Interference Canceling Using Normalized LMS Algorithm (October 2016)

Kumar Rohit Malhotra

***Abstract*— Normalized LMS is a variant of the Least Means Square algorithm with a data dependent learning rate. This paper explains how we can use normalized Least Means Square algorithm for the purpose of building an FIR filter for noise cancellation.**

***Index Terms*—FIR Filter, Normalized Least Mean Square, Interference cancellation, Machine learning**

I. Introduction

THIS paper presents the application of normalized Least Means Square (LMS) algorithm for building an adaptive FIR filter for noise cancellation. LMS algorithms are a class of adaptive filter used to mimic a desired filter by finding the filter coefficients that relate to producing the least mean squares of the error signal (difference between the desired and the actual signal). It is a stochastic gradient descent method in that the filter is only adapted based on the error at the current time [1]. The main drawback of the "pure" LMS algorithm is that it is sensitive to the scaling of its input. This makes it very hard (if not impossible) to choose a learning rate that guarantees stability of the algorithm [2].

The Normalized least mean squares filter (NLMS) is a variant of the LMS algorithm that solves this problem by normalizing with the power of the input. In this paper, we design and evaluate the performance of an adaptive filter using the NLMS algorithm. We use this algorithm for interference cancelling. For this purpose, we have two signals: one only with the machine noise, and the other one with the machine noise and the actual signal, and try to extract the actual signal from the second signal. We start by taking a filter order of 2 and find the most optimum learning rate and filter coefficients. We evaluate the correctness of our result using performance surface contours for various filter coefficients and the weight tracks of the filter coefficients. We experiment with different learning rates and get their learning curves interpreting the correctness of the chosen learning rate. Then we train the model for other filter orders and, choose the best filter order and corresponding filter coefficients and learning rate on the basis of least value of Mean Squares Error. Once again, we interpret the correctness of selected learning rate by comparing it with other learning rates. The actual sound signal is obtained by taking the error from the filter order and corresponding optimum filter coefficients. Since the actual signal is not correlated to the noise signal, the filter will only be able to replicate the noise signal in the output, and hence the error will give us the actual signal.

This paper was submitted on Oct 18, 2016.

Kumar Rohit Malhotra is with the Department of Computer and Information Science and Engineering at the University of Florida, Gainesville, FL 32611 USA (e-mail: krohitm@ufl.edu)

II. Experiment for filter order 2

We begin the experiment using a 2-tap filter. We have data for two signals in discrete form. The two signals are the noise signal (reference signal) and the noise plus actual sound signal (primary signal). In our experiment, the reference signal is used as an input signal and the primary signal as the desired signal. The model will be typically as shown in Fig. 1.

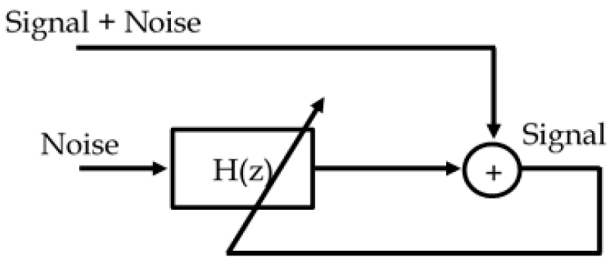

Fig. 1. Diagram of an interference canceling system. The system takes the noise signal as input, and signal + noise as the desired signal. The error obtained from the system is the signal.

Both the signals are split into 2 parts i.e. training, and testing. This can be done by taking various subsets of data for training and testing, and then cross-validating the results for the best parameters. The NLMS algorithm can be summarized as:

Parameters: M = Filter Order
α = Learning Rate
Initialization: ω = zeros (M)
Computation: For n = 0, 1, 2, …

$$X(n) = [x(n), x(n-1), \dots x(n-M+1)]^T$$

$$\omega(n) = [\omega_0(n), \omega_1(n), \omega_2(n), \dots \omega_{M-1}(n)]^T$$

$$y(n) = \omega(n)^T x(n) \quad (1)$$

$$e(n) = d(n) - y(n)$$

$$\omega(n+1) = \omega(n) + (\alpha e(n) x(n) / x^T(n) x(n)) \quad (2)$$

where $\omega(n)$ is the set of filter coefficients at any time index n, $y(n)$ is the filter output at time index n, and $e(n)$ is the error

at that time. The term $x^T(n)x(n)$ normalizes the learning rate α at every update of $\omega(n+1)$.

A. Training and testing the model

The model is initially trained for filter order 2 and learning rates ranging from 0.001 to 0.1, with an increment of 0.001. For each learning rate, the algorithm given above is implemented and the corresponding optimal filter coefficients are obtained over the training data. In order to avoid overfitting of the model, we choose varying sizes of training subset and test subset. In this experiment, we have taken the subsets as:

(i) First 60% of data as training data set and the rest for testing

(ii) First 75% of data as training data set and the rest for testing

(iii) Last 60% of data as training data set and the rest for testing

(iv) Last 75% of data as training data set and the rest for testing

As such, we cross validate the results obtained from the various data set intervals, and take the best learning rate and the filter coefficients obtained in terms of minimum Mean Squares Error. The idea behind this is that the filter would have already learnt the efficient filter coefficients in the training set, and if it performs fair enough in testing set as well, then that learning rate is optimum for the filter.

The training begins by initializing the weights to 0, and then implementing the algorithm for NLMS on the samples from the training data iteratively. By iteration, we mean that the filter coefficients would be updated for each sample $X(n+1)$ as per the error for the previous input $X(n)$. As the training goes on, the filter updates its coefficients on the basis of error and by normalizing the learning rate for each iteration.

Once we have the filter coefficients and learning rates from the training data, we apply those parameters on the testing data. As said above, we select the most optimum learning rate in terms of minimum MSE that we get from testing data. The learning rate and filter coefficients are finally used in the actual model. MSE can be calculated using the following formula:

$$MSE = (1/n)\sum_{i=1}^{n}(d_i\text{-}y_i)^2 \qquad (3)$$

where d_i is desired signal at instance i, and y_i is the filter output at instance i.

B. Performance Surface Contour and Weight tracks

For the chosen learning rate, we can verify the behavior of our algorithm using Performance Surface Contour with weight tracks. This also validates the optimality of the chosen filter coefficients. In a performance surface contour for a 2 tap filter, the mean square errors for various values of filter coefficients are plotted as contours against the filter coefficients as the two axes. We can draw a performance contour by taking the values around the optimal filter coefficients, ω^*, and finding the MSE for all of those filter coefficients. The weight track can be drawn from the weight updates that we got while training the model. The performance contour obtained in our experiment can be seen in Fig. 2.

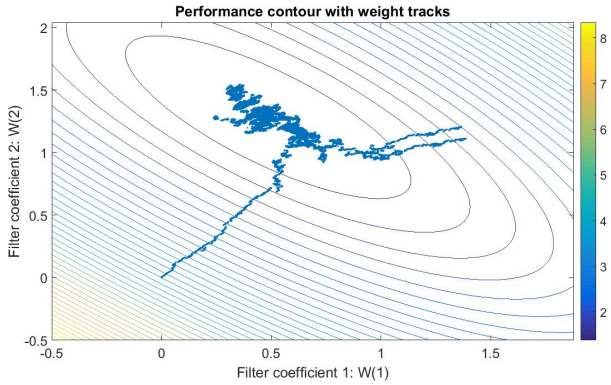

Fig. 2 Performance Surface Contour for 2 Tap FIR Filter

There are a few observations that can be made from the graph. The filter coefficients selected in part A fall within the contour depicting the minimum MSE. Hence, the chosen filter coefficients will give the minimum possible MSE for 2 tap filter.

It can be seen that as the graph approaches minimum value of MSE, the gaps between the contour bands keep decreasing. This implies that the MSE curve becomes more and more flat as it approaches ω^*. The tighter contours for other values of filter coefficients suggest steepness of the graph in those areas and hence, rapid change in MSE for slight changes of filter coefficients there.

Another inference that can be made is that since the minimum MSE for 2 tap filter is obtained for a high range of filter coefficients, the designed filter is not strong enough to give the best results. In other words, the filter needs to have more taps in order to predict the signal better, with lower MSE.

An additional observation to be made in this case is that the weight tracks don't simply converge to the optimum value, but rather shift suddenly in between, before once again reaching their optimum value, as can be seen in Fig. 3 for weight tracks as well. This may happen because of the signal being non-stationary. In other words, the source of the signal or the listener might be moving, which results in the sudden change in the filter function.

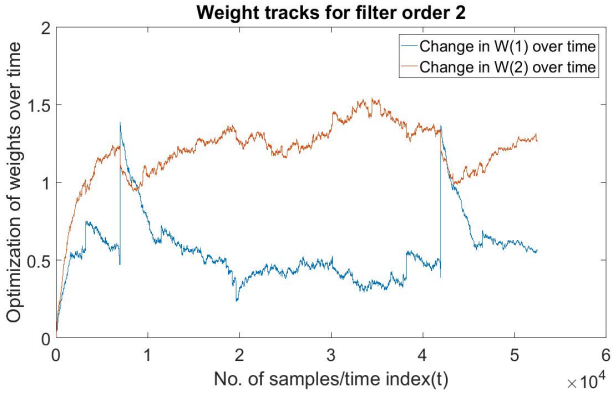

Fig. 3. Weight tracks for filter order 2. The graph shows the change in tap weights W(1) and W(2) as the models is trained on more and more samples.

C. *Learning Curve for various learning rates*

It can be shown that an optimum learning rate has been chosen, by plotting the learning curve of the selected learning rate against some other learning rates. A learning curve will show the prediction error, or the difference between the desired signal and the filter output, at every instance. We can choose two other learning rates, one bigger and one smaller than the chosen learning rate, and plot the learning curve for each one of them, depicting how the model learns to optimize its filter coefficients. Fig. 4 shows the learning curve for three learning rates. 0.001 is the chosen learning rate, and as can be seen from the graph, the error becomes for that learning rate. For the other learning rates, the error is only increasing with time.

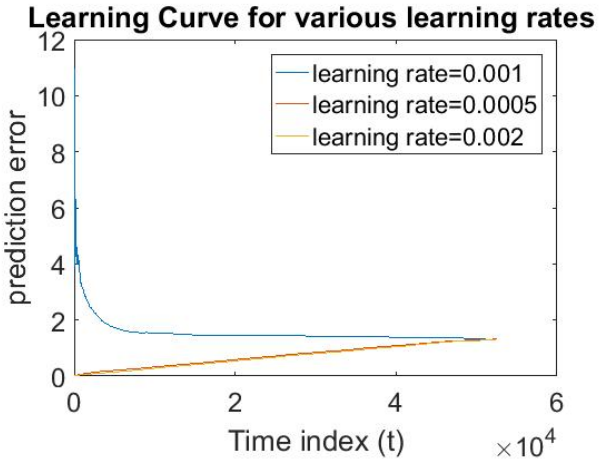

Fig. 4. Learning Curve for various learning rates

For the chosen learning rate, the filter converges at around 500 samples. The convergence of the filter can be verified by comparing the filter coefficients obtained from the selected filter order and learning rate against the optimum filter optimum filter coefficients obtained by using the formula:

$$W=R^{-1}P \qquad (4)$$

where R is the auto - correlation matrix for the training dataset, and P is the cross - correlation matrix. In our case, it has been observed that the filter coefficients are almost same as the optimal filter coefficients.

D. *Signal to noise ratio improvement*

We can check the signal-to-noise ratio of our filter using the formula:

$$ERLE = 10\log(\mathrm{E}[d^2]/\mathrm{E}[e^2]) \qquad (5)$$

where *ERLE* means Echo Return Loss Enhancement. In order to get this value, we can take the squared mean of the whole desired signal as $\mathrm{E}[d^2]$ and Mean Squared Error as $\mathrm{E}[e^2]$. The values obtained in our case for the Signal to Noise ratio is 15.9137. The signal to noise ratio for filter order 2 can be seen in Fig. 7. The final signal obtained, that is the so called noiseless signal, is as can be seen in Fig. 8. However, the sound of the actual signal is still not audible. Hence, we'll try with higher filter orders.

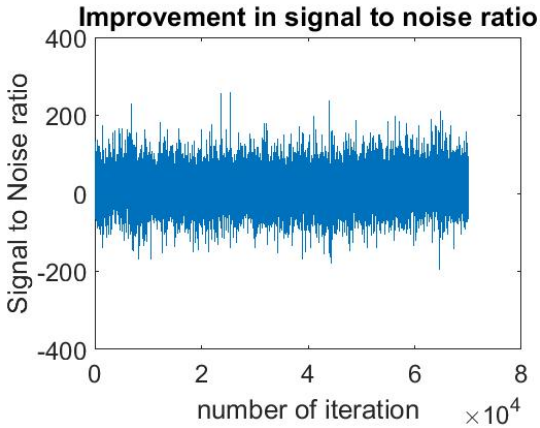

Fig. 7. Signal to noise ratio for filter order 2

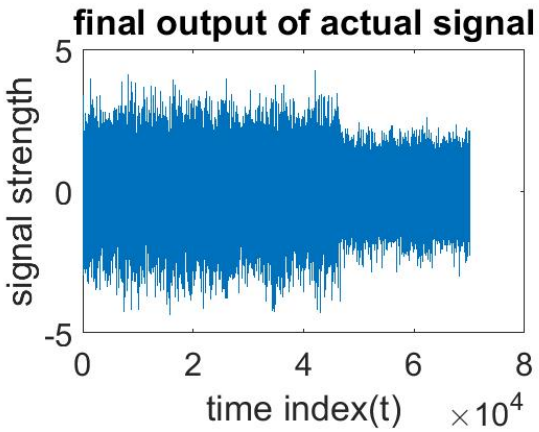

Fig. 8. Final signal obtained from filter of order 2, after removing noise

III. Experiment for higher filter orders

Since the filter order 2 does not suffice the purpose of noise cancellation in our experiment, we try to use higher filter orders. As such, we experiment with filters of order 10, 20 and 50. The methodology of training and testing, cross-validation, choosing the best learning rate, and building the final model is same as the methodology used for the filter order in part II. The results for the signal-to-noise ratio in dB by the ERLE is given in Table 1.

Filter Order	SNR
10	43.6532
20	44.1607
50	44.4914

Table 1. Sound to noise ratios in dB by the ERLE for the best learning rates for three filter orders 10, 20 and 50

Table 1 has the signal-to-noise ratio for the best learning rate in case of each filter order. The signal-to-noise ratio increases as the filter order increases. As such, the best filter order in this experiment is obtained at an order of 50. The signal-to-noise ratio for the three filter orders, for their best learning rates, can be seen in Fig. 9, 10 and 11.

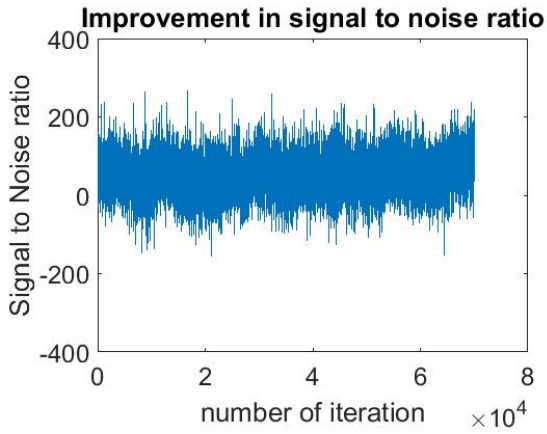

Fig. 9. Signal to noise ratio for filter order 10

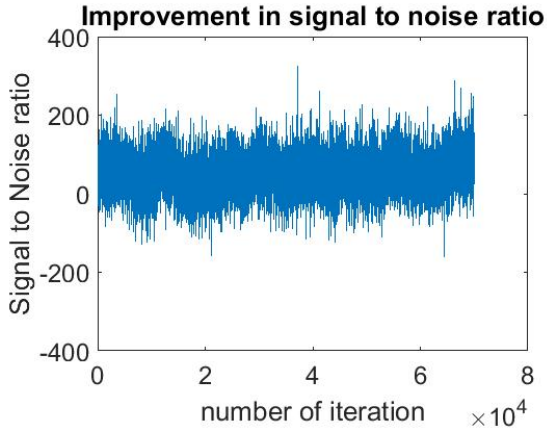

Fig. 10. Signal to noise ratio for filter order 20

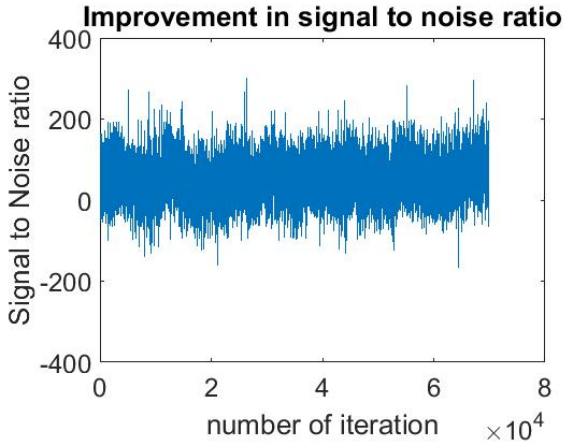

Fig. 11. Signal to noise ratio for filter order 50

We plot the learning curves for various learning rates in case of filter order 50, thereby showing that the selected filter order 0.001 in Fig. 12 is the best filter order.

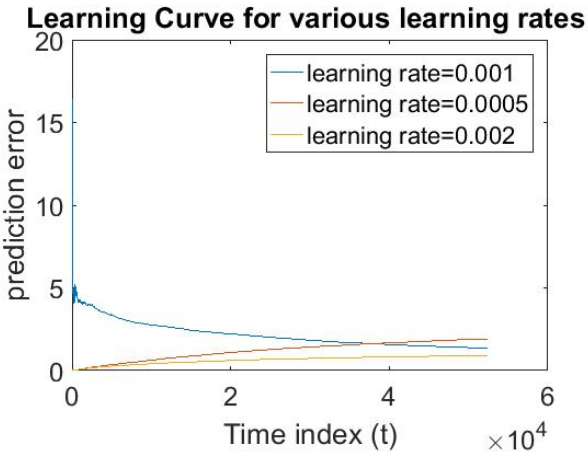

Fig. 12. Learning curve for various learning rates for filter order 50

For the other learning rates, the error is increasing.

The best filter, in terms of ERLE, agrees with our assessment of speech intelligibility. However, it misses the initial word in the signal. This happens because we are using a filter of order 50, which will start giving us the output only after it has 49 input samples. The sound of the rest of the signal is clear as it can be heard saying “I will not condone a course of action that will lead us to war”. The final signal can be seen in Fig. 15.

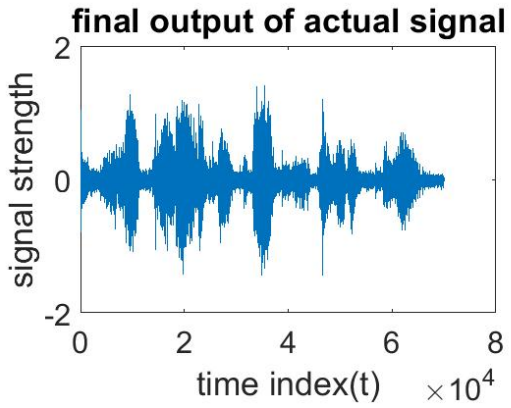

Fig. 13. Final signal obtained from filter order 50, after removing the noise with the help of the filter

IV. Conclusion

Normalized Least Means Square algorithm gives results as good as Least Squares Error algorithm, as the filter approximately converges to the optimal solution. Because of its adaptive nature, normalized LMS can be used in case of real time applications, where the filter coefficients have to adapt according to the changing environment. This is particularly useful in case of non-stationary signals, where the speaker or the listener may be actively moving.

References

[1] Wikipedia: Least Mean Squares Filter [Online]
[2] Simon Haykin, *Adaptive filter theory*, 4th ed., Upper Saddle River, N.J. : Prentice Hall, c2002, p. 870-911.