DATA-DEPENDENT PARTIAL UPDATE ADAPTIVE ALGORITHMS FOR LINEAR AND NONLINEAR SYSTEMS

Tyseer Aboulnasr and Qiongfeng Pan

School of Information Technology and Engineering
Faculty of Engineering, University of Ottawa
Ottawa, Ontario, Canada, K1N 6N5
aboulnasr@eng.uottawa.ca, qpan@site.uottawa.ca

ABSTRACT

In this paper, we will review partial update adaptive algorithms with special emphasis on data-dependant algorithms. We then demonstrate that the same approach applied in the MMax LMS partial update algorithm for linear adaptive filters [4] can be extended to the class of nonlinear filters known as Volterra filters. The impact of the fact that the input vector is no longer a set of delayed input values on the complexity reduction due to the partial update is noted. Simulation results show that, as for linear filters, considerable saving is possible with little deterioration in performance.

1. INTRODUCTION

Adaptive filters have been used routinely to model unknown, possibly time-varying systems. In many cases, the number of parameters used is prohibitive limiting the practical application of powerful algorithms because of the complexity of updating a large number of coefficients at the same time. Partial update algorithms attempt to address this issue by limiting the number of coefficients being updated in a given iteration. The selection of which coefficients to update is critical in determining the performance of the resulting algorithm. Initially, this selection was done on a preset, rotating basis [1]. These algorithms are simple to implement but invariably lead to significantly lower convergence rates given the arbitrary nature of choosing the subset of coefficients to update.

In this paper, we will concentrate on algorithms that select the subset of coefficients to be updated based on some criterion so as to reduce the performance deterioration due to slower update of coefficients. This implies a dynamic determination of the coefficients to update and allows for selecting these coefficients to minimize the impact of not updating the full set of system parameters.

In Section 2, we review the fundamentals of partial update algorithms. In Section 3, we consider algorithms where the set of coefficients to be updated is dynamically determined every iteration based on the received data. In Section 4, we highlight some of the variants of the data-dependent partial update algorithms. Section 5 presents possible extensions of the partial update concept to nonlinear Volterra filters with preliminary results confirming good performance for partial update.

2. FUNDAMENTAL PARTIAL UPDATE ALGORITHMS

Consider the standard adaptive filter set-up where x(n) is the input, y(n) is the output and d(n) is the desired output, all at instant n. The output error e(n) is given by

$$e(n) = d(n) - y(n) = d(n) - \mathbf{w}^T(n) \cdot \mathbf{x}(n)$$

where $\mathbf{w}(n)$ is the N*1 column vector of the filter coefficients and $\mathbf{x}(n)$ is the N*1 column vector of the current and past inputs to the filter, both at instant n. The i^{th} element of $\mathbf{w}(n)$ is $w_i(n)$ and it multiplies the i^{th} delayed input x(n-i), i =0,.. N-1.

The basic NLMS algorithm is known for its extreme simplicity providing for coefficient update as given by:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\, e(n)\, \mathbf{x}(n)/\|\mathbf{x}(n)\|^2$$

where μ is the step size determining the speed of convergence and the steady state error. The complexity of implementing such an adaptive filter is effectively 2N multiply/add with N operations needed for the update of the N coefficients and another N operations needed for the calculation of the output y(n).

The basic idea of partial update adaptive filtering is to allow for the use of filters with a number of coefficients N large enough to model the unknown system while reducing the overall complexity by updating only L coefficients at a time. This results in considerable savings for L<<N. Invariably, there are penalties for this partial update, the most obvious of which being reduced convergence rate. The question then becomes which

coefficients should we update and how do we minimize the impact of the partial update on the overall filter performance.

Early attempts at partial update of the coefficients simply divided the coefficients into sets that were selected either sequentially by dividing the coefficient vector into blocks of length L and updating one block every iteration in a sequential form or by updating every L^{th} coefficient, again in order [1]. There is minimal additional overhead in implementing this selective update and the savings are proportional to the ratio of N/L. Since each set of coefficients will be updated every N/L iterations, the more the savings, the lower the performance of the algorithm. It is inevitable that the performance deteriorates considerably since the available information about the system dynamics are not used at all in identifying which coefficients can result in the most error reduction and as such need to be updated.

In [2], it was proposed to update the blocks in a random order (as opposed to sequentially). It was shown that while this setup has performance comparable to the periodic partial update, it does result in faster convergence for some deterministic signals in the context of adaptive beamforming.

3. DATA-DEPENDANT PARTIAL UPDATE ALGORITHMS

While the above algorithms reduce the complexity, the price paid in convergence rate may not be tolerated, particularly for LMS algorithms in acoustic environments where the convergence speed is not fast to start with. This led to other approaches where the set of coefficients to be updated is not predetermined, rather is selected to maximize the performance of the system in some sense.

In [3], the max-NLMS presented an algorithm were only one coefficient is updated in every iteration (using a slightly modified update equation). This coefficient is selected as the one multiplying the input with the largest absolute value. While this algorithm provided considerable saving in complexity, it was shown to diverge for some data sets.

In [4], the set of M coefficients to be updated is selected as the one that provides the maximum reduction in error. It is shown that this criterion reduces to the set of coefficients multiplying inputs x(n-i) with the largest magnitude using the standard NLMS update equation. An analysis of the mean square error convergence is provided in [4] based on matrix formulation of data-dependent partial updates. Based on the analysis, it was shown that the MMax algorithm provides the closest performance to the full update case for any given number of coefficients to be updated. This was confirmed in [5].

Theoretically, the determination of this set of coefficients needs to be done every iteration through a sorting algorithm. However, the complexity is not significant given the fact that the input vector $\mathbf{x}(n)$ is a time series. Once the full set of input samples is sorted as the samples arrive one after the other, a new iteration results in dropping the oldest sample and deciding where to insert the newest sample in the already-sorted set.

In [6], it was proposed that a "short-sort" algorithm be used to further reduce the overhead of the M-Max algorithm needed for sorting. The impulse response is divided into two regions. For the first region where the bulk of the energy of the response exists, all coefficients are updated in each iteration. For the second region where the coefficients are likely to be very small, the M-Max partial update algorithm is used. Given the significantly smaller size of this set of coefficients, sorting overhead is reduced.

In [13], the partial update algorithm is formulated as a constrained optimization problem along the lines of the NLMS leading to a common framework incorporating several existing algorithms.

4. VARIATIONS OF DATA-DEPEMNDENT PARTIAL UPDATE ALGORITHMS

Even when the set of coefficients to be updated is predetermined, improved performance with reduced coefficient update can still be achieved if prior information regarding the nature of the response is utilized. For example, in [7] the system response is decomposed into two stages. The first stage representing an arbitrary main response receives full update. The second stage is an up-sampled adaptive filter, where nonzero coefficients are updated every iteration, followed by a fixed lowpass filter to perform the interpolation between the samples. This can be seen as a variant on the concept of partial update.

In [8], prior knowledge of the fact that the system response is sparse with large non-zero samples concentrated in the same region was used to speed up initial convergence by weighting the input vector with an estimate of the channel response. The coefficients are all updated initially to enable the system to differentiate between large and small values. Following this initial convergence, large coefficients are updated every iteration to speed up their convergence while small coefficients (who will likely stay small) are updated based on some partial update algorithm. Finally, once convergence is

achieved, partial update is used for all coefficients. It was shown that the performance of this selective update algorithm is roughly equivalent to that of the full update at reduced complexity.

The concept of partial update has also been applied to domains other than the time domain and to algorithms other than the LMS. In [9], the M-Max algorithm was used in a decomposed transform domain to reduce the overall complexity showing very good performance. In [10], it was used in the DCT domain resulting in performance comparable to the full update case. In [11], selective update was proposed in the subband domain showing strong performance with speech signals while [12] and [13] successfully applied partial update to the Affine Projection algorithm.

In [5], the savings of partial update of LMS were integrated in set membership filters where complexity is reduced by allowing coefficients to vary within a feasible set providing further reduction in complexity at minimal deterioration in performance.

5. PARTIAL UPDATE ALGORITHMS FOR NONLINEAR VOLTERRA FILTERS

Volterra filters provide a mathematically tractable model for nonlinear systems to which much of the literature developed for linear systems has been extended [14]. The output is expressed in polynomial form as the sum of a linear component and higher order products of the input.

Consider the output of a third order Volterra system given by:

$$y(n)=\sum_{k=0}^{N_1-1}h_1(k;n)x(n-k)+\sum_{k_1=0}^{N_2-1}\sum_{k_2=0}^{N_2-1-k_1}h_2(k_1,k_2;n)x(n-k_1)x(n-k_2)$$
$$+\sum_{k_1=0}^{N_3-1}\sum_{k_2=0}^{N_3-1-k_1}\sum_{k_3=0}^{N_3-1-k_1-k_2}h_3(k_1,k_2,k_3;n)x(n-k_1)x(n-k_2)x(n-k_3)$$

where N_1 is the filter length for the linear part, N_2 is the memory depth for the second-order part and N_3 is the memory depth for the third-order part; $h_1(k,n), h_2(k_1,k_2;n)$ and $h_3(k_1,k_2,k_3;n)$ are the linear, second-order and third-order coefficients of the adaptive filter at time n respectively. There are N_1 coefficients $h_1(k,n)$, $N_2(N_2+1)/2$ coefficients $h_2(k_1,k_2;n)$ and $N_3(N_3+1)(N_3+2)/6$ coefficients $h_3(k_1,k_2,k_3;n)$. In the following, we represent $h_1(k,n)$, $h_2(k_1,k_2;n)$ and $h_3(k_1,k_2,k_3;n)$ by the vectors

$$H_1(n)=\left[h_1(0;n),\cdots,h_1(N_1-1;n)\right];$$
$$H_2(n)=\left[h_2(0;n),\cdots,h_2\left(N_2(N_2+1)/2-1;n\right)\right]$$
$$H_3(n)=\left[h_3(0;n),\cdots,h_3\left(N_3(N_3+1)(N_3+2)/6-1;n\right)\right].$$

The LMS algorithm for the Volterra filter is described as follows.

Coefficient vector: $[H_1(n);H_2(n);H_3(n)]$

Input vector:

$$X_1(n)=\left[x(n),x(n-1),\cdots,x(n-N_1+1)\right]$$
$$X_2(n)=\left[x^2(n),x(n)x(n-1),\cdots,x^2(n-N_2+1)\right]$$
$$X_3(n)=\left[x^3(n),x^2(n)x(n-1),\cdots,x^3(n-N_3+1)\right]$$

Initialization: Arbitrarily choose $H_1(n);H_2(n);H_3(n)$

Main iteration:

$$e(n)=d(n)-\sum_{i=1}^{3}H_i(n)X_i^T(n)$$
$$H_i(n+1)=H_i(n)+\mu_i e(n)X_i(n);\quad i=1,2,3$$

In this section we consider the extension of the MMax algorithm [4] to the class of Volterra filters. Thus, only the coefficients multiplying the largest P% input values will be updated, where P=L/N*100%. It should be noted that in this case, "input values" refers to the elements of the vector $[\boldsymbol{X}] = [X_1, X_2, X_3]$. The main challenge compared to the linear case is that the input vector is no longer a set of shifted values. As such, to determine the P% largest values, we will need a full sorting of the elements of $\boldsymbol{X}$ in every iteration. To reduce this additional complexity, sorted time series of sub-lists X_1, X_2, X_3 are maintained separately through merging of even smaller sorted lists. To avoid resorting the whole list, there are two alternatives to selecting the coefficient set to update. In the first approach (option 1), these sorted sub-lists are merged in every iteration into one large sorted set (at some complexity lower than a full resort) and the coefficients multiplying the P% largest values are updated. In option 2, the P% largest coefficients of *every* sorted sub-list X_1, X_2, X_3 are updated in every iteration. Detailed complexity and performance analysis for partial update Volterra filters is being conducted.

Simulation Results

In our simulation, we set N_1=10, N_2=4 and N_3=3. The coefficients for the unknown system are set as follows.

$$H_1=[0.85,0.8,0.9,0.7,0.6,0.75,0.65,0.8,0.6,0.8]$$
$$H_2=[0.5,0.3,0.2,0.2,0.3,0.1,0.15,0.25,0.05,0.25]$$
$$H_3=[0.1,0.3,0.3,0.2,0.4,0.3,0.2,0.4,0.2,0.1]$$

For each option, we compare the system performance for P=100%, 70% or 50%. Simulation results are given in Figures [1-3].

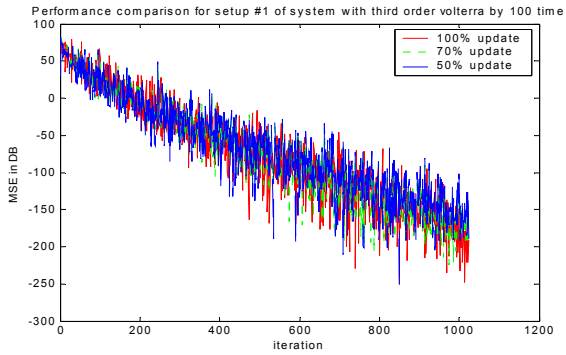

Fig 1: System performance for option 1 with 100%, 70% and 50% coefficient update

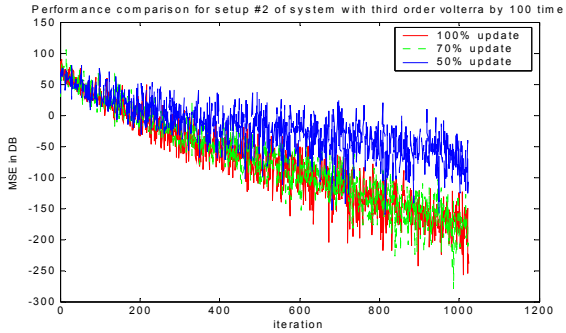

Fig 2: System performance for option 2 with 100%, 70% and 50% coefficient update

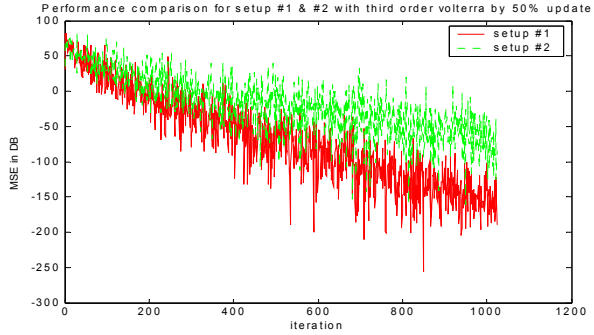

Fig 3: System performance for 50% coefficient update for options 1 and 2.

It is clear that option 1 provides the best performance with minimal deterioration even for a 50% coefficient update (at the cost of additional complexity for merging the sub-lists). However, the deterioration for option 2 is not significant as can be seen from Figure 3.

References:

[1] S. Douglas, "Adaptive filters Employing Partial Updates," IEEE Trans. on Circuits and Systems, CAS-II, Vol. 44, No 3., pp. 209-216, March 1997.

[2] M. Godavarti and A. Hero, "Stochastic partial update LMS algorithm for adaptive arrays," Proc. of the IEEE Sensor Array and Multichannel Signal Processing Workshop, pp.322-326, 2000.

[3] S.C. Douglas, " A family of normalized LMS algorithms," IEEE Signal Processing Letters, vol. 1, no. 3, pp.49-51, March 1994.

[4] T. Aboulnasr and K. Mayyas, "Complexity reduction of the NLMS algorithm via selective coefficient update," IEEE Transactions on Signal Processing, vol. 47, no. 5, pp. 1421-1424, May 1999.

[5] S. Werner, M. L.R. de Campos and Paulo Diniz, "Partial update NLMS algorithms with data-selective partial updating," IEEE Transactions on Signal Processing, vol. 52, no. 4, pp. 938-949, April 2004.

[6] P. Naylor and W. Sherliker, "A short-sort M-Max NLMS partial update adaptive filter with applications to echo cancellation," Proceedings of Int. Conf. on Acoustics, Speech and Signal Processing, ICASSP 2003, Vol. 5, pp. 373-376.

[7] A. Abousaada, T. Aboulnasr, W. Steenaart, "An echo tail canceler based on adaptive interpolated FIR filtering," IEEE Transactions on Circuits and Systems, vol. CAS-41, pp. 409-415, July 1992.

[8] H. Deng and M. Doroslovački, "New sparse adaptive filters with partial update," Proceedings of Int. Conf. on Acoustics, Speech and Signal Processing, ICASSP 2004, vol. 2, pp. 845-848.

[9] K. Mayyas and T. Aboulnasr, "Reduced complexity Transform-domain adaptive algorithm with selective coefficient update," IEEE Transactions on Circuits and Systems-II, vol. 51, no 1, pp. 136-142, March 2004 .

[10] S. Attallah and S.W. Liaw, "Analysis of DCTLMS algorithm with a selective coefficient updating," IEEE Transactions on circuits and Systems-II, Vol 48, No.6, pp.628-632, June 2001.

[11] K. Doğançay and O. Tanrikulu, "Generalized subband decomposition LMS algorithm employing selective partial updates," Proc. of the IEEE Conf. on Acoustics, Speech and Signal Processing, ICASSP 2002, vol. 2, pp 1377-1380.

[12] S. Werner, J. Apolinário and M.L.R. de Campos, "The data-selective constrained Affine Projection algorithm," Proc. of the IEEE Conf. on Acoustics, Speech and Signal Processing, ICASSP 2001, vol. 6, pp. 3745-3748.

[13] K. Doğançay and O. Tanrikulu, "Adaptive filtering algorithms with selective partial updates," IEEE Transactions on Circuits And Systems-II, vol. 48, no. 8, pp. 762-769, August 2001.

[14] V.J. Mathews and G.L. Sicuranza, *Polynomial Signal Processing,* John Wiley &Sons, New York, 2001.