PSI3471 – Fundamentos de Sistemas Eletrônicos Inteligentes
O algoritmo LMS

Magno T. M. Silva e Renato Candido

Escola Politécnica da USP

1 Regressão Linear Multivariada

Dado o conjunto de treinamento

$$\{(x_{11}, x_{21}, \cdots, x_{M1}, d_1), (x_{12}, x_{22}, \cdots, x_{M2}, d_2), \cdots,$$

$$(x_{1N_t}, x_{2N_t}, \cdots, x_{MN_t}, d_{N_t})\},$$

já aprendemos a obter o modelo de regressão linear multivariada

$$y = b + w_1x_1 + w_2x_2 + \cdots + w_Mx_M \approx d,$$

em que

- N_t é o número de dados utilizados no treinamento
- b o *bias*
- d o sinal desejado
- y a estimativa de d
- x o sinal de entrada e
- w_k, $k = 1, \cdots, M$ os pesos do regressor

2 Regressão Linear Multivariada

A solução

$$\boxed{\mathbf{w}^{\mathrm{o}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{d}}$$

em que

$$\mathbf{w}^{\mathrm{o}} = \begin{bmatrix} b^{\mathrm{o}} \\ w_1^{\mathrm{o}} \\ \vdots \\ w_M^{\mathrm{o}} \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{21} & \cdots & x_{M1} \\ 1 & x_{12} & x_{22} & \cdots & x_{M2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1N_t} & x_{2N_t} & \cdots & x_{MN_t} \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{N_t} \end{bmatrix}$$

minimiza

$$J(\mathbf{w}) = \|\mathbf{e}\|^2 = \|\mathbf{d} - \mathbf{X}\mathbf{w}\|^2,$$

de modo que $\mathbf{w}^{\mathrm{o}} = \mathrm{argmin}_{\mathbf{w}} J(\mathbf{w})$.

- O regressor é obtido a partir da matriz $\mathbf{X}$ e do vetor $\mathbf{d}$ que levam em conta todos os N_t exemplos de treinamento

3 Usando o índice n

O regressor também pode ser obtido a partir de um treinamento iterativo. Para obter esse algoritmo, vamos usar n $(n = 1{,}2, \cdots, N_t)$ para indicar sua n-ésima iteração:

- vetor de pesos: $\mathbf{w}(n) = [\, b(n)\ w_1(n)\ \cdots\ w_M(n)\,]^T$
- sinal desejado: $d(n) = d_n$,
- vetor de entrada: $\mathbf{x}(n) = [\, 1\ x_{1n}\ x_{2n}\ \cdots\ x_{Mn}\,]^T$
- sinal de "saída":

$$y(n) = \mathbf{x}^T(n)\mathbf{w}(n-1) = b(n-1) + \sum_{k=1}^{M} x_{kn} w_k(n-1),$$

  em que $\mathbf{w}(0) = \mathbf{0}$
- erro de "estimação":
  $e(n) = d(n) - y(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n-1)$

$$= d(n) - b(n-1) - \sum_{k=1}^{M} x_{kn} w_k(n-1)$$

4 O vetor gradiente

- Os pesos são ajustados para minimizar o MSE (*mean-square error*):

$$J_{\mathrm{MSE}}(\mathbf{w}) = \mathrm{E}\{e^2(n)\},$$

em que $\mathrm{E}\{\cdot\}$ representa esperança matemática

- Derivando $J_{\mathrm{MSE}}(\mathbf{w})$ em relação a $\mathbf{w}$, obtemos o vetor gradiente

$$\boldsymbol{\nabla}_{\mathbf{w}} J_{\mathrm{MSE}}(\mathbf{w}(n-1)) = \frac{\partial \mathrm{E}\{e^2(n)\}}{\partial \mathbf{w}(n-1)} = 2\mathrm{E}\left\{e(n)\frac{\partial e(n)}{\partial \mathbf{w}(n-1)}\right\}$$

$$= 2\mathrm{E}\left\{e(n)\begin{bmatrix} \frac{de(n)}{db(n-1)} \\ \frac{de(n)}{dw_1(n-1)} \\ \vdots \\ \frac{de(n)}{dw_M(n-1)} \end{bmatrix}\right\} = 2\mathrm{E}\left\{e(n)\begin{bmatrix} -1 \\ -x_{1n} \\ \vdots \\ -x_{Mn} \end{bmatrix}\right\}$$

$$= -2\mathrm{E}\{e(n)\mathbf{x}(n)\}.$$

5 A solução de Wiener

Igualando o gradiente ao vetor nulo, obtemos

$$\mathrm{E}\{\mathbf{x}(n)[d(n) - \mathbf{x}^T(n)\mathbf{w}(n-1)]\} = \mathbf{0} \Rightarrow$$
$$\underbrace{\mathrm{E}\{\mathbf{x}(n)\mathbf{x}^T(n)\}}_{\mathbf{R}}\mathbf{w}^{\text{wiener}} = \underbrace{\mathrm{E}\{d(n)\mathbf{x}(n)\}}_{\mathbf{p}}.$$

- A solução dessa equação leva ao MSE mínimo e é conhecida como solução de Wiener

$$\boxed{\mathbf{w}^{\text{wiener}} = \mathbf{R}^{-1}\mathbf{p}}$$

  - $\mathbf{R}$ é a matriz de autocorrelação dos dados de entrada e
  - $\mathbf{p}$ o vetor de correlação cruzada entre o sinal desejado $d(n)$ e os dados de entrada.

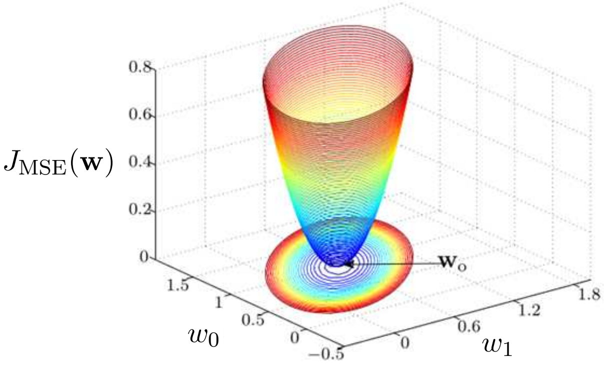
- Podemos estimar $\mathbf{R}$ e $\mathbf{p}$ como

$$\widehat{\mathbf{R}} = \frac{1}{N_t}\sum_{n=1}^{N_t}\mathbf{x}(n)\mathbf{x}^T(n) \quad \text{e} \quad \widehat{\mathbf{p}} = \frac{1}{N_t}\sum_{n=1}^{N_t} d(n)\mathbf{x}(n).$$

Neste caso, a solução obtida com a regressão linear multivariada coincide com a solução de Wiener.

6 MSE como função custo

- $J_{\mathrm{MSE}}(\mathbf{w})$ tem um único ponto de mínimo que corresponde à solução de Wiener



7 O algoritmo steepest descent

- A solução de Wiener pode ser obtida de maneira iterativa

$$\mathbf{w}(n) = \mathbf{w}(n-1) - \frac{\eta}{2}\boldsymbol{\nabla}_{\mathbf{w}} J_{\mathrm{MSE}}(\mathbf{w}(n-1)),$$

  em que η é um passo de adaptação.
- Substituindo a expressão do gradiente, chega-se a

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \eta \mathrm{E}\{e(n)\mathbf{x}(n)\},$$

  ou ainda

$$\boxed{\mathbf{w}(n) = \mathbf{w}(n-1) + \eta\left[\mathbf{p} - \mathbf{R}\mathbf{w}(n-1)\right].}$$

  Esse algoritmo é conhecido como *steepest descent algorithm* ou algoritmo do gradiente exato.
- Se o intervalo $0 < \eta < 2/\lambda_{\mathrm{max}}$ for atendido, em que λ_{max} é o autovalor máximo de $\mathbf{R}$, essa equação converge exatamente para a solução de Wiener.
- Sua única vantagem é evitar calcular a inversa da matriz $\mathbf{R}$

8 O algoritmo LMS

- Uma maneira de simplificar os cálculos é estimar $\mathbf{R}$ e $\mathbf{p}$ instantâneamente

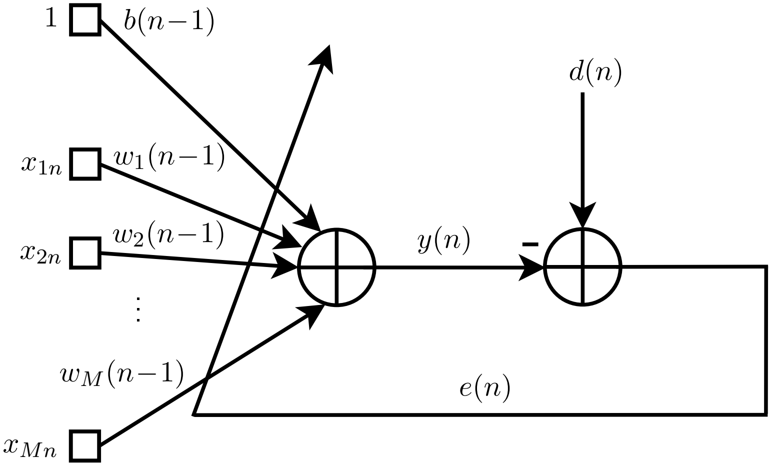
$$\widehat{\mathbf{R}}(n) = \mathbf{x}(n)\mathbf{x}^T(n) \quad \text{e} \quad \widehat{\mathbf{p}}(n) = d(n)\mathbf{x}(n).$$

- Substituindo essas aproximações no algoritmo *steepest descent*, chega-se ao algoritmo LMS (*least-mean-square*)

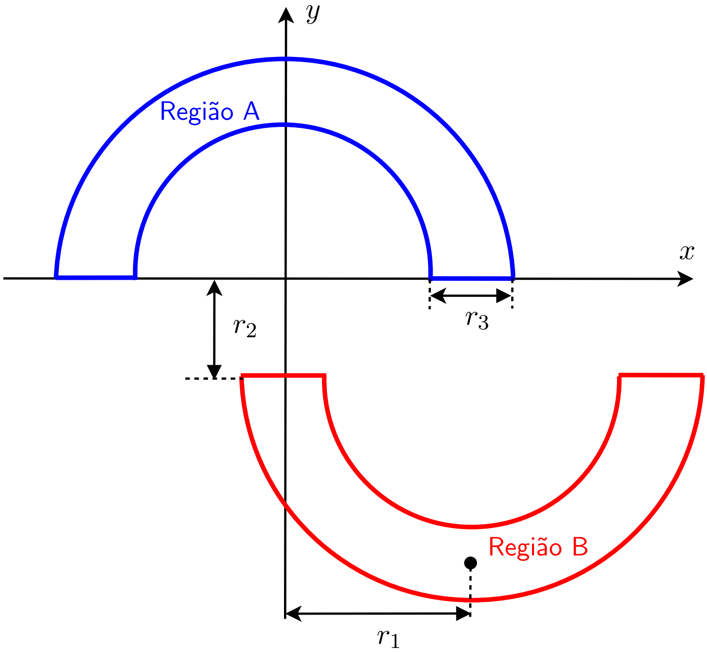
$$\boxed{\mathbf{w}(n) = \mathbf{w}(n-1) + \eta e(n)\mathbf{x}(n),}$$

- η tem um papel fundamental na convergência do LMS:
  - Quanto menor o valor de η, mais próximo da solução de Wiener o algoritmo LMS estará quando atingir o regime estacionário. No entanto, passos muito pequenos levam a uma convergência lenta.
  - Passos grandes podem representar convergências rápidas, mas também podem levar o algoritmo à divergência.

9 O algoritmo LMS



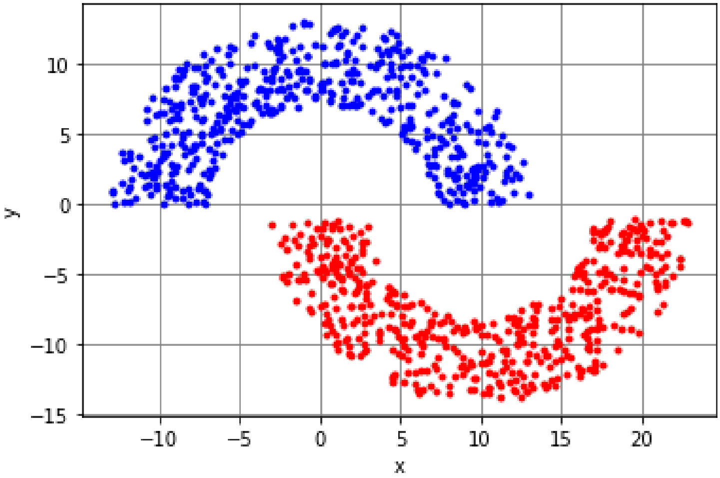
10 Exemplo de classificação com o LMS



11 Exemplo de classificação com o LMS

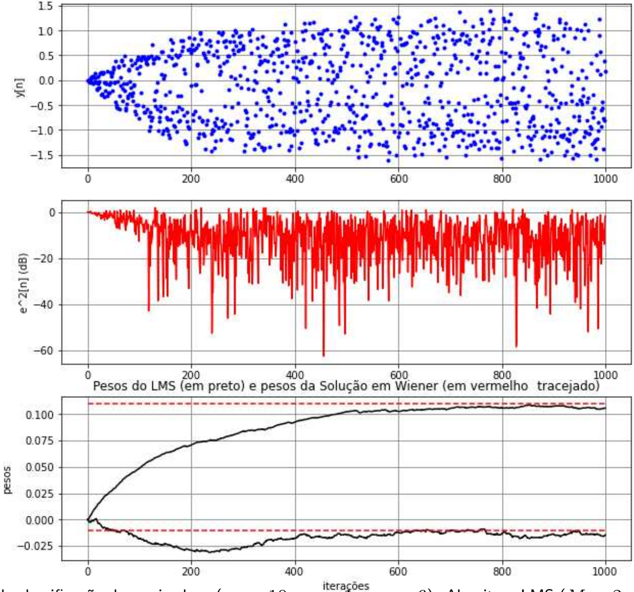
- 1000 pontos de treinamento (500 pertencentes a cada região)
- Para a Região A, considera-se $d = 1$ e para Região B, $d = -1$
- 2000 pontos de teste (1000 pertencentes a cada região), gerados de forma independente do conjunto de treinamento

12 Exemplo de classificação com o LMS



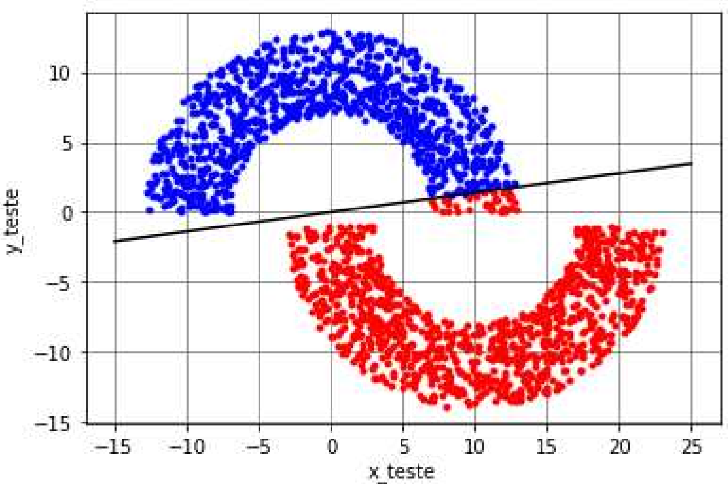
O problema de classificação das meias-luas ($r_1 = 10$, $r_2 = 1$ e $r_3 = 6$).
Dados de treinamento ($N_t = 1000$)

13 Exemplo de classificação com o LMS



O problema de classificação das meias-luas ($r_1 = 10$, $r_2 = 1$ e $r_3 = 6$). Algoritmo LMS ($M = 2$, $\eta = 10^{-4}$): saída do algoritmo, erro quadrático em dB e pesos ao longo das iterações. As retas vermelhas tracejadas representam os valores dos pesos da solução de Wiener.

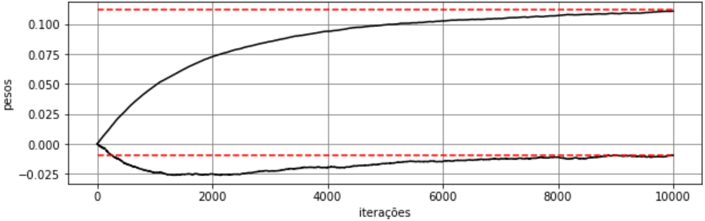
14 Exemplo de classificação com o LMS



O problema de classificação das meias-luas ($r_1 = 10$, $r_2 = 1$ e $r_3 = 6$). Dados de teste ($N_{\text{teste}} = 2000$) e reta de separação das duas regiões obtida com o LMS ($M = 2$, $\eta = 10^{-4}$); Taxa de erro de 2,5%.

15 Exemplo de classificação com o LMS

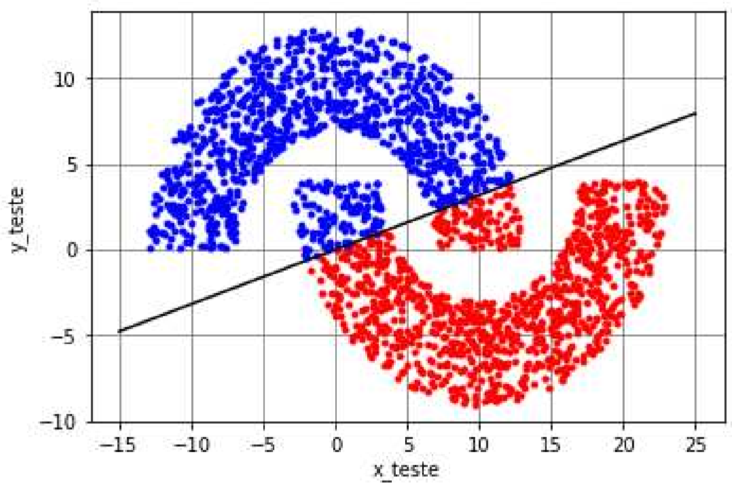
Passo de adaptação menor



O problema de classificação das meias-luas ($r_1 = 10$, $r_2 = 1$ e $r_3 = 6$). Algoritmo LMS ($M = 2$, $\eta = 10^{-5}$): pesos ao longo das iterações. As retas vermelhas tracejadas representam os valores dos pesos da solução de Wiener

16 Exemplo de classificação com o LMS

$r_2 = -4$



O problema de classificação das meias-luas ($r_1 = 10$, $r_2 = -4$ e $r_3 = 6$). Dados de teste ($N_{\text{teste}} = 2000$) e reta de separação das duas regiões obtida com o LMS ($M = 2$, $\eta = 10^{-4}$); Taxa de erro de 12%.

17 Época

O que fazer quando a quantidade de dados é limitada e insuficiente para possibilitar a convergência dos algoritmos no treinamento?

- A solução é utilizar os dados de treinamento mais de uma vez.
- O treinamento realizado com o conjunto completo dos dados é chamado de **época**.
- Os algoritmos podem levar várias épocas até convergir.
- Para gerar diversidade entre épocas, os dados de treinamento devem ser misturados antes de se iniciar uma nova época.

18 Modo estocástico e conceito de iteração

- O ajuste dos pesos do algoritmo LMS ocorre de maneira estocástica.
- O gradiente da função custo é estimado de maneira instantânea, a cada dado de treinamento.
- Neste modo, há N_t atualizações dos pesos do LMS por época.
- A **iteração** do algoritmo ocorre toda vez que os pesos são atualizados.
- No caso estocástico, temos N_t iterações e o índice n coincide com iteração, pois o vetor de pesos é atualizado a cada n, ou seja, a cada dado de treinamento.
- Essa forma de atualização estocástica é útil em problemas de tempo real.
- No entanto, problemas de tempo real não são a maioria entre os problemas de aprendizado de máquina.

19 Modo *batch*

- Em aprendizado de máquina, geralmente não estamos interessados em fazer a inferência durante o treinamento.
- A saída e o erro são utilizados no treinamento apenas para atualizar os pesos do algoritmo.
- Depois do treinamento, fixam-se os pesos para então se fazer a inferência e testar o classificador ou regressor.
- Vamos agora analisar outro caso extremo, em que todos os dados de treinamento são utilizados para estimar o vetor gradiente.
- Neste caso, o vetor de pesos será atualizado apenas uma vez a cada época. Portanto, teremos apenas uma iteração por época.

20 Modo *batch*

- Suponha que o vetor de pesos do LMS foi atualizado no final da época $k-1$: $\mathbf{w}(k-1)$.
- Ele será atualizado novamente apenas no final da época k.
- Durante a época k, estima-se o vetor gradiente como

$$\widehat{\boldsymbol{\nabla}}_{\mathbf{w}} J_{\text{MSE}}(\mathbf{w}(k-1)) = -\frac{2}{N_t} \sum_{n=1}^{N_t} \left[d(n) - \mathbf{x}^T(n)\mathbf{w}(k-1)\right] \mathbf{x}(n).$$

- Esse gradiente deve ser então utilizado no final da época k para atualizar $\mathbf{w}(k-1)$, ou seja,

$$\mathbf{w}(k) = \mathbf{w}(k-1) - \frac{\eta}{2} \widehat{\boldsymbol{\nabla}}_{\mathbf{w}} J_{\text{MSE}}(\mathbf{w}(k-1)).$$

- Na sequência, o vetor $\mathbf{w}(k)$ é utilizado para estimar o gradiente na época $k+1$ e assim sucessivamente.
- Essa forma de atualização do vetor de pesos é chamada de modo **batch**.

21 Modo *batch*

- No modo **batch**, o algoritmo LMS busca minimizar em cada época a seguinte aproximação da função custo:

$$\widehat{J}_{\mathrm{MSE}}(\mathbf{w}(k-1)) = \frac{1}{N_t}\sum_{n=1}^{N_t} e_{k-1}^2(n) = \frac{1}{N_t}\sum_{n=1}^{N_t}[d(n)-\mathbf{x}^T(n)\mathbf{w}(k-1)]^2,$$

em que $k = 1,2,\cdots,N_e$, sendo N_e o número de épocas.

22 Modo *batch*

Observações:

- O treinamento em modo **batch** não é utilizado em aplicações de tempo real, pois gera um atraso inaceitável.
- O índice n neste modo de treinamento não representa iteração e sim a posição do dado no banco de dados de treinamento.
- Como os dados são misturados de uma época para outra, o vetor $\mathbf{x}(5)$ da época k pode ser o vetor $\mathbf{x}(200)$ da época $k-1$.
- Na formulação anterior, a iteração foi representada por k, que coincide com as épocas do treinamento.
- Os índices $k-1$ e n no erro $e_{k-1}(n)$ foram utilizados para indicar que ele é calculado com o vetor de pesos $\mathbf{w}(k-1)$ e com os dados de treinamento $\mathbf{x}(n)$ e $d(n)$ da posição n, respectivamente.

23 Modo *batch* – Formulação matricial

Definindo

$$\mathbf{w}(k-1) = \begin{bmatrix} b(k-1) \\ w_1(k-1) \\ \vdots \\ w_M(k-1) \end{bmatrix}, \mathbf{d}(k) = \begin{bmatrix} d(1) \\ d(2) \\ \vdots \\ d(N_t) \end{bmatrix}, \mathbf{e}(k) = \begin{bmatrix} e_{k-1}(1) \\ e_{k-1}(2) \\ \vdots \\ e_{k-1}(N_t) \end{bmatrix}$$

e a matriz

$$\mathbf{X}(k) = \begin{bmatrix} \mathbf{x}^T(1) \\ \mathbf{x}^T(2) \\ \vdots \\ \mathbf{x}^T(N_t) \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{21} & \cdots & x_{M1} \\ 1 & x_{12} & x_{22} & \cdots & x_{M2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1N_t} & x_{2N_t} & \cdots & x_{MN_t} \end{bmatrix},$$

pode-se calcular o vetor de erros $\mathbf{e}(k)$ como

$$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{X}(k)\mathbf{w}(k-1),$$

e a estimativa do vetor gradiente como

$$\widehat{\boldsymbol{\nabla}}_{\mathbf{w}} J_{\mathrm{MSE}}(\mathbf{w}(k-1)) = -\frac{2}{N_t}\mathbf{X}^T(k)\mathbf{e}(k).$$

24 Modo *batch* – Formulação matricial

Essa estimativa do gradiente leva à seguinte atualização dos pesos:

$$\boxed{\mathbf{w}(k) = \mathbf{w}(k-1) + \frac{\eta}{N_t}\mathbf{X}^T(k)\mathbf{e}(k).}$$

Com essa notação, a aproximação da função custo que o LMS busca minimizar neste modo pode ser reescrita como

$$\widehat{J}_{\mathrm{MSE}}(\mathbf{w}(k-1)) = \frac{1}{N_t}\|\mathbf{e}(k)\|^2.$$

- Essa forma de atualização é mais eficiente, pois permite que as contas sejam realizadas em paralelo.

25 Modo *mini-batch*

- Considere que, em toda época, os dados de treinamento sejam divididos em conjuntos de tamanho $N_b < N_t$, que é chamado na literatura de tamanho do **mini-batch**.
- Neste caso, teremos $N_{mb} \triangleq \lfloor N_t/N_b \rfloor$ conjuntos de dados a cada época.
- Considere que o algoritmo utilize cada um desses conjuntos para estimar o vetor gradiente e com essa estimativa atualize os pesos.
- Os pesos serão atualizados N_{mb} vezes por época, a cada N_b dados de treinamento (haverá N_{mb} iterações por época).

26 Modo *mini-batch* – Formulação matricial

Definindo na iteração m

$$\mathbf{w}(m-1) = \begin{bmatrix} b(m-1) \\ w_1(m-1) \\ \vdots \\ w_M(m-1) \end{bmatrix}, \quad \mathbf{d}(\ell) = \begin{bmatrix} d(\ell N_b+1) \\ d(\ell N_b+2) \\ \vdots \\ d(\ell N_b+N_b) \end{bmatrix},$$

$$\mathbf{e}_{m-1}(\ell) = \begin{bmatrix} e_{m-1}(\ell N_b+1) \\ e_{m-1}(\ell N_b+2) \\ \vdots \\ e_{m-1}(\ell N_b+N_b) \end{bmatrix},$$

$$\mathbf{X}(\ell) = \begin{bmatrix} \mathbf{x}^T(\ell N_b+1) \\ \mathbf{x}^T(\ell N_b+2) \\ \vdots \\ \mathbf{x}^T(\ell N_b+N_b) \end{bmatrix} = \begin{bmatrix} 1 & x_{1(\ell N_b+1)} & x_{2(\ell N_b+1)} & \cdots & x_{M(\ell N_b+1)} \\ 1 & x_{1(\ell N_b+2)} & x_{2(\ell N_b+2)} & \cdots & x_{M(\ell N_b+2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1(\ell N_b+N_b)} & x_{2(\ell N_b+N_b)} & \cdots & x_{M(\ell N_b+N_b)} \end{bmatrix}$$

em que $m = 1,2,\cdots,N_e N_{mb}$ e $\ell = 0, 1, 2, \cdots, N_{mb} - 1$, o vetor de erros é dado por $\mathbf{e}_{m-1}(\ell) = \mathbf{d}(\ell) - \mathbf{X}(\ell)\mathbf{w}(m-1)$

27 Modo *mini-batch* – Formulação matricial

A estimativa do vetor gradiente é dada por

$$\widehat{\boldsymbol{\nabla}}_{\mathbf{w}} J_{\mathrm{MSE}}(\mathbf{w}(m-1)) = -\frac{2}{N_b}\mathbf{X}^T(\ell)\mathbf{e}_{m-1}(\ell),$$

que leva à seguinte atualização dos pesos:

$$\boxed{\mathbf{w}(m) = \mathbf{w}(m-1) + \frac{\eta}{N_b}\mathbf{X}^T(\ell)\mathbf{e}_{m-1}(\ell).}$$

A aproximação da função custo que o LMS busca minimizar a cada *mini-batch* é

$$\widehat{J}_{\mathrm{MSE}}(\mathbf{w}(m-1)) = \frac{1}{N_b}\|\mathbf{e}_{m-1}(\ell)\|^2.$$

- Observe que $N_b = 1$ leva ao modo de treinamento estocástico e $N_b = N_t$ ao modo *batch*.
- O modo *mini-batch* obtém uma melhor estimativa do gradiente em comparação com o estocástico e um menor custo computacional em comparação com o modo *batch*.

28 Exemplo do LMS nos três modos de treinamento

- Vamos considerar a identificação do seguinte sistema

$$\mathbf{w}^{\text{wiener}} = [\, w_0^{\text{wiener}} \;\; w_1^{\text{wiener}} \,]^T = [\, 2 \;\; -3 \,]^T$$

- Entrada: ruído branco gaussiano, com média zero e variância unitária com amostras organizadas na matriz

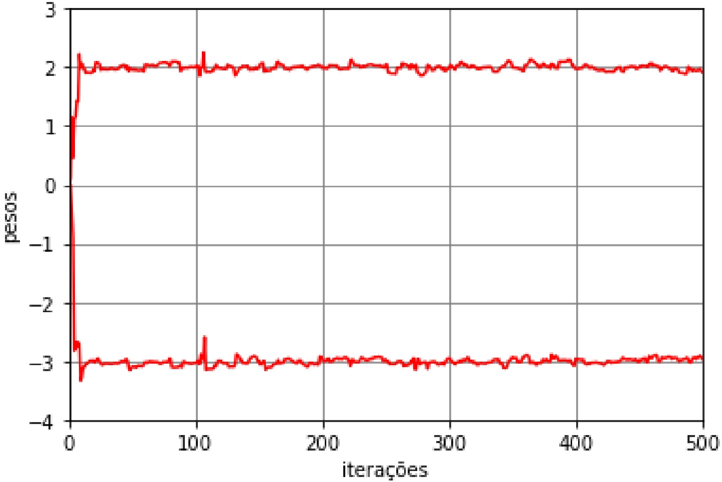
$$\mathbf{X} = \begin{bmatrix} x(0) & 0 \\ x(1) & x(0) \\ x(2) & x(1) \\ \vdots & \vdots \\ x(N_t-2) & x(N_t-3) \\ x(N_t-1) & x(N_t-2) \end{bmatrix}$$

- Sinal desejado:

$$d(n) = w_0^{\text{wiener}} x(n) + w_1^{\text{wiener}} x(n-1) + v(n) = 2x(n) - 3x(n-1) + v(n)$$

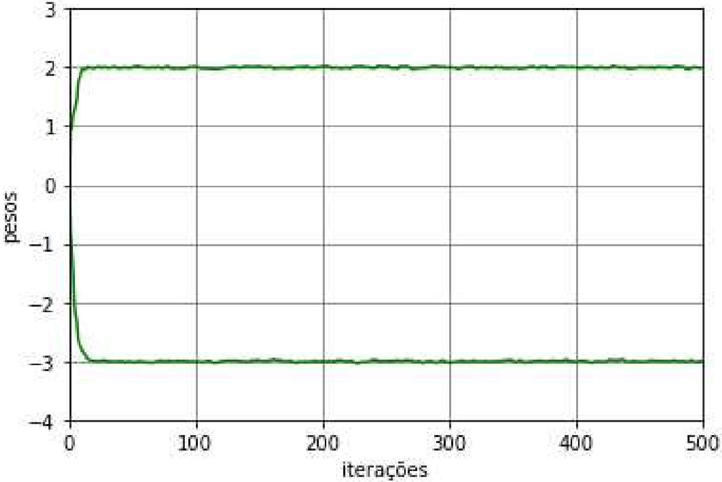
$v(n)$ é um ruído de medida, branco gaussiano, com média zero e variância $\sigma_v^2 = 0{,}01$

29 LMS no modo estocástico



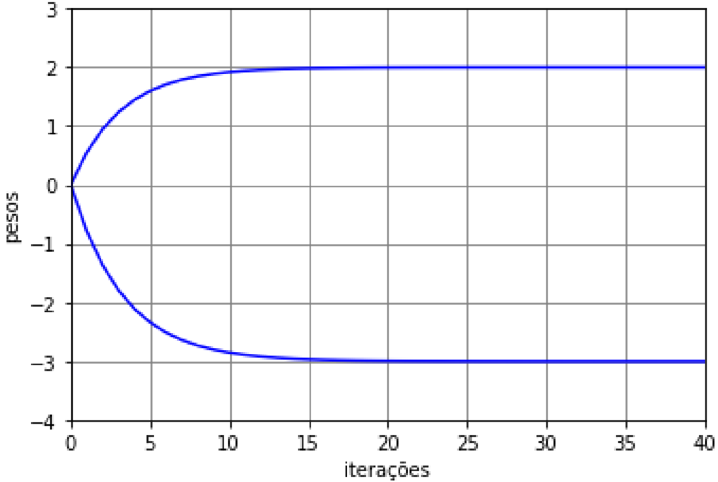
Pesos do algoritmo LMS no modo de treinamento estocástico ($M = 2$, $\eta = 0{,}25$, $N_t = 500$, $N_e = 1$ e $N_b = 1$). Identificação do sistema $\mathbf{w}^{\text{wiener}} = [\,2 \quad -3\,]^T$ com $\sigma_v^2 = 0{,}01$.

30 LMS no modo *mini-batch*



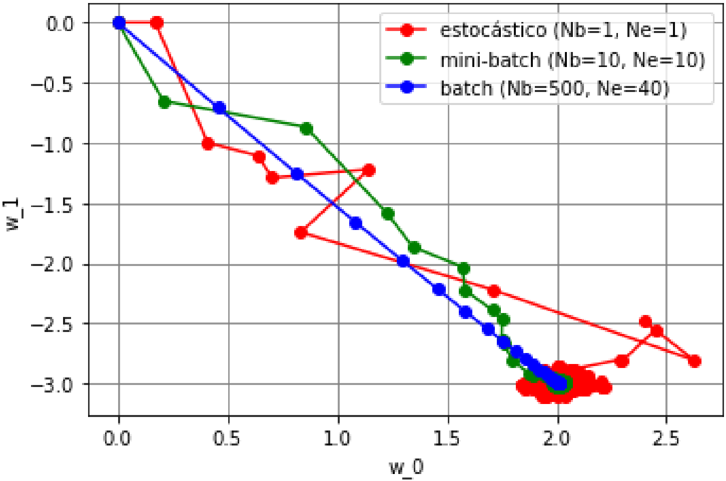
Pesos do algoritmo LMS no modo de treinamento *mini-batch* ($M = 2$, $\eta = 0{,}25$, $N_e = 10$ e $N_b = 10$). Identificação do sistema $\mathbf{w}^{\text{wiener}} = [\,2 \quad -3\,]^T$ com $\sigma_v^2 = 0{,}01$.

31 LMS no modo *batch*



Pesos do algoritmo LMS no modo de treinamento *batch* ($M = 2$, $\eta = 0{,}25$, $N_t = 500$, $N_e = 40$ e $N_b = N_t = 500$). Identificação do sistema $\mathbf{w}^{\text{wiener}} = [\,2 \quad -3\,]^T$ com $\sigma_v^2 = 0{,}01$.

32 Trajetórias dos pesos



Trajetória dos pesos do algoritmo LMS ($M = 2$, $\eta = 0{,}25$) nos três modos de treinamento ($N_t = 500$): estocástico ($N_e = 1$ e $N_b = 1$), *mini-batch* ($N_e = 10$ e $N_b = 10$) e *batch* ($N_e = 40$ e $N_b = 500$). Identificação do sistema $\mathbf{w}^{\text{wiener}} = [\,2 \quad -3\,]^T$ com $\sigma_v^2 = 0{,}01$.