A Novel Text Dependent Speaker Recognition Model using Adaptive Online Algorithms

Akhil Perincherry

Department of Electrical and Computer Engineering, University of Florida

***Abstract*—Speech modality is constantly used in security in the field of biometrics as it is a natural way to communicate and the recognition algorithms currently employed provide reliable recognition rates. Here, the use of adaptive online algorithms such as RLS/NLMS are proposed to perform speaker recognition. The filter coefficients from these adaptive filters were seen to model the voice characteristics efficiently. To highlight the advantages of the proposed technique over the ones used currently, the model is compared with the state of the art MFCC-VQ method for speaker recognition. It was observed that RLS performs even better than the state of the art technique while NLMS although providing reasonable performance was not as good as the traditional techniques. The RLS model resulted in 100% classification accuracy at low computation time hence justifying its use. Also, RLS provided Equal Error Rates (EER) of around 10% while NLMS was seen to generate around 15% EER. Detailed comparisons and it's implications are discussed in the results section.**

***Index Terms*—RLS, NLMS, MFCC, VQ, Adaptive filter, Prediction, Biometrics**

I. Introduction

AUTOMATIC speech recognition has wide application in the field of biometrics. It is a natural and easy way for man-machine communication. Speech signals are parametrized and the feature vectors for each subject are stored in the gallery. Decisions are arrived at by comparing the feature vectors from the test set or the query to the gallery database using an appropriate distance measure. Current standard techniques for speech recognition are Mel Frequency Cepstral coefficients (MFCC) feature extraction and pattern classification using Vector Quantization (VQ) technique via the Linde-Buzo-Gray algorithm [7]. [5] uses the aforementioned MFCC-VQ technique whose steps are outlined in the Section II. Its superior performance is owing to the fact that, the technique was formulated by incorporating the prior knowledge of the working of the human acoustic system. Other techniques that are currently used for feature extraction are Linear Predictive Coding, Perceptual Linear Prediction, Neural networks ET AL. Dynamic Time Warping, Hidden Markov Models and Support Vector Machines are the state of the art classification techniques. Typically, since the speech is recorded in an uncontrolled environment, the processing needs to be robust to noise and interference. [3] and [4] uses RLS/NLMS respectively for denoising the signal in the pre-processing stage before feeding it to the above stated feature extraction-classification techniques. [3] shows an improvement of 10-15% in classification accuracy by using NLMS as a denoising algorithm while [4] shows an improvement by 6% after using Fast Recursive Least Squares (FRLS) in the pre-processing stage. It needs to be noted that, for NLMS/RLS to be used in denoising, we need to record the background noise as well along with the speech. Noise can then be modeled using adaptive filters and subtracted from the input to get the denoised waveforms.

In this paper, RLS/NLMS are proposed as a complete feature extraction-classification technique rather than merely being used as a pre-processing technique. Since it is seen that speech signal needs to have a good SNR, RLS/NLMS can be seen as potential candidates for the considered problem. Also, speaker recognition thrives by discriminating the spectral content of the different subjects and RLS/NLMS performs a good job modeling the frequency information. Therefore, it is logical to use these online learning algorithms in the manner proposed in the upcoming sections. Moreover, the non stationarity of the speech can be well defined by the RLS owing to a dedicated free parameter known as the Forgetting Factor, which is elaborated in the next section. This proposed technique has not been implemented before but is seen to work better than the state of the art MFCC-VQ technique for the datasets considered here.

The LMS filter involves a filtering stage and an adaptive stage. The filtering stage involves computing the output of a filter to the input and estimating the error while the adaptive stage adjusts the filter weights aiming to minimize the error in a feedback loop. The normalized LMS (NLMS) is employed to alleviate the problem of input scaling of LMS by dividing the the normal LMS update equation with the input power. In addition, a regularization parameter is added to avoid divide by zero condition. It is to be noted that the LMS never gives the optimum solution (Wiener solution) but it fluctuates around the optimum solution marginally depending on the step size. NLMS updates the weights in a direction opposite to the gradient vector updating the weights such that they lead to minimum MSE. A large step size would mean less convergence time but also that the updation wouldn't be too stable around the optimum. A high updation step could reverse the gradient direction back and forth resulting in increased vacillations. A low step size might take a longer convergence time, hence an optimum step size has to be arrived at using this inherent tradeoff.

The Recursive least squares (RLS) is an adaptive filter which recursively estimates the coefficients by minimizing a weighted linear least squares cost function relating to the input signals. Its rate of convergence is an order faster than the LMS filter because the RLS filter whitens the data by using the inverse correlation matrix of the data although, this

also results in increase of computational complexity. The μ parameter in LMS or the step size is replaced by the inverse of the correlation matrix of the input vector which whitens the tap inputs. Also, the rate of convergence of the RLS algorithm does not vary with condition number of the ensemble average correlation matrix of the input vector. The forgetting factor in RLS helps tune the filter with respect to the environment being stationary or non stationary. The excess mean square in RLS asymptotically converges to zero.

A typical adaptive filter predictor model is shown in Figure 1. Since it's a predictor model, the input is the current sample and the desired is the next future sample. The filter models the system by minimizing the cost function at each sample.

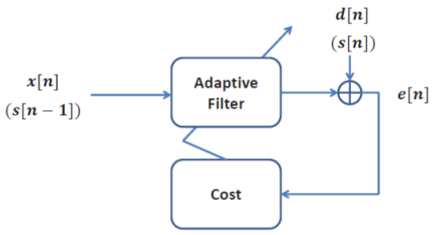


Figure 1. Adaptive predictor model

The paper is organized as follows. The theory and the underlying mathematical formulations behind the 3 algorithms are discussed in Section II. The implementation details including the dataset and the algorithm are explained in Section III. The results demonstrating the model's efficacy are elaborated in Section IV i.e. the filter trajectories, the predicted responses, the classification accuracy ET AL. Finally, the paper is concluded and future avenues are explored in Section V.

II. Theory

In adaptive filtering framework, the aim is to find a filter that minimizes the mean squared error between the desired $d(n)$ and the filter output $y(n)$. Finding FIR filter coefficients w that minimizes the squared error i.e.

$$min_w \;\; E\left[e^2(n)\right] \tag{1}$$

where $E[.]$ denotes the statistical expectation boils down to a solution of linear system of equations known as the Wiener Hopf equations (2).

$$w = R^{-1}p \tag{2}$$

where,

$R = E[x[n]x^T[n]]$ is the autocorrelation matrix of the input signal.

$p = E[d[n]x[n]]$ is the cross correlation vector between the desired and input signal.

Solving these set of linear equations is not computationally feasible as taking matrix inverse is $O(n^3)$ which is exactly what Wiener Filter does. NLMS and RLS approximates this procedure in a computationally feasible manner.

RLS finds the Wiener solution at every sample recursively using the matrix inversion lemma i.e. matrix inverse is not computed explicitly but is updated recursively. To form these estimates, it uses the past set of samples weighted in an exponential manner which is controlled by the Forgetting Factor parameter. This parameter controls the impact of previously observed data on the current estimates of the autocorrelation matrix and cross correlation vector. The relevant equations are discussed in section II-A.

NLMS unlike the RLS computes the sample estimates at every instant via a gradient descent approach. Since linear filters have their cost functions described by a convex function, they will have one optimum solution or the global minima and consequently, gradient descent has an easier job. Solution moves along the direction of the negative gradient of the cost function at every instant. The amount by which the weights are updated depends on the step size parameter or μ. Since the estimates are formed at every sample, NLMS estimates vacillates around the optimum and never truly achieves the optimum generally. The computation time is an order faster than that of RLS but the convergence time is an order higher in NLMS compared to RLS.

A. RLS Equations

The RLS algorithm is a system of initialize-update equations that solves the Wiener Hopf equations in a computationally efficient manner. The cost function that it is considered is the Least square error of the estimate. Let $order$ be the filter order, α be the forgetting factor, d be the desired signal.

1) Initialize the weight vector of length $order$ to zeros and the inverse autocorrelation matrix P to a diagonally loaded matrix - loading factor of 1000 (heuristically) was chosen here.
2) Consider the first $order$ number of elements of the input to get the input vector u.
3) Find the instantaneous output using the current weight vector and the input considered in the above step.
4) Compute the instantaneous error.
5) Use the above result to update the weight vector as in (3).
6) Slide the input used in step 2 by one sample and repeat the whole procedure.

Initialization equations:

$$w_{est}(0) = 0$$

$$P(0) = largeNumber * IdentityMatrix$$

The update equations are as follows:

$$t(n) = P(n-1)u(n)$$

$$k(n) = \frac{t(n)}{\alpha + u^H(n)t(n)}$$

$$e(n) = d(n) - w_{est}^{H}(n-1)u(n)$$

$$w_{est}(n) = w_{est}(n-1) + k(n)e(n) \tag{3}$$

$$P(n) = \alpha^{-1}P(n-1) - \alpha^{-1}k(n)u^{H}(n)P(n-1)$$

The forgetting factor α implies the memory of the learning algorithm, it implies how the past input samples are weighted. Here, the αvalues are generated in such a way that the exponential window is halved at previous L samples i.e.

$$\alpha = 0.5^{\frac{1}{halfSample}}$$

If window is to be designed such that the exponential window is half at the 500th sample, $\alpha = 0.5^{\frac{1}{500}} = 0.9986$ and so on.

In a nutshell, RLS is basically solving Wiener Hopf system of equations recursively and with an exponential window.

B. NLMS Equations

The NLMS algorithm is again a system of initialize-update equations. The cost function that it tries to minimize is the MSE assuming stationarity of the data. Let $order$ be the filter order, μ be the step size, d be the desired signal..

1) Initialize the weight vector of length $order$ to zeros i.e.
$$w_{est}(0) = 0$$
2) Consider the first $order$ number of elements of the input to get the input vector u.
3) Find the instantaneous output using the current weight vector and the input considered in the above step i.e $y_{est} = w_{est}^{T}u$
4) Compute the instantaneous error i.e $e(n) = d(n) - y_{est}(n)$
5) Use the above feedback to update the weight vector by (4).
6) Slide the input used in step 2 by one sample and repeat the whole procedure until the end of the signal is reached.

$$w_{i+1} = w_i + (Mu * u * e(n))/(RegularizationParameter + inputPower) \tag{4}$$

where,

- Mu is a step size scaling factor
  - $RegularizationParameter$- 0.0001 (heuristically chosen)
  - u- Current sequence of input samples considered
  - $e(n)$- instantaneous error

  It is to be noted that the maximum step size possible in NLMS before the model diverges is

$$(1/MaxEigenvalue(Inputautocorrelationmatrix))$$

The evaluation metrics used in making decisions are the Normalized MSE and the Frobenius norm as shown in (5) and (6) respectively.

$$NMSE = \frac{(abs(Output - estimatedOutput)^2)}{mean(abs(Input)^2)} \tag{5}$$

For a matrix G,

$$FrobeniusNorm = \sum_i \sum_j |g_{ij}|^2 \tag{6}$$

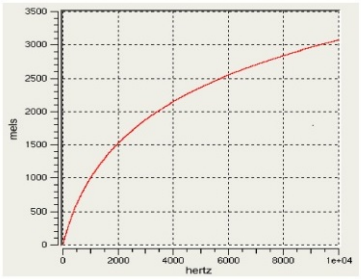
where, $g_{ij}$is the $(i,j)^{th}$element of the matrix G.

C. Mel Frequency Cepstral Coefficients - Vector Quantization (MFCC-VQ)

Speech is a quasi-stationary signal i.e. the statistics change throughout the duration of the signal. Hence, it is imperative that processing be done on individual frames of the signal where stationarity can be assumed. MFCCs are formulated based on the human acoustic system. Humans cannot discern between adjacent frequencies as the frequencies get higher. Mel scale uses this information and translates frequencies on the low range linearly while frequencies in the higher range get mapped in a logarithmic manner as seen in (7).

$$M(f) = 1127\, ln(1 + \frac{f}{700}) \tag{7}$$

where, M is the Mel scale and f is the frequency in Hz. The relation is graphically shown in Figure 2.



$m = 1127.01048 \log_e(1 + f/700)$ .....(1)

$f = 700(e^{m/1127.01048} - 1)$ .....(2)

Figure 2. Relation between Mel and Frequency scale

The steps involved in the MFCC-feature extraction are outlined below and shown in Figure 3:

- Frame the signals into small segments of around 10-20ms duration.
- Window each frame using a tapering window (Hamming here) to avoid discontinuities at the ends and to get a non negative power spectral estimate (Gibbs' phenomenon).
- Compute the power spectrum of each segment by taking the square of the FFT magnitude of the signal.

- $PowerSpectrum = \frac{1}{SignalLength}|FFT(signal)|^2$

- Compute the Mel spaced Filter bank. These are a set of 20 filter banks spaced equally on the Mel scale. Filter bank energies are calculated by multiplying each filter bank with the power spectral estimate and adding the coefficients. This is done for every frame.

- Take the log of each of the 20 energies and transform these log filter bank energies via Discrete Cosine transform. Therefore, for each frame you end up with 20 coefficients. These form your feature vector.

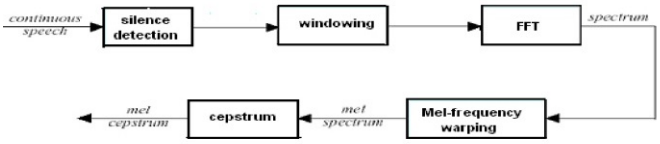


Figure 3. MFCC Feature extraction steps

Once, the feature vectors are computed, these are compressed by Vector Quantization technique via the LBG algorithm [7]. Vectors from a large space are mapped onto finite regions in the same space where each region is called a cluster and is represented by its centroid known as a codeword. The collection of all codewords form the code book which make up your compressed feature vector. The speech signals are therefore transformed into vectors in an acoustic space. Vector quantization techniques are used for matching patterns obtained from training vectors. LBG algorithm is the most common technique used for vector quantization which is a binary splitting task. A code book is formulated for each subject. In testing, feature vectors of the testing subject is compared to the codebook for each subject and distortion (distance measure) with respect to each subject is computed. The test subject will have minimum distortion with the true speaker. We can then obtain the identity of the true speaker.

III. Implementation

The datasets used and the algorithm steps are briefly discussed below:

A. Dataset

There were 2 datasets that were used. Dataset 1 (available online - [6]) comprised of 8 speakers in training and 8 speakers in testing. The training set included the speakers speaking the word 'Zero' while the testing included the same speakers speaking the same word but after a period of 6 months in order to account for voice variability. Dataset 2 (self recorded) included five speakers whose training and testing were recorded separately after a period of 7 days. The speakers spoke the same word 'Zero' for comparison purposes. The sampling frequency for dataset 1 is 12.5 kHz and 16kHz for dataset 2. Therefore, there are in total 13 speakers (9 female and 4 male).

B. Algorithm:

The steps involved in building the model for the 3 techniques are explained in brief below.

1) RLS/NLMS:

1) Extract the training signal and perform silence detection on it.
2) Run the RLS/NLMS algorithm on it to get the filter weight trajectories.
3) Extract only those weight trajectories which correspond to low NMSE values.
4) Perform steps 1-3 on the test signal.
5) Compute the Frobenius norm of the two weight matrices and obtain the similarity distance matrix. The lowest distance corresponds to the classified speaker.

2) MFCC-VQ:

1) Extract the training signal and perform silence detection on it.
2) Compute the MFCC feature vectors for each signal.
3) Form a code book for each subject in the training set using Vector Quantization via the LBG algorithm.
4) Perform steps 1-3 on the test signal.
5) Compute the Euclidean Distance or the distortion between the centroids in the code book to make the decision.

IV. Results

The algorithms were implemented in MATLAB. The environment used is Windows 8.1 Intel i7 processor with 8GB RAM. A forgetting factor of around 0.999 and an order of 7 was seen to generate the best results for RLS while for NLMS, a step size of 0.001, Regularization parameter of 0.0001 and filter order 7 was seen to work adequately well. Typically, for a non stationary signal the filter order is always low. The order reflects on the number of past samples being considered to arrive at the current estimate. Intuitively, one can see the extent of the past samples affecting the current estimate in a non stationary signal would be low since the statistics are rapidly changing in the signal. In MFCC-VQ, the signal was divided into 100 frames where each frame was windowed using a 256 length Hamming window. To compute the power spectrum, a 512 point FFT was employed to ensure oversampling and avoid aliasing. The power spectrum was passed through a total of 20 filter banks. The RLS/NLMS filter trajectories, NMSE curves and prediction estimates are analyzed with respect to a particular speech signal (speaker 3 here). This signal is shown in Figure 4.

A. Silence Detection

Before processing the signal, it is imperative that the silence regions be removed to make the model computationally efficient. These regions do not have any information that can be used to discern the speakers. Silence detection was implemented using techniques developed in [8]. The signal is divided into multiple frames and for each frame, 2 features are extracted i.e. the spectral centroid and the signal energy. Energy is obtained by squaring the amplitudes of the signal and summing them up - (8). Spectral centroid corresponds to the center of gravity of the spectrum. It essentially gives the 'brightness of the sound' - (9). Using these two features and thresholding them, one can detect the silence regions. The process is repeated for each frame. The working is shown in Figure 5 for speaker 3.

$$Energy = \frac{1}{N}\sum_{n=1}^{N}|x(n)|^2 \tag{8}$$

$$SpectralCentroid = \frac{\sum_{k=1}^{N}(k+1)X_i(k)}{\sum_{k=1}^{N}X_i(k)} \tag{9}$$

where,
$X_i(k)$ is the DFT of the i^{th}frame.

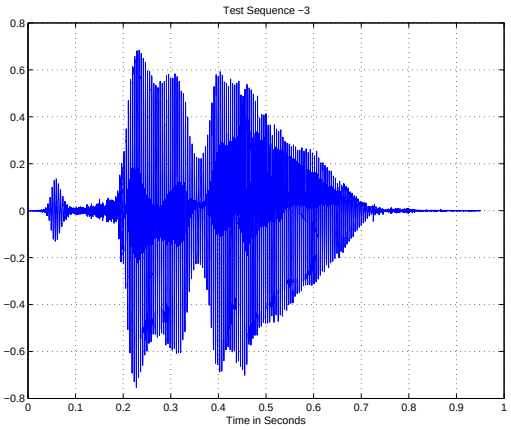


Figure 4. Speech signal for speaker 3

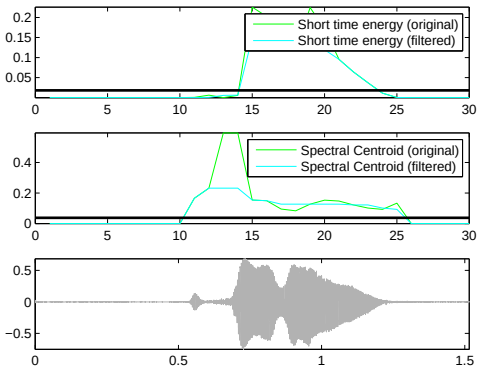


Figure 5. Silence Detection in the case of speaker 3

B. Discriminatory Features

One of the main motivations of using Adaptive learning algorithms was the fact that the discriminatory features used to distinguish speakers are the subject's audio spectral content. This can readily be seen in Figure 6. Spectrogram of 2 different speakers are shown here and it can be verified that the frequency content of the 2 speakers are different. This difference can be taken advantage of using RLS/NLMS since they are equipped to model frequency variations pretty well. The spectrogram plots were generated by dividing the signal into 150 windows using a hamming window and then compute the power spectrum using the corresponding next highest power radix 2 FFT to ensure oversampling. It is to be noted that MFCC technique also uses this difference to distinguish between subjects.

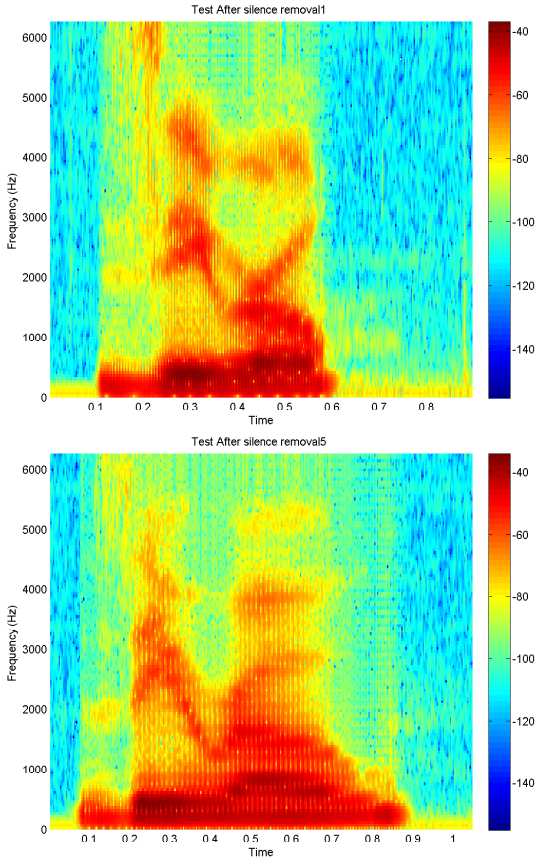


Figure 6. Spectrogram exhibiting spectral differences for 2 different speakers - speakers 1 and 5

C. Filter weight trajectories

Filter coefficients model the speech signal and is updated by the adaptive filter at every sample based on the feedback from the cost function. Typically for a stationary signal, the filter tracks are almost flat after the initial short convergence phase. This is not the case in speech and hence as can be seen in Figure 7 , the weight tracks undergo rapid transitions while trying to model the signal. In the figure, the weight tracks of NLMS and RLS models have been shown for a particular speaker (speaker 3 here). It is evident that RLS reaches its

steady state around the 0.5s mark while NLMS takes around 0.8s. This reinforces the previous statement made about the convergence time for NLMS being an order higher than that of RLS. These weight values at each instant comprises the feature vectors. Only a useful portion of this data needs to be extracted for discrimination between the speaker classes - specifically the portions corresponding to the maximum spectral differences as noted from the spectrogram.

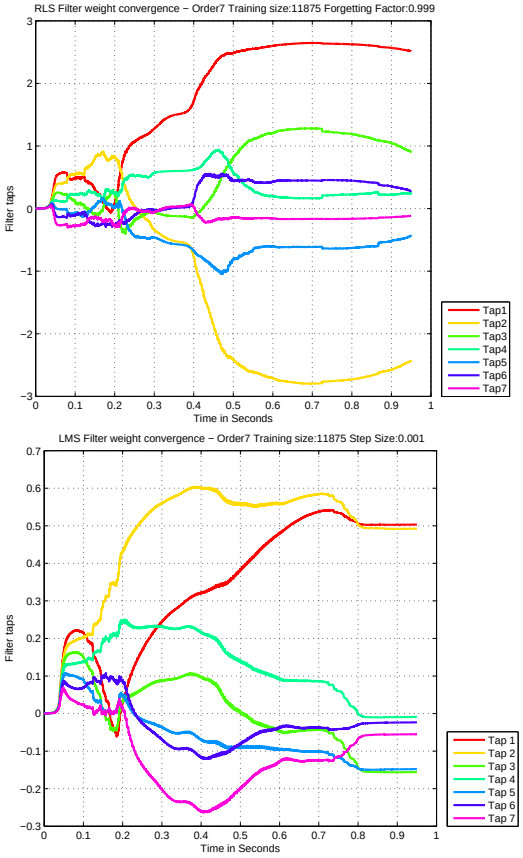


Figure 7. Filter weight trajectory of RLS (upper) and NLMS (lower) - Speaker 3

D. Predictor responses

Figure 8 shows the prediction estimates of both the RLS and NLMS for speaker 3. The superiority of RLS over NLMS can again be pointed out here. The prediction estimates by the RLS are pretty accurate while some of the NLMS estimates are out of place (specifically between 0.2-0.3s and 0.4-0.5s). This tells us that in this specific problem, RLS can better model the speech signals when compared to NLMS on account of the free parameter to control the assumed stationarity of the signal.

E. Learning Curves

The learning curves exhibit the learning ability of the model. NMSE values are computed using (5) and are smoothed using a 150 order moving average filter on account of the high frequency of the error signal and are shown in Figure 9. It can be seen initially when the model first receives the signal, it has no idea of its nature and therefore the filter coefficients cannot model the signal well and consequently leads to a high prediction error. However, as the model learns the signal traits, the error gradually decreases. The subsequent smaller peaks are due to the fact that the present portion of the signal has statistics different from what the filter was adapted to until that point and leads to spurious peaks. The error values are much lower in the RLS case than the NLMS (Y-axis) and also the NLMS has a harder time learning the signal and consequently leads to equal peaks in the learning curve.

F. Effect of step size and Forgetting Factor

Figure 10 and Figure 11 shows the effect of step size on NLMS filter tracks and the effect of forgetting factor on RLS weight tracks. It can be seen that a larger step size (0.01) in NLMS means that, the gradient descent updation happens with rapid fluctuations and the convergence values are not accurate since the estimates bounce around the optimum value violently. Similarly, in RLS, a lower forgetting factor (0.99) implies less past samples have been used to arrive at the autocorrelation and cross correlation estimates and consequently, the estimates are not as accurate as when higher samples are used. Therefore, the filter weights are not smooth and they undergo changes rapidly at every updation step. It is therefore necessary to find a good balance between accuracy

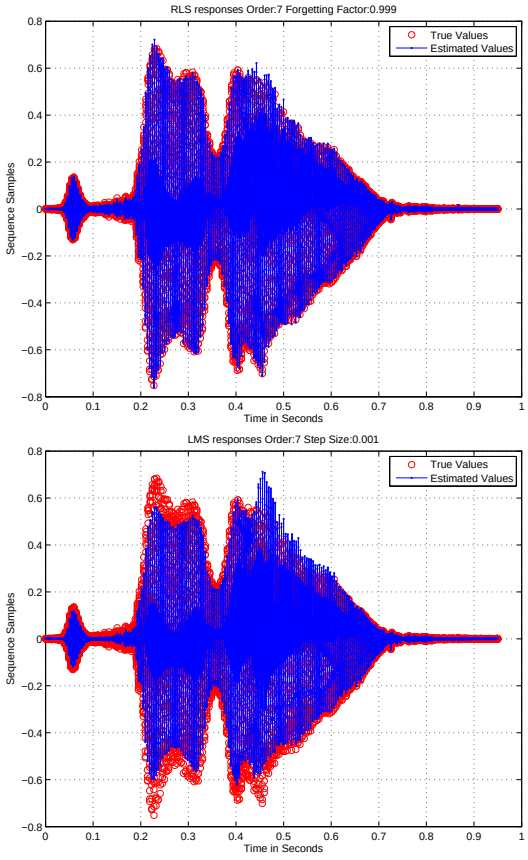


Figure 8. Prediction estimates of RLS (upper) and NLMS (lower) - Speaker 3

(given by the NMSE plots) and the smoothness of the filter trajectories.

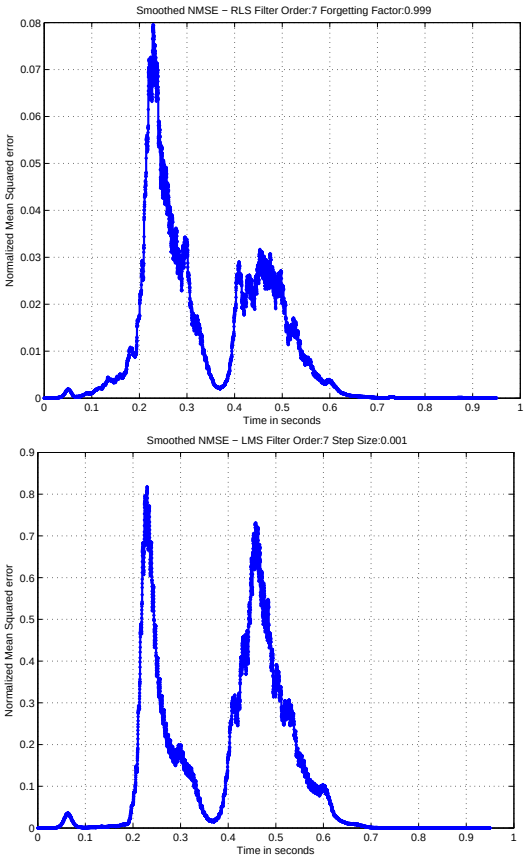


Figure 9. NMSE of RLS (upper) and NLMS (lower) - Speaker 3

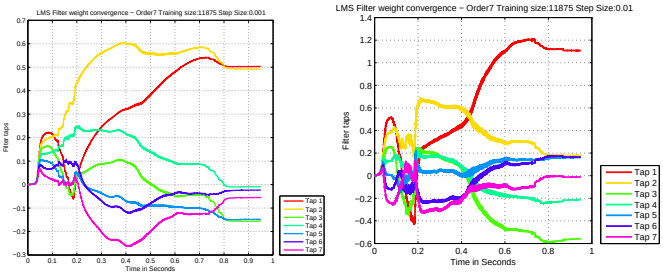


Figure 10. Effect of step size μ in NLMS

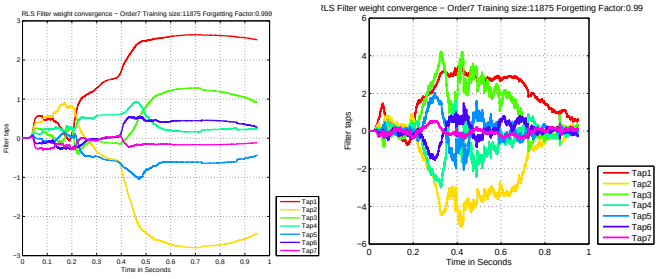


Figure 11. Effect of Forgetting Factor α in RLS

G. ROC Curves

The model is evaluated by plotting the ROC curves which is a variation of probability of detection with probability of false alarm as shown in Figure 12. The thresholds were generated by considering a linearly spaced values from the minimum distance value to the maximum distance value from the Distance similarity matrix. It is shown that RLS approximates the ideal ROC curve (inverted L) pretty closely when compared to the NLMS.

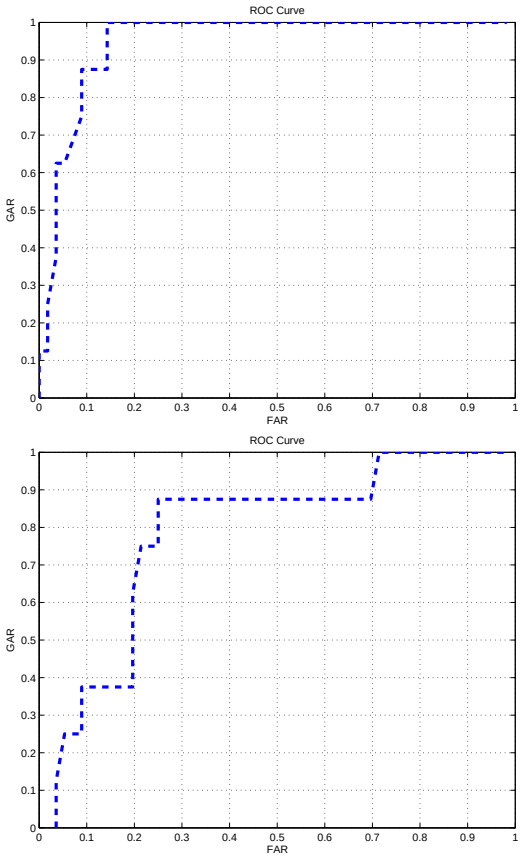


Figure 12. ROC Curves of RLS (upper) and NLMS (lower)

H. Error Curves

Error rate curves are obtained by plotting false reject rate with respect to false accept rate for varying thresholds. These are shown in Figure 13. Lower equal error rate (EER) i.e. the point of intersection of the 2 curves generally refers to the superior model. Here, RLS has an EER between 0.1-0.2 while NLMS has an EER between 0.2-0.3 again showing RLS as a better model.

I. Score distributions

The score distributions as shown in Figure 14 are more separated out in the RLS case and hence leads to a better decoding reducing the false alarm rates as expected.

J. Classification Accuracy

Now, to assess how the model performs when compared to the traditional MFCC-VQ, all 3 techniques were implemented for the same dataset and compared. It is seen that the proposed model works better than the traditional state of the art model. The results are shown in Table 1 justifying the technique proposed in this paper.

Model	Classification Accuracy (%)
RLS	100
NLMS	61.54
MFCC-VQ	76.92

Table I
CLASSIFICATION ACCURACY

K. Multiple models consideration

The limitation of this model is that it needs the speakers to speak the same words in both the training and testing. The filter coefficients are not adept at modeling the voice variations and the phoneme variations. RLS could be expanded to overcome this by probably considering multiple models for a particular speaker and selecting filter coefficients in multiple different regions. In the evaluation block, the models have to be compared in parallel and the one which results in a lower prediction error needs to be selected. This has to be done for every sample, but here, one also faces the problem of over fitting the model.

V. CONCLUSION AND FUTURE WORK

In conclusion, speaker recognition can be extremely vital in Biometric security and text dependence can be a reasonable assumption to make. These can be used in ATMs where the user could be asked to read out a particular word or a phrase. The proposed methods of RLS in this avenue has not been looked at before but here it is shown to generate good results, even better than the traditionally used techniques such as MFCC-VQ. The computation times are pretty low as well. This should be enough motivation to carry this work forward.

In the future, the model could be expanded to incorporate text Independence and needs to be evaluated on a wider database. In addition, RLS/NLMS could also be used as a denoiser in the pre processing block apart from merely being used in classification. This would need a noise model to be available though.

REFERENCES

[1] Simon Haykin. 1996. Adaptive Filter Theory (3rd Ed.). Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[2] .M. H. Hayes, Statistical Digital Signal Processing and Modeling, John Wiley & Sons, Inc

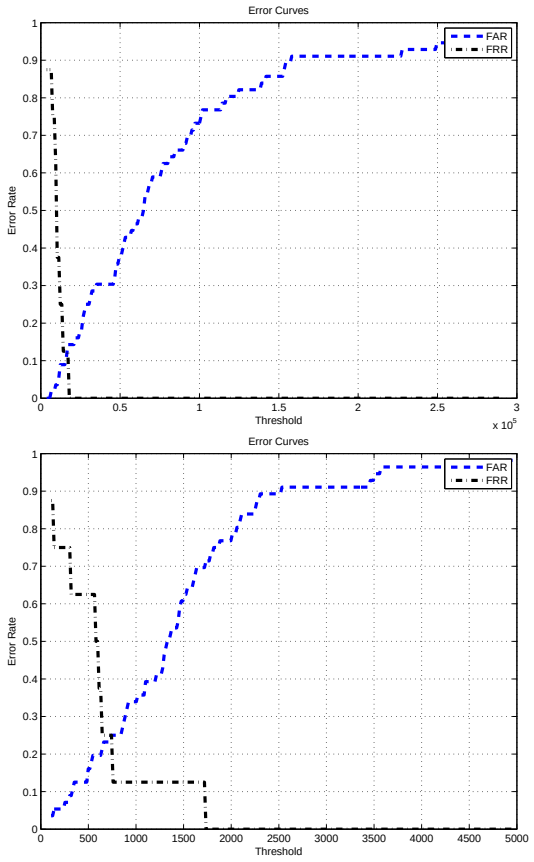


Figure 13. Error Curves of RLS (upper) and NLMS (lower)

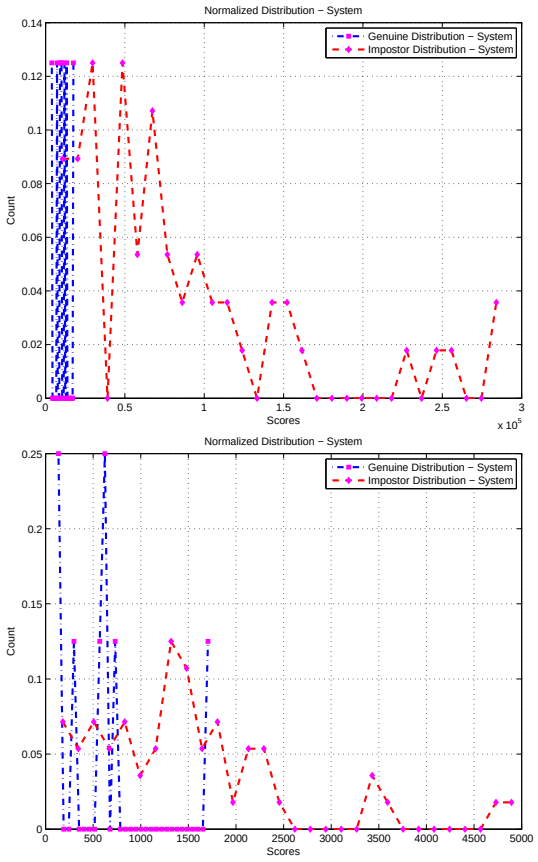


Figure 14. Score Distributions of RLS (upper) and NLMS (lower)

[3] Ilyas, M.Z, Univ. Malaysia Perlis (UniMAP), Arau, Malaysia – 'Improving hybrid speaker verification in noisy environments using least mean-square adaptive filters'

[4] Tadj, C, Ecole de Technol. Super., Montreal, Que., Canada, 'Towards robustness in speaker verification: enhancement and adaptation'

[5] Zulfiqar, A; Muhammad, A;ăăEnriquez, A.M.M, UoG, Gujrat, Pakistan, 'A Speaker Identification System Using MFCC Features with VQ Technique'

[6] http://minhdo.ece.illinois.edu/teaching/speaker_recognition/

[7] Y. Linde, A. Buzo & R. Gray, "An algorithm for vector quantizer design",IEEE Transactions on Communications, Vol. 28, pp.84-95, 1980.

[8] Theodoros Giannakopoulos, Computational Intelligence Laboratory (CIL) NCSR DEMOKRITOS, Greece, 'A method for silence removal and segmentation of speech signals, implemented in Matlab'