Strong Universal Consistency of Neural Network Classifiers

András Faragó and Gábor Lugosi

Abstract— **In statistical pattern recognition a classifier is called *universally consistent* if its error probability converges to the Bayes-risk as the size of the training data grows, for *all possible distributions* of the random variable pair of the observation vector and its class. It is proven that if a one layered neural network with properly chosen number of nodes is trained to minimize the empirical risk on the training data, then it results in a universally consistent classifier. It is shown that the exponent in the rate of convergence does not depend on the dimension if certain smoothness conditions on the distribution are satisfied. That is, this class of universally consistent classifiers does not suffer from the "curse of dimensionality." A training algorithm is also presented that finds the optimal set of parameters in polynomial time if the number of nodes and the space dimension is fixed and the amount of training data grows.**

Index Terms—**Pattern recognition, neural networks, nonparametric classification, consistency, training algorithms**

I. Introduction

THE PATTERN classification problem can be formulated as follows: Let the random variable pair (X,Y) take its values from $\mathcal{R}^d \times \{0,1\}$. $X \in \mathcal{R}^d$ is called the *observation (or feature) vector*, while $Y \in \{0,1\}$ is its *class*. Observing X one wants to guess the value of Y by a *classification rule* $g : \mathcal{R}^d \to \{0,1\}$ such that the *error probability* $\Pr\{g(X) \neq Y\}$ be small. The best possible classification rule is given by

$$g^*(x) = \begin{cases} 0, & \text{if } P_0(x) \geq 1/2, \\ 1, & \text{otherwise,} \end{cases}$$

where $P_0(x) = \Pr\{Y = 0 | X = x\}$ is the *a posteriori probability* of class 0. The minimal error probability $L^* = \Pr\{g^*(X) \neq Y\}$ is called the *Bayes-risk*. In practic, the a posteriori probabilities are rarely known, instead, a *training sequence*

$$\xi_n = ((X_1,Y_1),(X_2,Y_2),\cdots,(X_n,Y_n)) \tag{1}$$

is available, where $(X,Y),(X_1,Y_1),\cdots,(X_n,Y_n)$ are independent, identically distributed (i.i.d.) random variable pairs. Now, one can estimate Y by $g_n(X) = g_n(X,\xi_n)$, a measurable function of the observation and the training sequence. The *error probability* of g_n is denoted by

$$L(g_n) = \Pr\{g_n(X) \neq Y | \xi_n\},$$

Manuscript received March 18, 1992; revised November 11, 1992.

A. Faragó is with the Department of Telecommunication and Telematics, Technical University of Budapest, 1521 Stoczek u. 2, Budapest, Hungary.

G. Lugosi is with the Department of Mathematics, Faculty of Electrical Engineering, Technical University of Budapest, 1521 Stoczek u. 2, Budapest, Hungary.

IEEE Log Number 9209598.

and the *average error probability* by

$$r(g_n) = E(L(g_n)) = \Pr\{g_n(X) \neq Y\}.$$

A sequence of classification rules is said to be *(weakly) universally consistent* if

$$\lim_{n\to\infty} r(g_n) = L^* \tag{2}$$

regardless of the distribution of (X,Y). A sequence of rules is *strongly universally consistent* if

$$\lim_{n\to\infty} L(g_n) = L^* \quad \text{with probability one,} \tag{3}$$

for any distribution of (X,Y). Obviously, strong universal consistency implies weak universal consistency. Several classification rules have been proved to be universally consistent, such as k-nearest neighbor rules (Stone [24]), kernel based rules (Devroye, Wagner [16] and Spiegelman, Sacks [23]), or partitioning rules (Gordon and Olshen [17]). Strong universal consistency has also been shown for several classification rules, such as for k-nearest neighbor rules by Devroye [10], for kernel rules by Devroye and Krzyżak [15], and for histogram rules by Devroye and Györfi [13].

A classification rule realized by a *feedforward neural network with one (hidden) layer* can be expressed as

$$g(x,\theta_k) = \begin{cases} 0, & \text{if } f(x,\theta_k) \geq 1/2, \\ 1, & \text{otherwise,} \end{cases} \tag{4}$$

where

$$f(x,\theta_k) = \sum_{i=1}^{k} c_i \sigma(a_i^T x + b_i) + c_0. \tag{5}$$

Here k is the number of *nodes (hidden neurons)*, the *sigmoid* σ is a monotone increasing function with $\lim_{x\to-\infty} \sigma(x) = -1$ and $\lim_{x\to\infty} \sigma(x) = 1$, and $\theta_k = (a_1, \cdots, a_k, b_1, \cdots, b_k, c_0, c_1, \cdots, c_k)$ is the *parameter vector* of the neural network ($a_i \in \mathcal{R}^d$, $c_0, b_i, c_i \in \mathcal{R}, i = 1,\cdots,k$). ($a^T b$ denotes the scalar product of the vectors a and b.) We assume that the sigmoid is the step function

$$\sigma(x) = \begin{cases} 1, & \text{if } x \geq 0, \\ -1, & \text{if } x < 0. \end{cases}$$

Some of the arguments used in this paper do not immediately extend to general sigmoids. In order to generalize our results, the methods used in Haussler's [18] recent paper may be useful, but for the sake of simplicity we do not pursue this question here. Our goal is to choose the number of nodes k and set the parameters such that the error probability

$\Pr\{g(X,\theta_k) \neq Y\}$ be small. Our strategy is to minimize the *empirical error*, in other words, we choose a parameter vector $\theta^*_{k,n}$ for which the corresponding classification rule $g^*_{k,n}(x) = g(x,\theta^*_{k,n})$ commits the minimum number of errors on the training sequence:

$$\hat{L}(g^*_{k,n}) = \min_{\theta_k} \hat{L}(g(\cdot,\theta_k)), \tag{6}$$

where

$$\hat{L}(g(\cdot,\theta_k)) = \frac{1}{n}\sum_{j=1}^{n} I_{\{g(X_j,\theta_k)\neq Y_j\}}$$

is the empirical error probability of the classification rule $g(x,\theta_k)$. (I_A denotes the indicator of an event A).

In Section II, we show that such parameter selection has very good properties, in particular, the selected classification rule $g^*_{k,n}$ is strongly universally consistent if $k \to \infty$ and $k\log(n)/n \to 0$ as $n \to \infty$. We note here that the results of White [27] imply (2) for a large class of distributions with a different training strategy. In Section III, we point out the remarkable property that if the distribution satisfies some regularity conditions, then with a properly chosen number of nodes the rate of convergence of the error probability to the Bayes-risk is independent of the dimension. Finally, in Section IV, we deal with the problem of training algorithms that find the optimal parameters.

II. Consistency

We start with a consequence of the important denseness theorem obtained independently by Cybenko [9] and Hornik, Stinchcombe, and White [19].

Lemma 1:

$$\lim_{k\to\infty} \inf_{\theta_k} L(g(\cdot,\theta_k)) - L^* = 0,$$

where $L(g(\cdot,\theta_k)) = \Pr\{g(X,\theta_k) \neq Y\}$. That is, the error probability of the neural network classifier with the best possible parameters gets arbitrarily close to the Bayes-risk, if the number of neurons is large enough.

Proof: It is well known (see e.g., Devroye and Györfi [14]) that for any parameter vector θ_k

$$L(g(\cdot,\theta_k)) - L^* \leq 2E|f(X,\theta_k) - P_0(X)|. \tag{7}$$

By the famous theorem of Cybenko [9] and Hornik, Stinchcombe, and White [19], any continuous function with bounded support in $\mathcal{R}^d$ can be arbitrarily approximated in supremum norm by functions of form (5) if k is large enough. On the other hand, the set of continuous functions with compact support is dense in $L_1(\mu)$, that is, the set of functions integrable with respect to μ, the probability measure induced by the random variable X. But being bounded by one, $P_0 \in L_1(\mu)$ holds, therefore, any a posteriori probability function can be arbitrarily approximated in $L_1(\mu)$ by functions of form $f(x,\theta_k)$:

$$\lim_{k\to\infty} \inf_{\theta_k} E|f(X,\theta_k) - P_0(X)| = 0. \tag{8}$$

Thus, (8) and (7) imply the lemma. □

By Lemma 1 the *approximation error* tends to zero as the number of neurons goes to infinity. In practical implementations the number of neurons is often fixed (e.g., can not be increased as the size of the training data grows). In this case the message of Lemma 1 can be interpreted such that the approximation error is not too bad provided that k is large enough, even though we can not get universal consistency in this case. The next lemma deals with the *estimation error*, that is, with the difference of the error probability of the chosen rule $g^*_{k,n}$ and the one with optimal parameters.

Lemma 2: For every $\epsilon > 0$ and $n \geq 2/\epsilon^2$

$$\Pr\{L(g^*_{k,n}) - \inf_{\theta_k} L(g(\cdot,\theta_k)) \geq \epsilon\} \leq 2^{k+3}(2n+1)^{(k+1)d}e^{-n\epsilon^2/32}.$$

Proof: Recall Devroye's [12] simple, but extremely useful observation

$$L(g^*_{k,n}) - \inf_{\theta_k} L(g(\cdot,\theta_k)) \leq 2\sup_{\theta_k} |L(g(\cdot,\theta_k)) - \hat{L}(g(\cdot,\theta_k))|. \tag{9}$$

This observation enables us to use the Vapnik–Chervonenkis inequality (Vapnik, Chervonenkis [26], Devroye [12]): For every $n \geq 2/\epsilon^2$

$$\Pr\{\sup_{\theta_k} |L(g(\cdot,\theta_k)) - \hat{L}(g(\cdot,\theta_k))| \geq \epsilon\} \leq 4S(2n,k)e^{-n\epsilon^2/8}, \tag{10}$$

where $S(n,k)$ is the *shatter coefficient*, the maximal number of different ways the classifiers of form (4) with k neurons can classify n points in $\mathcal{R}^d$. Formally:

$$S(n,k) = \max_{x_1,\cdots,x_n\in\mathcal{R}^d} N_k((x_1),\cdots,(x_n)),$$

where $N_k((x_1),\cdots,(x_n))$ is the number of different binary $n-$ vectors

$$(g_{\theta_k}(x_1),\cdots,g_{\theta_k}(x_n)),$$

if θ_k is allowed to take all possible values.

It follows from the results of Baum and Haussler ([5], Theorem 1) that

$$S(n,k) \leq S^k_{n,d}S_{n,k},$$

where $S_{n,r}$ is the number of different ways n points can be classified by an r-input neuron. In our case the family of classification rules that can be realized by a neuron is just the family of linear classifiers. By a well known result of Cover [8] (see also Vapnik and Chervonenkis [26]) the number of different ways n points in $\mathcal{R}^r$ can be dichotomized by a hyperplane is at most $n^r + 1$, from which $S_{n,r} \leq 2(n^r+1)$ follows. Therefore,

$$\begin{aligned} S(n,k) &\leq (2n^d+2)^k(2n^k+2) \\ &\leq \left(2(n+1)^d\right)^k 2(n+1)^k \leq 2^{k+1}(n+1)^{d(k+1)}. \end{aligned} \tag{11}$$

The statement of the lemma follows from (9)–(11). □

Now we are prepared to prove the main result of the section.

Theorem 1: If the number of nodes k is chosen to satisfy

$$k \to \infty \tag{12}$$

and

$$\frac{k \log(n)}{n} \to 0 \tag{13}$$

as $n \to \infty$, then

$$\lim_{n\to\infty} L(g^*_{k,n}) = L^* \quad \text{with probability one,}$$

regardless of the distribution, that is, the sequence of rules $\{g^*_{k,n}\}$ is strongly universally consistent.

Proof: We can write

$$L(g^*_{k,n}) - L^* = \left(L(g^*_{k,n}) - \inf_{\theta_k} L(g(\cdot,\theta_k))\right) + \left(\inf_{\theta_k} L(g(\cdot,\theta_k)) - L^*\right).$$

Clearly, the second term on the right hand side (the approximation error) tends to zero by Lemma 1 and (12). For the first term (the estimation error) we can write by Lemma 2 that for every $\epsilon > 0$ and $n \geq 2/\epsilon^2$

$$\Pr\{L(g^*_{k,n}) - \inf_{\theta_k} L(g(\cdot,\theta_k)) \geq \epsilon\} \leq e^{-n\epsilon^2/32 + (k+3)\log 2 + (k+1)d\log(2n+1)}.$$

But it follows from (13) that $(k+3)\log 2 + (k+1)d\log(2n+1) \leq n\epsilon^2/64$ if $n > n_0(\epsilon)$ for some $n_0(\epsilon)$, therefore, for every $\epsilon > 0$

$$\sum_{n=1}^{\infty} \Pr\{L(g^*_{k,n}) - \inf_{\theta_k} L(g(\cdot,\theta_k)) \geq \epsilon\} < \infty,$$

thus, by the Borel–Cantelli lemma

$$\lim_{n\to\infty} \left(L(g^*_{k,n}) - \inf_{\theta_k} L(g(\cdot,\theta_k))\right) = 0$$

with probability one, and the proof is completed. □

III. Rate of Convergence

In the previous section, we saw that the error probability of the classification rule defined by (6) always converges to the Bayes risk if the number of neurons k is chosen such that $k \to \infty$ and $k\log(n)/n \to 0$. Two natural questions arise immediately: At what rate does $L(g^*_{k,n})$ go to zero, and what is the best choice of k? In this section, we address these problems. Devroye [11] demonstrated, that it is not possible to obtain upper bounds on the rate of convergence without making some assumptions on the distribution. Thus, we will have to impose some regularity conditions on the *a posteriori* probabilities. Surprisingly, however, we have a distribution-free upper bound for the rate of convergence of the estimation error:

Lemma 3:

$$r(g^*_{k,n}) - \inf_{\theta_k} L(g(\cdot,\theta_k)) \leq 4\sqrt{\frac{(k+3)\log 2 + (k+1)d\log(2n+1)}{n}},$$

where $r(g^*_{k,n}) = \Pr\{g^*_{k,n}(X) \neq Y\}$ is the average error probability of $g^*_{k,n}$.

Proof: The lemma can be derived from Lemma 2 and Devroye's Lemma 1 in [12]. We omit the details.

Lemma 3 states that the estimation error is always $O\left(\sqrt{kd\log(n)/n}\right)$. Therefore, to obtain some result for the rate of convergence, it is enough to bound the approximation error

$$\inf_{\theta_k} L(g(\cdot,\theta_k)) - L^*.$$

Such a result is due to Barron [1], which implies the following.

Lemma 4: Assume that there is a compact set $A \subset \mathcal{R}^d$ such that $\Pr\{X \in A\} = 1$ and the Fourier transform $\tilde{P}_0(\omega)$ of $P_0(x)$ satisfies $\int_{\mathcal{R}^d} |\omega||\tilde{P}_0(\omega)|d\omega < \infty$. Then,

$$\inf_{\theta_k} E(f(X,\theta_k) - P_0(X))^2 \leq \frac{c}{k},$$

where c is a constant depending on the distribution.

Remark: The previous condition on the Fourier transform could be informally explained as a weakened form of band limitation. For extensive discussion on the properties of functions satisfying the condition with many examples, we refer to Barron [1].

From Lemma 4 by the Cauchy–Schwarz inequality, we get

$$\inf_{\theta_k} E|f(X,\theta_k) - P_0(X)| \leq \sqrt{\frac{c}{k}} \tag{14}$$

Bringing Lemma 3, (14), and (7) together, we have the following.

Theorem 2: If the conditions of Lemma 4 are satisfied, then

$$r(g^*_{k,n}) - L^* = O\left(\sqrt{\frac{kd\log(n)}{n}} + \frac{1}{\sqrt{k}}\right).$$

If, further, the number of neurons is chosen to be $k = O(\sqrt{n/d\log(n)})$, then

$$r(g^*_{k,n}) - L^* = O\left(\left(\frac{d\log(n)}{n}\right)^{1/4}\right).$$

Theorem 2 points out a remarkable property of neural network classifiers, since for this smooth class of a posteriori probability functions, the *rate of convergence does not depend on the dimension,* in the sense that the exponent is constant. For most universally consistent nonparametric techniques, the rate of convergence is of $O(n^{-1/(\alpha+\beta d)})$ type $(\alpha, \beta > 0)$, where β depends on the smoothness of the functions in the class. Clearly, if this number does not depend on the dimension, then the size of the training sample necessary to achieve a certain performance increases *exponentially* with the dimension. This phenomenon is often called the *curse of dimensionality*. Stone [25] showed for regression estimation that the optimal rate of convergence is of this type, that is, no estimator can yield

faster rate of convergence. The connection between Stone's lower bound and Barron's condition is explained in Barron [4], where lower bounds are also given for the approximation rate. Similar results to Theorem 2 were obtained for function estimation by Barron [2], [3], using a different minimization strategy.

IV. The Problem of Training

The key issue in applications is: how to choose the parameter vector θ_k in (4), given only a realization of the training sequence (1). Unfortunately, the problem of optimal neural net training is computationally difficult. Blum and Rivest [6], Judd [20], Lin and Vitter [21] and others examined many instances of neural net training from the viewpoint of algorithmic complexity. It turns out that most interesting cases belong to the class of NP-hard problems, which leaves very little chance of producing efficient algorithms. In other words, it is conjectured that there exists no algorithm that could find an optimal set of parameters with running time bounded by a polynomial in the size of the input.

A possible compromise is to apply some iterative optimization technique that yields only a local optimum. The most popular method of this type is the Back Propagation Algorithm of Rumelhart *et al.* [22]. Unfortunately, nothing is known about how far the produced local optimum is from the global one, therefore, it offers little chance to prove any bound on the error probability. Indeed, it can happen, even in very simple cases, that back propagation gives definitely poor results, as it is shown by Brady et al. [7].

Here, we present a training algorithm for our model, which finds the globally optimal set of parameters, that is, a parameter setting which minimizes the empirical error. Though we cannot guarantee a polynomial running time as a function of all parameters, we can at least ensure polynomial time complexity in the length of the training sequence, which is still nontrivial. In other words, if the space dimension and the number of neurons are fixed and the training set grows, the running time of the algorithm does not grow exponentially, it is bounded by a polynomial in n.

We present the algorithm in two parts. First a *splitting algorithm* is described, which generates *all possible* splittings of n points in $\mathcal{R}^d$ into two subsets by hyperplanes. The number of different hyperplane-splittings is at most $n^d + 1$ [8], this fact has been already used in the proof of Lemma 2. It is not immediate, however, how to *generate* them in polynomial time, as one cannot check all the 2^n subsets. Using the splitting procedure as a subroutine, the *training algorithm* finds the parameters which (globally) minimize the empirical error.

The input of the splitting algorithm is a set $A = \{u_1, \cdots, u_n\}$ of points from $\mathcal{R}^d$. The output is a set S of at most $n^d + 1$ d-dimensional vectors, such that each $b \in S$ defines a different splitting of A into two subsets by the hyperplane given by the equation $b^T x = 2$. The obtained partitions of the form $\{A', A''\}$ are collected in a set $\mathcal{P}$. For notational convenience, a $d^{\leq}$-tuple means an l-tuple with $1 \leq l \leq d$.

Splitting Algorithm:

Step 1) $i := 1$; $S := \emptyset$; $\mathcal{P} := \emptyset$;

Step 2) Generate the lexicographically ith $d^{\leq}$-tuple $(r_1, \cdots, r_l)$ of numbers from $\{1, \cdots, n\}$ with $r_1 < \cdots < r_l$;

Step 3) $j := 1$;

Step 4) Generate the lexicographically jth l-tuple $(\alpha_1, \cdots, \alpha_l)$, $\alpha_1, \cdots, \alpha_l \in \{1, -1\}$;

Step 5) Find a solution $z_0 \in \mathcal{R}^d$ to the system

$$u_{r_\nu} z = 2 + \alpha_\nu, \qquad \nu = 1, \cdots, l$$

of linear equations. If no solution exists GO TO Step 8);

Step 6) Split A into two subsets A' and A" by the hyperplane given by $z_0^T x = 2$;

Step 7) IF $\{A', A''\} \notin \mathcal{P}$ THEN $S := S \cup \{z_0\}$; $\mathcal{P} := \mathcal{P} \cup \{\{A', A''\}\}$;

Step 8) IF $j < 2^l$ THEN $j := j + 1$; GO TO Step 4);

Step 9) IF $i < \sum_{q=1}^{d} \binom{n}{q}$ THEN $i := i + 1$; GO TO Step 3) ELSE STOP.

Lemma 5: The Splitting Algorithm generates all possible splittings of A into two subsets by a hyperplane in $O(n^{2d+2})$ time.

Proof: The algorithm solves $\sum_{i=1}^{d} 2^i \binom{n}{i}$ systems of linear equations, each in d variables and of $\leq d$ equations. If d is constant, this takes $O(n^{d+1})$ time. For each equation we have to check whether its solution generates a new splitting or not. That means, we have to make a comparison with every splitting collected in $\mathcal{P}$. As we know [8] that there are at most $O(n^d)$ different splittings, this yields $O(n^d)$ comparisons, each taking $O(n)$ time, altogether $O(n^{d+1})$. Since the complexities of all the auxiliary steps are "swallowed" by these terms, therefore, the overall time complexity is $O(n^{2d+2})$.

Now we have to show that all hyperplane-splittings are generated. Such a splitting can always be given by a hyperplane which contains neither the origin, nor the points in A. Let H be such a hyperplane. Its equation can be formulated as $b^T x = 2$ with some $b \in \mathcal{R}^d$. (Of course, any nonzero value on the right hand side would do). Let H_1, H_2 be the two open halfspaces on the two sides of H. For every $u_i \in A$,we have either $b^T u_i > 2$ or $bu_i < 2$, depending on which halfspace contains u_i. Clearly, such a b can be found by replacing > 2 and < 2 by ≥ 3 and ≤ 1, respectively, and solving the arising linear system of inequalities for b.

Define $\gamma_i, i = 1, \cdots, n$, by

$$\gamma_i = \begin{cases} 1, & \text{if } u_i \in H_1, \\ -1, & \text{if } u_i \in H_2. \end{cases}$$

So we have that given a partition $A \cap H_1, A \cap H_2$ of A, an appropriate b can be found by taking any solution to the linear system of inequalities

$$\gamma_i u_i z \geq \gamma_i(2 + \gamma_i), \qquad i = 1, \cdots, n. \tag{15}$$

Now the crucial point follows from the following considerations. A system of linear inequalities defines a polyhedron. It is known from the theory of polyhedra that every minimal facet (a facet that does not contain another one) of a polyhedron is an affine subspace, which may be a single point in the case of a vertex. (This also means that the only minimal facets

of bounded polyhedra are vertices.) Now, a minimal facet is determined by a subset of inequalities that are satisfied with equality for the points of the facet and the facet is exactly the solution set of this system of linear *equations*. Moreover, the system can be chosen to contain at most d equations. Thus, it is possible to select l inequalities for some $l \leq d$ such that they can be satisfied with equality if and only if the polyhedron is nonempty and then every solution of the arising system of linear equations also satisfies the whole system of inequalities.

So we see, that if we take a solution, if any, to all possible systems of linear equations of the form

$$u_{i_\nu} z = 2 + \alpha_\nu, \quad i_\nu \in \{1, \cdots, n\}, \quad \nu = 1, \cdots, l, l \leq d, \tag{16}$$

where $\alpha_1, \cdots, \alpha_\nu \in \{1, -1\}$, then this set of solutions includes a solution to (15) for all possible settings of the λ_i for which (15) is solvable at all. This is what the algorithm does, making use of the fact that the number of possible equations of the form (16) is polynomially bounded in n with d fixed. Thus, all splittings are generated in this way. □

Now we can describe the *training algorithm*. Its input consists of the parameters n, k, d, and a realization $(x_1, y_1), \cdots, (x_n, y_n)$ of the training sequence. The output is the parameter vector $\theta^*_{k,n} = (a_1, \cdots, a_k, b_1, \cdots, b_k, c_0, c_1, \cdots, c_k)$ which (globally) minimizes the empirical error (6). Note that, with appropriate scaling, the values of $b_1, \cdots, b_k, c_0$ can be set to be arbitrary nonzero numbers, we fix $b_1 = \cdots = b_k = -1, c_0 = -2$ here.

Training Algorithm:

Step 1) Run the Splitting Algorithm with input $A = \{x_1, \cdots, x_n\}$ and let S be its output;

Step 2) $i := 1; \quad E := \infty$;

Step 3) Generate the lexicographically ith k-tuple $(a_1, \cdots, a_k)$ from S with repetition allowed among the a_i's;

Step 4) Compute the vectors $v_j \in \mathcal{R}^k j = 1, \cdots, n$ such that the lth component v_{jl} of v_j is $v_{jl} = \sigma(a_l^T x_j - 1); l = 1, \cdots, k$;

Step 5) Run the Splitting Algorithm with input $A = \{v_1, \cdots, v_n\}$ and let S_1 be its output;

Step 6) For all $c = (c_1, \cdots, c_k) \in S_1$ compute the empirical error with the parameters set to $\theta = (a_1, \cdots, a_k, -1, \cdots, -1, -2, c_1, \cdots, c_k)$;

Step 7) Let $\tilde{\theta}$ be the value of θ for which the empirical error is the smallest in Step 6); let E_0 be the value of this error;

Step 8) IF $E_0 < E$ THEN $\theta^*_{k,n} := \tilde{\theta}$;

Step 9) IF $i < \binom{|S|}{k}$ THEN $i := i + 1$; GO TO Step 3) ELSE STOP.

Theorem 3: If the number of neurons and the dimension of space are fixed, then the Training Algorithm runs in polynomial time (in the size of the training sequence) and finds a parameter vector which minimizes the empirical error.

Proof: If k, d are fixed and n is varying then the Splitting Algorithm is called only polynomially many times and, clearly, the auxiliary computations can be executed also in polynomial time. This, together with Lemma 5, implies the polynomial time complexity. So it remains to prove the optimality.

Let us call a partition of the training points a *k-splitting* if it can be generated by k hyperplanes. Now observe that all k-splittings of the training points are probed by the algorithm. So it is enough to show that the parameters c_i of the linear combination are chosen optimally. That means, we have to prove that if the other parameters are fixed, then it is enough to restrict ourselves in Step 6) to those c vectors which are in S_1. To see this, let us consider Step 4). It follows from the definition of the vectors v_j, that $f(x_j, \theta) = c^T v_j + c_0 = c^T v_j - 2$. So the possible values of c which give different decisions on the training set when the other parameters are fixed, can be found by looking for vectors which produce different splittings of the set $\{v_1, \cdots, v_n\}$ by a hyperplane of equation $c^T x = 2$. The algorithm does exactly this, by calling the Splitting Algorithm with input $V = \{v_1, \cdots, v_n\}$ in Step 5). □

V. Conclusion

We proved that a one-layered neural network with properly chosen number of nodes provides a universally consistent classifier, if the parameters are chosen to minimize the empirical error on the training data. It is also shown that if certain smoothness conditions on the distribution are satisfied, then the exponent in the rate of convergence is independent of the dimension. In other words, this classifier does not suffer from the "curse of dimensionality."

A training algorithm is also given, which finds the optimal parameters. If the space dimension and the number of neurons are fixed, then the algorithm runs in polynomial time in the size of the training set.

Finally we mention an important open problem. It is not difficult to see that universal consistency still can be maintained if the training algorithm finds merely a suboptimal solution with, say, $O(1/\sqrt{k})$ relative error. Is it possible to construct a substantially faster algorithm if we are satisfied with such a suboptimal set of parameters? In general, it would be nice to find some provable trade-off between the accuracy and time complexity of training algorithms. In particular, based on Barron's [1] results of iterative approximation we have the following conjecture:

Conjecture: There exists an algorithm of complexity polynomial in n and k that provides a universally consistent classifier.

References

[1] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," to appear in *IEEE Trans. on Information Theory*, 1993.

[2] ——, "Complexity regularization with application to artificial neural networks," in *Proceedings of the NATO ASI on Nonparametric Functional Estimation*, G. Roussas, Ed. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1991.

[3] ——, "Approximation and estimation errors for artificial neural networks," in *Computational Learning Theory: Proceedings of the Fourth Annual Workshop*. San Fransisco, CA: Morgan Kaufman, 1991.

[4] ——, "Neural net approximation," in *Yale Workshop on Adaptive and Learning Syst.*, 1992.

[5] E. B. Baum and D. Haussler, "What size net gives valid generalization?" *Neural Computat.*, vol. 1, pp. 151–160, 1989.

[6] A. Blum and R. L. Rivest, "Training a 3-node neural network is NP-complete," in *Proc. 1st ACM Workshop on Computat. Learning Theory*, Cambridge, MA, 1988, pp. 9–18.
[7] M. L. Brady, R. Raghavan, and J. Slawny, "Back propagation fails to separate where perceptrons succeed," *IEEE Trans. Circuits and Syst.*, vol. 36, pp. 665–674, 1989.
[8] T. M. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," *IEEE Trans. Electron. Comput.* vol. 14, pp. 326–334, 1965.
[9] G. Cybenko, "Approximations by superpositions of sigmoidal functions," *Math. Contr., Signals, Syst.*, vol. 2, pp. 303–314, 1989.
[10] L. Devroye, "Necessary and sufficient conditions for the almost everywhere convergence of nearest neighbor regression function estimates," *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, vol. 61, pp. 467–481, 1982.
[11] ——, "Any discrimination rule can have an arbitrarily bad probability of error for finite sample size," *IEEE Trans. Pattern Anal Machine Intell.*, vol. PAMI-4, no. 2, Mar. 1982.
[12] ——, "Automatic pattern recognition: A study of the probability of error," *IEEE Trans. Pattern Anal. Machine Intell.* vol. 10, pp. 530–543, 1988.
[13] L. Devroye, L. Györfi, "Distribution-free exponential bound on the L_1 error of partitioning estimates of a regression function," in *Proc. of the Fourth Pannónian Symposium on Mathematical Statistics*, F. Konecny, J. Mogyoródi, and W. Wertz, Eds. Budapest, Hungary: Akadémiai Kiadó, 1983, pp. 67–76.
[14] L. Devroye and L. Györfi, *Nonparametric Density Estimation: The L_1-View* New York: Wiley, 1985.
[15] L. Devroye and A. Krzyżak, "An equivalence theorem for L_1 convergence of the kernel regression estimate," *J. Statistical Planning and Inference*, vol. 23, pp. 71–82, 1989.
[16] L. Devroye and T. J. Wagner, "Distribution-free consistency results in nonparametric discrimination and regression function estimation," *Ann. Statist.*, vol. 8, pp. 231–239, 1980.
[17] L. Gordon and R. A. Olshen, "Asymptotically efficient solutions to the classification problem," *Ann. Statist.*, vol. 6, pp. 515–533, 1978.
[18] D. Haussler, "Decision theoretic generalizations of the PAC model for neural net and other learning applications," *Inform. Computat.*, vol. 100, pp. 78–150, 1992.
[19] K. Hornik, M. Stinchcombe, and H. White, "Multi-layer feedforward networks are universal approximators," *Neural Net.*, vol. 2, pp. 359–366, 1989.
[20] J. S. Judd, "On the complexity of loading shallow neural networks," *J. Complexity*, vol. 4, pp. 177–192, 1988.
[21] J. H. Lin and J. S. Vitter, "Complexity results on learning by neural nets," *Machine Learning*, vol. 6, pp. 211–230, 1991.
[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing*, vol. I, D .E. Rumelhart, J. L. McCelland, and the PDP Research Group, Eds. Cambridge, MA: M.I.T. Press, 1986.
[23] C. Spiegelman and J. Sacks, "Consistent window estimation in nonparametric regression," *Ann. Statist.*, vol. 8, pp. 240–246, 1980.
[24] C. J. Stone, "Consistent nonparametric regression," *Ann. Statist.*, vol. 5, pp. 595–645, 1977.
[25] ——, "Optimal global rates of convergence for nonparametric estimators," *Ann. Statist.*, vol. 10, pp. 1040–1053, 1982.
[26] V. N. Vapnik and A. Ya. Chervonenkis, "On the uniform convergence of relative frequencies of events to their probabilities," *Theory of Probab. and its Applicat.*, vol. 16, pp. 264–280, 1971.
[27] H. White, "Connectionist nonparametric regression: multilayer feedforward networks can learn arbitrary mappings," *Neural Net.*, vol. 3, pp. 535–549, 1990.