Speech Denoising Based on Normalized Least Mean Squares Filter

Pengyu Zhang, Graduate Student, University of Florida

***Abstract*—The Least Mean Squares（LMS）and its derivative algorithms have been widely used in speech signal denoising. This project implements Normalized Least Mean Squares (NLMS) filter which is an adaptive filter for noise cancellation. The mathematic theory of the algorithm is stochastic gradient descent with normalization to ensure least disturbance and converges of the algorithm. Frequency response and Signal-to-noise ratio (SNR) are used for evaluating the performance of NLMS. Best filter orders and corresponding step size, as well as evaluation of mis-adjustment obtained from the given signal, are presented through experiments. Problems related to non-stationary signals when applying NLMS are discussed.**

***Index Terms*—Adaptive Filter, Normalized Least Mean Squares, Speech Denoising, Non-stationary Signal.**

I. Introduction

Least Mean Squares (LMS) has been proved an efficient method of dealing with noise cancellation since the 1960s. The method of LMS is stochastic gradient descent focus on finding the nearest minimum on the estimated performance surface obtained by mean square error (MSE) [1]. One of its variants, Normalized Least Mean Squares (NLMS), is more widely used because it's been shown that the NLMS can utilize different step sizes based on time-varying inputs [2] which provide NLMS faster speed of convergence and smallest disturbance during the process of weight Iteration.

In this project, I deploy the NLMS on the given two-channel speech signals collected separately by two microphones in one noisy room. The first channel signal (d(n)) contains speech and noise information is captured by one microphone on the table. The second channel signal (n(n)) is collected very close to the source of the noise. The block diagram in Fig. 1 shows the system I implement on the denoising project, where the input signal n(n) and desired signal d(n) are directly imported from the given two-channel audio signal. The unknow y(n) is the output after implementing the adaptive filter H(n) on n(n) by matrix multiplication. The adaptive filter H(n) attempts to estimate the output y(n) for approximating to desired signal d(n). The error e(n) is obtained by subtracting y(n) from d(n). Then the e(n) is used for parameter updating through NLMS algorithm and e(n) is exactly the speech signal which I aim at separating from the noise. The NLMS algorithm is mathematically given as formula (1) and formula (2):

$$e(n) = d(n) - \boldsymbol{H}(n)^T \boldsymbol{n}(n) \tag{1}$$

$$\boldsymbol{H}(n+1) = \boldsymbol{H}(n) + \frac{\eta}{\| n(n) \|^2 + \delta} e(n)\boldsymbol{n}(n) \tag{2}$$

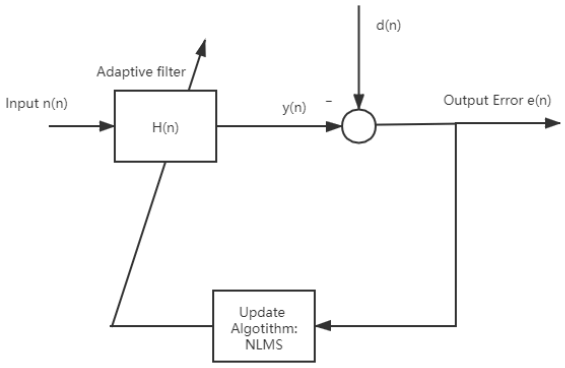

Fig. 1. Block diagram of NLMS Adaptive Filter

where $\mathbf{n}(n) = [n(n), n(n-1), n(n-2),\ldots, n(n-L+1)]^T$. η is the learning rate and δ is a constant that regularizes the denominator in case of being zero. The value of δ is 1 in my experiments.

The second section explains the design of my experiments. The third section focuses on analysis of experiments. The fourth section concludes results produced by experiments, followed by demonstrating some problems when dealing with convergence of the algorithm in non-stationary signals.

II. Experiments Design

A. 2-tap Filter

I start estimation from a 2-tap filter with 0.01 as the value of the learning rate η. Then provide the result of:

- Weight tracks
- Learning curve
- Short-Time Fourier Transform (STFT) as the frequency response of both the d(n) and e(n)
- Signal-to-noise ratio (SNR) evaluated by formula (3):

$$SNR = 10\log\left({E[d^2]}/{E[e^2]}\right) \tag{3}$$

The performance surface contour is also shown under the condition of filter order 2 to provide better visualization.

B. n-tap Filter

Repeating the aforementioned procedures with the increasing

of filter order and obtain the four results mentioned above respectively for further analysis. The filter performance with the change of learning rate is evaluated by SNR curve, and the misadjustment is provided for each experiment.

III. Experiments Result

A. Performance surface of 2-tap filter

In Fig.1 the contours of performance and the weight values demonstrate that the performance surface e(n)^2 defined by the input signal is a quadratic function of the weights of the filter.

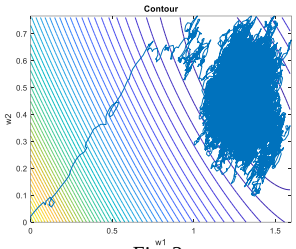

Fig. 2

B. Misadjustment of n-tap filter

The evaluation of misadjustment defined by formula (4):

$$M = \frac{L+1}{4\delta_{MSE}} \quad (4)$$

where the numerator is the number of weights of the filter and the denominator is the average of mean square error. Table I shows the misadjustment of 10 different filters.

TABLE I

Misadjustment	MSE	Filter Order
0.4761	1.0502	2
6.9328	0.2524	7
19.3241	0.1035	8
29.2151	0.0770	9
32.4890	0.0769	10
35.0552	0.0784	11
44.2094	0.0848	15
48.1336	0.0883	17
215.4801	0.0928	20
276.4290	0.1085	30

When the filter order is in the range of 2 to 10, the MSE keeps decreasing while filter order increases which contributes to the increasing of misadjustment. Whereas the MSE decreases when the filter order increases over 10, and the increasing of filter order at the same time jointly contribute to the continuously increasing of misadjustment. The results from Table I indicate that the appropriate filter order is 10.

C. Weights track, learning curve, frequency response, SNR curve

The weights track, learning curve, frequency response, and SNR curve are presented in Fig.2 to Fig.7 for filter order 2, 7, 8, 9, 10, 15 separately. The four evaluations in each figure are displayed from left to right in order.

It's apparently to see that the smaller value of order causes underfitting from the frequency response because the speech signal is still polluted by multiple frequencies with high energy. With the number of orders increasing to 10, more parts of noises are canceled. But it starts overfitting when the order is greater than 10 because some of the noises whose frequencies are approximately in the range of 6KHz to 8KHz appear again if comparing the frequency response in order of 10 with that in order of 15.

The weights track can also explain the trend from underfitting to overfitting with increasing of filter order. The weights are unstable which means the filter performance is not good under a small number of orders until the number of orders increases to 8. When the filter orders are greater than 10, the values of redundant weights concentrate near the zero and the weights which contains truly information we need to remove.

It's also clear to observe that the filter performance keeps increasing as long as the number of order increase to 10 but not greater than 10 if taking the learning curves and SNR curves into consideration.

D. The best filter order

In Table II, I compared 10 SNR under 10 different filter orders. It can be observed the SNR becomes larger before the filter order increases to 10 and has the largest value when the filter order is exactly 10. Whereas SNR starts decreasing with the continuously increasing form10 of filter order. The trend can strongly prove that the most appropriate filter order is 10.

TABLE II

SNR.	Filter Order
13.7970	2
28.0537	7
36.9693	8
39.9248	9
39.9333	10
39.7405	11
38.9590	15
38.5578	17
38.0586	20
36.4948	30

E. Filter performance vary with step-size

The best filter order is decided from the former results. From formula (2), the step-size is mainly controlled by learning rate. So, I use SNR curve as the evaluation of filter performance concerning learning rate. In the left figure of Fig.8, I compare SNR obtained in 9 different learning-rate from 0.001 to 0.025 at step of 0.003. It's obvious to observe that I obtained better convergence when increasing learning-rate to 0.025 although the best comprehensive performance (frequency response, learning curve, etc.) is obtained when learning rate is 0.01.

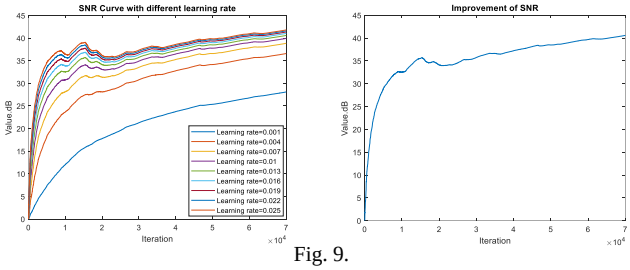

Fig. 9.

After several updates on learning rate and related comparison, the best SNR curve is obtained when setting 0.013 as the value learning rate. The corresponding SNR curve is shown in the right figure in Fig.8

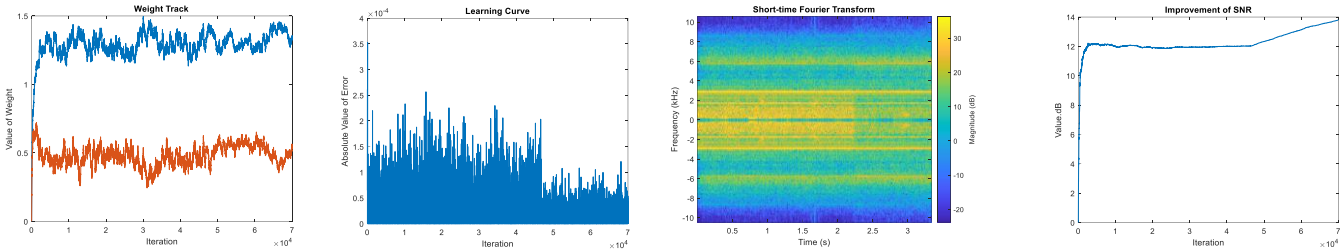

Fig. 3. Order 2

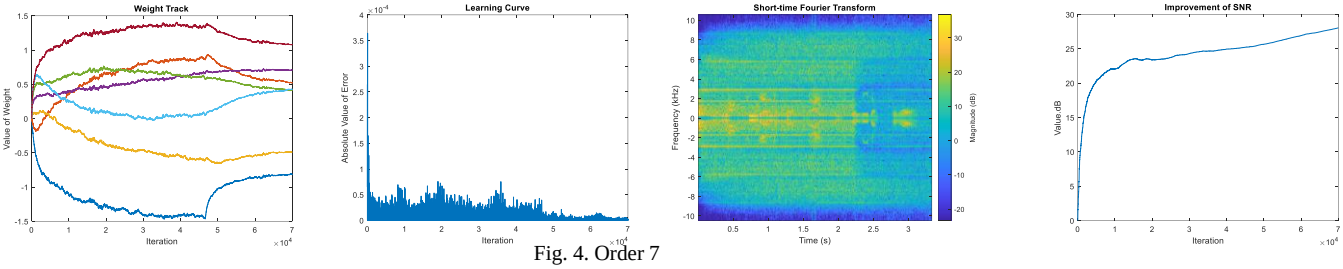

Fig. 4. Order 7

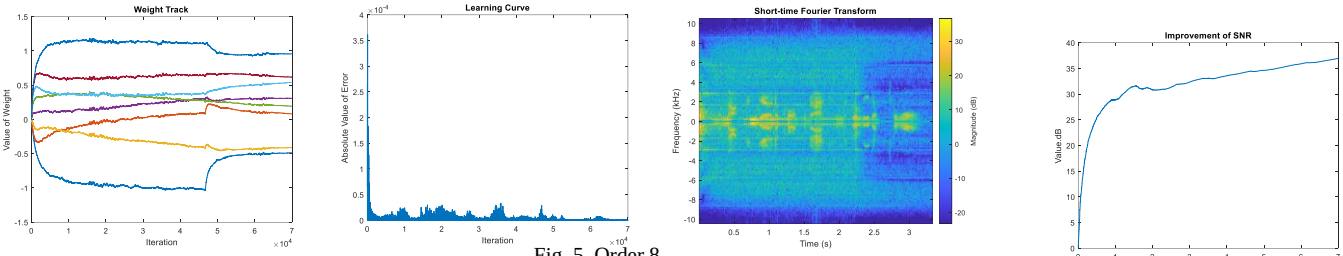

Fig. 5. Order 8

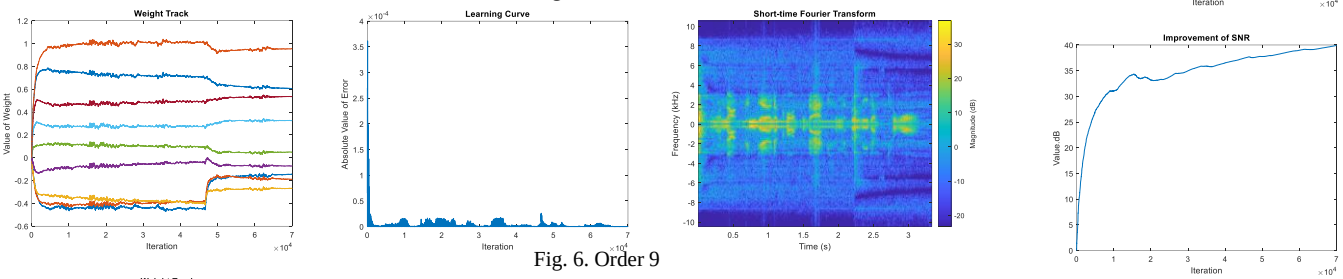

Fig. 6. Order 9

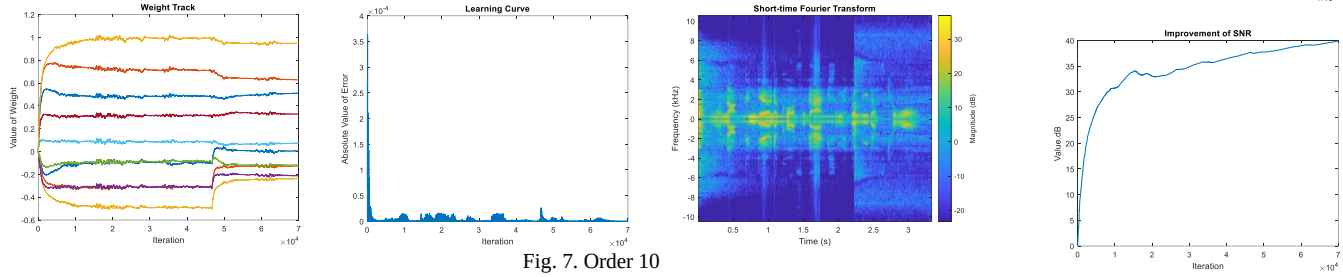

Fig. 7. Order 10

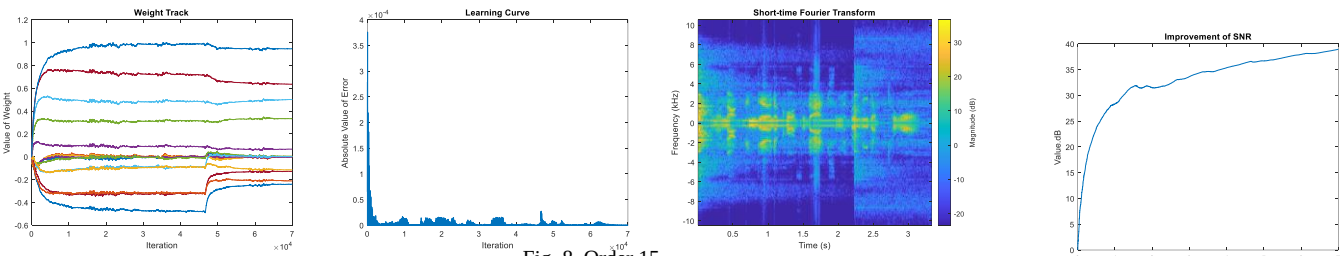

Fig. 8. Order 15

IV. Conclusion and Discussion

A. Conclusion

In this project, I implement the NLMS algorithm on two-channel audio signals denoising. I compare the performance of the adaptive filter by evaluating weights track, learning curve, frequency response, and SNR curve. The work mentioned above helps to decide the best number of filter orders which is 10 as well as the learning rate which is 0.013 that provides the best result of convergence. The comparison between different learning rates also verifies that the algorithm can converge in a range of learning rates, but the optimal learning rate is half of the maximum learning rate that can ensure the convergence.

B. The problem in non-stationary situation

One of the significant problems is the performance of NLMS on non-stationary signals. The problem is distinguished from learning curve which evaluates the extent of the dissimilarity between error and desired signal. If the signal is stationary, or as least part of it is stationary, I should expect a smooth learning curve because that is the goal of stochastic gradient descent. But it's crystal clear that the learning curve is not as smooth as I expect.

Problems begin with the original signal. n(n) is stationary because it's directly produced by the noise generator whereas d(n) is the mixture of speech signal and noise signal which only has spatial differences with n(n). After all, n(n) is collected by a microphone that is very close to the noise source. Therefore, the only non-stationary part is the speech signal. Looking at the frequency, the speech signal normally consists of low frequency and it's not stationary in the range of the whole signal. And the changes of syllable reflected by frequency response own the same properties with learning curve. The intermittent peaks in learning curve show when these syllables change which directly causes large error.

One way to implement NLMS in non-stationary signal is windowing. Using windows to separate non-stationary signal into numbers of short signals to ensure every short signal can be approximated to stationary signal. Then the peaks in learning curve can be smoothed.

References

[1] Weisstein, Eric W. "Method of Steepest Descent." From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/MethodofSteepestDescent.html

[2] D. T. M. Slock, "On the convergence behavior of the LMS and the normalized LMS algorithms," in IEEE Transactions on Signal Processing, vol. 41, no. 9, pp. 2811-2825, Sept. 1993.