Improving Convergence of the PNLMS Algorithm for Sparse Impulse Response Identification

Hongyang Deng, *Student Member, IEEE,* and Miloš Doroslovački, *Member, IEEE*

***Abstract*—A proportionate normalized least mean square (PNLMS) algorithm has been proposed for sparse impulse response identification. It provides fast initial convergence, but it begins to slow down dramatically after the initial period. In this letter, we analyze the coefficient adaptation process of the steepest descent algorithm and derive how to calculate the optimal proportionate step size in order to achieve the fastest convergence. The results bring forward a novel view of the concept of proportion. We propose a μ-law PNLMS (MPNLMS) algorithm using an approximation of the optimal proportionate step size. Line segment approximation and partial update techniques are discussed to bring down the computational complexity.**

***Index Terms*—Adaptive filtering, convergence, network echo cancellation, proportionate normalized least mean square (PNLMS) algorithm, steepest descent algorithm.**

I. Introduction

The main idea of the proportionate normalized least mean square (PNLMS) algorithm [1], [2] is to assign different step sizes to different coefficients based on their optimal magnitudes. The bigger the magnitude, the larger the step size assigned. Since, in practice, we do not know the optimal magnitudes, we use the current estimated magnitudes instead. For the majority of the time indices, sparse impulse responses are zero. An adaptive filter initialized with zero coefficients should update only the coefficients whose optimal values are nonzero.

Let $x(k)$ be the input signal that excites the unknown system and $\hat{\mathbf{w}}(k)$ be the adaptive filter coefficients. Furthermore, let $y(k)$ be the output of the unknown system and $v(k)$ the measurement noise. The observation $d(k)$ is the sum of $y(k)$ and $v(k)$. The error signal $e(k)$ between the output of the adaptive filter $\hat{y}(k)$ and $d(k)$ drives the adaptive algorithm. The PNLMS algorithm is specified in Table I, where L is the length of the adaptive filter, β is the step-size parameter, and δ is a small positive number used to avoid overflowing. The constant δ_p is important when all the coefficients are 0 (in the beginning) and, together with ρ, prevent the very small coefficients from stalling. Note that $\sum_{l=1}^{L} g_l(k) = L$, and the value of $g_l(k)$ is dependent on the coefficients' current estimate, except at the beginning, when special care is taken about zero or extremely small coefficients.

TABLE I
PNLMS Algorithm

$$\boldsymbol{x}(k) = [x(k)\ \ x(k-1)\ \cdots\ x(k-L+1)]^T$$
$$\hat{y}(k) = \boldsymbol{x}^T(k)\hat{\boldsymbol{w}}(k)$$
$$e(k) = d(k) - \hat{y}(k)$$
$$\hat{\boldsymbol{w}}(k+1) = \hat{\boldsymbol{w}}(k) + \frac{\beta\, \boldsymbol{G}(k+1)\boldsymbol{x}(k)e(k)}{\boldsymbol{x}^T(k)\boldsymbol{G}(k+1)\boldsymbol{x}(k)+\delta}$$
$$\boldsymbol{G}(k+1) = \mathrm{diag}\{g_1(k+1),\cdots,g_L(k+1)\}$$
$$\gamma_{\min}(k+1) = \rho\max\{\delta_p, |\hat{w}_1(k)|,\cdots,|\hat{w}_L(k)|\}$$
$$\gamma_l(k+1) = \max\{\gamma_{\min}(k+1), |\hat{w}_l(k)|\}$$
$$g_l(k+1) = \frac{\gamma_l(k+1)}{\frac{1}{L}\sum_{i=1}^{L}\gamma_i(k+1)}, \quad 1 \le l \le L.$$

The PNLMS algorithm, as demonstrated in [1] and [2], has very fast initial convergence speed, which is favorable for applications such as network echo cancellation. However, after the initial period, it begins to slow down dramatically, even becoming slower than the NLMS algorithm. Since, at that point, the error $e(k)$ is still big, the overall performance degrades. Although the PNLMS++ [2] algorithm improves the convergence of the PNLMS algorithm somewhat, it does not solve the problem. What we want to achieve is to keep the fast initial convergence speed during the whole adaptation process until the adaptive filter reaches its steady state. Some ideas for reaching this goal are presented and evaluated in the next sections.

II. Optimal Proportionate Step Size

From the steepest descent algorithm [3], we have

$$\hat{\mathbf{w}}(k+1) = \hat{\mathbf{w}}(k) + \beta[\mathbf{p} - \mathbf{R}\hat{\mathbf{w}}(k)] \tag{1}$$

where $\mathbf{R} = E[\mathbf{x}(k)\mathbf{x}^T(k)]$ is the auto-correlation matrix of the input signal, and $\mathbf{p} = E[d(k)\mathbf{x}(k)]$ is the cross-correlation vector between the input signal vector and the desired signal. The adaptation equation of coefficients can be written as

$$\hat{\mathbf{w}}(k) - \mathbf{w}_{\text{opt}} = (\mathbf{I} - \beta\mathbf{R})^k(\hat{\mathbf{w}}(0) - \mathbf{w}_{\text{opt}}) \tag{2}$$

where $\mathbf{w}_{\text{opt}} = \mathbf{R}^{-1}\mathbf{p}$ is the optimal solution, $\mathbf{I}$ is the identity matrix, k is the number of iterations, and $\hat{\mathbf{w}}(0)$ is the initial guess for the adaptive filter coefficients. Usually, we choose $\hat{\mathbf{w}}(0) = \mathbf{0}$, which is a good choice for sparse systems. Thus

$$\hat{\mathbf{w}}(k) = [\mathbf{I} - (\mathbf{I} - \beta\mathbf{R})^k]\mathbf{w}_{\text{opt}}. \tag{3}$$

Manuscript received May 5, 2004; revised September 19, 2004. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Zhi Ding.

The authors are with the Department of Electrical and Computer Engineering, The George Washington University, Washington, DC 20052 USA (e-mail: denghy@gwu.edu; doroslov@gwu.edu).

Digital Object Identifier 10.1109/LSP.2004.842262

Let us assume that the input signal is white and stationary; hence, $\mathbf{R} = \sigma_x^2 \mathbf{I}$, where σ_x^2 is the variance of the input. Then

$$\hat{\mathbf{w}}(k) = \left[1 - \left(1 - \beta\sigma_x^2\right)^k\right] \mathbf{w}_{\mathrm{opt}}. \tag{4}$$

Let us define

$$\beta\sigma_x^2 = \lambda. \tag{5}$$

Now, we can write

$$\hat{\mathbf{w}}(k) = [1 - (1 - \lambda)^k]\mathbf{w}_{\mathrm{opt}}. \tag{6}$$

Note that $\hat{\mathbf{w}}(k)$ converges to $\mathbf{w}_{\mathrm{opt}}$ for $\lambda \in (0, 2)$.

Now, let us define the convergence time as a quantitative measure of the convergence speed of the adaptive filter.

Definition 1 (Coefficient Convergence Time): The convergence time of the pth coefficient is the number of iterations k_p that is needed for the coefficient to reach the ε-vicinity of its optimal value for a given small positive number ε.

Definition 2 (Overall Convergence Time): The overall convergence time is the number of iterations k_0 that is needed for all the coefficients to reach the ε-vicinity of their optimal values, for a given small positive number ε.

Note that both convergence times depend on the value of ε.

From (6), we can calculate the coefficient convergence time of the pth coefficient as

$$k_p = \ln \frac{|w_{\mathrm{opt},p}|}{\varepsilon} \Bigg/ \ln \frac{1}{|1 - \lambda|}. \tag{7}$$

Since ε and λ are constants, the convergence time of each coefficient is solely dependent on its optimal magnitude. The bigger the optimal magnitude, the longer it takes the coefficient to converge. The overall convergence time defined above is actually equal to the longest coefficient convergence time (i.e., $k_0 = \max\{k_1, k_2, \ldots, k_L\}$) and, of course, corresponds to the biggest optimal magnitude. This conclusion directly leads to the idea of the PNLMS algorithm: assigning big step sizes to big coefficients in order to make them converge faster, thus decreasing the overall convergence time.

In the proportionate framework, we use step-size control factors $g_p, p = 1, 2, \ldots, L$ to assign different step sizes to different coefficients, that is

$$\beta_p = \beta g_p \tag{8}$$

and

$$\lambda_p = \beta_p \sigma_x^2 = \beta g_p \sigma_x^2 = \lambda g_p, \quad p = 1, \ldots, L. \tag{9}$$

The convergence time for the pth coefficient can be found from (7) as

$$k_p = \ln \frac{|w_{\mathrm{opt},p}|}{\varepsilon} \Bigg/ \ln \frac{1}{|1 - \lambda g_p|}. \tag{10}$$

Therefore, for proportionate algorithms, the coefficient convergence times depend on both the coefficients' optimal magnitudes and the step-size control vector $\mathbf{g} = [g_1, \ldots, g_L]^T$. Now, the question is how to choose $\mathbf{g}$ to achieve the fastest overall convergence.

Proposition 1: For proportionate algorithms, which use step-size control factors to assign different step sizes to different coefficients based on their optimal magnitudes, the fastest overall convergence speed (the smallest overall convergence time) is achieved when the convergence times for all the coefficients are the same, i.e., all coefficients reach the ε-vicinity of their respective optimal values simultaneously [4].

For any coefficient $\hat{w}_p(k)$, let $k_p = k_0$. Now, we can find that

$$g_p = \frac{1}{\lambda}\left[1 - s_p\left(\frac{|w_{\mathrm{opt},p}|}{\varepsilon}\right)^C\right] \tag{11}$$

where $C = -1/k_0$ and $s_p \in \{-1, +1\}$. Since the step-size control vector $\mathbf{g}$ must satisfy the condition $\sum_{p=1}^{L} g_p = L$, we conclude that

$$\sum_{p=1}^{L} \frac{1}{\lambda}\left[1 - s_p\left(\frac{|w_{\mathrm{opt},p}|}{\varepsilon}\right)^C\right] = L. \tag{12}$$

In order to achieve the fastest overall convergence speed, we have to calculate the step-size control factor for the pth coefficient g_p according to (11) and the constant C according to (12). The PNLMS algorithm proposed in [1] and [2] does not use this approach to get the step-size control vector. It emphasizes too much the update of the big coefficients. The big coefficients converge very fast (the initial fast convergence) at the cost of slowing down dramatically the convergence of the small coefficients (after the initial period).

III. μ-Law PNLMS

A. Algorithm Description

It is difficult to find the close-form expression for the constant C by solving (12). We can solve (12) by using a numeric method, but the added computational complexity makes the above optimal proportionate step-size control mechanism impractical. On the other hand, by using some reasonable assumptions, we can derive a method that has nearly the same performance with a much lower computational burden.

For $0 < \lambda g_p \ll 1$, we have $\ln(1 - \lambda g_p) \approx -\lambda g_p$. Then, from (10), the convergence time for any coefficient $\hat{w}_p(k)$ is

$$k_p = \ln \frac{\varepsilon}{|w_{\mathrm{opt},p}|} \Bigg/ \ln(1 - \lambda g_p) \approx \ln \frac{|w_{\mathrm{opt},p}|}{\varepsilon} \Bigg/ \lambda g_p. \tag{13}$$

Let $k_p = k_0$; then, from (13), we get

$$g_p \approx \frac{1}{k_0 \lambda} \ln \frac{|w_{\mathrm{opt},p}|}{\varepsilon}. \tag{14}$$

Based on the condition $\sum_{p=1}^{L} g_p = L$, we can calculate k_0 as

$$k_0 = \frac{1}{\lambda L} \sum_{p=1}^{L} \ln \frac{|w_{\mathrm{opt},p}|}{\varepsilon}. \tag{15}$$

Therefore

$$g_p \approx \frac{\ln(|w_{\text{opt},p}|/\varepsilon)}{\frac{1}{L}\sum_{i=1}^{L}\ln(|w_{\text{opt},i}|/\varepsilon)}. \tag{16}$$

Although the results are derived in the steepest descent algorithm framework, we can use LMS/NLMS-type algorithms as a stochastic approximation of the steepest descent algorithm, expecting that they will have similar convergence behavior. Now, we propose the MPNLMS algorithm that is listed in Table II. We choose function $\mathrm{F}(|\hat{w}_l(k)|)$ based on (16). The quantity $|\hat{w}_l(k)|$ is the estimate of $|w_{\text{opt},l}|$, and μ represents $1/\varepsilon$. The function $\mathrm{F}(|\hat{w}_l(k)|)$ is nothing but the μ-law used in nonuniform compression in telecommunication applications. We purposely add the constant 1 inside the logarithm in order to avoid having a singular point when $|\hat{w}_l(k)| = 0$. The denominator $\ln(1+\mu)$ normalizes $\mathrm{F}(|\hat{w}_l(k)|)$ to be in the range [0, 1].

The MPNLMS algorithm is motivated by the optimal convergence results we obtained for the steepest descent algorithm with different but fixed step-size control factors. Based on that, we expect that the proposed algorithm will behave similarly after the initial period, i.e., when the step-size control factors do not change a lot. Therefore, we also expect that the proposed algorithm will converge faster than the PNLMS and PNLMS++ algorithm, which calculate the step-size factors without being motivated by the optimization of any quantitative measure of convergence speed. The expectations are verified by the simulation results.

The parameter ε is a very small positive number, and its value should be chosen based on the measurement noise level. For applications such as network echo cancellation, $\varepsilon = 0.001$ is a good choice because the echo below -60 dB is negligible. Therefore, we can choose $\mu = 1000$.

The above μ-law function is only defined on [0, 1]. If some coefficient magnitudes fall out of this range, or the entire active coefficient magnitudes are too small or too big, the function cannot scale the coefficients correctly, thus making the MPNLMS algorithm fail. Therefore, normalization should be done before we calculate the step-size control factors [4].

B. Computational Complexity

The μ-law function is expensive for computation. In addition to the computational load of the PNLMS algorithm, which needs approximately 50% more computations than the NLMS algorithm [1], it adds L logarithm calculations, L multiplications, and L additions per iteration. Since calculation of the logarithm may be an expensive operation for a digital signal processor (DSP), the added computational overhead can be big. Therefore, in order to make this algorithm more attractive, we may need to lower its computational complexity.

One technique is to use line segments to approximate the μ-law function, which is widely used in telecommunication applications. For our application, we can just divide the whole function range into two sections: one for very small values and one for other values. Therefore, we can use two straight lines to replace the μ-law function. The sharpness of the initial slope is critical because we want to emphasize the small values.

TABLE II
MPNLMS ALGORITHM

$$\boldsymbol{x}(k) = [x(k)\ \ x(k-1)\ \cdots\ x(k-L+1)]^T$$
$$\hat{y}(k) = \boldsymbol{x}^T(k)\hat{\boldsymbol{w}}(k)$$
$$e(k) = d(k) - \hat{y}(k)$$
$$\hat{\boldsymbol{w}}(k+1) = \hat{\boldsymbol{w}}(k) + \frac{\beta\,\boldsymbol{G}(k+1)\boldsymbol{x}(k)e(k)}{\boldsymbol{x}^T(k)\boldsymbol{G}(k+1)\boldsymbol{x}(k)+\delta}$$
$$\boldsymbol{G}(k+1) = \mathrm{diag}\{g_1(k+1),\cdots,g_L(k+1)\}$$
$$\mathrm{F}(|\hat{w}_l(k)|) = \frac{\ln(1+\mu|\hat{w}_l(k)|)}{\ln(1+\mu)},\ |\hat{w}_l(k)| \le 1,\ 1<l<L,\ \mu = 1/\varepsilon$$
$$\gamma_{\min}(k+1) = \rho\max\{\delta_p, \mathrm{F}(|\hat{w}_1(k)|),\cdots,\mathrm{F}(|\hat{w}_L(k)|)\}$$
$$\gamma_l(k+1) = \max\{\gamma_{\min}(k+1), \mathrm{F}(|\hat{w}_l(k)|)\}$$
$$g_l(k+1) = \frac{\gamma_l(k+1)}{\frac{1}{L}\sum_{i=1}^{L}\gamma_i(k+1)},\quad 1\le l\le L.$$

One of the functions we can use to scale the coefficients is defined as

$$F(x) = \begin{cases} 200x, & x < 0.005 \\ 1, & \text{otherwise.} \end{cases} \tag{17}$$

Now, the function needs less than L extra multiplications and comparisons, which is very easy to implement using a DSP. We call the corresponding algorithm a Segment PNLMS (SPNLMS) algorithm.

Another technique we can use is the partial update. In [5], the partial update NLMS algorithm is proposed to achieve nearly the same performance as the full update by only updating a small portion of the coefficients where selection is based on the amplitude of the corresponding inputs. Simulations show that we can keep the good performance of our proposed algorithms by using the partial update, thus making the algorithm's computational complexity low enough to be performed in real time, even for long adaptive filters.

IV. Simulation Results

We evaluate our proposed algorithms using network echo cancellation as a typical example. Network echo path impulse response is sparse in nature. Although we need to use an adaptive filter with a large number of coefficients to model and estimate the echo path impulse response, only a few of them are different from zero (we call them active coefficients). All others are just zero or unnoticeably small (nonactive coefficients). Fig. 1 shows the impulse response of a typical network echo path that we are using in simulations. Among 512 taps (64 ms for 8 KHz sampling frequency), only about 60 (less than 8 ms) are active.

In the simulations, we use white Gaussian noise with zero mean and unit variance as the input. We also add white Gaussian noise to the output of the echo path to control the steady-state error (SNR $=$ 60 dB). The step-size parameter is 0.3. For all proportionate type algorithms, we use $\delta_p = 0.01$ and $\rho = 0.01$.

In Fig. 2, by comparing the learning curves of different algorithms, we can see that our proposed MPNLMS algorithm has a much faster convergence than the PNLMS and

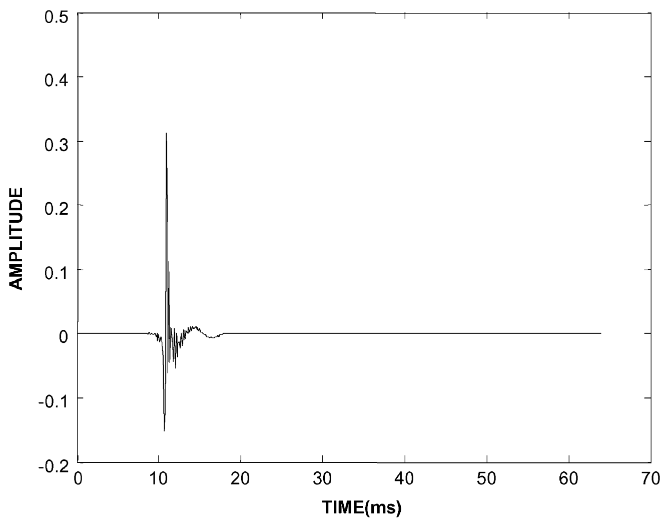

Fig. 1. Network echo path impulse response.

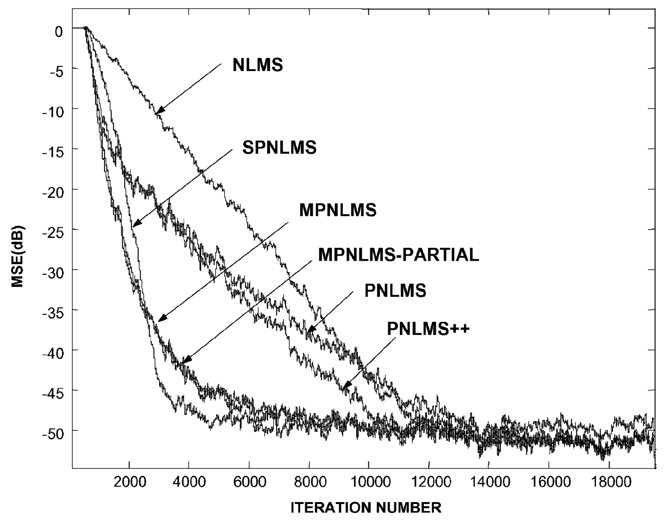

Fig. 2. Learning curves of algorithms.

PNLMS++ algorithms. It not only converges fast initially, but it also keeps this fast convergence rate until it reaches -50 dB. For many practical purposes, it would mean that the algorithm has "converged." The line segment approximation SPNLMS algorithm, with fewer computations, has an equivalently good performance. One interesting observation is that the SPNLMS algorithm, although slower initially, outperforms the MPNLMS eventually. In addition to the fact that we use estimated coefficient magnitudes instead of optimal coefficient magnitudes to calculate the step-size control factors, a second reasonable explanation of the observation is that the SPNLMS algorithm assigns more update emphasis to small coefficients than the MPNLMS algorithm. Another simulation illustrates that the partial update MPNLMS algorithm has nearly the same performance as full update with only updating one third of the coefficients per iteration.

V. Conclusion

We have derived how to calculate the optimal proportionate step-size control factors starting from the coefficient adaptation equation of the steepest descent algorithm and minimization of its overall convergence time. The analysis expands the scope to the idea of proportion, which is previously understood mainly as linearly proportional to the weight magnitudes. Motivated by the obtained results and by using reasonable approximations, we have proposed an algorithm that improves the convergence performance of the PNLMS algorithm. Simulations show that the proposed MPNLMS algorithm can converge much faster than the PNLMS and PNLMS++ algorithms. It not only has the same fast initial convergence as PNLMS and PNLMS++, but it also keeps the fast convergence until it reaches the steady state. To lower the computational complexity, a scaling function approximation and partial update techniques are employed.

References

[1] D. L. Duttweiler, "Proportionate normalized least-mean-squares adaptation in echo cancellers," *IEEE Trans. Speech Audio Process.*, vol. 8, no. 5, pp. 508–518, Sep. 2000.

[2] S. L. Gay, "An efficient, fast converging adaptive filter for network echo cancellation," in *Proc. Asilomar Conf.*, Monterey, CA, Nov. 1998, pp. 394–398.

[3] S. Haykin, *Adaptive Filter Theory*, 4th ed. Englewood Cliffs, NJ: Prentice-Hall, 2002.

[4] H. Deng and M. Doroslovački, "Modified PNLMS adaptive algorithm for sparse echo path estimation," in *Proc. Conf. Inf. Sci. Syst.*, Princeton, NJ, Mar. 2004.

[5] K. Dogancay and O. Tanrikulu, "Adaptive filtering algorithms with selective partial update," *IEEE Trans. Circuit Syst. II, Analog Digit. Signal Process.*, vol. 48, no. 8, pp. 762–769, Aug. 2001.