Using Additive Noise in Back-Propagation Training

Lasse Holmström, *Member, IEEE,* and Petri Koistinen

***Abstract*— We discuss the possibility of improving the generalization capability of a neural network by introducing additive noise to the training samples. The network considered is a feedforward layered neural network trained with the back-propagation algorithm. Back-propagation training is viewed as nonlinear least-squares regression and the additive noise is interpreted as generating a kernel estimate of the probability density that describes the training vector distribution. Two specific application types are considered: pattern classifier networks and estimation of a nonstochastic mapping from data that are corrupted by measurement errors. We do not prove that the introduction of additive noise to the training vectors always improves network generalization. However, our analysis suggests mathematically justified rules for choosing the characteristics of noise if additive noise is used in training. Further, using results of mathematical statistics we establish various asymptotic consistency results for the proposed method. We also report numerical simulations that give support to the applicability of the suggested training method.**

I. Introduction

NEURAL networks can be trained to perform a great variety of tasks. A crucial measure of network performance is its ability to generalize, i.e., to respond properly to previously unseen input data. In the case of a pattern classifier network, this means classifying correctly samples that have not been used in training the network. If the network is trained to approximate a function, then generalization means the ability to interpolate in a meaningful way between the training samples. Poor generalization can arise, for example, in a situation where the number of parameters to be learned is too high compared with the number of training samples available. The remedy discussed in this paper is to generate an unlimited source of training samples by adding random noise to the available training vectors. The problem is to choose the characteristics of the additive noise so that true improvement in network performance is achieved.

The idea of introducing noise to the training vectors of a neural network has appeared in the work of several authors. Results in [1] and [2] suggest the idea of using additive noise to expand the training vector set. For other examples see [3]–[5]. Experiments with more complicated distortions of training vectors are reported in [6]–[8]. A number of people have studied from the viewpoint of statistical mechanics the generalization properties of Hopfield-type recurrent networks [9]–[10] and perceptrons [11]–[14] in the presence of noise.

Manuscript received December 19, 1990: revised June 12, 1991. This work was supported in part by grants from the Academy of Finland and the Technology Development Centre (TEKES).

The authors are with the Rolf Nevanlinna Institute, University of Helsinki, Teollisuuskatu 23, 00510 Helsinki, Finland.

IEEE Log Number 9102398.

Our main interest is the feedforward layered network with back-propagation training described in [15]. However, the theoretical results developed remain valid in the more general context of nonlinear regression. Two different applications are studied. The first is a network trained to classify Euclidean input vectors into a finite number of classes when the different classes are described by probability densities. The second application is a network trained to serve as a nonlinear regression function when the joint distribution of the independent and dependent variables is described by a probability density. As a concrete example we consider the approximation of a nonstochastic mapping when the available training data are corrupted by noise.

The key idea of our work is that the training samples to which random noise has been added can be regarded as being drawn from a so-called kernel or Parzen–Rosenblatt estimate [16]–[18] of the true training vector density. The general definition of a kernel estimate is as follows. Let X be a random vector taking values in a Euclidean space $\boldsymbol{R}^d$ and suppose that the distribution of X is described by a probability density function f. Let K be a probability density, the "kernel," and suppose that $h > 0$. If $X_1, \cdots, X_n$ is a sample of n independent observations of X, then the n-point kernel estimate of f corresponding to K and h is the density

$$f_{n,h}(x) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h^d} K\left(\frac{x - X_i}{h}\right), \qquad x \in \boldsymbol{R}^d.$$

Note that $f_{n,h}$ depends on $X_1, \cdots, X_n$ and is therefore in fact a random function. The characteristics of the additive noise are controlled by the choice of K and h.

Now, suppose that there are n network training samples available. Replacing an original probability density f by its n-point kernel estimate $f_{n,h}$ results in a modified loss or error function for the network to minimize (Section II). It makes sense to choose the additive noise so that the modified loss function approximates the original loss function as well as possible. The difference between the modified and original loss functions is shown to be dominated by the integral $\int_{R^d} |f_{n,h} - f|$, the L_1 distance between the density f and its kernel estimate. This observation leads to several results of both practical and theoretical interest. First, mathematically justified methods for choosing the characteristics of the additive noise can be suggested. We do not give conditions under which the use of additive noise in network training is guaranteed to improve generalization but our analysis does provide useful mathematical insight and suggests practical methods to design effective noise if use of additive noise is attempted. Second, using results of mathematical statistics, we prove the asymptotic consistency of the estimated noisy loss

function and its minimizing weight vector as the number of training samples tends to infinity provided that the amount of noise used is chosen appropriately. We also report numerical simulations that give support to the applicability of the suggested training method.

The paper is organized as follows. Section II presents a probabilistic analysis of the proposed training method together with several consistency results. Section III discusses practical methods for choosing the characteristics of the additive noise. Numerical experiments are presented in Section IV and a summary is given in Section V.

II. Analyzing the Effects of Additive Noise

A. Training of Feedforward Layered Networks

Consider a feedforward layered network defined by using a bounded continuous activation function such as $t \mapsto (1+e^{-t})^{-1}, t \in \boldsymbol{R}$ (see [15]). Suppose that the vector $w \in \boldsymbol{R}^l$ contains all the weights and biases of the network. Further, let the dimensions of the input and output vectors of the network be k and m, respectively. Then the network represents a bounded continuous mapping $g : \boldsymbol{R}^k \times \boldsymbol{R}^l \to \boldsymbol{R}^m, y = g(x,w)$. If the weights w are restricted to a subset $W \subset \boldsymbol{R}^l$, then in the following we will in fact only need the weaker boundedness condition

$$M_W(g) := \sup_{(x,w)\in \boldsymbol{R}^k \times W} \|g(x,w)\| < \infty. \tag{1}$$

Here $\|\cdot\|$ denotes the Euclidean norm. Further, we will not utilize the network structure of the mapping g in any special way. Therefore, in the rest of this section we will only assume that $W \subset \boldsymbol{R}^l, g : \boldsymbol{R}^k \times W \to \boldsymbol{R}^m$ is bounded (i.e., (1) holds), that $w \mapsto g(x,w)$ is continuous for each $x \in \boldsymbol{R}^k$, and that $x \mapsto g(x,w)$ is Borel measurable for each $w \in \boldsymbol{W}$. Because of the underlying motivation, such a g is still referred to as a network.

We will interpret the training of the network as nonlinear least-squares regression. For a good discussion of such a statistical approach, see [19]. Thus, we consider data given by a pair $Z = (X,Y)$ of random vectors where X has values in $\boldsymbol{R}^k$, Y in $\boldsymbol{R}^m$, and Z in $\boldsymbol{R}^p$, $p = k+m$. The goal of network training is to select w so that the random vector $g(X,w)$ approximates Y as well as possible. More precisely, denoting $l(z,w) = \|y - g(x,w)\|^2, z = (x,y)$, one attempts to minimize with respect to w the loss or error function

$$\lambda(w) = E(l(Z,w)) = \int_{\boldsymbol{R}^p} l(z,w)\, dP_Z(z), \qquad w \in W, \tag{2}$$

where E denotes expectation and P_Z is the distribution of Z in $\boldsymbol{R}^p$. The particular function l chosen here corresponds to training by least-squares minimization but other kinds of error functions are also in use (e.g. [19]).

In practice, there are available only finitely many training samples $Z_i = (X_i, Y_i)$, $i = 1, \cdots, n$, of Z and one tries to minimize the sample estimate

$$\hat{\lambda}_n(w) = \frac{1}{n}\sum_{i=1}^{n} l(Z_i,w) = \frac{1}{n}\sum_{i=1}^{n} \|Y_i - g(X_i,w)\|^2, \qquad w \in W.$$

Here we assume that the Z_i's are independent random vectors distributed as Z. Given an actual sequence $z_1, z_2, \cdots$, of values of $Z_1, Z_2, \cdots$, one may then ask how well $\hat{\lambda}_n$ actually approximates λ. It turns out that under some rather general conditions $\hat{\lambda}_n$ in fact converges to λ uniformly. Suppose, for example, that W is compact and that $l(z,w) \le q(z), (z,w) \in \boldsymbol{R}^p \times W$, with q integrable with respect to P_Z. Then for a randomly chosen sample sequence $z_1, z_2, \cdots$, one has

$$\lim_{n\to\infty} \sup_{w\in W} \left| \frac{1}{n}\sum_{i=1}^{n} l(z_i,w) - \lambda(w) \right| = 0$$

with probability 1 (i.e., "almost surely") by the uniform law of large numbers [20, theorem 2]. In the two cases considered below (subsections II-B and II-C), such a dominating function q exists and can be taken as $q(z) = (\|y\| + M_W(g))^2, z \in \boldsymbol{R}^p$. Note also that the assumption of a compact set W is not completely unreasonable since in practice $\hat{\lambda}_n$ is minimized using a computer.

Consider now a fixed set $\{z_1, \cdots, z_n\}$ of training vectors. As explained below (subsections II-B and II-C), introducing additive noise to the vectors z_i can be modeled by a new random vector $Z_h^{(n)}$, where the parameter h reflects the level of noise used. The loss function λ is then replaced by a new loss function, $\lambda_h^{(n)}$:

$$\lambda_h^{(n)}(w) = E\Big(l\Big(Z_h^{(n)}, w\Big)\Big) = \int_{\boldsymbol{R}^p} l(z,w)\, dP_{Z_h^{(n)}}(z), \qquad w \in W. \tag{3}$$

Training of the network with noisy training vectors corresponds to minimizing (3), but in practice one minimizes a finite sum

$$\hat{\lambda}_{r,h}^{(n)}(w) = \frac{1}{r}\sum_{i=1}^{r} l\Big(Z_{i,h}^{(n)}, w\Big), \qquad w \in W, \tag{4}$$

where $Z_{1,h}^{(n)}, \cdots, Z_{r,h}^{(n)}$ are independent random vectors distributed as $Z_h^{(n)}$.

Suppose that W is compact and that $\overline{w}, \overline{w}_h^{(n)}$ and $\hat{w}_{r,h}^{(n)}$ minimize (2), (3), and (4), respectively. Assume for the moment that such minimizing weight vectors are unique. We want the noisy training vectors $Z_{i,h}^{(n)}$ to produce weights $\hat{w}_{r,h}^{(n)}$ that approximate $\overline{w}$ as closely as possible. By the uniform law of large numbers we can make $\sup_{w\in W} |\hat{\lambda}_{r,h}^{(n)}(w) - \lambda_h^{(n)}(w)|$ arbitrarily small by choosing r large enough. It seems plausible then that $\hat{w}_{r,h}^{(n)}$ will also be close to $\overline{w}_h^{(n)}$. Now, if the additive noise is chosen so that $\sup_{w\in W} |\lambda_h^{(n)}(w) - \lambda(w)|$ will also be small, then one can hope to have $\hat{w}_{r,h}^{(n)}$ close to $\overline{w}$. We will next turn this intuitive idea into rigorous analysis in two different contexts.

B. Pattern Classifier Networks

Suppose that the input data vectors $x \in \boldsymbol{R}^k$ originate from m different pattern classes. This can be modeled as a pair (X,J), where X is a random vector with values in $\boldsymbol{R}^k$ and J is a discrete random variable with possible values $1, \cdots, m$.

The standard way to build a pattern classifier feedforward neural network is to train a network to map a vector x from class j to the unit vector $e_j = \left(0, \cdots, 0, \overset{j}{1}, 0, \cdots\right)$. Formally, we can use an auxiliary function $e : \boldsymbol{R} \to \boldsymbol{R}^m$ such that $e(j) = e_j, j \in \{1, \cdots, m\}$, and set $Y = e \circ J, Z = (X, Y)$. The trained network then classifies an input vector x to class j_0 if the j_0th output is the maximum output.

We assume that the distribution P_X of X can be described by a probability density function f,

$$P_X(B) = P(X \in B) = \int_B f$$

where $B \subset \boldsymbol{R}^k$ is a Borel set and P denotes probability. We assume that the class *a priori* probabilities $P^j = P(J = j)$ are positive for all $j = 1, \cdots, m$. Then, for each j, the conditional probability measure

$$P(X \in B | J = j) = \frac{P(X \in B \text{ and } J = j)}{P^j}$$

is defined and has a density f^j, the so-called class-conditional density,

$$P(X \in B | J = j) = \int_B f^j$$

for a Borel set $B \subset \boldsymbol{R}^k$.

For each j, the *a posteriori* probability at $x \in \boldsymbol{R}^k$ is defined as

$$P^j(x) = \begin{cases} P^j f^j(x)/f(x) & \text{if } f(x) \neq 0 \\ 0 & \text{if } f(x) = 0. \end{cases}$$

For a fixed x, the quantity $P^j(x)$ is the probability that x comes from the class j. Consider then the conditional expectation of Y given $X = x$,

$$\begin{aligned} E(Y|X = x) &= \sum_{j=1}^{m} P^j(x) e_j \\ &= \left(P^1(x), \cdots, P^m(x)\right) =: P(x). \end{aligned}$$

Denoting by $\langle \cdot, \cdot \rangle$ the inner product in $\boldsymbol{R}^m$, we get

$$\begin{aligned} \lambda(w) &= E\left(\|Y - g(X, w)\|^2\right) \\ &= E\left(\|Y - P(X)\|^2\right) + 2E(\langle Y - P(X), P(X) \\ &\quad - g(X, w)\rangle) \\ &\quad + E\left(\|P(X) - g(X, w)\|^2\right), \qquad w \in W. \end{aligned} \tag{5}$$

By computing the expectation iteratively, the second term on the right-hand side can be seen to vanish (cf. [19, p. 432]). Therefore, minimizing λ is equivalent to finding $\overline{w}$ for which $E(\|P(X) - g(X, w)\|^2)$ has its smallest value. This means that one minimizes the quantity

$$\sum_{j=1}^{m} \int_{\boldsymbol{R}^k} \left(P^j(x) - g_j(x, w)\right)^2 f(x)\, dx. \tag{6}$$

If (6) can be made to vanish for some $\overline{w}$ then the network outputs $g_j(x, \overline{w})$ coincide with the *a posteriori* probabilities $P^j(x)$ for all x in a subset of $\boldsymbol{R}^k$ where X occurs with probability 1. The network then defines the optimal Bayesian classifier that minimizes the probability of misclassification. In practice, however, the network classifier only approximates the optimal classifier. Further, (6) shows that the approximation tends to be best in those regions of $\boldsymbol{R}^k$ where f is large, typically near class centers, when good performance of a classifier would require good accuracy also near the class boundaries.

Suppose now that $x_1, \cdots, x_n \in \boldsymbol{R}^k$ are vectors from classes $j_1, \cdots, j_n \in \{1, \cdots, m\}$. Thus, we have n training vectors $z_i = (x_i, e_{j_i}),\ i = 1, \cdots, n$. We will generate new training vectors by adding noise to the vectors x_i. This can be modeled by introducing a noise random vector S with probability density K. A typical choice would be to use the normal density $K(x) = (2\pi)^{-k/2} \exp(-\|x\|^2/2),\ x \in \boldsymbol{R}^k$. The level of noise for class j is controlled with a parameter $h^j > 0$ and we denote $h = \left(h^1, \cdots, h^m\right)$. Consider the following sampling procedure.

Procedure 1:

1) Select $i \in \{1, \cdots, n\}$ with equal probability for each index.
2) Draw a sample s from the density K.
3) Set $x_h^{(n)} = x_i + h^{j_i} s, j_h^{(n)} = j_i$.

This scheme generates samples $x_h^{(n)}$ and $j_h^{(n)}$ of a new random vector $X_h^{(n)}$ and a new discrete random variable $J_h^{(n)}$. Define $Y_h^{(n)} = e \circ J_h^{(n)}$. The noisy training vectors $(x_h^{(n)}, e_{j_h^{(n)}})$ can then be thought of as realizations of the random vector $Z_h^{(n)} = (X_h^{(n)}, Y_h^{(n)})$. It is easy to see that $X_h^{(n)}$ has density

$$f_{n,h}(x) = \frac{1}{n} \sum_{i=1}^{n} K_{h^{j_i}}(x - x_i), \qquad x \in \boldsymbol{R}^k, \tag{7}$$

where we have used the notation $K_\rho(x) = \rho^{-k} K(x/\rho),\ \rho > 0,\ x \in \boldsymbol{R}^k$. Thus, introducing additive noise can be interpreted as replacing a finite set $\{x_1, \cdots, x_n\}$ of vectors drawn from the original density f by an infinite source of training vectors represented by $f_{n,h}$. Such a resampling procedure can be regarded as an instance of a general estimation procedure called smoothed bootstrap (e.g. [21]). The term *smoothed* refers to how $f_{n,h}$ is obtained from $\{x_1, \cdots, x_n\}$. Each term in the sum (7) represents a "bump" centered at x_i. The width of the bump is controlled by the parameter h^{j_i}. Thus, h determines how much the discrete samples x_i are smoothed (Fig. 1).

For each $j = 1, \cdots, m$, let $M_n^j = \{i | j_i = j, i = 1, \cdots, n\}$ and denote by N_n^j the number of indices in M_n^j. Corresponding to the class *a priori* probabilities P^j we define $P_n^j = P(J_h^{(n)} = j) = N_n^j/n$. When $N_n^j > 0$ one shows easily that for a Borel set $B \subset \boldsymbol{R}^k$,

$$\begin{aligned} P\left(X_h^{(n)} \in B \mid J_h^{(n)} = j\right) &= \frac{P\left(X_h^{(n)} \in B \text{ and } J_h^{(n)} = j\right)}{P_n^j} \\ &= \int_B f^j_{N_n^j, h^j}, \end{aligned}$$

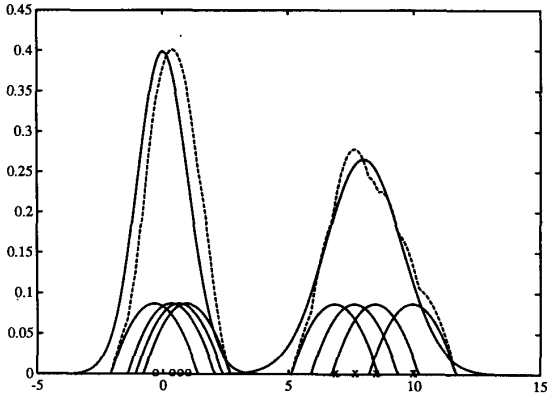


Fig. 1. Adding random noise to a set of the finite number of training vectors (o's and x's) drawn from a density (solid curve) means sampling from a smoothed density (dashed curve). Noise is drawn from the Bartlett kernel defined in Section III.

where

$$f^j_{N^j_n,h^j}(x) = \frac{1}{N^j_n} \sum_{i \in M^j_n} K_{h^j}(x - x_i), \qquad x \in \boldsymbol{R}^k$$

is an N^j_n-point kernel estimate of f^j. For convenience, define $f^j_{N^j_n,h^j} = 0$ when $N^j_n = 0$.

We are now ready to estimate the difference between the loss functions λ and $\hat{\lambda}^{(n)}_{r,h}$. Let $w \in W$. From the inequality $|\hat{\lambda}^{(n)}_{r,h}(w) - \lambda(w)| \le |\hat{\lambda}^{(n)}_{r,h}(w) - \lambda^{(n)}_h(w)| + |\lambda^{(n)}_h(w) - \lambda(w)|$ it follows that

$$\sup_{w \in W} |\hat{\lambda}^{(n)}_{r,h}(w) - \lambda(w)| \le \sup_{w \in W} |\hat{\lambda}^{(n)}_{r,h}(w) - \lambda^{(n)}_h(w)| + \sup_{w \in W} |\lambda^{(n)}_h(w) - \lambda(w)|. \tag{8}$$

By the uniform law of large numbers, at least when W is compact, the term $\sup_{w \in W} |\hat{\lambda}^{(n)}_{r,h}(w) - \lambda^{(n)}_h(w)|$ can be made arbitrarily small by choosing r large enough. Note that, in principle, we are free to choose r as large as we please because we can use an unlimited source of synthetic noise vectors. On the other hand, one can try to make the term $\sup_{w \in W} |\lambda^{(n)}_h(w) - \lambda(w)|$ small by choosing h appropriately (Proposition 1 and Theorem 1). These facts can then be combined to prove the convergence of $\sup_{w \in W} |\hat{\lambda}^{(n)}_{r,h}(w) - \lambda(w)|$ to 0 as n and r tend to infinity appropriately (Theorem 2). The proofs of the results of this section can be found in the Appendix.

Proposition 1: We have

$$\sup_{w \in W} \left|\lambda^{(n)}_h(w) - \lambda(w)\right| \le C \left(\sum_{j=1}^{m} |P^j_n - P^j| + \sum_{j=1}^{m} \int_{\boldsymbol{R}^k} \left| f^j_{N^j_n,h^j} - f^j \right| \right), \tag{9}$$

where $C = (1 + M_W(g))^2$.

Proposition 1 gives us two insights into the minimization of $|\lambda^{(n)}_h(w) - \lambda(w)|$ in the weight space W. First, assuming that the *a priori* probabilities P^j are known, one should try to employ nP^j training vectors from class j because this minimizes the terms $|P^j_n - P^j| = |N^j_n/n - P^j|$. Second, the density K and the smoothing parameters h^j should be chosen so that the L_1 distances $\int_{\boldsymbol{R}^k} |f^j_{N^j_n,h^j} - f^j|$ are minimized. This second insight will be discussed further in Section III.

We will now turn to the asymptotic behavior of $\lambda^{(n)}_h$, $\hat{\lambda}^{(n)}_{r,h}$ and $\hat{w}^{(n)}_{r,h}$ as n and r tend to infinity. The discussion has a rather theoretical flavor but, as pointed out in [19, p. 438], rigorous analysis of such questions in connection with any proposed learning technique should be considered standard practice.

So far we have had n fixed training vectors $z_1, \cdots, z_n$. When an infinite random sample $Z_1, Z_2, \cdots$ is considered, we obtain infinite sequences $(P^j_n)_n, (f^j_{N^j_n,h^j})_n$ and $(\lambda^{(n)}_h)_n$, where, for each n, P^j_n, $f^j_{N^j_n,h^j}$ and $\lambda^{(n)}_h$ are random and depend on the first n random vectors $Z_1, \cdots, Z_n$ only. Using (9) it is possible to study probabilistically the convergence of $\sup_{w \in W} |\lambda^{(n)}_h(w) - \lambda(w)|$ as n tends to infinity. Instead of constant smoothing parameters h^j, we will let the smoothing vary as a function of the number of samples available from each class. Thus, we consider m infinite sequences $\left(h^j_n\right)_n$. To simplify the notation we denote $H^j_n := h^j_{N^j_n}$ and $H_n = \left(H^1_n, \cdots, H^m_n\right)$. The next result gives general sufficient conditions for the sequences $(h^j_n)_n, j = 1, \cdots, m$, to guarantee the uniform convergence of $\lambda^{(n)}_h$ to λ. These conditions are familiar from the theory of kernel estimates (cf. [16] and [18]).

Theorem 1: Suppose that the smoothing parameter sequences $(h^j_n)_n, j = 1, \cdots, m$, satisfy the conditions

$$\lim_{n \to \infty} h^j_n = 0, \qquad \lim_{n \to \infty} n\left(h^j_n\right)^k = \infty. \tag{10}$$

Then,

$$\lim_{n \to \infty} \sup_{w \in W} \left|\lambda^{(n)}_{H_n}(w) - \lambda(w)\right| = 0$$

with probability 1.

On the convergence of $\hat{\lambda}^{(n)}_{r,H_n}$ to λ we have the following result. The notation $\overline{\lim}$ will be used for the upper limit or "lim sup" of a sequence.

Theorem 2: Suppose that the smoothing parameter sequences $(h^j_n)_n, j = 1, \cdots, m$, satisfy the conditions (10) and that W is compact. Then

$$\lim_{n \to \infty} \left(\overline{\lim_{r \to \infty}} \sup_{w \in W} \left|\hat{\lambda}^{(n)}_{r,H_n}(w) - \lambda(w)\right| \right) = 0 \tag{11}$$

with probability 1.

Training with additive noise means minimizing the estimated loss function $\hat{\lambda}^{(n)}_{r,H_n}$. This leads to an optimal weight vector $\hat{w}^{(n)}_{r,H_n}$, which we hope is close to a vector $\overline{w}$ that minimizes the ideal loss function λ. We will next present a consistency theorem which establishes this desirable behavior of $\hat{w}^{(n)}_{r,H_n}$ in an asymptotic sense.

Let W be compact. Define

$$W^0 = \{\overline{w} \in W | \lambda(\overline{w}) \le \lambda(w) \text{ for all } w \in W\}.$$

The set W^0 consists of those vectors $\overline{w}$ that minimize the loss function λ. Since λ is continuous, W^0 is a compact nonempty subset of W. For $w \in W$ let $d(w, W^0) = \min_{\overline{w} \in W^0} \|w - \overline{w}\|$ be the distance of w from W^0.

Theorem 3: Suppose that the smoothing parameter sequences $(h_n^j)_n, j = 1, \cdots, m$, satisfy the conditions (10) and that W is compact. For $n, r \in \boldsymbol{N}$, let $\hat{w}_{r,H_n}^{(n)}$ be a minimizer of $\hat{\lambda}_{r,H_n}^{(n)}$. Then

$$\lim_{n \to \infty} \left(\overline{\lim_{r \to \infty}} \, d\left(\hat{w}_{r,H_n}^{(n)}, W^0 \right) \right) = 0 \tag{12}$$

with probability 1.

A slight clarification of the statement of the theorem should be made. A minimizing weight vector $\hat{w}_{r,H_n}^{(n)}$, exists because $\hat{\lambda}_{r,H_n}^{(n)}$ is continuous and W is compact but it does not have to be unique. In case there are several minimizing vectors one can choose $\hat{w}_{r,H_n}^{(n)}$ arbitrarily as one of them. Further, $d(\hat{w}_{r,H_n}^{(n)}, W^0)$ may not be a random variable (a Borel measurable function); therefore (12) may not define an event in the probability space considered. What is shown is that there is an event A with $P(A) = 0$ outside which (12) holds. The statement of the theorem becomes precise by assuming that the probability measure P used is complete.

Theorem 3 considers global minimization over a set W of possible weight vectors. However, a local interpretation is immediate if W is chosen as a compact neighborhood of a local minimum $\overline{w}$ of λ. If the minimization of $\hat{\lambda}_{r,H_n}^{(n)}$ is restricted to W and $W^0 = \{\overline{w}\}$, then $\hat{w}_{r,H_n}^{(n)}$ approaches $\overline{w}$ asymptotically (in the sense of (12)) when $n, r \to \infty$.

C. Estimating a Mapping with a Network

Besides pattern classification, another popular way to apply neural networks is to utilize the rich collection of functions representable by a network to approximate a complicated mapping. One example is the use of a network to solve the inverse kinematics problem of robotics. Then the mapping that one wants to approximate maps a target location in three-dimensional space to a set of robot arm joint angles in a Euclidean configuration space.

We assume that the mapping considered is a Borel function $g_0 : \boldsymbol{R}^k \to \boldsymbol{R}^m$ and that data points $(x_0, y_0), y_0 = g_0(x_0)$, are obtained by sampling a k-dimensional random vector X_0 with density f_{X_0}. In the robot example, the use of a random vector X_0 can model the fact that some target locations may be more probable than others. Define $Y_0 = g_0(X_0)$ and $Z_0 = (X_0, Y_0)$. We assume that the actual observations x_0 and y_0 are corrupted by measurement errors. As explained below, this implies that the random vectors X_0 and Y_0 are replaced by new random vectors X and Y such that the distribution of $Z = (X, Y)$ can be described by a probability density. We will therefore carry out the theoretical development assuming that $Z = (X, Y)$ is a random vector taking values in $\boldsymbol{R}^p, p = k + m$, and that Z has density f. We will identify $\boldsymbol{R}^p$ and $\boldsymbol{R}^k \times \boldsymbol{R}^m$ and use the notation $z = (x, y)$, etc.

Assume that we have n training vectors $z_i = (x_i, y_i) \in \boldsymbol{R}^p, i = 1, \cdots, n$. Suppose that K is a density in $\boldsymbol{R}^p$; let $h > 0$ and consider the following procedure for generating new training vectors by introducing additive noise:

Procedure 2:

1) Select $i \in \{1, \cdots, n\}$ with equal probability for each index.
2) Draw a sample s from the density K.
3) Set $z_h^{(n)} = z_i + hs$.

The vectors $z_h^{(n)}$ can be thought of as realizations of a random vector $Z_h^{(n)} = \left(X_h^{(n)}, Y_h^{(n)}\right)$ whose distribution is described by the density

$$f_{n,h}(x, y) = \frac{1}{n} \sum_{i=1}^{n} K_h(x - x_i, y - y_i),$$

where $K_h(x, y) = h^{-p} K(x/h, y/h), (x, y) \in \boldsymbol{R}^k \times \boldsymbol{R}^m$. We note that $f_{n,h}$ is a kernel estimate of f.

The least-squares loss function λ corresponding to Z is

$$\lambda(w) = \int_{\boldsymbol{R}^m} \int_{\boldsymbol{R}^k} \|y - g(x, w)\|^2 f(x, y) \, dx \, dy, \qquad w \in W. \tag{13}$$

When training samples are generated with Procedure 2, the loss function is given by

$$\lambda_h^{(n)}(w) = \int_{\boldsymbol{R}^m} \int_{\boldsymbol{R}^k} \|y - g(x, w)\|^2 f_{n,h}(x, y) \, dx \, dy, \qquad w \in W. \tag{14}$$

It is easy to see that the integrals in (13) and (14) are finite if f and K satisfy the conditions

$$\int_{\boldsymbol{R}^m} \int_{\boldsymbol{R}^k} \|y\|^2 f(x, y) \, dx \, dy < \infty \tag{15}$$

and

$$\int_{\boldsymbol{R}^m} \int_{\boldsymbol{R}^k} \|y\|^2 K(x, y) \, dx \, dy < \infty. \tag{16}$$

Moreover, by [22, Section 16.8(i)], (15) and (16) imply that λ and $\lambda_h^{(n)}$ are continuous. Corresponding to Proposition 1 we have the following result.

Proposition 2: Suppose that for some $\epsilon > 0$,

$$B_\epsilon(f) = \int_{\boldsymbol{R}^m} \int_{\boldsymbol{R}^k} \|y\|^{2+\epsilon} f(x, y) \, dx \, dy < \infty \tag{17}$$

and

$$B_\epsilon(K) = \int_{\boldsymbol{R}^m} \int_{\boldsymbol{R}^k} \|y\|^{2+\epsilon} K(x, y) \, dx \, dy < \infty. \tag{18}$$

Then

$$\sup_{w \in W} \left| \lambda_h^{(n)}(w) - \lambda(w) \right| \leq C_\epsilon \left(\int_{\boldsymbol{R}^m} \int_{\boldsymbol{R}^k} |f_{n,h} - f| \right)^{\epsilon/(2+\epsilon)} \tag{19}$$

where

$$C_\epsilon = 4\Bigg[M_W(g)^{2+\epsilon} + \frac{1}{2}B_\epsilon(f) + 2^\epsilon\left(h^{2+\epsilon}B_\epsilon(K) + \frac{1}{n}\sum_{i=1}^{n}\|y_i\|^{2+\epsilon}\right)\Bigg]^{2/(2+\epsilon)}. \tag{20}$$

Note that conditions (17) and (18) imply (15) and (16), respectively.

The inequality (19) shows that we can try to make $|\lambda_h^{(n)}(w) - \lambda(w)|$ small by choosing K and h so that the L_1 distance $\int_{\boldsymbol{R}^m}\int_{\boldsymbol{R}^k}|f_{n,h} - f|$ becomes small. This gives an insight into what kind of noise one should use in Procedure 2. For further discussion see Section III.

Let us then consider the asymptotic behavior of the proposed training scheme. Thus, we consider an infinite random sample $Z_1, Z_2, \cdots$ and a sequence (h_n) of smoothing parameters.

Theorem 4: Suppose that f and K satisfy (17) and (18), respectively, and that

$$\lim_{n\to\infty} h_n = 0, \qquad \lim_{n\to\infty} n(h_n)^p = \infty. \tag{21}$$

Then,

$$\lim_{n\to\infty}\sup_{w\in W}\left|\lambda_{h_n}^{(n)}(w) - \lambda(w)\right| = 0$$

with probability 1.

The proofs of the next two theorems are not given in the Appendix because they are analogous to the proofs of Theorems 2 and 3.

Theorem 5: Suppose that f, K, and (h_n) satisfy (17), (18), and (21), respectively, and that W is compact. Then

$$\lim_{n\to\infty}\left(\overline{\lim_{r\to\infty}}\sup_{w\in W}\left|\hat{\lambda}_{r,h_n}^{(n)}(w) - \lambda(w)\right|\right) = 0$$

with probability 1.

Theorem 6: Suppose that f, K, and (h_n) satisfy (17), (18), and (21), respectively, and that W is compact. Denote by W^0 the set of minimizers of λ. For $n, r \in \boldsymbol{N}$, let $\hat{w}_{r,h_n}^{(n)}$ be a minimizer of $\hat{\lambda}_{r,h_n}^{(n)}$. Then

$$\lim_{n\to\infty}\left(\overline{\lim_{r\to\infty}} d\left(\hat{w}_{r,h_n}^{(n)}, W^0\right)\right) = 0$$

with probability 1.

We will now explain how the random vector $Z = (X, Y)$ is obtained from $Z_0 = (X_0, Y_0) = (X_0, g_0(X_0))$. The measurement errors in X_0 and Y_0 are modeled by random vectors N_x and N_y with densities f_{N_x} and f_{N_y}, respectively. Two different cases are considered:

1) Only Y_0 is corrupted: $X = X_0$ and $Y = g_0(X_0) + N_y$.
2) Both X_0 and Y_0 are corrupted: $X = X_0 + N_x$ and $Y = g_0(X_0) + N_y$.

In case 1, Z has density

$$f(x, y) = f_{N_y}(y - g_0(x))f_{X_0}(x) \tag{22}$$

and in case 2, Z has density

$$f(x, y) = \int_{\boldsymbol{R}^k} f_{N_y}(y - g_0(u))f_{N_x}(x - u)f_{X_0}(u)\,du, \quad (x, y) \in \boldsymbol{R}^k \times \boldsymbol{R}^m. \tag{23}$$

Formulas (22) and (23) are easy to verify by showing that the characteristic function of Z is equal to the Fourier transform of f.

Suppose that

$$E\left(\|g_0(X_0)\|^2\right) = \int_{\boldsymbol{R}^k}\|g_0(x)\|^2 f_{X_0}(x)\,dx < \infty. \tag{24}$$

In case 1, when a network $x \mapsto g(x, w)$ estimates the function g_0, a natural performance measure is the average value of the squared error $\|g_0(x) - g(x, w)\|^2$,

$$E\left(\|g_0(X_0) - g(X_0, w)\|^2\right) = \int_{\boldsymbol{R}^k}\|g_0(x) - g(x, w)\|^2 \cdot f_{X_0}(x)\,dx. \tag{25}$$

On the other hand, in case 2 it is natural to measure how much $g_0(x_0)$ differs from $g(x, w)$, where $x = x_0 + n_x$ is the observed value of x_0 corrupted by a measurement error. Thus, we consider the performance measure

$$E\left(\|g_0(X_0) - g(X, w)\|^2\right). \tag{26}$$

Making use of conditional expectation as in (5), one can replace (26) by the more easily computable performance measure

$$\int_{\boldsymbol{R}^k}\left\|\frac{f_{N_x} * g_0 f_{X_0}(x)}{f_{N_x} * f_{X_0}(x)} - g(x, w)\right\|^2 f_{N_x} * f_{X_0}(x)\,dx, \tag{27}$$

where for almost all $x \in \boldsymbol{R}^k$,

$$\frac{f_{N_x} * g_0 f_{X_0}(x)}{f_{N_x} * f_{X_0}(x)} = E(g_0(X_0)|X = x) \quad \text{if } f_{N_x} * f_{X_0}(x) \neq 0.$$

It is easy to verify that in both case 1 and case 2 condition (17) is satisfied if, for some $\epsilon > 0$,

$$E\left(\|N_y\|^{2+\epsilon}\right) = \int_{\boldsymbol{R}^m}\|y\|^{2+\epsilon} f_{N_y}(y)\,dy < \infty$$

and

$$E\left(\|g_0(X_0)\|^{2+\epsilon}\right) = \int_{\boldsymbol{R}^k}\|g_0(x)\|^{2+\epsilon} f_{X_0}(x)\,dx < \infty. \tag{28}$$

Note that (28) implies (24).

III. Designing Noise

The analysis of the previous section suggests that the noise density K and the smoothing parameter $h > 0$ should be chosen so that the L_1 distance

$$\int_{R^d} |f_{n,h} - f| \tag{29}$$

between a density f and its n-point kernel estimate $f_{n,h}$ becomes as small as possible. In the notation of the previous section, for a pattern classifier network $d = k$ and for a mapping estimating network $d = k + m = p$ (see (9) and (19)). The problem of selecting a kernel and its smoothing parameter so that the kernel estimate in some sense optimally approximates an original density function is generally difficult and a large amount of work has been published on it during the last few decades. A very accessible general introduction to this topic can be found in [21]. For results specifically in the context of the L_1 norm (29), see [23] and [24].

In practice, there are two design decisions to be made. First, the "shape" of K is selected. In some cases it is possible to base this decision on optimization (see [23, ch. 7]), but often in practice one simply chooses K to be Gaussian, i.e., a normal density function. We have experimented with one optimal kernel and Gaussians. It may also be useful to estimate first the covariance structure of f and enforce this same structure on K (cf. [25]). Alternatively, one can first "prewhiten" the data to have approximately unit covariance and then use a radially symmetric kernel [21, section 4.2.1]. Second, and more importantly, a good value for the smoothing parameter h must be found. This is our main interest and we have considered three different approaches: 1) use of explicit formulas obtained by minimizing with respect to h an upper bound for the expected value of (29), 2) maximizing with respect to h a modified likelihood function obtained by cross-validation, and 3) optimizing with respect to h the performance of $f_{n,h}$ directly at the given task.

A. Explicit Formulas for the Smoothing Parameter

To get an explicit minimizable upper bound for the expectation $E\int_{R^d} |f_{n,h} - f|$, assumptions about the smoothness of f and K have to be made. Let $d = 1$, suppose that the density f is twice differentiable, $f'' \in L_1(\boldsymbol{R})$, and that $\int_R x^2 f(x)\,dx < \infty$. Suppose further that K is even and that $\int_R (1 + x^2)K(x)^2\,dx < \infty$. It is shown in [23, theorem 7.6] that minimization of a suitable upper bound then leads to the formula

$$h_n = \frac{\left(\int_R K^2\right)^{1/5}\left(\int_R \sqrt{f}\right)^{2/5}}{\left(2\int_R |f''| \int_R x^2 K(x)\,dx\right)^{2/5}}\, n^{-1/5} \tag{30}$$

for the optimal smoothing parameter. An optimal kernel K turns out to be the Bartlett kernel $K(x) = (3/4)(1 - x^2)_+$ (see [23, lemma 7.4]).

Note that in order to use a formula like (30) we must first estimate expressions that involve the function f, the very quantity we set out to estimate in the first place. This clearly limits the applicability of explicit analytical formulas such as (30) (see also the discussion in [23, pp. 126–127]). We experimented with known artificial densities, so precise evaluation of the smoothing parameter from (30) was possible. For the Bartlett kernel and a normal density f with variance σ^2, formula (30) reduces approximately to $h_n = 1.8218\sigma n^{-1/5}$. It is also possible to consider generalizations of (30) to $d > 1$ [26]. If I is the $d \times d$ identity matrix and f is a normal density function with the covariance matrix $\sigma^2 I$, then for the Gaussian kernel K with the covariance matrix I one can generalize (30) to get

$$h_n = \left(8^{d/4-1} d^{-d/2+1} e^{d/2} \Gamma(d/2)\right)^{2/(d+4)} \sigma n^{-1/(d+4)}.$$

B. Maximizing a Cross-Validated Likelihood Function

An ideal way to choose the smoothing parameter would be a completely data driven method that would allow us to compute a good value of h directly from the available data $X_1, \cdots, X_n$. The following method was proposed in [27] and [28] (see also [24, ch. 6]). Denote by $f_{n,h,i}$ the kernel estimate of f obtained by leaving out the ith observation, that is,

$$f_{n,h,i}(x) = \frac{1}{(n-1)h^d} \sum_{\substack{j=1 \\ j \neq i}}^{n} K_h(x - X_j).$$

Now consider the cross-validated likelihood-like expression

$$L(h) = \prod_{i=1}^{n} f_{n,h,i}(X_i) \tag{31}$$

and choose h so that $L(h)$ is maximized. An intuitive motivation for this procedure is the following. For each i, $f_{n,h,i}(X_i)$ can be viewed as a measure of the degree to which the samples $X_j, j \neq i$, when smoothed with h, "explain" the observation X_i. A large value of $f_{n,h,i}(X_i)$ indicates a good "fit" and hence a good h, whereas a small value of $f_{n,h,i}(X_i)$ indicates a poor value of h. These n measures are then combined to give (31).

A major difficulty with such cross-validated density estimation methods is to prove that the resulting kernel estimate is consistent. In our context we would like to know that (29) tends to zero as n tends to infinity. The first such consistency proof for the cross-validated density estimate based on maximizing (31) appears in [29]. For a more general version see [24, ch. 6, theorem 4]. For problems with the density estimates obtained from (31) and some suggested modifications, see, for example, [30].

As a practical note, optimizing $L(h)$ in our experiments turned out to be easy. In most cases a global maximum was the only extremum present (cf. [27]).

C. Optimizing the Smoothing Parameter by Jackknifing

The first two smoothing parameter selection methods were more or less explicitly targeted to make (29) small. Our third approach is to optimize the performance of the kernel estimates directly at the given task. We have experimented with this method only in a pattern recognition problem with two classes. Thus, we try to select the parameters h^1 and h^2

so that the kernel estimates $f^1_{N_n^1,h^1}$ and $f^2_{N_n^2,h^2}$ of the two class densities perform as well as possible at the classification task considered.

As discussed in Section I, the use of noise could be motivated by a small number of available training vectors. Thus, it may not be possible to estimate reliably the error rate of a $f^1_{N_n^1,h^1}$, $f^2_{N_n^2,h^2}$-based classifier by the usual method of partitioning the training data into two independent sets of training and test vectors. More efficient use of the available data may be needed. We reused the data many times to obtain several pairs of training and test vector sets. The resulting large number of required training epochs would have made it impractical to base the selection of h^1 and h^2 on the network classifier itself. Instead, we decided to use a separate fast classifier. The classifier which naturally suggests itself in the present situation is the kernel-estimate-based Bayesian classifier (cf. [26], [31], and [32]). By [24, ch. 10, theorem 1], such selection of the smoothing parameters can also be viewed as being related to making (29) small.

In reusing the training data we employed the following jackknife procedure, proposed in [25], [33], and [34]. Divide the set of the n available training samples into two subsets T_1 and T_2 such that the set T_j contains all vectors from class j. Let (h^1, h^2) be fixed. For each pair $(x_1, x_2) \in T_1 \times T_2$ form the density estimates $f^1_{N_n^1-1,h^1}$ and $f^2_{N_n^2-1,h^2}$ of f^1 and f^2 using the samples $T_1 \backslash \{x_1\}$ and $T_2 \backslash \{x_2\}$, respectively. Now classify x_1 and x_2 with the Bayesian classifier based on the densities $f^1_{N_n^1-1,h^1}$ and $f^2_{N_n^2-1,h^2}$. Thus, an input vector x is placed in class 1 if $P^1 f^1_{N_n^1-1,h^1}(x) > P^2 f^2_{N_n^2-1,h^2}(x)$ and similarly for class 2. By repeating this for all pairs (x_1, x_2) one gets an estimate of the classifying error associated with (h^1, h^2). Suitable smoothing parameters h^1 and h^2 are found by repeating this process for several pairs (h^1, h^2).

A few critical remarks are in order. The above procedure may become quite time consuming if a large range of h^1 and h^2 values must be examined. We avoided this problem by using as an initial estimate for (h^1, h^2) the values obtained from the cross-validation method. The search in the (h^1, h^2) space can then be limited to a neighborhood of this initial estimate. Another problem is the generality of this approach. While the availability of a suitable fast classifier makes this method feasible in the present problem, it is not clear how to handle other possible applications. Finally, there do not seem to be any theoretical results to guarantee the consistency of the resulting density estimates.

IV. Experimental Results

In Section II we analyzed training with noise theoretically. We showed its asymptotic consistency provided that the smoothing parameter sequences satisfy certain conditions. In practice, there are only a finite number of original samples available, computational considerations limit the number of samples to be generated from the kernel estimate, and there is no guarantee of finding the global minimum of the loss function. To find out how the proposed method performs in practice, it is therefore necessary to investigate its properties experimentally.

In the experiments the loss functions were minimized using Marquardt's method [35]. One iteration step then requires a computation time proportional to l^3, where l is the number of adaptable weights. Therefore this method could not be recommended if there were radically more weights to be optimized than in our experiments. The required derivatives of the loss function were calculated with the well-known back-propagation formulas [15]. The theory asks us to minimize the loss function over a compact set W for which we chose the set of l-tuples representable by l floating point numbers. Because there are no reliable global minimization methods available, we employed the heuristic of starting local optimizations from many starting points. Specifically, in each minimization we started the iteration from three random starting points, continuing the calculation until 100 iterations were completed, and then picked from the three results the one yielding the minimum value for the loss function.

We experimented with feedforward layered networks with one hidden layer. The notation FN-k-h-m identifies such a network, with k denoting the input dimension, h the number of units in the hidden layer, and m the output dimension. Such a network has $h(k+1)+m(h+1)$ adaptable weights. We used the logistic function $t \mapsto (1+e^{-t})^{-1}$ as activation function in the hidden and output layers. Therefore the ranges of all the m components of $g(x, w)$ are included in the interval]0, 1[.

A. Classification Experiments

We experimented with the synthetic classification problems studied by Kohonen *et al.* [36]. These are k-dimensional two-class problems with the following normal class conditional densities:

$$\textbf{easy}(k): \mu^1 = 0, \quad C^1 = I; \quad \mu^2 = 2.32e_1, \quad C^2 = 4I$$
$$\textbf{hard}(k): \mu^1 = 0, \quad C^1 = I; \quad \mu^2 = 0, \quad C^2 = 4I$$

Here μ^j and C^j denote the mean vector and the covariance matrix of the class-conditional density of class j, I is the $k \times k$ identity matrix, and e_1 is the unit vector $(1, 0, \cdots, 0) \in \boldsymbol{R}^k$. The class *a priori* probabilities P^1 and P^2 are equal to $1/2$ in both easy and hard cases.

Recall from subsection II-B that we train the network to map a vector x from class j to the unit vector e_j, where now $e_j \in \boldsymbol{R}^2$. The justification for this procedure was that this kind of training produces an approximation to the vector of *a posteriori* probabilities $P(x)$ and thus an approximation to the optimal Bayesian classifier minimizing the error of misclassification. However, in the present experiments we scaled the desired outputs so that the networks were trained to associate vectors from class 1 with the vector (0.9, 0.1) and from class 2 with the vector (0.1, 0.9), i.e., instead of training the network to associate X with $e \circ J$ we trained it to associate X with $a(e \circ J) + b$, where $a = 0.8$ and $b = 0.1$. Then the trained network $x \mapsto g(x, \overline{w})$ will approximate the mapping $x \mapsto E(a(e \circ J) + b | X = x)$. By linearity of conditional expectation,

$$\begin{aligned} E(a(e \circ J) + b | X = x) &= aE(e \circ J | X = x) + b \\ &= aP(x) + b. \end{aligned}$$

TABLE I
RESULTS FOR **easy**(1)

	Error rate		h^1		h^2	
	mean	std	mean	std	mean	std
easy(1),	$n = 10$,		$r = 100$,		$l = 14$	
O	0.295	0.0837				
EF(B)	0.253	0.0539	1.32		2.64	
EF	0.261	0.0766	0.596		1.19	
XV	0.278	0.0773	0.900	0.41	1.58	0.49
JK	0.283	0.0851	1.01	0.61	1.70	0.86
easy(1),	$n = 28$,		$r = 280$,		$l = 14$	
O	0.254	0.0403				
EF(B)	0.221	0.0202	1.07		2.15	
EF	0.230	0.0322	0.485		0.971	
XV	0.229	0.0317	0.628	0.24	1.26	0.44
JK	0.234	0.0385	0.738	0.40	1.34	0.67

TABLE II
RESULTS FOR **hard**(1)

	Error rate		h^1		h^2	
	mean	std	mean	std	mean	std
hard(1),	$n = 10$,		$r = 100$,		$l = 14$	
O	0.450	0.0507				
EF(B)	0.405	0.0467	1.32		2.64	
EF	0.401	0.0525	0.596		1.19	
XV	0.422	0.0605	0.869	0.41	1.54	0.49
JK	0.425	0.0665	0.963	0.63	1.85	0.97
hard(1),	$n = 28$,		$r = 280$,		$l = 14$	
O	0.395	0.0467				
EF(B)	0.375	0.0329	1.07		2.15	
EF	0.382	0.0367	0.485		0.971	
XV	0.382	0.0429	0.637	0.21	1.27	0.44
JK	0.380	0.0379	0.648	0.30	1.49	0.78

If, as before, we use the classification rule whereby the network classifies x to class j_0 if the j_0th output is the maximum output, then we again arrive at a classifier which approximates the optimal Bayesian classifier. The reason for scaling the desired outputs is connected with the fact that in the example classification tasks the *a posteriori* probabilities $P^j(x)$ take on values near 0 and 1. Approximating such a function well with a network which has the logistic activation function in the output layer would require weights of large magnitude between the hidden and the output layer. With scaling, however, the magnitudes of the weights need not be as large and the resulting approximations seem to be better, at least in one-dimensional examples. Notice that in a two-class case we would not really need separate approximations to $P^1(x)$ and $P^2(x)$ because $P^2(x) = 1 - P^1(x)$ with probability 1; thus a network with only one output would suffice for classification. We did not adopt this improvement because here we only wanted to demonstrate the principle of training with noise.

Our classification experiments are summarized in Tables I through IV. These statistics are based on 100 repeated experiments in the one-dimensional cases and on 25 repeated experiments in the five-dimensional cases. One experiment consists of 1) generating n samples from the original densities, $n/2$ from each class, and then designing a classifier by minimizing the loss function $\hat{\lambda}_n$ based on the n original samples (label O in the tables) and 2) generating r samples from the kernel estimates obtained by choosing the smoothing parameters h^1 and h^2 according to the three different methods introduced in Section III and then designing classifiers by minimizing the loss function $\hat{\lambda}^{(n)}_{r,h}$, $h = (h^1, h^2)$. The three methods are labeled in the tables as EF (explicit formulas), XV (cross-validation) and JK (jackknifing). We used the standard Gaussian kernel $K(x) = (2\pi)^{-k/2} \exp(-\|x\|^2/2)$, $x \in \boldsymbol{R}^k$, in all the cases but the one-dimensional ones labeled EF(B), where we used explicit formulas and the Bartlett kernel (see subsection III-A). The network architectures are FN-1-3-2 in the one-dimensional cases and FN-5-8-2 in the five-dimensional cases. In the jackknifing experiments the following values were tried: $h^i = a_j h^i_{XV}$, $a_j = 1.3^j$, $j =$

TABLE III
RESULTS FOR **easy**(5)

	Error rate		h^1		h^2	
	mean	std	mean	std	mean	std
easy(5),	$n = 62$,		$r = 620$,		$l = 66$	
O	0.301	0.033				
EF	0.179	0.014	0.832		1.66	
XV	0.182	0.018	0.844	0.075	1.60	0.15
JK	0.187	0.021	0.670	0.18	2.06	0.59
easy(5),	$n = 66$,		$r = 660$,		$l = 66$	
O	0.298	0.046				
EF	0.175	0.013	0.826		1.65	
XV	0.174	0.015	0.816	0.049	1.58	0.12
JK	0.175	0.020	0.696	0.10	1.80	0.49
easy(5),	$n = 132$,		$r = 1320$,		$l = 66$	
O	0.249	0.032				
EF	0.149	0.0077	0.765		1.53	
XV	0.150	0.0087	0.750	0.032	1.51	0.11
JK	0.152	0.016	0.578	0.098	1.73	0.61
easy(5),	$n = 264$,		$r = 2640$,		$l = 66$	
O	0.209	0.027				
EF	0.135	0.0071	0.708		1.42	
XV	0.134	0.0081	0.688	0.042	1.39	0.083
JK	0.129	0.0076	0.562	0.082	1.39	0.46

TABLE IV
RESULTS FOR **hard**(5)

hard(5),	$n = 132$,		$r = 1320$,		$l = 66$	
	Error rate		h^1		h^2	
	mean	std	mean	std	mean	std
O	0.306	0.029				
EF	0.215	0.0093	0.765		1.53	
XV	0.211	0.015	0.738	0.060	1.46	0.087
JK	0.206	0.016	0.529	0.098	1.96	0.49

$-2, -1, 0, 1, 2$. Here h^i_{XV} denote the values returned by cross-validation. Out of these 5^2 possibilities the pair giving best performance was picked out as described in subsection III-C. We have characterized the distribution of the error rate (i.e., probability of misclassification) of the resulting classifiers by giving its sample mean and standard deviation. Also, the sample means and standard deviations of the smoothing parameters are given where appropriate. As the class-conditional densities were known, the error rate of a classifier could be estimated with the risk averaging method [37, ch. 10.8.1]; we generated 5000 new samples from both classes and averaged the probability of misclassification at each point.

Tables I through IV show the results for the tasks **easy**(1), **hard**(1), **easy**(5), and **hard**(5). The error rates of the Bayesian classifiers for these tasks are **easy**(1):0.200, **hard**(1): 0.339, **easy**(5): 0.098, and **hard**(5): 0.148. The results show that training with noise was somewhat better in the one-dimensional cases and clearly better in the five-dimensional cases than training with the original samples. Notice also that in the one-dimensional examples where the smoothing parameter is chosen with explicit formulas, the results are practically as good with the Gaussian kernel as with the Bartlett kernel. However, the h values good for one kernel should not be used to train with noise derived from a different kernel as this may lead to performance deterioration; e.g., if we use the h values of Table I on rows labeled EF with the Bartlett kernel we obtain the mean error rates 0.268 ($n = 10$) and 0.244 ($n = 28$). In the first three tasks we experimented with many choices of n, leading to both underdetermined and overdetermined least-squares problems when minimizing the loss function $\hat{\lambda}_n$. The distribution of error rate sharpens and moves toward lower values as n increases. This effect is illustrated in Fig. 2, where we show histogram-based probability density estimates of the error rate corresponding to the second and fourth cases of Table III.

Fig. 3 demonstrates the dependence of the error rate in the task **easy**(5) on the smoothing parameters when $h^1 = h^2$ and the Gaussian kernel is used.

B. Mapping Estimation Experiments

We tried to fit a network g to data originating from the function

$$g_0(x) = a \sin x + b, \quad x \in \mathbf{R}, \quad a = 0.4, \quad b = 0.5.$$

The condition (28) of subsection II-C is therefore satisfied. We considered the cases 1 and 2 introduced in subsection II-C.

In case 1, the data are independent samples from the random vector (X, Y), where $X = X_0$ and $Y = g_0(X_0) + N_y$. We have taken X_0 as uniformly distributed on $[-\pi, \pi]$ and assumed that N_y is normal with mean 0 and variance σ_y^2, i.e., $N_y \sim N(0, \sigma_y^2)$. The performance measure (25) is then

$$J_1 = \frac{1}{2\pi} \int_{-\pi}^{\pi} (a \sin x + b - g(x, w))^2 \, dx.$$

In case 2, $X = X_0 + N_x$ and $Y = g_0(X_0) + N_y$, where $X_0 \sim N(0, \sigma^2), N_x \sim N(0, \sigma_x^2)$, and $N_y \sim N(0, \sigma_y^2)$. The

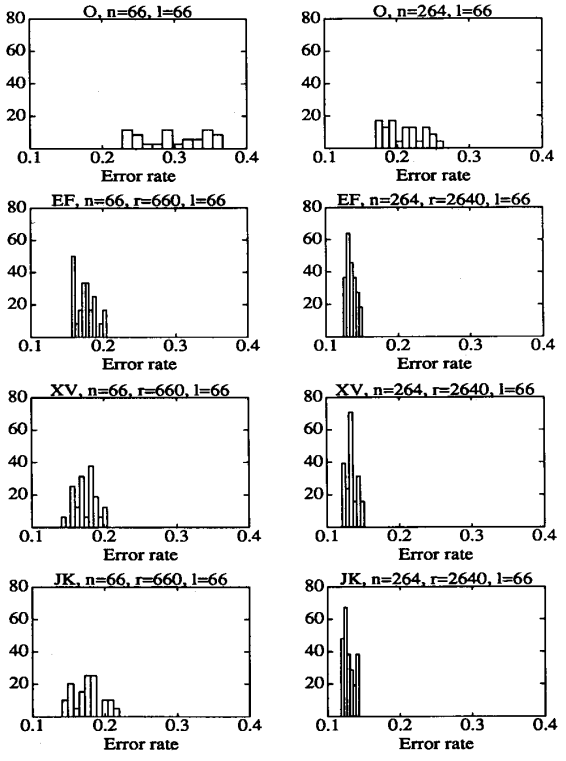


Fig. 2. Probability density estimates for the error rates of FN-5-8-2 classifiers for the task **easy**(5) for two values of n (the number of original samples) and for different methods of adding noise (O: original samples only, EF: explicit formulas, XV: cross validation, JK: jackknifing).

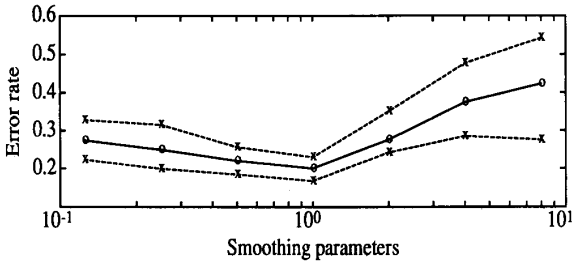


Fig. 3. Error rate of an FN-5-8-2 classifier in the task **easy**(5) when $h^1 = h^2 = 2^j, j = -3, -2, \cdots, 3$. The number of original samples $n = 66$ and of artificial samples $r = 660$. The mean error rate (solid line) and its minimum and maximum (dashed lines) based on 25 repetitions are shown.

performance measure (27) then equals

$$J_2 = \frac{1}{\sqrt{2\pi}\sqrt{\sigma_x^2 + \sigma^2}} \int_R [\tilde{g}_0(x) - g(x, w)]^2 \cdot \exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2 + \sigma^2}\right)\right] dx, \tag{32}$$

where

$$\tilde{g}_0(x) = a \exp\left[-\frac{1}{2}\left(\frac{\sigma_x^2 \sigma^2}{\sigma_x^2 + \sigma^2}\right)\right] \sin\left(\frac{\sigma^2 x}{\sigma_x^2 + \sigma^2}\right) + b, \quad x \in \boldsymbol{R}.$$

TABLE V
RESULTS FOR MAPPING ESTIMATION EXPERIMENTS

$\sigma_y = 0.1$,			$l = 40$		$r = 10_n$	
	$O : J_1$		$XV : J_1$		$XV : h$	
n	mean	std	mean	std	mean	std
36	0.0265	0.018	0.0104	0.0050	0.158	0.020
40	0.0218	0.016	0.0104	0.0079	0.146	0.028
80	0.00764	0.0048	0.00526	0.0018	0.111	0.010
$\sigma = \frac{3}{4}\pi$,			$\sigma_x = \sigma_y = 0.1$,		$l = 40$	$r = 10_n$
	$O : J_2$		$XV : J_2$		$XV : h$	
n	mean	std	mean	std	mean	std
36	0.0301	0.020	0.0246	0.011	0.313	0.088
40	0.0282	0.019	0.0230	0.0088	0.312	0.079
80	0.0119	0.0071	0.00939	0.0044	0.232	0.083

Note that even for a perfect fit, i.e., for a weight vector w such that (32) vanishes, the network will not realize the function g_0 but, instead, the modified function $\tilde{g}_0$.

Table V summarizes our results for the two cases. To calculate the numbers on one row we repeated the following experiment 100 times in both cases. First, generate n original samples from the random vector (X, Y). Next, select the smoothing parameter using cross-validation and the Gaussian kernel and generate r samples from the kernel estimate. Finally, train an FN-1-13-1 network first with the original samples and then separately with the kernel estimate samples and assess the performance of the trained networks by calculating the performance measure J_i by numerical integration. Notice that, in all cases, adding noise seems to have improved learning. This is in part due to the fact that σ_x and σ_y have been selected favorably. Cross-validation tends to choose large values for h if σ_x and σ_y are too large or if there are originally too few samples compared with the spread of the samples. This can result in deterioration in learning when additive noise is used in training. Fig. 4 shows how adding noise can improve generalization when the smoothing parameter is chosen appropriately.

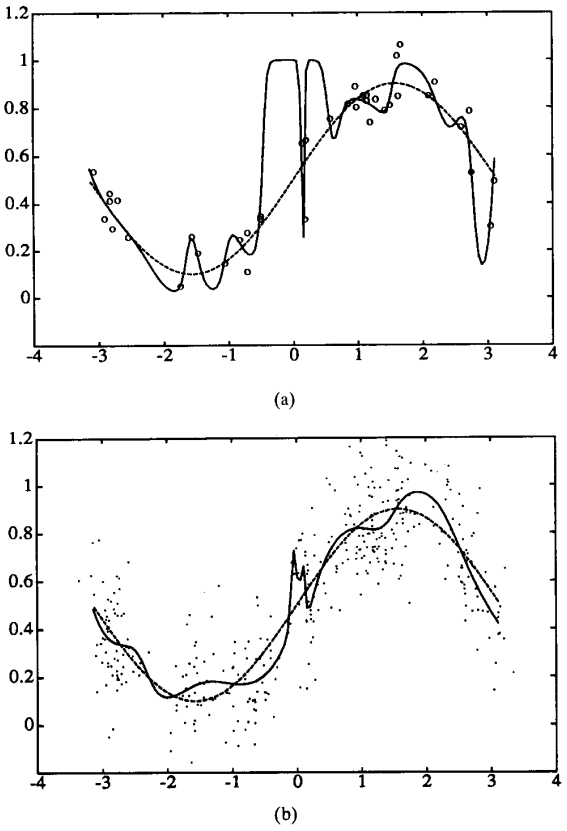


Fig. 4. Approximating noise corrupted $a \sin x + b$ with FN-1-13-1 network. Dashed line: $0.4 \sin x + 0.5$; Solid line: network output trained with (a) $n = 40$ original samples (circles) and (b) $r = 400$ artificial samples (dots). Parameters: 40 adaptable weights, $\sigma_y = 0.1$, Gaussian kernel with smoothing parameter $h = 0.13$.

V. CONCLUSIONS

We have discussed the training of feedforward layered neural networks. To improve the generalization capability of such networks, the addition of noise to the available training vectors has sometimes been suggested and used with success. By interpreting additive noise as simulating a kernel estimate of an original data density we have put this training method into a mathematically analyzable form in two typical neural network applications, pattern classification and mapping estimation. The analysis shows that, under very general conditions, this training method produces asymptotically consistent estimates of optimal weight vectors. Further, practical methods for selecting the characteristics of the additive random noise can be suggested.

In order to obtain weight vectors $\hat{w}_{h,r}^{(n)}$ that remain close to a minimizing weight vector $\overline{w}$ of the loss function λ, the estimated loss function $\hat{\lambda}_{h,r}^{(n)}$ should approximate the ideal loss function λ as well as possible. The inequality (8) and the uniform law of large numbers show that this happens if $\sup_{w \in W} |\lambda_h^{(n)}(w) - \lambda(w)|$ is small. Through Propositions 1 and 2 this leads to the idea of minimizing (29) as a useful condition in choosing the characteristics of the additive noise.

The goal of network training can be regarded as finding a least-squares solution $\overline{w}$ to the system of equations $y_i = g(x_i, w)$, $i = 1, \cdots, n$. In an underdetermined situation, i.e., when the number of available training vectors n is smaller than the number of weights l, it is easy to understand intuitively why adding random noise can be helpful. Owing to the nonuniqueness of minimizing vectors, the set of minimizing vectors of $\hat{\lambda}_n$ can contain large submanifolds of $\boldsymbol{R}^l$. Thus, depending on the initial value of w, minimization of the loss function $\hat{\lambda}_n$ can produce weight vectors that are far from minimizing vectors of λ. Adding random noise alleviates the nonuniqueness problem by making the system overdetermined. Further, by choosing h appropriately, i.e., by minimizing $\sup_{w \in W} |\hat{\lambda}_{h,r}^{(n)}(w) - \lambda(w)| \approx \sup_{w \in W} |\lambda_h^{(n)}(w) - \lambda(w)|$, the solution $\hat{w}_{h,r}^{(n)}$ of this new, overdetermined problem can be forced near $\overline{w}$. The simulations show that this method can work

well even when the original problem is slightly overdetermined to begin with.

The numerical experiments conducted support the applicability of the proposed network training method provided that the smoothing parameter h can be chosen appropriately. In a favorable situation, e.g., in all the cases reported in Section IV, each of the tested noise design alternatives improved the generalization capability of the network. If a single general-purpose noise design method should be suggested, we would pick maximizing the cross-validated likelihood function. This method is easy to implement, is completely data-driven, and has a validity that is supported by theoretical consistency results. The use of analytical formulas can produce excellent results but may be hard to use in many practical applications. The jackknife optimization of the smoothing parameter can yield good results given a reasonable initial estimate, but it is not clear whether it can be applied to other applications besides pattern classification.

APPENDIX

Proof of Proposition 1: Let $w \in W$ be fixed. Using the conditional densities we get

$$\lambda(w) = \sum_{j=1}^{m} P^j \int_{R^k} \|e_j - g(x,w)\|^2 f^j(x)\,dx$$

$$\lambda_h^{(n)}(w) = \sum_{j=1}^{m} P_n^j \int_{R^k} \|e_j - g(x,w)\|^2 f^j_{N_n^j,h^j}(x)\,dx.$$

Thus, setting $C = (1 + M_W(g))^2$,

$$\left|\lambda_h^{(n)}(w) - \lambda(w)\right| =$$

$$\left|\sum_{j=1}^{m} \int_{R^k} \|e_j - g(x,w)\|^2 \left(P_n^j f^j_{N_n^j,h^j}(x) - P^j f^j(x)\right)dx\right|$$

$$\le C \sum_{j=1}^{m} \int_{R^k} \left|P_n^j f^j_{N_n^j,h^j} - P^j f^j\right|$$

$$= C \sum_{j=1}^{m} \int_{R^k} \left|(P_n^j - P^j + P^j) f^j_{N_n^j,h^j} - P^j f^j\right|$$

$$\le C\left(\sum_{j=1}^{m} \left|P_n^j - P^j\right| \int_{R^k} f^j_{N_n^j,h^j} + \sum_{j=1}^{m} P^j \int_{R^k} \left|f^j_{N_n^j,h^j} - f^j\right|\right)$$

$$\le C\left(\sum_{j=1}^{m} \left|P_n^j - P^j\right| + \sum_{j=1}^{m} \int_{R^k} \left|f^j_{N_n^j,h^j} - f^j\right|\right),$$

where the last inequality follows from $\int_{R^k} f^j_{N_n^j,h^j} = 1, P^j \le 1,\ j = 1, \cdots, m$. □

We note that the continuity of $w \mapsto g(x, w)$ was not used in the proof.

Proof of Theorem 1: The quantities $\sup_{w \in W} |\lambda_{H_n}^{(n)}(w) - \lambda(w)|, n \in \boldsymbol{N}$, are random variables because λ and $\lambda_{H_n}^{(n)}$ are continuous functions of w (cf. [22, 16.8(i)]) and the supremum can therefore be taken over a dense countable subset of W. By Proposition 1 it is enough to show that, for each j, $\left|P_n^j - P^j\right|$ and $\int_{R^k} \left|f^j_{N_n^j,H_n^j} - f^j\right|$ tend to zero with probability 1 as $n \to \infty$.

The random variable N_n^j has a binomial distribution with parameters n and P^j. To handle $\left|P_n^j - P^j\right|$ we need estimates for the tail probabilities associated with the binomial distribution. For the proof of the following inequalities see [38, pp. 160–161]. Let $0 < \epsilon < \min\left\{P^j, 1 - P^j\right\}$. Then

$$P\left(\frac{N_n^j}{n} - P^j < -\epsilon\right) \le \exp\left(-\frac{\epsilon^2}{2P^j} n\right)$$

$$P\left(\frac{N_n^j}{n} - P^j > \epsilon\right) \le \exp\left(-\frac{\epsilon^2}{2(1-P^j)} n\right) \quad \text{(A1)}$$

for all $n \in \boldsymbol{N}$. Thus, for $0 < \epsilon < \min\left\{P^j, 1 - P^j\right\}$ we have

$$P\left(\left|P_n^j - P^j\right| > \epsilon\right) = P\left(\left|\frac{N_n^j}{n} - P^j\right| > \epsilon\right)$$

$$\le \exp\left(-\frac{\epsilon^2}{2P^j} n\right) + \exp\left(-\frac{\epsilon^2}{2(1-P^j)} n\right)$$

for all $n \in \boldsymbol{N}$. Consequently, $\sum_{n=1}^{\infty} P\left(\left|P_n^j - P^j\right| > \epsilon\right) < \infty$ and it follows from the Borel–Cantelli lemma that $\left|P_n^j - P^j\right| \to 0$ with probability 1.

Then, let $0 < \epsilon < P^j$ and denote by A_n the event

$$\int_{R^k} \left|f^j_{N_n^j,H_n^j} - f^j\right| > \epsilon.$$

Using the notation $Z_i = (X_i, e \circ J_i), i \in \boldsymbol{N}$, we get

$$P(A_n) = \sum_{j_1,\cdots,j_n=1}^{m} P(A_n | J_i = j_i, i = 1, \cdots, n) \cdot P(J_i = j_i, i = 1, \cdots, n)$$

$$= \sum_{r=0}^{n} \sum_{\substack{j_1,\cdots,j_n \\ N_n^j = r}} P(A_n | J_i = j_i, i = 1, \cdots, n) \cdot P(J_i = j_i, i = 1, \cdots, n),$$

where the condition $N_n^j = r$ refers to the nonrandom quantity N_n^j, the number of those indices i for which $j_i = j$. For $N_n^j = r$, the conditional probability $P(A_n | J_i = j_i, i = 1, \cdots, n)$ is the probability of

$$\int_{R^k} \left|f^j_{r,h_r^j} - f^j\right| > \epsilon \quad \text{(A2)}$$

where the kernel estimate f^j_{r,h_r^j} is obtained by using r samples of a random vector with density f^j. Call the event (A2) B_r. We can then apply the L_1-consistency result of [39, Theorem 1], which states that, under the conditions (10), there is $\rho > 0$ and $r_0 \in \boldsymbol{N}$ such that

$$P(B_r) \le \exp(-r\rho) \qquad \text{for } r \ge r_0. \quad \text{(A3)}$$

We then get, for $n > r_0/(P^j - \epsilon)$,

$$P(A_n) = \sum_{r=0}^{n} \sum_{\substack{j_1,\cdots,j_n \\ N_n^j = r}} P(B_r)P(J_i = j_i, i = 1, \cdots, n)$$

$$= \sum_{r=0}^{n} P(B_r)P(N_n^j = r) = \sum_{0 \le r < (P^j-\epsilon)n} P(B_r)$$

$$\cdot P(N_n^j = r) + \sum_{(P^j-\epsilon)n \le r \le n} P(B_r)P(N_n^j = r)$$

$$\le \sum_{0 \le r < (P^j-\epsilon)n} P(N_n^j = r) + \sum_{(P^j-\epsilon)n \le r \le n}$$

$$\cdot P(B_r)P(N_n^j = r) \le P\left(\frac{N_n^j}{n} - P^j < -\epsilon\right)$$

$$+ \sum_{(P^j-\epsilon)n \le r \le n} \exp(-r\rho)P(N_n^j = r)$$

$$\le \exp\left(-\frac{\epsilon^2}{2P^j}n\right) + \exp\left(-\rho(P^j - \epsilon)n\right),$$

where the last two inequalities follow from (A3) and (A1). Thus,

$$\sum_{n=1}^{\infty} P\left(\int_{R^k} \left|f^j_{N_n^j, H_n^j} - f^j\right| > \epsilon\right) = \sum_{n=1}^{\infty} P(A_n) < \infty$$

so that $\lim_{n\to\infty} \int_{R^k} |f^j_{N_n^j,H_n^j} - f^j| = 0$ with probability 1, again by the Borel–Cantelli lemma. This completes the proof. □

Proof of Theorem 2: From (8) it follows that

$$\overline{\lim_{r\to\infty}} \sup_{w\in W} \left|\hat{\lambda}^{(n)}_{r,H_n}(w) - \lambda(w)\right|$$

$$\le \overline{\lim_{r\to\infty}} \sup_{w\in W} \left|\hat{\lambda}^{(n)}_{r,H_n}(w) - \lambda^{(n)}_{H_n}(w)\right|$$

$$+ \sup_{w\in W} \left|\lambda^{(n)}_{H_n}(w) - \lambda(w)\right|. \tag{A4}$$

Denote

$$D_{n,r} = \sup_{w\in W} \left|\hat{\lambda}^{(n)}_{r,H_n}(w) - \lambda^{(n)}_{H_n}(w)\right|, \qquad n, r \in \boldsymbol{N}$$

Here $D_{n,r}$ can be thought to be formed as follows. Let $I_{n,r}$, $n, r \in \boldsymbol{N}$, be independent uniformly distributed random variables that take values in $\{1, \cdots, n\}$ and let $S_{n,r}$, $n, r \in \boldsymbol{N}$, be independent random vectors with density K. The addition of random noise to the n training vectors $Z_1, \cdots, Z_n$ can be modeled by using two independent sequences $(I_{n,r})_r$ and $(S_{n,r})_r$ (cf. Procedure 1). Now $D_{n,r}$ depends on $Z_1, \cdots, Z_n, I_{n,1}, \cdots, I_{n,r}, S_{n,1}, \cdots, S_{n,r}$, and (Z_n) is independent of $(I_{n,r})_r$ and $(S_{n,r})_r$. It can be shown that $D_{n,r}$ is a random variable (cf. proof of Theorem 1). For fixed nonrandom $z_1, \cdots, z_n$ we have by the uniform law of large numbers that

$$\lim_{r\to\infty} D_{n,r} = 0 \tag{A5}$$

with probability 1. Using the independence of (Z_n) from $(I_{n,r})_r$ and $(S_{n,r})_r$, it then follows from Fubini's theorem that (A5) holds with probability 1 also with random training vectors $Z_1, \cdots, Z_n$ (a detailed proof constitutes a straightforward exercise in standard probability theory). Repeating this for $n = 1, 2, \cdots$ and using the countable subadditivity of probability, we have that, except on an event A_1 with $P(A_1) = 0$, (A5) holds for all $n \in \boldsymbol{N}$.

On the other hand, by Theorem 1 we have that

$$\lim_{n\to\infty} \sup_{w\in W} \left|\lambda^{(n)}_{H_n}(w) - \lambda(w)\right| = 0 \tag{A6}$$

except on an event A_2 with $P(A_2) = 0$. Take $A = A_1 \cup A_2$. Then $P(A) = 0$ and, except on the event A, we have from (A4) and (A5) that

$$0 \le \overline{\lim_{r\to\infty}} \sup_{w\in W} \left|\hat{\lambda}^{(n)}_{r,H_n}(w) - \lambda(w)\right| \le \sup_{w\in W} \left|\lambda^{(n)}_{H_n}(w) - \lambda(w)\right|$$

and by (A6) that

$$\lim_{n\to\infty} \left(\overline{\lim_{r\to\infty}} \sup_{w\in W} \left|\hat{\lambda}^{(n)}_{r,H_n}(w) - \lambda(w)\right|\right) = 0.$$

□

Proof of Theorem 3: The argument used is the same as in [19, theorem 1]. Let A be the event on which (11) fails. Then $P(A) = 0$ and we will only consider realizations of training samples (Z_n) and noise data $(I_{n,r}), (S_{n,r})$ outside A (for notation see proof of Theorem 2).

Suppose that (12) fails for minimizing vectors $\left(\hat{w}^{(n)}_{r,H_n}\right)$. Thus, there exists $\epsilon > 0$, a subsequence (n_i) of positive integers and, for each i, a subsequence $(r_{i,j})_j$ of positive integers such that if $v_{i,j} := \hat{w}^{(n_i)}_{r_{i,j},H_{n_i}}$, then

$$d(v_{i,j}, W^0) \ge \epsilon, \qquad i, j \in \boldsymbol{N}. \tag{A7}$$

Denote $\mu_{i,j} = \hat{\lambda}^{(n_i)}_{r_{i,j},H_{n_i}}$, $i, j \in \boldsymbol{N}$. By (11) we can find subsequences (i_k) and (j_k) such that

$$\sup_{w\in W} |\mu_{i_k,j_k}(w) - \lambda(w)| < \frac{1}{k}, \qquad k \in \boldsymbol{N}. \tag{A8}$$

Since W is compact, by passing to a subsequence of $(v_{i_k,j_k})_k$ we may assume that $(v_{i_k,j_k})_k$ converges to a point $w^* \in W$. By (A7), $d(w^*, W^0) \ge \epsilon$, so $w^* \notin W^0$. On the other hand, for an arbitrary $w \in W$, we have

$$\lambda(w^*) - \lambda(w) = [\lambda(w^*) - \lambda(v_{i_k,j_k})] + [\lambda(v_{i_k,j_k})$$
$$- \mu_{i_k,j_k}(v_{i_k,j_k})]$$
$$+ [\mu_{i_k,j_k}(v_{i_k,j_k}) - \mu_{i_k,j_k}(w)] + [\mu_{i_k,j_k}(w)$$
$$- \lambda(w)].$$

Now, suppose $\delta > 0$. By the continuity of λ and $\lim_{k\to\infty} v_{i_k,j_k} = w^*$, we can find an index k_δ such that $|\lambda(w^*) - \lambda(v_{i_k,j_k})| < \delta$ for $k \ge k_\delta$. By (A8),

$$|\lambda(v_{i_k,j_k}) - \mu_{i_k,j_k}(v_{i_k,j_k})| + |\mu_{i_k,j_k}(w) - \lambda(w)| < 2\delta$$

for $k \ge 1/\delta$. By the optimality of $v_{i_k,j_k}, \mu_{i_k,j_k}(v_{i_k,j_k}) - \mu_{i_k,j_k}(w) \le 0$ so that for $k \ge \max\{k_\delta, 1/\delta\}$ we have $\lambda(w^*) - \lambda(w) < 3\delta$. Since $w \in W$ and $\delta > 0$ were arbitrary we must have $w^* \in W^0$, which is a contradiction. □

Proof of Proposition 2: Fix $w \in W$. Then

$$\begin{aligned}\left|\lambda_h^{(n)}(w) - \lambda(w)\right| &\le \int_{R^m}\int_{R^k} \|y - g(x,w)\|^2 \\ &\quad \cdot |f_{n,h}(x,y) - f(x,y)|\,dx\,dy \\ &= \int_{R^m}\int_{R^k} \|y - g(x,w)\|^2 \mid f_{n,h}(x,y) \\ &\quad - f(x,y)\,|^{2/(2+\epsilon)}\,|f_{n,h}(x,y) \\ &\quad \cdot - f(x,y)|^{\epsilon/(2+\epsilon)}dx\,dy \\ &\le \left(\int_{R^m}\int_{R^k} \|y - g(x,w)\|^{2+\epsilon}|f_{n,h}(x,y)\right. \\ &\quad \left. -f(x,y)|\,dx\,dy\right)^{2/(2+\epsilon)}\left(\int_{R^m}\int_{R^k}|f_{n,h} - f|\right)^{\epsilon/(2+\epsilon)}\end{aligned} \tag{A9}$$

by Hölder's inequality. The upper bound C_ϵ for the first factor of the last expression in (A9) can be obtained in a straightforward manner and we skip the details. □

As in Proposition 1, the continuity of $w \mapsto g(x,w)$ was not needed.

Proof of Theorem 4: The measurability of $\sup_{w\in W}|\lambda_{h_n}^{(n)}(w) - \lambda(w)|$ follows as in Theorem 1. By [39, theorem 1] we have $\lim_{n\to\infty}\int_{R^m}\int_{R^k}|f_{n,h_n} - f| = 0$ with probability 1. Thus, by (19) it is enough to show that C_ϵ in (20) stays bounded with probability 1 as n tends to infinity. The only random quantity in C_ϵ is the average $(1/n)\sum_{i=1}^n \|Y_i\|^{2+\epsilon}$ and

$$\begin{aligned}E(\|Y_i\|^{2+\epsilon}) &= \int_{R^m}\int_{R^k}\|y\|^{2+\epsilon}f(x,y)\,dx\,dy \\ &= B_\epsilon(f) < \infty.\end{aligned}$$

Thus, by the strong law of large numbers, $\lim_{n\to\infty}(1/n)\sum_{i=1}^n \|Y_i\|^{2+\epsilon} = B_\epsilon(f) < \infty$ with probability 1. Therefore,

$$\lim_{n\to\infty} C_\epsilon = 4\left[M_W(g)^{2+\epsilon} + \left(\frac{1}{2} + 2^\epsilon\right)B_\epsilon(f)\right]^{2/(2+\epsilon)}$$

with probability 1 and the theorem is proved. □

ACKNOWLEDGMENT

The authors wish in particular to thank one of the referees for pointing out related works.

REFERENCES

[1] S. M. Peeling, R. K. Moore, and M. J. Tomlinson, "The multi-layer perceptron as a tool for speech pattern processing research," in *Proc. IoA Autumn Conf. Speech and Hearing,* 1986.

[2] D. C. Plaut, S. J. Nowlan, and G. E. Hinton, "Experiments on learning by back-propagation," Technical report, Carnegie-Mellon University, 1986.

[3] A. Lindon and J. Kindermann, "Inversion of multilayer nets," in *Proc. Int. Joint Conf. Neural Networks,* 1989, pp. II:425–430.

[4] M. M. Moya and L. D. Hostetler, "One-class generalization in second-order backpropagation networks for image classification," in *Proc. Int. Joint Conf. Neural Networks,* 1990, pp. II:221–224.

[5] J. Sietsma and R. J. F. Dow, "Creating artificial neural networks that generalize," *Neural Networks,* vol. 4, pp. 67–79, 1991.

[6] J. L. Elman and D. Zipser, "Learning the hidden structure of speech," *J. Acoust. Soc. Amer.* vol. 83, no. 4, pp. 1615–1626, 1988.

[7] B. R. Kämmerer and W. A. Küpper, "Experiments for isolated-word recognition with single- and two-layer perceptrons," *Neural Networks,* vol. 3, pp. 693–706, 1990.

[8] T. Matsuoka, H. Hamada, and R. Nakatsu, "Syllable recognition using integrated neural networks," in *Proc. Int. Joint Conf. Neural Networks,* 1989, pp. I:251–258.

[9] E. J. Gardner, N. Stroud, and D. J. Wallace, "Training with noise and the storage of correlated patterns in a neural network model," *J. Phys. A: Math. Gen.,* vol. 22, pp. 2019–2030, 1989.

[10] K. Y. M. Wong and D. Sherrington, "Training noise adaptation in attractor neural networks," *J. Phys. A: Math. Gen.,* vol. 23, pp. L175–L182, 1990.

[11] Ph. Refregier and J.-M. Vignolle, "An improved version of the pseudo-inverse solution for classification and neural networks," *Europhys. Lett.* vol. 10, no. 4, pp. 387–392, Oct. 1989.

[12] G. Györgyi, "Inference of a rule by a neural network with thermal noise," *Phys. Rev. Lett.,* vol. 64, no. 24, pp. 2957–2960, June 1990.

[13] A. Krogh and J. A. Hertz, "Generalization in a linear perceptron in the presence of noise," NORDITA preprint, 1991.

[14] A. Krogh, "Learning with noise in a linear perceptron," NORDITA preprint, 1991.

[15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition,* vol. 1, *Foundations,* Cambridge, MA: ch. 8, MIT Press, 1986.

[16] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Statist.* vol. 33, pp. 1065–1076, 1962.

[17] M. Rosenblatt, "Remarks on some nonparametric estimates of a density function," *Ann. Math. Statist.* vol. 27, pp. 832–837, 1956.

[18] T. Cacoullos, "Estimation of a multivariate density," *Ann. Inst. Statist. Math.* vol. 18, pp. 179–189, 1966.

[19] H. White, "Learning in artificial neural networks: A statistical perspective," *Neural Computation,* vol. 1, pp. 425–464, 1989.

[20] R. I. Jennrich, "Asymptotic properties of non-linear least squares estimators," *Ann. Math. Statist.* vol. 40, no. 2, pp. 633–643, 1969.

[21] B. W. Silverman, *Density Estimation for Statistics and Data Analysis,* Chapman and Hall, 1986.

[22] P. Billingsley, *Probability and Measure,* New York: Wiley, 1979.

[23] L. Devroye, *A Course in Density Estimation.* Birkhäuser Boston, 1987.

[24] L. Devroye and L. Györfi, *Nonparametric Density Estimation: The L_1 View.* New York: Wiley, 1985.

[25] J. Van Ness, "On the dominance of non-parametric Bayes rule discriminant algorithms in high dimensions," *Pattern Recog. J.,* vol. 12, pp. 355–368, 1980.

[26] L. Holmström, J. Klemelä, "An asymptotic upper bound for the expected L_1 error of a multivariate kernel density estimator," *J. Multivariate Analysis,* to be published.

[27] R. P. D. Duin, "On the choice of smoothing parameters for Parzen estimators of probability density functions," *IEEE Trans. Comput.,* vol. C-25, pp. 1175–1179, 1976.

[28] J. D. F. Habbema, J. Hermans, and K. van den Broek, "A stepwise discriminant analysis program using density estimation," in *COMPSTAT 1974,* G. Bruckmann, Ed. Vienna:Physica Verlag, pp. 101–110.

[29] Y.-S. Chow, S. Geman, and L.-D. Wu, "Consistent cross-validated density estimation," *Ann. Statist.,* vol. 11, no. 1, pp. 25–38, 1983.

[30] J. S. Marron, "An asymptotically efficient solution to the bandwidth problem of kernel density estimation," *Ann. Statist.,* vol. 13, no. 3, pp. 1011–1023, 1985.

[31] D. F. Specht, "Probabilistic neural networks," *Neural Networks,* vol. 3, no. 1, pp. 109–118, 1990.

[32] P. Koistinen and L. Holmström, "Kernel regression and backpropagation training with noise," in *Advances in Neural Information Processing Systems 4,* J. E. Moody, S. J. Hanson, and R. P. Lippmann, Eds. San Mateo, CA: Morgan Kaufmann, 1992.

[33] J. Van Ness and C. Simpson, "On the effects of dimension in discriminant analysis," *Technometrics,* vol. 18, no. 2, pp. 175–187, 1976.

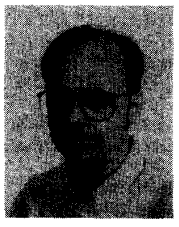
[34] J. Van Ness, "On the effects of dimension in discriminant analysis for unequal covariance populations," *Technometrics,* vol. 21, no. 1, pp. 119–127, 1979.

[35] J. C. Nash, *Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation.* Bristol: Adam Hilger, 1979.

[36] T. Kohonen, G. Barna, and R. Chrisley, "Statistical pattern recognition with neural networks: Benchmarking studies," *Proc. IEEE Int. Conf. Neural Networks* (San Diego), pp. I:61–68, 1988.

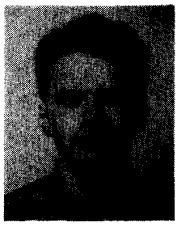
[37] P. A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach.* Englewood Cliffs, NJ: Prentice-Hall, 1982.

[38] D. E. Knuth, *The Art of Computer Programming,* vol. 2, *Seminumerical Algorithms,* 2nd ed. Reading, MA: Addision-Wesley, 1981.
[39] L. Devroye, "The equivalence of weak, strong and complete convergence in L_1 for kernel density estimates," *Ann. Statist.,* vol. 11, no. 3, pp. 896–904, 1983.



Lasse Holmström (M'88) received the B.S., M.S., and Ph.L. degrees in mathematics from the University of Helsinki, Helsinki, Finland, in 1974, 1975, and 1977, respectively. He received the Ph.D. degree in mathematics from Clarkson College of Technology, Potsdam, NY, in 1980.

His research interests have included functional analysis, geometric modeling, and computer-aided design. His current interests include neural networks, pattern recognition, and nonparametric density estimation. He is a Senior Fellow of the Academy of Finland and leads a neural network research group in the Rolf Nevanlinna Institute at the University of Helsinki.



Petri Koistinen received the M.Sc. degree in computer science from the Helsinki University of Technology in 1988. He is currently a research assistant in the Rolf Nevanlinna Institute at the University of Helsinki and is pursuing the Ph.D. degree. His research interests include neural networks and their application to pattern recognition and computer vision problems.