CONF-9505145--2

Finite-Sample Based Learning Algorithms for Feedforward Networks †

Nageswara S. V. Rao, V. Protopopescu, R. C. Mann, and E. M. Oblow
Center for Engineering Systems Advanced Research
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831-6364

S. Sitharama Iyengar
Department of Computer Science
Lousiana State University
Baton Rouge, LA 70803

Submitted to: *1995 ACM Symposium on Theory of Computing*, May 20-June 1, 1995, Las Vegas, Nevada.

†Research sponsored by the Engineering Research Program of the Office of Basic Energy Sciences, of the U.S. Department of Energy, under Contract No. DE-AC05-840R21400 with Martin Marietta Energy Systems, Inc.

Finite-Sample Based Learning Algorithms for Feedforward Networks [1]

Nageswara S.V. Rao [2], V. Protopopescu, R. C. Mann, E. M. Oblow
Center for Engineering Systems Advanced Research
P. O. Box 2008
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6364
{raons,protopopesva,mannrc,oblowem}@ornl.gov

S. Sitharama Iyengar
Department of Computer Science
Louisiana State University
Baton Rouge, LA 70803
iyengar@bit.csc.lsu.edu

Summary

We discuss two classes of convergent algorithms for learning continuous functions (and also regression functions) that are represented by FeedForward Networks (FFN). The first class of algorithms, applicable to networks with unknown weights located only in the output layer, is obtained by utilizing the potential function methods of Aizerman *et al.* [1]. The second class, applicable to general feedforward networks, is obtained by utilizing the classical Robbins-Monro style stochastic approximation methods [22]. Conditions relating the sample sizes to the error bounds are derived for both classes of algorithms using martingale-type inequalities. For concreteness, the discussion is presented in terms of neural networks, but the results are applicable to general feedforward networks, in particular to wavelet networks. The algorithms can also be directly applied to concept learning problems. A main distinguishing feature of the this work is that the sample sizes are based on explicit algorithms rather than information-based methods.

1 Introduction

The problem of learning a function or a set from a finite set of examples has been the focus of considerable research in areas such as pattern recognition, machine learning, neural networks, etc. Recent density results indicate that finite-sized networks can approximate continuous or indicator functions within a specified precision. These results enable us to formulate the learning problem as one of estimating finite dimensional vectors (that typically represent connection weights of a network). The learning methods that compute the connection weights of a required network based on a finite sample have been extensively studied recently both analytically and also by using implementation/experimentation. The recent renewal of interest in such methods can be attributed, at least in part, to the success of neural networks in a wide variety of applications. Typically, the performance of such methods depends on (a) the form of network employed, and (b) the learning algorithm that computes

[1]Research sponsored by the Engineering Research Program of the Office of Basic Energy Sciences, of the U.S. Department of Energy, under Contract No. DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.

[2]Address all correspondence to the first author.

the parameters of the network. Our focus is on the performance of the algorithm when applied to *finite-sized samples.* For most existing algorithms such results are not available, yet they are needed and useful in practical applications, where the samples are always finite. We illustrate the application of two well-known methods, the potential function methods (Aizerman *et al.* [1]) and stochastic approximation (Polyak [17], Benveniste *et al.* [3]), to obtain function learning algorithms implemented on two basic FFN architectures. The first method is applicable only to FFN with unknown weights located only in the output layer, and the second technique is applicable to general FFN.

A continuous function can be approximated by a finite-sized network of non-polynomial units (with a single hidden layer) within a specified precision (Leshno *et al.* [13], and Mhaksar and Micchelli [15]). When the units are sigmoid functions, the networks are called the *feedforward artificial neural networks* with a single hidden layer (Cybenko [5], Funahashi [9], Barron [2]). These density properties can be utilized for approximating arbitrary continuous mappings or concepts by networks whose parameters (in the form of connection weights) are to be determined by the function being approximated.

For the tasks of learning functions from a finite sample, the utility of the above density results critically depends on the availability of suitable learning (or training) algorithms. There are several algorithms that train the networks of sigmoidal units based on a sample (Werbos [29], van der Smagt [25], Tang and Koehler [24]). The performance of such algorithms is known only to a limited extent, and moreover, in cases where the performance has been analyzed, the results are typically asymptotic.

The aim of this paper is to provide a comprehensive framework for designing learning algorithms based on *finite-sized samples.* Our contributions include the following:

(a) combination of empirical estimation and potential function (or stochastic approximation) methods to obtain finite sample results for function and regression estimation;

(b) application of Lyapunov methods of Polyak [17] to obtain finite sample results for learning algorithms based on stochastic approximation;

(c) combination of density results of FFN with the Hilbert space methods of Revesz [21] to obtain learning algorithms for FFN;

(d) constructive algorithms for solving several classes of Probably and Approximately Correct (PAC) concept learning problems; and

(e) finite sample results for concept and for function learning algorithms based on wavelet networks.

Our sample estimates are based on algorithms which distinguishes them from information based estimates (for example those in Haussler [10]). This is an extended summary of Rao *et al.* [19], which contains the detailed proofs and various extensions and variations.

2 Preliminaries

2.1 Function and Regression Learning Problems

A *training n-sample* of a function $f : [0,1]^d \mapsto \Re$ is given by $(x_1, f(x_1))$, $(x_2, f(x_2))$, $\ldots$, $(x_n, f(x_n))$ where the random variables $x_1, x_2, \ldots, x_n$, $x_i \in [0,1]^d$, are independently and identically generated according to a distribution P_X ($X = [0,1]^d$). The *function learning problem* is to estimate a function $\hat{f} : [0,1]^d \mapsto \Re$, based on the sample, such that $\hat{f}(x)$ "closely" approximates $f(x)$. More precisely, we consider either the expected square error

$$I(\hat{f}) = \int [\hat{f}(x) - f(x)]^2 dP_X \tag{2.1.1a}$$

or the expected absolute error

$$J(\hat{f}) = \int |\hat{f}(x) - f(x))| dP_X \tag{2.1.1b}$$

which is to be minimized over a family of functions $\mathcal{F}$ based on the given n-sample. Let $f_* \in \mathcal{F}$ minimize $I(\hat{f})$ (or $J(\hat{f})$) over all $\hat{f} \in \mathcal{F}$. In general, f_* cannot be computed from (2.1.1a) or (2.1.1b) since the underlying probability distribution is unknown. We obtain conditions under which an approximation $\hat{f}$ to f_* can be computed such that for a sufficiently large sample we have

$$P[I(\hat{f}) - I(f_*) > \epsilon] < \delta \tag{2.1.2a}$$

or

$$P[J(\hat{f}) - J(f_*) > \epsilon] < \delta \tag{2.1.2b}$$

corresponding to (2.1.1a) and (2.1.1b) respectively for arbitrarily specified $\epsilon > 0$ and δ, $0 < \delta < 1$, where $P = P_X^n$ is the product measure on the set of all independently and identically distributed n-samples. Here ϵ and δ are called the *precision* and *confidence* parameters respectively, and the pair (ϵ, δ) is called the *performance pair.*

A more general problem is *regression learning.* Namely, we are given a training sample (x_1, y_1), (x_2, y_2), ..., (x_n, y_n) generated independently and identically according to the probability distribution $P_{X,Y}$ ($X = [0,1]^d, Y = \Re$) such that $f(x)$ is equal to the conditional expectation $E(y|x)$. The problem is to compute an estimate $\hat{f}$ of the regression function f that satisfies the condition (2.1.2a) or (2.1.2b) with $P = P_{X,Y}^n$.

2.2 Approximation by Neural Networks

The general architecture of a multilayer FFN consists of an input layer with d units and output layer with m units, and one or more hidden layers. We shall first consider FFN with a single hidden layer and a single output unit. The hidden unit j has a weight vector $b_j \in \Re^d$ and a threshold $t_j \in \Re$. The output of the jth hidden unit is $\sigma(b_j^T x - t_j)$, where $x = (x^1, x^2, \ldots x^d)$ is the input vector, $b_j^T x$ denotes the scalar product, and $\sigma : \Re \mapsto \Re$ is called an *activation function* (usually a sigmoid function). The output of the output node is given by

$$h(w, x) = \sum_{j=1}^{M} a_j \sigma(b_j^T x - t_j) \tag{2.2.1}$$

where $a = (a_1, a_2, \ldots, a_M)$ is the weight vector of the output node, M is the number of neurons in the hidden layer, and w denotes the *parameter* or *weight vector* of the network that consists of a, $b_1, b_2, \ldots, b_M$, and $t_1, t_2, \ldots, t_M$. Then $h(w, x)$ is called the *output* of the neural network with the *weight vector w.*

For sigmoids functions σ, Cybenko [5] showed that any continuous and bounded function $f : [0,1]^d \mapsto \Re$ can be approximated by a function $h(w, x)$ of the form (2.2.1) in the sense

$|f(x) - h(w,x)| < \epsilon$ for all $x \in [0,1]^d$. Here the training of a neural network corresponds to computing a suitable weight vector w based on a sample. The unknowns a_j's correspond to the weights of the last layer, but the b_j's correspond to weights of the hidden layer. Networks with unknown weights located only in the last layer are amenable to learning algorithms based on potential functions. Such networks are proposed by Kurkova [11] who showed that any continuous function can be represented within an arbitrarily specified precision ϵ in the form

$$\sum_{i=1}^{M} a_i \eta_i(x) \tag{2.2.2}$$

where the functions $\eta_i(.)$ are universal and the weights a_i's depend on the function being approximated.

2.3 Stochastic Approximation and Potential Function Methods

One of the simplest of stochastic approximation algorithms takes the following form

$$w_{n+1} = w_n + \gamma_n s_n(w_n, \zeta_n) \tag{2.3.1}$$

where the real vector w_n is an estimate of the parameter of interest at nth step, $\{\gamma_n\}$ is a sequence of scalars, $\{\zeta_n\}$ is a sequence of random variables, and $s_n(w_n, \zeta_n)$ is a random variable called the *update rule*. For example, in solving $\min_w f(w)$, where gradient estimates of $f(.)$ involve random error terms, $s_n(.)$ could correspond to the noisy estimate of the gradient. The convergence conditions of this type of algorithm have been extensively studied (Kushner and Clark [12], Benveniste *et al.* [3], Wasan [28]); typically the convergence results are asymptotic. Notice that the algorithm (2.3.1) incrementally estimates a vector of fixed dimension, and the function and regression learning problems involve estimation of functions. The density results of last section enable us to approximate continuous functions by finite dimensional vectors. Thus at the expense of settling only for an approximation, the stochastic approximation algorithms can be used for function estimation.

For illustration we consider an algorithm based on the potential functions of Aizerman *et al.* [1] (see also Fisher and Yakowitz [8]). Consider a function of the form

$$f(x) = \sum_{i=1}^{M} a_i \phi_i(x) \tag{2.3.2}$$

where $\phi_i(x)$ form a linearly independent set of functions. For some real $\lambda_1, \lambda_2, \ldots, \lambda_M$ let

$$K(y,z) = \sum_{i=1}^{M} \lambda_i^2 \phi_i(y) \phi_i(z). \tag{2.3.3}$$

Given an n-sample $(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n))$, and for $\Lambda > \frac{1}{2} \max_{y \in Y} K(y,y)$ consider the following algorithm

$$f^{n+1}(x) = f^n(x) + \frac{1}{\Lambda}[f(x_n) - f^n(x_n)] K(x, x_n) \tag{2.3.4}$$

that can be easily expressed in terms of the coefficients a_i and actually used to update these coefficients. The conditions under which $f^n(.)$ converges to $f(.)$ have been studied extensively. A survey of these results is provided in [1]; our application involves the results shown by Braverman and Pjatnickii [4], which deal with the case where M is finite. The density results of Kurkova [11] enable us to apply these results to wide classes of learning algorithms. Notice that these results are not directly applicable to the functions of the form (2.2.1) since the parameters a_j, b_j, and t_j all depend on the function being approximated; these functions can be handled by the stochastic approximation methods.

2.4 Empirical Estimation

For family $\{A_\gamma\}_{\gamma\in\Gamma}$, $A_\gamma \subseteq A$, and for a finite set $\{a_1, a_2, \ldots, a_n\} \subseteq A$ we define:

$$\Pi_{\{A_\gamma\}}(\{a_1, a_2, \ldots, a_n\}) = \{\{a_1, a_2, \ldots, a_n\} \cap A_\gamma\}_{\gamma\in\Gamma},$$

$$\Pi_{\{A_\gamma\}}(n) = \max_{a_1,a_2,\ldots,a_n} |\Pi_{\{A_\gamma\}}(\{a_1, a_2, \ldots, a_n\})|.$$

The following identity is established in [26].

$$\Pi_{\{A_\gamma\}}(n) = \begin{cases} 2^n & \text{if } n \leq k \\ < 1.5\frac{n^k}{k!} & \text{if } n > k. \end{cases}$$

Notice that for a fixed k, the right hand side increases exponentially with n until it reaches k and then varies as a polynomial in n with fixed power k. This quantity k is called the *Vapnik-Chervonenkis dimension* of the family of sets A_γ.

For a set of functions, the *capacity* [27] is defined as the largest number h of pairs (x_i, y_i) that can be subdivided in all possible ways into two classes by means of rules of the form $\{\Theta[(y - f(x))^2 + \beta]\}_{f,\beta}$ where

$$\Theta(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{f } z < 0. \end{cases}$$

Formally, the capacity of a family of functions $\mathcal{F}$ is the Vapnik-Chervonenkis dimension of the set of indicator functions $\{\Theta[(y - f(x))^2 + \beta]\}_{(f,\beta)\in\mathcal{F}\times\Re}$.

Consider $f_* \in \mathcal{F}$ that minimizes the expected error in

$$Q(\hat{f}) = \int_{X,Y} [y - \hat{f}(x)]^2 dP_{X,Y} \tag{2.4.1}$$

over all $\hat{f} \in \mathcal{F}$ based on a finite sample $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ generated independently and identically according to the probability distribution $P_{X,Y}$. Let $\hat{f} = f_{emp}$ minimize the empirical error

$$Q_{emp}(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{f}(x_i)]^2 \tag{2.4.2}$$

over all $\hat{f} \in \mathcal{F}$. The closeness of f_{emp} to f_* is specified by the parameters *precision* ϵ and *confidence* δ in the condition $P[Q(f_{emp}) - Q(f_*) > \epsilon] < \delta$ where $P = P^n_{X,Y}$ is the product measure on the set of all independently and identically distributed n-samples. In order to ensure the (ϵ, δ)-condition, two types of conditions are to be satisfied [26]: (a) the capacity of $\mathcal{F}$ must be bounded; and (b) the error $I(.)$ must be bounded.

Theorem 2.1 *Suppose that the error is bounded as* $\sup_{x,y,\hat{f}}(y - \hat{f}(x))^2 \leq \tau$ *for* $\hat{f} \in \mathcal{F}$.

(i) Let h be the capacity of $\mathcal{F}$. Then given n examples, we have

$$P[Q(f_{emp}) - Q(f_*) \geq 2\tau\kappa] \leq 9\frac{(2n)^h}{h!}e^{-\kappa^2 n/4}.$$

(ii) Let the hypothesis space be finite in that $\mathcal{F} = \{f_1(x), f_2(x), \ldots f_s(x)\}$. Then given n examples, we have $P\left[Q(f_{emp}) - Q(f_*) > 2\tau\kappa\right] < 18sne^{-\kappa^2 n/4}$.

Parts (i) and (ii) of this theorem directly follow from Theorem 7.1 and 7.3 of Vapnik [26] respectively. Similar results can be shown under the conditions of bounded error and simpler solution conditions (see Rao [18]). The required boundedness results of sigmoidal FFN are established in Macintyre and Sontag [14]. Note that the results of this theorem are mainly existential in nature in that they do not directly yield computational methods to obtain the required f_{emp}. Some methods to compute f_{emp} or its approximation will be needed in order to utilize the results of this theorem in learning algorithms.

The minimization of (2.4.1) is intimately connected to the estimation of a regression function $f(x) = E(y|x)$. The function (2.4.1) can be rewritten as follows [26]

$$Q(\hat{f}) = \int_{X,Y} [y - f(x)]^2 dP_{X,Y} + \int_X [\hat{f}(x) - f(x)]^2 dP_X - \int_{X,Y} [\hat{f}(x) - f(x)][y - f(x)] dP_{X,Y}.$$

The last term can be expanded as $\int_X [\hat{f}(x) - f(x)] \left[\int_Y [y - f(x)] dP_{Y|X}\right] dP_X$ where $P_{Y|X}$ is the distribution of the conditional random variable Y given X. This term is zero since the quantity inside square brackets is zero. Thus, the minimum of $Q(\hat{f})$ is achieved at the regression function $\hat{f} = f$ since the first term of $Q(\hat{f})$ is independent of $\hat{f}$.

3 Learning Algorithms Based on Potential Functions

3.1 Function Estimation

The following condition is utilized for the method of potential functions.

Condition 3.1 *For a fixed M, any function $f \in \mathcal{F}$ is given by $f(x) = \sum_{j=1}^{M} a_j \phi_j(x)$, where w is the parameter vector with components a_i such that $\sum_{j=1}^{M} a_j^2 \neq 0$, and $\int_X f^2(x) dP_X > 0$.*

This condition is satisfied if $f(.)$ is continuous and vanishes at no more than a finite number of points. This condition implies that the $M \times M$ matrix $[\int_X \phi_i(x)\phi_j(x)p(x)dx] = [\rho_{ij}]$ is positive definite. Thus we have $r\sum_{i=1}^{M} a_i^2 \leq \sum_{i=1}^{M}\sum_{j=1}^{M} a_i \rho_{ij} a_j \leq R \sum_{i=1}^{M} a_i^2$ where r and R are the smallest and largest eigenvalues of the matrix $[\rho_{ij}]$.

Theorem 3.1 *Under Condition 3.1, for $f \in \mathcal{F}$ and f^n produced by the algorithm (2.3.4), we have $P[I(f^n) < \epsilon] > 1 - \delta$ for sufficiently large sample size $n = \ln(\delta\epsilon/RC)/\ln(1 - ra)$ where $C = \sum_{i=1}^{M} a_i^2$ and $a = \frac{1}{\Lambda}\left[2 - \frac{\max_{x \in X} K(x,x)}{\Lambda}\right]$, with $1 - ra \geq 0$, r and R are the smallest and largest eigenvalues of the matrix $[\rho_{ij}]$, and Λ is a free parameter chosen such that $a > 0$. Furthermore $I(f^n)$ converges to 0 with probability one.*

Outline of Proof: Braverman and Pyatnitskii (Theorem 1 of [4]) showed that $E[I(f^n)] \leq RC(1-ra)^n$, which is combined with the Chebyshev's inequality to show the theorem. The detailed proof can be found in [19]. □

The algorithm (2.3.4) uses constant *step-size* $\frac{1}{\Lambda}$, which is referred to as the learning rate in the context of neural network algorithms. Similar results can be obtained for variable step-size as in the following algorithm:

$$f^{n+1}(x) = f^n(x) + \gamma_n sign[f(x_{n+1}) - f^n(x_{n+1})]K(x, x_{n+1}) \tag{3.1.1}$$

where γ_n is a sequence of positive numbers such that $\sum_{i=1}^{\infty} \gamma_i \to \infty$ and $\sum_{i=1}^{\infty} \gamma_i^2 < \infty$.

We now address the question of using the algorithm of the form (2.3.4), based on functions of the form $f(x) = \sum_{i=1}^{M} a_i\phi_i(x)$, to approximate a function of the form $g(x) = \sum_{i=1}^{M_1} c_i\chi_i(x)$. The question is answered in the affirmative by Theorem 3.2 in [19].

3.2 Regression Estimation

Now consider the problem of minimizing $Q(\hat{f}) = \int_{X,Y} [\hat{f}(x) - y]^2 dP_{X,Y}$ over all $\hat{f} \in \mathcal{F}$ based on a finite sample. From Section 2.4, the regression function $f(x)$ minimizes $Q(\hat{f})$, since

$$Q(\hat{f}) = \int_{X,Y} [y - f(x)]^2 dP_{X,Y} + I(\hat{f})$$

and $I(f) = 0$. By treating the sample as if it had been generated by a function f_{emp} (i.e. $y_i = f_{emp}(x_i)$ for $i = 1, 2, \ldots, n$), we compute an approximation $\hat{f}_{emp}$ to f_{emp} such that

$$P[I(\hat{f}_{emp}) > \epsilon_{emp}] < \delta$$

by using the potential function method. Then we show that $\hat{f}_{emp}$ is close to the regression function $f(x)$ in the following theorem under Condition 3.1. We now state a more general result applicable to the case when there is no $\hat{f} \in \mathcal{F}$ consistent with the sample.

Theorem 3.2 *Let* $f^* = \min_{\hat{f} \in \mathcal{F}} I(\hat{f}, f)$ *and* $\epsilon^* = I(f^*, f)$. *Under Condition 3.1 together with* $sup_{x,y,f}(y - f(x))^2 \leq \tau$ *and finite capacity of* $\mathcal{F}$, *and* $\delta = 9\, e^{-\epsilon^2 n/16\tau^2}$ *and* $\hat{f}(.) = f^n$ *produced by algorithm (3.1.1), we have*

$$P[I(\hat{f}, f) < (\sqrt{\epsilon^*} + \sqrt{\epsilon} + \sqrt{\epsilon_{emp}})^2] > 1 - \delta$$

for sufficiently large sample size n *such that* $\epsilon_{emp} = \frac{c}{\delta}(1 - ra)^n$, *where* c *and* r *are constants.*

4 Learning Algorithms Based on Stochastic Approximation

We now consider the problem of learning parameters of the network of the form (2.2.1) for which the potential function methods are not directly applicable.

Condition 4.1 *For a fixed* M, *any function* $f \in \mathcal{F}$ *is given by a FFN of single hidden layer* $f(x) = h(w, x) = \sum_{j=1}^{M} a_j\sigma(b_j^T x + t_j)$ *where* $\sigma(x)$ *is a sigmoid function and* w *is the* parameter vector *of the network that consists of* a, $b_1, b_2, \ldots, b_M$, *and* $t_1, t_1, \ldots, t_M$.

The basic structure of the algorithm is based on iteratively updating the weight vector of the network as follows

$$w_{n+1} = w_n + \Gamma_n[|h(w_n, x_{n+1}) - f(x_{n+1})|] \tag{4.1}$$

where $w_n, w_{n+1} \in \Re^N$, each component of $\Gamma_n \in \Re^N$ consists of scalar γ_n (same in all components of Γ_n) called the *step size*, and $(x_n, f(x_n))$ is the nth example. The expression $|h(w_n, x_n) - f(x_n)|$ is the *update rule.* In the context of neural networks, algorithms similar to (4.1) have been studied by Nedeljkovic [16], Finnoff [7], Darken and Moody [6], and Stankovic and Milosavljevic [23]; the results are typically asymptotic.

Note $J(w) = \int |h(w, X) - f(X)| dP_X$ denotes the expected absolute error made by the hypothesis (corresponding to a FFN with parameter w) and $E[(|h(w_n, x_n) - f(x_n)|)|w_n] = J(w_n)$. The performance of the stochastic algorithms of type described in (4.1), can be characterized in terms of a Lyapunov function $V(w_n)$ as described in Polyak [17]; here, by the specific choice of the update rule, $J(w)$ plays the same role as $V(w)$ of [17]. In order to ensure the convergence of the algorithm (4.1), we utilize the following conditions on the probability measure generated by P_X.

Condition 4.2 *Let $J(w)$ be differentiable and let its gradient satisfy the following Lipschitz condition: for all $u, v \in \Re^N$, there exists a positive constant L such that*

$$|| \bigtriangledown J(u) - \bigtriangledown J(v)|| \leq L||u - v||.$$

Condition 4.3 *There exists a scalar θ such that for any w and N-dimensional vector $\Psi(w) = (J(w), \ldots, J(w))^T$, we have*

$$\bigtriangledown J(w)^T \Psi(w) \geq \theta J(w),$$

where $\bigtriangledown J(w)^T = \left(\frac{\partial J(w)}{\partial w_1}, \ldots, \frac{\partial J(w)}{\partial w_N}\right)$. Let us denote by $\underline{1}$ the column vector with all entries equal to 1. Then the above condition implies $\bigtriangledown J(w)^T \underline{1} \geq \theta$, everywhere except at $J(w) = 0$.

Theorem 4.1 *Under Conditions 4.1-4.3, algorithm (4.1) satisfies the following properties.*

(i) (a) Under the condition $\sum_{j=0}^{\infty} \gamma_j^2 < \infty$, the sequence of random variables $\{J(w_n)\}$ generated by the algorithm (4.1) converges to a finite random variable J_w.

(b) Under the conditions $\gamma_n \to 0$ and $\sum_{i=1}^{\infty} \gamma_i \to \infty$, as $n \to \infty$ we have $E[J(w_n)] \to 0$ as $n \to \infty$.

(c) In addition to conditions in (b), under the condition $\lim_{n\to\infty} \frac{1}{\theta}\left(\frac{1}{\gamma_{n+1}} - \frac{1}{\gamma_n}\right) \geq \gamma \geq 1$, we have $J(w_n) \to 0$ with probability one.

(ii) For $\gamma_n < 1/\theta$ for all n, we have $P[J(w_n) < \epsilon] > 1 - \delta$ for sufficiently large sample of size n given by $\prod_{j=0}^{n-1}(1 - \gamma_j \theta) = \frac{\delta\epsilon - \frac{LN}{2\theta^2}}{1 - \frac{LN}{2\theta^2}}$. The required sample size can also be given by $\sum_{j=0}^{n-1} \gamma_j = \frac{1}{\theta} \ln\left(\frac{1 - \frac{LN}{2\theta^2}}{\epsilon\delta - \frac{LN}{2\theta^2}}\right)$.

(iii) Under the additional conditions: $\gamma_n \leq \theta$, $\gamma_n \to 0$, $\sum_{j=0}^{\infty} \gamma_j = \infty$, *and* $1/\gamma_{n+1} - 1/\gamma_n \geq \rho\theta \geq \theta$, *we have* $P[J(w_j) < \epsilon \text{ for all } j] > 1 - \delta$ *for sufficiently large sample of size* n *given by* $\sum_{j=0}^{n-1} \gamma_j = \ln\left(\frac{1+\frac{LN\gamma_0}{2\theta}}{\epsilon}(1 + 1/\delta)\right)$.

Outline of Proof: Almost super-martigale results are used to obtain bounds on the expectation of $J(w_n)$, which are combined with Chebyshev's inequality to obtain the sample sizes [19]. □

Part (i) of this theorem characterizes the asymptotic behavior of the algorithm (4.1). The condition (i)(b) corresponds to the Robbins-Monro conditions. Part (ii) and (iii) yield the sample sizes needed to ensure (ϵ, δ)-condition at the n step of the algorithm. The condition ensured in (ii) is just on $J(w_n)$ at the nth step, and this condition might not have been satisfied before the nth step, whereas the condition (iii) is valid uniformly for all n, and hence is stronger than (ii).

More generally, the algorithm of the form (4.1) based on functions of the form $f(x) = h(w, x) = \sum_{j=1}^{M} a_j\sigma(b_j^T x + t_j)$ can be used to approximate a function of the form $g(x) = \sum_{j=1}^{M_1} c_j\sigma(d_j^T x + e_j)$ as described in the extended version [19].

4.1 Regression Estimation

We consider a different class of algorithms based on the Hilbert space methods of Revesz [21]. The applicability of the stochastic approximation methods based on kernel functions for regression estimation problem is established by Revesz [21]. Now consider the algorithm

$$f^{n+1}(x) = f^n(x) + \frac{1}{(n+1)r_{n+1}} K\left(\frac{x - x_{n+1}}{r_{n+1}}\right)(y_{n+1} - f^n(x)) \qquad (4.1.1)$$

where $e_n = n^{-\alpha}$ $(0 < \alpha < 1)$ and kernel function $K(x)$ is an arbitrary density function. In order to convert (4.1.1) into an algorithm for a neural network of fixed size, the update function $\frac{1}{(n+1)r_{n+1}} K\left(\frac{x-x_{n+1}}{r_{n+1}}\right)(y_{n+1} - f^n(x))$ must be converted to a form that can be used to change the weights of the network representing $f^n(n)$. In general, such computation may not be possible because the update function may require more than M terms to be expressed exactly as a sum of units. We impose additional conditions on the sigmoidal functions used in Condition 4.1 to facilitate this update.

Condition 4.4 *Under Condition 4.1, let the family of functions* $\Xi = \{a\sigma(b^T x + t) | a \in \Re, b \in \Re^d, t \in \Re\}$, *which consists of translated and scaled versions of sigmoid function* $\sigma(.)$, *be closed with respect to the* product *operation such that for any* $\sigma(b_1^T x + t_1) \in \Xi$, $\sigma(b_2^T x + t_2) \in \Xi$, *we have the sum* $a_1\sigma(b_1^T x + t_1) + a_2\sigma(b_2^T x + t_2) \in \Xi$ *and product* $a_1 a_2 \sigma(b_1^T x + t_1)\sigma(b_2^T x + t_2) \in \Xi$.

For the case of the sigmoid function employed in Cybenko [5] this condition is satisfied. We choose the Kernel function as $K(x) = \sum_{i=1}^{M} c_i\sigma(d_i^T x + e_i)$ such that it corresponds to a density function (note that in view of Condition 4.4, any $K(x)$ expressed as a sum of M_1 terms can be converted into this form). The implementation of algorithm (4.1.1) for a network that satisfies Conditions 4.1 and 4.4 leads to an explicit recursive formula [19]. Then one can prove the following result.

Theorem 4.2 *Under Conditions 4.1 and 4.4, additionally suppose that (i) $f(y)$ is measurable and bounded, (ii) x has an absolutely continuous distribution with density $p(x)$ for which $1/2 \leq p(x) \leq \infty$, (iii) y is bounded with probability 1. For $f(x) \in \mathcal{F}$ we have for $\hat{f} = f^n$ $P[I(\hat{f}, f) > \epsilon] < \delta$ for sufficiently large sample size n such that $n = (C^{-1}\ln(1/\delta))^{\frac{1}{1-\rho}}$ where C is a function of only ϵ and ρ.*

Outline of Proof: The proof is similar to that in Theorem 4.1 except that bounds of Revesz [21] are used for the expectations.

5 Discussion

5.1 PAC Learning

Let f be an indicator function (which is not necessarily a continuous function) of any finite measurable set of $[0,1]^d$. For any $\varepsilon > 0$, for some M, there exists [5] a function $g(x)$ of the form (2.2.1) and a set $D \subset [0,1]^d$ with Lebesgue measure of at least $1-\varepsilon$ and $|g(x)-f(x)| < \varepsilon$ for $x \in D$. We impose a condition analogous to Condition 4.1 as follows:

Condition 5.1 *Let $X = [0,1]^d$ and for a fixed M, the set of functions of form $h(w,x) = \sum_{j=1}^{M} a_j\sigma(b_j^T x - t_j)$ approximate the set of indicator functions $\{1_c(.)\}_{c\in C}$ of the concept class C such that for each $1_c(.)$ and $\varepsilon > 0$, there exists some $h(w,.)$ such that $|1_c(x) - h(w,x)| < \varepsilon$ for all $x \in D \subseteq X$ such that the Lebesgue measure of D is at least $1-\varepsilon$.*

By employing stochastic approximation methods of Section 4, hypothesis $h(w_n)$ can be computed such that $P[\mu(h(w_n)\Delta c) < \epsilon] > 1-\delta$ for a sufficiently large sample under Conditions 4.2, 4.3 and 5.1 (note here $J[h(w,.)] = \int |h(w,X) - 1_c(X)|dP_X = \mu(h\Delta C)$). A detailed derivation of this class of algorithms can be found in Rao *et al.* [20]. The relationship between PAC learning and neural network methods is well-known (Haussler [10]); the present work provides an algorithmic framework for these aspects.

5.2 Wavelet Networks

Consider the *wavelet network* of Zhang and Benveniste [30] of the form

$$h(w,x) = \sum_{i=1}^{M} a_i\psi(D_i x - t_i) + \bar{g} \qquad (5.2.1)$$

where $a_i \in \Re$, $\psi : \Re^d \mapsto \Re$ is a *wavelet function* $t_i \in \Re^d$, $g \in \Re$, and D_i is $d \times d$ diagonal matrix with the diagonal entries given by $d_i \in \Re^d$. Stochastic approximation algorithms of last section can be applied to these networks. Also wavelets can be employed to play the similar role of sigmoids in the networks proposed by Kurkova [11], thus they are amenable to the potential function methods.

References

[1] M. A. Aizerman, E. M. Braverman, and L. I. Rozonoer. *Extrapolative problems in automatic control and method of potential functions*, volume 87 of *American Mathematical Society Translations*, pages 281–303. 1970.

[2] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):931–945, 1993.

[3] A. Benveniste, M. Metivier, and P. Priouret. *Adaptive Algorithms and Stochastic Approximations.* Springer-Verlag, New York, 1990.

[4] E. M. Braverman and E. S. Pjatnickii. Estimation of the rate of convergence of algorithm based on the potential functions method. *Automation and Remote Control*, 27(1):80–100, 1966.

[5] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Controls, Signals, and Systems*, 2:303–314, 1989.

[6] C. Darken and J. Moody. Towards faster stochastic gradient search. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances In Neural Information Processing Systems*, volume 4, pages 1010–1016. Morgan Kaufmann Publishers, San Mateo, California, 1992.

[7] W. Finnoff. Diffusion approximations fot the constant learning rate backpropagation algorithm and resistance to local minima. In S. T. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances In Neural Information Processing Systems*, volume 5, pages 459–466. Morgan Kaufmann Publishers, San Mateo, California, 1993.

[8] L. Fisher and S. J. Yakowitz. Uniform convergence of the potential function algorithm. *SIAM J. Control and Optimization*, 63(9):95–103, 1976.

[9] K. Funahashi. On the approximate realizations of continuous mappings by neural networks. *Neural Networks*, 2:183–192, 1989.

[10] D. Haussler. Decision theoretic generalizations of the pac model for neural net and other learning applications. *Information and Computation*, 100:78–150, 1992.

[11] V. Kurkova. Kolmogorov's theorem and multilayer neural networks. *Neural Networks*, 5:501–506, 1992.

[12] H. J. Kushner and D. S. Clark. *Stochastic Approximations for Constrained and Unconstrained systems.* Springer-Verlag, New York, 1979.

[13] M. Leshno, V. Ya. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6:861–867, 1993.

[14] A. Macintyre and E. D. Sontag. Finiteness results for sigmoidal neural networks. In *Proc. 25th Annual ACM Symp. on Theory of Computing*, pages 325–334. 1993.

[15] H. N. Mhaskar and C. A. Micchelli. Approximation by superposition of sigmoidal and radial basis functions. *Advances in Applied Mathematics*, 13:350–373, 1992.

[16] V. Nedeljkovic. A novel multilayer neural networks training algorithm that minimizes the probability of classification error. *IEEE Transactions on Neural Networks*, 4(4):650–659, 1993.

[17] B. T. Polyak. Convergence and convergence rate of iterative stochastic algorithms I: General case. *Automation and Remote Control*, (12):1858–1868, 1976.

[18] N. S. V. Rao. Fusion rule estimation in multiple sensor systems with unknown noise distributions. In *Proc. Indo-US Workshop on Parallel and Distributed Signal and Image Integration Problems*. 1993. to appear.

[19] N. S. V. Rao, V. Protopopescu, R. C. Mann, E. M. Oblow, and S. S. Iyengar. Learning algorithms for feedforward networks based on finite samples. Technical Report ORNL/TM-12819, Oak Ridge National Laboratory, September 1994.

[20] N. S. V. Rao, V. R. R. Uppuluri, and E. M. Oblow. Stochastic approximation algorithms for classes PAC learning problem. In *Proceedings of 1994 IEEE Conf. on Neural Networks*, pages 791–796. 1994.

[21] P. Revesz. How to apply the method of stochastic approximation in the non-parametric estimation of a regression function. *Math. Operationsforsch. Statist., Ser. Statistics*, 8(1):119–126, 1977.

[22] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[23] S. Stankovic and M. Milosavljevic. Training of multilayer perceptrons by stochastic approximation. In V. Milutinovic and P. Antognelli, editors, *Neural Networks, Concepts, Applications and Implementations*, volume IV, pages 201–239. Prentice-Hall, Englewood Cliffs, NJ, 1991.

[24] Z. Tang and G. J. Koehler. Deterministic global optimal FNN training algorithms. *Neural Networks*, 7(2):301–311, 1994.

[25] P. P. van der Smagt. Minimisation methods for training feedforward neural networks. *Neural Networks*, 7(1):1–11, 1994.

[26] V. N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer Verlag, New York, 1982.

[27] V. N. Vapnik. Inductive principles of the search for empirical dependences. In *Proceedings of Second Ann. Workshop on Computational Learning Theory*, pages 3–21, 1989.

[28] M. T. Wasan. *Stochastic Approximation*. Cambridge University Press, 1969.

[29] P. J. Werbos. Backpropagation through time:What it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[30] Q. Zhang and A. Benveniste. Wavelet networks. *IEEE Transactions on Neural Networks*, 3(6):889–898, 1992.