The Cooper Union Department of Electrical Engineering
Prof. Fred L. Fontaine
ECE416 Adaptive Filters
Problem Set III: LMS and NLMS
June 2, 2019

Note: Problems refer to Haykin 5th ed, or 4th ed. (problem numbers are the same).

1. The purpose of this problem is to examine effects of various parameters on performance of LMS and NLMS algorithms. The white noise signals that drive the models here are *complex Gaussian.* For the signal $u[n]$ assume an AR model:

$$u[n] = v_1[n] - a_1 u[n-1] - a_2 u[n-2]$$

where v_1 is unit variance white noise and a_1, a_2 are chosen to achieve prescribed model poles. Reference the handout on rational PSD. We will take two cases: poles at 0.3 and 0.5, and poles at 0.3 and 0.95. For model orders, take $M = 3$, $M = 6$, and $M = 10$. For the underlying desired signal, assume a linear regression model:

$$d[n] = \beta^H u[n] + v_2[n]$$

where v_2 is unit variance white noise, and β is a vector of length 6, given by:

$$\beta_{0k} = \frac{1}{k^2}\ ,\ 1 \le k \le 6$$

   (a) For each model order, compute R_M, p_M, the Wiener filter w_M^0,$|w_M^0 - \beta|$ (to compute this for vectors of different length, extend by 0 as necessary). Compute the exact values from the specified coefficients in this problem, not values obtained by time-averaging the random signals. Also calculate the eigenvalue spread, and the bound on μ for a stable LMS algorithm (call it $\mu_{\max}$), and similarly the bound on $\tilde{\mu}$ for stable NLMS (call it $\tilde{\mu}_{\max}$). Comment on the relation between eigenvalue spread and the locations of the poles. Also, theory suggests what should happen when the order of the Wiener filter matches or exceeds that of the underlying model for d: check that.

   (b) You have two sets of possible poles, and three model orders. That is 6 cases total. For each case, run 4 different adaptive filters: LMS with step sizes $0.05\mu_{\max}$, $0.5\mu_{\max}$ and $0.8\mu_{\max}$, and NLMS with step size $0.2\tilde{\mu}_{\max}$. Obtain learning curves by averaging results over 100 runs. Ideally, all runs should be over the same number of iterations, and you should do enough iterations that all seem to converge to the steady-state condition (assuming they are stable). Generate graphs of the learning curve $J(n)$ and the mean-square deviation curve $\mathcal{D}(n)$ (remember, $\mathcal{D}(n)$ represents the error of the adaptive filter from the corresponding Wiener filter, not from the model vector β). In some cases, zooming in on a smaller number of iterations may make more sense.

   (c) Comments- stability, rate of convergence versus misadjustment, effect of eigenvalue spread, effect of model order, relation between $J(n)$ and $\mathcal{D}(n)$ (theory says they should always behave "similarly"- do they?). **Remark:** The analysis of LMS relied on "small step size theory" and therefore is less reliable for larger μ. So, for example, your value of $\mu_{\max}$ may not necessarily guarantee stability. As stated, the problem asks for 24 scenarios. I am less interested in filling up the computer video buffer with 24 graphs than with seeing some intelligent observations.

2. This problem involves adaptive equalization. Refer to problem 6.18 in Haykin. Here are some additional comments.

   - If you look at the channel models, neglecting the interference terms and noise, you have $u_n \approx x_{n-1}$. Notice that both a prior and a future symbol interfere with this; in other words, although the channel is presented as causal, the ISI has a sort-of non-causal behavior. Therefore, it is advantageous to look at both future and previous symbols. With $M = 21$ taps, we would like our "target" symbol $x_{n-\Delta}$ to appear in the center tap, having an equal number of "future" and "previous" values. Thus, take $\Delta = 10$ (taps 0 through 9 represent "future" symbols, that enter your receiver after the target point, and 11 through 20 represent "past" symbols that entered before your target point).
   - For your initial tap weight vector, set 1 at the center tap and 0 elsewhere. This represents just pulling out a target symbol without correction for ISI, and is thus a reasonable starting point.
   - Set up your code so you don't start adaptation until the transversal filter is completely full. Thus, load in the first 21 values of u, then start running the adaptive algorithm.
   - Haykin proposes a μ that is actually too low. You should use trial and error, adjusting μ by a factor of 10 or so, until you get reasonable results. Hopefully, you should get good convergence after about $N \approx 5M$ to $10M$ iterations. Just do this for the first channel model that is given. Then use this μ for each of the three cases (I want the same μ in each case), plot the learning curves.
   - Generate about 1000 samples of u (for each model) and use time-average estimates of R and p to find the Wiener filter, and $J_{\min}$, and compare with the mean of the tap weight vector you get at convergence in each case.
   - Now use NLMS. Again you may need to do some work to get reasonable $\tilde{\mu}$.

3. This problems involves adaptive beamforming. The last problem in Problem Set I involved setting up for array processing, and specifically a cross array (two linear arrays along x- and y-axes, respectively). The last problem in Problem Set II involved a linear array. You are going to repeat those situations. Just take the basic case for each, e.g., for the linear arrays assume $d = \lambda/2$ only (not the other variations). In each case, there are three sources plus background noise, with prescribed powers. MVDR refers to distortionless response in the prescribed source direction, and GSC adds nulls in the other two directions. Again, take only the basic case (e.g., only the originally prescribed SNR values).

(a) In each case, run an adaptive MVDR (meaning, an adaptive beamformer with the single distortionless constraint), and adaptive GSC (adaptive beamformer with three constraints, one for each source (one distortionless, the other two being nulls). Use the LMS algorithm, and adjust your step size μ so that near convergence occurs after $N = 5M$ iterations ($M=$ # antenna elements). If this seems too short, then try $N = 10M$. Feel free to do a trial and error. If you can, try to use the same μ in all cases. So, basically, you are going to compute 4 adaptive beamformers all together.

(b) For each beamformer, if w_o represents the optimal beamformer obtained with exact knowledge of the correlation matrix R, graph the mean-square deviation $\mathcal{D}(n)$ (averaged over 100 iterations). We are also interested in the array pattern; averaging the computed $\hat{w}$ vectors is not really a good idea. What I suggest you do is look at the array pattern for several instances of specific $\hat{w}(N)$, and pick a "typical" result. Choose appropriate graphical representations, showing the "optimal" array patterns with the adaptively computed array patterns.

(c) The MVDR beamformers (optimal and adaptively computed) will put strong attenuations in the two interferer directions. Compute the amount of attenuation and compare.

(d) Make qualitative observations. For example, as was discussed in the lectures, sometimes strong lobes are placed in certain directions (the distortionless response places a response of 1 at a desired direction, but does not ensure the response stays below 1 elsewhere); basically, if you notice anything that looks odd or "bad" about your beamformers, make that comment.