Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 70 (2006) 9–13
NEUROCOMPUTING
www.elsevier.com/locate/neucom

# Using non-linear even functions for error minimization in adaptive filters

Allan Kardec Barros[a,*], Jose Principe[b], Yoshinori Takeuchi[c], Noboru Ohnishi[c]

[a]*Universidade Federal do Maranhao, Brazil*
[b]*University of Florida at Gainsville, USA*
[c]*Nagoya University, Japan*

Available online 18 August 2006

**Abstract**

In this work, we analyze algorithms for adaptive filtering based on non-linear cost function of the error, which we named *non-linear even moment* (NEM) algorithms. We assume that this non-linear function can be generally described in a Taylor series as a linear combination of the even moments of the error. NEM is a generalization of the well-known *least mean square* (LMS). We study the NEM convergence behavior and derive equations for misadjustment and convergence. We found a good approximation for the theoretical results and we show that there are various combinations of the even moments which yields better results than the LMS as well as other algorithms proposed in the literature.


*Keywords:* Non-linear error; Least mean square

## 1. Introduction

In many signal processing applications using adaptive filtering, there is a need of algorithms that yield small error, fast convergence and low computational complexity. Usually, these algorithms are analyzed under a framework where a number of linearizations are carried out, so that one can easily access both the convergence time and the misadjustment error. Moreover, underlying these methods are some assumptions about the statistics of the signals under study. This yields important simplifications in the analysis of the algorithm. However, those linearizations and assumptions may oversimplify the problem or hide important properties of the algorithms.

Among the adaptive filters, the *least mean square* algorithm (LMS) of Widrow and Hoff [5] appears as one of the most widely used. The LMS belongs to a class of algorithms that can be designated as *second order statistics* (SOS), in opposition to *higher order statistics* (HOS). The use of SOS methods are sufficient when the signals involved in the application are Gaussian distributed, yielding a number of simplifications in the algorithm analysis, as well as leading to computationally less expensive methods.

Interestingly, probably due to the increase in the computational power in the last decades, HOS methods have drawn more attention of the research community. Indeed, instead of dealing only with the signal's power (i.e., SOS), HOS allows access to the information contained in all moments of the signal [6], yielding therefore a better approximation of the actual distribution of the signal under study. As a result, one can expect that algorithms designed under the HOS framework behave more efficiently.

An interesting idea would be to explore the HOS of the error, such as carried out in the works of Walach and Widrow [7], Chambers et al. [1] or Erdogmus et al. [3]. There is an interesting property which is: the mean of the error raised to even powers is a convex function of the weight vector. This can be interpreted as the error cannot have local minima [4]. Here we generalize the work of Chambers et al. [1], that proposed a weighted sum of the moments of order two and four. The idea behind the sum

*Corresponding author.
*E-mail addresses:* akbarros@ieee.org, allan@dee.ufma.br (A.K. Barros).

0925-2312/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2006.07.001

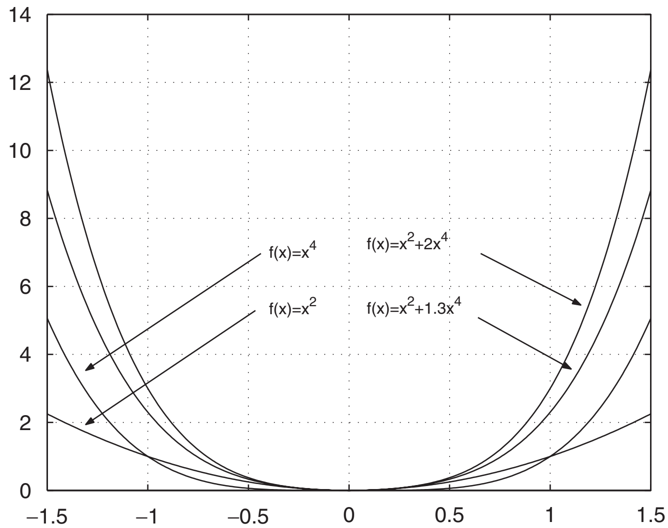


Fig. 1. Here we show the behavior of four functions to illustrate the idea of adding power functions. One is the usual square, another is the variable to the fourth, and a third is a weighted sum of the previous two. One can see that around zero the function has less variance in the case of $x^2$, while $x^4$ has a large drop as it gets far from zero. The idea in this work is to use the advantages of both, as shown in the third and fourth functions.

of errors is that one can have the good behavior of the second order moment in steady state allied to the fast convergence of higher order even moments, as shown in Fig. 1.

Moreover, it is worth saying that in the study of convergence time or misadjustment of adaptive algorithms, one analyzes their behavior near the optimum solution, which yields interesting linearizations [2,7]. This policy makes sense in the case of misadjustment, which should be studied when the learning reaches steady state. However, it may lead to large errors in the case of convergence time, as it is an indication of how fast the algorithm has *started* the learning. Thus, we also propose a new way of evaluating the convergence time here, by analyzing the algorithm behavior in the beginning of the learning task.

## 2. The method

Let us consider that we observe a given signal $d_j$ and a number of others, which can be included into a vector $\mathbf{X}_j = [x_{j,1} \;\; x_{j,2} \;\; \cdots \;\; x_{j,M}]$, called reference input. Moreover, let us define $d_j = s_j + n_j$, where $s_j$ is the signal we want to extract and $n_j$ is the noise. Let us also assume that $n_j$ is statistically independent of $s_j$ and $\mathbf{X}_j$, whereas all these variables have probability distributions which are not necessarily Gaussian. Our aim is to estimate $s_j$, after optimally calculating the weight $\mathbf{W}_j = [w_{j,1} \;\; w_{j,2} \;\; \cdots \;\; w_{j,M}]$ and the current error $\varepsilon_j = d_j - y_j$, where the output signal is given by $y_j = \mathbf{W}_j^{\mathrm{T}}\mathbf{X}_j$. We assume that the weight vector coefficients are statistically independent of the input vector.

In this optimization, the Widrow–Hoff algorithm uses an instantaneous estimation of the gradient of $E[\varepsilon_j^2]$. However, our interest is to minimize a general cost function $\zeta_K = f\{E[\varepsilon]\}$. We will assume that $f\{\cdot\}$ is a even function and therefore it can be rewritten in a Taylor series as a sum of even moments of the error. Thus, we can write

$$\zeta_K = \sum_{K=1}^{N} a_K (2K)^{-1} E[\varepsilon_j^{2K}], \tag{1}$$

where $a_K$ is a scaling factor. The term $2K^{-1}$ was introduced only for ease of manipulation.

Thus, the instantaneous gradient of (1), $\nabla(\zeta_K) = -2(\sum_{K=1}^{N} a_K \varepsilon_j^{2K-1})\mathbf{X}_j$, will lead to the following simple update weight rule:

$$\mathbf{W}_{j+1} = \mathbf{W}_j + 2\mu\left(\sum_{K=1}^{N} a_K \varepsilon_j^{2K-1}\right)\mathbf{X}_j, \tag{2}$$

where $\mu$ is a learning constant, controlling the stability and rate of convergence.

## 3. Adaptation analysis

The first task for analyzing the algorithm behavior should be to check the conditions under which it converges to the desired solution, and how it behaves until it reaches steady state. This can be carried out by analyzing the misadjustment error and the convergence time.

Let us first make a change of variable, by defining the vector $\mathbf{V}_j = \mathbf{W}_j - \mathbf{W}_*$, where $\mathbf{W}_*$ is the optimum solution, i.e., $s_j = \mathbf{W}_*^{\mathrm{T}}\mathbf{X}_j$. Thus, (2) becomes,

$$\mathbf{V}_{j+1} = \mathbf{V}_j + 2\mu\left(\sum_{K=1}^{N} a_K \varepsilon_j^{2K-1}\right)\mathbf{X}_j. \tag{3}$$

More specifically, (3) can be rewritten in the form of a binomial expansion as follows:

$$\mathbf{V}_{j+1} = \mathbf{V}_j + 2\mu\left[\sum_{K=1}^{N}\sum_{i=0}^{2K-1} a_K \binom{2K-1}{i} \times n_j^i(-\mathbf{X}_j^{\mathrm{T}}\mathbf{V}_j)^{2K-1-i}\right]\mathbf{X}_j. \tag{4}$$

One can study the misadjustment, which is a measure of how far the output differs from the ideal solution. The misadjustment calculation can be performed in the neighborhood of the optimal solution, i.e., $\mathbf{V}_j \to 0$. Hence, we can neglect the higher powers of $\mathbf{V}_j$ in (4). By remembering that $\varepsilon_j = s_j + n_j - \mathbf{W}_j^{\mathrm{T}}\mathbf{X}_j = n_j - \mathbf{V}_j^{\mathrm{T}}\mathbf{X}_j$, we have,

$$\mathbf{V}_{j+1} \simeq \mathbf{V}_j + 2\mu\left[\sum_{K=1}^{N} a_K \mathbf{X}_j(n_j^{2K-1} - (2K-1)n_j^{2K-2}\mathbf{X}_j^{\mathrm{T}}\mathbf{V}_j)\right], \tag{5}$$

where we made an approximation up to the second order.

Defining $\mathbf{R} = E[\mathbf{X}_j\mathbf{X}_j^{\mathrm{T}}]$, and recalling that $\mathbf{X}_j$ and $n_j$ were assumed to be mutually independent, we can study the behavior of $\mathbf{V}_j$, by taking the expectations at