Learning Algorithms for Feedforward Networks Based on Finite Samples

Nageswara S. V. Rao, *Member, IEEE,* Vladimir Protopopescu, Reinhold C. Mann, *Member, IEEE,* E. M. Oblow, and S. Sitharama Iyengar, *Fellow, IEEE*

***Abstract*—We present two classes of convergent algorithms for learning continuous functions and regressions that are approximated by feedforward networks. 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.* [2]. The second class, applicable to general feedforward networks, is obtained by utilizing the classical Robbins–Monro style stochastic approximation methods. 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 be directly adapted to concept learning problems.**

I. Introduction

THE problem of learning (or inferring) a function or a set (concept) 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 guarantee that finite-sized networks can approximate continuous or indicator functions within any specified precision. These results enable us to formulate the function learning problem as one of estimating finite dimensional vectors (that typically represent connection weights of a network) at the cost of settling for an approximation. 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 heuristically. 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 1) the form of network employed and 2) the learning algorithm that computes the parameters of the network.

We illustrate the application of two well-known methods, namely the potential function method [2] and stochastic approximation [4], [37], to obtain learning algorithms implemented in two basic feedforward network architectures. The first method is applicable to feedforward networks with unknown weights located only in the output layer, and the second technique is applicable to general feedforward networks. For concreteness, we illustrate the first method using the networks of Kurkova [29] and the second using the networks of Cybenko [14]. Yet our approach can also be used to obtain: 1) efficient algorithms for learning sets (concepts)—for which most existing methods are nonalgorithmic [35], [53], and 2) learning algorithms for wavelet networks [59]—for which no finite sample results are known.

General density results guarantee that a continuous function can be approximated by a finite-sized network of nonpolynomial units with a single hidden layer within a specified precision [31], [34]. When the units are sigmoid functions, the networks are called feedforward neural networks [3], [14], [22]. Density properties of sigmoidal feedforward networks with two hidden layers have been studied by Kurkova [29] by using Kolmogorov's superposition theorem [28]. Similar density properties of slightly different architectures based on wavelets have been studied by Zhang and Benveniste [59]. These density results 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 task of learning functions from a finite sample, the utility of the above density results depends critically on the availability of suitable learning (or training) algorithms. There are several algorithms that train the networks of sigmoidal units based on finite samples [51], [54], [57]. The performance of such algorithms has been assessed only to a limited extent, and mostly in asymptotic cases. The popular backpropagation algorithm [46], [57], which is a gradient descent method based on mean square error, seems to be effective in some cases but very slow to converge in others. Significant effort has been spent to improve the convergence of this and similar gradient search algorithms [10], [15], [20], [27], [48]. Convergence properties of learning algorithms based on wavelet networks [59] are unknown. Also, to our knowledge, no learning algorithms have been published for the networks based on Kurkova's model [29].

In the 1960's, the method of potential functions was studied for the purpose of learning functions from samples by Aizerman *et al.* [2]. A number of properties of these algorithms, including finite sample results, have been extensively studied. Due to the specific representation used in [2], this method is not applicable *a priori* to wide classes of functions. By using Kurkova's networks [29], however, we employ this method to

Manuscript received September 19, 1994; revised February 3, 1996. This work was sponsored by the Engineering Research Program of the Office of Basic Energy Sciences of the U.S. Department of Energy, under Contract DE-AC05-96OR22464 with Lockheed Martin Energy Research Corp.

N. S. V. Rao, V. Protopopescu, R. C. Mann, and E. M. Oblow are with the Center for Engineering Systems Advanced Research, Oak Ridge National Laboratory, Oak Ridge, TN 37831-6364 USA.

S. S. Iyengar is with the Department of Computer Science, Louisiana State University, Baton Rouge, LA 70803 USA.

Publisher Item Identifier S 1045-9227(96)04395-0.

learn arbitrary continuous maps within a specified precision. In the more difficult case of estimating a regression function, we additionally employ the empirical estimation results of Vapnik [55] to obtain suitable learning algorithms.

The area of stochastic approximation has been well established since the pioneering works of Robbins and Monro in 1951 [44] (see also [4] and [30]). The relevance of these methods to the learning algorithms for neural networks has been recognized by a number of investigators. For example, White [58] showed that the popular backpropagation algorithm is an implementation of the Robbins–Monro-style algorithm for the problem of minimizing the mean-square error. Similar asymptotic results are also shown by Nedeljkovic [36] and also by Stankovic and Milosavljevic [50]. Rao *et al.* [42] establish that the concept learning problem can be solved by using a network of nonpolynomial units by employing stochastic approximation algorithms. We extend these results to the case of function and regression learning problems; in particular, the Hilbert space methods of Revesz [43] are utilized to obtain algorithms for feedforward networks.

The aim of this paper is to provide a comprehensive framework for designing efficient function learning algorithms based on two general classes of feedforward network architectures. Our main criteria are performance guarantees based on finite-sized samples. For most existing algorithms such results are not available, yet they are strongly needed in practical applications, where the samples are always finite. Although the elemental methods, e.g., stochastic approximation, potential functions, empirical estimation, have been individually well established, we provide a new synthesis of these methods tailored to the present problem formulation and solution. In particular, our contributions include the following:

1) combination of empirical estimation and potential function methods to obtain finite sample results for function and regression estimation (Section III);
2) application of Lyapunov methods of Polyak [37] to obtain finite sample results for learning algorithms based on stochastic approximation (Section IV);
3) combination of density results of feedforward networks with the Hilbert space methods of Revesz [43] to obtain learning algorithms for feedforward networks (Section IV-A);
4) constructive algorithms for solving several classes of concept learning problems (outlined in Section V-A); and
5) finite sample results for concept and function learning algorithms based on wavelet networks (outlined in Section V-B).

The organization of the paper is as follows. Some preliminary discussion on function and regression learning problems, empirical estimation methods, neural network approximations, stochastic approximation and potential function algorithms, are presented in Section II. The potential function algorithms are utilized in conjunction with Kurkova's networks [29] to learn arbitrary continuous functions and regressions in Section III. Learning algorithms based on stochastic approximation are described in Section IV. The concept learning problems, and wavelet network algorithms are described in Section V. To make the treatment self-contained, our presentation is partly tutorial in nature: we provide, in one place, the adapted versions of some existing theorems and proofs (that are otherwise scattered in various papers). Some useful results on combinatorial and martingale inequalities are reviewed in the Appendix.

II. Preliminaries

The basic formulation of the function and concept learning problem is reviewed in Section II-A. Empirical estimation of functions from finite samples is, in principle, possible as outlined in Section II-B. Yet most results to date are only abstract existence and/or asymptotic results. To construct concrete efficient learning algorithms, we use the density properties described in Section II-C to approximate functions by neural networks to any desired accuracy. As a result of this approximate representation, one can reduce the problem of function (and concept) learning to the much simpler problem of "learning" the approximating neural network. Such a network is actually represented by the finite dimensional vector formed by its parameters (essentially its weights). Two convergent algorithms for these finite-dimensional iterations are briefly discussed in Section II-D.

The following is a list of symbols that will be used throughout:

1) $\sigma(.)$: sigmoid function.
2) $\eta_i(.)$: component function of Kurkova's network.
3) $\psi(.)$: wavelet function.
4) γ_i: step-size of learning algorithms.
5) $\phi_i(.)$: constituent potential function.

Throughout the paper, X and x denote unconditional random and deterministic variables, respectively, and it is assumed that all functions satisfy the required measurability conditions.

A. Function and Regression Learning Problems

A training n-sample of a function f: $[0, 1]^d \mapsto \Re$ is given by $[X_1, f(X_1)]$, $[X_2, f(X_2)]$, $\cdots$, $[X_n, f(X_n)]$ where $X_1, X_2, \cdots, X_n$, $X_i \in [0, 1]^d$, are independently and identically distributed (IID) according to P_X. 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 $f(x)$. We consider either the expected square error

$$I(\hat{f}) = \int [\hat{f}(X) - f(X)]^2 \, dP_X \tag{1a}$$

or the expected absolute error

$$J(\hat{f}) = \int |\hat{f}(X) - f(X)| \, dP_X \tag{1b}$$

which is to be minimized over a family of functions $\mathcal{F}$ based on the given n-sample. Let $f^* \in \mathcal{F}$ minimize $I(.)$ [or $J(.)$] over $\mathcal{F}$. In general, f^* cannot be computed from (1a) or (1b) since P_X is unknown. Furthermore, since no restrictions are placed on the underlying distribution, it is not possible in general to infer the exact f^* (i.e., with probability one) based on a finite sample. Consequently, most often only an approximation $\hat{f}$ to f^* is feasible. We shall give sufficient conditions under which an approximation $\hat{f}$ to f^* can be computed such that for a sufficiently large sample we have

$$P[I(\hat{f}) - I(f^*) > \epsilon] < \delta \tag{2a}$$

or

$$P[J(\hat{f}) - J(f^*) > \epsilon] < \delta \tag{2b}$$

corresponding to (1a) and (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 IID n-samples. Thus the "error" due to $\hat{f}$ is to be bounded within an arbitrarily specified precision ϵ of minimum possible error, with an arbitrarily specified confidence $1 - \delta$ (given a sufficiently large sample). A special case of this formulation, where f is an indicator function, constitutes a basic version of PAC learning problem formulated by Valiant [53]. A general formulation of this nature, has been studied extensively in empirical risk minimization methods [55]. In the context of neural network learning, $\mathcal{F}$ corresponds to a class of all networks, and the problem of computing $\hat{f}$ involves computing a weight vector based on the given sample.

In the same context, we consider the more general regression learning problem, which is more general than the above problem. We are given (X_1, Y_1), (X_2, Y_2), $\cdots$, (X_n, Y_n) IID according to $P_{X,Y}$ ($X \in [0, 1]^d$, $Y \in \Re$) such that $f(x) = E_Y(Y|x)$. The problem is to compute an estimate $\hat{f}$ of the regression function f that satisfies (2a) or (2b) with $P = P_{X,Y}^n$.

B. Empirical Estimation of Functions

One of the basic questions in network learning problems deals with conditions under which a solution to (1) can be obtained when only a finite sample is given. The empirical estimation methods of Vapnik [55] (which are applicable to more general classes of problems) provide a basis for the feasibility of such solutions. An application of such ideas establishes that a consistent solution to (1) can be obtained in the context of (neural) network learning problems as shown by Farago and Lugosi [17]. These results are, however, existential in nature and do not provide concrete algorithms to evaluate the functions or suitable approximations thereof. We briefly summarize the results from [55] which will be used subsequently.

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

$$\begin{aligned}&\Pi_{\{A_\gamma\}}(\{a_1, a_2, \cdots, a_n\})\\ &\quad = \{\{a_1, a_2, \cdots, a_n\} \cap A_\gamma\}_{\gamma\in\Gamma}\\ &\Pi_{\{A_\gamma\}}(n)\\ &\quad = \max_{a_1, a_2, \cdots, a_n} |\Pi_{\{A_\gamma\}}(\{a_1, a_2, \cdots, a_n\})|.\end{aligned}$$

The following identity is established in [55]:

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

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; it can also be alternatively defined as the largest size h of a set $\{a_1, a_2, \cdots, a_n\} \subseteq A$ that can be subdivided in all possible ways into two classes by means of sets A_γ.

For a set of functions, the capacity 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

$$(\Theta\{[y - f(x)]^2 + \beta\})_{(f,\beta)\in\mathcal{F}\times\Re}$$

where

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

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

$$Q(\hat{f}) = \int_{X,Y} [Y - \hat{f}(X)]^2\, dP_{X,Y} \tag{3}$$

over all $\hat{f} \in \mathcal{F}$ based on the sample (X_1, Y_1), (X_2, Y_2), $\cdots$, (X_n, Y_n) IID according to $P_{X,Y}$. Consider the empirical error functional

$$Q_{emp}(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n} [Y_i - \hat{f}(X_i)]^2. \tag{4}$$

The minimizer of $Q_{emp}(.)$ over $\mathcal{F}$ is denoted by f_{emp}. The closeness of f_{emp} to f^* is specified by the parameters ϵ and δ in the condition

$$P[Q(f_{emp}) - Q(f^*) > \epsilon] < \delta$$

where $P = P_{X,Y}^n$. To ensure the (ϵ, δ)-bound, two conditions have to be satisfied: 1) the capacity of $\mathcal{F}$ must be bounded; and 2) the error $I(.)$ must be bounded [55].

Theorem 1: Suppose that the error is bounded as $\sup_{x,y,\hat{f}}[y - \hat{f}(x)]^2 \le \tau$ for $\hat{f} \in \mathcal{F}$.

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

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

2) Let $\mathcal{F} = \{f_1(x), f_2(x), \cdots f_P(x)\}$ for finite P. Then given n examples, we have

$$P[Q(f_{emp}) - Q(f^*) > 2\tau\kappa] < 18Pne^{-\kappa^2 n/4}.$$

Parts 1) and 2) of this theorem directly follow from [55, Theorems 7.1 and 7.3], respectively. Similar results can be shown under the conditions of bounded error and simpler solution conditions (see [40]).

The minimization of (3) is intimately connected to the estimation of a regression function $f(x) = E_Y(Y|x)$. The function (3) can be rewritten as follows [55]:

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

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 conditional distribution of Y given X. The above term is equal to 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 $\hat{f}$.

C. Function Approximation by Neural Networks

We first consider feedforward neural networks with a single hidden layer. A general architecture of a multilayer feedforward network consists of an input layer with d units and output layer with m units, and one or more hidden layers. Consider a network with a single hidden layer and single output node ($m = 1$). 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 $x = (x^1, x^2, \cdots x^d)$ is the input vector, $b_j^T x$ denotes the scalar product, and $\sigma: \Re \mapsto \Re$ is called an activation function. The output of the network is given by

$$h(w, x) = \sum_{j=1}^{M} a_j \sigma(b_j^T x - t_j) \qquad (5)$$

where w is the parameter vector consisting of $a_1, a_2, \cdots, a_M$, $b_1, b_2, \cdots, b_M$, and $t_1, t_2, \cdots, t_M$.

Cybenko [14] considered a continuous sigmoid function that is a specific form of continuous $\sigma(.): \Re \mapsto [0, 1]$ such that $\sigma(t) \to 1$ as $t \to +\infty$ and $\sigma(t) \to 0$ as $t \to -\infty$. He showed that for a continuous and bounded f: $[0, 1]^d \mapsto \Re$ there exists w such that the function $g(w, x)$ of the form (3) such that $|f(x) - g(w, x)| < \epsilon$ for all $x \in [0, 1]^d$. The training of a neural network here corresponds to computing a suitable weight vector w based on a sample. The unknowns a_j's correspond to the output layer, while b_j's correspond to the hidden layer.

The networks with unknown weights located only in the output layer are amenable to potential function method, as will be shown subsequently; such networks have been proposed by Kurkova [29]. We consider now feedforward networks with two hidden layers[1] [29]. It can be shown [29, Theorem 2] that any continuous and bounded function can be represented within an arbitrarily specified precision ϵ in the following form:

$$\sum_{q=1}^{m} \sum_{j=1}^{m(m+1)^n} \left\{ d_j \rho \left[\sum_{p=1}^{n} \sum_{i=1}^{m+1} v_j w_{pq} a_{qi} \rho(b_{qi} x_p + c_{qi}) \right] + u_j \right\}$$

where $\rho(.)$ is a fixed function and the other parameters depend on the function to be learned. Furthermore, this function can be represented in the following simpler algebraic form:

$$\sum_{i=1}^{M} a_i \eta_i(x) \qquad (6)$$

where the functions $\eta_i(.)$ are universal and the weights a_i's depend on the function being approximated. The functions $\eta_i(.)$ correspond to single hidden layer feedforward networks consisting of sigmoid functions (see Kurkova [29] for details on the construction of these functions). As shown in the original formulation of Kolmogorov [28], when $\epsilon = 0$, the elemental functions η_i are highly nonsmooth functions (see also [33]), which do not seem to be directly amenable to computer implementations. Approximate versions of these functions, however, have been implemented by Frisch *et al.* [21].

D. Stochastic Approximation and Potential Function Algorithms

One of the simplest stochastic approximation algorithms takes the form

$$w_{n+1} = w_n + \gamma_n s_n(w_n, \zeta_n) \qquad (7)$$

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 using a variety of techniques (for example, see [4], [30], and [56]). Notice that algorithm (5) 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.

We now consider an algorithm based on the potential functions of Aizerman *et al.* [2] (see also [19]). Consider a function that can be represented in the form

$$f(x) = \sum_{i=1}^{M} a_i \phi_i(x) \qquad (8)$$

where $\phi_i(x)$ are linearly independent functions. Now for some real $\lambda_1, \lambda_2, \cdots, \lambda_M$ let

$$K(y, z) = \sum_{i=1}^{M} \lambda_i^2 \phi_i(y) \phi_i(z). \qquad (9)$$

[1] Although the feedforward networks with single and double hidden layers have similar density properties, they might be quite different from other viewpoints. From a control perspective, the networks with two hidden layers possess stabilization properties that the networks with a single hidden layer do not, as illustrated by Sontag [49]. From a computational viewpoint, if a network is allowed to have size proportional to the sample size, a network with two-hidden layers that is consistent with all training examples can be easily produced as shown by Blum and Li [7]. The problem of computing a network with a single hidden layer that is consistent with the entire sample could be computation-intensive as shown by Blum and Rivest [6].

Given the sequence $[X_1, f(X_1)], [X_2, f(X_2)], \cdots$, consider the following algorithm:

$$f^n(x) = f^{n-1}(x) + \frac{1}{\Lambda}[f(X_n) - f^{n-1}(X_n)] \cdot K(x, X_n) \qquad (10)$$

such that $\Lambda > \frac{1}{2}\max_{x\in[0,1]^d} K(x, x)$. The conditions under which $f^n(.)$ converges to $f(.)$ have been studied extensively. A survey of these results is provided in [2]; our application involves the results shown by Braverman and Pjatnickii [9], which deal with the case where M is finite. The density results of Kurkova [29] 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 (2.5) since the parameters a_j, b_j, and t_j all depend on the function being approximated; these functions can be handled by stochastic approximation methods (see Section IV). The relationship between the potential function methods and the stochastic approximation methods has been discussed by Aizerman *et al.* [1] and Tsypkin [52].

III. Learning Algorithms Based on Potential Functions

Given a finite sample $[X_1, f(X_1)], [X_2, f(X_2)], \cdots, [X_n, f(X_n)]$ IID according to an unknown P_X, consider algorithm (10) which can be implemented in terms of coefficients as follows

$$a_i^n = a_i^{n-1} + \frac{1}{\Lambda}[f(X_n) - f^{n-1}(X_n)]\phi_i(X_n). \qquad (11)$$

We shall now provide sufficient conditions under which algorithms of this type can be used for solving the function and regression learning problems.

A. Function Estimation

The following condition is utilized for the potential function method.

Condition 1: There exists a natural number M such that any function $f \in \mathcal{F}$ with $\int_X f^2(x)\,dP_X > 0$ admits the expansion $f(x) = \sum_{j=1}^M a_j\phi_j(x)$ where $\{\phi_j\}$ is a linearly independent set and $\sum_{j=1}^M a_j^2 \neq 0$.

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 $[\rho_{ij}] = [\int_X \phi_i(x)\phi_j(x)p(x)\,dx]$ is positive definite. Thus

$$r\sum_{i=1}^M a_i^2 \le \sum_{i=1}^M\sum_{j=1}^M a_i\rho_{ij}a_j \le R\sum_{i=1}^M a_i^2$$

where r and R are the smallest and largest eigenvalues of the matrix $[\rho_{ij}]$.

Theorem 2: Suppose the sample size, n, is given by

$$n = \frac{\ln\left(\frac{\delta\epsilon}{RC}\right)}{\ln(1 - ra)}$$

where

$$C = \sum_{i=1}^M a_i^2$$

and

$$a = \frac{1}{\Lambda}\left\{2 - \frac{\max_{x\in[0,1]^d} K(x, x)}{\Lambda}\right\}$$

with $1 - ra \ge 0$, where r and R are the smallest and largest eigenvalues of the matrix $[\rho_{ij}]$, and Λ is a free parameter chosen such that $a > 0$. Then under Condition 1, for $f \in \mathcal{F}$, and f^n given by algorithm (11), we have

$$P[I(f^n) < \epsilon] > 1 - \delta.$$

Furthermore $I(f^n)$ converges to zero with probability one.

Proof: The outline of the proof is direct: Braverman and Pjatnickii ([9, Theorem 1]) showed that $E[I(f^n)] \le RC(1 - ra)^n$, which is combined with the Chebyshev's inequality to show the theorem.

We provide the details here for completeness (this proof can be found in [9] which makes use of results from earlier publications) and also to facilitate the proof of Theorem 5. Define the following quantities:

$$f(x) - f^n(x) = \sum_{i=1}^M \Delta a_i^n\phi_i(x)$$

where

$$\Delta a_i^n = a_i - a_i^n$$

and

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

and

$$\alpha_n = \sum_{i=1}^M (\Delta a_i^n)^2$$

$$I(f^n) = \beta_n = \int_X [f(X) - f^n(X)]^2\,dP_X = \sum_{i=1}^M\sum_{j=1}^M \Delta a_i^n\rho_{ij}\,\Delta a_j^n.$$

We express α_{n+1} in terms of α_n as

$$\alpha_{n+1} = \sum_{i=1}^M (\Delta a_i^n)^2 - 2r_{n+1}[\Delta a^n, \phi(X_{n+1})] + r_{n+1}^2\sum_{i=1}^M[\phi_i(X_{n+1})]^2$$
$$= \alpha_n - 2r_{n+1}[f(X_{n+1}) - f^n(X_{n+1})] + r_{n+1}^2K(X_{n+1}, X_{n+1})$$