Sparse Basis Selection: New Results and Application to Adaptive Prediction of Video Source Traffic

Amir F. Atiya, *Senior Member, IEEE*, Mohamed A. Aly, and Alexander G. Parlos, *Senior Member, IEEE*

***Abstract*—Real-time prediction of video source traffic is an important step in many network management tasks such as dynamic bandwidth allocation and end-to-end quality-of-service (QoS) control strategies. In this paper, an adaptive prediction model for MPEG-coded traffic is developed. A novel technology is used, first developed in the signal processing community, called sparse basis selection. It is based on selecting a small subset of inputs (basis) from among a large dictionary of possible inputs. A new sparse basis selection algorithm is developed that is based on efficiently updating the input selection adaptively. When a new measurement is received, the proposed algorithm updates the selected inputs in a recursive manner. Thus, adaptability is not only in the weight adjustment, but also in the dynamic update of the inputs. The algorithm is applied to the problem of single-step-ahead prediction of MPEG-coded video source traffic, and the developed method achieves improved results, as compared to the published results in the literature. The present analysis indicates that the adaptive feature of the developed algorithm seems to add significant overall value.**

***Index Terms*—Internet traffic, MPEG, sparse basis, sparse representation, video traffic prediction.**

I. Introduction

CURRENTLY, the motivation for predicting video source traffic flow arises from at least two important considerations in multimedia networks. Dynamic bandwidth allocation and end-to-end quality-of-service (QoS) control of real-time multimedia streams transported over networks that do not offer service guarantees, e.g., such as Internet protocol (IP) networks. It is quite possible that future networked computing needs and applications may bring forward additional circumstances in which video and audio source traffic prediction becomes critical.

Efficient and fair utilization of available bandwidth is a topic that has garnered much attention. In order to adapt the allocated bandwidth among network end-users dynamically, it is imperative to predict the traffic generated by end-users. As the fraction of video traffic over IP networks increases, accurate video source traffic prediction algorithms could significantly aid in the design of efficient dynamic bandwidth allocation mechanisms. Among the various problems associated with real-time video and audio transport over IP networks, is their delay and loss sensitive nature. An obvious, and simple, solution to this problem is destination-side buffering and/or edge-caching. The tradeoff is that the media content is not delivered to the destination in real-time or even in near real-time. Many applications, such as streaming and ondemand video and audio, are tolerant to such large delays in delivery, even enabling the reconstruction of lost packets at destination. There are many applications, however, that require real-time or near real-time packet delivery, such as gaming, conferencing, IP telephony, and tele-operation applications. In attempts to develop edge-based end-to-end QoS control for such IP network applications with real-time or near real-time content delivery needs, one is faced with the need to predict the source traffic bit-rate time-series.

Most of the research reported in the literature on multimedia source traffic prediction can be classified into two broad categories. The first category deals with the development of stochastic source models. The paper written by Bae and Suda [4] is a good survey of such video models [6], [23], [28], [31]. These approaches model video source traffic using a variety of methods such as Markov chains, statistical techniques, and correlation functions.

The other category is the empirical modeling and prediction approach. There is less work done in this category. Since the proposed method follows this approach, the empirical modeling approach is discussed in more detail. The majority of the proposed empirical models are linear. For example, Adas in [1] and [2] proposes two linear models: a Wiener-Hopf model and a normalized least mean square (NLMS) adaptive model. Yoo [43] develops an adaptive traffic prediction scheme for variable bit range (VBR) MPEG video sources that focuses on predicting the effects of scene changes. Both Adas's and Yoo's approaches are adaptive, in the sense that they continually update the predictor weight coefficients with time. Chodorek and Chodorek [10] develop a linear predictor for MPEG-coded video traffic based on partitioning of the phase-space into subregions.

Recognizing that nonlinearities in the traffic dynamics are present, a number of researchers considered models such as neural networks. Chang and Hu [7] implemented a model called pipelined recurrent neural network (PRNN) for the adaptive traffic prediction of MPEG video signals over asynchronous transfer mode (ATM) networks. Doulamis *et al.* in [18] and [19], investigate the application of neural networks for nonlinear traffic prediction of VBR MPEG-coded video sources. In [20], the authors propose an adaptable neural network architecture that can operate offline as well as online. Also, in [5], Bhattacharya *et al.* propose a feedforward and a recurrent

Manuscript received December 1, 2003; revised March 17, 2005. The work of A. G. Parlos was supported in part by the State of Texas Advanced Technology Program under Grants 999903–083, 999903–084, and 512-0225-2001, in part by the U.S. Department of Energy under Grant DE-FG07-98ID12641, and in part by the National Science Foundation under Grants CMS-0100238 and CMS-0097719.

A. F. Atiya and M. A. Aly are with the Department of Computer Engineering, Cairo University, Giza, Egypt (e-mail: amiratiya@link.net).

A. G. Parlos is with the Department of Mechanical Engineering, Texas A&M University, College Station, TX 77843 USA (e-mail: a-parlos@tamu.edu).

Digital Object Identifier 10.1109/TNN.2005.853426

neural network model [38] for the multistep-ahead prediction (MSP) of MPEG4-coded video source traffic.

In this paper, a novel method for single-step-ahead prediction (SSP) of video source traffic flow is presented. A key feature of the method is its online adaptation. The method can be used to adapt to changing video characteristics, such as changing scene types, or when a new video is presented. The authors believe that it is more effective to have a method that fine-tunes itself and tracks variations of different video types than having a robust method that is intended to work for all circumstances. This is true, while taking into account that video variations might often be swift and at times there may be only small amounts of recent and relevant data available for adjusting the predictors quickly.

The proposed model is based on a novel technology developed in the signal processing community, called sparse basis representation (or sparse subset selection) [9], [11], [27], [30], [32], [35], [42]. It is based on the premise of having a large dictionary of basis signals. Assuming that the basis is combined linearly, a selection algorithm chooses the most effective small set of basis signals. This technology has almost exclusively been confined to the signal processing community, though it has applications in signal representation and data compression. The machine learning community has started exploring these techniques in the last few years, including exploring its relationship to support vector machines [30], [34], [37], [41]. There have been hardly any applications of this technology to time-series prediction, so this work is also a contribution toward exploring this aspect of applications of sparse basis selection. As it is pointed out in the next section, the proposed method can also be extended to modeling *nonlinear* functions. It has been acknowledged in the literature that it is fairly hard to have a truly adaptive parametric nonlinear model. Training the model requires significant effort from the designer. It is risky to allow the model to adapt its parameters without supervision, as many things could go wrong, such as encountering a bad local minimum, too slow or too fast adaptation, etc. These issues will be alleviated, to some extent, in the proposed approach, though the proposed model is still a nonlinear parametric model.

The main two contributions of this paper, are the following.

- Propose a new sparse basis selection algorithm, based on updating the basis selection adaptively and online. All of the previous work on sparse basis selection approaches is offline in nature (even though the basis is selected sequentially). When a new data point arrives, it is desirable to update the selected basis starting from the previous choice in a recursive manner, rather than computing it from scratch. In this work, a computationally efficient approach is proposed to accomplish this update. This approach can handle very large data sets very fast. This is especially needed in this particular application, as some video traces could have close to 100 000 frames or more.
- Apply the proposed method to video source traffic prediction. The purpose is to explore the premise of the sparse basis selection technology as an adaptive forecasting tool, as well as solve a problem that could have a big impact on the efficiency and quality of video transport.

The paper is organized as follows. The next section presents a short introduction into the sparse basis representation problem. Section III presents the new adaptive algorithm. In Section IV, we present the simulations experiments, followed by a conclusion section.

II. Introduction to Sparse Basis Selection

A. Overview

Consider a desired linear representation of a signal or a group of signals in terms of a number of basis functions. The traditional approach has been to consider a fixed family of basis functions of varying parameters (such as sinusoids with varying frequencies, or Gaussian functions of varying centers). It would be desirable to have the signal represented by as small set of basis functions as possible. In the sparse basis selection problem, one considers a very large dictionary of possible basis functions (to have a large choice). For example, one can have a number of functions, such as sinusoids, Gaussians, sinc functions, wavelets, etc. From these, the smallest set of basis functions is selected that can represent the given signal(s) to a desired accuracy.

This topic has generated a lot of research activity since the mid nineties. The main contributions in the early phase of developing the idea can be found in [8], [11], [32], and [35]. Most of the work focused on developing efficient algorithms for selecting the best basis functions (from among the dictionary functions). Because of the combinatoric nature of the problem, it has been shown to be NP-hard, as demonstrated by Natarajan [35] and Davis [15], [16]. Most developed methods produce suboptimal solutions. A group of methods is based on first selecting the best vector, then the vector that together with the selected one produce best fit, and so on. Such an approach is termed *the forward greedy selection approach*, and the majority of the proposed methods are related to this type [3], [12], [32], [35]. Another approach, *the backward greedy selection approach*, starts with the full dictionary, and sequentially eliminates one vector after the other [13], [14], [40]. Other approaches are based on minimizing the L_p norm as well as some entropy-based criterion [39], minimizing the L_1 norm [17], or based on some algebraic approach [33].

The disadvantage of the forward selection approach is that once a basis vector is selected, it is kept forever. There is no mechanism to reconsider a previous choice. Backward selection algorithms tend to be computationally intensive. A reasonable alternative to these two approaches is to consider a *forward-backward* approach [34]. For such an approach, the selection is initiated with a set of randomly chosen basis vectors. The "worst" vector among the set, that is the one whose removal results in the least error increase, is removed. Subsequently, from the remaining pool the "best" vector, that is the one that results in the largest decrease in error, is added. The removal and addition processes are continued until the algorithm converges. This is the approach considered in this paper. In the next section, a new computationally efficient approach is presented to perform the forward-backward iteration in a recursive manner.

Even though the sparse basis selection problem focuses mainly on linear models, the approach utilized produces nonlinear prediction models. The way this is achieved is to have many of the model inputs as nonlinear transformations of

the original inputs. This is akin to the philosophy of support vector machines, but rather than regularizing the model through margin maximization, it is regularized through selecting a small subset of inputs. The idea of creating nonlinear transformations of inputs for use in a linear model has its origins in Pao's functional link networks [36]. Also, to be mentioned is the traditional traditional feature selection literature [26]. However the philosophy in these approaches is different from the sparse basis selection philosophy, which is based on very large input dictionaries. Also, an interesting analogy here is with boosting methods [22] where a large number of weak learners (like the basis here) are combined with different weights. However, unlike boosting, the methodology proposed here is to select the best few rather than combine all possible inputs.

In the proposed approach a number of inputs are extracted from the time series, such as differences, second derivative measures, the outputs of some moving average of past series values, differences from mean levels, and also some nonlinear inputs such as absolute differences, etc. Let these inputs be $u_i, i = 1, \ldots, I$. Nonlinear transformations of each of these inputs as well, such as

$$e^{au_i}, \quad \sqrt{bu_i + c}, \quad 1/\left(1 + e^{-du_i}\right) \ldots i = 1, \ldots, I \tag{1}$$

where a, b, c, d are constants, are also considered. Once the appropriate basis is selected, the prediction will be a nonlinear function of the inputs. For example, it could be

$$\hat{x} = w_1 e^{au_3} + w_2\sqrt{bu_5 + c} + w_3/\left(1 + e^{-du_5}\right) + w_4/\left(1 + e^{-du_{18}}\right). \tag{2}$$

where the w_i's are the weights. In addition, one could apply the nonlinear transformations on the raw time series before extracting the inputs (through moving averages and otherwise).

These nonlinear transformations, coupled with a large number of inputs extracted, provide the sparse basis selection method a lot of power and flexibility in terms of the type of (nonlinear) input–output mapping that could be achieved. In essence, the "learning" is mainly in the dynamic choice of the inputs. Without a basis selection procedure, there would be a very large number of adjustable parameters (weights), and over-fitting is inevitable. It is to be emphasized though that the search for the best sparse set of inputs is considered part of the model search, and therefore a very large dictionary of inputs (such as thousands) could lead to some over-fitting. Fortunately, because of the large data sets available (several thousands to several tens of thousands of data points), this was not an issue in the current application. Caution must be exercised when applying the proposed models to problems with small data sets.

B. Mathematical Preliminaries

Consider a pool of K possible inputs. Let $x(i) \in \mathcal{R}^{1\times K}$ and $y(i) \in \mathcal{R}$ represent, respectively, the training set input vectors and target outputs (the values to be forecasted), where i indexes time. Let the size of the training set be N. Let us arrange the input vectors in a matrix $X \in \mathcal{R}^{N\times K}$ with the rows being $x(i)$, and let us also arrange the target outputs $y(i)$ in a column vector y.

In the sparse basis selection problem the goal is to find a small number $J \ll K$ of effective inputs that lead to minimum error. Let $S \subset \{1, 2, \ldots, K\}$ denote the set of selected inputs and let $\bar{S} \equiv \{1, 2, \ldots, K\} - S$ represent the remaining inputs that were not selected. The problem can be posed mathematically as finding the set S of size J and the weight vector $w \in \mathcal{R}^{J\times 1}$ such that the error function

$$E = \|X_S w - y\|^2 \tag{3}$$

is minimized, where X_S is the matrix constructed from X by selecting the columns indexed by the set S. Analogously, let $X_{\bar{S}}$ be the matrix constructed from X by selecting the columns indexed by the set $\bar{S}$. As mentioned, the problem of selecting the optimal inputs is NP-hard. Most researchers consider incremental approaches, such as adding inputs sequentially (forward greedy approach), or eliminating them sequentially (backward greedy approach).

Consider the forward approach. At any step assume a number of inputs (corresponding to the columns X_S) with $|S| < J$ have been selected, and it is desired to select a vector from $\bar{S}$ to add to S. The least square error solution for the weight vector is given as

$$w = \left(X_S^T X_S\right)^{-1} X_S^T y \tag{4}$$

and the corresponding error is

$$E = y^T \mathcal{P}_{X_S}^{\perp} y = y^T \left(I - X_S \left(X_S^T X_S\right)^{-1} X_S^T\right) y \tag{5}$$

where the matrix $\mathcal{P}_{X_S}^{\perp}$ represents the projection on the null space of X_S. An efficient way to update the solution vector and to compare the contribution of each vector in $\bar{S}$ to determine which one to be added to S, is to consider the projection of $X_{\bar{S}}$ onto the null space of X_S, defined as

$$Z = \mathcal{P}_{X_S}^{\perp} X_{\bar{S}}. \tag{6}$$

It can be shown that the best element (in terms of minimizing error) to move from $\bar{S}$ to S is the one corresponding to column z of Z that achieves maximum

$$\kappa(z) = \frac{|z^T y|^2}{\|z\|^2}. \tag{7}$$

A proof will be given in Section III-A.1 and in the Appendix. Natarajan [35] derived a clever algorithm to update the projected matrix Z every forward selection step to reflect the added column in X_S.

III. Proposed Adaptive Algorithm

The developed algorithm is a forward/backward approach. The main steps are given as the following:

1) initialize using any J randomly selected inputs S;
2) remove the element from S that increases the resulting error by the least amount;
3) add the element of $\bar{S}$ in S that reduces the resulting error by the most amount;
4) repeat the last two steps until desired convergence is achieved.

An efficient new method is proposed here, that adaptively updates the selected vectors as new measurements arrive, without

applying the forward or backward algorithm on all the data from scratch.

Assume that at time $n-1$ a set of inputs S has already been selected. Also, assume that new measurements are now available at time n, and that it is desirable to update the set S and its corresponding weight vector w. To simplify the notation, let

$$U \equiv X_S \tag{8}$$
$$V \equiv X_{\bar{S}}. \tag{9}$$

To make the method adaptive and have the recent data have a larger influence on the selected vectors, a discounted error function is used

$$E(n) = \sum_{i=1}^{n} \alpha^{n-i} \left[u^T(i)w - y(i)\right]^2 \tag{10}$$

where $u^T(i)$ is the ith row of U, and α is the discount weight, in the range from 0 to 1 (usually closer to 1).

Let R denote the discount matrix, being an $n \times n$ diagonal matrix with $R_{ii} = \alpha^{n-i}$. The projection matrix is given by

$$\mathcal{P}_S = RU(U^TRU)^{-1}U^TR \tag{11}$$

and the minimum error is given by

$$E(n) \equiv y^T\mathcal{P}_S^{\perp}y = y^T[R - RU(U^TRU)^{-1}U^TR]y. \tag{12}$$

These formulas are obtained in a straightforward manner by adapting least squares theory to the case of a discounted error function.

A. Forward Update

When new input–output measurements arrive at time n, two tasks have to be performed. The first task is to examine each of the unselected vectors (the columns of V), and find the one vector leading to maximum reduction in error. This must be done in a computationally efficient recursive manner. Once determining the best vector from V to add to U, the second task is to update the relevant matrices and variables to reflect this new addition. The algorithms addressing both of these tasks are described in the sequel.

1) Determining the Best Column Vector to Add: The main approach in this work is to use (7) for each of the columns v of V in a recursive manner. Compute the new error at time n as a function of the error and other variables that are available at time $n-1$. Define

$$D(n-1) = (U^TRU)^{-1}. \tag{13}$$

Upon moving forward in time, and getting a new training data point, each of the selected basis vectors (the columns of U) are augmented by a new element. Denote the new U matrix as

$$U_{\text{new}} = \begin{pmatrix} U \\ u^T(n) \end{pmatrix} \tag{14}$$

with U being the value of the matrix at time $n-1$, and $u^T(n)$ being the new added row in U. The discount matrix of time n can be shown to be

$$R_{\text{new}} = \begin{pmatrix} \alpha R & 0 \\ 0^T & 1 \end{pmatrix} \tag{15}$$

where R is the old discount matrix and $0 = (0\ 0\ \cdots\ 0)^T$. The new $D(n)$ can be written as

$$\begin{aligned} D(n) &\equiv \left(U_{\text{new}}^T R_{\text{new}} U_{\text{new}}\right)^{-1} \\ &= \left(\alpha U^T RU + u(n)u^T(n)\right)^{-1}. \end{aligned} \tag{16}$$

Using the small-rank adjustment inversion formula [29], the following is obtained:

$$\begin{aligned} D(n) = \frac{1}{\alpha}(U^TRU)^{-1} \\ - \frac{\frac{1}{\alpha^2}(U^TRU)^{-1}u(n)u^T(n)(U^TRU)^{-1}}{1 + u^T(n)(U^TRU)^{-1}u(n)/\alpha}. \end{aligned} \tag{17}$$

For abbreviation, let

$$b(n) = (U^TRU)^{-1}u(n). \tag{18}$$

Then

$$D(n) = \frac{1}{\alpha}D(n-1) - \frac{b(n)b^T(n)}{\alpha^2 + \alpha b^T(n)u(n)}. \tag{19}$$

The quantity $D(n)$ is a key variable in the adaptive update. The other two key quantities are

$$C(n) = X^TRX \tag{20}$$
$$d(n) = X^TRy \tag{21}$$

(remember that X is the matrix consisting of all vectors including selected and unselected ones). Upon moving forward in time, all these quantities are updated in a straightforward manner, as follows:

$$C(n) = \alpha C(n-1) + x(n)x^T(n) \tag{22}$$
$$d(n) = \alpha d(n-1) + y(n)x(n) \tag{23}$$

where $x^T(n)$ is the new row added to matrix X corresponding to that data point at time n.

The matrices defined thus far will serve multiple purposes. To extract components from these matrices, define $C_{QZ}(n)$ as the submatrix of $C(n)$ consisting of rows indexed by set Q and columns indexed by set Z. Certainly, each of Q or Z can be a scalar. An analogous definition applies to the vector $d_Q(n)$, and generally this convention is used for any matrix or vector.

Assume that it is desired to add to the pool of selected inputs an input i_v from $\bar{S}$ (corresponding to column of v of matrix V). Equation (7) can be rewritten as

$$p(v) \equiv \frac{|v^T\mathcal{P}_S^{\perp}y|^2}{v^T\mathcal{P}_S^{\perp}v} \tag{24}$$

noting that $(\mathcal{P}_S^{\perp})^2 = \mathcal{P}_S^{\perp}$. Also

$$E'(n) = E(n) - p(v) \tag{25}$$

where $E(n)$ and $E'(n)$ are, respectively, the error functions before and after adding the new column. A proof for (24) is given in the Appendix. Substitutions into (24) results in

$$p(v) = \frac{(v^T Ry - v^T RUD(n)U^T Ry)^2}{v^T Rv - v^T RUD(n)U^T Rv}. \tag{26}$$

Equation (26) is solved in a computationally efficient manner by first updating the $C(n)$ and $d(n)$ matrices with the new data point at time n, as in (22) and (23). Then the following variables are computed

$$A_1 = D(n)C_{Si_v} \tag{27}$$
$$A_2 = D(n)d_S(n). \tag{28}$$

These computations results in

$$p(v) = \frac{\left(d_{i_v}(n) - C_{Si_v}^T A_2\right)^2}{g(v)} \tag{29}$$

where

$$g(v) = C_{i_v i_v}(n) - C_{Si_v}^T A_1. \tag{30}$$

Once $p(v)$ is computed for all v, then the vector v leading to maximum $p(v)$ is chosen to be added to the set S.

The proposed method is computationally efficient because it has avoided any computations of order N. The largest computation is of order K^2, in the update of (22), even after computing $p(v)$ for all possible vectors v. Another point to note is that one can use the fact that $D(n)$ is a rank-one adjustment of the previous matrix D, whether D results from the data point at $n-1$ (17), or a column addition (35) or a column deletion (36). This can make the computations in (27) and (28) of order $J(K-J)$ rather than $J^2(K-J)$, including the cycling over all v's. In order to avoid being sidetracked by algorithmic details, no further details of this technique will be discussed.

2) Updating the Matrices: Assume column v of matrix V resulted in the lowest error. It is now required to augment the set of selected columns U with the new addition v, and recompute all necessary matrices and quantities. Let U' be the new augmented U matrix. Thus

$$U' = (U \quad v). \tag{31}$$

Then, the matrix $D(n)$ can be updated recursively. Let $D'(n)$ be the updated matrix. It is given by

$$D'(n) = \begin{pmatrix} U^T RU & U^T Rv \\ v^T RU & v^T Rv \end{pmatrix}^{-1}. \tag{32}$$

By the block-partitioned matrix inversion formula

$$D'(n) = \begin{pmatrix} (U^T RU - \\ U^T Rvv^T RU)^{-1} & -\frac{(U^T RU)^{-1}U^T Rv}{c} \\ -\frac{v^T RU(U^T RU)^{-1}}{c} & \frac{1}{c} \end{pmatrix}. \tag{33}$$

where

$$c = v^T Rv - v^T RU(U^T RU)^{-1}U^T Rv. \tag{34}$$

The upper left submatrix of the previous matrix can be evaluated using the small-rank adjustment inversion formula, resulting in

$$D'(n) = \begin{pmatrix} (U^T RU)^{-1} + \\ \frac{(U^T RU)^{-1}U^T Rvv^T RU(U^T RU)^{-1}}{c} & -\frac{(U^T RU)^{-1}U^T Rv}{c} \\ -\frac{v^T RU(U^T RU)^{-1}}{c} & \frac{1}{c} \end{pmatrix}. \tag{35}$$

All quantities in the previous equation are available as they have been previously computed, resulting in

$$D'(n) = \begin{pmatrix} D(n) + \frac{A_1 A_1^T}{g(v)} & -\frac{A_1}{g(v)} \\ -\frac{A_1^T}{g(v)} & \frac{1}{g(v)} \end{pmatrix} \tag{36}$$

where the terms A_1 and v in this equation pertain to the vector that was selected for addition to the chosen set.

B. Backward Update

In the backward update a column vector from the matrix of selected vectors U is eliminated. By eliminating one vector at time, and checking the effect of each elimination on the error, the best vector to eliminate can be determined. To derive the backward update recursively, some of Reeves's formulas [40] for the backward basis update are utilized. Reeves has developed one of the most efficient backward greedy algorithms. Reeves showed that upon eliminating column number k in U, the error function becomes

$$E_k(n) = E(n) + \frac{|[(U^T RU)^{-1}U^T Ry]_k|^2}{[(U^T RU)^{-1}]_{kk}} \tag{37}$$

where $E_k(n)$ denotes the error after eliminating the kth column, $E(n)$ is the error before eliminating any column, and the subscript indexing in the RHS denotes the vector or matrix row/column number. Note that the discounting effect to Reeves's formulas have been added. All of the quantities needed are now readily available. For example

$$[(U^T RU)^{-1}]_{kk} = D_{kk}(n) \tag{38}$$

where $D_{kk}(n)$ is kth diagonal element of $D(n)$. Also

$$\begin{aligned}[(U^T RU)^{-1}U^T Ry]_k &= [D(n)d_S(n)]_k \\ &= [A_2]_k.\end{aligned} \tag{39}$$

For each $k = 1, \ldots, J$, $E_k(n)$ must be computed, to obtain the k that results in minimum $E_k(n)$. Once it is decided which column k to eliminate, the matrices and quantities used can be updated in a straightforward manner. Reeves's derivation will be closely followed, and adjusted as needed to reflect the discounting factor. Let $D(n)$ and $D'(n)$ denote the old value and the updated value of $(U^T RU)^{-1}$, respectively. Partition $D(n)$ as follows:

$$D(n) = \Pi \begin{pmatrix} H_k & h_k \\ h_k^T & D_{kk}(n) \end{pmatrix} \Pi^T \tag{40}$$

where Π represents a permutation matrix that moves the kth column and the kth row to the last column and row, respectively. The column vector $h_k \in \mathcal{R}^{(J-1)\times 1}$ represents the kth column of $D(n)$ with the kth element of that column excluded, that is without the diagonal element $D_{kk}(n)$. Using the block matrix inversion formula results in

$$D'(n) = H_k - \frac{h_k h_k^T}{D_{kk}(n)}. \tag{41}$$

C. Summary of the Algorithm

The algorithm with its forward update and backward update components can be summarized as follows.

1) *Initialize:* Start at time $n = n_0$. Choose the set S randomly as any J numbers in $\{1,\ldots,K\}$. Let $\bar{S} = \{1,\ldots,K\} - S$.
2) Compute initial variables:

$$D(n) = \left(X_S^T R X_S\right)^{-1} \tag{42}$$
$$C(n) = X^T R X \tag{43}$$
$$d(n) = X^T R y. \tag{44}$$

3) For $n = n_0 + 1$ to N do:

a) Update the matrices:

$$D(n) = \frac{1}{\alpha} D(n-1) - \frac{b(n) b^T(n)}{\alpha^2 + \alpha u^T(n) b(n)} \tag{45}$$

where

$$b(n) = D(n-1) u(n) \tag{46}$$
$$C(n) = \alpha C(n-1) + x(n) x^T(n) \tag{47}$$
$$d(n) = \alpha d(n-1) + y(n) x(n). \tag{48}$$

b) For I cycles repeat Steps c) then d).

c) *Backward recursion:* Compute

$$\hat{k} = \text{argmax}_k \frac{[D(n) d_S(n)]_k}{D_{kk}(n)} \quad k \in S. \tag{49}$$

Remove $\hat{k}$ from S: $S \equiv S - \{\hat{k}\}$, $\bar{S} = \bar{S} \cup \{\hat{k}\}$. Update $D(n)$ according to (40) and (41).

d) *Forward recursion:* For all $k \in \bar{S}$:

$$A_{1k} = D(n) C_{Sk} \tag{50}$$
$$A_2 = D(n) d_S(n) \tag{51}$$
$$g_k = C_{kk}(n) - C_{Sk}^T A_{1k} \tag{52}$$
$$\hat{k} = \text{argmax}_k \frac{\left(d_k(n) - C_{Sk}^T A_2\right)^2}{g_k}. \tag{53}$$

Move $\hat{k}$ from $\bar{S}$ to S: $\bar{S} \equiv \bar{S} - \{\hat{k}\}$, $S = S \cup \{\hat{k}\}$. Update $D(n)$ as follows:

$$D(n) \equiv \begin{pmatrix} D(n) + A_{1\hat{k}} \, A_{1\hat{k}}^T \big/ g_{\hat{k}} & -A_{1\hat{k}}/g_{\hat{k}} \\ -A_{1\hat{k}}^T \big/ g_{\hat{k}} & 1/g_{\hat{k}} \end{pmatrix}. \tag{54}$$

D. Computational Complexity

In computing the number of operations (flops), the discussion is limited to the most significant terms. For example, $5J^2 + 6J$ will be stated as $5J^2$. As before, it is assumed that $J \ll K \ll N$. The most expensive computation in step 3)a) occurs in (47), which needs around $3K^2/2$ flops. Consider now Step 3)b). The most expensive computation occurs in (50), which needs $2J^2(K-J)$ flops since it must be computed for all $k \in \bar{S}$. Each of (52) and (53) need $2J(K-J)$ flops. The total computation needed per new data point is approximately $3K^2/2 + 2IJ^2(K-J)$, where I is the number of backward-forward iterations performed per data point. Typically, the iterations I should be less than J.

As mentioned in Section III-A, the computation in (27) or (50) can be accelerated. Noting that $D(n)$ gets updated every time by a small rank adjustment, A_{1k} can be computed recursively, utilizing this fact, from the old A_{1k}. If this is considered, then the algorithm complexity becomes $3K^2/2 + J(K-J)(18I+5)$. If the number J is very large, then this approach results in computational savings.

IV. Video Source Traffic Predictors

A. Description of Data Streams Used

A typical MPEG-coded video consists of three types of frames. The intra-frames (or I-frames) are encoded using information from within that frame only. Nonintra frames are of two types, P-frames, and B-frames. The encoding of nonintra-frames utilizes motion compensation and relies on information from outside the frame. The frames are typically arranged in the following repetitive pattern (called GOP pattern):

$$IBBPBBPBBPBB. \tag{55}$$

Even though, this is not a standard requirement, the majority of videos use this convention.

Therefore, there are three prediction problems to address. That is prediction of the three time-series for the I-frames, the P-frames, and the B-frames. Each of the time-series has different characteristics. Because of their residual nature, the P-frames and the B-frames are usually harder to predict than the I-frames. Using cross-inputs from the other two series is fairly useful in the prediction of any of the three series.

The data used in this study have been obtained from a repository of downloadable MPEG-4 video traces maintained by the Technical University of Berlin [21]. The site offers high-, medium-, or low-quality encoded video sequences. The high-quality traces are chosen for this study.

B. Input Dictionaries

The following inputs are used for the I-frame prediction model (note that the numbering of the B-frames represents its order in the GOP, i.e., Bi is the B frame number i in the GOP):

1) first backward difference $b(t) = f(t) - f(t-1)$ where $b(t)$ is the input and $f(t)$ is the frame size
2) absolute value of input (1)
3) exponential of the difference in (1) after dividing it by the mean of input (2)
4) exponential of the absolute difference in (2) after dividing it by the mean of input (2)
5) sigmoid of (1) with appropriate scaling, where sigmoid is the function $1/(1+e^{-x})$
6) second backward difference $b(t) = f(t) - 2f(t-1) + f(t-2)$
7) spike-detecting input

$$b(t) = \begin{cases} 1 & \text{if } f(t) - f(t-1) > \\ & \text{mean}|f(t) - f(t-1)| \\ -1 & \text{if } f(t) - f(t-1) < \\ & -\text{mean}|f(t) - f(t-1)| \\ 0 & \text{otherwise} \end{cases}$$

8) another spike-detecting input

$$b(t) = \begin{cases} 1 & \text{if } f(t) - f(t-1) > \\ & \text{std}(f(t) - f(t-1)) \\ -1 & \text{if } f(t) - f(t-1) < \\ & -\text{std}(f(t) - f(t-1)) \\ 0 & \text{otherwise} \end{cases}$$

9) current frame minus the running average of past frames
10) a binarization of input (9)
11–15) difference between two moving averages, one using a small window of sizes 1, 3, 5, 8, and 12, and the other using a large window of sizes 2, 9, 15, 24, and 36, respectively
16) mean of previous 12 frames (including I-, B-, and P-frames)
17) mean of previous 2 B-frames
18) mean of previous 4 B-frames
19) mean of previous (B3, B4, B7, and B8) frames
20) mean of previous (B1, B2, B5, and B6) frames
21) mean of previous 8 B-frames
22) previous P-frame
23) mean of previous 3 P-frames
24) previous I-frame minus mean of 2 previous B-frames
25–48) logarithm of inputs (1)–(24) appropriately scaled
49–72) exponential of inputs (1)–(24) appropriately scaled
73–96) square root of inputs (1)–(24) appropriately scaled
97–120) sigmoid of inputs (1)–(24) appropriately scaled.

The inputs for the P-frame prediction model are as follows:

1–16) the same as for the previous I-frames but applied on P-frames
17) previous P-frame minus mean of P-frames
18) difference between previous 2 P-frames
19) mean of previous 2 P-frames
20) mean of previous 2 I-frames
21) mean of previous 2 B-frames
22) mean of previous 4 B-frames
23) mean of previous 8 B-frames
24) previous I-frame minus mean of previous 2 B-frames
25–48) logarithm of inputs (1)–(24) appropriately scaled
49–72) exponential of inputs (1)–(24) appropriately scaled
73–96) square root of inputs (1)–(24) appropriately scaled
97–120) sigmoid of inputs (1)–(24) appropriately scaled.

The inputs for the B-frame model were as follows:

1–16) the same as for the previous I-frames but applied on B-frames
17) previous P-frame minus mean of P-frames
18) mean of previous 2 P-frames
19) mean of previous 3 P-frames
20) mean of previous (B1, B2) for B1 and B2 frame types, similar input for other B-frame types
21) mean of (B5, B6) for B1 and B2 frames and vice versa, mean of (B7, B8) for B3 and B4 frames and vice versa
22) previous I-frame minus mean of I-frames
23) mean of previous 2 B-frames
24) mean of previous 2 I-frames
25–48) logarithm of inputs (1)–(24) appropriately scaled
49–72) exponential of inputs (1)–(24) appropriately scaled
73–96) square root of inputs (1)–(24) appropriately scaled
97–120) sigmoid of inputs (1)–(24) appropriately scaled.

C. Prediction Results

The traces used for training purposes are Aladdin, The Firm, and Star Wars. These video streams are used to adjust some parameters of the algorithm, such as the adaption rate α, the number of selected basis J, and the number of iterations for the backward forward algorithm NITER. Based on these runs these parameters are selected as follows: $\alpha = 0.999, J = 5, \text{NITER} = 30$. Furthermore, the training set is used to develop the pool of inputs. Some of the inputs are initially screened for predictive power using the training set, and they are added to the pool if proven useful.

The prediction results are compared to those of the neural network prediction model developed by Bhattacharya *et al.* [5]. It should be noted that if the present algorithm is applied starting at time $n = 1$ then matrix ill-conditioning problems will be encountered. The reason is that the outer product matrix $U^T R U$ becomes close to singular. To avoid this, the algorithm is started at some initial time n_0 other than time $n = 1$. In the event of starting the algorithm at $n = 1$, the matrices at time n_0 must be obtained by brute force rather than by the recursive approach (42)–(44). Subsequently, the matrices are adapted according to the recursive algorithm starting at $n = n_0 + 1$ till the end of the trace (Step 3 in Section III-C). For every video trace used, n_0 is set to 500 for the I-frame time-series, and equal to 2000 for the

TABLE I
PREDICTION PERFORMANCE FOR THE I-FRAME MODEL (IN %NMSE)

Trace	This Work	[5]
Aladdin*	2.50	2.6
Die Hard III	1.89	2.9
Jurassic Park I	0.75	0.8
Lecture Room	58.3	0.2
Mr Bean	2.22	-
Silence of the Lambs	1.61	3.6
Simpsons	1.33	-
Skiing	1.12	2.0
Starwars IV*	1.27	1.5
The Firm*	1.28	-

* indicates traces used for training.

TABLE II
PREDICTION PERFORMANCE FOR THE P-FRAME MODEL (IN %NMSE).

Trace	This Work	[5]
Aladdin*	9.67	10.3
Die Hard III	4.10	9.0
Jurassic Park I	3.36	4.0
Lecture Room	1.36	6.9
Mr Bean	6.15	-
Silence of the Lambs	4.05	11.0
Simpsons	25.20	-
Skiing	2.33	6.2
Starwars IV*	9.05	9.2
The Firm*	7.71	-

* indicates traces used for training.

TABLE III
PREDICTION PERFORMANCE FOR THE B-FRAME MODEL (IN %NMSE)

Trace	This Work	[5]
Aladdin*	9.25	8.2
Die Hard III	1.51	4.0
Jurassic Park I	1.70	2.2
Lecture Room	0.72	28.3
Mr Bean	3.08	-
Silence of the Lambs	1.53	27.8
Simpsons	10.95	-
Skiing	1.10	14.7
Starwars IV*	2.97	3.5
The Firm*	1.23	

* indicates traces used for training.

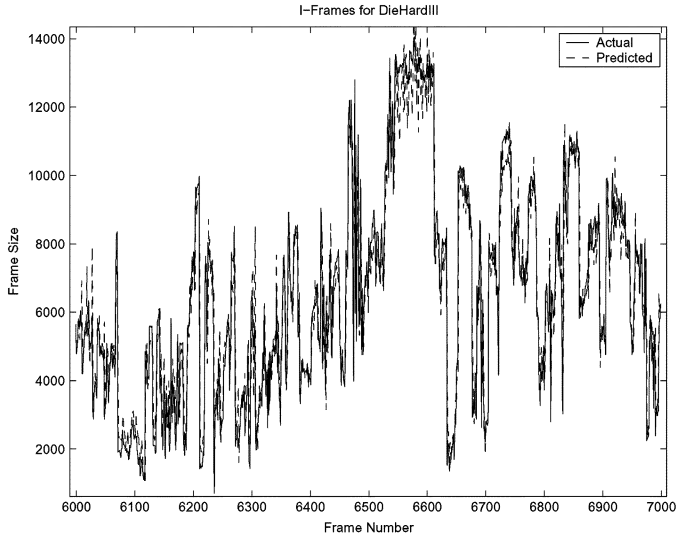


Fig. 1. Prediction and Actual Time-Series for the I-Frame of the Die Hard III Trace.

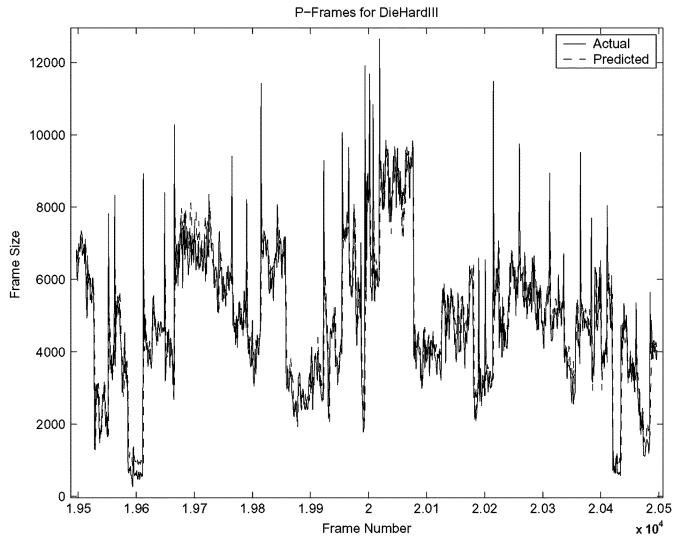


Fig. 2. Prediction and Actual Time-Series for the P-Frame of the Die Hard III Trace.

P-frame and B-frame time-series (the total sizes of the I-frame, P-frame, and B-frame time-series for one trace are typically around 7000, 20000, 60000, respectively). It is always preferable to have n_0 sufficiently large, to avoid ill-conditioning. As n_0 is increased ill-conditioning will be much less likely. On the other hand, for this particular comparison it is preferred to have n_0 as small as possible. As no prediction is performed in the interval from 1 to n_0, it is desirable to have the excluded initial part of the video trace as small as possible for the comparisons with the results of [5] to be as fair as possible. So the value of n_0 is selected as the smallest possible value that will still avoid ill-conditioning.

An error measure similar to the one used in [5] is employed in this study. Namely, the normalized mean square error (it is essentially SNR^{-1}) defined as

$$\text{NMSE} = \frac{\sum_{n=n_0}^{N}(y_n - \hat{y}_n)^2}{\sum_{n=n_0}^{N} y_n^2} \tag{56}$$

where y_n is the actual time-series value, $\hat{y}_n$ is the predicted value, and N is the length of the trace. The following traces are used as out-of-sample test data for the proposed approach: Die Hard III, Jurassic Park I, Lecture Room, Mr Bean, Silence of the Lambs, Simpsons, and Skiing [21]. Tables I–III summarize the prediction results for the I-frame model, the P-frame model, and the B-frame model, respectively, for all the traces used, including those used for training and for out-of-sample testing. Also included in these tables are the results, whenever available, from the work in [5], for comparison purposes. Figs. 1–3 depict the prediction versus actual value for a portion of Die Hard III's I-frame, P-frame, and B-frame time-series, respectively.

D. Discussion of the Prediction Results

The results presented thus far demonstrate that for most traces the proposed approach produces considerably better prediction results than the results in [5]. In turn, the results of [5] are some of the best that appeared in the literature. The improvement is

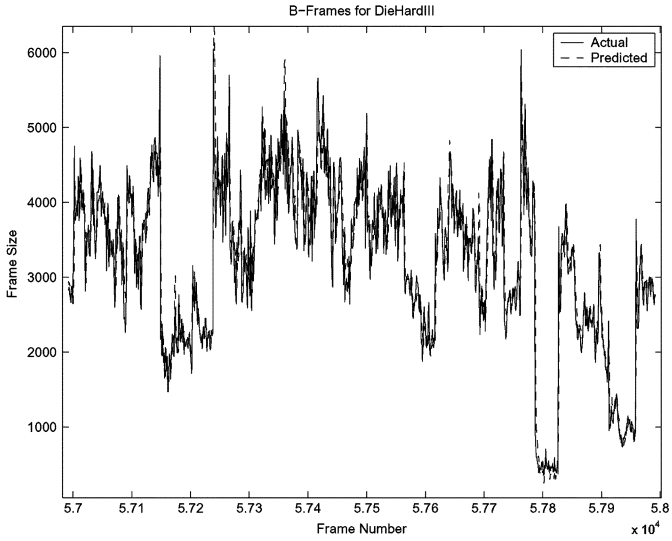


Fig. 3. Prediction and Actual Time-Series for the B-Frame of the Die Hard III Trace.

more pronounced for the P-frame and B-frame models. The exception is the Lecture Room trace I-frame results, where the proposed approach gives 58.3% error. The reason is due to the matrix ill-conditioning problem. This problem was under control for all other traces, but in Lecture Room apparently the value of n_0 was not sufficiently large. In real-world prediction situations, one could use a value for n_0 as large as desirable, and therefore the ill-conditioning problem can probably be completely avoided.

It is worth mentioning though that the approach developed in [5] is not designed to be adaptive. As a result, the comparison of the current study in essence does not say that the sparse basis selection method proposed here is better than the neural network approach of [5]. But rather, it says that an adaptive method appears to work better for this class of applications than a nonadaptive method. One must be aware though that adaptive approaches need special attention during design. The adaptation speed (represented by parameter α in the current algorithm) must be made to roughly match the speed of change of the underlying time-series model, otherwise model adaptation will be "out of sync". An adaptive approach (as opposed to a nonadaptive or a robust approach) carries with it the penalty of using shorter data windows and, hence, noisier estimates. So if the underlying time-series is not time-varying, there is no benefit in using an adaptive model. As it turns out, for the MPEG-coded video source traffic prediction problem, the adaptive approach is really needed and adds significant value.

V. Summary and Conclusion

In this paper, the problem of MPEG-coded video source traffic prediction problem is considered, and a new prediction model is developed. The proposed model is based on adapting the sparse basis selection methodology to the time-series prediction problem. The new algorithm is based on updating all matrices in a recursive fashion when a new data point is received, making the method truly adaptive. The method is applied on a number of MPEG4-coded video traces. The achieved results are improved as compared to another, nonadaptive approach which attests to the need for adaptive prediction methods for such applications.

Appendix
Proof of (7) [Similarly (24)]

Adding vector v to the chosen basis U results in

$$U' = [U \quad | \quad v]. \tag{57}$$

The old error (12) is given by

$$E(n) = y^T[R - RU(U^TRU)^{-1}U^TR]y. \tag{58}$$

The term $D(n)$ is augmented by a row and a column, to become [see (35)]

$$D'(n) = \begin{pmatrix} (U^TRU)^{-1} + \frac{(U^TRU)^{-1}U^TRvv^TRU(U^TRU)^{-1}}{c} & -\frac{(U^TRU)^{-1}U^TRv}{c} \\ -\frac{v^TRU(U^TRU)^{-1}}{c} & \frac{1}{c} \end{pmatrix}. \tag{59}$$

Using (57) and (59)

$$\begin{aligned} RU'(U'^TRU')^{-1}U'^TR = {} & RUD(n)U^TR \\ & - Rvv^TRUD(n)U^TR/c \\ & - RUD(n)U^TRvv^TR/c \\ & + RUD(n)U^TRvv^TRUD(n)U^TR/c \\ & + Rvv^TR/c \end{aligned} \tag{60}$$

which can be rearranged as

$$\begin{aligned} RU'(U'^TRU')^{-1}U'^TR = {} & RUD(n)U^TR + (RUD(n)U^TRv \\ & - Rv)(v^TRUD(n)U^TR - v^TR)/c. \end{aligned} \tag{61}$$

Substituting (61) and (34) into (58), and noting that $(\mathcal{P}_S^{\perp})^2 = \mathcal{P}_S^{\perp}$ results in the new error

$$\begin{aligned} E'(n) &= y^T[R - RU'(U'^TRU')^{-1}U'^TR]y \\ &= E(n) - \frac{|y^TRUD(n)U^TRv - y^TRv|^2}{v^TRv - v^TRUD(n)U^TRv} \end{aligned} \tag{62}$$

where the quotient in the RHS equals $p(v)$ in (24).

Acknowledgment

The authors would like to acknowledge the helpful comments of A. Bhattacharya.

References

[1] A. M. Adas, "Using adaptive linear prediction to support real-time VBR video under RCBR network service model," *IEEE/ACM Trans. Netw.*, vol. 6, no. 5, pp. 635–644, Oct. 1998.

[2] ——, "Supporting real-time VBR using dynamic reservation based on linear prediction," Georgia Inst. Technol., Atlanta, GA, Report GIT-CC-95/26, Aug. 1995.
[3] J. Adler, B. D. Rao, and K. Kreutz-Delgado, "Comparison of basis selection methods," in *Proc. 30th Asilomar Conf. Signals, Systems & Computers*, vol. 1, Pacific Grove, CA, Nov. 1996, pp. 252–257.
[4] J. J. Bae and T. Suda, "Survey of traffic control schemes and protocols in ATM networks," *Proc. IEEE*, vol. 79, no. 2, pp. 170–189, Feb. 1991.
[5] A. Bhattacharya, A. G. Parlos, and A. F. Atiya, "Prediction of MPEG-coded video source traffic using recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 51, no. 8, pp. 2177–2190, Aug. 2003.
[6] P. Bocheck and S. F. Chang, "A content based video traffic model using camera operations," in *Proc. IEEE Int. Conf. Image Processing*, Lausanne, Switzerland, Sep. 1996, pp. 817–820.
[7] P. R. Chang and J. T. Hu, "Optimal nonlinear adaptive prediction and modeling of MPEG video in ATM networks using pipelined recurrent neural networks," *IEEE J. Sel. Areas Commun.*, vol. 15, no. 8, pp. 1087–1100, Aug. 1997.
[8] S. Chen and D. Donoho, "Basis pursuit," in *Proc. 28th Asilomar Conf. Signals, Systems & Computers*, vol. 1, 1994, pp. 41–44.
[9] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 33–61, 1999.
[10] A. Chodorek and R. R. Chodorek, "An MPEG-2 video traffic prediction based on phase space analysis and its application to on-line dynamic bandwidth allocation," in *Proc. 2nd Eur. Conf. Universal Multiservice Networks*, Apr. 8–10, 2002, pp. 44–55.
[11] R. R. Coifman and M. V. Wickerhauser, "Entropy-based algorithms for best basis selection," *IEEE Trans. Inf. Theory*, vol. 38, no. 2, pp. 713–718, Mar. 1992.
[12] S. F. Cotter, M. N. Murthi, and B. D. Rao, "Fast basis selection methods," in *Proc. 31st Asilomar Conf. Signals, Systems & Computers*, vol. 2, 1997, pp. 1474–1478.
[13] S. Cotter, K. Kreutz-Delgado, and B. Rao, "Backward elimination for sparse vector subset selection," *Signal Process.*, vol. 81, pp. 1849–1864, 2001.
[14] C. Couvreur and Y. Bresler, "On the optimallity of the backward greedy algorithm for the subset selection problem," *SIAM J. Matrix Anal. Applicat.*, 1997.
[15] G. Davis, S. Mallat, and Z. Zhang, "Adaptive time-frequency decompositions," *Opt. Eng.*, vol. 33, no. 7, pp. 2183–2191, Jul. 1994.
[16] G. Davis, S. Mallat, and M. Avellaneda, "Greedy adaptive approximation," *J. Construct. Approx.*, vol. 13, pp. 57–98, 1997.
[17] D. Donoho and M. Elad, "Optimally sparse representation in general (nonorthogonal) dictionaries via l^1 minimization," *Proc. Nat. Acad. Sci.*, vol. 100, no. 10, pp. 2197–2202, Mar. 2003.
[18] A. D. Doulamis, N. D. Doulamis, and S. D. Kollias, "Nonlinear traffic modeling of VBR MPEG-2 video sources," in *Proc. IEEE Int. Conf. Multimedia Expo*, New York, Jul.–Aug. 2000, pp. 1318–1321.
[19] A. D. Doulamis, N. D. Doulamis, and S. D. Kollias, "Recursive nonlinear models for on line traffic prediction of VBR MPEG coded video sources," in *Proc. IEEE-INNS-ENNS Int. Joint Conf. Neural Networks IJCNN*, Como, Italy, Jul. 2000, pp. 114–119.
[20] A. D. Doulamis, N. D. Doulamis, and S. D. Kollias, "An adaptable neural network model for recursive nonlinear traffic prediction and modeling of MPEG video sources," *IEEE Trans. Neural Netw.*, vol. 14, no. 1, pp. 150–166, Jan. 2003.
[21] F. H. P. Fitzek and M. Reisslein, MPEG-4 and H:263 video traces for network performance evaluation, , Nov. 2003.
[22] Y. Freund and R. Shapire, "Experiments with a new boosting algorithm," in *Proc. 13th Int. Conf. Machine Learning*, San Francisco, CA, 1996, pp. 148–156.
[23] V. S. Frost and B. Melamed, "Traffic modeling for telecommunications networks," *IEEE Commun. Mag.*, vol. 32, no. 3, pp. 70–81, Mar. 1994.
[24] F. Girosi, "An equivalence between sparse approximation and support vector machines," *Neural Computat.*, vol. 10, no. 6, pp. 1455–1480, Aug. 1998.
[25] I. F. Gorodnitsky and B. D. Rao, "Sparse signal reconstruction from limited data using focuss: a re-weighted minimum norm algorithm," *IEEE Trans. Signal Process.*, vol. 45, no. 3, pp. 600–616, Mar. 1997.
[26] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003.
[27] G. Harikumar, C. Couvreur, and Y. Bresler, "Fast optimal and suboptimal algorithms for sparse solutions to linear inverse problems," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, vol. III, Seattle, WA, May 1998, pp. 1877–1880.
[28] D. P. Heyman and T. V. Lakshman, "Source models for VBR broadcast-video traffic," *IEEE/ACM Trans. Netw.*, vol. 4, no. 1, pp. 40–48, Feb. 1996.
[29] R. Horn and C. Johnson, *Matrix Analysis*. Cambridge, MA: Camridge Univ. Press, 1985.
[30] K. Kreutz-Delgado, J. Murray, B. Rao, K. Engan, T.-W. Lee, and T. Sejnowski, "Dictionary learning algorithms for sparse representation," *Neural Computat.*, vol. 15, no. 2, pp. 349–396, Feb. 2003.
[31] M. Krunz and H. Hughes, "A traffic model for MPEG-coded VBR streams," in *Proc. Joint Int. Conf. Measurement and Modeling of Computer Systems, ACM SIGMETRICS*, May 1995, pp. 47–55.
[32] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3397–3415, Dec. 1993.
[33] M. Nafie and A. Tewfik, "Deterministic and iterative solutions to subset selection problems," *IEEE Trans. Signal Process.*, vol. 50, no. 7, pp. 1591–1601, Jul. 2002.
[34] P. Nair, A. Choudhury, and A. Keane, "Some greedy learning algorithms for sparse regression and classification with Mercer kernels," *J. Mach. Learn. Res.*, vol. 3, pp. 781–801, 2002.
[35] B. K. Natarajan, "Sparse approximate solutions to linear system," *SIAM J. Comput.*, vol. 24, no. 2, pp. 227–234, Apr. 1995.
[36] Y.-H. Pao, *Adaptive Pattern Recognition and Neural Networks*. Reading, MA: Addison-Wesley, 1989.
[37] T. Poggio and F. Girosi, "A sparse representation for function approximation," *Neural Computat.*, vol. 10, no. 6, pp. 1445–1454, Aug. 1998.
[38] A. G. Parlos, K. T. Chong, and A. F. Atiya, "Application of the recurrent multilayer perceptron in modeling complex process dynamics," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 255–266, Mar. 1994.
[39] B. D. Rao and K. Kreutz-Delgado, "An affine scaling methodology for best basis selection," *IEEE Trans. Signal Process.*, vol. 47, no. 1, pp. 187–200, Jan. 1999.
[40] S. J. Reeves, "An efficient implementation of the backward greedy algorithm for sparse signal reconstruction," *IEEE Signal Process. Lett.*, vol. 6, no. 10, pp. 266–268, Oct. 1999.
[41] J. A. K. Suykens, L. Lukas, and J. Vandewalle, "Sparse approximation using least squares support vector machines," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS'00)*, Geneva, Switzerland, May 2000, pp. II757–II760.
[42] K. Wang, C.-H. Lee, and B. Juang, "Selective feature extraction via signal decomposition," *IEEE Signal Process. Lett.*, vol. 4, no. 1, pp. 8–11, Jan. 1997.
[43] S. J. Yoo, "Efficient traffic prediction scheme for real-time VBR MPEG video transmission over high-speed networks," *IEEE Trans. Broadcast.*, vol. 48, no. 1, pp. 10–18, Mar. 2002.

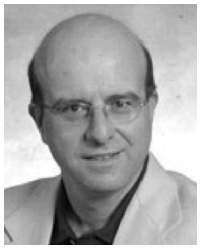
Amir F. Atiya (S'86–M'90–SM'97) was born in Cairo, Egypt, in 1960. He received the B.S. degree in electrical engineering from Cairo University, Cairo, Egypt, in 1982 and the M.S. and Ph.D. degrees in electrical engineering from Caltech, Pasadena, CA, in 1986 and 1991, respectively.

He is currently an Associate Professor in the Department of Computer Engineering, Cairo University. He held various appointments in industry, such as in QANTXX, Houston, TX; Simplex Technology, Hong Kong; Countrywide, California; and Dunn Capital Management, Florida. From 1997 to 2001, he was a Visiting Associate in Caltech. His research interests are in the areas of neural networks, learning theory, pattern recognition, Monte Carlo methods, data compression, and optimization theory. His most recent interests are the application of learning theory and computational methods to financial prediction and to communications networks prediction problems.

Dr. Atiya received the highly regarded Egyptian State Prize for Best Research in Science and Engineering in 1994. He also received the Young Investigator Award from the International Neural Network Society, in 1996. He has been an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS since 1998. He was a Guest Co-editor of the special issue of the IEEE TRANSACTIONS ON NEURAL NETWORKS on "Neural Networks in Financial Engineering," July 2001. He served on the organizing and program committees of several conferences, including being Program Co-Chair for the IEEE Conference on Computational Intelligence in Financial Engineering (CIFER-03), March 2003, Hong Kong.

Mohamed A. Aly was born in Cairo, Egypt in 1980. He received the B.S. degree (with honors) from the Department of Computer Engineering, Cairo University, Cairo, Egypt, in 2003. He is currently working toward the M.S. degree at the same university.

His research interests include machine learning, computer vision, robotics, computer networks, embedded systems, and signal processing.



Alexander G. Parlos (S'81–M'86–SM'92) received the B.S. degree in nuclear engineering from Texas A&M University, College Station, and the S.M. degree in nuclear engineering, the S.M. degree in mechanical engineering, and the Sc.D. degree in automatic control and systems engineering, all from the Massachusetts Institute of Technology (MIT), Cambridge.

He is currently a Professor of Mechanical Engineering at Texas A&M University, holding joint appointments with the Department of Nuclear Engineering, and, by courtesy, the Department of Electrical Engineering. He has established and is heading the Networked and Intelligent Machines Laboratory (NIML), where research on wireless sensor/actuator networks focuses on enhancing sensor/actuator performance and on machine-to-machine communication over standard IP networks. He holds numerous patents involving "sensorless" machine condition assessment technologies and algorithms for training neural networks for use in estimation and control—many of which have been licensed for commercial use. Since his arrival to Texas A&M University he has supervised the research of over 50 graduate students. He has published over 160 research papers. He is a member of the editorial board of the *Transactions on Control, Automation and Systems Engineering*.

Dr. Parlos is currently a member of the editorial board of the IEEE TRANSACTIONS ON NEURAL NETWORKS. He is also the recipient of multiple awards and honors.