

Procesamiento de señales I

86.51

Cecilia G. Galarza

FIUBA

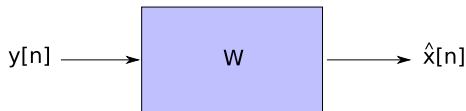
Laboratorio de Procesamiento de Señales y Comunicaciones

2do Cuatrimestre 2022

Filtrado Adaptativo

Filtro de Wiener

Vimos que es posible estimar un proceso a partir de observaciones ruidosas del mismo utilizando el filtro de Wiener (FIR o IIR).



$$\hat{x}(n) = \sum_{k=-\infty}^n w(n-k)y(n)$$

donde $\{w(n)\}$ es tal que minimiza

$$\mathbb{E} [\|x - \hat{x}\|^2]$$

Hipótesis de diseño

- 1 $x(n)$ e $y(n)$ son conjuntamente ESA, es decir:
 - ▶ $\mathbb{E}[x(n)] = \mu_x$, $\mathbb{E}[y(n)] = \mu_y$
 - ▶ $\mathbb{E}[x(n)x^*(n+k)] = R_x(k)$, $\mathbb{E}[y(n)y^*(n+k)] = R_y(k)$
 - ▶ $\mathbb{E}[x(n)y^*(n+k)] = R_{xy}(k) = R_{yx}^*(-k)$
- 2 Conocemos $R_y(k)$ y $R_{xy}(k)$.

Hipótesis de diseño

Qué sucede si

- Los procesos no son estacionarios?
 - ▶ Ejemplo 1: $x(n)$ tiene una deriva en el tiempo y $\mathbb{E}[x(n)] = \mu_x(n)$.
 - ▶ Ejemplo 2: $x(n)$ es una constante, pero la fuente del ruido que afecta la observación varía con el tiempo.
- No se conoce la estadística de $x(n)$ o de $y(n)$?
 - ▶ Por ejemplo, se sabe que $y(n) = x(n) + v(n)$, pero no se tienen observaciones previas que permitan estimar R_y y/o R_{xy} .

Un modo de encarar este problema es estimar $x(n)$ a medida que se colectan las observaciones, es decir, ir *adaptando* el diseño del filtro a medida que se observa $y(n)$.

Filtrado FIR adaptativo

En esta clase, vamos a plantear el problema adaptativo considerando una ventana finita de observaciones.

- Señal escalar: $x(n) \in \mathbb{C}$
- Medias nulas: $\mathbb{E}[x(n)] = \mathbb{E}[y(n)] = 0$
- Filtrado FIR: $\hat{x}(n) = \sum_{l=0}^M w(l)y(n-l)$
- Criterio MMSE: $\mathbb{E}[|\hat{x}(n) - x(n)|^2]$ es pequeña para todo n .


Filtrado FIR

El problema de filtrado con ventana finita utiliza un filtro FIR del tipo:

$$\begin{aligned}\hat{x}(n) &= \sum_{l=0}^M w(l)y(n-l) \\ &= \begin{bmatrix} w(0) & \cdots & w(M) \end{bmatrix} \begin{bmatrix} y(n) \\ \vdots \\ y(n-M) \end{bmatrix} = \mathbf{w}^* \mathbf{y}[n]\end{aligned}$$

Los pesos $w(i)$ se obtienen minimizando el error cuadrático medio (MSE)¹:

$$\begin{aligned}J(\mathbf{w}) &= \mathbb{E} [|x(n) - \mathbf{w}^* \mathbf{y}[n]|^2] \\ &= \mathbb{E} [|x(n)|^2] - \mathbb{E} [x(n) \mathbf{y}^*[n] \mathbf{w}] - \mathbb{E} [\mathbf{w}^* \mathbf{y}[n] x(n)^*] + \mathbb{E} [\mathbf{w}^* \mathbf{y}[n] \mathbf{y}^*[n] \mathbf{w}] \\ &= \sigma_x^2 - \mathbf{R}_{xy} \mathbf{w} - \mathbf{w}^* \mathbf{R}_{yx} + \mathbf{w}^* \mathbf{R}_y \mathbf{w}\end{aligned}$$

¹Fijense que, para simplificar notación en lo que sigue, estamos utilizando el vector $\mathbf{w} \in \mathbb{C}^{M+1 \times 1}$ con sus elementos conjugados para hacer la convolución. 

donde

$$\begin{aligned}\mathbf{R}_{xy} &= \begin{bmatrix} \mathbb{E}[x(n)\bar{y}(n)] & \cdots & \mathbb{E}[x(n)\bar{y}(n-M)] \end{bmatrix} \\ &= \begin{bmatrix} R_{xy}(0) & \cdots & R_{xy}(M) \end{bmatrix} = \mathbf{R}_{yx}^*\end{aligned}$$

$$\begin{aligned}\mathbf{R}_y &= \begin{bmatrix} \mathbb{E}[y(n)\bar{y}(n)] & \cdots & \mathbb{E}[y(n)\bar{y}(n-M)] \\ \vdots & & \vdots \\ \mathbb{E}[y(n-M)\bar{y}(n)] & \cdots & \mathbb{E}[y(n-M)\bar{y}(n-M)] \end{bmatrix} \\ &= \begin{bmatrix} R_y(0) & \cdots & R_y(M) \\ & \ddots & \\ R_y(-M) & \cdots & R_y(0) \end{bmatrix}\end{aligned}$$

Re-planteo del filtrado FIR

Sea $\mathbf{w} = \mathbf{w}_R + j\mathbf{w}_I$ la descomposición en parte real e imaginaria del vector \mathbf{w} . Luego, para minimizar J con respecto a \mathbf{w} calculamos el gradiente e igualamos a cero:

$$\nabla J = \begin{bmatrix} \frac{\partial J}{\partial w_R(0)} + j \frac{\partial J}{\partial w_I(0)} \\ \vdots \\ \frac{\partial J}{\partial w_R(M)} + j \frac{\partial J}{\partial w_I(M)} \end{bmatrix} = 2 (\mathbf{R}_y \mathbf{w} - \mathbf{R}_{xy}^*) = 0$$
$$\implies \mathbf{w}_{opt} = \mathbf{R}_y^{-1} \mathbf{R}_{xy}^*$$

Observen que esta solución coincide con el filtro del Wiener (filtro MMSE) que dedujimos utilizando el principio de ortogonalidad.

Algoritmo *steepest descent* para optimización de funciones multivariables

- Sea $f : \mathbb{C}^L \rightarrow \mathbb{R}$ y \bar{x} un mínimo local de f . Suponemos que $f(\cdot)$ es diferenciable en todo punto de $B(\bar{x}) \subset \mathbb{C}^L$, un vecindario de \bar{x} .
- Para hallar \bar{x} de forma numérica armamos una trayectoria que luego de n pasos resulta

$$x_{n+1} = x_n + p$$

donde p es la dirección de actualización². Indica hacia dónde ir a buscar la siguiente aproximación de \bar{x} .

- Se dice que p es una dirección de descenso de $f(\cdot)$ cuando

$$\nabla f^* p < 0.$$

²Referencia [??] desarrolla este tema en el contexto general de optimización de funciones.

Algoritmo *steepest descent* para optimización de funciones multivariantes

- Si p es una dirección de descenso de $f(\cdot)$, entonces para δ pequeño,

$$f(x_n + \delta p) < f(x_n)$$

- El descenso más rápido se obtiene cuando

$$p = -\frac{\nabla f}{\|\nabla f\|}$$

- Luego, el algoritmo de descenso por la mayor pendiente o de *steepest descent* es

$$x_{n+1} = x_n - \mu \nabla f$$

donde μ es un parámetro que define el tamaño del paso a dar.

Steepest descent para obtener el filtro de Wiener

Vimos que el filtro óptimo se obtiene minimizando $J : \mathbb{C}^{M+1} \rightarrow \mathbb{R}$,

$$J(\mathbf{w}) = \sigma_x^2 - \mathbf{R}_{xy}\mathbf{w} - \mathbf{w}^*\mathbf{R}_{yx} + \mathbf{w}^*\mathbf{R}_y\mathbf{w}.$$

Más aún, obtuvimos que $\nabla J = 2(\mathbf{R}_y\mathbf{w} - \mathbf{R}_{xy}^*)$. Luego, proponemos minimizar $J(\mathbf{w})$ utilizando el algoritmo de descenso por la pendiente más rápida.

Filtrado de Wiener utilizando Steepest-descent

Luego de obtener \mathbf{w}_n , la siguiente actualización es

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \mu (\mathbf{R}_y\mathbf{w}_n - \mathbf{R}_{xy}^*) \quad n = 0, 2, \dots$$

Notar que depende de \mathbf{R}_y y \mathbf{R}_{xy}^* , la mejora es que ahora es recursivo y converge al \mathbf{W}_{opt} .

Steepest descent para obtener el filtro de Wiener

El término de corrección luego del paso n es $\mathbf{R}_y \mathbf{w}_n - \mathbf{R}_{xy}^*$. Esto puede ser re-escrito del siguiente modo:

$$\begin{aligned}\mathbf{R}_y \mathbf{w}_n - \mathbf{R}_{xy}^* &= \mathbb{E} [\mathbf{y}[n] \mathbf{y}[n]^*] \mathbf{w}_n - \mathbb{E} [\mathbf{y}[n] x(n)^*] \\ &= \mathbb{E} [\mathbf{y}[n] (\mathbf{w}_n^* \mathbf{y}[n] - x(n))^*] \\ &= \mathbb{E} [\mathbf{y}[n] e(n)^*].\end{aligned}$$

Para salvar la rependencia con las autocorrelaciones se va a usar esto mas adelante.

Es decir, la corrección en el n -ésimo paso corresponde a la esperanza del producto entre el error de estimación $e(n) = \mathbf{w}_n^* \mathbf{y}[n] - x(n)$ y el vector de $M + 1$ observaciones $\mathbf{y}[n]$, ambos evaluados en el instante n . Volveremos sobre esta observación más adelante.

Estabilidad del algoritmo adaptivo

La expresión para el procesamiento de \mathbf{w}_n es de tipo recursivo. Entonces, es pertinente indagar la estabilidad de dicho procesamiento. En particular, la pregunta es

Bajo qué condiciones el algoritmo converge a la solución de Wiener?

$$\mathbf{w}_n \rightarrow \mathbf{w}_{opt}$$

Estabilidad del algoritmo adaptativo

Sea $\mathbf{c}_n = \mathbf{w}_n - \mathbf{w}_{opt}$. Luego,

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \mu (\mathbf{R}_y \mathbf{w}_n - \mathbf{R}_{xy}^*) \quad , \quad \mathbf{w}_{opt} = \mathbf{R}_y^{-1} \mathbf{R}_{xy}^*$$

$$\mathbf{w}_{n+1} - \mathbf{w}_{opt} = \mathbf{w}_n - \mathbf{w}_{opt} - \mu (\mathbf{R}_y (\mathbf{c}_n + \mathbf{w}_{opt}) - \mathbf{R}_{xy}^*)$$

$$\mathbf{c}_{n+1} = \mathbf{c}_n - \mu (\mathbf{R}_y (\mathbf{c}_n + \mathbf{R}_y^{-1} \mathbf{R}_{xy}^*) - \mathbf{R}_{xy}^*) .$$

$$\begin{aligned} \mathbf{c}_{n+1} &= \mathbf{c}_n - \mu \mathbf{R}_y \mathbf{c}_n + \mu \mathbf{R}_y \mathbf{R}_y^{-1} \mathbf{R}_{xy}^* - \mu \mathbf{R}_{xy}^* \\ &= \mathbf{c}_n - \mu \mathbf{R}_y \mathbf{c}_n + \mu \mathbf{R}_{xy}^* - \mu \mathbf{R}_{xy}^* \end{aligned}$$

Estabilidad

La dinámica del algoritmo adaptativo queda determinada por

$$\mathbf{c}_{n+1} = (\mathbf{I}_{M+1} - \mu \mathbf{R}_y) \mathbf{c}_n ,$$

donde \mathbf{I} es la matriz identidad. La estabilidad del algoritmo depende de μ y de los autovalores de \mathbf{R}_y

Estabilidad del algoritmo adaptivo

- La matriz \mathbf{R}_y es hermítica en forma general. Luego,

$$\mathbf{R}_y = \mathbf{P}\mathbf{\Lambda}\mathbf{P}^*,$$

\mathbf{P} es una matriz unitaria y $\mathbf{\Lambda} \succ 0$ es diagonal con entradas $\lambda_k > 0$.

- Sea $\mathbf{d}_n = \mathbf{P}^* \mathbf{c}_n$. Observe que \mathbf{P} es una matriz de rotación. Luego,

$$\begin{aligned}\mathbf{d}_{n+1} &= \mathbf{P}^* \mathbf{c}_{n+1} \\ &= \mathbf{P}^* (\mathbf{I}_{M+1} - \mu \mathbf{R}_y) \mathbf{c}_n \\ &= \mathbf{P}^* \mathbf{c}_n - \mu \mathbf{\Lambda} \mathbf{P}^* \mathbf{c}_n \\ &= (\mathbf{I}_{M+1} - \mu \mathbf{\Lambda}) \mathbf{d}_n\end{aligned}$$

Estabilidad del algoritmo adaptivo

$$\mathbf{d}_{n+1} = (\mathbf{I}_{M+1} - \mu\Lambda) \mathbf{d}_n$$

La matriz $(\mathbf{I}_{M+1} - \mu\Lambda)$ es diagonal. Luego, el k -ésimo elemento de \mathbf{d}_{n+1} resulta

$$d_{n+1,k} = (1 - \mu\lambda_k)d_{n,k}$$

Sea \mathbf{w}_0 la condición inicial a partir de la cual se comienza la adaptación. Luego, definimos el error inicial rotado, $\mathbf{d}_0 = \mathbf{P}^*(\mathbf{w}_0 - \mathbf{w}_{opt})$. Resolviendo la iteración para $d_{n+1,k}$ obtenemos,

$$d_{n+1,k} = (1 - \mu\lambda_k)^{n+1}d_{0,k},$$

Si $\forall k, |1 - \mu\lambda_k| < 1$, luego $\forall \mathbf{d}_0 \in \mathbb{C}^{M+1}$,

$$\mathbf{d}_n \rightarrow 0$$

Convergencia de la adaptación con *steepest-descent*

Si

$$0 < \mu < \frac{2}{\lambda_{max}} \quad (1)$$

entonces

$$\lim_{n \rightarrow \infty} \mathbf{w}_n = \mathbf{w}_{opt}$$

Si $\mathbf{d}_n \rightarrow 0$, la diferencia $\mathbf{c}_n = \mathbf{w}_n - \mathbf{w}_{opt}$ también converge a 0, es decir, $\mathbf{w}_n \rightarrow \mathbf{w}_{opt}$. Pero vimos que esto se cumple cuando $|1 - \mu\lambda_k| < 1$ para todo k . Como λ_k es positivo, si $\mu > 0$, la condición $1 - \mu\lambda_k < 1$ se cumple siempre. Por otro lado, $1 - \mu\lambda_k > -1$ es cierta si $\mu\lambda_k < 2$. Juntando ambas condiciones obtenemos (1).

Estabilidad del algoritmo adaptivo

Obtuvimos la condición bajo la cual el algoritmo adaptativo converge a la solución de Wiener. Ahora la pregunta es: con qué velocidad converge?

Retomamos el vector $\mathbf{d}_n = \mathbf{P}^* \mathbf{c}_n = \mathbf{P}^* (\mathbf{w}_n - \mathbf{w}_{opt})$. Sea $\mathbf{P} = [\mathbf{p}_1 \ \cdots \ \mathbf{p}_{M+1}]$. Entonces,

$$\begin{aligned}\mathbf{w}_n &= \mathbf{w}_{opt} + \mathbf{P} \mathbf{d}_n \quad , \quad \mathbf{P} = [\mathbf{p}_1 \ \cdots \ \mathbf{p}_{M+1}] \\ &= \mathbf{w}_{opt} + \sum_{k=1}^{M+1} \mathbf{p}_k d_{n,k} \\ &= \mathbf{w}_{opt} + \sum_{k=1}^{M+1} (1 - \mu \lambda_k)^n d_{0,k} \mathbf{p}_k\end{aligned}$$

La velocidad de convergencia hacia \mathbf{w}_{opt} depende de λ_{max} y λ_{min} . Obviamente, el término $(1 - \mu \lambda_{max})^n$ es el que alcanza el valor 0 más rápido y $(1 - \mu \lambda_{min})^n$ el más lento.

Estabilidad del algoritmo adaptivo

Ahora que vimos cómo converge el vector \mathbf{w}_n la siguiente pregunta es: cómo converge la función de costo $J(\mathbf{w}_n)$?

Retomamos la expresión para

$$J(\mathbf{w}_n) = \sigma_x^2 - \mathbf{R}_{xy}\mathbf{w}_n - \mathbf{w}_n^*\mathbf{R}_{yx} + \mathbf{w}_n^*\mathbf{R}_y\mathbf{w}_n.$$

Cuando alcanzamos el óptimo, $\mathbf{w} = \mathbf{w}_{opt}$, entonces

$$J_{opt} = J(\mathbf{w}_{opt}) = \sigma_x^2 - \mathbf{R}_{xy}\mathbf{R}_y^{-1}\mathbf{R}_{xy}^*.$$

Estabilidad del algoritmo adaptivo

Considerando que $\mathbf{w}_n = \mathbf{c}_n + \mathbf{w}_{opt}$ tenemos,

$$\begin{aligned} J(\mathbf{w}_n) &= J_{opt} + \mathbf{c}_n^* \mathbf{R}_y \mathbf{c}_n = J_{opt} + \mathbf{d}_n^* \Lambda \mathbf{d}_n \\ &= J_{opt} + \sum_{k=1}^{M+1} \lambda_k |d_{n,k}|^2 \\ &= J_{opt} + \sum_{k=1}^{M+1} \lambda_k |(1 - \mu \lambda_k)^n d_{0,k}|^2 \\ &= J_{opt} + \sum_{k=1}^{M+1} \lambda_k (1 - \mu \lambda_k)^{2n} |d_{0,k}|^2 \end{aligned}$$

El valor de μ controla la velocidad de convergencia. Cuando μ es grande, $J(\mathbf{w}_n)$ converge más rápido a J_{opt} .

Steepest-descent Recap

- La actualización del filtro es de acuerdo a

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \mu (\mathbf{R}_y \mathbf{w}_n - \mathbf{R}_{xy}^*) \quad n = 0, 2, \dots$$

- Si $0 < \mu < \frac{2}{\lambda_{max}}$, entonces

$$\lim_{n \rightarrow \infty} \mathbf{w}_n = \mathbf{w}_{opt}$$

siguiendo la dinámica

$$\mathbf{w}_n = \mathbf{w}_{opt} + \sum_{k=1}^{M+1} (1 - \mu \lambda_k)^n d_{0,k} \mathbf{p}_k$$

Steepest-descent Recap

La convergencia al filtro de Wiener depende de

- El error inicial rotado \mathbf{d}_0 o en forma equivalente, \mathbf{w}_0 y de los autovectores de \mathbf{R}_y
- El valor de μ
- Los autovalores de \mathbf{R}_y .

Complicación que no pudimos salvar con este esquema: necesitamos conocer \mathbf{R}_y y \mathbf{R}_{xy} .

- El algoritmo de *steepest descend* nos permite obtener la solución MMSE de modo recursivo. Es decir, resulta sólo una astucia numérica para obtener la solución.
- Pregunta: para qué se usa el procesamiento iterativo?

Filtrado Adaptativo

- El algoritmo de *steepest descend* nos permite obtener la solución MMSE de modo recursivo. Es decir, resulta sólo una astucia numérica para obtener la solución.
- Pregunta: para qué se usa el procesamiento iterativo?
- Al realizar cálculos en forma iterativa, nos podemos adaptar a condiciones cambiantes del medio.
- Es posible *identificar* relaciones dinámicas a partir de patrones conocidos.

Filtrado adaptativo:ejemplos

- Ecualización de canal de comunicaciones (supervisado o no-supervisado/ciego)
- Cancelador de ruido (medición de señal correlacionada)
- Deconvolución de señales sísmicas
- etc

Filtrado adaptativo

Un filtro adaptativo requiere definir:

- Estructura del filtro
- Criterio de performance
- Algoritmo de adaptación

Algoritmo LMS

Sea $e(n) = \mathbf{w}_n^* \mathbf{y}[n] - x(n)$ el error de estimación de $x(n)$ utilizando las observaciones en $\mathbf{y}[n]$. Al analizar el algoritmos de adaptación *steepest-descent*, vimos que el término de corrección es

$$\mathbf{R}_y \mathbf{w}_n - \mathbf{R}_{xy}^* = \mathbb{E} [\mathbf{y}[n] e(n)^*].$$

Luego,

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \mu \mathbb{E} [\mathbf{y}[n] e(n)^*].$$

Este algoritmo requiere conocer la estadística conjunta entre $\mathbf{y}[n]$ y $e(n)$. Esto no siempre es posible, o puede resultar muy complejo.

La idea del algoritmo LMS (*Least-Mean-Square*) es aproximar el operador $\mathbb{E}[\cdot]$ por el resultado de una realización del proceso, es decir

$$\mathbb{E} [\mathbf{y}[n] e(n)^*] \quad \text{pasa a ser} \quad \mathbf{y}[n] e(n)^*.$$

Adaptación de acuerdo al algoritmo *Least-Mean-Square*

Sea \mathbf{k}_n el filtro luego del n -ésimo paso de adaptación. En cada paso, tenemos que $\hat{x}(n) = \mathbf{k}_n^* \mathbf{y}[n]$. El algoritmo de adaptación es como sigue:

Algoritmo LMS

$$\begin{aligned}\mathbf{k}_{n+1} &= \mathbf{k}_n - \mu (\mathbf{y}[n]e(n)^*) \\ &= \mathbf{k}_n - \mu [\mathbf{y}[n] (\mathbf{k}_n^* \mathbf{y}[n] - x(n))^*]\end{aligned}$$

Algoritmo LMS

Algunas observaciones preliminares

- La dirección de descenso del LMS no es determinística. Este algoritmo pertenece a la clase de algoritmos de gradiente estocástico.
- La convergencia del algoritmo no está asegurada, si bien su performance es muy buena bajo ciertas condiciones
- La enorme ventaja (y razón de su popularidad) del algoritmo LMS es su simplicidad.

Convergencia del Algoritmo *Least-Mean-Square*

Como en el caso del algoritmo de *steepest-descent*, vamos a analizar si \mathbf{k}_n converge a medida que aumenta n . Para ello debemos determinar el criterio de convergencia. Sea el vector de diferencias $\varepsilon_n = \mathbf{k}_n - \mathbf{w}_{opt}$. El criterio de convergencia en el error cuadrático medio es:

$$\lim_{n \rightarrow \infty} \mathbb{E}[\|\varepsilon(n)\|^2] = \text{constante}$$

- Como el límite de $\varepsilon_n \neq 0$, entonces \mathbf{k}_n no converge a \mathbf{w}_{opt} .
- El valor del costo $J_{LMS}(\mathbf{k}_n)$ es superior al costo de Wiener,
 $J_{opt} = \sigma_x^2 - \mathbf{R}_{xy} \mathbf{R}_y^{-1} \mathbf{R}_{xy}^*$.

Convergencia LMS

Error LMS:

$$e(n) = x(n) - \overset{\text{Kn} = \text{eps}_n + W_{opt}}{\mathbf{k}_n^*} \mathbf{y}[n] = \underbrace{x(n) - \mathbf{w}_{opt}^* \mathbf{y}[n]}_{e_{opt}(n)} - \varepsilon_n^* \mathbf{y}[n]$$

$$\begin{aligned} J_{LMS}(n) &= \mathbb{E}[|e(n)|^2] = \mathbb{E}[|e_{opt}(n) - \varepsilon_n^* \mathbf{y}[n]|^2] \\ &= \mathbb{E}[|e_{opt}(n)|^2] + \mathbb{E}[|\varepsilon_n^* \mathbf{y}[n]|^2] \\ &= J_{opt} + \mathbb{E}[\text{tr}[\mathbf{y}[n]^* \varepsilon_n \varepsilon_n^* \mathbf{y}[n]]] \\ &= J_{opt} + \text{tr}[\mathbb{E}[\varepsilon_n \varepsilon_n^* \mathbf{y}[n] \mathbf{y}^*[n]]] \\ &= J_{opt} + \underbrace{\text{tr}[\mathbb{E}[\varepsilon_n \varepsilon_n^*] \mathbf{R}_y]}_{J_{ex}(n)} \end{aligned}$$

Se puede demostrar que

$$\lambda_{min} \leq \frac{J_{ex}(n)}{\mathbb{E}[||\varepsilon(n)||^2]} \leq \lambda_{max} \quad \forall n$$

Convergencia del Algoritmo

- En general, la convergencia del algoritmo LMS es *ruidosa*, ya que depende de una aproximación ruidosa del gradiente de la función costo J .
- El algoritmo converge a un valor mayor al costo de Wiener J_{opt} , pero la convergencia no tiene comportamiento exponencial como en el caso del algoritmo de *steepest-descent*.
- Para convergencia, necesitamos

$$0 < \mu < \frac{2}{\text{trace}(\mathbf{R}_y)}.$$

Cómo seleccionar μ ?

Cuando no se conoce \mathbf{R}_y o sus autovalores, es difícil verificar que se cumpla la condición de convergencia. Observemos que

$$\text{trace}(\mathbf{R}_y) = (M + 1)R_y(0) = (M + 1)\mathbb{E}[|y(n)|^2] \simeq \sum_{l=0}^M |y(n - l)|^2 = E_y,$$

donde E_y es la energía contenida en la señal observada durante la ventana de duración $M + 1$. Luego, una indicación para elegir μ puede ser

$$0 < \mu < \frac{2}{E_y}$$

Algoritmo normalizado: NLMS

Uno de los problemas del LMS es que su paso de actualización depende de la realización de la observación y del error de aproximación, $\mathbf{y}[n]$ y $e(n)$. Como la dirección del paso puede no ser de descenso, se trata de evitar actualizaciones demasiado grandes de una vez. Para ello, se propone resolver, en cada instante de tiempo n , el siguiente problema de optimización para obtener \mathbf{k}_{n+1} :

$$\min \|\mathbf{k}_{n+1} - \mathbf{k}_n\|^2 \quad \text{sujeto a} \quad \hat{x}(n) = \mathbf{k}_{n+1}^* \mathbf{y}[n]$$

Para resolver este problema, debemos recurrir a la teoría de optimización convexa y los multiplicadores de Lagrange.

Lagrange en 5 min

"~Resuelve un problema de optimización con restricciones"

Considere el siguiente problema de optimización convexa

Puede que el menor f_0 no este dentro del conjunto de factibilidad(restricciones) y entonces tendría que buscar el mínimo dentro del conjunto de factibilidad que no coincidiría con el mínimo sin restricciones.

func. objetivo/criterio (g:FCM)

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \leq 0 \quad i = 1 \cdots m \\ & h_i(x) = 0 \quad i = 1 \cdots p \end{aligned}$$

quiero minimizar f_0 pero dentro de un dominio. Comparar los resultados de x permitidos.

Un conjunto convexo es un conjunto, en el cual si yo uno dos puntos cuales quiera del conjunto, cuerda queda dentro del conjunto, ej: la pelota es un conjunto convexo, la cuerda queda dentro del conjunto

también debe ser convexo



donde $f_0(x) : \mathbb{R}^n \rightarrow \mathbb{R}$.

Un problema de optimización es convexo cuando $f_0(x)$ es una función convexa y las restricciones definen un conjunto convexo donde se encuentra la solución. Que el problema sea convexo me garantiza que si el problema es factible, tiene una única solución.

Lagrange en 5 min

Al problema de optimización convexa, le asociamos un Lagrangiano que es la función $L(x, \lambda, \alpha) : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \rightarrow \mathbb{R}$

$$\mathcal{L}(x, \lambda, \alpha) := f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) + \sum_{i=1}^p \alpha_i h_i(x)$$

Las variables λ_i, α_i son los multiplicadores de Lagrange.

Voy a minimizar el lagrangiano sin ningún tipo de restricción, es lo mismo que minimizar f_0 con las restricciones

Lagrange en 5min

Si x^* , λ^* , α^* son las soluciones óptimas, entonces satisfacen las Condiciones KKT (Karush–Kuhn–Tucker)

El gradiente del lagrangiano evaluado en los puntos óptimos (.) tienen que ser igual a cero.

$$\nabla_x \mathcal{L}(x^*, \lambda^*, \alpha^*) = \nabla_x f_0(x^*) + \sum_{i=1}^m \lambda_i^* \nabla_x f_i(x^*) + \sum_{i=1}^p \alpha_i^* \nabla_x h_i(x^*) = 0$$

Cond. de Kar.

$$\begin{aligned} f_i(x^*) &\leq 0, \quad i = 1, \dots, m & h_i(x^*) &= 0, \quad i = 1, \dots, p \\ \lambda_i^* &\geq 0, \quad \lambda_i^* f_i(x^*) &= 0, \quad i = 1, \dots, m \end{aligned}$$

va a ser un punto de la frontera cuando el moltiplicador de lagrange sea no nulo.

Deducción del NLMS

$$\min \| \mathbf{k}_{n+1} - \mathbf{k}_n \|^2 \quad / \quad \hat{x}(n) = \mathbf{k}_{n+1}^T \mathbf{y}(n)$$

Volvemos al planteo del NLMS. Para poder trabajar con funciones reales, consideramos la descomposición en parte real e imaginaria de las variables.

$$x = \begin{bmatrix} \mathbb{R}(\mathbf{k}_{n+1}) \\ \text{Im}(\mathbf{k}_{n+1}) \end{bmatrix} \in \mathbb{R}^{2(M+1)}$$

$$f_0(x) = \sum_{l=0}^M \left[(\mathbb{R}(\underline{k}_{n+1,l}) - \mathbb{R}(\underline{k}_{n,l}))^2 + (\text{Im}(\underline{k}_{n+1,l}) - \text{Im}(\underline{k}_{n,l}))^2 \right]$$

$\sum (\mathbf{k}_{n+1} - \mathbf{k}_n)^2$

$$h_1(x) = \mathbb{R}(\hat{x}(n)) - \sum_{l=0}^M [\mathbb{R}(k_{n+1,l})\mathbb{R}(y(n-l)) + \text{Im}(k_{n+1,l})\text{Im}(y(n-l))]$$

R: $\hat{x}(n) - \mathbf{k}_{n+1}^T \mathbf{y}(n) = 0$

$$h_2(x) = \text{Im}(\hat{x}(n)) - \sum_{l=0}^M [\mathbb{R}(k_{n+1,l})\text{Im}(y(n-l)) - \text{Im}(k_{n+1,l})\mathbb{R}(y(n-l))].$$

Deducción del NLMS

El Lagrangiano resulta entonces

$$\mathcal{L}(\mathbb{R}(k_{n+1,l}), \text{Im}(k_{n+1,l}), \alpha_1, \alpha_2) = f_0 + \alpha_1 h_1 + \alpha_2 h_2.$$

Para aplicar las condiciones KKT, tenemos que calcular ∇_x , es decir, calcular las derivadas parciales de $\mathcal{L}(\mathbb{R}(k_{n+1,l}), \text{Im}(k_{n+1,l}), \nu_1, \nu_2)$:

$$\partial_{\mathbb{R}(k_{n+1,l})} : 2(\mathbb{R}(k_{n+1,l}) - \mathbb{R}(k_{n,l})) - \alpha_1 \mathbb{R}(y(n-l)) - \alpha_2 \text{Im}(y(n-l))$$

$$\partial_{\text{Im}(k_{n+1,l})} : 2(\text{Im}(k_{n+1,l}) - \text{Im}(k_{n,l})) - \alpha_1 \text{Im}(y(n-l)) + \alpha_2 \mathbb{R}(y(n-l))$$

Observar que $\partial_{\alpha_1} = h_1 = 0$ y $\partial_{\alpha_2} = h_2 = 0$ por ser las dos restricciones del problema. Para compactar notación, definimos

$$\partial_{k_{n+1,l}} := \partial_{\mathbb{R}(k_{n+1,l})} + j\partial_{\text{Im}(k_{n+1,l})}$$

$$\alpha = \alpha_1 + j\alpha_2$$

Deducción del NLMS

Luego,

$$\begin{aligned}\partial_{k_{n+1,l}} \mathcal{L} &= 2(k_{n+1,l} - k_{n,l}) - \alpha y(n-l) = 0 \\ \implies k_{n+1,l} &= k_{n,l} + \frac{\alpha}{2} y(n-l)\end{aligned}$$

Aplicando la condición KKT sobre las restricciones resulta,

$$\sum_{l=0}^M k_{n+1,l}^* y(n-l) = \hat{x}(n)$$

Reemplazando la expresión para $k_{n+1,l}$ tenemos,

$$\begin{aligned}\sum_{l=0}^M k_{n,l}^* y(n-l) + \frac{\alpha^*}{2} y(n-l)^* y(n-l) &= \hat{x}(n) \\ \frac{\alpha^*}{2} &= \frac{1}{\sum_{l=0}^M y(n-l)^* y(n-l)} \left[\hat{x}(n) - \sum_{l=0}^M k_{n,l}^* y(n-l) \right]\end{aligned}$$

Finalmente

$$\begin{aligned}\mathbf{k}_{n+1} &= \mathbf{k}_n + \frac{\alpha}{2} \mathbf{y}[n] \\ &= \mathbf{k}_n + \frac{1}{\|\mathbf{y}[n]\|^2} [\hat{x}(n) - \mathbf{k}_n^* \mathbf{y}[n]]^* \mathbf{y}[n] \\ &= \mathbf{k}_n - \frac{e(n)^*}{\|\mathbf{y}[n]\|^2} \mathbf{y}[n]\end{aligned}$$

donde $e(n) = \mathbf{k}_n^* \mathbf{y}[n] - \hat{x}(n)$.

Algoritmo normalizado: NLMS

Actualización del ^{norm}NLMS

$$\mathbf{k}_{n+1} = \mathbf{k}_n - \frac{\mu_{norm}}{\|\mathbf{y}[n]\|^2} \mathbf{y}[n] e(n)^*$$

→ Se ajusta con la señal.

Actualización del LMS

$$\mathbf{k}_{n+1} = \mathbf{k}_n - \mu \mathbf{y}[n] e(n)^*$$

Algoritmo normalizado: NLMS

Observaciones:

- El algoritmo NLMS puede ser visto como el algoritmo LMS con ancho de paso dependiente del tiempo n .
- El valor de μ no tiene por qué ser el mismo en ambos casos.
- El parámetro μ para LMS tiene unidades de potencia⁻¹, pero en el caso del NLMS, es adimensional
- Que sucede si $\|y(n)\| \simeq 0$? Entonces, hay que considerar

$$\mathbf{k}_{n+1} = \mathbf{k}_n - \frac{\mu_{norm}}{\kappa + \|\mathbf{y}[n]\|^2} \mathbf{y}[n]e(n)^*$$

donde $\kappa > 0$ es un parámetro de regularización a seleccionar.

Un nuevo criterio ... o no tanto

Empezamos a tratar el tema de la estimación adaptiva como una aproximación al estimador óptimo de menor error cuadrático medio. Pero una pregunta latente sigue presente: qué sucede si no se tiene acceso a la estadística de los procesos?

Supongamos que contamos con las observaciones

$$y(-M), \dots, y(0), \dots, y(N).$$

A partir de estas observaciones y para cada instante $n, 0 \leq n \leq N$, deseamos estimar la señal $x(n)$ combinando en forma lineal las observaciones $y(n-M), \dots, y(n)$.

Estimador LS

Como antes,

$$\hat{x}(n) = \sum_{l=0}^M w_l y(n-l) = \mathbf{w}^* \mathbf{y}[n] = \mathbf{y}[n]^t \mathbf{w}$$

Estimador de cuadrados mínimos (*Least-Squares* (LS))

Obtener $w_l, l = 0, \dots, M$, tal que resuelva

$$\min_{w_l} \sum_0^N |e(n)|^2 \quad (2)$$

antes minimizaba la esperanza del error cuadrático medio, ahora, de hecho no estoy haciendo ningún tipo de estimación estadística. es todo determinístico.

donde $e(n) = x(n) - \hat{x}(n)$.

→ Estoy minimizando la suma de los cuadrados del error.

Estimador de cuadrados mínimos

Sean $\mathbf{e} \in \mathbb{C}^{N+1}$, $\mathbf{x} \in \mathbb{C}^{N+1}$, $\mathbf{Y} \in \mathbb{C}^{(N+1) \times (M+1)}$ definidos de acuerdo a:

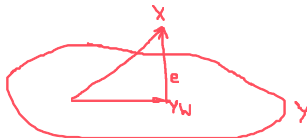
$$\begin{bmatrix} e(N) \\ e(N-1) \\ \vdots \\ e(0) \end{bmatrix} = \begin{bmatrix} x(N) \\ x(N-1) \\ \vdots \\ x(0) \end{bmatrix} - \begin{bmatrix} \mathbf{y}[N]^t \\ \mathbf{y}[N-1]^t \\ \vdots \\ \mathbf{y}[0]^t \end{bmatrix} \mathbf{w}$$
$$\mathbf{e} = \mathbf{x} - \mathbf{Y}\mathbf{w}$$

Estimador de cuadrados mínimos

Observaciones:

- El vector $\mathbf{Y}\mathbf{w} \in \mathcal{R}(\mathbf{Y})$, el rango o espacio columna de \mathbf{Y} .
- $\|\mathbf{e}\|^2$ mide la distancia entre \mathbf{x} y $\mathbf{Y}\mathbf{w}$.
- El criterio LS equivale a minimizar esta distancia
- El elemento de $\mathcal{R}(\mathbf{Y})$ que minimiza $\|\mathbf{e}\|^2$ es la proyección ortogonal de \mathbf{x} sobre $\mathcal{R}(\mathbf{Y})$.

Con estas observaciones, podemos reformular el criterio de cuadrados mínimos.



Estimador de cuadrados mínimos: reformulación

Estimador de cuadrados mínimos (reformulado)

El estimador de cuadrados mínimos que resuelve

$$\min_{w_l} \sum_0^N \left| x(n) - \sum_{l=0}^M w_l y(n-l) \right|^2$$

corresponde a la proyección ortogonal de \mathbf{x} sobre $\mathcal{R}(\mathbf{Y})$.

Estimador de cuadrados mínimos: reformulación

Utilizando la observación anterior, planteamos el principio de ortogonalidad según el cual el vector \mathbf{e} es ortogonal a $\mathcal{R}(\mathbf{Y})$

$$\mathbf{e}^* \mathbf{y} = 0 \quad \forall \mathbf{y} \in \mathcal{R}(\mathbf{Y})$$



Vemos que la condición de ortogonalidad establece que

$$\mathbf{e}^* \mathbf{Y} = \mathbf{0}_{1 \times (M+1)} \quad (3)$$

Comentario: Dadas las observaciones $y(N), y(N-1), \dots, y(-M)$, el espacio columna de \mathbf{Y} es un espacio vectorial determinístico. La proyección ortogonal sobre el mismo, depende de la realización de $y(n)$ observada.

Ecuaciones Normales

A partir del principio de ortogonalidad, construimos las ecuaciones normales que nos permiten obtener el vector \mathbf{w} para resolver el problema de estimación. En (3), reemplazamos la expresión para \mathbf{e} resultando:

$$\overbrace{[\mathbf{x} - \mathbf{Y}\mathbf{w}]^* \mathbf{Y}}^0 = 0$$



Desarrollando y tomando la transpuesta, obtenemos,

$$\left(\mathbf{x}^* \mathbf{Y} - \mathbf{w}^* \mathbf{Y}^* \mathbf{Y} \right)^T = 0^T \quad \Rightarrow \quad \underbrace{\mathbf{Y}^* \mathbf{Y}}_{\Phi_y(N)} \mathbf{w} = \underbrace{\mathbf{Y}^* \mathbf{x}}_{\rho_{xy}(N)}$$

Luego, el sistema de ecuaciones normales es

$$\Phi_y(N) \mathbf{w} = \rho_{xy}(N) \quad (4)$$

Ecuaciones Normales

Recordemos que $\mathbf{Y} \in \mathbb{C}^{(N+1) \times (M+1)}$. Más aún

$$\mathbf{Y} = \begin{bmatrix} \vdots \\ \mathbf{y}[N]^t \\ \vdots \\ \mathbf{y}[0]^t \end{bmatrix}$$

donde

$$\mathbf{y}[n] \in \mathbb{C}^{(M+1)}.$$

$$\Phi_y(N) = \mathbf{Y}^* \mathbf{Y}$$

$\hookrightarrow \text{Hermitiana} = \mathbf{Y}^* \mathbf{Y}$
 $\hookrightarrow \text{Simétrica Real} = \mathbf{Y}^T \mathbf{Y} \in \mathbb{R}^M, \forall \mathbf{Y} \in \mathbb{R}^M$

El elemento (i, j) de la matriz $\Phi_y(N)$ y el elemento i del vector $\rho_{xy}(N)$ son:

$$\Phi_{y(i,j)}(N) = \sum_{l=0}^N y(l-j+1)y^*(l-i+1) \quad 1 \leq i, j \leq M+1$$

← estimación de la autocorr. del proceso $\sim R_y$

$$\rho_{xy_i}(N) = \sum_{l=0}^N x(l)y^*(l-i+1)$$

← $\sim R_{xy}$

$$\phi_y(N) = W_{xy}(N)$$

$\sim R_y$ $\sim R_{xy}$
 Señal muestra el filtro de blancos

Parámetro de MSE

Ecuaciones Normales

si bien es una matriz semidefinida positiva y la mayoría de las veces va a ser definida positiva, es una matriz que se obtiene de datos experimentales y no suele ser agradable de invertir.
muchas veces voy a empezar a resolver problemas numericos para resolver la ecuacion 4.

Observaciones:

- La matriz $\Phi_y(N)$ resulta una aproximación a \mathbf{R}_y y el vector $\rho_{xy}(N)$ una aproximación a \mathbf{R}_{xy} .³
- Siguiendo un abuso de lenguaje, llamamos matriz de autocorrelación a Φ_y y vector de correlación cruzada a ρ_{xy} .
- Para resolver (4) se requiere invertir $\Phi_y(N)$.

³Es interesante comparar este sistema con el que se obtiene en el diseño del filtro de Wiener.

A dónde llegamos?

Recapitulando el camino que recorrimos hasta ahora, tenemos lo siguiente:

- El **filtro de Wiener** es la solución al problema de estimación con criterio de menor error cuadrático medio (MMSE).
- El algoritmo de ***Steepest Descent*** es un algoritmo iterativo que converge a la solución óptima.
- En ambos casos, se requiere conocer las estadísticas del proceso observado, $y(n)$ y de la señal a estimar, $x(n)$.
- El algoritmo **LMS (*Least-Mean Square*)** aborda este problema utilizando una dirección de actualización aleatoria.
- El algoritmo LMS no tiene convergencia garantizada en el sentido MMSE.
- Otra alternativa para encarar el desconocimiento de la estadística es utilizar el criterio de **Cuadrados Mínimos**.
- El estimador se obtiene resolviendo las ecuaciones normales
- El problema es que seguimos necesitando invertir una matriz.

Cuadrados mínimos recursivos

Retomamos las expresiones de $\Phi_y(N)$ y ρ_{xy}

Puedo calcular $\text{PHI}_y(N)$ con $\text{PHI}_y(N-1)$ mas el termino actual y así sucesivamente.

$$\begin{aligned}\Phi_y(N) &= \mathbf{Y}^* \mathbf{Y} = \sum_{n=0}^N \text{conj}(\mathbf{y}[n]) \mathbf{y}^t[n] \\ &= \underbrace{\sum_{n=0}^{N-1} \text{conj}(\mathbf{y}[n]) \mathbf{y}^t[n]}_{\Phi_y(N-1)} + \text{conj}(\mathbf{y}[N]) \mathbf{y}^t[N]\end{aligned}$$

$\in \mathbb{C}^{(N+1) \times (N+1)}$

$\mathbf{y} = \begin{bmatrix} \mathbf{y}[N]^T \\ \vdots \\ \mathbf{y}[0]^T \end{bmatrix}$ $\mathbf{x}(m) \in \mathbb{C}^{M \times 1}$

Posteriori

$$\begin{aligned}\rho_{xy}(N) &= \mathbf{Y}^* \mathbf{x} = \sum_{n=0}^N \text{conj}(\mathbf{y}[n]) x(n) \\ &= \underbrace{\sum_{n=0}^{N-1} \text{conj}(\mathbf{y}[n]) x(n)}_{\rho_{xy}(N-1)} + \text{conj}(\mathbf{y}[N]) x(N)\end{aligned}$$

Priori

se puede hacer lo mismo con la "correlacion cruzada".

Lema de la inversión de matrices

Matrix Inversion Lemma (Woodbury's Lemma)

Sean $\mathbf{A}, \mathbf{B} \in \mathbb{C}^{s \times s}$ y $\mathbf{D} \in \mathbb{C}^{r \times r}$ tres matrices cuadradas con determinante no nulo. Considere $\mathbf{C} \in \mathbb{C}^{s \times r}$ tal que

$$\mathbf{A} = \underbrace{\mathbf{B}}_{s \times s} + \underbrace{\mathbf{C}}_{s \times r} \underbrace{\mathbf{D}}_{r \times r} \underbrace{\mathbf{C}^*}_{r \times s}.$$

Luego,

$$\mathbf{A}^{-1} = \mathbf{B}^{-1} - \mathbf{B}^{-1} \mathbf{C} [\mathbf{D}^{-1} + \mathbf{C}^* \mathbf{B}^{-1} \mathbf{C}]^{-1} \mathbf{C}^* \mathbf{B}^{-1}.$$

Cuadrados mínimos recursivos (RLS)

Vimos que

$$\Phi_y(N) = \Phi_y(N-1) + \text{conj}(\mathbf{y}[N])\mathbf{y}^t[N]$$

$\overset{A}{\text{---}}$
 $\overset{B}{\text{---}}$
 $\overset{C D C^* = C C^*}{\text{---}}$

En la identidad de Woodbury, asociamos

$$s = M + 1, \quad r = 1$$

$$\mathbf{A} = \Phi_y(N), \quad \mathbf{B} = \Phi_y(N-1), \quad \mathbf{D} = 1, \quad \mathbf{C} = \text{conj}(\mathbf{y}[N])$$

Luego, obtenemos $\mathbf{A}^{-1} = \mathbf{B}^{-1} - \mathbf{B}^{-1} \mathbf{C} [\mathbf{D}^{-1} + \mathbf{C}^* \mathbf{B}^{-1} \mathbf{C}]^{-1} \mathbf{C}^* \mathbf{B}^{-1}$

$$\Phi_y(N)^{-1} = \Phi_y(N-1)^{-1}$$

$$- \underbrace{\Phi_y(N-1)^{-1}}_{\mathbf{B}^{-1}} \underbrace{\text{conj}(\mathbf{y}[N])}_{\mathbf{C}} \cdot \underbrace{\left[\overset{\mathbf{D}}{1} + \underbrace{\mathbf{y}^t[N] \Phi_y(N-1)^{-1} \text{conj}(\mathbf{y}[N])}_{\text{escalar}} \right]^{-1}}_{\text{escalar}} \cdot \underbrace{\mathbf{y}^t[N] \Phi_y(N-1)^{-1}}_{\mathbf{C}^* \mathbf{B}^{-1}}$$

Vemos que $[1 + \mathbf{y}^t[N] \Phi_y(N-1)^{-1} \text{conj}(\mathbf{y}[N])]$ es un escalar, luego

$$\Phi_y(N)^{-1} = \Phi_y(N-1)^{-1} - \frac{\underbrace{\Phi_y(N-1)^{-1}}_{\mathbf{B}^{-1}} \underbrace{\text{conj}(\mathbf{y}[N])}_{\mathbf{C}} \underbrace{\mathbf{y}^t[N] \Phi_y(N-1)^{-1}}_{\mathbf{C}^* \mathbf{B}^{-1}}}{\underbrace{1 + \mathbf{y}^t[N] \Phi_y(N-1)^{-1} \text{conj}(\mathbf{y}[N])}_{\text{escalar}}}$$

en vez de hacer una inversión de matrices voy a hacer multiplicación de matrices, menos problemas numericos

Cuadrados mínimos recursivos (RLS)

Al resolver las ecuaciones normales, obtenemos $\mathbf{w} = \mathbf{w}(N)$ ya que utilizamos las observaciones hasta el instante N . Sean

$$\mathbf{P}(N) = \Phi_y(N)^{-1} \quad \mathbf{k}(N) = \frac{\Phi_y(N-1)^{-1} \text{conj}(\mathbf{y}[N])}{1 + \mathbf{y}^t[N] \Phi_y(N-1)^{-1} \text{conj}(\mathbf{y}[N])}.$$

Luego,

$$\mathbf{P}(N) = \mathbf{P}(N-1) - \mathbf{k}(N) \mathbf{y}^t[N] \mathbf{P}(N-1)$$

Vimos que el estimador de mínimos cuadrados tiene la expresión:

$$\mathbf{w}(N) = \Phi_y(N)^{-1} \rho_{xy}(N) = \mathbf{P}(N) \rho_{xy}(N)$$

Introduciendo la recursión en $\mathbf{P}(N)$ y $\rho_{xy}(N)$ tenemos

$$\mathbf{w}(N) = \left\{ \mathbf{P}(N-1) - \mathbf{k}(N) \mathbf{y}^t[N] \mathbf{P}(N-1) \right\} \cdot \left\{ \rho_{xy}(N-1) + \text{conj}(\mathbf{y}[N]) x(N) \right\}$$

Cuadrados mínimos recursivos (RLS)

$$\mathbf{w}(N) = \{ \mathbf{P}(N-1) - \mathbf{k}(N)\mathbf{y}^t[N]\mathbf{P}(N-1) \} \{ \rho_{xy}(N-1) + \text{conj}(\mathbf{y}[N])x(N) \}$$

- $\frac{\mathbf{P}(N-1)\rho_{xy}(N-1) - \mathbf{k}(N)\mathbf{y}^t[N]\mathbf{P}(N-1)\rho_{xy}(N-1)}{\mathbf{w}(N-1) - \mathbf{k}(N)\mathbf{y}^t[N]\mathbf{w}(N-1)} =$

- $\{ \mathbf{P}(N-1) - \mathbf{k}(N)\mathbf{y}^t[N]\mathbf{P}(N-1) \} \text{conj}(\mathbf{y}[N])x(N)$

$$\begin{aligned} & \left\{ \mathbf{P}(N-1)\text{conj}(\mathbf{y}[N]) - \frac{\mathbf{P}(N-1)\text{conj}(\mathbf{y}[N])}{1 + \mathbf{y}^t[N]\mathbf{P}(N-1)\text{conj}(\mathbf{y}[N])} \mathbf{y}^t[N]\mathbf{P}(N-1)\text{conj}(\mathbf{y}[N]) \right\} x(N) = \\ & = \mathbf{P}(N-1)\text{conj}(\mathbf{y}[N]) \left\{ 1 - \frac{\mathbf{y}^t[N]\mathbf{P}(N-1)\text{conj}(\mathbf{y}[N])}{1 + \mathbf{y}^t[N]\mathbf{P}(N-1)\text{conj}(\mathbf{y}[N])} \right\} x(N) \\ & = \frac{\mathbf{P}(N-1)\text{conj}(\mathbf{y}[N])}{1 + \mathbf{y}^t[N]\mathbf{P}(N-1)\text{conj}(\mathbf{y}[N])} x(N) = \mathbf{k}(N)x(N) \end{aligned}$$

Cuadrados mínimos recursivos (RLS)

Reemplazando estas expresiones en $\mathbf{w}(N)$, obtenemos

para hacer esto lo unico que asumi es que mi proceso no está cambiando, es ESA

$$\begin{aligned}\mathbf{w}(N) &= \mathbf{w}(N-1) - \mathbf{k}(N)\mathbf{y}^t[N]\mathbf{w}(N-1) + \mathbf{k}(N)x(N) \\ &= \mathbf{w}(N-1) + \mathbf{k}(N) [x(N) - \mathbf{y}^t[N]\mathbf{w}(N-1)] \\ &= \mathbf{w}(N-1) + \mathbf{k}(N) [x(N) - \mathbf{w}(N-1)^t\mathbf{y}[N]] \\ &= \mathbf{w}(N-1) + \mathbf{k}(N)\epsilon(N)\end{aligned}$$

donde $\epsilon(N) = x(N) - \mathbf{w}(N-1)^t\mathbf{y}[N]$ es el error *a priori* en la estimación de $x(N)$. El vector $\mathbf{k}(N)$ es conocido como la ganancia del filtro.

Observación: El criterio LS minimiza el error $e(N) = x(N) - \mathbf{w}(N)^t\mathbf{y}[N]$ que representa el error *a posteriori*. En general, $\epsilon(N) \neq e(N)$.



RLS con factor de olvido

si mi proceso tiene sus variaciones, es decir si mi proceso no es perfectamente ESA, entonces de alguna forma tengo que olvidar que es lo que paso en el pasado sino en el presente, ponderamos... con un factor de olvido

En algunos casos, interesa minimizar una suma ponderada de cuadrados, es decir en cada instante n , buscamos minimizar la siguiente expresión:

$$\sum_{i=1}^n \lambda^{n-i} |e(i)|^2$$

donde $e(i) = x(i) - \mathbf{w}(i)^t \mathbf{y}[i]$ es el error *a posteriori* y $0 < \lambda \leq 1$. El valor de λ determina la memoria del esquema.

- Cuando $\lambda = 1$, obtenemos la solución LS
- A medida que $\lambda \rightarrow 0$ menos influencia tienen las observaciones pasadas en el valor de $\mathbf{w}(n)$ que obtenemos.

RLS con factor de olvido

Si definimos

$$\begin{aligned} e^{\lambda_n}(i) &= \sqrt{\lambda^{n-i}} e(i) \\ &= \sqrt{\lambda^{n-i}} (x(i) - \mathbf{w}(i)^t \mathbf{y}[i]) \\ &= x^{\lambda_n}(i) - \mathbf{w}(i)^t \mathbf{y}^{\lambda_n}[i] \end{aligned}$$

vemos que a cada instante n tenemos que resolver un problema LS sin peso para estimar $x^{\lambda_n}(i)$ a partir de las observaciones $\mathbf{y}^{\lambda_n}[i]$. Es decir, el mismo esquema que desarrollamos antes lo podemos utilizar para resolver este problema obteniendo el algoritmo que se detalla a continuación.

RLS con factor de olvido

- Inicializar el algoritmo

- ▶ $\mathbf{P}(0) = \delta^{-1} \mathbb{I}$
- ▶ $\mathbf{w}(0) = 0$.

- Para cada instante n , computar

- ▶ Actualización *a priori*

Ganancia: $\mathbf{k}(n) = \frac{\lambda^{-1} \mathbf{P}(n-1) \text{conj}(\mathbf{y}[n])}{1 + \lambda^{-1} \mathbf{y}[n]^t \mathbf{P}(n-1) \text{conj}(\mathbf{y}[n])}$

Error: $\epsilon(n) = x(n) - \mathbf{w}(n-1)^t \mathbf{y}[n]$

- ▶ Actualización *a posteriori*

Pesos: $\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{k}(n) \epsilon(n)$.

Correlación: $\mathbf{P}(n) = \lambda^{-1} \mathbf{P}(n-1) - \lambda^{-1} \mathbf{k}(n) \mathbf{y}[n]^t \mathbf{P}(n-1)$

Estimación: $\hat{x}(n) = \mathbf{w}(n)^t \mathbf{y}[n]$



Bibliografía

- ① *Discrete-Time Signal Processing*, Alan V. Oppenheim, Ronald W. Schaffer, and John R. Buck, Prentice-Hall Signal Processing Series.
- ② *Digital Signal Processing*, J Proakis, D. Manolakis, Prentice Hall
- ③ *Multirate Systems And Filter Banks*, P. P. Vaidyanathan, Prentice Hall.
- ④ *Introduction to Spectral Analysis*, P. Stoica, R. Moses, Prentice Hall.
- ⑤ *Linear Estimation*, Thomas Kailath, Ali H. Sayed, and Babak Hassibi Prentice Hall.
- ⑥ L.R. Rabiner, B. Gold, C. A. McGonegal, "An Approach to the Approximation Problem for Nonrecursive Digital Filters", IEEE Trans. Audio and Electroacoustics, vol. AU-18, June 1970.
- ⑦ Neuvo Y., D. Cheng-yu and S. Mitra , "Interpolated finite impulse response filters." IEEE Transaction on Acoustics Speech, and Signal Processing, Vol. ASSP-32, 1984, pp. 563-570.