Least Mean Squares and Recursive Least Squares

Signal with noise

- $d(n) = s(n) + n(n)$
- $x(n) := \mathrm{F}[n(n)]$ where F means an operator to apply on the series $n(n)$, such as: difference, delay, summing up

We want to use the history data of $x(n)$ to predict $d(n)$, the predicted value is $y(n) = (x[n], \cdots, x[n-(N-1)])^\top \cdot w$, [which requires $ds/dt << dn/dt$, then we can use $d(n) - y(n)$ to approximate $s(n)$]

treat random variables $x(n), \ldots, x(n-N+1)$ and $d(n)$ independent for each n

$$\begin{aligned} X &\equiv [x_0, x_1, \ldots, x_{N-1}] \\ d &\quad \text{our target} \end{aligned}$$

now find the best parameter w for $y = X^\top \cdot w$ to predict d, we name the difference between them as $e \equiv d - X^T w$, which approximates signal s

$$\sum_{n'=1}^{n} e^2(n') = \sum_{n'=1}^{n} [X^T(n') \cdot w - d(n')]^T [X^T(n') \cdot w - d(n')]$$

As above, consider total n samples $[X(n), d(n)]$

Wiener Filter - the minimal error energy

$$e_w^2(n) = \sum_{n'=1}^{n} e^2(n') = \sum_{n'=1}^{n} [X^T(n') \cdot w - d(n')]^T [X^T(n') \cdot w - d(n')]$$

Define:

$$R(n) \equiv [\sum_{n'=0}^{n} X^T(n') X(n')]$$
$$P(n) \equiv [\sum_{n'=0}^{n} d(n') X(n')]$$
$$\sigma^2(n) \equiv [\sum_{n'=0}^{n} d^2(n')]$$

Then we have

$$e_w^2(n) = w^T R(n) w - 2P^T(n) w + \sigma^2(n)$$

To minimize the $e_w^2(n)$, we can make sure

$$\frac{de_w^2(n)}{dw} = 2R(n)w - 2P(n) = \vec{0}$$
$$w = R^{-1}(n)P(n)$$

Assumption: Random Process x is a **wide-sense ergodic process**(must be an **wide-sense stationary process**), x, d are **jointly wide-sense ergodic**, as $n \to \infty$, we have

$$\lim_{n' \to \infty} \frac{1}{n} \sum_{n'=0}^{n} x(n'-k)x(n'-k-\tau) = \mathrm{E}[x(n'-k)x(n'-k-\tau)] = r_x(\tau) \quad [\tau = 0, \ldots, N-1]$$
$$\lim_{n' \to \infty} \frac{1}{n} \sum_{n'=0}^{n} d(n')x(n'-\tau) = r_{d,x}(\tau)$$

for element $\lim_{n \to \infty} \frac{1}{n}[R(n)]_{i,j} = r_x(\tau),\ \text{where } k = \min(i,j), \tau = |i-j|$

for element $\lim_{n \to \infty} \frac{1}{n}[P(n)]_i = r_{d,x}(\tau)$, where $\tau = i$

$$\lim_{n \to \infty} w = \lim_{n \to \infty} (\frac{1}{n} R)^{-1}(n) \frac{1}{n} P(n) = [r_x(|i-j|)]^{-1}[r_{d,x}(\tau)] \equiv R^{-1}P$$

note: $R^{-1}P$ can be solved by **Levinson–Durbin Recursion**

- https://en.wikipedia.org/wiki/Levinson_recursion
- https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-341-discrete-time-signal-processing-fall-2005/lecture-notes/lec13.pdf
- http://sepwww.stanford.edu/sep/prof/fgdp/c7/paper_html/node6.html
- https://link.springer.com/content/pdf/bbm%3A978-0-387-68899-2%2F1.pdf

Remark on Assumption

Random Process x is a **wide-sense ergodic process**(must be an **wide-sense stationary process**), x, d are **jointly wide-sense ergodic**, as $n \to \infty$, we have

Random Process x is a **wide-sense ergodic process**(must be an **wide-sense stationary process**),

- for an **wide-sense stationary process**: for $\forall t, t+h \in T$
  $(1)\ \ E[X(t)] = \mu_X \ \ \text{(const)} \qquad (2)\ \ E[X(t)X(t+\tau)] = r_X(\tau)$
- https://en.wikipedia.org/wiki/Ergodic_process

  for constant $\mu_X, r_X(\tau)$

  we compute the estimate with samples among $[0, T]$

$$\hat{\mu}_X = \frac{1}{T}\int_0^T X(t)dt$$

  $\xrightarrow{L^2}$ the expectation $\mu_X = E[X]$ as $T \to \infty$

$$\hat{r}_X(\tau) = \frac{1}{T}\int_0^T [X(t+\tau) - \mu_X]\,[X(t) - \mu_X]dt$$

  $\xrightarrow{L^2}$ the expectation $r_X(\tau) = E[X(t)X(t+\tau)]$ as $T \to \infty$

LMS - least mean squares filter

We approximates the n-th sample of $X \cdot e : X(n)e(n)$ as the expectation of $X \cdot e$

$$\frac{d\mathrm{E}[e^2]}{dw} = 2\mathrm{E}[X(X^T \cdot w - d)] = -2\mathrm{E}[Xe] \approx -2X(n)e(n)$$

Notice **Note3**, it requires that the inequality holds for the learning rate α and the eigenvalues of R: $\lambda_i > 0$ to ensure the convergence of our method

$$0 < \alpha < \frac{1}{\lambda_{\max}}$$

Notice **Note1, Note2**

When $n \to \infty, r_{ii} \to \mathrm{E}[x^2(n' - i + 1)] = \mathrm{E}[x^2(n')] = P_x$,

$$\lambda_{\max} < \sum \lambda_i = \mathrm{tr}(R) = \sum r_{ii} < N \cdot r_{11} = NP_x$$

we can choose α to ensure the convergence of our method

$$\alpha = \frac{1}{NP_x}$$

Note 1

$$\det(tI - A) = t^N - (\mathrm{tr}\, A)t^{N-1} + \cdots + (-1)^N \det A$$

The coefficient of t_{N-1}, t_{N-1} must come from $(t - a_{11}) \cdots (t - a_{NN})$,

because any term involving an off-diagonal$(i \neq j)$ element $[tI - A]_{ij}$ eliminates $t - a_{ii}$ and $t - ajj$,

hence any such term does not involve t^{N-1}.

So, the coefficient of t^{N-1} is $\mathrm{tr}(A) = -\sum a_{ii}$

$$\det(tI - A) = \prod_{i=1}^{N}(t - \lambda_i) = t^n - (\sum \lambda_i)t^{n-1} + \ldots$$

Thus

$$\sum \lambda_i = \mathrm{tr}(A)$$

recursive least squares

Note 2

Consider

$$R \equiv \frac{1}{n} \sum_{n'=1}^{n} X(n') X^T(n')$$

R must be positive definite

because $\forall p \in R^n = [p_1, p_2, \ldots, p_n]$,

$$p^T R \cdot p = \frac{1}{n} \sum_{n'=1}^{n} [p^T X(n') \cdot X^T(n') p] = \frac{1}{n} \sum_{n'=1}^{n} [X^T(n') p]^2 > 0$$

Here $X^T(n')p = 0$ can NOT be satisfied for all n'

So, for all λ_i, having $\lambda_i > 0$, thus

$$\lambda_{\max} < \sum \lambda_i = \operatorname{tr}(R)$$

When $n \to \infty, r_{ii} \to \mathrm{E}[x^2(n' - i + 1)] = \mathrm{E}[x^2(n')] = P_x$,

$$\lambda_{\max} < \sum \lambda_i = \operatorname{tr}(R) = \sum r_{ii} < N \cdot r_{11} = N P_x$$

and notice $P_x = \frac{1}{n} \sum_{n'=0}^{N-1} x^2(n') = r_{11} = r_{ii} + \frac{1}{n} \sum_{j=0}^{i-1} x^2(n-j)$

Note 3

Suppose that the performance index is a quadratic function:

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{d}^T\mathbf{x} + c\,. \tag{9.18}$$

From Eq. (8.38) the gradient of the quadratic function is

$$\nabla F(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{d}\,. \tag{9.19}$$

If we now insert this expression into our expression for the steepest descent algorithm (assuming a constant learning rate), we obtain

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha\mathbf{g}_k = \mathbf{x}_k - \alpha(\mathbf{A}\mathbf{x}_k + \mathbf{d}) \tag{9.20}$$

or

$$\mathbf{x}_{k+1} = [\mathbf{I} - \alpha\mathbf{A}]\mathbf{x}_k - \alpha\mathbf{d}\,. \tag{9.21}$$

This is a linear dynamic system, which will be stable if the eigenvalues of the matrix $[\mathbf{I} - \alpha\mathbf{A}]$ are less than one in magnitude (see [Brog91]). We can express the eigenvalues of this matrix in terms of the eigenvalues of the Hessian matrix $\mathbf{A}$. Let $\{\lambda_1, \lambda_2, \dots, \lambda_n\}$ and $\{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n\}$ be the eigenvalues and eigenvectors of the Hessian matrix. Then

$$[\mathbf{I} - \alpha\mathbf{A}]\mathbf{z}_i = \mathbf{z}_i - \alpha\mathbf{A}\mathbf{z}_i = \mathbf{z}_i - \alpha\lambda_i\mathbf{z}_i = (1 - \alpha\lambda_i)\mathbf{z}_i\,. \tag{9.22}$$

Therefore the eigenvectors of $[\mathbf{I} - \alpha\mathbf{A}]$ are the same as the eigenvectors of $\mathbf{A}$, and the eigenvalues of $[\mathbf{I} - \alpha\mathbf{A}]$ are $(1 - \alpha\lambda_i)$. Our condition for the stability of the steepest descent algorithm is then

why

$$|(1 - \alpha\lambda_i)| < 1\,. \tag{9.23}$$

If we assume that the quadratic function has a strong minimum point, then its eigenvalues must be positive numbers. Eq. (9.23) then reduces to

$$\alpha < \frac{2}{\lambda_i}\,. \tag{9.24}$$

Where $R = \frac{1}{2}A$, so the eigenvalues of R: $\lambda_i > 0$ should $\frac{1}{2}$ of the corresponding eigenvalues of A, thus:

$$|1 - \alpha 2\lambda_i| < 1 \quad \forall\lambda_i$$

$$\alpha 2\lambda_i - 1 < 1 \quad \forall\lambda_i$$

$$0 < \alpha < \frac{1}{\lambda_i} \quad \forall\lambda_i$$

$$0 < \alpha < \frac{1}{\lambda_{\max}}$$

RLS - recursive least squares filter

$(\lambda < 1, \delta >$ Big Number), consider the error

$$
\begin{aligned}
e_w^2(n) &\equiv \lambda^{n+1} w^T (\frac{1}{\delta} I) w + \sum_{n'=0}^{n} \lambda^{n-n'} e^2(n') \\
&= \lambda^{n+1} w^T (\frac{1}{\delta} I) w + \sum_{n'=0}^{n} \lambda^{n-n'} [X^T(n') \cdot w - d(n')]^T [X^T(n') \cdot w - d(n')] \\
&= w^T [\lambda^{n+1} (\frac{1}{\delta} I) + \sum_{n'=0}^{n} \lambda^{n-n'} X^T(n') X(n')] w - 2[\sum_{n'=0}^{n} \lambda^{n-n'} d(n') X^T(n')] w + [\sum_{n'=0}^{n} \lambda^{n-n'} d
\end{aligned}
$$

Define:

$$
\begin{aligned}
R(n) &\equiv [\lambda^{n+1} (\frac{1}{\delta} I) + \sum_{n'=0}^{n} \lambda^{n-n'} X^T(n') X(n')] \\
P(n) &\equiv [\sum_{n'=0}^{n} \lambda^{n-n'} d(n') X(n')] \\
\sigma^2(n) &\equiv [\sum_{n'=0}^{n} \lambda^{n-n'} d^2(n')]
\end{aligned}
$$

Then we have

$$
e_w^2(n) = w^T R(n) w - 2P^T(n) w + \sigma^2(n)
$$

To minimize the $e_w^2(n)$, we can make sure

$$
\begin{aligned}
\frac{de_w^2(n)}{dw} = 2R(n)w - 2P(n) = \vec{0} \\
w = R^{-1}(n) P(n)
\end{aligned}
$$

estimate $R^{-1}(n), P(n)$

We need derive the expression to update $R(n)^{-1}, P(n)$, which should be easy to compute，then we can update w

We define $Q(n) \equiv R^{-1}(n)$

We know the initial value, $R(-1) = \frac{1}{\delta} I$, and

$$R(n) = \lambda R(n-1) + X(n)X^T(n)$$

With Woodbury matrix identity

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

With Sherman–Morrison formula

$$\left(A + uv^\top\right)^{-1} = A^{-1} - \frac{A^{-1}uv^\top A^{-1}}{1 + v^\top A^{-1}u}$$

Where $A = \lambda R(n-1), u = X(n), v = X(n)$

$$\begin{aligned}
Q(n) = R^{-1}(n) &= \lambda^{-1}R^{-1}(n-1) - \lambda^{-2}\frac{R^{-1}(n-1)X(n)X^T(n)R^{-1}(n-1)}{1 + \lambda^{-1}X^T(n)R^{-1}(n-1)X(n)} \\
&= \lambda^{-1}\left[I - \frac{\lambda^{-1}R^{-1}(n-1)X(n)}{1 + \lambda^{-1}X^T(n)R^{-1}(n-1)X(n)}X^T(n)\right]R^{-1}(n-1) \\
&= \lambda^{-1}\left[I - \frac{\lambda^{-1}Q(n-1)X(n)}{1 + \lambda^{-1}X^T(n)Q(n-1)X(n)}X^T(n)\right]Q(n-1)
\end{aligned}$$

Therefore, we can define a **vector** $k(n)$ computed based on **known** $Q(n-1), X(n)$

$$\begin{aligned}
k(n) &\equiv \left[\frac{\lambda^{-1}Q(n-1)X(n)}{1 + \lambda^{-1}X^T(n)Q(n-1)X(n)}\right] \\
Q(n) &= \lambda^{-1}\left[I - k(n)X^T(n)\right]Q(n-1)
\end{aligned}$$

Notice

$$\begin{aligned}
k(n) + k(n)\lambda^{-1}X^T(n)Q(n-1)X(n) &= \lambda^{-1}Q(n-1)X(n) \\
k(n) &= \lambda^{-1}\left[I - k(n)X^T(n)\right]Q(n-1)X(n)
\end{aligned}$$

Therefore, we have

$$k(n) = Q(n)X(n)$$

update $P(n)$

Expression to update the **vector** $P(n)$ based on the **known** $P(n-1), d(n), X(n)$

$$P(n) = \lambda P(n-1) + d(n)X(n)$$

update $w(n)$

$$\begin{aligned}
w(n) &= R^{-1}(n)P(n) = Q(n)P(n) \\
&= \lambda^{-1}\Bigg[I - k(n)X^T(n)\Bigg]Q(n-1)\Bigg[\lambda P(n-1) + d(n)X(n)\Bigg] \\
&= \lambda^{-1}\Bigg[I - k(n)X^T(n)\Bigg]\Bigg[\lambda Q(n-1)P(n-1) + d(n)Q(n-1)X(n)\Bigg] \\
&= \Bigg[I - k(n)X^T(n)\Bigg]w(n-1) + d(n)k(n) \\
&= w(n-1) - [X^T(n)w(n-1) - d(n)]k(n)
\end{aligned}$$

Optional, define a **scalar** $\alpha(n) \equiv [d(n) - X^T(n)w(n-1)]$, then

$$w(n) = w(n-1) + \alpha(n)k(n)$$

because $d(-1) = 0,$ so
$w(-1) = Q(-1)P(-1) = Q(-1)[d(-1)X(-1)] = Q(-1) \cdot \vec{0} = 0$

Summary

$$w(n) = \Big[w(n-1) + \alpha(n)k(n)\Big] \longleftarrow [w(n-1)], \alpha(n), k(n)$$

$$\alpha(n) = \Big[d(n) - X^T(n)w(n-1)\Big] \longleftarrow [w(n-1)], X(n), d(n)$$

$$k(n) = \left[\frac{\lambda^{-1}Q(n-1)X(n)}{1 + \lambda^{-1}X^T(n)Q(n-1)X(n)}\right] \longleftarrow [\lambda, Q(n-1)], X(n)$$

$$Q(n) = \lambda^{-1}\left[I - k(n)X^T(n)\right]Q(n-1) \longleftarrow [\lambda, Q(n-1)], k(n), X(n)$$

calculation order:

$k(n)$	**1ST**	$[\lambda, Q(n-1)], X(n)$
$Q(n)$	2nd	$[\lambda, Q(n-1)], k(n), X(n)$
$\alpha(n)$	3rd	$[w(n-1)], X(n), d(n)$
$w(n)$	4th	$[w(n-1)], \alpha(n), k(n)$

Initialization: given value for λ, set $Q(-1) = \delta I, w(-1) = \vec{0}$

where we can choose $\delta = \frac{1}{\mathrm{E}[x^2(n')]}$, to ensure the algorithm is stable