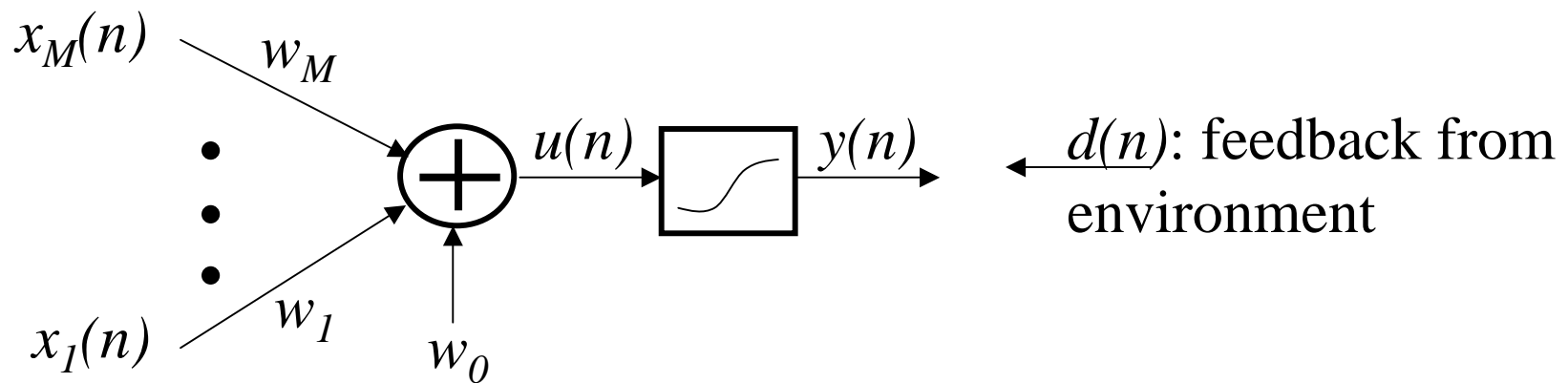


Lecture 7.
Learning (IV):
Error correcting Learning and
LMS Algorithm

Outline

- Error Correcting Learning
 - LMS algorithm
 - Nonlinear activation
- Batch mode Least Square Learning

Supervised Learning with a Single Neuron



Input: $x(n) = [1 \ x_1(n) \ \dots \ x_M(n)]^T$

Parameter: $w(n) = [w_0(n) \ w_1(n) \ \dots \ w_M(n)]^T$

Output: $y(n) = f[u(n)] = f[w^T(n)x^T(n)]$

Feedback from environment: $d(n)$, desired output.

Error-Correction Learning

- Error @ time n = desired value – actual value

$$e(n) = d(n) - y(n)$$

- **Goal**: modify $w(n)$ to minimized square error

$$E(n) = e^2(n)$$

- This leads to a steepest descent learning formulation:

$$w(n+1) = w(n) - \eta' \nabla_w E(n)$$

$$\begin{aligned} \text{where } \nabla_w E(n) &= [\partial E(n)/\partial w_0(n) \dots \partial E(n)/\partial w_M(n)]^T \\ &= 2 e(n) [\partial y(n)/\partial w_0(n) \dots \partial y(n)/\partial w_M(n)]^T \end{aligned}$$

is the gradient of $E(n)$ w.r.t. $w(n)$.

η' is a learning rate constant.

Case 1. *Linear Activation:* LMS Learning

If $f(u) = u$, then $y(n) = u(n) = w^T(n)x(n)$

$$\begin{aligned} \text{Hence } \nabla_w E(n) &= 2 e(n) [\partial y(n)/\partial w_0(n) \dots \partial y(n)/\partial w_M(n)]^T \\ &= 2 e(n) [1 \quad x_1(n) \dots x_M(n)]^T = 2 e(n) x(n) \end{aligned}$$

Note $e(n)$ is a scalar, and $x(n)$ is a $M+1$ by 1 vector. Let $\eta = 2\eta'$, we have the least mean square (LMS) learning formula as a special case of error-correcting learning:

$$w(n+1) = w(n) + \eta e(n) \cdot x(n)$$

Observation The amount of corrections made to $w(n)$, $w(n+1) - w(n)$ is proportional to the magnitude of the error $e(n)$ and along the direction of the input vector $x(n)$.

Example

Let $y(n) = w_0(n) \cdot 1 + w_1(n) x_1(n) + w_2(n) x_2(n)$. Assume the inputs are:

n	1	2	3	4
$x_1(n)$	0.5	-0.4	1.1	0.7
$x_2(n)$	0.8	0.4	-0.3	1.2
$d(n)$	1	0	0	1

Assume $w_0(1) = w_1(1) = w_2(1) = 0$, and $\eta = 0.01$.

$$e(1) = d(1) - y(1) = 1 - [0 \cdot 1 + 0 \cdot 0.5 + 0 \cdot 0.8] = 1$$

$$w_0(2) = w_0(1) + \eta e(1) \cdot 1 = 0 + 0.01 \cdot 1 \cdot 1 = 0.01$$

$$w_1(2) = w_1(1) + \eta e(1) x_1(1) = 0 + 0.01 \cdot 1 \cdot 0.5 = 0.005$$

$$w_2(2) = w_2(1) + \eta e(1) x_2(1) = 0 + 0.01 \cdot 1 \cdot 0.8 = 0.008$$

Results

n	1	2	3	4
$w_0(n)$	0.01	0.0099	0.0098	0.0195
$w_1(n)$	0.005	0.0050	0.0049	0.0117
$w_2(n)$	0.008	0.0080	0.0080	0.0197

Matlab source file: [Learner.m](#)

Case 2. Non-linear Activation

In general,

$$\begin{aligned}\nabla_w E(n) &= -2e(n) \cdot \frac{df[u(n)]}{du(n)} \nabla_w u(n) = -2e(n) \cdot \frac{df[u(n)]}{du(n)} x(n) \\ &= -2e(n) \cdot f'[u(n)] \cdot x(n)\end{aligned}$$

Observation:

The additional term is $f'[u(n)]$. When this term becomes small, learning will NOT take place.

Otherwise, the update formula is similar to LMS.

LMS and Least Square Estimate

Assume that the parameters w remain unchanged for $n = 1$ to N ($> M$). Then, $e^2(n) = d^2(n) - 2d(n)w^T x(n) + w^T x(n)x^T(n)w$. Define an expected error (Mean square error)

$$E = \sum_{n=1}^N e^2(n) = \sum_{n=1}^N d^2(n) - 2w^T \sum_{n=1}^N x(n)d(n) + w^T \left(\sum_{n=1}^N x(n)x^T(n) \right) w$$

Denote $R = \sum_{n=1}^N x(n)x^T(n)$, and $\rho = \sum_{n=1}^N x(n)d(n)$

Then $E = \sum_{n=1}^N d^2(n) - 2w^T \rho + w^T R w$

where R : correlation matrix, ρ : cross correlation vector.

Least Square Solution

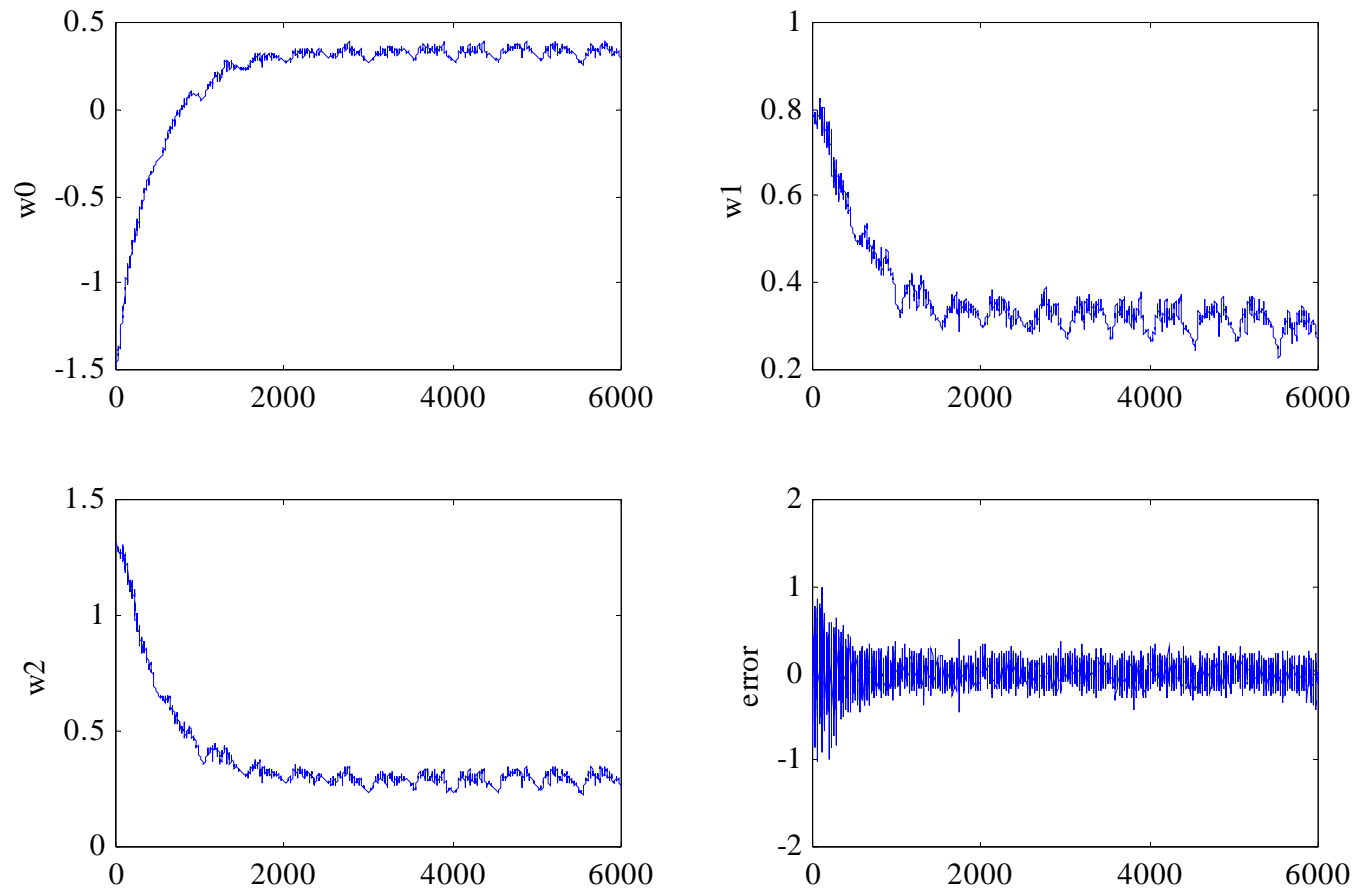
- Solve $\nabla_w E = 0$, for W , we have

$$w_{LS} = R^{-1}\rho$$

- When $\{x(n)\}$ is a *wide-sense stationary* random process, the LMS solution $w(n)$ converges in probability to the least square solution w_{LS} .

[LMSdemo.m](#)

LMS Demonstration



LMS output comparison

