Simulation assignment 1
Performance of Adaptive Algorithms in Channel estimation

Balasubramaniam M C
Roll- EE18B155

November 24th 2020

Abstract:

This assignment is about showing how the ensemble average MSE(mean squared error) varies with iterations for different algorithms- LMS, eps-NLMS and RLS. Normally the Mean squared error is given by the expression:

$$J(i) = E\ |\mathbf{d} - \mathbf{u}w_{i-1}|^2 \tag{1}$$

where i >= 0

We approximate this to an ensemble average:

$$\hat{J}(i) = \frac{1}{L}\ (\sum_{j=1}^{L} |e^{(j)}(i)|^2) \tag{2}$$

where,
(iteration) i >= 0
j is the experiment number
L is the number of experiments

Using this expression we compare performance of different algorithms- RLS(Recursive Least squares), ϵ-NLMS (epsilon Normalized Least squares) and LMS (least mean squares) algorithms by performing channel estimation of the channel with the given channel impulse response:

$$C(z) = 1 + 0.5z^{-1} - z^{-2} + 2z^{-3} \tag{3}$$

Introduction

The input data u(i) of unit variance and the additive noise of variance 0.01 is generated as follows:

```
var_white = 0.01; %white noise variance
u = randn(N,1); %input normal distribution of unit variance 0 mean
y = filter(C,[1],u); % filter output
n = sqrt(var_white)*wgn(N,1,0); % generating white noise with var close to 0.01
d = y+n;  %o/p desired signal.
```

We pass input data 'u' through the channel impulse response and add the noise 'n' to the output to generate 'd' . A total of N =600 samples of data are used to train the adaptive filter of M=4 taps. We run all the algorithms for 600 iterations(no. of output samples d(i)) and average the MSE over L = 300 experiments. Hereafter, we discuss the procedure and the results and observations.

Procedure:

In each experiment, we generate 600 samples of separate input data and obtain the corresponding desired o/p data. We split 'u' into blocks of M=4 samples and slide this block one sample per iteration for 600 iterations (hence 600 samples) during which the weight vector 'wi' and error value e(i) is updated. This is done for all the algorithms. Code for one iteration is as follows:

```
% l =length(channel) =4
% splitting into blocks of M =4
ui = flip(u(c-(l-1):c));
%error update
ei = di-ui*wi;
mse(i) = abs(ei)^2;
```

As for the weight update, its different for each algorithm with their respective parameters and using 'ui' or 'ei' obtained above.

LMS:
step size (mew1) = 0.01
Weight update:

```
wi = wi + mew1*ui'*(di-ui*wi);
%di = d(i) ith value of desired o/p vector
```

ϵ-NLMS:
step size(mew2) = 0.2
ϵ(eps) =0.001
Weight update:

```
wi = wi + mew2*ui'*(ei)/(eps+ ui*ui');
%ei is the error value defined earlier
```

RLS:
lamda = 0.995
ϵ (eps) = 0.001
P_{-1} = identity matrix(l x l)/(eps)
where, l is channel length = M = 4
Update expressions:

```
%update of Pi matrix starting from given initial condition
pi = (lamda^-1)*(pi - ( (lamda^-1)*pi*(ui')*ui*pi)/(1 + (lamda^-1)*ui*pi*(ui')));
%updating wi vector using pi and ui,ei defined earlier
wi = wi+ pi*(ui')*(ei);
```

After all iterations in an experiment for all algorithms we separately accumulate the MSE vector to an ensemble average MSE vector and finally average it over L experiments (done for each algorithm) . The plot of this averaged MSE with iteration number is shown in figure 1.

Observations and Conclusions on Graphs:

- From the graph we can see that the average MSE for all three algorithms converge to a minimum MSE(dB) of -20dB (around 400 iterations) which is the noise variance.
- The LMS algorithm converges to the minimum MSE the last, i.e, slope (magnitude) of MSE vs. iterations is the lowest. This is obvious since, we know the convergence of the LMS algorithm is slower than that of ϵ-NLMS and RLS. This is because evolution of the MSE is not solely dependent on the step size which can be shown from the Least Perturbation Property. The LMS algorithm solves the following problem:

$$min\ _{w_i}\ ||w_i - w_{i-1}||^2 \tag{4}$$

subject to,

$$r(i) = (1 - \mu||u_i||^2)e(i) \tag{5}$$

where,e(i), r(i) is the a-priori and a-posteriori error respectively,

$$e(i) = d(i) - u_i w_{i-1} \tag{6}$$

$$r(i) = d(i) - u_i w_i \tag{7}$$

If the LMS algorithm solves this, then we can see that the evolution of the a-priori error e(i) which we plot evolves with a $(1 - \mu||u_i||^2))$ factor which has a dependence on 'ui', the observation or i/p block (length M) for that iteration. Hence, with a step size of just 0.01 and varying norm of 'ui', the MSE converges slowly.

- A similar argument can be made for the epsilon-NLMS algorithm, whose Least Perturbation Property is

$$min\ _{w_i}\ ||w_i - w_{i-1}||^2 \tag{8}$$

subject to,

$$r(i) = (1 - \frac{\mu||u_i||^2}{\epsilon + ||u_i||^2})e(i) \tag{9}$$

This shows that the evolution of e(i), a-priori error is almost only dependent on μ if ϵ is very small implying if we use a reasonably high step size, the error converges fast. Relatively, it's faster than the LMS algorithm. Hence, with a given step size of 0.2(higher than earlier) and an epsilon of 0.001 (which is very small) the MSE due to the epsilon-NLMS algorithm converges faster than that of the LMS method.

- The RLS algorithm uses Newton's method of gradient descent with regularization with a variable step size and epsilon. Additionally, it uses an approximation for R_u in the hessian different from that in the gradient which uses instantaneous approximation. So, it's obvious that due to the varying parameters and a better approximation, the MSE converges faster than that of ϵ -NLMS algorithm. Starting from the Newton's recursion,

$$w_i = w_{i-1} + \mu(i)[\epsilon(i)I + R_u]^{-1}[R_{du} - R_u w_{i-1}], \quad i >= 0 \tag{10}$$

In the epsilon-NLMS method, we had constant epsilon and mew. We also used the instantaneous approximation in the hessian of the expression. In the RLS algorithm, we use the following approximations:

$$\hat{R}_u = \frac{1}{i+1}\sum_{j=1}^{i} \lambda^{i-j} {u_j}^* u_j \tag{11}$$

$$\mu(i) = 1/(i+1) \tag{12}$$

$$\epsilon(i) = \lambda^{i+1}\epsilon/(i+1) \tag{13}$$

where, $0 << \lambda <= 1$
A special case of lamda = 1 would give a "growing memory", where we take the average of ${u_j}^* u_j$ from the first iteration.
The reducing regularization and step size gives better convergence. The different approximation for R_u in the hessian allows to give more weights

to recent samples forgetting older or past samples, which is better than that used in the epsilon-NLMS algorithm. Moreover, the RLS algorithm is the exact solution to the Exponentially weighted regularized least squares cost function which involves minimizing sum of squared error through all iterations directly. The previous algorithms exactly solved problems involving Least perturbation in the weight vector. Hence, it is no surprise that it offers a better convergence rate than eps- NLMS and LMS.

- Here, for the eps-NLMS and RLS algorithms, the channel coefficients are predicted within 100 iterations atmost. Intuitively, this is because the channel length M =4. If it were larger, more iterations would have been needed.

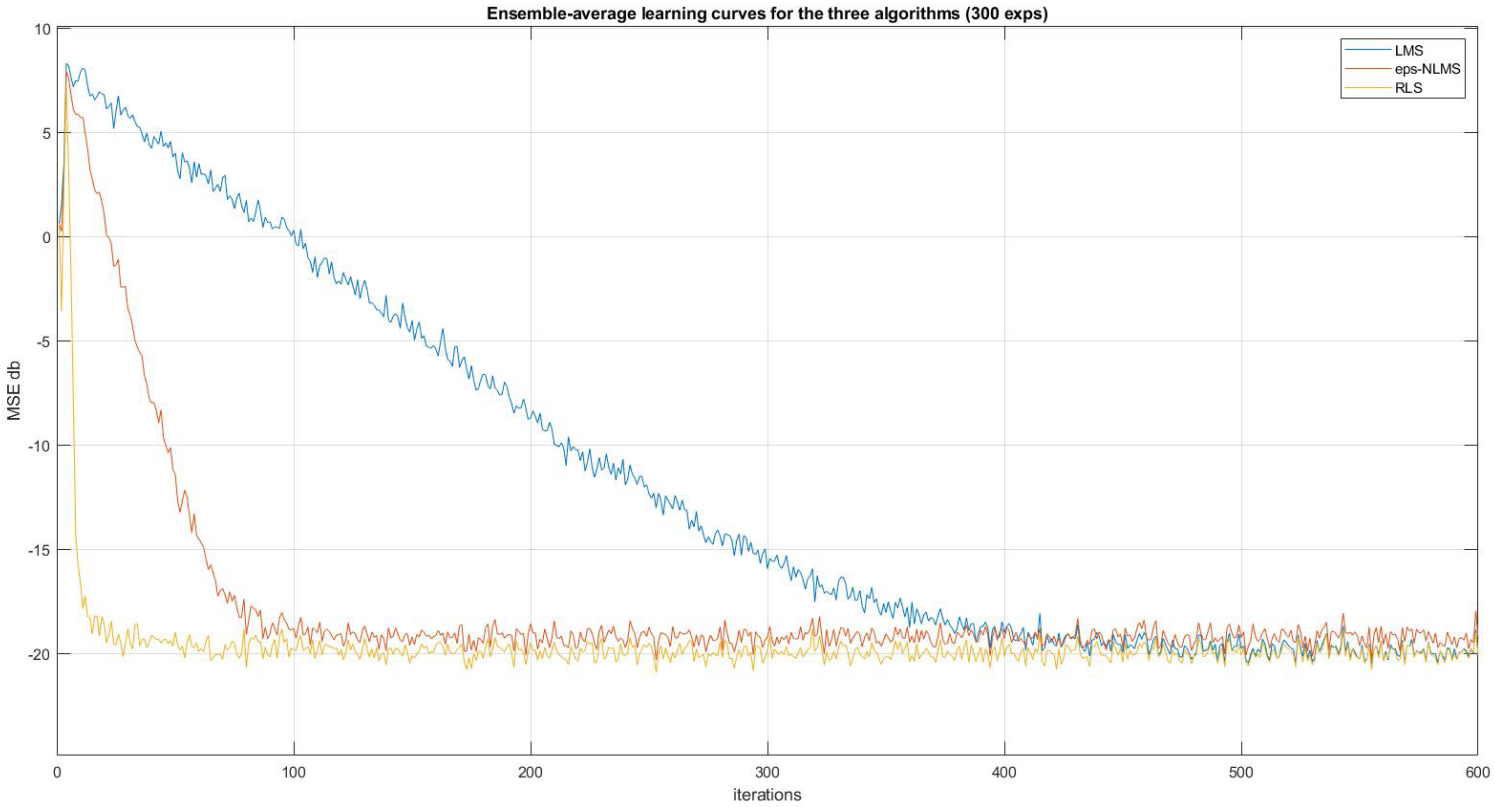

Figure 1: