbol{O}, \quad \left\langle \boldsymbol{d}\boldsymbol{d}^T \right\rangle = \sigma^2 \boldsymbol{E}_{N_I} \tag{10}$$

where $\boldsymbol{E}_{N_I}$ is the identity matrix of size N_I and $\langle \cdot \rangle$ denotes the statistical expectation of the argument.

We define the sensitivity R of the network as the mean ratio of the variances of $|\delta \boldsymbol{y}^{(m)}|$ and $|\boldsymbol{d}|$:

$$\begin{aligned}R(\boldsymbol{w}, \boldsymbol{\theta}) &= \sum_m \left\langle \left|\delta \boldsymbol{y}^{(m)}\right|^2 \right\rangle \Big/ \left\langle |\boldsymbol{d}|^2 \right\rangle \\ &= \sum_m \left\langle \delta \boldsymbol{y}^{(m)T} \delta \boldsymbol{y}^{(m)} \right\rangle \Big/ \left\langle \boldsymbol{d}^T \boldsymbol{d} \right\rangle.\end{aligned} \tag{11}$$

According to (9) and (10), R can be approximated as

$$\begin{aligned}R(\boldsymbol{W}, \boldsymbol{\theta}) &\sim \sum_m \Bigg\langle \boldsymbol{d}^T \left(\partial \boldsymbol{f}\left(\hat{\boldsymbol{s}}^{(m)}\right)\Big/\partial \boldsymbol{s}\right)^T \\ &\quad \cdot \left(\partial \boldsymbol{f}\left(\hat{\boldsymbol{s}}^{(m)}\right)\Big/\partial \boldsymbol{s}\right)\boldsymbol{d} \Bigg\rangle \Big/ \left\langle \boldsymbol{d}^T \boldsymbol{d} \right\rangle \\ &= \sum_m \Bigg\langle \operatorname{trace}\Bigg\{ \left(\partial \boldsymbol{f}\left(\hat{\boldsymbol{s}}^{(m)}\right)\Big/\partial \boldsymbol{s}\right) \\ &\quad \cdot \boldsymbol{d}\boldsymbol{d}^T \left(\partial \boldsymbol{f}\left(\hat{\boldsymbol{s}}^{(m)}\right)\Big/\partial \boldsymbol{s}\right)^T \Bigg\} \Bigg\rangle \\ &\quad \div \left\langle \operatorname{trace} \boldsymbol{d}\boldsymbol{d}^T \right\rangle \\ &= \sum_m \sigma^2 \operatorname{trace}\Bigg\{ \left(\partial \boldsymbol{f}\left(\hat{\boldsymbol{s}}^{(m)}\right)\Big/\partial \boldsymbol{s}\right) \\ &\quad \cdot \left(\partial \boldsymbol{f}\left(\hat{\boldsymbol{s}}^{(m)}\right)\Big/\partial \boldsymbol{s}\right)^T \Bigg\} \Big/ (N_I \sigma^2) \\ &= \sum_m \left\|\partial \boldsymbol{f}\left(\hat{\boldsymbol{s}}^{(m)}\right)\Big/\partial \boldsymbol{s}\right\|^2 \Big/ N_I\end{aligned} \tag{12}$$

where $\|\cdot\|$ denotes the Euclidean norm of a matrix defined as $\|\boldsymbol{A}\|^2 = \operatorname{trace} \boldsymbol{A}\boldsymbol{A}^T = \sum_{i,j} a_{ij}^2$ ($\boldsymbol{A} = [a_{ij}]$). Notice that (12) does not involve σ^2.

It is obvious that the smaller R is the less sensitive the network output is to the disturbance in the input pattern. So, if the previously mentioned principle is valid, it may be reasonable to make Q and R be as small as possible simultaneously. This consideration leads to the following performance function to be minimized:

$$\begin{aligned}P(\boldsymbol{W}, \boldsymbol{\theta}) &= Q(\boldsymbol{W}, \boldsymbol{\theta}) + aR(\boldsymbol{W}, \boldsymbol{\theta}) \\ &\sim \sum_m \Bigg\{ \left|\boldsymbol{y}^{(m)} - f\left(\hat{\boldsymbol{s}}^{(m)}\right)\right|^2 + (a/N_I) \\ &\quad \cdot \left\|\partial \boldsymbol{f}\left(\hat{\boldsymbol{s}}^{(m)}\right)\Big/\partial \boldsymbol{s}\right\|^2 \Bigg\} \quad (a > 0)\end{aligned} \tag{13}$$

where a is a weighting parameter.

In order to (locally) minimize this performance function by the steepest decent method we have to calculate $\partial R/\partial w_{ij}$ and $\partial R/\partial \theta_i$. ($\partial Q/\partial w_{ij}$ and $\partial Q/\partial \theta_i$ can be obtained by the standard back-propagation.) Although those calculations are not so difficult, we can take an interesting alternative approach as follows. If we introduce random noise $\boldsymbol{n} = [n_1, \cdots, n_{N_I}]^T$ whose mean and covariance are given as

$$\langle \boldsymbol{n} \rangle = \boldsymbol{O}, \quad \langle \boldsymbol{n}\boldsymbol{n}^T \rangle = \varepsilon \boldsymbol{E}_{N_I} \quad (\varepsilon = a/N_I) \tag{14}$$

then (13) can be rewritten as

$$\begin{aligned}P(\boldsymbol{W}, \boldsymbol{\theta}) &\sim \sum_m \left\{ \left|\hat{\boldsymbol{y}}^{(m)} - \boldsymbol{f}\left(\hat{\boldsymbol{s}}^{(m)}\right)\right|^2 + \varepsilon \left\|\partial \boldsymbol{f}\left(\hat{\boldsymbol{s}}^{(m)}\right)\Big/\partial \boldsymbol{s}\right\|^2 \right\} \\ &= \sum_m \Bigg\{ \left|\hat{\boldsymbol{y}}^{(m)} - \boldsymbol{f}\left(\hat{\boldsymbol{s}}^{(m)}\right)\right|^2 \\ &\quad + \left\langle \left| \left(\partial \boldsymbol{f}\left(\hat{\boldsymbol{s}}^{(m)}\right)\Big/\partial \boldsymbol{s}\right)\boldsymbol{n} \right|^2 \right\rangle \Bigg\} \\ &= \sum_m \left\langle \left| \hat{\boldsymbol{y}}^{(m)} - \boldsymbol{f}\left(\hat{\boldsymbol{s}}^{(m)}\right) - \left(\partial \boldsymbol{f}\left(\hat{\boldsymbol{s}}^{(m)}\right)\Big/\partial \boldsymbol{s}\right)\boldsymbol{n} \right|^2 \right\rangle \\ &\sim \left\langle \sum_m \left| \hat{\boldsymbol{y}}^{(m)} - \boldsymbol{f}\left(\hat{\boldsymbol{s}}^{(m)} + \boldsymbol{n}\right) \right|^2 \right\rangle.\end{aligned} \tag{15}$$

Equation (15) implies that minimizing P is virtually equivalent to training the network so as to minimize the error between the desired output $\hat{\boldsymbol{y}}^{(m)}$ and the actual output produced by noisy input $\boldsymbol{s}^{(m)} + \boldsymbol{n}$. Obviously it can be achieved by the standard back-propagation without any modification. We have only to add the noise n to input.

Plaut, Nowlan, and Hinton [2] and Sietsma and Dow [3] found experimentally that adding some noise to the input applied to the back-propagation network had a remarkable effect on the generalization capability of the network. The above mathematical derivation reveals that the noise injected in the input contributes to reduce the sensitivity of the network to variation in the input.

The magnitude of noise, or ε, can be determined in the following way. In most situations, an allowable bound can be designated on the error, say

$$Q(W\boldsymbol{W}, \boldsymbol{\theta}) \leqq \phi. \tag{16}$$

Since a in (13), which is a weighting parameter for the network's sensitivity, is proportional to ε, we should let ε be as large as possible, as long as $Q(\boldsymbol{W}, \boldsymbol{\theta})$ is smaller than ϕ. In order to achieve this we monitor $Q(\boldsymbol{W}, \boldsymbol{\theta})$ during the learning process and control the magnitude of ε as

1) when $Q(\boldsymbol{W}, \boldsymbol{\theta}) > \phi$, then decrease ε by $\Delta\varepsilon$
2) when $Q(\boldsymbol{W}, \boldsymbol{\theta}) < \phi$, then increase ε by $\Delta\varepsilon$.

$\Delta\varepsilon$ should be sufficiently small. By this rule, the value of $Q(\boldsymbol{W}, \boldsymbol{\theta})$ approaches ϕ after a time of learning, while $P(\boldsymbol{W}, \boldsymbol{\theta})$ continues to decrease toward one of its local minima.

IV. Examples

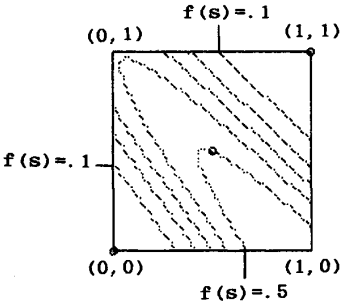
Here, we shall show a couple of simple examples. In the first example a three-layer network with two input units, five hidden units, and one output unit was trained for three pairs of patterns:

$$\hat{\boldsymbol{s}}^{(1)} = [0,0]^T, \hat{\boldsymbol{y}}^{(1)} = 0 \quad \hat{\boldsymbol{s}}^{(2)} = [0.5,0.5]^T, \hat{\boldsymbol{y}}^{(2)} = 0.5$$
$$\hat{\boldsymbol{s}}^{(3)} = [1,1]^T, \hat{\boldsymbol{y}}^{(3)} = 0.$$

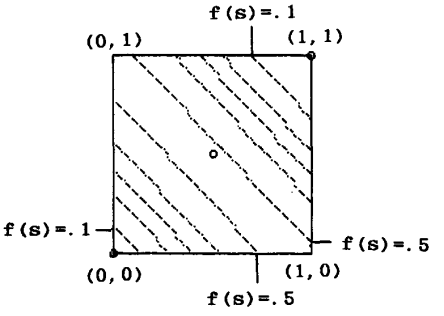
There were no direct connection from the input layer to the output layer. The standard back-propagation learning were executed with and without adding noises to input. Noises n_i $(i = 1,2)$ were chosen randomly from the uniform distribution between -0.5 and $+0.5$. Typical results are shown by the contour maps in Fig. 1. Although the learning was successful in both cases (i.e., the error converged to zero), the mapping obtained by the noiseless inputs is usually somewhat distorted as shown in Fig. 1(a), while the noisy inputs always produced apparently simpler mappings as shown in Fig. 1(b). Since every input vector, $\hat{\boldsymbol{s}}^{(m)}$, lies on line $s_1 = s_2$, it might be reasonable to expect the mapping after learning to be flat in the direction perpendicular to that line. Only the result with noise injection meets this expectation.

In the second example the following patterns (exclusive OR) were applied to the same network as previously:

$$\hat{\boldsymbol{s}}^{(1)} = [0,0]^T \hat{\boldsymbol{s}}^{(2)} = [1,1]^T \hat{\boldsymbol{y}}^{(1)} = \hat{\boldsymbol{y}}^{(2)} = 0,$$
$$\hat{\boldsymbol{s}}^{(3)} = [0,0]^T \hat{\boldsymbol{s}}^{(4)} = [1,1]^T \hat{\boldsymbol{y}}^{(3)} = \hat{\boldsymbol{y}}^{(4)} = 1.$$



(a)



(b)

Fig. 1. Results of the learning of the first example. (a) Using noiseless input. (b) Using noisy input.

The mappings obtained by the learning with noiseless and noisy inputs are shown in Fig. 2. From the symmetric configuration of the training vectors with respect to $\boldsymbol{s} = [0.5,0.5]^T$, it seems reasonable to expect $\boldsymbol{y} = 0.5$ at $\boldsymbol{s} = [0.5,0.5]^T$. It was always achieved only with noisy input (Fig. 2(b)).

The validity of the controlling strategy for the noise magnitude, ε, was tested using the same training patterns as the above two examples. In this case the noise was randomly chosen from the uniform distribution between $-\delta$ and $+\delta$, in which $\delta(\propto \varepsilon)$ varied according to the controlling rule with $\phi = 0.5$. Mappings by the obtained networks are shown in Fig. 3(a) and (b), being similar to Fig. 1(b) and Fig. 2(b), respectively.

V. Conclusion

If the network has no hidden unit and g is a linear function (i.e., $g(z) = z$ and $\theta_i = 0$ in (2)), we can show that the back-propagation with noise-injected inputs produces a network which embodies the optimal associative mapping proposed by Kohonen [5], as follows. The I/O relation of the linear network is represented by

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{s}) = \boldsymbol{W}\boldsymbol{s} \tag{17}$$

where $\boldsymbol{W} = [w_{ij}]$ and w_{ij} is the weight of the connection from input unit j to output unit i. Since $\partial \boldsymbol{f}(\boldsymbol{s})/\partial \boldsymbol{s} = \boldsymbol{W}$, the performance function (13) reads

$$P(\boldsymbol{W}) = \sum_m \left| \hat{\boldsymbol{y}}^{(m)} - \boldsymbol{W}\hat{\boldsymbol{s}}^{(m)} \right|^2 + \varepsilon \|\boldsymbol{W}\|^2$$

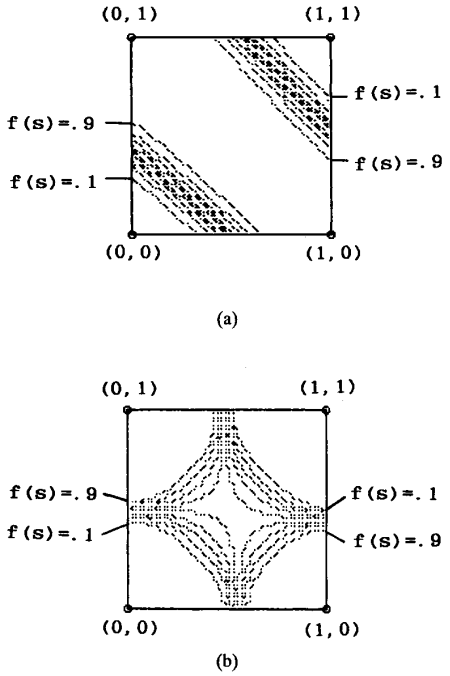


Fig. 2. Results of the learning of the second example. (a) Using noise less input. (b) Using noisy input.

$$= \|\boldsymbol{Y} - \boldsymbol{WS}\|^2 + \varepsilon\|\boldsymbol{W}\|^2 \tag{18}$$

where $\boldsymbol{S} = \left[\hat{\boldsymbol{s}}^{(1)}, \cdots, \hat{\boldsymbol{s}}^{(M)}\right]$ and $\boldsymbol{Y} = \left[\hat{\boldsymbol{y}}^{(1)}, \cdots, \hat{\boldsymbol{y}}^{(M)}\right]$. During the learning process $\boldsymbol{W}$ is iteratively modified as

$$\Delta\boldsymbol{W} = -\eta\, \partial P(\boldsymbol{W})/\partial\boldsymbol{W} = -2\eta\left(\boldsymbol{WSS}^T - \boldsymbol{YS}^T\right) - 2\eta\varepsilon\boldsymbol{W}. \tag{19}$$

According to (19), (18) has only one local minimum, i.e., global minimum

$$\boldsymbol{W} = \boldsymbol{YS}^T\left(\boldsymbol{SS}^T + \varepsilon\boldsymbol{E}_{N_I}\right)^{-1}. \tag{20}$$

Hence, $\boldsymbol{W}$ approaches this value with time for any initial values if the learning coefficient η is appropriately small. (20) converges with $\varepsilon \to 0$ $(\varepsilon \neq 0)$ to

$$\boldsymbol{W} = \boldsymbol{YS}^+ \tag{21}$$

where $(\cdot)^+$ represents the pseudoinverse of $(\cdot)$. This implies that if we choose ϵ to be very small, $\boldsymbol{W}$ becomes nearly (21), being the optimal associative mapping proposed by Kohonen [5]. The fact that the noise injection method provides a nearly optimal mapping in the linear network might encourage the present approach.

The above linear analysis suggests another interpretation of the noise injection method. The global minima of the error function $Q(\boldsymbol{W}) = \|\boldsymbol{Y} - \boldsymbol{WS}\|^2$ are attained by any $\boldsymbol{W}$ satisfying

$$-\boldsymbol{WSS}^T + \boldsymbol{YS}^T = \boldsymbol{0}$$

or

$$\boldsymbol{W} = \boldsymbol{YS}^+ + \boldsymbol{Z}\left(\boldsymbol{E}_{N_I} - \boldsymbol{SS}^+\right) \tag{22}$$

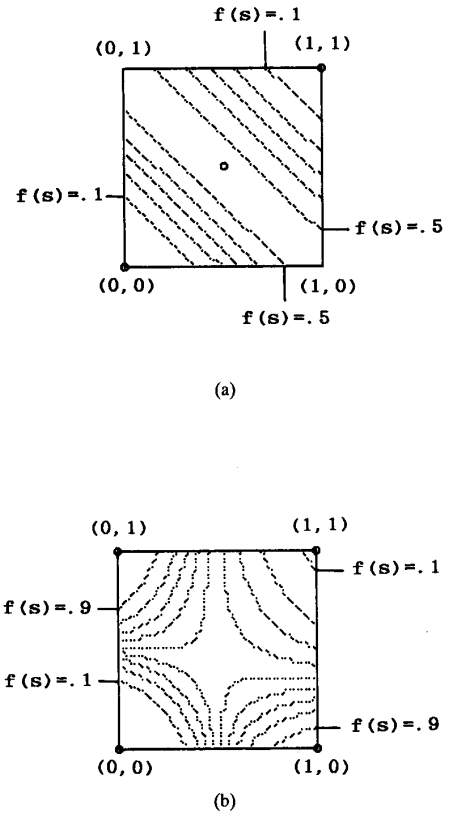


Fig. 3. Results of the learning with controlling the variance of noise.

where $\boldsymbol{Z}$ is an arbitrary $N_O \times N_I$ matrix. If $\boldsymbol{SS}^T$ is nonsingular (in other words if $\hat{\boldsymbol{s}}^{(1)}, \cdots, \hat{\boldsymbol{s}}^{(M)}$ span the entire of the input space), then (22) becomes (21) because $\boldsymbol{SS}^+ = \boldsymbol{E}_{N_I}$ holds. On the other hand, if $\boldsymbol{SS}^T$ is singular (in other words, if $\hat{\boldsymbol{s}}^{(1)}, \cdots, \hat{\boldsymbol{s}}^{(M)}$ span a strict subspace in the input space), then $\boldsymbol{W}$ cannot uniquely be determined. Even in the latter case, if we determine $\boldsymbol{W}$ such that it takes the minimum norm among $\boldsymbol{W}$ given by (22), then we have (21) uniquely. Thus, when the noise injection is applied to the linear network, it has a role to determine the optimal value of $\boldsymbol{W}$ uniquely in any case.

It is often pointed out that, in networks with hidden layers, reducing the number of hidden units to an appropriate size is effective for improving the generalization capability of the network [2], [6]. This is because the dimension of the image of the mapping is lowered by reducing the number of hidden units. In what follows, we show that the noise addition is useful even after the hidden units size is appropriately reduced.

Consider a three-layer, linear network with connection matrix $\boldsymbol{W}_1$ from the input layer to the hidden layer and connection matrix $\boldsymbol{W}_2$ from the hidden layer to the output layer. In this case, the overall I/O mapping is

$$\boldsymbol{y} = \boldsymbol{W}_2\boldsymbol{W}_1 s \tag{23}$$

and the error function reads

$$Q(\boldsymbol{W}_1, \boldsymbol{W}_2) = \sum_m \left|\boldsymbol{W}_2\boldsymbol{W}_1\hat{\boldsymbol{s}}^{(m)}\hat{\boldsymbol{y}}^{(m)}\right|^2$$

$$= \|\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{S} - \boldsymbol{Y}\|^2 \tag{24}$$

Let $\boldsymbol{W}_1 = \boldsymbol{V}_1$ and $\boldsymbol{W}_2 = \boldsymbol{V}_2$ minimize Q. Then, in general

$$\boldsymbol{W}_1 = \boldsymbol{V}_1 + \boldsymbol{Z}\left(\boldsymbol{E}_{N_I} - \boldsymbol{S}\boldsymbol{S}^+\right), \quad \boldsymbol{W}_2 = \boldsymbol{V}_2 \tag{25}$$

(where $\boldsymbol{Z}$ is an arbitrary matrix with proper size) give the minimum of Q because the network generate the same output for $\boldsymbol{s}^{(m)}$:

$$\boldsymbol{W}_1\boldsymbol{S} = \left\{\boldsymbol{V}_1 + \boldsymbol{Z}\left(\boldsymbol{E}_{N_I} - \boldsymbol{S}\boldsymbol{S}^+\right)\right\}\boldsymbol{S} = \boldsymbol{V}_1\boldsymbol{S} \tag{26}$$

or

$$\boldsymbol{W}_1\hat{\boldsymbol{s}}^{(m)} = \boldsymbol{V}_1\hat{\boldsymbol{s}}^{(m)} \quad (m = 1, \cdots, M). \tag{27}$$

This means that even if the dimension of Im $\boldsymbol{W}_2\boldsymbol{W}_1$ is lowered by reducing the number of hidden units, the mapping minimizing $Q(\boldsymbol{W}_1, \boldsymbol{W}_2)$ or

$$\boldsymbol{W}_2\boldsymbol{W}_1 = \boldsymbol{V}_2\boldsymbol{V}_1 + \boldsymbol{V}_2\boldsymbol{Z}\left(\boldsymbol{E}_{N_I} - \boldsymbol{S}\boldsymbol{S}^+\right) \tag{28}$$

is not necessarily unique. By applying the noise injection method to such as case, the mapping $\boldsymbol{W}_2\boldsymbol{W}_1$ is uniquely determined such that a weighted sum of the error function and $\|\boldsymbol{W}_2\boldsymbol{W}_1\|^2$ are minimized. Thus, even after reducing the hidden units to an appropriate number, noise injection to inputs can yield a remarkable effect on the generalization; one can see a good example in Seitsma and Dow [3].

When the number of training patterns is not less than the dimension of the input space, it will rarely happen that the training input vectors lie on the subspace in the input space. However, it is by no means rare that the training input vectors are located near a subspace because the patterns to be learned are more or less similar to each other in actual applications. In such a situation, the error function $Q(\boldsymbol{W})$ takes the shape of a long ravine as shown in Fig. 4, and its minimum point ($\overline{W}$ in the figure) becomes quite sensitive to the samples that have happened to be used for learning. This problem is the collinearity problem which often occurs in linear regression analysis. A well-known solution to this problem is to minimize $P(\boldsymbol{W})$ with finite ε even when $Q(\boldsymbol{W})$ has a unique minimum point (the Ridge estimation). Then, the optimal mapping is obtained as a nearest point in the ravine from the origin ($\tilde{W}$ in Fig. 4). In Section III we implemented this well-known technique in linear regression analysis into the back-propagation learning for multilayered networks with nonlinear units.

The second term in (19), $-2\eta\varepsilon w_{ij}$, which cause w_{ij} to decay in proportion to current w_{ij}, can be interpreted as a kind of forgetting. In the case of the two-layer linear network, noise injection into input is equivalent to adding the decay term to the standard back-propagation. The idea of incorporating this kind of decay terms into the learning of nonlinear multilayer networks were suggested in [2], [7], [8]. As for the present method, however, the decay term for w_{ij} in a nonlinear, multilayered network takes a more complicated

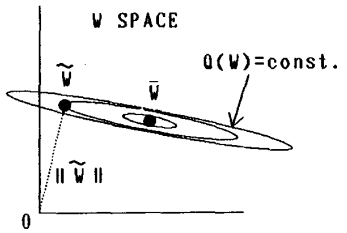


Fig. 4. Optimal weight for the error function and that for the proposed performance function.

form of $-\eta\varepsilon\sum_m \partial\|\boldsymbol{f}(\boldsymbol{s}^{(m)})/\partial\boldsymbol{s}\|^2/\partial w_{ij}$, depending on $\boldsymbol{s}^{(m)}$ $(m = 1, \cdots, M)$ and other connection weights.

Finally, it should be noted that the acquisition of generalization capability by noise injection into inputs relies on the assumption that the (unknown) mapping from the input space to the output space should be smooth. When this assumption is valid, the method can enhance the generalization ability of the back-propagation learning. On the other hand, when this condition is not the case, the noise injection might result in poor generalization capability.

REFERENCES

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," *Parallel Distributed Processing,* D. E. Rumelhart and J. L. McClelland and the PDP Research Group, Eds. Cambridge, MA: MIT Press, pp. 318–362, 1986.

[2] D. C. Plaut, S. J. Nowlan and G. E. Hinton, "Experiments on learning by back-propagation," Tech. Rep. CMU-CS-86-126, 1986.

[3] J. Sietsma and R. J. F. Dow, "Neural network pruning—Why and how," in *Proc. IEEE Int. Conf. Neural Networks,* vol. I, 1988, pp. 325–333.

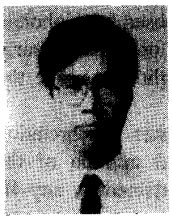
[4] E. Erusü and H. Tolle, "Learning control structures with neuron-like associative memory systems," *Organization of Neural Networks,* W. von Seelen, G. Shaw, and U. M. Leinhos, Eds. Weinheim, Germany: VCH Publishers, 1988, pp. 369–393.

[5] T. Kohone, *Self-Organization and Associative Memory,* second ed. New York: Springer, 1987.

[6] S. K. Kung and J. N. Hwang, "An algebraic projection analysis for optimal hidden units size and learning rates in back-propagation learning," in *Proc. IEEE Int. Conf. Neural Networks,* vol. I, 1988, pp. 363–370.

[7] M. Ishikawa, "A structural learning algorithm with forgetting of link weights," IEICE Tech. Rep., MBE-88-144, pp. 143–148 (in Japanese), 1988.

[8] S. J. Hanson and L. Y. Pratt, "Comparing biases for minimal network construction with back-propagation," *Advances in Neural Information Processing Systems,* D. S. Touretzky, Ed. Palo Alto: Morgan Kaufmann, 1989, pp. 177–185.



Kiyotoshi Matsuoka received the B.E. degree in mechanical engineering in 1971 from Kyushu Institute of Technology, Kitakyushu, and the M.S. and Ph.D. degrees in mechanical engineering in 1973 and 1976, respectively, from the University of Tokyo, Tokyo.

From 1976 to 1978 he was a Lecturer at the Department of Control Engineering, Kyushu Institute of Technology, Kitakyushu, Japan. He is currently, a Professor there. His research interests include neural networks, bioengineering, and artificial intelligence.