Online data centering modification for adaptive filtering with NLMS algorithm

Matouš Cejnek*
Adaptive Signal Processing and Informatics Computational Centre (ASPICC)
Department of Instrumentation and Control
Faculty of Mechanical Engineering, Czech Technical University
Technicka 4, Prague, Czech Republic.

Abstract

This paper presents a method of real-time data transformation for better performance of normalized least mean squares (NLMS) algorithm. The method centers input vector for adaptive filter online according to temporary statistical attributes of the input vector. The method is derived for an adaptive filter with NLMS adaptation. The filter implementation is the linear neural unit. The stability condition for the given filter is also presented. The filter is tested on multiple simulated time series contaminated with white noise and also on real measured signal. The convergence of the suggested algorithm is also analyzed and time complexity is discussed.

Key-words: adaptive filters; least mean square algorithms

1. Introduction

Filtering is important part of signal processing in general. The adaptive filter is such a filter that adaptively changes its transfer function during process of filtering to minimize the filtering error [1]. The one of the most common adaptive algorithm for adaptive filters is least mean squares (LMS) [2]. Applicability of this algorithm for the adaptive filters was extensively studied in past [3]. Plenty of LMS modifications and variants have been derived and developed since this time. For example: normalized LMS [4], total least mean squares [5], generalized normalized gradient descent [6], proportionate normalized least mean squares [7]. Probably the most common modification for LMS is the mentioned learning rate normalization, well-known as normalized least mean squares (NLMS) [4]. The NLMS algorithm could produce better results than plain LMS. A lot of comparisons of LMS and NLMS have been done [8, 9, 10]. According to [7] the NLMS should work better than classical LMS in cases where is big difference between standard deviation of input vectors and standard deviation of all input data. The derivation and analysis of NLMS is based on usual independence assumptions [6], what also means, that input vector and adaptive weights of filter should be zero mean. In practice this condition may not be fulfilled. Common practice how to deal with not zero mean data for further improvement of LMS or NLMS adaptation result could be achieved with transformation of measured data, as it was used in studies [11, 12, 13, 14]. Commonly used transformation is input vector centering or z-scoreing. According to structure and scale of input data these transformations could or could not improve the filtering result with various influence. Problem is that the mentioned normalization-like transformations are applicable only offline, when the data are already measured. Our proposed algorithms are based on the assumption, that accuracy of learning process is related to the condition number of input matrix and thus the proposed methods is using online data centering to achieve better filtering result. In this workpropose a novel approach. The input vector is centered according to actual mean value of input vector.must note that our usage of this transformation does not respect data structure and it can cause information loss. But with correct usage of our method this loss cause smaller performance decrease than the sub optimal data scale and structure. The proposed algorithm is tested and compared with plain NLMS and NLMS used on z-scored data to display, how could be improved plain NLMS with our method according to input data structure and scale. The NLMS with offline z-score data results stands in this comparison as unachievable goal for online adaptation.

1.1. Review of NLMS

An adaptive filter could be described with equation

$$\tilde{y}(k) = w_1 \cdot x_1(k) + ... + w_n \cdot x_n(k), \qquad (1)$$

or in a vector form

$$\tilde{y}(k) = \mathbf{w}^T(k) \cdot \mathbf{x}(k), \qquad (2)$$

where k is discrete time index, $\tilde{y}(k)$ is filtered signal, $\mathbf{w}$ is vector of filter adaptive parameters (at the beginning the parameters are set to small random numbers) and $\mathbf{x}$ is input vector made from measured signal $y(k)$ and bias ($= 1$)

$$\mathbf{x}(k) = [1, y(k-n-1), ..., y(k-1)], \qquad (3)$$

where n is the size of input vector. Our method is based on normalized least mean squares (NLMS) algorithm that is a modification of classical least squares algorithm (LMS) also known as stochastic gradient descent. The LMS weights adaptation could be described as follows

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \Delta\mathbf{w}(k), \qquad (4)$$

where $\Delta\mathbf{w}(k)$ is

$$\Delta\mathbf{w}(k) = \mu \cdot e(k) \cdot \frac{\partial y(k)}{\partial \mathbf{w}(k)} = \mu \cdot e(k) \cdot \mathbf{x}(k), \qquad (5)$$

where μ is the learning rate (step size) and e is error defined as

$$e(k) = y(k) - \tilde{y}(k). \qquad (6)$$

*Kontakt na autora: matous.cejnek@fs.cvut.cz

According to the general stability criteria of LMS [6]

$$|1 - \mu \cdot \mathbf{x}(k)^T \cdot \mathbf{x}(k)| \leq 1, \tag{7}$$

the NLMS adaptation rule could be described as follows

$$\Delta\mathbf{w}(k+1) = \frac{\mu}{\epsilon + \mathbf{x}(k)^T \cdot \mathbf{x}(k)} \cdot \mathbf{x}(k) \cdot \mathbf{w}(k), \tag{8}$$

where ϵ is a constant (regularization term) introduced to preserve stability for inputs close to zero [6]. The model is stable if

$$0 \leq \mu \leq 2 + \frac{2\epsilon}{\mathbf{x}(k)^T \cdot \mathbf{x}(k)}, \tag{9}$$

or in case without regularization term ϵ

$$\mu \in < 0, 2 > . \tag{10}$$

2. Filter model

Common transformation for improving the condition number of input data matrix $\mathbf{x}$ is z-score of input data

$$y_t(k) = \frac{y(k) - \bar{y} \cdot \vec{1}}{\sigma_y}, \tag{11}$$

where $\bar{y}$ is mean value of y, σ_y is standard deviation of y and $\vec{1}$ is n sample length vector of all ones. The result of transformed signal filtering $\tilde{y}_t$ could be transformed back as simple as

$$\tilde{y}(k) = (\tilde{y}_t(k) \cdot \sigma_y) + \bar{y} \cdot \vec{1}. \tag{12}$$

Filter with normalized data could be defined according to (2) as follows

$$\tilde{y}_t(k) = \mathbf{w}_t^T(k) \cdot \mathbf{x}_t(k), \tag{13}$$

where $\mathbf{x}_t(k)$ is input vector build from transformed data y_t according to (3) and $\mathbf{w}_t(k)$ is set of parameters of adaptive filter for transformed data. From (11) and (13) is possible to obtain

$$\frac{\tilde{y}(k) - \bar{y}}{\sigma_y \cdot \vec{1}} = \mathbf{w}_t^T(k) \cdot \left(\frac{\mathbf{x}(k) - \bar{y} \cdot \vec{1}}{\sigma_y} \right), \tag{14}$$

what could be simplified to

$$\tilde{y}(k) = \mathbf{w}_t^T(k) \cdot (\mathbf{x}(k) - \bar{y} \cdot \vec{1}) + \bar{y}. \tag{15}$$

The adaptation rule for such an adaptive filter could be obtained in the same way as filter equation (15) from (5) and (11) as follows

$$\Delta\mathbf{w}_t(k) = \frac{\mu}{\sigma_y^2} \cdot e(k) \cdot (\mathbf{x}(k) - \bar{y} \cdot \vec{1}). \tag{16}$$

This is still not beneficial for online filtering, becauseneed to know mean value for all the data, what is impossible during real time filtering. Because of that,propose to substitute the $\bar{y}(k)$ with mean value of input vector $\bar{\mathbf{x}}(k)$ and σ_y with σ_x. These parameters of input vectorcould obtain for every single sample just from vector $\mathbf{x}(k)$. That means that the input vector will be centered

$$\mathbf{x}_c(k) = \mathbf{x}(k) - \bar{\mathbf{x}}(k) \cdot \vec{1}. \tag{17}$$

This usability suggestion is based on following assumptions

$$\bar{y} \approx \bar{\mathbf{x}}(k) \wedge \sigma_y \approx \sigma_x. \tag{18}$$

Now the equation for online centered adaptation looks

$$\Delta\mathbf{w}_t(k) = \frac{\mu}{\sigma_y^2} \cdot e(k) \cdot \mathbf{x}_c(k), \tag{19}$$

and the filter equation stands as follows

$$\tilde{y}(k) = \mathbf{w}_t^T(k) \cdot \mathbf{x}_c(k) + \bar{\mathbf{x}}(k). \tag{20}$$

The general stability criteria can be obtain from (19) and (17) as

$$|1 - \frac{\mu}{\sigma_y^2} \cdot \mathbf{x}_c(k)^T \cdot \mathbf{x}_c(k)| \leq 1. \tag{21}$$

The NLMS algorithm is already using the learning rate normalization (8) according to power of input. For that reason there is no need to normalize the learning rate furthermore according to power σ_x. This simplification decrease the error caused by $\sigma_x \neq \sigma_y$. Finally the proposed learning rule could be described as follows

$$\Delta\mathbf{w}(k+1) = \frac{\mu}{\epsilon + \mathbf{x}_c(k)^T \cdot \mathbf{x}_c(k)} \cdot \mathbf{x}_c(k) \cdot \mathbf{w}(k). \tag{22}$$

3. Experimental analysis

3.1. Noise removal performance

Three different time series contaminated with white noise are used For experimental analysis of noise removal. First used time series is sum of two sinus waves according to equation

$$y(k) = \frac{a}{2} \cdot \sin\left(\frac{k}{21}\right) + a \cdot \sin\left(\frac{k}{36}\right) + b, \tag{23}$$

where a is amplitude and b is an offset. Second used time series is chirp signal generated according the equation

$$y(k) = a \cdot \sin\left(\frac{k}{44 - 5 \cdot \sin\left(\frac{k}{500}\right)}\right) + b. \tag{24}$$

The last time series is discrete Mackey-Glass system with parameters causing chaotic behaviour

$$y_m(k+1) = 1.79 \cdot y_m(k) + \frac{0.2 \cdot y_m(k-23)}{0.8 + y_m(k-23)^{10}}. \tag{25}$$

The data are scaled using parameters a and b To demonstrate the advantages of the proposed method on badly scaled data

$$y = a y_m + b. \tag{26}$$

where a is scale and b is offset as in the previously introduced time series. An example of generated time series is in Fig. 1.

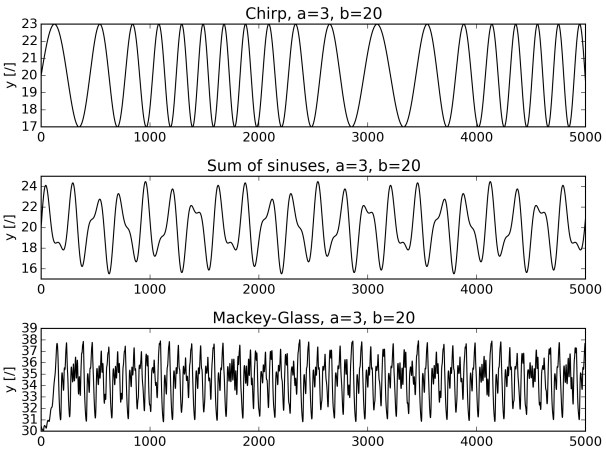

Fig. 1. *Time series used for experimental analysis, described by (23), (24) and (25).*

For every tested equationgenerated multiple time series with different parameters a, b and constant length $N = 10000$ of samples. First half of this series was used for filter training and second one was used for testing, so error was measured only on second half of the data. The noise that was used for the data contamination is white noise with mean value in 0 and standard deviation of 4. The parameter μ was tested in range from 0.05 to 1.8 with step 0.05 and as the final result is presented only the best performance achieved. The regularization parameter ϵ was set to 1 for all simulations. The length n of the used filers was set to 20. As the filter performance criteria was used MSE evaluated according to filter output and signal without noise. The MSE of noisy and original signal should be approximately 16 with the noise whatdescribed earlier. The results of experimental analysis are summarized in Table 1. As expected the NLMS with offline z-score data has the lowest MSE. The second best algorithms was the online centered NLMS in all cases where the offset $b \neq 0$.

Table 1. *Results of filtering performance analysis*

Data			**MSE** (noise MSE ≈ 16)		
Signal	**a**	**b**	**Online centered**	**Offline z-scored**	**Plain**
chirp	50	-100	4.633	6.4	14.774
chirp	50	100	5.793	7.961	8.298
chirp	5	-20	1.467	1.475	3.108
chirp	5	0	2.19	1.497	1.628
chirp	5	20	1.713	1.525	2.74
mackay	50	-100	17.253	15.122	28.65
mackay	50	100	24.151	16.591	56.222
mackay	5	-20	2.443	1.737	4.038
mackay	5	0	3.197	1.793	2.384
mackay	5	20	2.78	1.789	4.482
sinus	50	-100	6.851	6.875	12.497
sinus	50	100	8.372	7.188	14.383
sinus	5	-20	2.108	2.125	3.326
sinus	5	0	2.742	2.028	1.992
sinus	5	20	2.125	2.011	2.984

3.2. ECG signal prediction

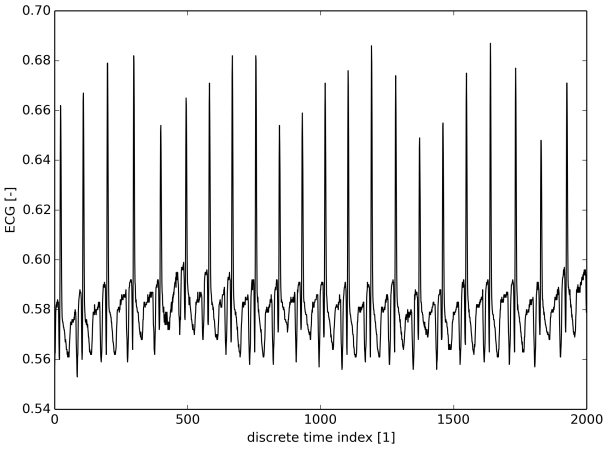

Fig. 2. *The ECG record used for experimental analysis.*

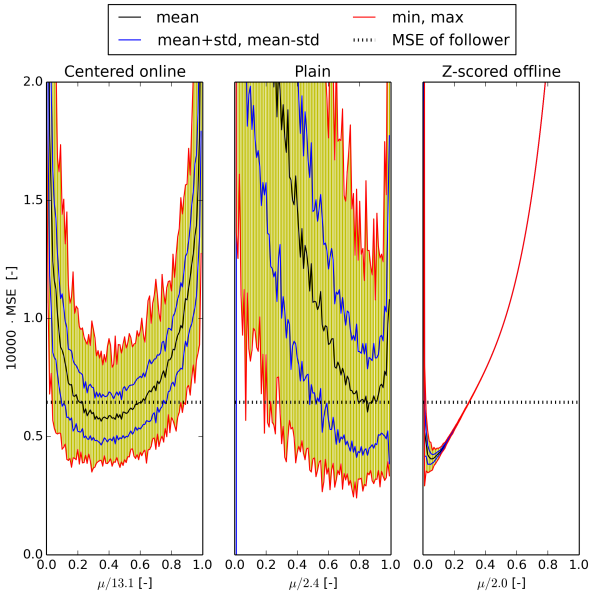

Fig. 3. *Diagram of MSE dependency on default learning rate μ during ECG prediction (5 epochs of training).*

In this analysis a real measured ECG signal is used. The signal is displayed in Fig. 2. The goal was one sample ahead prediction. The criteria was to achieve the lowest MSE of prediction. All three methods was compared. As the input of the filter was used the vector of 10 last samples and bias (bias=1). The regularization term ϵ was set to 1 and the adaptive weights was chosen randomly in range from -0.5 to 0.5. Because the initial adaptive weights are chosen randomly at the beginning, it was done 100 simulations for every tested setting of learning rate μ. The length of used ECG signal was 2000 samples. First half of data set was used for training and the second half of data was used after that for testing (MSE was calculated only on this part of the data).

The results of this analysis with 5 epochs of training are in Fig. 3 and the results of 10 epochs training are in Fig. 4. From this results is possible to suggest values for default learning rate μ to achieve best performance.

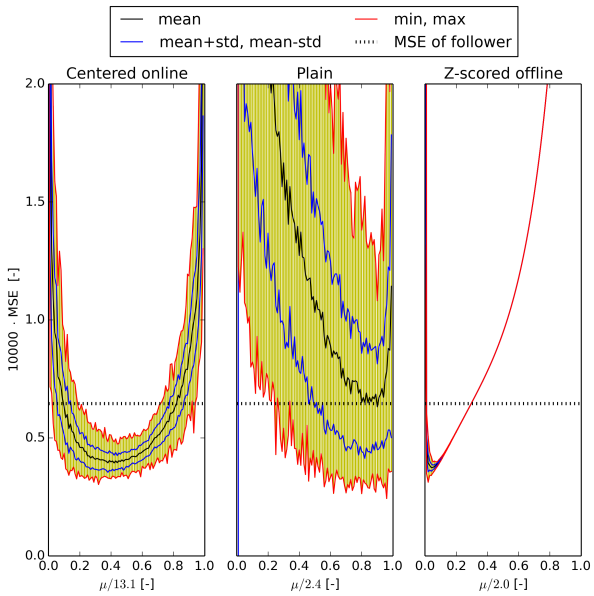

Fig. 4. *Diagram of MSE dependency on default learning rate μ during ECG prediction (10 epochs of training).*

3.3. Convergence study on ECG signal

The optimal learning rates founded in previous subsection are used as settings for this convergence study. The used signal was the previously used ECG record. For every epoch of training 10 simulation runs was made and average of it was used of the result to decrease the influence of randomly chosen initial weights. The results of this analysis are displayed in Fig. 5. From the resultscan see, that plain NLMS reach best performance during first epoch. The other two methods reach much lower MSE. After eight epochs the proposed methods with the online centering reach the similarly low value of MSE as the method with offline z-scored data.

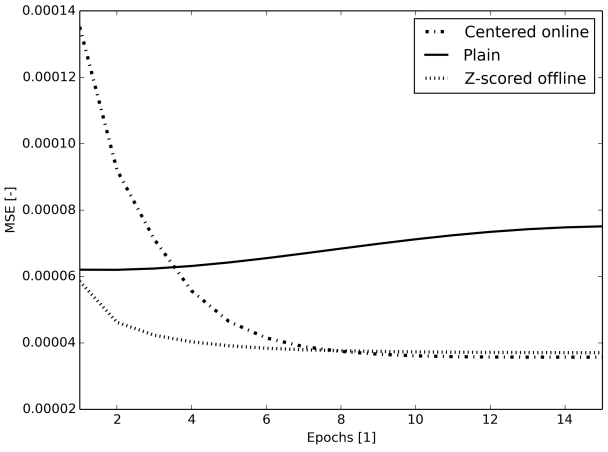

Fig. 5. *The MSE of ECG prediction as a function of training epoch with learning rate optimized for 5-10 epochs.*

3.4. Computational complexity

The computational complexity of NLMS is $O(n)$, where n is amount of operations. It contains division, so it has more operations than LMS, but the time complexity is still linear (division with fixed length - arbitrary precision is not used). In case of our modification, the time complexity is increased even more, but again just with the division (just higher number of operations, but still linear time). So it does not slow down the algorithm significantly. The speed of algorithm was tested during the simulations. The implementation was done in language Python with numerical library Numpy. The NLMS with online input centering consume 3.08x more time than the plain NLMS during the simulations.

4. Conclusion

In this paper the modification of NLMS algorithm was proposed to achieve better results during the real-time adaptive filtering for real-life tasks. The method is inspired be the commonly used offline data transformations. As it is demonstrated in experimental analysis, the method cannot achieve the performance of filtering with offline prepared data, but the usage of the proposed method is still significantly advantageous in comparison with plain NLMS. The time complexity of suggested algorithm is just about 3 times higher than the one of the plain NLMS algorithm, so it is still lower than other more sophisticated methods.

Nomenclature

$\vec{1}$	vector of all ones (−)
^{T}	transposition (−)
a	data scale (1)
b	data offset (1)
k	discrete time index (1)
$\mathbf{w}$	vector of adaptive weights (−)
$\mathbf{w}_t$	vector of adaptive weights for transformed inputs (−)
$\mathbf{x}$	input matrix (−)
$\mathbf{x}_c$	centered input vector (−)
$\mathbf{x}_t$	transformed input vector (−)
$\bar{\mathbf{x}}$	mean value of input vector (1)
y	target variable (1)
$\bar{y}$	mean value of target variable (1)
$\tilde{y}$	estimated variable (1)
σ_x	standard deviation of input vector (1)
σ_y	standard deviation of target variable (1)
μ	learning rate (1)
η	normalized learning rate (1)

References

[1] Paulo SR Diniz. *Adaptive filtering*. Springer, 1997.

[2] B Widrow, RE Kalman, and N DeClaris. "Aspects of network and system theory". In: *Adaptive Filters* (1970), pp. 563–587.

[3] Bernard Widrow et al. "Stationary and nonstationary learning characteristics of the LMS adaptive filter". In: *Aspects of Signal Processing*. Springer, 1977, pp. 355–393.

[4] Bernard Widrow and Samuel D Stearns. "Adaptive signal processing". In: *Englewood Cliffs, NJ, Prentice-Hall, Inc., 1985, 491 p.* 1 (1985).

[5] Da-Zheng Feng, Zheng Bao, and Li-Cheng Jiao. "Total least mean squares algorithm". In: *Signal Processing, IEEE Transactions on* 46.8 (1998), pp. 2122–2130.

[6] Danilo P Mandic. "A generalized normalized gradient descent algorithm". In: *Signal Processing Letters, IEEE* 11.2 (2004), pp. 115–118.

[7] Donald L Duttweiler. "Proportionate normalized least-mean-squares adaptation in echo cancelers". In: *Speech and Audio Processing, IEEE Transactions on* 8.5 (2000), pp. 508–518.

[8] Markus Rupp. "The behavior of LMS and NLMS algorithms in the presence of spherically invariant processes". In: *Signal Processing, IEEE Transactions on* 41.3 (1993), pp. 1149–1160.

[9] Moshe Tarrab and Arie Feuer. "Convergence and performance analysis of the normalized LMS algorithm with uncorrelated Gaussian data". In: *Information Theory, IEEE Transactions on* 34.4 (1988), pp. 680–691.

[10] Neil J Bershad. "Analysis of the normalized LMS algorithm with Gaussian inputs". In: *Acoustics, Speech and Signal Processing, IEEE Transactions on* 34.4 (1986), pp. 793–806.

[11] Ivo Bukovsky et al. "A Fast Neural Network Approach to Predict Lung Tumor Motion during Respiration for Radiation Therapy Applications". In: *BioMed research international* 2015 (2015).

[12] Ivo Bukovsky et al. "Learning entropy for novelty detection a cognitive approach for adaptive filters". In: *Sensor Signal Processing for Defence (SSPD), 2014*. IEEE. 2014, pp. 1–5.

[13] Matous Cejnek et al. "Adaptive polynomial filters with individual learning rates for computationally efficient lung tumor motion prediction". In: *Computational Intelligence for Multimedia Understanding (IWCIM), 2015 International Workshop on*. IEEE. 2015, pp. 1–5.

[14] Matous Cejnek, Ivo Bukovsky, and Oldrich Vysata. "Adaptive classification of EEG for dementia diagnosis". In: *Computational Intelligence for Multimedia Understanding (IWCIM), 2015 International Workshop on*. IEEE. 2015, pp. 1–5.