Estimação de Parâmetros de Filtros Lineares

Guilherme de Alencar Barreto

gbarreto@ufc.br

Grupo de Aprendizado de Máquinas – GRAMA
Programa de Pós-Graduação em Engenharia de Teleinformática
Universidade Federal do Ceará – UFC
www.researchgate.net/profile/Guilherme_Barreto2/

Estimação de Parâmetros

Conteúdo da Apresentação

1. Modelos Autoregressivos (AR) e AR com Entradas Exógenas (ARX)
2. Estimação de Parâmetros pelo Método de Yule-Walker
3. Estimação de Parâmetros pelo Método dos Mínimos Quadrados
4. Estimação Recursiva (Filtros LMS e RLS)
5. Estimação Recursiva Robusta a Outliers (Filtros LMM e RLM)

Estimação de Parâmetros

Pré-Requisitos

1. Noções de Cálculo Diferencial
2. Noções de Álgebra Linear
3. Variáveis Aleatórias
4. Noções de Matlab/Octave/Scilab

Estimação de Parâmetros

Objetivo Geral da Aula

Definir conceitos relacionados à estimação dos parâmetros de modelos lineares, tal como os modelos autorregressivos (AR) e AR com entradas exgenas (ARX).

Referências Importantes

1. Charles W. Therrien (1992). *Discrete Random Signals and Statistical Signal Processing*, .
2. Sílvio A. Abrantes (2000). *Processamento Adaptativo de Sinais*, Fundação Calouste Gulbekian, Lisboa.
3. Paulo S. R. Diniz (2020). *Adaptive Filtering: Algorithms and Practical Implementation*, Springer, 5th edition.
4. Luis A. Aguirre (2015). *Introduo à Identificação de Sistemas*, Editora UFMG, 4a. edição.
5. A. Hyvärinen, J. Karhunen & E. Oja (2001). *Independent Component Analysis*, John Wiley & Sons.

Estimação de Parâmetros

Prolegômenos

- A estimação de parâmetros é uma importante etapa de modelagem de um sistema dinâmico (i.e., com memória) a partir de dados observados, seja o sistema natural ou artificial.

Etapa 1 - Coleta/tratamento/visualização dos dados.

Etapa 2 - Identificação do modelo mais adequado.

Etapa 3 - **Estimação dos parâmetros** do modelo.

Etapa 4 - Validação do modelo (análise dos resíduos).

Etapa 5 - Utilização do modelo para fins prático-teóricos.

Parte I

Estimação de Parâmetros

Seja o processo autoregressivo de ordem p, AR(p), descrito como

$$\begin{aligned} x(n) &= a_1 x(n-1) + a_2 x(n-2) + \cdots + a_p x(n-p) + v(n), \\ &= [a_1 \; a_2 \; \cdots \; a_p] \begin{bmatrix} x(n-1) \\ x(n-2) \\ \vdots \\ x(n-p) \end{bmatrix} + v(n), \qquad (1) \\ &= \mathbf{a}^T \mathbf{x}(n-1) + v(n), \qquad (2) \end{aligned}$$

em que $\{a_i\}_{i=1}^{p}$ sãos parâmetros do processo e $v(n)$ simboliza um processo de ruído branco gaussiano, de média nula e variância σ_v^2.

O processo AR(p) é fundamental em processamento de sinais, pois é usado como modelo de muitos sistemas dinâmicos lineares.

Pode-se mostrar que a função coeficiente de autocorrelação (FCAC) do modelo AR(p) pode ser escrita como

$$\rho(\tau) = a_1\rho(\tau-1) + a_2\rho(\tau-2) + \cdots + a_p\rho(\tau-p), \quad \tau > 0 \quad (3)$$

em que $\rho(\tau)$ é definida a partir da função de autocorrelação (FAC), $R_x(\tau)$, como

$$\rho(\tau) = \frac{R_x(\tau)}{\sigma_x^2} = \frac{E[x(n)x(n-\tau)]}{E[x^2]}, \quad (4)$$

para o caso em que $E[x(n)] = 0$.

- A Eq. (3) é chamada de equação de *Yule-Walker*.
- E se $E[x(n)] \neq 0$? Como fica a definição de $\rho(\tau)$?

- A equação de Yule-Walker nos fornece um procedimento simples para estimar os coeficientes a_i, $i = 1, \ldots, p$.
- Para isso, temos que substituir $\rho(\tau)$ por sua versão amostral $r(\tau)$, dada por

$$r(\tau) = \frac{\sum_{k=1}^{N-\tau} x(k)x(k+\tau)}{\sum_{k=1}^{N} x^2(k)}, \quad \tau \geq 0 \tag{5}$$

  em que $x(k)$ consiste na k-ésima amostra de $x(n)$.
- Assim, a equação de Yule-Walker passa a ser escrita como

$$r(\tau) = a_1 r(\tau-1) + a_2 r(\tau-2) + \cdots + a_p r(\tau-p), \quad \tau > 0. \tag{6}$$

Portanto, se fizermos $\tau = 1, 2, \ldots, p$ na Eq. (6) obtemos

$$\begin{aligned} r(1) &= a_1 + a_2 r(1) + \cdots + a_p r(p-1) \\ r(2) &= a_1 r(1) + a_2 + \cdots + a_p r(p-2) \\ &\vdots \quad\quad \vdots \\ r(p) &= a_1 r(p-1) + a_2 r(p-2) + \cdots + a_p, \end{aligned} \tag{7}$$

em que usamos as seguintes propriedades da FCAC:

- $r(\tau) = r(-\tau)$, e.g. $r(2) = r(-2)$.
- $r(0) = 1$ (por quê?)

Em forma matricial o sistema de equações mostrado na Eq. (7) pode ser escrito como

$$\mathbf{R}\mathbf{a} = \mathbf{r}, \tag{8}$$

$$\begin{bmatrix} 1 & r(1) & \cdots & r(p-1) \\ r(1) & 1 & \cdots & r(p-2) \\ \vdots & \vdots & \ddots & \vdots \\ r(p-1) & r(p-2) & \cdots & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} r(1) \\ r(2) \\ \vdots \\ r(p) \end{bmatrix}$$

em que

$$\dim(\mathbf{R}) = p \times p, \quad \dim(\mathbf{a}) = \dim(\mathbf{r}) = p \times 1. \tag{9}$$

Assim, para estimarmos o vetor de coeficientes $\mathbf{a}$ basta inverter a matriz $\mathbf{R}$, ou seja

$$\hat{\mathbf{a}} = \mathbf{R}^{-1}\mathbf{r}. \tag{10}$$

Observação 1

A principal vantagem do método de Yule-Walker está na sua simplicidade de uso.

Observação 2

Como saber se as estimativas pontuais dos parâmetros, computadas via método de Yule-Walker, são realmente boas?

Observação 3

A questão reside na definição do que vem a ser um **bom estimador**! Para isso precisamos definir um importante conceito: a **polarização** (ou viés) de um estimador.

Uma propriedade desejável de um estimador é que ele esteja "próximo", de alguma maneira, do verdadeiro valor do parâmetro desconhecido.

Definição Formal

Seja $\hat{\theta}$ uma estimativa pontual do parâmetro desconhecido θ. Dizemos que $\hat{\theta}$ é um estimador não-polarizado (ou não-viesado) do parâmetro θ se

$$E[\hat{\theta}] = \theta. \tag{11}$$

- Assim, o viés de um estimador é definido como $b = \theta - E[\hat{\theta}]$.
- Em outras palavras, $\hat{\theta}$ é um estimador não-polarizado de θ se, "na média", seus valores forem iguais a θ.

- Note que isto equivale a exigir que a média da distribuição amostral de $\hat{\theta}$ seja igual a θ.

- Seja X uma variável aleatória com média $\mu = E[X]$ e variância $\sigma^2 = Var[X]$.
- Seja $\{x_1, x_2, \ldots, x_N\}$ um comjunto de observações i.i.d de tamanho N extraída de X, i.e. $E[X_i] = \mu$ e $Var[X_i] = \sigma^2$.
- Mostre que $\bar{x}$ é um estimador não-polarizado de μ.

Solução: Pela definição de estimador não-polarizado, temos que

$$\begin{aligned} E[\bar{x}] &= E\left[\frac{x_1 + x_2 + \cdots + x_N}{N}\right] = \frac{E[x_1] + E[x_2] + \cdots + E[x_N]}{N} \\ &= \frac{\mu + \mu + \cdots + \mu}{N} = \frac{N\mu}{N} = \mu \end{aligned} \tag{12}$$

Cconclui-se que $\bar{x}$ é, de fato, um estimador não-polarizado de μ.

- Seja a expressão da FCAC na Eq. (5) e repetida abaixo.

$$r(\tau) = \frac{\sum_{k=1}^{N-\tau} x(k)x(k+\tau)}{\sum_{k=1}^{N} x^2(k)}, \tag{13}$$

para $\tau = 0, \pm 1, \pm 2, \ldots, \pm \tau_{max}$.

- Representando o sinal como um vetor-coluna com N componentes, ou seja,

$$\mathbf{x}_{[1:N]} = [x(1) \;\; x(2) \;\; \cdots \;\; x(N-1) \;\; x(N)]^T, \tag{14}$$

podemos calcular a expressão na Eq. (13) com operações vetoriais em ambientes do tipo Octave/Matlab:

$$r_b(\tau) = \frac{\mathbf{x}^T_{[1:N-\tau]}\mathbf{x}_{[\tau+1:N]}}{\mathbf{x}^T_{[1:N]}\mathbf{x}_{[1:N]}}. \tag{15}$$

- Vale lembrar que expressão da FCAC mostrada na Eq. (15) é um estimador viesado da FAC.
- Pode-se obter a versão não-viesada da FAC, faz-se a seguinte operação:

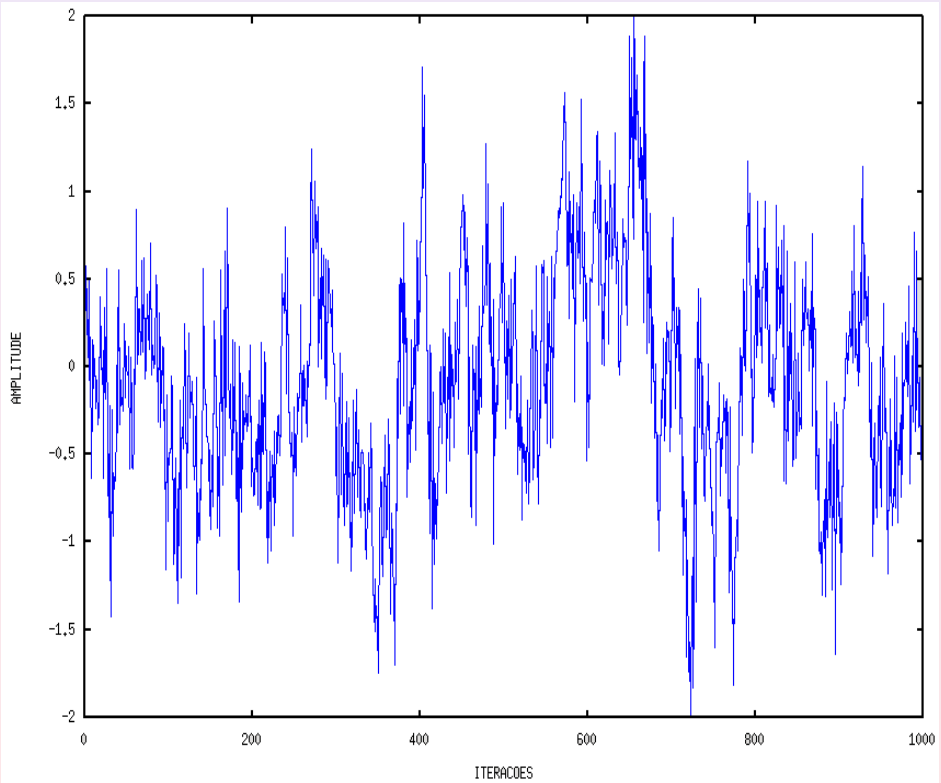
$$r_{ub}(\tau) = \frac{N}{N-\tau} \cdot r_b(\tau), \quad (16)$$

$$= \frac{N}{N-\tau} \cdot \frac{\mathbf{x}_{[1:N-\tau]}^T \mathbf{x}_{[\tau+1:N]}}{\mathbf{x}^T\mathbf{x}}, \quad (17)$$

$$= \frac{\frac{\mathbf{x}_{[1:N-\tau]}^T \mathbf{x}_{[\tau+1:N]}}{N-\tau}}{\frac{\mathbf{x}_{[1:N]}^T \mathbf{x}_{[1:N]}}{N}} = \frac{\hat{R}_x(\tau)}{\hat{\sigma}_x^2}, \quad (18)$$

em que $\hat{R}_x(\tau)$ e $\hat{\sigma}_x^2$ são, respectivamente, a FAC e a variância amostrais de $x(n)$.

- Considere o processo AR(2) simulado no Octave com coeficientes a_1=0,3 e a_2=0,6, e σ_v^2=0,15. Uma realização deste processo de comprimento $N = 1000$ é mostrada abaixo.



- Para este exemplo, a matriz $\mathbf{R}$ e o vetor $\mathbf{r}$ são dados por

$$\mathbf{R} = \begin{bmatrix} 1.0000 & 0.6521 \\ 0.6521 & 1.0000 \end{bmatrix} \quad \text{e} \quad \mathbf{r} = \begin{bmatrix} 0.6521 \\ 0.7715 \end{bmatrix}$$

- Assim, o vetor de coeficientes do modelo AR(2) é dado por

$$\begin{aligned} \hat{\mathbf{a}} &= \mathbf{R}^{-1}\mathbf{r} = \begin{bmatrix} 1.7398 & -1.1395 \\ -1.1395 & 1.7398 \end{bmatrix} \begin{bmatrix} 0.6521 \\ 0.7715 \end{bmatrix} \\ &= \begin{bmatrix} 0.2592 \\ 0.6024 \end{bmatrix}. \end{aligned} \tag{19}$$

- Note que as estimativas são bem próximos dos valores reais.
- Quanto maior N, melhor serão as estimativas obtidas.

- Uma outra técnica de estimação de parâmetros muito usada é conhecida como método dos **mínimos quadrados ordinários** (OLS, *ordinary least squares*).
- Vamos aplicar esta técnica para estimar os parâmetros de um processo AR(p). Para isso, vamos supor que temos uma realização de comprimento N, $\{x(1), x(2), x(3), \cdots, x(N)\}$.
- Pela expressão do *modelo* AR(p), que é a do *processo* AR(p) sem o ruído, chega-se ao seguinte sistema de equações:

$$\begin{aligned}
x(p+1) &= a_1 x(p) + a_2 x(p-1) + \cdots + a_p x(1), \\
x(p+2) &= a_1 x(p+1) + a_2 x(p) + \cdots + a_p x(2), \\
x(p+3) &= a_1 x(p+2) + a_2 x(p+1) + \cdots + a_p x(3), \qquad (20) \\
\vdots &= \quad \vdots \qquad\qquad \vdots \\
x(N) &= a_1 x(N-1) + a_2 x(N-2) + \cdots + a_p x(N-p),
\end{aligned}$$

- Podemos escrever o sistema de equações em (20) em forma matricial, $\mathbf{p} = \mathbf{X}\mathbf{a}$, se fizermos as seguintes definições:

$$\mathbf{X} = \begin{bmatrix} x(p) & x(p-1) & \cdots & x(1) \\ x(p+1) & x(p) & \cdots & x(2) \\ \vdots & \vdots & \ddots & \vdots \\ x(N-1) & x(N-2) & \cdots & x(N-p) \end{bmatrix}_{(N-p)\times p}$$

e

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix}_{p\times 1} \quad \text{e} \quad \mathbf{p} = \begin{bmatrix} x(p+1) \\ x(p+2) \\ \vdots \\ x(N) \end{bmatrix}_{(N-p)\times 1}$$

- Note que não podemos inverter a matriz $\mathbf{X}$ para encontrar $\mathbf{a}$ pois ela não é quadrada. Em geral, temos $N \gg p$.
- Matematicamente, a estratégia a ser usada consiste em encontrar o erro gerado por $\hat{\mathbf{a}}$ ao determinar $\mathbf{p}$ quando usado no lugar do "verdadeiro" vetor de coeficientes $\mathbf{a}$.
- Este erro é definido como

$$\mathbf{e} = \mathbf{p} - \hat{\mathbf{p}} = \mathbf{p} - \mathbf{X}\hat{\mathbf{a}} \tag{21}$$

- O ideal é que esse erro seja mínimo, pois assim $\hat{\mathbf{a}}$ deverá ser bem "semelhante" ao verdadeiro vetor $\mathbf{a}$.

- Podemos definir uma função-custo que quando minimizada, resulte na estimativa $\hat{\mathbf{a}}$ que produza a menor norma do vetor de erros quadráticos:

$$J_{OLS}(\mathbf{a}) = \frac{1}{2}\|\mathbf{e}\|^2 = \frac{1}{2}(\mathbf{p} - \mathbf{X}\mathbf{a})^T(\mathbf{p} - \mathbf{X}\mathbf{a}). \qquad (22)$$

- Para minimizar $J(\mathbf{a})$, calculamos sua derivada em relação a $\hat{\mathbf{a}}$, igualamos a zero, e resolvemos a equação resultante.
- Pode-se mostrar (fica como exercício) que a estimativa $\hat{\mathbf{a}}$ resultante é dada por:

$$\hat{\mathbf{a}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{p}. \qquad (23)$$

- O estimador obtido pela minimização da Eq. (22) é chamado **estimador dos mínimos quadrados ordinários**.

Curiosidades sobre o Método dos Mínimos Quadrados

- Foi proposto em 1795 por **Carl Friedrich Gauss** (30/Abr/1777 - 23/Set/1855).

- Gauss aplicou o método no cálculo de órbitas de planetas e cometas a partir de medidas obtidas por telescópios.
- **Adrien Marie Legendre** (1752-1833) desenvolveu o mesmo método independentemente e o publicou primeiro em 1806.

Regularização de Tikhonov

- Muitas vezes a matriz $\mathbf{X}^T\mathbf{X}$ é singular, ou seja, seu posto menor que p.
- Isso certamente causará problemas numéricos durante a inversão desta matriz.
- Para evitar tais problemas sugere-se reescrever a Eq. (23) como

$$\hat{\mathbf{a}} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}. \tag{24}$$

  em que $0 \leq \lambda \ll 1$ é uma constante de valor bem pequeno e $\mathbf{I}$ é a matriz identidade de dimensão $p \times p$.
- A função custo que leva à Eq. (24) dada por

$$J_{OLS}^{reg}(\mathbf{a}) = \frac{1}{2}\|\mathbf{e}\|^2 + \lambda\|\mathbf{a}\|^2. \tag{25}$$

Regularização de Thikonov

- A Eq. (25) busca um vetor de coeficientes $\mathbf{a}$ que minimize a norma quadrática do vetor de erro $\mathbf{e}$ e que ao mesmo tempo tenha norma mínima.
- Essa mesma função custo pode ser interpretada como o lagrangiano do seguinte problema de otimização com restrições de desigualdade:

$$\begin{aligned} \text{Minimizar} \quad & J_{OLS}(\mathbf{a}) = \frac{1}{2}\|\mathbf{e}\|^2 \\ \text{sujeito a} \quad & \|\mathbf{a}\|^2 \leq \frac{1}{\lambda} \end{aligned} \tag{26}$$

- Usando a mesma realização de um processo AR(2) usada no exemplo do método de Yule-Walker, definimos a matriz e os vetores necessários ao método MQ:

$$\mathbf{X} = \begin{bmatrix} x(2) & x(1) \\ x(3) & x(2) \\ \vdots & \vdots \\ x(999) & x(998) \end{bmatrix}, \tag{27}$$

e

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \quad \text{e} \quad \mathbf{p} = \begin{bmatrix} x(3) \\ x(4) \\ \vdots \\ x(1000) \end{bmatrix} \tag{28}$$

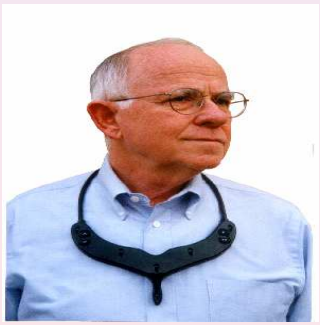
- Aplicando a fórmula da Eq. (23), chegamos ao seguinte resultado:

$$\hat{\mathbf{a}} = \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \end{bmatrix} = \begin{bmatrix} 0.2724 \\ 0.6017 \end{bmatrix} \tag{29}$$

- Note que as estimativas MQ também são bem próximos dos valores reais.
- Em virtude de suas melhores propriedades, tais como polarização, consistência e convergência, o uso do estimador MQ é preferível em relação ao estimador de Yule-Walker.

- Os métodos de estimação de parâmetros de modelos AR descritos anteriormente (MoM e MQ) são métodos **OFF-LINE**.
- Ou seja, dependem da construção de matrizes e vetores usando todas as amostras do sinal.
- Em muitas aplicações é necessária a estimação **ON-LINE** dos parâmetros do modelo, tais como
  1. Equalização adaptativa de canais de comunicação
  2. Cancelamento de eco na linha telefônica
  3. Supressão de ruído em cabines de avião
  4. Identificação adaptativa de sistemas dinâmicos
  5. Cancelamento de interferências em instrumentação médica

- Proposto em 1960 por **Dr. Bernard Widrow** (1929 -) e seu primeiro aluno de PhD Marcian "Ted" Hoff, Jr. (1937 -)[1].
- Ted Hoff é considerado o "inventor" do microprocessador (1a. patente), entrando na Intel Corporation em 1967 como o empregado número 12.
- Lá, ele projetou o primeiro chip de processador (1968), que chegou ao mercado como o Intel 4004 (1971)[2].



[1] B. Widrow & M.E. Hoff, Jr., "Adaptive Switching Circuits," IRE WESCON Convention Record, 4:96-104, August 1960.

[2] More at `www.thocp.net/biographies/hoff_ted.html`

- O objetivo da estimação recursiva é o mesmo da estimação off-line; ou seja, obter o vetor de parâmetros ótimo ($\mathbf{a}^* \in \Re^p$) que minimiza o erro quadrático médio.
- Na estimação recursiva o vetor de parâmetros, $\mathbf{a}$, é modificado (atualizado) a cada nova ocorrência de um novo vetor de entrada $\mathbf{x}$.
- Sejam $\mathbf{x}(n)$ e $\mathbf{a}(n)$, respectivamente, o vetor de entrada e o vetor de parâmetro no instante n.
- Logo, o *erro de estimação* no instante n é definido como

$$e(n) = x(n) - \hat{x}(n) = x(n) - \mathbf{a}^T(n)\mathbf{x}(n), \qquad (30)$$

em que $x(n)$ é a saída observada no instante n e $\hat{x}(n) = \mathbf{a}^T(n)\mathbf{x}(n)$ é a saída predita correspondente.

- Uma das técnicas mais simples e eficazes de estimação recursiva de parâmetros é o **Método do Gradiente**, cuja a expressão recursiva é dada por

$$\mathbf{a}(n+1) = \mathbf{a}(n) - \mu\nabla(n), \tag{31}$$

em que a constante $0 < \mu \ll 1$ é chamada de *passo de adaptação* e $\nabla(n)$ é o gradiente da função-custo.

- Visto que a função-custo de interesse é o erro quadrático médio, $J(n) = E\left[\frac{1}{2}e^2(n)\right]$, então seu gradiente é dado por

$$\begin{aligned}\nabla(n) &= \frac{\partial J(n)}{\partial \mathbf{a}(n)} = \frac{1}{2}\frac{\partial E[e^2(n)]}{\partial \mathbf{a}(n)} = \frac{1}{2}E\left[\frac{\partial e^2(n)}{\partial \mathbf{a}(n)}\right], \\ &= \frac{1}{2}E\left[2e(n)\frac{\partial e(n)}{\partial \mathbf{a}(n)}\right] = -E[e(n)\mathbf{x}(n)]. \end{aligned} \tag{32}$$

- Assim, a Eq. (31) pode ser reescrita como

$$\mathbf{a}(n+1) = \mathbf{a}(n) + \mu E[e(n)\mathbf{x}(n)]. \tag{33}$$

- Note que à medida que $n \to \infty$, $\mathbf{a}(n) \to \mathbf{a}^*$.
- Nesta situação, o gradiente $\nabla(n) \to 0$.
- Assim, pode-se concluir a partir da Eq. (33) que, quando $\mathbf{a}(n+1) = \mathbf{a}(n) = \mathbf{a}^*$, o erro $e(n)$ e os sinais de entrada não são correlacionados:

$$E[e(n)\mathbf{x}(n)] = 0 \quad (\text{se } \mathbf{a}(n+1) = \mathbf{a}(n) = \mathbf{a}^*) \tag{34}$$

- Esta conclusão é conhecida como *Princípio da Ortogonalidade* em estimação recursiva.

- Na prática, usam-se estimativas do gradiente, $\hat{\nabla}(n)$, em vez do verdadeiro gradiente $\hat{\nabla}(n)$.
- Isto ocorre porque não é possível calcular valores médios com base num único instante de tempo n.
- Para isto ser possível, teríamos que conhecer as estatísticas de conjunto (*ensemble means*) do processo.
- Assim, a Eq. (31) passa a ser escrita como

$$\mathbf{a}(n+1) = \mathbf{a}(n) - \mu\hat{\nabla}(n). \tag{35}$$

- Sendo $e(n) = x(n) - \mathbf{a}^T(n)\mathbf{x}(n)$, a estimativa instantânea do gradiente que passa a ser usada é dada por

$$\hat{\nabla}(n) = \frac{1}{2}\frac{\partial e^2(n)}{\partial \mathbf{a}(n)} = -e(n)\mathbf{x}(n). \quad (36)$$

  que é uma simplificação da Eq. (32).
- Assim, o algoritmo LMS (*least mean squares*) resume-se, portanto, à seguinte equação:

$$\mathbf{a}(n+1) = \mathbf{a}(n) + \mu e(n)\mathbf{x}(n). \quad (37)$$

Resumo do Algoritmo LMS

- Função custo: Erro Quadrático Instantâneo

$$J_{LMS}(n) = \frac{1}{2}e^2(n) = \frac{1}{2}(x(n) - \hat{x}(n))^2, \quad (38)$$

$$= \frac{1}{2}(x(n) - \mathbf{a}^T(n)\mathbf{x}(n))^2, \quad (39)$$

tal que $\mathbf{a}(n)$ é a estimativa atual do vetor de parâmetros.

- A regra de ajuste recursivo de $\mathbf{a}(n)$ é então dada por

$$\mathbf{a}(n+1) = \mathbf{a}(n) - \mu\frac{\partial J(n)}{\partial \mathbf{a}(n)}, \quad (40)$$

$$= \mathbf{a}(n) + \mu e(n)\mathbf{x}(n), \quad (41)$$

$$= \mathbf{a}(n) + \mu(x(n) - \hat{x}(n))\mathbf{x}(n), \quad (42)$$

$$= \mathbf{a}(n) + \mu(x(n) - \mathbf{a}^T(n)\mathbf{x}(n))\mathbf{x}(n), \quad (43)$$

tal que $0 < \mu < 1$ é o passo de aprendizagem.

- Pode-se mostrar que $\hat{\nabla}(n)$ e $\mathbf{a}(n+1)$ são estimativas não-polarizadas (não-viesadas) do gradiente $\nabla(n)$ e do vetor de parâmetros ótimo $\mathbf{a}^*$.
- Mantendo $\mathbf{a}(n) = \mathbf{a}$ (constante) e tomando o valor esperado de ambos os lados da Eq. (36), temos que

$$\begin{aligned} E[\hat{\nabla}(n)] &= -E[e(n)\mathbf{x}(n)] \\ &= -E[(x(n) - \mathbf{a}^T(n)\mathbf{x}(n))\mathbf{x}(n)] \\ &= E[\mathbf{x}(n)\mathbf{x}(n)^T\mathbf{a}] - E[x(n)\mathbf{x}(n)] \end{aligned} \quad (44)$$

- Supondo que $\mathbf{a}(n)$ e $\mathbf{c}$ são independentes, temos que

$$E[\mathbf{x}(n)\mathbf{x}(n)^T\mathbf{a}] = E[\mathbf{x}(n)\mathbf{x}(n)^T]E[\mathbf{a}] = E[\mathbf{x}(n)\mathbf{x}(n)^T]\mathbf{a} \tag{45}$$

- Daí, concluímos que

$$\begin{aligned} E[\hat{\nabla}(n)] &= E[\mathbf{x}(n)\mathbf{x}(n)^T]\mathbf{a} - E[x(n)\mathbf{x}(n)] \\ &= \mathbf{R}\mathbf{a} - \mathbf{p} = \nabla(n) \end{aligned} \tag{46}$$

- Portanto, temos que $\hat{\nabla}(n)$ é uma estimativa não-polarizada de $\nabla(n)$.

Exercício Proposto: Mostrar que $\mathbf{a}(n+1)$ é também uma estimativa não-polarizada de $\mathbf{a}^*$.

Otimalidade e Robustez do Algoritmo LMS

- O algoritmo LMS é usualmente considerado uma aproximação estocástica do problema de estimação de mínimos quadrados.
- Porém, Hassibi et al. (1996)[3] mostram que o algoritmo LMS provê a solução exata de um problema de minimização.
- Assim, o algoritmo LMS é um filtro *minimax* pois minimiza o máximo ganho de energia das perturbações nos erros de predição.
- O algoritmo LMS normalizado, por sua vez, minimiza o máximo ganho de energia devido às perturbações nos erros filtrados.
- Esta propriedade garante que se as perturbações são pequenas (em energia), então os erros de estimação serão tão pequenos quanto possível (em energia), não importa o que sejam tais perturbações.

[3] B. Hassibi, A. Sayed & T. Kailath (1996). "H_∞ optimality of the LMS algorithm. *IEEE Transactions on Signal Processing*, vol. 44, no. 2, pp. 267–280.

(1) Algoritmo LMS Normalizado

$$\mathbf{a}(n+1) = \mathbf{a}(n) + \frac{\mu}{\alpha + \|\mathbf{x}(n)\|^2} e(n)\mathbf{x}(n), \qquad (47)$$

em que $\|\cdot\|$ é a norma euclidiana de um vetor e $\alpha > 0$ é uma cosntante usada para evitar divisão por zero.

- O termo $\|\mathbf{x}(n)\|^2$ pode ser calculado recursivamente como

$$\|\mathbf{x}(n)\|^2 = x^2(n) + \|\mathbf{x}(n-1)\|^2 - x^2(n-p), \qquad (48)$$

lembrando que o vetor $\mathbf{x}(n)$ é definido como

$$\mathbf{x}(n) = [x(n) \;\; x(n-1) \;\; \cdots \;\; x(n-p)]^T \qquad (49)$$

(2) Algoritmo do Sinal

$$\mathbf{a}(n+1) = \mathbf{a}(n) + \mu \text{sign}[e(n)]\mathbf{x}(n), \tag{50}$$

em que a função sinal é definida como

$$\text{sign}[e(n)] = \begin{cases} +1 & x \geq 0 \\ -1 & x < 0 \end{cases} \tag{51}$$

- Função-custo: $J_{LMS}^{sign}(n) = |e(n)| = |x(n) - \mathbf{a}^T(n)\mathbf{x}(n)|$
- A principal vantagem deste algoritmo em relação ao LMS original é a sua simplicidade.
- No entanto, o seu tempo de convergência é menor que o do LMS original.

(3) Algoritmo Leaky LMS

$$\mathbf{a}(n+1) = (1-\lambda)\mathbf{a}(n) + \mu e(n)\mathbf{x}(n), \qquad (52)$$

em que $0 < \lambda < 1$ é chamado de parâmetro de fuga (*leaky*) ou de decaimento (*decay*).

- O algoritmo leaky LMS é popularmente conhecido na área de redes neurais artificiais como regularização por decaimento dos pesos[4] (*weight decay regularization*).
- No caso em que o vetor $\mathbf{a}(n)$ não é atualizado (i.e., $e(n) = 0$), seu valor será diminuído de uma fração $-\lambda\mathbf{a}(n)$.
- Isso faz com que o vetor não assuma valores elevados, mantendo a sua norma pequena de modo equivalente à regularização L_2 (Tikhonov).

[4] Anders Krogh & John A Hertz (1992). "A simple weight decay can improve generalization". In: Advances in Neural Information Processing Systems, pp. 950-957.

(3) Algoritmo Median LMS (MLMS)

$$\mathbf{a}(n+1) = \mathbf{a}(n) + \mu\Delta\mathbf{a}(n), \tag{53}$$

tal que

$$\Delta\mathbf{a}(n) = \text{MED}_m[e(n)\mathbf{x}(n), \ldots, e(n-m+1)\mathbf{x}(n-m+1)], \tag{54}$$

sendo $m \geq 1$ o tamanho da janela deslizante e MED$(\cdot)$ o operador mediana.

- Quando $m = 1$, o algoritmo MLMS reduz-se ao algoritmo LMS padrão.
- O algoritmo MLMS tende a ser mais robusto a outliers que o LMS e as variantes anteriores, uma vez que a mediana é uma estatística robusta.

- Uma outra técnica de estimação recursiva muito importante em identificação de sistemas é conhecida como *método dos mínimos quadrados recursivos* (RLS)[5].
- Esta técnica baseia-se na minimização da seguinte função-custo:

$$J_{RLS}(n) = \sum_{j=0}^{n} \alpha^{n-j} e^2(j) = \sum_{j=0}^{n} \alpha^{n-j} \left[d(j) - \mathbf{a}^T(j)\mathbf{x}(j)\right]^2, \tag{55}$$

  em que $0 \leq \alpha \leq 1$ é chamado de fator de esquecimento.
- Esta função-custo é minimizada a cada iteração e o número de parcelas do somatório cresce em função de n.

[5] Simon Haykin, *Adaptive Filter Theory*, Prentice Hall, 2002

- Uma equação recursiva do método RLS pode ser obtida[6], também conhecida como LMS-Newton[7], sendo dada por

$$\mathbf{a}(n+1) = \mathbf{a}(n) + \hat{\mathbf{R}}^{-1}(n)\mathbf{x}(n)e(n), \tag{56}$$

em que $\hat{\mathbf{R}}(n)$ é a estimativa atual da matriz de correlação, dada por

$$\hat{\mathbf{R}}(n) = \sum_{j=0}^{n} \alpha^{n-j}\mathbf{x}(j)\mathbf{x}^T(j), \tag{57}$$

$$= \alpha\hat{\mathbf{R}}(n-1) + \mathbf{x}(n)\mathbf{x}^T(n) \tag{58}$$

com $e(n) = x(n) - \hat{x}(n) = x(n) - \mathbf{a}^T(n)\mathbf{x}(n)$.

- Se $\hat{\mathbf{R}}(n) = \frac{1}{\mu}\mathbf{I}$ (i.e. constante e diagonal), então RLS = LMS.

[6] https://en.wikipedia.org/wiki/Recursive_least_squares_filter

[7] B. Widrow & M. Kamenetsky (2003). "Statistical efficiency of adaptive algorithms", Neural Networks, 16(5-6):735–744.

- A principal desvantagem do algoritmo RLS na forma da Eq. (56) é o grande número de cálculos exigidos.
- A cada iteração é preciso, por exemplo, calcular a matriz $\hat{\mathbf{R}}(n)$, armazená-la e invertê-la.
- Para uma matriz de dimensão $N \times N$, são necessários N^3 operações em cada iteração, só para a inversão, em vez das cerca de $2N$ operações requeridas pelo algoritmo LMS.
- Por exemplo, se $N = 50$ a diferença é de 100 para 125.000 operações em cada instante.
- Como veremos a seguir, felizmente, é possível reduzir o número de cálculos exigidos pelo algoritmo RLS.

- Um dos nosso maiores problemas está em inverter a matriz $\hat{\mathbf{R}}(n)$ a cada iteração n.
- Para evitar que a inversão seja feita a cada iteração, vamos utilizar uma relação matricial conhecida como **lema de inversão de matrizes** ou também como Identidade de Woodbury.

Lema de Inversão de Matrizes (Identidade de Woodbury)

$$(\mathbf{A}+\mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}\left(\mathbf{DA}^{-1}\mathbf{B}+\mathbf{C}^{-1}\right)^{-1}\mathbf{DA}^{-1}$$

- Agora, vamos fazer as seguintes associações:

$$\mathbf{A} = \alpha\hat{\mathbf{R}}(n-1), \quad \mathbf{B} = \mathbf{x}(n), \quad \mathbf{C} = 1 \text{ e } \mathbf{D} = \mathbf{a}^T(n),$$

tal que tenhamos $\hat{\mathbf{R}}(n) = \mathbf{A}+\mathbf{BCD}$.

- Assim, usando as associações feitas no slide anterior e escrevendo $\mathbf{P}(n) = \hat{\mathbf{R}}^{-1}(n)$, chegamos a

$$\mathbf{P}(n+1) = \frac{1}{\alpha}\left[\mathbf{P}(n) - \frac{\mathbf{P}(n)\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{P}(n)}{\alpha + \mathbf{x}^T(n)\mathbf{P}(n)\mathbf{x}(n)}\right],$$

  em que $\mathbf{P}(0) = \sigma\mathbf{I}$, σ sendo uma constante positiva de valor elevado e $\mathbf{I}$ é a matriz identidade de dimensão adequada.
- A expressão acima é conhecida como **Equação de Riccati** do algoritmo RLS.
- A subtração exigida na equação de Riccati pode ser fonte de problemas numéricos.
- Os problemas surgem se os diferentes cálculos forem feitos em precisão finita, com a introdução de erros de arrendondamento.

- A diferença entre as duas matrizes semidefinidas positivas que compõem a equação de Riccati pode dar origem a uma matriz $\mathbf{P}(n)$ não-simétrica, o que é incoerente com a teoria.
- Esta perda de simetria também pode tornar $\mathbf{P}(n)$ singular, ao ser atualizada a cada iteração.
- Problemas numéricos geralmente ocorrem após dezenas ou centenas de milhares de iterações.
- Estes problemas se traduzem em mudanças de sinal espúrias que, embora não causem $overflow$, conduzem a resultados muito diferentes dos que se atingem com precisão infinita.

- A solução para mitigar problemas numéricos consiste em formular a equação de Riccati em termos do vetor **ganho de Kalman**, definido como

$$\mathbf{k}(n) = \mathbf{P}(n)\mathbf{x}(n) = \hat{\mathbf{R}}^{-1}(n)\mathbf{x}(n), \tag{59}$$

que nada mais é do que o vetor de entrada $\mathbf{x}(n)$ transformado pela inversa de $\hat{\mathbf{R}}(n)$.

- Assim, podemos escrever a Eq. (56) como

$$\mathbf{a}(n+1) = \mathbf{a}(n) + \mathbf{k}(n)e(n). \tag{60}$$

Resumo do Algoritmo RLS

- Função custo: Soma do Erro Quadrático com Esquecimento Exponencial

$$J_{RLS}(n) = \sum_{j=0}^{n} \alpha^{n-j} e^2(j) \tag{61}$$

- A regra de ajuste recursivo é escrita como

$$\mathbf{a}(n+1) = \mathbf{a}(n) + \mathbf{k}(n)e(n) \tag{62}$$

- Com a fórmula recursiva do ganho de Kalman dada por

$$\mathbf{k}(n) = \frac{\mathbf{P}(n)\mathbf{x}(n)}{\alpha + \mathbf{x}^T(n)\mathbf{P}(n)\mathbf{x}(n)} \tag{63}$$

- E a atualização da matriz $\mathbf{P}(n)$ dada em função de $\mathbf{k}(n)$:

$$\mathbf{P}(n+1) = \frac{1}{\alpha}\left[\mathbf{P}(n) - \mathbf{k}(n)\mathbf{x}^T(n)\mathbf{P}(n)\right] \tag{64}$$

- Se substituirmos a Eq. (63) na Eq. (62), obtemos

$$\mathbf{a}(n+1) = \mathbf{a}(n) + \frac{\mathbf{P}(n)\mathbf{x}(n)e(n)}{\alpha + \mathbf{x}^T(n)\mathbf{P}(n)\mathbf{x}(n)}. \tag{65}$$

- Além disso, se fizermos $\mathbf{P}(n) = \mu\mathbf{I}_p$, vamos obter a seguinte expressão:

$$\mathbf{a}(n+1) = \mathbf{a}(n) + \frac{\mathbf{x}(n)e(n)}{\alpha + \mathbf{x}^T(n)\mathbf{x}(n)}, \tag{66}$$

$$= \mathbf{a}(n) + \frac{\mu}{\alpha + \|\mathbf{x}(n)\|^2}e(n)\mathbf{x}(n), \tag{67}$$

que é exatamente a expressão do algoritmo LMS normalizado mostrada na Eq. (47).

Parte II

Estimação Robusta

Suposições por Trás dos Métodos OLS/LMS/RLS

1. Baseados em funções-objetivo que dão mesma importância a todos os erros.

Suposições por Trás dos Métodos OLS/LMS/RLS

1. Baseados em funções-objetivo que dão mesma importância a todos os erros.
2. Todos os erros contribuem igualmente para a solução final.

Suposições por Trás dos Métodos OLS/LMS/RLS

1. Baseados em funções-objetivo que dão mesma importância a todos os erros.
2. Todos os erros contribuem igualmente para a solução final.
3. A solução é ótima apenas sob erros gaussianos!

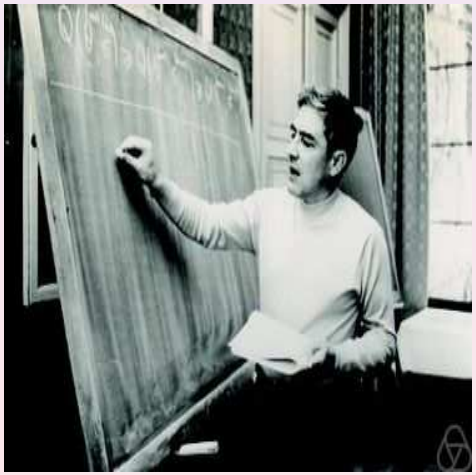
Suposições por Trás dos Métodos OLS/LMS/RLS

1. Baseados em funções-objetivo que dão mesma importância a todos os erros.
2. Todos os erros contribuem igualmente para a solução final.
3. A solução é ótima apenas sob erros gaussianos!
4. Porém, outliers geram erros maiores (i.e. não gaussianos) ...

Suposições por Trás dos Métodos OLS/LMS/RLS

1. Baseados em funções-objetivo que dão mesma importância a todos os erros.
2. Todos os erros contribuem igualmente para a solução final.
3. A solução é ótima apenas sob erros gaussianos!
4. Porém, outliers geram erros maiores (i.e. não gaussianos) ...
5. ... que distorcem a solução em direção aos outliers.

- **Peter J. Huber**[8] (1934 -) introduz o conceito de estimação-M.
- A letra M refere-se a uma estimação do tipo *máxima verossimilhança*.
- A robustez a outliers é obtida pela minimização de uma função diferente daquelas envolvendo soma de erros quadráticos.



[8] P. J. Huber (1964). "Robust Estimation of a Location Parameter", Annals of Mathematical Statistics, 35(1):73–101.

- Baseado na teoria de Huber, um estimador M minimiza a seguinte função objetivo:

$$J_M(n) = \sum_{t=1}^{N} \rho(e(n)) = \sum_{t=1}^{N} \rho(x(n) - \hat{x}(n)), \quad (68)$$

$$= \sum_{t=1}^{N} \rho(x(n) - \mathbf{a}^T(n)\mathbf{x}(n)), \quad (69)$$

em que a função $\rho(\cdot)$ computa a contribuição de cada erro $e(n) = x(n) - \hat{x}(n)$ para a função objetivo.

- A regra OLS é um tipo particular de estimador M, obtido fazendo-se $\rho(e(n)) = e^2(n)$.

- A função $\rho(\cdot)$ deve possuir as seguintes propriedades:

  Propriedade 1 : $\rho(e(n)) \geq 0$.

  Propriedade 2 : $\rho(0) = 0$.

  Propriedade 3 : $\rho(e(n)) = \rho(-e(n))$.

  Propriedade 4 : $\rho(e(n)) \geq \rho(e(n'))$, for $|e(n)| > |e(n')|$.

- A título de exemplo, vamos considerar a função-custo proposta por Huber:

$$\rho(e(n)) = \begin{cases} \frac{1}{2}e^2(n), & |e(n)| \leq k \\ k|e(n)| - \frac{1}{2}k^2, & |e(n)| > k \end{cases} \quad (70)$$

em que $k > 0$ é o limiar de outlier.

- A correspondente função de ponderação $q(n)$ é dada por

$$q(e(n)) = \frac{1}{e(n)}\frac{\partial \rho(e(n))}{\partial e(n)} = \begin{cases} 1, & |e(n)| \leq k \\ \frac{k}{|e(n)|}, & |e(n)| > k \end{cases} \quad (71)$$

- O valor $k = 1{,}345\hat{\sigma}$ é comumente escolhido, onde $\hat{\sigma}$ é uma estimativa robusta da dispersão dos erros.
- Usualmente, escolhe-se $\hat{\sigma} = \text{MAR}/0{,}6745$, em que MAR é a mediana dos resíduos absolutos.
- O valor constante 0,6745 torna $\hat{\sigma}$ uma estimativa não-viesada para erros gaussianos.

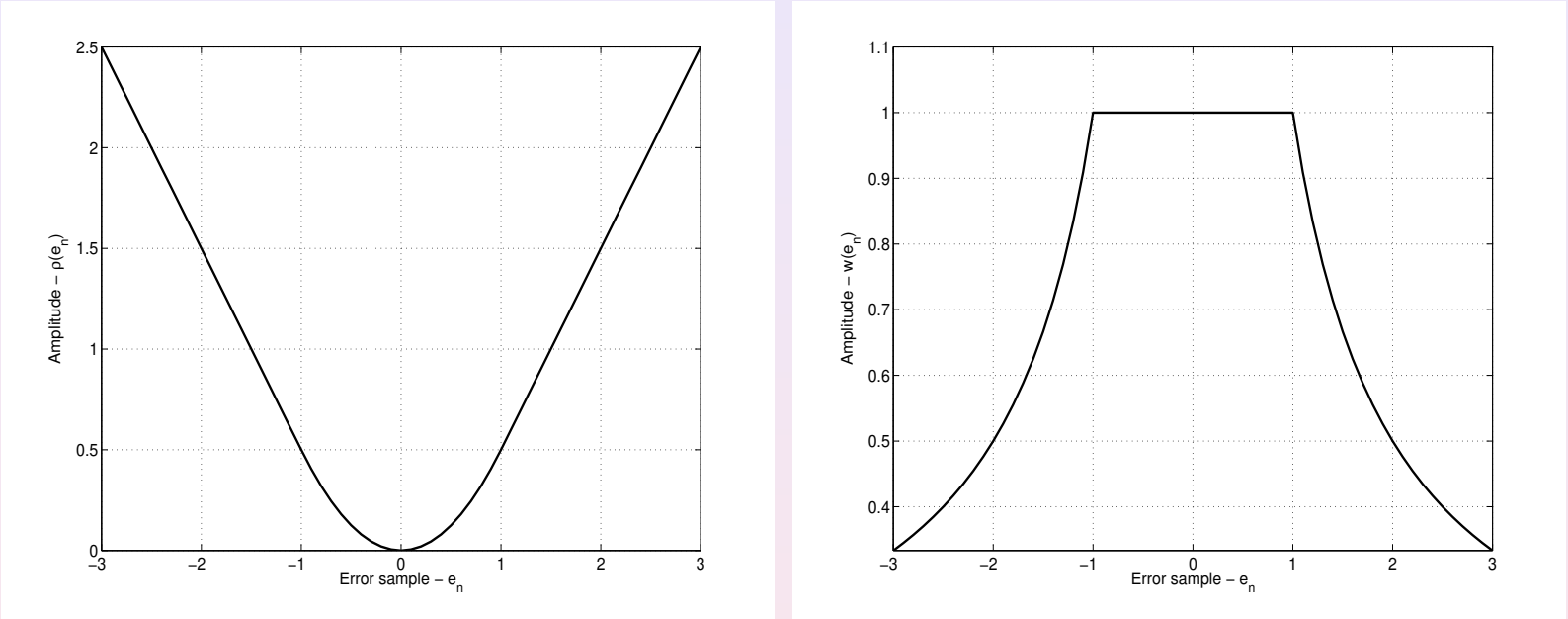


Figure: À esquerda - $\rho(e(n))$: Função-custo de Huber. À direita - $q(e(n))$: função de ponderação correspondente.

- A fim de se obter uma regra de ajuste baseada no estimador M, vamos definir a *função escore* $\psi(e(n)) = \partial\rho(e(n))/\partial e(n)$.
- Derivando ρ com respeito ao vetor de coeficientes $\mathbf{a}(n)$, obtém-se

$$\sum_{t=1}^{N} \psi(x(n) - \mathbf{a}^T\mathbf{x}(n))\mathbf{x}^T(n) = \mathbf{0}, \tag{72}$$

  onde $\mathbf{0}$ é um vetor-linha de zeros de dimensão $(p+1)$.
- A partir da definição da função de ponderação $q(e(n)) = \psi(e(n))/e(n)$, e fazendo $q(n) = q(e(n))$, chegamos a

$$\sum_{t=1}^{N} q(n)(x(n) - \mathbf{a}^T\mathbf{x}(n))\mathbf{x}^T(n) = \mathbf{0}. \tag{73}$$

- Resolver esta equação equivale a resolver um problema de mínimos quadrados ponderados, com função-custo dada por

$$J_M(n) = \sum_{t=1}^{N} q^2(n)e^2(n) = \mathbf{e}^T(n)\mathbf{Q}(n)\mathbf{e}(n), \qquad (74)$$

  em que $\mathbf{Q}(n) = \text{diag}\{q(n)\}$ é uma matriz de pesos $N \times N$.
- Contudo, os valores de $q(n)$ dependem dos resíduos, que dependem dos coeficientes estimados, que dependem dos pesos $q(n)$. Logo, uma solução analítica não é possível como no mtodo OLS.
- Por isso, um método iterativo como o algoritmo IRLS[9] (*iteratively reweighted least-squares*) é usado para este fim.

[9] J. Fox (2002). "An R and S-PLUS Companion to Applied Regression", SAGE Publications.

Algoritmo IRLS

- **Passo 1** - Obtenha uma estimativa inicial $\mathbf{a}(0)$ usando a regra OLS. Faça $r = 1$.
- **Passo 2** - A cada iteração r, calcule os resíduos $e^{(r-1)}(n)$ e os pesos correspondentes $q^{(r-1)}(n) = q[e^{(r-1)}(n)]$ para todos os vetores de entrada $\mathbf{x}(n)$, $n = 1, \ldots, N$, usando a estimativa atual do vetor de coeficientes.
  Passo 3 - Compute a nova estimativa de mínimos quadrados ponderados de $\mathbf{a}^{(r)}$:

$$\mathbf{a}^{(r)} = (\mathbf{X}\mathbf{Q}^{(r-1)}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{Q}^{(r-1)}\mathbf{y}, \tag{75}$$

  onde $\mathbf{Q}^{(r-1)} = \text{diag}\{q^{(r-1)}(n)\}$ é uma matriz de pesos $N \times N$ obtida a partir dos resíduos de todos os N vetores de entrada. Faça $r = r + 1$ e repita os Passos 2 e 3 até a convergência do vetor de coeficientes $\mathbf{a}^{(r)}$.

Algoritmo LMM (Least mean M-estimate)

- Função-custo[10]:

$$J_{LMM}(n) = \rho(e(n)) = \rho(x(n) - \hat{x}(n)), \quad (76)$$
$$= \rho(x(n) - \mathbf{a}^T(n)\mathbf{x}(n)), \quad (77)$$

onde $\mathbf{a}(n)$ é a estimativa atual do vetor de coeficientes.

- A regra de ajuste recursivo de $\mathbf{a}(n)$ é dada por

$$\mathbf{a}(n+1) = \mathbf{a}(n) - \mu\frac{\partial J_{LMM}(n)}{\partial \mathbf{a}(n)}, \quad (78)$$
$$= \mathbf{a}(n) + \mu q(e(n))e(n)\mathbf{x}(n), \quad (79)$$
$$= \mathbf{a}(n) + \mu\psi(e(n))\mathbf{x}(n), \quad (80)$$

onde usamos o fato de que $q(e(n)) = \psi(e(n))/e(n)$.

[10] Y. Zou, S. C. Chan & T. S. Ng (2000). "Least mean M-estimate algorithms for robust adaptive filtering in impulsive noise", IEEE Transactions on Circuits and Systems II, 47(12):1564–1569.

Algoritmo LMS

- **Função-objetivo**: $J_{LMS}(n) = \frac{1}{2}e^2(n) = \frac{1}{2}(x(n) - \hat{x}(n))^2$
- **Regra de Ajuste**:

$$\mathbf{a}(n+1) = \mathbf{a}(n) + \mu e(n)\mathbf{x}(n) \quad (81)$$
$$= \mathbf{a}(n) + \mu(x(n) - \hat{x}(n))\mathbf{x}(n) \quad (82)$$

Algoritmo LMM

- **Função-objetivo**: $J_{LMM}(n) = \rho(e(n)) = \rho(x(n) - \hat{x}(n))$
- **Regra de Ajuste**:

$$\mathbf{a}(n+1) = \mathbf{a}(n) + \mu q(e(n))e(n)\mathbf{x}(n) \quad (83)$$
$$= \mathbf{a}(n) + \mu q(e(n))(x(n) - \hat{x}(n))\mathbf{x}(n) \quad (84)$$

- Uma outra técnica de estimação recursiva robusta muito interessante para identificação de sistemas é conhecida como algoritmo RLM (*recursive least M-estimate*)[11].
- Esta técnica baseia-se na minimização da seguinte função-custo:

$$J_{RLM}(n) = \sum_{j=0}^{n} \alpha^{n-j} \rho(e(j)), \quad (85)$$

$$= \sum_{j=0}^{n} \alpha^{n-j} \rho(x(j) - \mathbf{a}^T(j)\mathbf{x}(j)), \quad (86)$$

em que $0 \leq \alpha \leq 1$ é o fator de esquecimento.

[11] Y. Zou, S. C. Chan & T. S. Ng (2000). "A Recursive Least M-Estimate (RLM) Adaptive Filter for Robust Filtering in Impulse Noise", IEEE Signal Processing Letters, 7(11):324–326.

- De modo similar ao algoritmo RLS, uma equação recursiva para o algoritmo RLM é dada por

$$\mathbf{a}(n+1) = \mathbf{a}(n) + \hat{\mathbf{R}}^{-1}(n)\mathbf{x}(n)e(n), \tag{87}$$

em que a matriz de correlação robusta sendo dada por

$$\begin{aligned} \hat{\mathbf{R}}(n) &= \sum_{j=0}^{n} \alpha^{n-j} q(e(n))\mathbf{x}(j)\mathbf{x}^T(j), & (88) \\ &= \alpha\hat{\mathbf{R}}(n-1) + q(e(n))\mathbf{x}(n)\mathbf{x}^T(n) & (89) \end{aligned}$$

com $e(n) = x(n) - \mathbf{a}^T(n)\mathbf{x}(n)$.

- Contudo, esta equação exige a inversão da matriz de correlação a cada instante n, o que não é interessante em aplicações práticas.

Resumo do Algoritmo RLM

- Função custo: $J_{RLM}(n) = \sum_{j=0}^{n} \alpha^{n-j} \rho(e(j))$
- A regra de ajuste recursivo é escrita como

$$\mathbf{a}(n+1) = \mathbf{a}(n) + \mathbf{k}(n)e(n) \tag{90}$$

- Com a fórmula recursiva do ganho de Kalman dada por

$$\mathbf{k}(n) = \frac{q(e(n))\mathbf{P}(n)\mathbf{x}(n)}{\alpha + q(e(n))\mathbf{x}^T(n)\mathbf{P}(n)\mathbf{x}(n)} \tag{91}$$

- E a atualização da matriz $\mathbf{P}(n)$ dada em função de $\mathbf{k}(n)$:

$$\mathbf{P}(n+1) = \frac{1}{\alpha}\left[\mathbf{P}(n) - \mathbf{k}(n)\mathbf{x}^T(n)\mathbf{P}(n)\right] \tag{92}$$

- O limiar de outliers (k) que aparece nas Eqs. (70) e (71) é mantido constante durante o processo de estimação. Contudo, em *tracking* de sinais não-estacionários, este procedimento não é indicado e o valor do limiar deve ser adaptativo.
- Assim, deve-se estimar a variância do erro recursivamente:

$$\hat{\sigma}^2(n) = \lambda_\sigma \hat{\sigma}^2(n-1) + (1-\lambda_\sigma)e^2(n), \qquad (93)$$

em que λ_σ é uma constante de esquecimento.

- Contudo, esta equação não é robusta a outliers, sendo dada preferência à seguinte equação:

$$\hat{\sigma}^2(n) = \lambda_\sigma \hat{\sigma}^2(n-1) + C_1(1-\lambda_\sigma)\text{MED}_m[A_e(n)], \qquad (94)$$

em que $A_e(n) = \{e^2(n), \ldots, e^2(n-m+1)\}$, m é o tamanho da janela e $C_1 = 1,483(1+5/(m-1))$.

Algoritmo RLS

- **Função-objetivo**: $J_{RLS}(n) = \sum_{j=0}^{n} \alpha^{n-j} e^2(j)$
- **Regra de Ajuste**: $\mathbf{a}(t+1) = \mathbf{a}(n) + \mathbf{k}(n)e(n)$,

$$\mathbf{k}(n) = \frac{\mathbf{P}(n)\mathbf{x}(n)}{\alpha + \mathbf{x}^T(n)\mathbf{P}(n)\mathbf{x}(n)} \tag{95}$$

Algoritmo RLM

- **Função-objetivo**: $J_{RLM}(n) = \sum_{j=0}^{n} \alpha^{n-j} \rho(e(j))$
- **Regra de Ajuste**: $\mathbf{a}(t+1) = \mathbf{a}(n) + \mathbf{k}(n)e(n)$,

$$\mathbf{k}(n) = \frac{q(e(n))\mathbf{P}(n)\mathbf{x}(n)}{\alpha + q(e(n))\mathbf{x}^T(n)\mathbf{P}(n)\mathbf{x}(n)} \tag{96}$$

Para ambas as versões:

$$\mathbf{P}(n+1) = \frac{1}{\alpha}\left[\mathbf{P}(n) - \mathbf{k}(n)\mathbf{x}^T(n)\mathbf{P}(n)\right] \tag{97}$$