EE5027 Adaptive Signal Processing
Homework Assignment #3

Notice

- Due at 9:00pm, December 15, 2020 (Tuesday) $= T_d$ for the electronic copy of your solution.
- Please submit your solution to NTU COOL (`https://cool.ntu.edu.tw/courses/3062`)
- **All answers have to be fully justified**.
- All the figures should include labels for the horizontal and vertical axes, a title for a short description, and grid lines. Add legends and different line styles if there are multiple curves in one plot.
- No extensions, unless granted by the instructor one day before T_d.

Problems

1. (Block LMS algorithm, 15 points) The block LMS algorithm are characterized by the following equations

$$y(kL+\ell) = \widehat{\mathbf{w}}^H(k)\mathbf{x}(kL+\ell), \qquad \ell = 0,1,2,\ldots,L-1, \tag{1}$$

$$e(kL+\ell) = d(kL+\ell) - y(kL+\ell), \qquad \ell = 0,1,2,\ldots,L-1, \tag{2}$$

$$\widehat{\mathbf{w}}(k+1) = \widehat{\mathbf{w}}(k) + \mu_B\left(\frac{1}{L}\sum_{\ell=0}^{L-1}\mathbf{x}(kL+\ell)e^*(kL+\ell)\right). \tag{3}$$

Draw a block diagram for the block LMS algorithm with $L = 2$. You may use the L-fold decimator in Figure 1.

2. (Principle of orthogonality in LS filters, 15 points) In LS filters, the optimal weight vector is denoted by $\widehat{\mathbf{w}}_{\mathrm{LS}}$. With the optimal weight vector, the associated output signal and the error signal are represented by $y_{\mathrm{LS}}(n)$ and $e_{\mathrm{LS}}(n)$, respectively. Show that

$$\langle y_{\mathrm{LS}}(n), e_{\mathrm{LS}}(n)\rangle \triangleq \sum_{\ell=M}^{N} y_{\mathrm{LS}}(\ell)e^*_{\mathrm{LS}}(\ell) = 0. \tag{4}$$

In other words, the LS filter satisfies the principle of orthogonality.

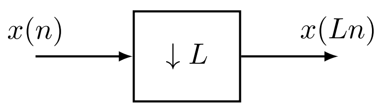

Figure 1: A schematic diagram for the L-fold decimator.

3. (RLS adaptive filters, 20 points)

   (a) (10 points) In the literature, the matrix inversion lemma can be expressed as follows

   $$(\mathbf{A}+\mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}\left(\mathbf{C}^{-1}+\mathbf{VA}^{-1}\mathbf{U}\right)^{-1}\mathbf{VA}^{-1}, \tag{5}$$

   where all the matrices are conformable and all the matrix inversions exist. Prove the matrix inversion lemma (5).
   Remark: The form in (5) is more general than the expression introduced in the lecture.

   (b) (10 points) Show that the gain vector $\mathbf{k}(n)$ in RLS filters satisfies

   $$\mathbf{k}(n) = \mathbf{P}(n)\mathbf{x}(n). \tag{6}$$

4. (MVDR spectrum, 20 points) We consider the array model $\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t) + \mathbf{n}(t)$ with the N-element ULA of inter-element spacing $\lambda/2$. There is only *one source* with DOA θ_1 and source power p_1. We assume that

   $$\mathbb{E}[\mathbf{s}(t)] = \mathbf{0}, \qquad \mathbb{E}[\mathbf{n}(t)] = \mathbf{0}, \tag{7}$$

   $$\mathbb{E}[\mathbf{s}(t_1)\mathbf{s}^H(t_2)] = p_1\delta_{t_1,t_2}, \quad \mathbb{E}[\mathbf{n}(t_1)\mathbf{n}^H(t_2)] = (p_n\mathbf{I})\delta_{t_1,t_2}, \quad \mathbb{E}[\mathbf{s}(t_1)\mathbf{n}^H(t_2)] = \mathbf{0}. \tag{8}$$

   (a) (5 points) Find the covariance matrix $\mathbf{R}_\mathbf{x}$ of $\mathbf{x}(t)$.

   (b) (15 points) Let $P_{\mathrm{MVDR}}(\theta)$ be the MVDR spectrum associated with $\mathbf{R}_\mathbf{x}$ in Problem 4a. The look direction in the MVDR spectrum is θ_1. Show that

   $$P_{\mathrm{MVDR}}(\theta_1) \geq P_{\mathrm{MVDR}}(\theta), \tag{9}$$

   for all $-\pi/2 < \theta < \pi/2$.

5. (Performance study of adaptive filters, 30 points) In this problem, we will implement the LMS adaptive filter, the NLMS adaptive filter, and the RLS adaptive filter. Next we will study the performance of these adaptive filters, through *Monte-Carlo simulations*.

   Monte-Carlo simulations consist of three steps. First, the input data is randomly generated according to a pre-defined probability distribution. Second, the algorithm is executed to obtain the attribute to be studied. The first two steps are called *one Monte-Carlo run*, or *one Monte-Carlo trial*. Third, the previous two steps are repeated multiple times and the attributes are averaged over all these Monte-Carlo runs. This simulation technique is very useful in studying the statistical performance of algorithms.

   In the lecture, the learning curve is defined as $J(n) \triangleq \mathbb{E}[|e(n)|^2]$, which involves the expectation over the probability space. Alternatively, in Monte-Carlo simulations, we generate the samples for input signal and the desired signal. In order to do so, we assume the random process $v(n)$ to be a zero-mean, circularly-symmetric complex Gaussian, white, wide-sense stationary random process with unit variance ($\sigma_v^2 = 1$). In the r-th Monte-Carlo run, we first consider the samples for the random process $v(n)$, denoted by $v^{(r)}(n)$. In the mat file `ASP_HW3_Problem_5.mat`, the numerical values for $v^{(r)}(n)$ are provided with the following layout:

$$\texttt{matV} = \begin{bmatrix} v^{(1)}(0) & v^{(1)}(1) & v^{(1)}(2) & \dots & v^{(1)}(L-1) \\ v^{(2)}(0) & v^{(2)}(1) & v^{(2)}(2) & \dots & v^{(2)}(L-1) \\ v^{(3)}(0) & v^{(3)}(1) & v^{(3)}(2) & \dots & v^{(3)}(L-1) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v^{(1000)}(0) & v^{(1000)}(1) & v^{(1000)}(2) & \dots & v^{(1000)}(L-1) \end{bmatrix}. \tag{10}$$

We assume $v^{(r)}(n) = 0$ if $n < 0$ or $n \geq L$. Next we generate the samples for the input signal and the desired signal as follows:

$$x^{(r)} = h(n) * v^{(r)}(n), \qquad d^{(r)}(n) = v^{(r)}(n), \tag{11}$$

where $h(n) * v^{(r)}(n)$ denotes the convolution of $h(n)$ and $v^{(r)}(n)$. The z transform of the causal impulse response $h(n)$ is

$$H(z) = \sum_{n=0}^{\infty} h(n) z^{-n} = \frac{1 + \frac{1}{10} z^{-1}}{\left(1 - \frac{1}{2} z^{-1}\right)\left(1 + \frac{1}{3} z^{-1}\right)}, \tag{12}$$

with the region of convergence $|z| > \frac{1}{2}$. Based on $x^{(r)}(n)$ and $d^{(r)}(n)$, the adaptive filters produces the output samples $y^{(r)}(n)$, the weight vectors $\widehat{\mathbf{w}}^{(r)}(n)$, and the error samples $\mathfrak{e}^{(r)}(n)$. Finally the empirical learning curve is obtained by

$$\widehat{J}(n) \triangleq \frac{1}{R} \sum_{r=1}^{R} \left| \mathfrak{e}^{(r)}(n) \right|^2. \tag{13}$$

The empirical learning curve in (13) can be viewed as a surrogate of the true learning curve $J(n)$.

Guidelines for implementation: For debugging purposes, the adaptive filters are implemented in separate MATLAB functions. Please write `ASP_LMS.m` for the LMS adaptive filter, `ASP_NLMS.m` for the NLMS adaptive filter, and `ASP_RLS.m` for the RLS adaptive filter.

(a) Supposing $M = 5$ and $\widehat{\mathbf{w}}(0) = \mathbf{0}$, we consider the following cases for comparison:

- The LMS adaptive filter: $\mu = 0.1$.
- The LMS adaptive filter: $\mu = 0.2$.
- The NLMS adaptive filter: $\widetilde{\mu} = 0.2$.
- The NLMS adaptive filter: $\widetilde{\mu} = 0.8$.
- The RLS adaptive filter: $\lambda = 0.75$, and $\delta = 0.01$.
- The RLS adaptive filter: $\lambda = 0.75$, and $\delta = 0.1$.

- The RLS adaptive filter: $\lambda = 0.95$, and $\delta = 0.01$.

 Plot the empirical learning curves with $R = 10$. Examples of empirical learning curves can be found in [Haykin, Figure 6.15, page 308].

 (b) **Repeat Problem 5a for $R = 1000$.**

6. (Bonus problem, 10 points) **Explain the results in Problem 5b as much as possible**. You may look for the following terminologies: the rate of convergence, the time constant, excess MSE, and misadjustment, in [1].
 [1] S. Haykin, Adaptive Filter Theory, Fifth Edition, Prentice Hall, 2013.

MATLAB Submission Checklist

1. `ASP_HW3_Problem_5.m`: This is the main program. We will obtain all the results in Problem 5 after execution.
2. `ASP_LMS.m`: The function for the LMS adaptive filter.
3. `ASP_NLMS.m`: The function for the NLMS adaptive filter.
4. `ASP_RLS.m`: The function for the RLS adaptive filter.
5. `ASP_HW3_Problem_5_R_10.fig`: The MATLAB fig file for the results in Problem 5a.
6. `ASP_HW3_Problem_5_R_1000.fig`: The MATLAB fig file for the results in Problem 5b.

Last updated November 24, 2020.