Fast Orthogonal Least Squares Algorithm for Efficient Subset Model Selection

S. Chen and J. Wigger

***Abstract*—An efficient implementation of the orthogonal least squares algorithm for subset model selection is derived in this correspondence. Computational complexity of the algorithm is examined and the result shows that this new fast orthogonal least squares algorithm significantly reduces computational requirements.**

I. Introduction

A general nonlinear modelling approach is first to perform a fixed nonlinear expansion and then to combine the resulting terms linearly. This gives rise to the following regression model:

$$\mathbf{y} = \mathbf{X}\Theta + \mathbf{e} \tag{1}$$

where the size of the matrix $\mathbf{X}$ is $N \times M$, N is the number of data and M is the number of model regressors. Examples of this kind of model include the Volterra series model [1], the radial basis function network [2], the fuzzy basis function network [3], and the general functional-link network [4].

Because the model size M is usually excessively large, subset model selection is necessary. Optimal subset selection techniques are computationally prohibitive and impractical. A practical method is the forward selection, and the orthogonal least squares (OLS) algorithm [1], [2] is an efficient implementation of this subset selection procedure. In the case of $M \ll N$, computational requirements can further be reduced by employing a preprocessing scheme based on Gram–Schmidt orthogonalization procedure [5]. In this correspondence, we derive a fast implementation of the OLS algorithm for subset model selection, which results in significant reduction in computational complexity. We refer to this new version of the OLS algorithm as the fast OLS (FOLS) algorithm. This FOLS algorithm is much simpler than the schemes presented in [5].

II. The OLS Algorithm

We briefly summarize the OLS algorithm [1], [2]. This will enable us to analyze where saving in computation can be made. Let an orthogonal decomposition of $\mathbf{X}$ be $\mathbf{X} = \mathbf{WA}$. The model (1) can be rewritten as

$$\mathbf{y} = \mathbf{Wg} + \mathbf{e} \tag{2}$$

Because

$$\mathbf{y}^T\mathbf{y} = \sum_{j=1}^{M} g_j^2 \mathbf{w}_j^T\mathbf{w}_j + \mathbf{e}^T\mathbf{e} \tag{3}$$

where $\mathbf{w}_j$ are the columns of $\mathbf{W}$, the error reduction ratio due to $\mathbf{w}_p$ is given by

$$[err]_p = g_p^2\mathbf{w}_p^T\mathbf{w}_p/\mathbf{y}^T\mathbf{y} \tag{4}$$

Manuscript received September 12, 1994; revised January 17, 1995. The associate editor coordinating the review of this paper and approving it for publication was Prof. Ali N. Akansu.

The authors are with the Department of Electrical and Electronic Engineering, University of Portsmouth, Anglesea Building, Portsmouth PO1 3DJ, UK.

IEEE Log Number 9411987.

This error reduction ratio provides a criterion for forward subset selection.

At the beginning of the pth stage of the selection procedure, $\mathbf{X}$ has been transformed into $\mathbf{X}^{(p-1)} = [\mathbf{w}_1 \cdots \mathbf{w}_{p-1}\mathbf{x}_p^{(p-1)} \cdots \mathbf{x}_M^{(p-1)}]$ and $\mathbf{y}$ into $\mathbf{y}^{(p-1)}$. The pth stage consists of:

i) For $p \le j \le M$, compute

$$\left.\begin{aligned} g_p^{(j)} &= (\mathbf{x}_j^{(p-1)})^T\mathbf{y}^{(p-1)}/((\mathbf{x}_j^{(p-1)})^T\mathbf{x}_j^{(p-1)}) \\ [err]_p^{(j)} &= (g_p^{(j)})^2(\mathbf{x}_j^{(p-1)})^T\mathbf{x}_j^{(p-1)}/(\mathbf{y}^T\mathbf{y}) \end{aligned}\right\}. \tag{5}$$

ii) Find

$$[err]_p = [err]_p^{(j_p)} = \max\{[err]_p^{(j)}, p \le j \le M\}. \tag{6}$$

The j_pth column of $\mathbf{X}^{(p-1)}$ is interchanged with the pth column of $\mathbf{X}^{(p-1)}$, and the j_pth column of $\mathbf{A}$ is interchanged up to the $(p-1)$th row with the pth column of $\mathbf{A}$.

iii) Calculate the pth row of $\mathbf{A}$, and transform $\mathbf{X}^{(p-1)}$ and $\mathbf{y}^{(p-1)}$ into $\mathbf{X}^{(p)}$ and $\mathbf{y}^{(p)}$

$$\left.\begin{aligned} \mathbf{w}_p &= \mathbf{x}_p^{(p-1)} \\ \alpha_{p,j} &= \mathbf{w}_p^T\mathbf{x}_j^{(p-1)}/(\mathbf{w}_p^T\mathbf{w}_p), p+1 \le j \le M \\ \mathbf{x}_j^{(p)} &= \mathbf{x}_j^{(p-1)} - \alpha_{p,j}\mathbf{w}_p, p+1 \le j \le M \end{aligned}\right\} \tag{7}$$

$$\left.\begin{aligned} g_p &= \mathbf{w}_p^T\mathbf{y}^{(p-1)}/(\mathbf{w}_p^T\mathbf{w}_p) \\ \mathbf{y}^{(p)} &= \mathbf{y}^{(p-1)} - g_p\mathbf{w}_p \end{aligned}\right\}. \tag{8}$$

The selection is terminated at the M_sth stage when a preset tolerance is satisfied

$$1 - \sum_{p=1}^{M_s}[err]_p < \xi. \tag{9}$$

This produces a subset model containing M_s significant regressors. This procedure is suboptimal. However, it offers a realistic and efficient method for tackling the problem of huge model dimension encountered in nonlinear modelling. Subset models found using the forward-selection search are generally good enough for practical applications.

III. The Fast OLS Algorithm

It is clear that most of the computation in the above OLS algorithm is consumed in the calculation of inner products such as $(\mathbf{x}_j^{(p-1)})^T\mathbf{y}^{(p-1)}$. Instead of updating column vectors as in (7) and (8) and then computing inner products in the next stage, substantial saving in computation can be achieved by directly updating scalar inner products themselves. This can easily be done, for example

$$(\mathbf{x}_j^{(p)})^T\mathbf{x}_k^{(p)} = (\mathbf{x}_j^{(p-1)})^T\mathbf{x}_k^{(p-1)} - \alpha_{p,j}\alpha_{p,k}\mathbf{w}_p^T\mathbf{w}_p,$$
$$p+1 \le j \le M \text{and} j \le k \le M. \tag{10}$$

Define two matrices $\mathbf{B} = [\mathbf{X}|\mathbf{y}]^T[\mathbf{X}|\mathbf{y}]$ and $\mathbf{C} = [\mathbf{A}|\mathbf{g}]$. The elements of $\mathbf{B}$ and $\mathbf{C}$ are denoted by $b_{i,j}$ and $c_{i,j}$, respectively. Obviously, $b_{M+1,M+1} = \mathbf{y}^T\mathbf{y}$. We propose the following fast implementation of the OLS algorithm for subset model selection.

The pth stage of the selection procedure consists of:

i) For $p \le j \le M$, compute

$$\left.\begin{aligned} c_{p,M+1}^{(j)} &= b_{j,M+1}/b_{j,j} \\ [err]_p^{(j)} &= (c_{p,M+1}^{(j)})^2 b_{j,j}/b_{M+1,M+1} \end{aligned}\right\}. \tag{11}$$

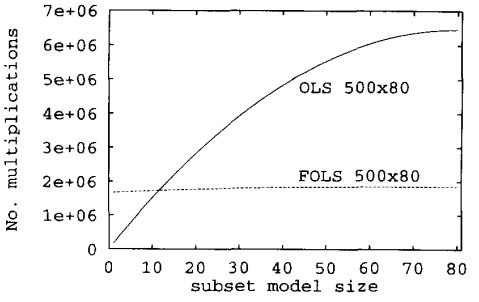

Fig. 1. Computational requirement for $\mathbf{X}$ matrix of size 500×80.

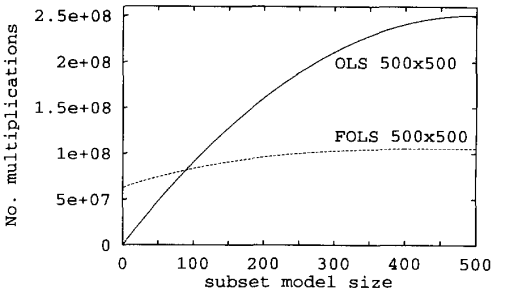

Fig. 2. Computational requirement for $\mathbf{X}$ matrix of size 500×500.

ii) Find

$$[err]_p = [err]_p^{(j_p)} = \max\{[err]_p^{(j)}, p \le j \le M\}. \tag{12}$$

The j_pth column of $\mathbf{B}$ is interchanged from the pth row upwards with the pth column of $\mathbf{B}$, and then the j_pth row of $\mathbf{B}$ is interchanged from the pth column upwards with the pth row of $\mathbf{B}$. The j_pth column of $\mathbf{C}$ is interchanged up to the $(p-1)$th row with the pth column of $\mathbf{C}$.

iii) For $p+1 \le j \le M+1$, compute

$$c_{p,j} = b_{p,j}/b_{p,p} \tag{13}$$

For $p+1 \le j \le M$ and $j \le k \le M+1$, compute

$$\left.\begin{aligned} b_{j,k} &= b_{j,k} - c_{p,j}c_{p,k}b_{p,p}, \\ b_{k,j} &= b_{j,k} \end{aligned}\right\}. \tag{14}$$

The selection is terminated at the M_s stage when the stopping criterion (9) is satisfied. As in the case of the OLS algorithm, a simple mechanism can be included to avoid ill-conditioning problem by comparing $b_{j,j}$ with a small energy threshold. If $b_{j,j}$ is smaller than this threshold, it will not be selected.

IV. Complexity Analysis

The number of multiplications required by the OLS algorithm to select a subset model of size M_s from the matrix $\mathbf{X}$ of size $N \times M$ is

$$\text{no. of multiplications (OLS)} = N - M_s(N+1) + \sum_{p=1}^{M_s}(4N+5)(M-p+1). \tag{15}$$

The number of multiplications required by the FOLS algorithm to perform the same subset model selection is

$$\text{no. of multiplications (FOLS)} = \frac{N(M+1)(M+2)}{2} - 3M_s + \sum_{p=1}^{M_s}(M-p+7)(M-p+1). \tag{16}$$

For the normal case of $M \le N$, the FOLS algorithm requires significantly less computation compared with the OLS algorithm. Only in the extreme case of $N \ll M$, does the original OLS algorithm have obvious advantages. Several examples for different sizes of $\mathbf{X}$ are illustrated in Figs. 1–4.

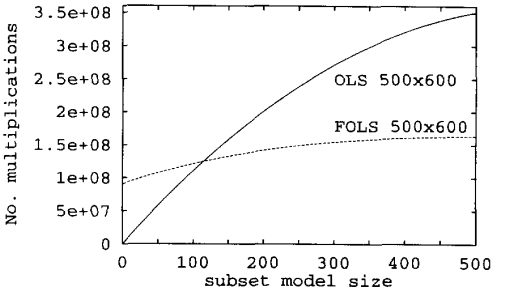

Fig. 3. Computational requirement for $\mathbf{X}$ matrix of size 500×600.

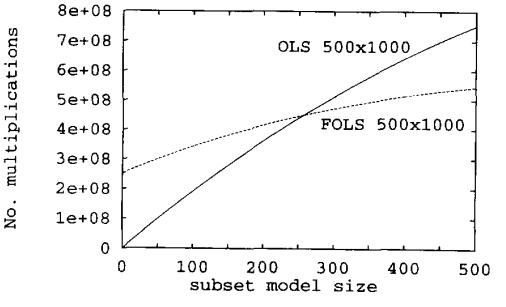

Fig. 4. Computational requirement for $\mathbf{X}$ matrix of size 500×1000.

V. Conclusions

A fast version of the orthogonal least squares algorithm has been presented, which offers significant reduction in computational complexity for forward selection of subset models. In the case that the number of model regressors is smaller than the number of data points, this algorithm has a further advantage of saving in memory requirement.

References

[1] S. Chen, S. A. Billings, and W. Luo, "Orthogonal least squares methods and their application to non-linear system identification," *Int. J. Control*, vol. 50, no. 5, pp. 1873–1896, 1989.

[2] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Networks*, vol. 2, no. 2, pp. 302–309, 1991.

[3] L. X. Wang and J. M. Mendel, "Fuzzy basis functions, universal approximation, and orthogonal least-squares learning," *IEEE Trans. Neural Networks*, vol. 3, no. 5, pp. 807–814, 1992.

[4] Y. H. Pao, *Adaptive Pattern Recognition and Neural Networks*. Reading, MA.: Addition-Wesley, 1989.

[5] E. S. Chng, S. Chen, and B. Mulgrew, "Efficient computational schemes for the orthogonal least squares algorithm," *IEEE Trans. Signal Processing*, vol. 43, no. 1, pp. 373–376, 1994.

Adaptive Cone-Kernel Time-Frequency Analysis

Richard N. Czerwinski and Douglas L. Jones

Abstract—**We present a technique to adaptively optimize the performance of the cone-kernel distribution (CKD) by varying the cone length in response to changing signal properties. The result is an adaptive CKD that preserves the outer hull of signal time support, yields excellent results on real signals, outperforms fixed-length CKD's, and requires only slightly more computation than a fixed-kernel distribution.**

I. Introduction

Time-frequency analysis [1] is used in many fields to study signals with time-varying spectral content. Recent work [2]–[6] has illustrated the benefits of *signal-dependent* time-frequency analysis, in which the algorithm is allowed to adapt in response to the signal to optimize performance. Unfortunately, many adaptive techniques involve complicated and expensive optimization procedures whose cost may overshadow the benefits of adaptation. These methods can also represent overkill in applications where one or two degrees of freedom are sufficient for near-optimal results.

In this correspondence, we develop an approach to adapting the popular cone-kernel distribution (CKD) that is simple, fast, versatile, and offers high performance. We introduce a technique for adapting a single important parameter of the distribution, and derive a fast algorithm to efficiently implement this adaptation. Finally, we apply the adaptive CKD to synthetic and naturally occurring test data to demonstrate its capability. On real data sets, we have often found that this algorithm gives better results than any other technique known to us.

The constraint that defines the CKD was first discussed in Claasen and Mecklenbräuker [7], and has more recently been developed into a sound, general purpose distribution by Zhao *et al.* [8], Loughlin *et al.* [9], and Oh and Marks [10]. Because of its excellent resolution of abrupt signal transitions, the CKD has gained widespread support.

The CKD is defined as

$$CKD(n,f) = \sum_k \phi_T(k) R(n,k) e^{-j2\pi kf} \tag{1}$$

where

$$R(n,k) = \sum_{p=-|k|}^{|k|} x(n+p+k)\, x^*(n+p-k) \tag{2}$$

Manuscript received August 28, 1993; revised January 1, 1995. This work was supported by the National Science Foundation under Grant no. MIP 90-12747, and the Joint Services Electronics Program under Grant no. N00014-90-J-1270. The associate editor coordinating the review of this paper and approving it for publication was Dr. Patrick Flandrin.

The authors are with Coordinated Science Laboratory, Urbana, IL 61801 USA.

IEEE Log Number 9412003.

and the kernel $\phi_T(k)$ has the form

$$\phi_T(k) = e^{-\beta k^2/T^2} \text{ if } -T \le |k| \le T. \tag{3}$$

The β parameter controls the rate of tapering in the kernel; the kernel used by Zhao *et al.* [8], corresponds to $\beta \approx 4.6$. The properties of the CKD can be altered by changing the cone length T. Impulse-like transient signal components are best analyzed by a very short cone, while longer duration components call for a longer cone length. Since any given signal may contain components of both short and long duration, it is desirable to select the cone length adaptively in response to changing signal structure.

II. A Technique For Adapting the Cone Length

Every bilinear TFD is affected to some degree by oscillating cross-terms that appear midway between every two auto-components in the time-frequency plane. The cross-terms contribute to certain theoretical properties of the TFD, but they can also obscure true signal features in the time-frequency plane. Kernel design has traditionally been a matter of imposing constraints on the kernel to trade off cross-term suppression and desirable properties.

Since auto-components are centered at the origin of the θ-τ (ambiguity) plane, and cross-components are located away from the origin, a lowpass kernel is needed to suppress the cross-terms [3], [9]. A CKD of finite τ-extent is inherently lowpass, so it suffices to match the length of the cone to the extent of the auto-component energy. The benefits of matching the kernel to the signal are well known in high-resolution time-frequency analysis [3], [11]; the lowpass cone kernel can only effectively match the lowpass auto-components. We propose a procedure that approximately optimizes the match between the kernel and the auto-components while rejecting highpass cross-components, thus producing a desirable TFD.

To select the optimal cone length at each point in time, we consider a family of normalized-energy cone kernels of different lengths and choose the one whose pointwise product with a locally defined short-time ambiguity function has the greatest energy. Equivalently, this can be formulated as a problem of selecting the normalized cone kernel that produces the TFD of maximum energy. Using a manipulation similar to that in [12], we write the TFD energy at each time as a function of T, the cone length, which is described in (4) through (6) at the top of the next page. The parameters $M_1(n,k)$ and $M_2(n,k)$ control the length and shape of the analysis interval, the signal values that are allowed to affect the optimization procedure. Ideally, $M_1(n,k)$ and $M_2(n,k)$ should vary in time so that no magnitude-squared terms from outside the local signal component come under consideration. In this discussion, however, we fix $M_1(n,k) = -N/2$ and $M_2(n,k) = N/2 - 1$ for a total of N samples, where N is the FFT length. In practice, one should be very careful in selecting the analysis interval, since a large interval can lead to the cone length being adapted to nonlocal signal characteristics.

In adaptively optimizing the CKD, we seek the value of T maximizing (6) at each time. That expression has the form of an inner product between a window function (squared), and a bracketed term strictly independent of the cone length, T. Denoting the bracketed quantity by E_n, we compute the optimal T by

$$T_{opt}^n = \arg\max_T \left\langle A_T e^{-\beta k^2/T^2} \operatorname{rect}\left(\frac{k}{T}\right), E_n(k) \right\rangle \tag{7}$$

where A_T is a normalization constant and $\langle \cdot, \cdot \rangle$ denotes the inner product. The normalization constant A_T is the reciprocal of the energy of the kernel of length T, and is included in (7) as a penalty against increasing the cone length; we are effectively choosing as