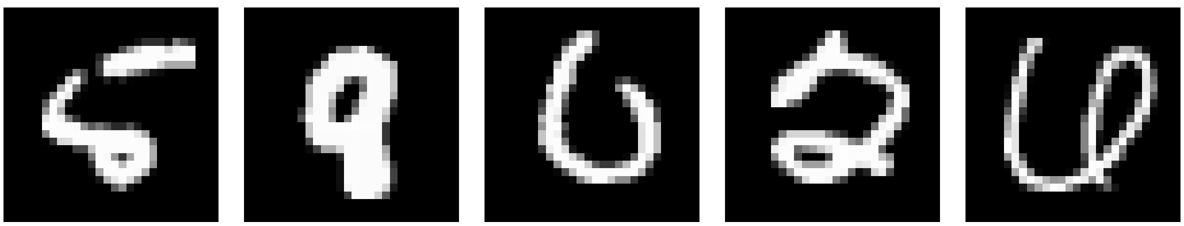
EE599: Homework #2

Yifan Wang
wang608@usc.edu #3038184983

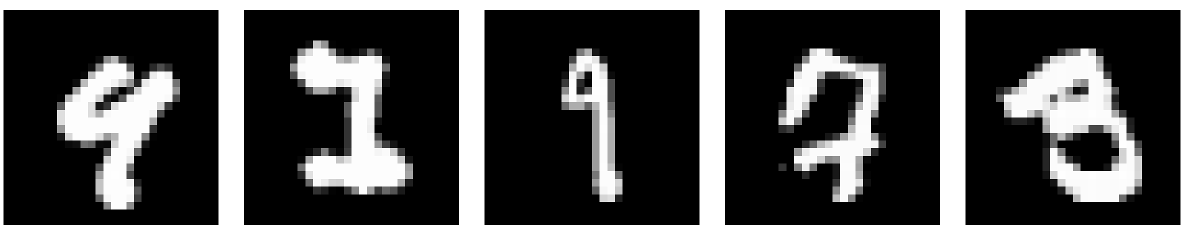
February 2, 2019

Problem 1

Some of wrong classified numbers are shown in $Figure1$. Among these, some are really tough to recognize even for human while majorities are simple.



(a) $y_{pred} = 6, y = 5$ (b) $y_{pred} = 9, y = 8$ (c) $y_{pred} = 6, y = 0$ (d) $y_{pred} = 0, y = 2$ (e) $y_{pred} = 0, y = 6$



(f) $y_{pred} = 9, y = 4$ (g) $y_{pred} = 2, y = 2$ (h) $y_{pred} = 1, y = 9$ (i) $y_{pred} = 9, y = 7$ (j) $y_{pred} = 8, y = 3$

Figure 1: Wrong classified example

Problem 2

From example of generating the sequences, main difference from human and computer maybe as follows:

When we are asked to generate a long random sequence, at first, we are concentrated and trying to make sequence as random as we thought, so that many short similar patterns like '010101' etc are avoided from occurring.

Because of the long length of the sequence, for example 500 in our cases, we would get boring and tired so that may not concentrate well. When we are unconscious, it is likely that we would follow some rhythm and generate sequences with similar patterns.

After a while, we would come back to real world and become concentrate again, and repeat concentrate, un-concentrate .....until end. When sequence is long enough, sharp changes can be observed in sequence between random subsequences and subsequences with some patterns....

What's more, for most people, even realizing that the sequence generated might not random as they thought, only if it is long enough, they would never want to review and modify it since it is super boring.

Even if some one tried to correct some patterns that repeatedly occur in their sequence, they would likely to destroy all the repeated patten that may occur in true random sequence, for example, they would never allow pattern like 11111 occur while it would happened when generated by a machine.

Problem 3

(3-1)

(3-1-a)

$$E[v_n(u)z_n(u)] = E[v_n(u)(y_n(u) + q_n(u))] = E[v_n(u)y_n(u)] + E[v_n(u)q_n(u)] \tag{1}$$

due to the inependence of $v_n(u)$ and $q_n(u)$,

$$E[v_n(u)q_n(u)] = E[v_n(u)]E[q_n(u)]$$

Noise follows a zero mean Gaussian distribution, so that $E[q_n(u)] = 0$, and equation (1) becomes:

$$E[v_n(u)z_n(u)] = E[v_n(u)y_n(u)] \tag{2}$$

Proof done.

(3-1-b)

$$\begin{aligned} R_{xz}[m] = E[x_n(u)z_{n+m}(u)] &= E[x_n(u)(y_{n+m}(u) + q_{n+m}(u))] \\ &= E[x_n(u)y_{n+m}(u)] = R_{xy}[m] \\ &= E[x_n(u)(h_0x_{n+m}(u) + h_1x_{n+m-1}(u) + h_2x_{n+m-2}(u))] \\ &= h_m, m = (0,1,2) \end{aligned} \tag{3}$$

(3-1-c)

Due to independence $E[x_n(u)x_{n-m}(u)] = 0$ when $m \neq 0$:

$$\begin{aligned} R_{v_n} = E[v_n(u) * v_n(u)^T] = E \begin{bmatrix} x_n(u)x_n(u) & x_n(u)x_{n-1}(u) & x_n(u)x_{n-2}(u) \\ x_{n-1}(u)x_n(u) & x_{n-1}(u)x_{n-1}(u) & x_{n-1}(u)x_{n-2}(u) \\ x_{n-2}(u)x_n(u) & x_{n-2}(u)x_{n-1}(u) & x_{n-2}(u)x_{n-2}(u) \end{bmatrix} \\ = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{4}$$

(3-1-d)

From $Equation(3)$:

$$\begin{aligned} r_n = E[v_n(u)z_n(u)] = E[v_n(u)(y_n(u) + q_n(u)] = E[v_n(u)y_n(u)] \\ = E \begin{bmatrix} x_n(u)y_n(u) \\ x_{n-1}(u)y_n(u) \\ x_{n-2}(u)y_n(u) \end{bmatrix} = E \begin{bmatrix} x_n(u)h_0x_n(u) + x_n(u)h_1x_{n-1}(u) + x_n(u)h_2x_{n-2}(u) \\ x_{n-1}(u)h_0x_n(u) + x_{n-1}(u)h_1x_{n-1}(u) + x_{n-1}(u)h_2x_{n-2}(u) \\ x_{n-2}(u)h_0x_n(u) + x_{n-2}(u)h_1x_{n-1}(u) + x_{n-2}(u)h_2x_{n-2}(u) \end{bmatrix} \\ = \begin{bmatrix} h_0 \\ h_1 \\ h_2 \end{bmatrix} \end{aligned} \tag{5}$$

(3-1-e)

The taps for Wiener filter for $3dB$ and $10dB$ is same:

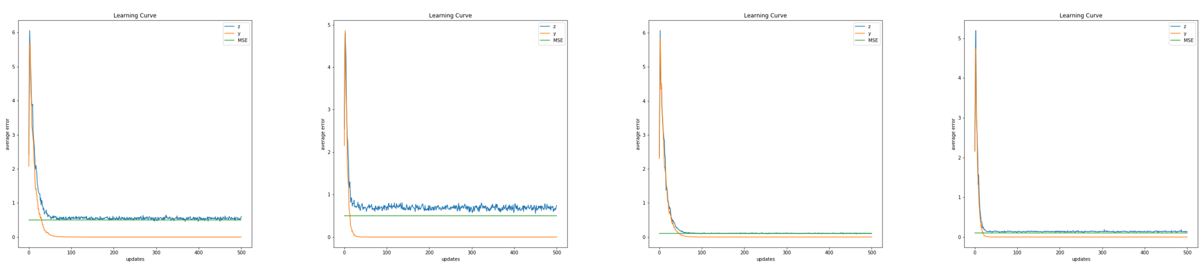
$$\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}^{-1} * \begin{bmatrix}h_0\\h_1\\h_2\end{bmatrix} = \begin{bmatrix}1\\0.5\\0.25\end{bmatrix} \tag{6}$$

(3-1-f)

Residual mean squared error when estimating $z_n(u)$ is: σ_q^2 ($3dB$: 0.5011, $10dB$: 0.1). While estimating $y_n(u)$ is: 0, which means random noise can not be predicted or estimated in Wiener Filter.

(3-2)

(3-2-b)



(a) $3dB$, $\alpha = 0.05$ (b) $3dB$, $\alpha = 0.15$ (c) $10dB$, $\alpha = 0.05$ (d) $10dB$, $\alpha = 0.15$

Figure 2: LMMSE for estimating $z_n(u)$ and $y_n(u)$

(3-2-c)

When estimating $z_n(u)$, MSE for these learning curves can get slightly smaller error than LMMSE does ($Figure2$). What's more a small learning rate can lead to a smaller LMMSE which is closer to MSE. When estimating $y_n(u)$, both methods can achieve the MSE at 0.

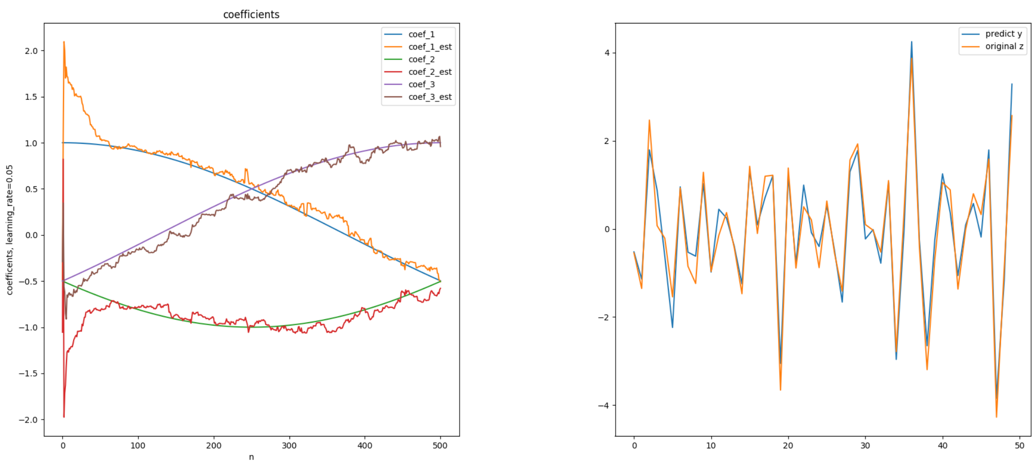
(3-2-d)

The largest learning rate we can use is about 0.4 when SNR is 10dB and 0.3 when SNR is 3dB. However, if choosing such larger learning rate, even though LMMSE would not divergent, LMMSE at these learning rate would be larger than that in a small learning rate.

Besides, when convergent, large learning rate system can be easily affected by some noise, one of its result is LMMSE would fluctuate more dramatically than a small learning rate system. This fact can be explained that a large learning rate can provide more change update in each step, which means they are more sensitive to changes in system, even if changes come from noise. Notice that a large learning rate can lead to the LMMSE convergent more quickly as well.

(3-3)

Results are shown in $Figure4$. When coefficients changes slow, LMS works pretty well, which matches my previous experience on using LMS in active noise control system, LMS can follow (cancel) low frequency noise pretty well.



(a) Coefficient change (Problem3.3)

(b) predicted output (Problem3.3)

Figure 3

(3-4)

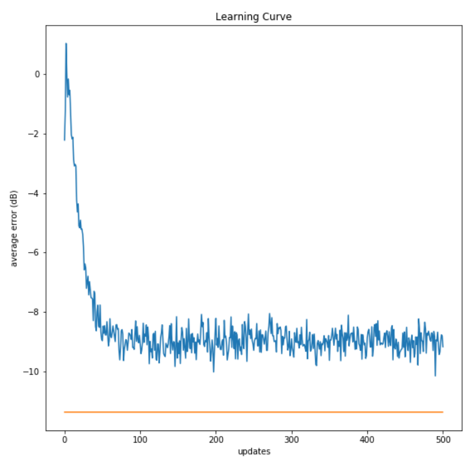


Figure 5: Learning Cure (Problem3.4) $\alpha = 0.05$

The LLSE from entire data set is $-11.35dB$, which is lower than LMS learning curve after convergence.

$$R_{v_n} = \begin{bmatrix} 0.8228 & 0.00017 & -0.0284 \\ 0.00017 & 0.8171 & -0.00065 \\ -0.0284 & -0.00065 & 0.8170 \end{bmatrix}$$

$$r_n = \begin{bmatrix} 0.3350 \\ 0.3530 \\ 0.2965 \end{bmatrix}$$