NEW SPARSE ADAPTIVE ALGORITHMS USING PARTIAL UPDATE

Hongyang Deng and Miloš Doroslovački

Department of Electrical and Computer Engineering
The George Washington University
801 22nd St. NW
Washington DC, 20052
denghy@gwu.edu

ABSTRACT

In this paper, we propose two new sparse adaptive filtering algorithms using partial update. By taking advantage of both impulse response sparseness and partial update, we design different criteria to determine which coefficients to be updated in order to improve the performance of typical partial update algorithms. Compared with the Normalized Least Mean Square (NLMS), Selective Partial Update NLMS (SPUNLMS) and Proportionate NLMS (PNLMS++) algorithm, the proposed Partial update Sparse NLMS (PSNLMS) algorithms achieve faster convergence speed with even less computational complexity. Simulation results show that they perform well in applications where identification of long sparse impulse responses is needed. Network echo cancellation is a typical example.

1. INTRODUCTION

Network echo cancellation has been an active research field for several decades. Recently, this problem is facing more challenges because of the development of wide-area-network centric applications such as Voice over IP (VOIP). The increased transmission delay of the network and complexity of the application programs introduce more delay in the voice path, resulting in a longer echo return time. To cancel the echo effectively, we have to use longer adaptive filters to model the actual echo path, which will increase the computational complexity and decrease greatly the convergence speed of adaptive algorithms.

Most popular adaptive algorithms for dealing with this problem are Normalized Least Mean Square (NLMS) algorithms due to their simplicity and robustness. Since NLMS algorithm has slow convergence speed for color inputs and long adaptive filters can have large misadjustment, low convergence speed and high computational demand, several variants of this algorithm has been proposed to improve its performance [1]-[6]. Partial update algorithm [2]-[4] is an effective way to lower the computational complexity by only updating some coefficients at each sample period.

A very important characteristic of the echo path we want to take advantage of to boost the performance of the adaptive filtering algorithm is sparseness. If we can locate the big coefficients and only update these coefficients, we actually adapt a much shorter filter, thus increasing the convergence speed and decreasing the computational complexity. Unfortunately, in general we don't have this information a priori, so we have to search for the big coefficients. Well-known algorithms that take advantage of the sparseness of the echo path are the so-called Proportionate NLMS (PNLMS) [5] and its improved version, PNLMS++ [6].

In this paper, we propose two algorithms that combine the advantage of the sparseness of impulse response and the partial update.

2. PARTIAL UPDATE NLMS ALGORITHMS

Several Researchers have recently proposed partial update adaptive filtering algorithms [2-4]. The basic idea is to partially update only a portion of adaptive filter coefficients instead of updating all of them, and in that way to save computations while still maintaining acceptable convergence performance.

Generally, there are two ways to perform partial update. One is proposed in [2]. It chooses the coefficients to be updated by a predefined scheduling scheme, either periodically or sequentially. This type of algorithm is very simple and has nearly no computational overhead, but suffers from slow convergence because now we need more iteration to achieve the same number of updates for certain coefficients. Note that the algorithm doesn't use any information from the underlying environment, what makes it inflexible.

The other way to do the partial update is investigated in [3] and [4] independently. They choose the coefficients corresponding to large input value to be updated and

derive the algorithm based on the same principle as NLMS. Theoretical and simulation results show that the Selective Partial Update NLMS (SPUNLMS) algorithm achieves very close performance to the fully updated NLMS, with small computational overhead of sorting.

One of the reasons why SPUNLMS outperforms the algorithms that use a predefined scheduling scheme for updating is that it takes advantage of some information from the environment, i.e. the input. From the update equation, we can see that the magnitude of update is directly proportional to the input amplitude, so it's natural to choose only those coefficients corresponding to the significant input values to be updated and leave others untouched.

But we can make a step further. Besides the input, the system we try to model is another source of information we may use to improve the performance, especially for some highly structured systems, for example, sparse echo paths. An algorithm utilizing the coefficient amplitudes is proposed in [7]. The objective is not to improve the convergence but to implement efficient sorting.

Now let us consider how the sparseness of the adaptive filter can help us choose appropriate coefficients to update in order to improve the convergence speed. By observing the convergence process, we can see that each coefficient, from an initial position, usually set for 0, will converge to its optimal value along a route. The number of iterations required for each coefficient to reach its optimal value (more accurately speaking, the number of iterations needed to approach the ε-vicinity of the optimal value, where ε is a small number) is different, depending on the amplitude of that coefficient's optimal value. The bigger the value, the more the iterations it will take to converge. Small coefficients with optimal value zero (or nearly zero) will fluctuate around their optimal values from the very beginning, if we use the zero initial value, and the input is white.

In sparse impulse responses, big coefficients are a small portion of the total number of the coefficients. Therefore, the number of iterations needed for convergence mainly depends on the big coefficients. The SPUNLMS gives all the coefficients equal chance to be updated, thus, waste a lot of time on updating the small coefficients, which are already near the optimal values after a very short initial convergence period. To improve the performance, we need to find a way to give the big coefficients more chance to be updated, especially during the initial transient period.

3. ALGORITHM DESCRIPTION

Based on the analysis of the previous section, we propose two algorithms to improve the convergence performance of the partial update algorithm using the sparseness property of the system we are trying to model.

3.1. SPNLMS-I

The first proposed algorithm, Sparse Partial update NLMS Type I (SPNLMS-I), divides the whole adaptation period to three regimes. The first one is a very short initial stage, which uses SPUNLMS to update the coefficients. After this stage, all the small coefficients are converged to the vicinity of their optimal values. For the big coefficients, their values will become significantly large so that they can be distinguished from the small coefficients.

Now we enter the second regime by updating big coefficients every sample and still giving the small coefficients chance to be updated based on input value, but much less frequently than big coefficients. To distinguish big coefficients, we have to search for the location of the big coefficients, and for small coefficients, we have to search for big input, therefore, using a function of both the input and coefficient value seems to be a good criterion to choose which coefficients to be updated. For simplicity, we use the output amplitude value of each coefficient tap as a criterion to determine which coefficients are to be updated within the sampling period. We will only update M out of L coefficients where the M coefficients correspond to the M biggest outputs. The algorithm is

$$w_{i,k+1} = \begin{cases} w_{i,k} + \dfrac{\mu}{\|X_s\|^2} e_k x_{k-i} & \text{if } i \text{ corresponds to one of the } M \\ & \text{biggest } \left|x_{k-i} w_{i,k}\right|, i = 0,1,...,N-1 \\ w_{i,k} & \text{else} \end{cases} \quad (1)$$

Where $w_{i,k}$ is a coefficient, x_k is the input, e_k is the output error that is defined as $e_k = d_k - \sum_{i=0}^{N-1} w_{i,k} x_{k-i}$ with d_k being the desired output, μ is the adaptation step size parameter, i is the coefficient index, k is the time index and N is the total number of the coefficients. X_s is a vector of input values corresponding to the M biggest output values.

After this period, all the coefficients (big and small) are close to their optimal values. Now there is no need to emphasize the big coefficients because now every coefficient is fluctuating around its optimal value with very small changes. Therefore, we switch back to SPUNLMS to give each coefficient equal chance to be updated based on the input. If we continued to concentrate on big coefficients, we would have big steady state error and a slowed down final convergence.

3.2. SPNLMS-II

The SPNLMS-I has very good performance, especially in the second regime, it converges much faster than the fully

updated NLMS. But we have to design a mechanism to monitor the output of the system and to switch from one adaptation regime to the other. An alternative to this control mechanism is the SPNLMS-II algorithm.

For SPNLMS-II, we partially update the coefficients, but use two alternating criteria to determine which coefficients are to be updated. For every T sample, we use the input value as a criterion, thus the update equation is

$$w_{i,k+1} = \begin{cases} w_{i,k} + \dfrac{\mu}{\|X_s\|^2} e_k x_{k-i} & \text{if } i \text{ corresponds to one of the } M \\ & \text{biggest } |x_{k-i}|,\ i = 0,1,...,N-1 \\ w_{i,k} & \text{else.} \end{cases} \quad (2)$$

In this case, X_s is a vector of input values corresponding to the M biggest input values.

For other samples, we switch to use output value of each coefficient tap as the criterion, therefore (1) is applied. If we want to give the big coefficients more chances to be updated, usually we choose bigger T.

By using these alternating criteria, we get a compromise between giving all the coefficients equal chance to be updated based on SPUNLMS (which leads to good convergence to small coefficients) and emphasizing on updating the big coefficients more frequently to speed up the overall convergence.

3.3. Computational Complexity Analysis

Since we need the tap outputs to calculate the output error, there is no overhead if we choose the outputs as the criterion to determine which coefficients to be updated. For fully updated NLMS, we need $3L$ multiplications, L divisions and L additions. For PSNLMS, we only need to update M out of L coefficients, thus we can save $3(L-M)$ multiplications, $L-M$ divisions and $L-M$ additions. The only down side is that in order to find out the M largest outputs or inputs, we have to sort the output or input values. If we choose fast sorting algorithm [8], only $2\log_2(L)+2$ comparisons will be added. For large L and small M, which is appropriate for sparse impulse response, big computational savings are expected. And for implementing the sorting algorithm, small extra memory, which is proportional to the number of coefficients we update, is needed.

We can see that both proposed algorithms have the same computational complexity as SPUNLMS except for SPNLMS-I, we have to add some control operation to switch between the different adaptation regimes.

4. SIMULATION RESULTS

To evaluate the performance of the proposed algorithms, we used an adaptive filter to identify an echo path with 512 taps (64ms for sampling frequency of $8KHz$). Its impulse response is shown in Figure 1. It is clear that the active part only occupies a small portion of the filter length (around 8 ms).

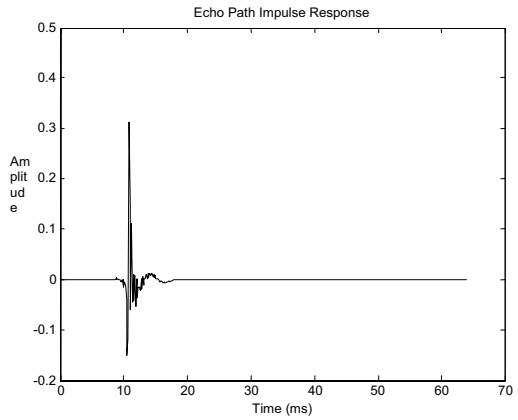

Fig.1 Impulse response of the network echo path

We used both white Gaussian input with unit variance and color input to test the algorithm. We generated the color input by passing the white Gaussian noise through a low pass filter with one pole at 0.9.

In the first example, we compared the learning curves of PSNLMS-I, NLMS, PNLMS++ and SPUNLMS algorithms. We used $\mu = 1$ for all cases and $\sigma = 0.01$ and $\rho = 0.01$ for PNLMS++. For PSNLMS and SPUNLMS, we only selected 162 coefficients to be updated, thus in every sample, we left 350 taps unchanged. We also added a white Gaussian noise with variance 0.001 to the reference signal to model the environmental noise. To smooth the squared output error, we ran 10 independent simulations and got the average results. From the results, we saw that the PSNLMS-I algorithm converged faster than NLMS and SPUNLMS for both white and color input. PNLMS++ had a faster initial convergence speed than PSNLMS-I, but after certain point, it began to slow down.

In the second example, we compared the learning curves of PSNLMS-II, NLMS, PNLMS and SPUNLMS algorithms. We used the same settings as in the first example. From the results we saw that PSNLMS-II, if appropriately chose T, had better performance than PSNLMS-I.

5. CONCLUSION

In this paper, we propose two PSNLMS algorithms for modeling impulse responses using partial update. Coefficients to be updated are selected based on the product of the delayed input and the corresponding coefficient value. Simulation results show that the algorithms produce better results than NLMS and SPUNLMS in both convergence speed and computational

efficiency. Although the proposed algorithms have slower initial convergence than PNLMS++, it demands much less computations. Further research will concentrate on theoretical verification of the performance of the algorithms and on analytical insight into the convergence property.

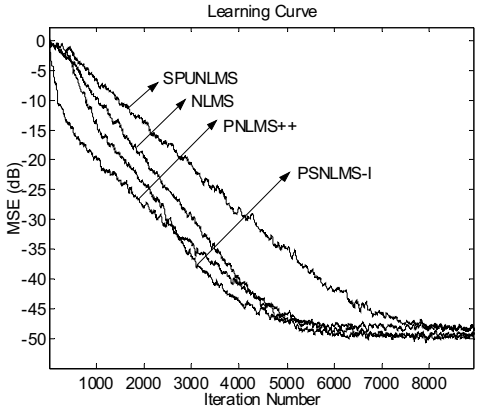

Fig.2 Learning curves for PSNLMS-I, NLMS, PNLMS and SPUNLMS (white input case)

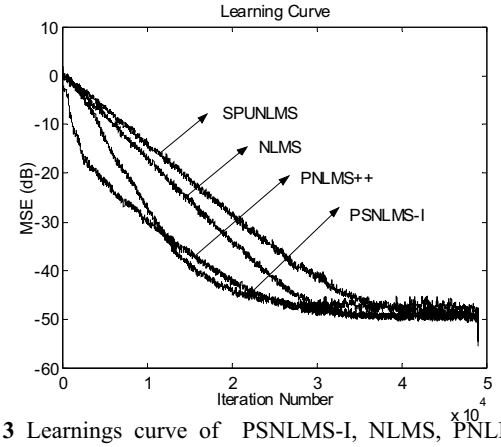

Fig. 3 Learnings curve of PSNLMS-I, NLMS, PNLMS and SPUNLMS (color input case)

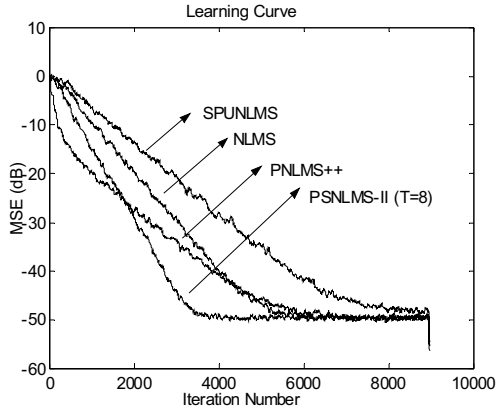

Fig. 4 Learning curves of PSNLMS-II, NLMS, PNLMS and SPUNLMS (white input case)

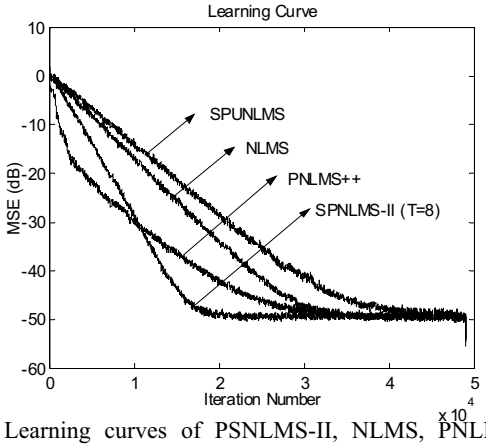

Fig.5 Learning curves of PSNLMS-II, NLMS, PNLMS and SPUNLMS (color input case)

6. REFERENCES

[1]Simon Haykin, *Adaptive Filter Theory*, Fourth Edition, Prentice Hall, ISBN 0-13-090126-1

[2]S. Douglass, "Adaptive Filters Employing Partial Update", *IEEE Transaction on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. 44, No.3, Mar 1997, pp 209-216

[3]K. Dogancay and O. Tanrikulu, "Adaptive Filtering Algorithms With Selective Partial Updates", *IEEE Transaction on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. 48, No.8, Aug 2001, pp 762-769

[4]T. Aboulnasr and K. Mayyas, "Complexity Reduction of the NLMS algorithm via Selective Coefficient Update", *IEEE Trans. On Signal Processing*, Vol. 47, No. 5, May 1999, pp 1421-1424.

[5]D. Duttweiler, "Proportionate Normalized Leas-Mean-Square Adaptation in Echo Cancelers", *IEEE Transaction on Speech and Audio Processing*, Vol. 8, No.5, Sep. 2000, pp 508-518.

[6]S. Gay, "An Efficient, Fast converging Adaptive Filter for Network Echo Cancellation". *The Thirty-Second Asilomar Conference on Signals, Systems & Computers*, Nov. 1998, Vol. 1, pp 394-398.

[7]P. Naylor and W. Sherliker, "A Short-Sort M-Max NLMS Partial-Update Adaptive Filter With Application to Echo Cancellation", *ICASSP 2003*.

[8]C. Shaffer, *Practical Introduction to Data Structures and Algorithm Analysis*, Second Edition, Prentice Hall, ISBN 0130284467