LINEAR FIR ADAPTIVE FILTERING (I I)
Stochastic Gradient-based Algorithms:
(Least-Mean-Square [LMS])

Dr. Yogananda Isukapalli

• Why the LMS Adaptive Filter ?

• Steepest descent algorithm has been used to obtain an iterative solution to fixed normal equations.

• We need to design a filter which is responsive to changes in the input signal environment, that is we need an interactive structure that is dependent on the input data.

• It is possible to construct a steepest-descent algorithm by replacing the fixed auto and cross-correlation matrices by their time dependent equivalents.

$$\underline{w}(n+1) = w(n) + \mu[\underline{p}(n) - \underset{\sim}{R}(n)w(n)]$$

We note two main problems with this approach:

A. It is very computationally intensive and expensive since we have to compute $\underset{\sim}{R}$ and $\underline{p}$ at each point (n).

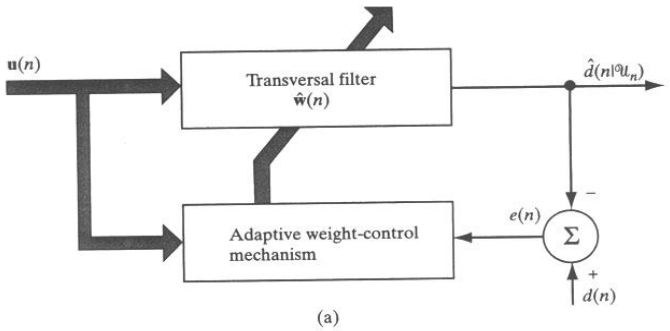
B. It may not be possible to compute $\underset{\sim}{R}(n)$ and $\underline{p}(n)$ if only a single realization of the process is available.

LMS algorithm is based on estimating the gradient of the mean-squared error by the gradient of the instantaneous value of the squared error.

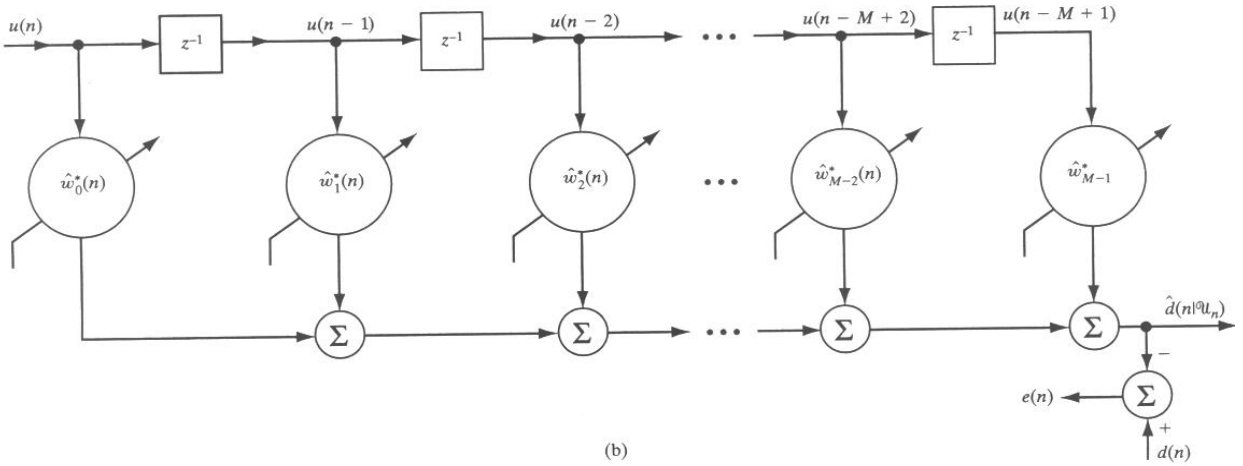
$$\nabla(J(n)) = \frac{\partial J}{\partial \underline{w}(n)} = \frac{\partial E\{e^2(n)\}}{\partial \underline{w}(n)}$$

Is replaced by:

$$\hat{\nabla}(J(n)) = \frac{\partial \hat{J}}{\partial \underline{w}(n)} = \frac{\partial e^2(n)}{\partial \underline{w}(n)}$$

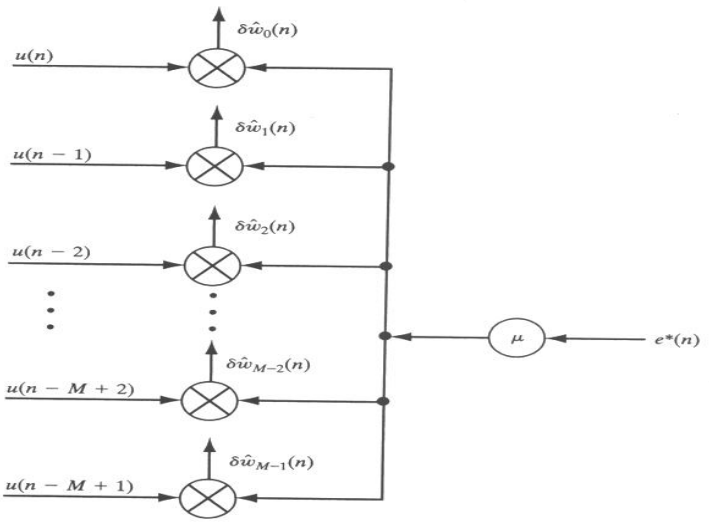


(a)



(b)

FIGURE 5.1 *(continued on the next page)*



(c)

FIGURE 5.1 (a) Block diagram of adaptive transversal filter. (b) Detailed structure of the transversal filter component. (c) Detailed structure of the adaptive weight-control mechanism.

Notes on Previous Figure:

Fig 5.1a Adaptive transversal filter consists of a transversal filter around which the LMS algorithm is built, note also a mechanism for performing the adaptive control process on the tap weights of the transversal filter.

Fig 5.1b The tap input vector is:

u(n) : M-1 is the number of delay elements.

w(n) : Mx1 tap weigth vectors

Fig 5.1c Note the correction $\delta\hat{w}_k(n)$ applied to the tap weight $\hat{w}_k(n)$ at time n+1. μ is called the adaptation constant or step-size parameter.

The Least-Mean-Square (LMS) algorithm

From the steepest-descent algorithm, reproduce:

$$\nabla(J(n)) = -2\underline{p} + 2\underset{\sim}{R}\, w(n).....(1)$$

$$\underline{w}(n+1) = \underline{w}(n) + \frac{1}{2}\mu[-\nabla J(n)].....(2)$$

Define:

$$\underset{\sim}{R}(n) = \underline{u}(n)u^{H}(n).....(3)$$

$$\hat{p}(n) = \underline{u}(n)d^{*}(n).....(4)$$

By using (3) and (4) in (1):

$$\hat{\nabla}(J(n)) = -2\underline{u}(n)d^{*}(n) + 2\underline{u}(n)\underline{u}^{H}(n)\overline{W}(n)...(5)$$

The instantaneous estimate of the gradient vector.

Putting equation (5) in quation (2):

$$\underline{\hat{w}}(n+1) = \underline{\hat{w}}(n) + \mu \underline{u}(n)[d^*(n) - u^H(n)\underline{\hat{w}}(n)]...(6)$$

LMS Algorithm: Three Basic Steps

1. Filter Output:

$$y(n) = \hat{w}^H(n)\underline{u}(n).....(7a)$$

2. Estimation of errors:

$$e(n) = d(n) - y(n)....(7b)$$

3. Tap weight adaptation:

$$\underline{\hat{w}}(n+1) = \underline{\hat{w}}(n) + \underbrace{\mu \underline{u}(n)e^*(n)}_{\text{Correction Term}}...(7c)$$

The algorithm gets initiated with:

$$\underline{\hat{w}}(0) = 0$$

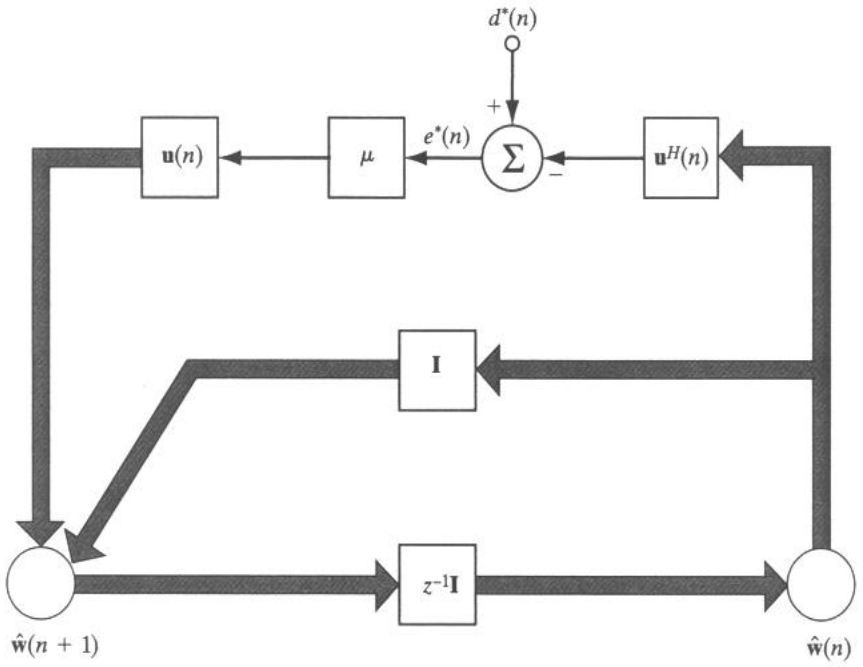


FIGURE 5.2 Signal-flow graph representation of the LMS algorithm.

Notes on the LMS algorithms from the above figure: $\underline{\hat{w}}(n)$

1.) Algorithm is multivariable since has a dimension greater then 1.

2.) Nonlinear since outer feedback loop depends on tap input vector. $\hat{w}(n)$

3.) becomes a random vector fro n > 0.

Stability Analysis of the LMS algorithm:

As a rule of thumb, we have to choose μ so that the following two forms of convergence are satisfied:

Convergence in the mean:

$$E[\hat{w}(n)] \rightarrow \underline{W}_0 \text{ as } n \rightarrow \infty$$

Convergence in the mean square:

$$J(n) \rightarrow J(\infty) \text{ as } n \rightarrow \infty$$

where $J(\infty) > J_{min}$

and $J(\infty)$ is finite

J_{min} corresponds to the weiner solution.

Fundamental assumption for analysis of the LMS algorithm.

1. The tap input vectors $\underline{u}(1)$, $\underline{u}(2)$ …$\underline{u}(n)$ constitute a sequence of statistically independent vectors.

2. At time n, the tap-input vector $\underline{u}(n)$ is statistically independent of all previous samples of the desired response d(1), d(2),…..d(n-1).

3. At time n, the desired response d(n) is dependent on the $\underline{u}(n)$, but statistically indepenedent of all previous samples of the desired response.

4. u(n) and d(n) consist of mutually gaussian-distributed random variables for all n.

With the assumption of mutually gaussian distributed random variables in (4), (1) and (2) are equivalent to conditions of uncorrelatedness:

$$E[\underline{u}(n)u^H(k)] = \underline{0}.....k = 0,1,....n-1$$

$$E[\underline{u}(n)d^*(k)] = 0.....k = 0,1,....n-1$$

the assumption of independence theory [(1) - (3)] is more general when (4) is not assumed.

Average tap weight analysis

Define: $\underline{\varepsilon}(n) = \hat{w}(n) - \underline{w}(0)$

the weight error vector

where $\underline{\hat{w}}(n)$ the estimate produced by the LMS algorithm $\underline{w}_0$ the optimum weiner solution.

Start from:

$$\underline{\hat{w}}(n+1) = \underline{\hat{w}}(n) + \mu \underline{u}(n)[d^*(n) - u^H(n)\underline{\hat{w}}(n)]$$

Rewrite:

$$\underbrace{\underline{\hat{w}}(n+1) - \underline{w}_0}_{\underline{\varepsilon}(n+1)} = \underbrace{\underline{\hat{w}}(n) - \underline{w}_0}_{\underline{\varepsilon}(n)} + \mu \underline{u}(n)[d^*(n) - u^H(n)\underline{\hat{w}}(n)]$$

$$\underline{\varepsilon}(n+1) = \underline{\varepsilon}(n) + \mu \underline{u}(n)[\underbrace{d^*(n) - u^H(n)\underline{\hat{w}}(n)}_{e_0^*(n) - \underline{\varepsilon}^H(n)\underline{u}(n)}]$$

$$\underline{\varepsilon}(n+1) = [I - \mu \underline{u}(n)u^H(n)]\underline{\varepsilon}(n) + \mu \underline{u}(n)e_0^*(n)$$

$$e(n) = d(n) - \hat{w}^H(n)\underline{u}(n)$$

$$= d(n) - w_0^H \underline{u}(n) - \underline{\varepsilon}^H(n)\underline{u}(n)$$

$$= e_0(n) - \underline{\varepsilon}^H(n)\underline{u}(n)$$

e_0: the estimation error produced in the optimum weiner solution.

Now by taking the mathematical expectation:

$$E[\underline{\varepsilon}(n+1)] = E\left[[I - \mu\underline{u}(n)u^H(n)]\underline{\varepsilon}(n)\right] + \mu\left[E\underline{u}(n)e_0^*(n)\right]$$

Since $\underline{\varepsilon}$(n) and $\underline{u}$(n) are independent, the first term:

$$E\left[[I - \mu\underline{u}(n)u^H(n)]\underline{\varepsilon}(n)\right]$$

$$= [I - \mu E[\underline{u}(n)u^H(n)]]E[\underline{\varepsilon}(n)]$$
$$= [I - \mu \underset{\sim}{R}]E[\underline{\varepsilon}(n)]$$

Where:

$$\underset{\sim}{R} = E[\underline{u}(n)\underline{u}^H(n)]$$

From the orthogonality principle:

$$E\left[\underline{u}(n)e_0^*(n)\right] = \underline{0}$$

$$E[\underline{\varepsilon}(n+1)] = [I - \mu \underset{\sim}{R}]E[\underline{\varepsilon}(n)].....(A)$$

Compare equation (A) with that we obtained in the case of steepest-decent algorithm where:

$$\underline{c}(n+1) = [I - \mu \underset{\sim}{R}]\underline{c}(n)$$

We therefore conclude from (A) that:

the mean of $\underline{\varepsilon}(n)$ converges to zero as $n \to \infty$,

provided:

$$0 < \mu < \frac{2}{\lambda_{max}} \qquad \text{(B)}$$

Conclusion: if μ is set as in B

$$E[\underline{\hat{w}}(n)] \to \underline{w_0} \qquad \text{as } n \to \infty$$

that is: the LMS algorithm is convergent in the mean.

Mean-Squared-error Analysis:

Start From:

$$e(n) = d(n) - \underline{\hat{w}}^H(n)\underline{u}(n)$$
$$= d(n) - \underline{w}_0^H \underline{u}(n) - \underline{\varepsilon}^H(n)\underline{u}(n)$$
$$= e_0(n) - \underline{\varepsilon}^H(n)\underline{u}(n)$$

Now:

$$J(n) = E[|e(n)|^2]$$
$$= E[(e_0(n) - \underline{\varepsilon}^H(n)\underline{u}(n))(e_0^*(n) - \underline{\varepsilon}(n)\underline{u}^H(n))]$$
$$= J_{\min} - E[\varepsilon^H(n)\underline{u}(n)\underline{\varepsilon}(n)\underline{u}^H(n)]$$

Where J_{min} is the minimum mean-squared error produced by the optimum weiner filter.

The second term can be manipulated to:

$$E[\varepsilon^H(n)\underline{u}(n)\underline{\varepsilon}(n)\underline{u}^H(n)]$$

$$= tr\{E[\varepsilon^H(n)\underline{u}(n)\underline{\varepsilon}(n)\underline{u}^H(n)]\}$$
$$= tr\{\underset{\sim}{R}\,\underset{\sim}{K}(n)\}$$

$$J(n) = J_{\min} - tr[\underline{R}\,\underline{K}(n)] \qquad \text{(C)}$$

Where $\underset{\sim}{R}$: correlation matrix

$\underset{\sim}{K}$: the weight-error correlation matrix

Conclusion: since $tr\{\underset{\sim}{R}\,\underset{\sim}{K}(n)\}$ is positive definite for all *n*. LMS allways produces a mean-squared error J(n) that is in excess of the minimum-mean-squared-error $J_{\min}$

Rewrite equation (C):

$$J(n) = J_{\min} - tr\{\underset{\sim}{R}\,\underset{\sim}{K}(n)\} \qquad \text{(C)}$$

= the excess mean squared error.

By transforming to eigen-coordinate system:

$$Q^H \underset{\sim}{R}\,\underset{\sim}{Q} = \underset{\sim}{\Lambda}$$

and defining:

$$Q^H \underset{\sim}{K}(n)\underset{\sim}{Q} = \underset{\sim}{X}(n)$$

We can write:

$$tr[\underset{\sim}{R}\,\underset{\sim}{K}(n)] = tr[\underset{\sim}{\Lambda}\,\underset{\sim}{X}(n)]$$

$$J_{ex}(n) = tr[\underset{\sim}{\Lambda}\,\underset{\sim}{X}(n)] \qquad \text{(D)}$$

In diagonal coordinate system:

$$J_{ex}(n) = \sum_{i=1}^{M} \lambda_i x_i(n) \qquad \text{(E)}$$

where $x_i(n)$, I=1,2,……M, diagonal elements of $\underline{X}(n)$.

Equation (E) can be re-written as:

$$J_{ex}(n) = \underline{\lambda}^T x(n)$$

Where $\underline{x}(n)$ satisfies

$$\underset{\sim}{x}(n+1) = \underset{\sim}{B}\,\underline{x}(n) + \mu^2 J_{\min}\,\underline{\lambda} \qquad \text{(F)}$$

where $\underset{\sim}{B}$ is the MxM matrix with:

$$b_{ij} = \begin{cases} (1-\mu\lambda_i)^2 + \mu^2\lambda_i^2 \;\ldots\ldots\ldots i = j \\ \mu^2\lambda_i\lambda_j \;\ldots\ldots\ldots i \neq j \end{cases}$$

The matrix B is real, positive and symmetric.

By solving the first order vector differennce equation (F) gives:

$$\underline{x}(n) = \underset{\sim}{B}^{n}\,\underline{x}(0) + \mu^{2} J_{\min} \sum_{i=0}^{n-1} \underset{\sim}{B}^{i}\,\underset{\sim}{\lambda} \qquad \text{(G)}$$

$$\sum_{i=0}^{n-1} \underset{\sim}{B}^{i} = (\underset{\sim}{I} - \underset{\sim}{B})^{n} (\underset{\sim}{I} - \underset{\sim}{B})^{-1}$$

Then equation (G) becomes:

$$\underline{x}(n) = \underline{B}^{n}[\underline{x}(0) - \mu^{2} J_{\min} (\underset{\sim}{I} - \underset{\sim}{B})^{-1} \underline{\lambda}] + \mu^{2} J_{\min} (\underset{\sim}{I} - \underset{\sim}{B})^{-1} \underline{\lambda}$$

transient Component

Steady state Component

Since $\underset{\sim}{B}$ is symmetric, we can apply an orthogonal similarity transformation.

$$\underset{\sim}{G}^T \; \underset{\sim}{B} \, \underset{\sim}{G} = \underline{C}$$

G: orthogonal matrix

$$\underset{\sim}{G}^T \; \underset{\sim}{G} = \underset{\sim}{I}$$

$$\underset{\sim}{B}^n = \underset{\sim}{G} \, \underset{\sim}{C}^n \, \underset{\sim}{G}^T = \underline{C}$$

$\underline{C}$: Diagonal matrix with C_i, i = 1,2, …..M

g_i : eigen vectors of B associated with eigenvalues c_i

We can then manipulate the equation for $\underline{x(n)}$ to:

$$\underline{x}(n) = \sum_{i=1}^{M} c_i^n \, \underline{g}_i \, \underline{g}_i^T [\underline{x}(0) - \underline{x}(\infty)] + \underline{x}(\infty) \qquad \text{(H)}$$

Where for stability:

$0 < c_i < 1$ for all i

$$x(\infty) = \mu^2 J_{\min} (\underset{\sim}{I} - \underset{\sim}{B})^{-1} \underline{\lambda}$$

And:

$$\underset{\sim}{G} \, \underset{\sim}{C}^n \, \underset{\sim}{G}^T = \sum_{i=1}^{M} c_i^n \, \underline{g}_i \, \underline{g}_i^T$$

Now the excess mean-squared error:

$$J_{ex}(n) = \underline{\lambda}^T \underline{x}(n)$$

$$= \sum_{i=1}^{M} c_i^n \, \underline{\lambda}^T \, \underline{g}_i \, \underline{g}_i^T [\underline{x}(0) - \underline{x}(\infty)] + J_{ex}(\infty) \qquad \text{(I)}$$

Where $J_{ex}(\infty) = \underline{\lambda}^T \underline{x}(\infty)$

$$= \sum_{j=1}^{M} \lambda_j x_j(\infty)$$

After some more algebra:

We may write an expression for:

$$J_{ex}(\infty) = J_{\min} \frac{\sum_{i=1}^{M} \frac{\mu\lambda_i}{2-\mu\lambda_i}}{1-\sum_{i=1}^{M} \frac{\mu\lambda_i}{2-\mu\lambda_i}} \qquad \text{(J)}$$

Now Since:

$$J_{ex}(n) = J(n) - J_{\min}$$
$$= tr\{\underset{\sim}{R}\,\underset{\sim}{K}(n)\}$$

Putting equations (J) in (I), the time evolution of the mean squared error for the LMS algorithm:

$$J(n) = \sum_{i=1}^{M} r_i C_i^n + \frac{Jim}{1-\sum_{i=1}^{M} \frac{\mu\lambda_i}{2-\mu\lambda_i}} \qquad \text{(K)}$$

Where: $r_i = \underline{\lambda}^T \underline{g}_i g_i^T [\underline{x}(0) - \underline{x}(\infty)]$

$x_i(0) = E[q_i^H \underline{\varepsilon}(0)\varepsilon^H(0)\underline{q}_i] \ldots . i = 1,2,\ldots . M$

$\underline{\varepsilon}(0) = \hat{w}(0) - \underline{w}(0)$

q: eigen vector of R

Conclusions from equation (K)

a.) The first term is (K)

$$\sum_{i=1}^{M} r_i c_i^n$$

Represents the transient component of J(n) where r_i are constants and c_i are the eigenvalues of $\underline{B}$ and they are real positive numbers since $\underline{B}$ is a real symmetric positive definite matrix.

b.) $J(n) \rightarrow J(\infty)$ if, and only if, μ: satisfies two conditions:

$$0 < \mu < \frac{2}{\lambda_{\max}}$$

$$\sum_{i=1}^{M} \frac{\mu\lambda_i}{2 - \mu\lambda_i} < 1$$

λ_i I=1,2,….M, eigenvalues of $\underset{\sim}{R}$.

The LMS algorithm is convergent in the mean square.

c. The mean squared error produced by the LMS algorithm has the final value:

$$J(\infty) = \frac{J_{\min}}{1 - \sum_{i=1}^{M} \frac{\mu\lambda_i}{2 - \mu\lambda_i}}$$

d. The misadjustment is defined as:

$$M = \frac{J_{ex}(\infty)}{J_{\min}}$$

$$M = \frac{\sum_{i=1}^{M} \frac{\mu\lambda_i}{2 - \mu\lambda_i}}{1 - \sum_{i=1}^{M} \frac{\mu\lambda_i}{2 - \mu\lambda_i}}$$

(i) If μ is small compared to $2/\lambda_{max}$

$$M \approx \frac{\mu}{2} \sum_{i=1}^{M} \lambda_i$$

(ii) By defining:

$$\lambda_{av} = \frac{1}{M} \sum_{i=1}^{M} \lambda_i$$

We can define the convergence time constant for the LMS algorithm:

$$(\tau)_{mse,av} \approx \frac{1}{2\mu\lambda_{av}}$$

$$\mathrm{M} = \frac{\mu M \lambda_{av}}{2} \approx \frac{M}{4\tau_{mse,av}}$$

Note that we have conflicting requirements in that if μ is reduced to reduce $\boldsymbol{M}$ then $(\tau)_{\text{mse av}}$ is increased. Conversely if μ is increased to reduce $(\tau)_{\text{mse av}}$ then $\boldsymbol{M}$ is increased. The choice of μ becomes an important compromise.

For real data, it should be noted the real LMS algorithm in the mean square is give by:

$$\sum_{i=1}^{M} \frac{\mu\lambda_i}{2-\mu\lambda_i} < 1$$