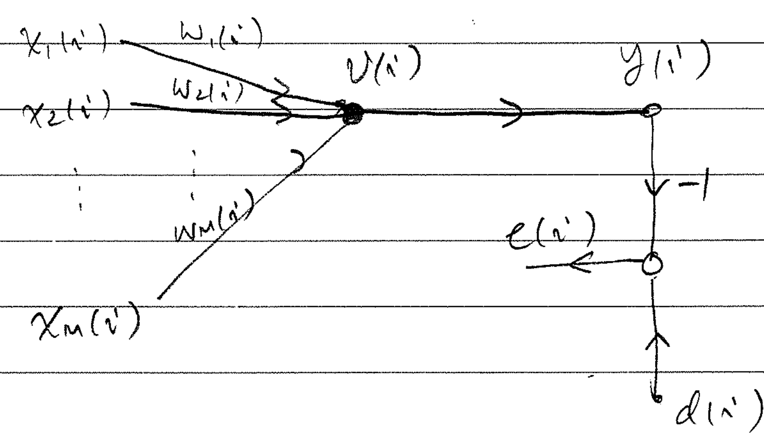
The Least-Mean-Square Algorithm

- the 1st linear adaptive filtering algorithm
- applications include prediction, communication-channel equalization, ... system ID
- inspired by perceptron, linear combiner
- "linear" complexity yet effective in practice
- simple to code
- robust to disturbance
- set the stage for backpropagation

Filtering structure

$X(i) = [x_1(i), x_2(i), \dots, x_M(i)]^T$

$W(i) = [w_1(i), w_2(i), \dots, w_M(i)]^T$



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

$y(i) = \sum_{k=1}^{M} w_k(i)\, x_k(i)$

$y(i) = X^T(i)\, W(i)$

where $i = 1, 2, \dots, n$ denotes time index

sample pair $\mathcal{T} : \{ x(i), d(i) ;\ i = 1, \dots, n, \dots \}$

① $X(i)$ can be a snapshot of data

② $X(i)$ can be past values of $X(i-1), \dots$

e.g. $y(i) = w_1(i)\, x_1(i) + w_2(i)\, x_1(i-1) + \dots + w_M(i)\, x_1(i-(M-1))$

The Least-Mean Square Algorithm.

Introduce the instantaneous cost function

$$\mathcal{E}(\hat{w}) = \frac{1}{2} e^2(n)$$

for a linear neuron,

$$e(n) = d(n) - x^T(n)\,\hat{w}(n)$$

where $x(n)$ - input vector

$d(n)$ - desired response

$\hat{w}(n)$ - weight vector.

then $$\frac{\partial \mathcal{E}(\hat{w})}{\partial \hat{w}} = e(n) \frac{\partial e(n)}{\partial \hat{w}}$$

and $$\frac{\partial e(n)}{\partial \hat{w}} = -x(n).$$

$$\Rightarrow \quad \frac{\partial \mathcal{E}(\hat{w})}{\partial \hat{w}(n)} = -x(n)\, e(n)$$

or $$\hat{g}(n) = -x(n)\, e(n)$$

The LMS algorithm:

$$\hat{w}(n+1) = \hat{w}(n) + \eta\, x(n)\, e(n).$$

Wiener Filter

consider an LTI system

$$x(t) \longrightarrow \boxed{LTI} \longrightarrow y(t)$$

$$\underset{\text{output}}{y(t)} = \underset{\substack{\text{impulse} \\ \text{response} \\ \text{(parameter } w)}}{h_w(t)} * (\underset{\text{input}}{x(t)} + \underset{\text{noise}}{n(t)})$$

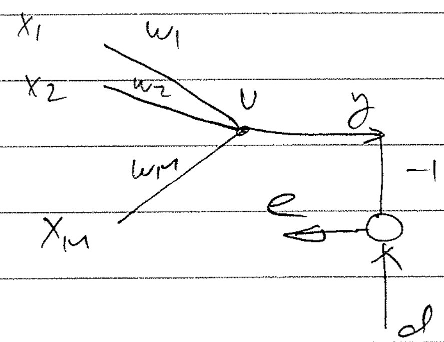
Filter – causal system.

Now consider unknow, stationary, stochastic system,

$$\underset{\substack{\text{to probe} \\ \text{the environment}}}{x} \longrightarrow \boxed{w} \longrightarrow y$$

↑ scalar w/o loss of generality

$$y = \sum_{j=1}^{M} w_j x_j$$



in matrix form

$$y = w^T x$$

where $x = (x_1 \; x_2 \; \cdots \; x_M)^T$

$w = (w_1 \; w_2 \; \cdots \; w_M)^T$

Wiener filter can be spatial

$$y = \sum_{k=1}^{M} w_k x_k$$

or temporal, e.g. MA filter.

Let d: desired response

e: error signal.

$$e = d - y$$

$\mathcal{E}$: mean-square error.

$$\mathcal{E} = \frac{1}{2} E(e^2) = \frac{1}{2} E(d-y)^2$$

The linear optimal filter problem:

Determin the optimal set of weights, $w_{01}, w_{02}, \ldots w_{0M}$, for which the mean-squared error $\mathcal{E}$ is minimized

→ the solution of the linear optimal filter problem is the Wiener filter.

Detailed derivations:

$$\mathcal{E} = \frac{1}{2} E(e^2) = \frac{1}{2} E\left((d-y)^2\right)$$

$$= \frac{1}{2} E(d^2) - E\left(\sum_{k=1}^{M} w_k x_k d\right) + \frac{1}{2} E\left(\sum_{j=1}^{M} \sum_{k=1}^{M} w_j w_k x_j x_k\right)$$

$$= \frac{1}{2} E(d^2) - \sum_{k=1}^{M} w_k E(x_k d) + \frac{1}{2} \sum_{j=1}^{M} \sum_{k=1}^{M} w_j w_k E(x_j x_k)$$

let $r_d = E(d^2)$: MS value of desired response

$r_{dx}(k) = E(d x_k)$, $k = 1, \dots, M$, cross-correlation

$r_x(j, k) = E(x_j x_k)$, $j, k = 1, \dots, M$, auto-correlation

Therefore:

$$\mathcal{E} = \frac{1}{2} r_d - \sum_{k=1}^{M} w_k r_{dx}(k) + \frac{1}{2} \sum_{j=1}^{M} \sum_{k=1}^{M} w_j w_k r_x(j, k)$$

necessary condition of optimal $\mathcal{E}$,

$$\nabla_{w_k} \mathcal{E} = 0, \quad k = 1, 2, \dots, M$$

where $\nabla_{w_k} \mathcal{E} = \frac{\partial \mathcal{E}}{\partial w_k} \quad k = 1, \dots, M$

$$= -r_{dx}(k) + \sum_{j=1}^{M} w_j r_x(j, k)$$

The Wiener filter:

$$\sum_{j=1}^{M} w_{oj} r_x(j, k) = r_{xd}(k), \quad k = 1, \dots, M$$

In matrix form

$$R_{xx} W_o = R_{xd}$$

$$W_o = R_{xx}^{-1} R_{xd}$$

Remark: R_{xx} is a Toeplitz Matrix

using Levinson-Durbin recursion to solve, $O(M_w^2)$

Alternatively, one can use steepest descent.

Since $\nabla_{w_k} \mathcal{E}(n) = -\gamma_{dx}(k) + \sum_{j=1}^{M} w_j(n) \gamma_x(j,k)$

$$\Delta w_k(n) = -\eta \frac{\partial \mathcal{E}}{\partial w_k(n)} = -\eta \nabla_{w_k} \mathcal{E}(n)$$

$k=1, \ldots, M$

or $w_k(n+1) = w_k(n) + \Delta w_k(n)$

$$= w_k(n) - \eta \nabla_{w_k} \mathcal{E}(n)$$

$$= w_k(n) + \eta \left[\gamma_{dx}(k) - \sum_{j=1}^{M} w_j(n) \gamma_x(j,k) \right],$$

$k=1, \ldots, M.$

Reminder in the above,

$$\mathcal{E}(n) = \frac{1}{2} E\left[e^2(n)\right].$$

If we let

$$\hat{r}_x(j,k;n) = x_j(n)\,x_k(n)$$

$$\hat{r}_{dx}(k;n) = x_k(n)\,d(n).$$

$$\Rightarrow \hat{w}_k(n+1) = \hat{w}_k(n) + \eta\left[x_k(n)d(n) - \sum_{j=1}^{M} \hat{w}_j(n)\,x_j(n)\,x_k(n)\right]$$

$$= \hat{w}_k(n) + \eta\left[d(n) - \sum_{j=1}^{M} \hat{w}_j(n)\,x_j(n)\right] x_k(n)$$

The LMS! $$= \hat{w}_k(n) + \eta\left[d(n) - y(n)\right] x_k(n)$$

$$k = 1, \ldots, M$$

where $$y(n) = \sum_{j=1}^{M} \hat{w}_j(n)\,x_j(n)$$

Remarks on LMS algorithm

- "approximate", however very simple
- minimizing instantaneous error $\mathcal{E}(n)$ where

$$\mathcal{E}(n) = \frac{1}{2} e^2(n)$$

- no memory requirement
- optimal in H^∞ - cater to worst-case scenario
- slow convergence especially if poor condition # of R_{xx}, or sensitive to condition # of R_{xx}
- Convergence of the LMS depends on statistical property of $x(n)$ and learning rate η.

convergence in the mean

$$E[\hat{w}(n)] \to w_o \quad \text{as} \quad n \to \infty$$

convergence in MS sense

$$E[e^2(n)] \to \text{constant} \quad \text{as} \quad n \to \infty$$

$$\text{conv. in MS} \underset{\not\leftarrow}{\rightarrow} \text{conv in mean}$$

Sufficient condition for convergence,

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

relaxed $\quad 0 < \eta < \frac{2}{tr[R_x]}$

where $R_{xx} = E\left[\bar{X}(n)\,\bar{X}^T(n)\right]$

λ_{max} – largest eigenvalue of R_{xx}

- η in practice (annealing schedules)

LMS : $\eta(n) = \eta_0$ = small constant for all n

Robbins-Monro : $\eta(n) = \frac{c}{n}$, where c is constant

reference Lin & Si in Neural Comp 1998

Darken-Moody: $\eta(n) = \dfrac{\eta_0}{1 + n/\tau}$ where η_0, τ are chosen constants

can be more aggressive initially,

approaches Robbins-Monro as $n \to \infty$.