Signal Processing Systems

Machine learning for signal processing *[5LSL0]*

Ruud van Sloun, Rik Vullings

Technische Universiteit
Eindhoven
University of Technology

Where innovation starts

April 2019

Main purpose of this course:

Describe machine learning from a signal processing perspective + hands-on experience

Introduction

- **Main content:**

Introduction

- **Main content:**
  - Optimum linear filters
  - Adaptive Signal Processing
  - Activation functions for classification
  - Optimization and regularization
  - Deep (convolutional) neural networks
  - Variational neural networks

Technische Universiteit Eindhoven University of Technology

Organization and assessment

- **Organization:**
  - Lab sessions: work in groups of 2 students and submit report

Organization and assessment

- **Organization:**
  - Lab sessions: work in groups of 2 students and submit report
  - Project: work in groups of 5 students and give presentation

Organization and assessment

- **Organization:**
  - Lab sessions: work in groups of 2 students and submit report
  - Project: work in groups of 5 students and give presentation

- **Assessment:**

Organization and assessment

- **Organization:**
  - Lab sessions: work in groups of 2 students and submit report
  - Project: work in groups of 5 students and give presentation

- **Assessment:**
  - Lab sessions: 20% of grade

Organization and assessment

- **Organization:**
  - Lab sessions: work in groups of 2 students and submit report
  - Project: work in groups of 5 students and give presentation

- **Assessment:**
  - Lab sessions: 20% of grade
  - Oral exam:

Organization and assessment

- **Organization:**
  - Lab sessions: work in groups of 2 students and submit report
  - Project: work in groups of 5 students and give presentation

- **Assessment:**
  - Lab sessions: 20% of grade
  - Oral exam:
    - With group in week 26 and 27

Organization and assessment

- **Organization:**
  - Lab sessions: work in groups of 2 students and submit report
  - Project: work in groups of 5 students and give presentation

- **Assessment:**
  - Lab sessions: 20% of grade
  - Oral exam:
    - With group in week 26 and 27
    - Counts for 80% of grade

Organization and assessment

- **Organization:**
  - Lab sessions: work in groups of 2 students and submit report
  - Project: work in groups of 5 students and give presentation

- **Assessment:**
  - Lab sessions: 20% of grade
  - Oral exam:
    - With group in week 26 and 27
    - Counts for 80% of grade
    - Each student assessed individually

Organization and assessment

- **Organization:**
  - Lab sessions: work in groups of 2 students and submit report
  - Project: work in groups of 5 students and give presentation

- **Assessment:**
  - Lab sessions: 20% of grade
  - Oral exam:
    - With group in week 26 and 27
    - Counts for 80% of grade
    - Each student assessed individually
    - (anonymous) peer review

Organization and assessment

- **Organization:**
  - Lab sessions: work in groups of 2 students and submit report
  - Project: work in groups of 5 students and give presentation

- **Assessment:**
  - Lab sessions: 20% of grade
  - Oral exam:
    - With group in week 26 and 27
    - Counts for 80% of grade
    - Each student assessed individually
    - (anonymous) peer review
  - To pass 5LSL0 $\rightarrow$ ORAL ≥ 5 AND LABS+ORAL $\geq$ 5.5

Course material

1. Book: "Statistical and Adaptive Array Signal Processing: Spectral estimation, signal modeling, adaptive filtering and array processing";
   Dimitris G. Manolakis, Vinay K. Ingle and Stephen M. Kogon;
   McGraw Hill; 2003
   - Optimum linear filters (Chapter 6)
   - Adaptive filters (Chapter 10)

Course material

1. Book: "Statistical and Adaptive Array Signal Processing: Spectral estimation, signal modeling, adaptive filtering and array processing";
   Dimitris G. Manolakis, Vinay K. Ingle and Stephen M. Kogon; McGraw Hill; 2003
   - Optimum linear filters (Chapter 6)
   - Adaptive filters (Chapter 10)
2. Book: "Deep learning";
   Ian Goodfellow, Yoshua Bengio and Aaron Courville; The MIT Press; 2016.
   - Machine learning basics (Chapter 5)
   - Deep feedforward networks (Chapter 6)
   - Regularization (Chapter 7)
   - Optimization for training deep models (Chapter 8)
   - Convolutional networks (Chapter 9)

Course material

3. These slides

Course material

3. These slides
4. Udacity

Course material

3. These slides
4. Udacity

Books available online at

https://lr.ttu.ee/~ameister/materjale/Dig_spe_anal/Statistical%20and%20Adaptive%20Signal%20Processing.pdf

and

https://www.deeplearningbook.org/

Global schedule

Week	Monday	Thursday
Monday: hour 3-4, Thursday: hour 7-8		
Apr 22	*No lecture*	Optimum linear filters
Apr 29	Adaptive filters	Adaptive filters
May 6	Activation functions	Optimization
May 13	Regularization	Regularization
May 20	CNN	CNN
May 27	Variational network	Variational network
Jun 3	Intro project	Project
Jun 10	Project	Project
Jun 17	Project	Project

Deadlines and credits

Code	Deadline	Credits
Ass 1	May 6, 10:30	5
Ass 2	May 16, 13:30	5
Ass 3	May 23, 13:30	5
Ass 4	June 3, 10:30	5
Oral	June 24- July 6	80
Total		100

Deadlines and credits

Code	Deadline	Credits
Ass 1	May 6, 10:30	5
Ass 2	May 16, 13:30	5
Ass 3	May 23, 13:30	5
Ass 4	June 3, 10:30	5
Oral	June 24- July 6	80
Total		100

To pass 5LSL0 $\Rightarrow$ ORAL ≥ 5 and TOTAL ≥ 5.5

Optimum linear filters and adaptive filters

Content

Focus on single channel adaptive algorithms using FIR structures

Technische Universiteit Eindhoven University of Technology

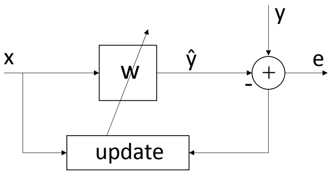
Content

Focus on single channel adaptive algorithms using FIR structures

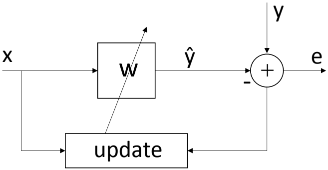
- Minimum Mean Squared Error
- Gradient Descent Algorithm
- Adaptive (N)LMS
- Newton algorithm
- Recursive Least Squares (RLS)

Technische Universiteit
Eindhoven
University of Technology

General Adaptive Filter model



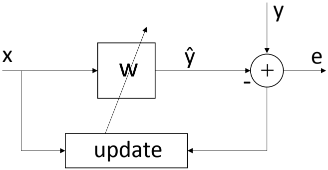
General Adaptive Filter model



Notes:

- Input signal x and desired response y correlated

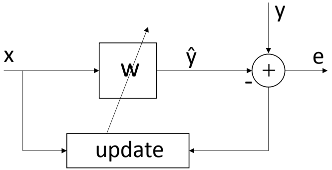
General Adaptive Filter model



Notes:

- Input signal x and desired response y correlated
- Pragmatic choices:
  - All signals have zero average
  - Filter w: FIR

General Adaptive Filter model



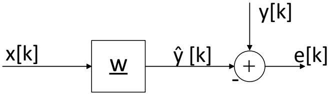
Notes:

- Input signal x and desired response y correlated
- Pragmatic choices:
  - All signals have zero average
  - Filter w: FIR
- Calculation of weight of filter w:
  - Use quadratic cost function: $J = f(e^2)$
  - First fixed weights (MMSE), then adaptive

Technische Universiteit Eindhoven University of Technology

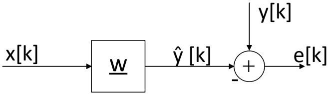
Fixed weights: MMSE

General Minimum Mean Squared Error (MMSE) model:



Fixed weights: MMSE

General Minimum Mean Squared Error (MMSE) model:

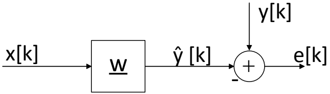


Goal:

Given N samples $\underline{\mathbf{x}}[k] = (x[k], x[k-1], \cdots, x[k-N+1])^t$ calculate coefficients fixed filter $\underline{\mathbf{w}} = (w_0, w_1, \cdots, w_{N-1})^t$ such that Mean Squared Error (MSE) $J = E\left\{e^2[k]\right\} = E\{(y[k] - \hat{y}[k])^2\}$ is minimized.

Fixed weights: MMSE

General Minimum Mean Squared Error (MMSE) model:



Goal:

Given N samples $\underline{\mathbf{x}}[k] = (x[k], x[k-1], \cdots, x[k-N+1])^t$ calculate coefficients fixed filter $\underline{\mathbf{w}} = (w_0, w_1, \cdots, w_{N-1})^t$ such that Mean Squared Error (MSE) $J = E\left\{e^2[k]\right\} = E\{(y[k] - \hat{y}[k])^2\}$ is minimized.

MMSE Optimization problem:

Given FIR samples $x[k-i]$ for $i = 0, 1, \cdots N-1$

$$\underline{\mathbf{w}}_o = \text{arg}\min_{\underline{\mathbf{w}}} \left(E\left\{e^2[k]\right\}\right)$$

Fixed weights: MMSE

$$
\begin{aligned}
J &= E\{(y[k] - \underline{\mathbf{w}}^t \cdot \underline{\mathbf{x}}[k]) \cdot (y[k] - \underline{\mathbf{x}}^t[k] \cdot \underline{\mathbf{w}})\} \\
&= E\{y^2[k]\} - \underline{\mathbf{w}}^t E\{\underline{\mathbf{x}}[k] y[k]\} - E\{y[k]\underline{\mathbf{x}}^t[k]\}\underline{\mathbf{w}} + \underline{\mathbf{w}}^t E\{\underline{\mathbf{x}}[k]\underline{\mathbf{x}}^t[k]\}\underline{\mathbf{w}}
\end{aligned}
$$

Fixed weights: MMSE

$$
\begin{aligned}
J &= E\{(y[k] - \underline{\mathbf{w}}^t \cdot \underline{\mathbf{x}}[k]) \cdot (y[k] - \underline{\mathbf{x}}^t[k] \cdot \underline{\mathbf{w}})\} \\
&= E\{y^2[k]\} - \underline{\mathbf{w}}^t E\{\underline{\mathbf{x}}[k] y[k]\} - E\{y[k] \underline{\mathbf{x}}^t[k]\} \underline{\mathbf{w}} + \underline{\mathbf{w}}^t E\{\underline{\mathbf{x}}[k] \underline{\mathbf{x}}^t[k]\} \underline{\mathbf{w}}
\end{aligned}
$$

$$
\Rightarrow \boxed{J = E\{y^2[k]\} - \underline{\mathbf{w}}^t \underline{\mathbf{r}}_{yx} - \underline{\mathbf{r}}_{yx}^t \underline{\mathbf{w}} + \underline{\mathbf{w}}^t \mathbf{R}_x \underline{\mathbf{w}}}
$$

Fixed weights: MMSE

$$
\begin{aligned}
J &= E\{(y[k] - \underline{\mathbf{w}}^t \cdot \underline{\mathbf{x}}[k]) \cdot (y[k] - \underline{\mathbf{x}}^t[k] \cdot \underline{\mathbf{w}})\} \\
&= E\{y^2[k]\} - \underline{\mathbf{w}}^t E\{\underline{\mathbf{x}}[k] y[k]\} - E\{y[k]\underline{\mathbf{x}}^t[k]\}\underline{\mathbf{w}} + \underline{\mathbf{w}}^t E\{\underline{\mathbf{x}}[k]\underline{\mathbf{x}}^t[k]\}\underline{\mathbf{w}}
\end{aligned}
$$

$$
\Rightarrow \boxed{J = E\{y^2[k]\} - \underline{\mathbf{w}}^t \underline{\mathbf{r}}_{yx} - \underline{\mathbf{r}}_{yx}^t \underline{\mathbf{w}} + \underline{\mathbf{w}}^t \mathbf{R}_x \underline{\mathbf{w}}}
$$

with cross correlation $\rho_{yx}[\tau] = E\{y[k]x[k-\tau]\}$:

$$
\underline{\mathbf{r}}_{yx} = E\{y[k]\underline{\mathbf{x}}[k]\} = (\rho_{yx}[0], \rho_{yx}[1], \cdots, \rho_{yx}[N-1])^t
$$

Fixed weights: MMSE

$$
\begin{aligned}
J &= E\{(y[k] - \underline{\mathbf{w}}^t \cdot \underline{\mathbf{x}}[k]) \cdot (y[k] - \underline{\mathbf{x}}^t[k] \cdot \underline{\mathbf{w}})\} \\
&= E\{y^2[k]\} - \underline{\mathbf{w}}^t E\{\underline{\mathbf{x}}[k]y[k]\} - E\{y[k]\underline{\mathbf{x}}^t[k]\}\underline{\mathbf{w}} + \underline{\mathbf{w}}^t E\{\underline{\mathbf{x}}[k]\underline{\mathbf{x}}^t[k]\}\underline{\mathbf{w}}
\end{aligned}
$$

$$
\Rightarrow \boxed{J = E\{y^2[k]\} - \underline{\mathbf{w}}^t \underline{\mathbf{r}}_{yx} - \underline{\mathbf{r}}_{yx}^t \underline{\mathbf{w}} + \underline{\mathbf{w}}^t \mathbf{R}_x \underline{\mathbf{w}}}
$$

with cross correlation $\rho_{yx}[\tau] = E\{y[k]x[k-\tau]\}$:

$$
\underline{\mathbf{r}}_{yx} = E\{y[k]\underline{\mathbf{x}}[k]\} = (\rho_{yx}[0], \rho_{yx}[1], \cdots, \rho_{yx}[N-1])^t
$$

and autocorrelation: $\rho_x[\tau] = E\{x[k]x[k-\tau]\} = \rho_x[-\tau]$

$$
\mathbf{R}_x = E\{\underline{\mathbf{x}}[k]\underline{\mathbf{x}}^t[k]\} = \begin{pmatrix} \rho_x[0] & \rho_x[1] & \cdots & \rho_x[N-1] \\ \rho_x[1] & \rho_x[0] & \cdots & \rho_x[N-2] \\ \vdots & \vdots & \vdots & \vdots \\ \rho_x[N-1] & \rho_x[N-2] & \cdots & \rho_x[0] \end{pmatrix}
$$

TU/e Technische Universiteit Eindhoven University of Technology

Fixed weights: MMSE

$$J = E\{y^2[k]\} - \underline{\mathbf{w}}^t \underline{\mathbf{r}}_{yx} - \underline{\mathbf{r}}_{yx}^t \underline{\mathbf{w}} + \underline{\mathbf{w}}^t \mathbf{R}_x \underline{\mathbf{w}}$$

Fixed weights: MMSE

$$J = E\{y^2[k]\} - \underline{\mathbf{w}}^t \underline{\mathbf{r}}_{yx} - \underline{\mathbf{r}}_{yx}^t \underline{\mathbf{w}} + \underline{\mathbf{w}}^t \mathbf{R}_x \underline{\mathbf{w}}$$

$$\Rightarrow \text{ Optimum: } \underline{\nabla} = \frac{\mathrm{d}J}{\mathrm{d}\underline{\mathbf{w}}} = -2(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}) = \underline{\mathbf{0}}$$

Fixed weights: MMSE

$$J = E\{y^2[k]\} - \underline{\mathbf{w}}^t \underline{\mathbf{r}}_{yx} - \underline{\mathbf{r}}_{yx}^t \underline{\mathbf{w}} + \underline{\mathbf{w}}^t \mathbf{R}_x \underline{\mathbf{w}}$$

$$\Rightarrow \text{ Optimum: } \underline{\nabla} = \frac{\mathrm{d}J}{\mathrm{d}\underline{\mathbf{w}}} = -2(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}) = \underline{\mathbf{0}}$$

$\Rightarrow$ **Normal Equations** $\boxed{\mathbf{R}_x \cdot \underline{\mathbf{w}} = \underline{\mathbf{r}}_{yx}}$

Fixed weights: MMSE

$$J = E\{y^2[k]\} - \underline{\mathbf{w}}^t \underline{\mathbf{r}}_{yx} - \underline{\mathbf{r}}_{yx}^t \underline{\mathbf{w}} + \underline{\mathbf{w}}^t \mathbf{R}_x \underline{\mathbf{w}}$$

$$\Rightarrow \text{ Optimum: } \underline{\nabla} = \frac{\mathrm{d}J}{\mathrm{d}\underline{\mathbf{w}}} = -2(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}) = \underline{\mathbf{0}}$$

$\Rightarrow$ **Normal Equations** $\boxed{\mathbf{R}_x \cdot \underline{\mathbf{w}} = \underline{\mathbf{r}}_{yx}}$

$\Rightarrow$ **Wiener filter** $\boxed{\underline{\mathbf{w}}_o = \mathbf{R}_x^{-1} \cdot \underline{\mathbf{r}}_{yx}}$

Fixed weights: MMSE

$$J = E\{y^2[k]\} - \underline{\mathbf{w}}^t \underline{\mathbf{r}}_{yx} - \underline{\mathbf{r}}_{yx}^t \underline{\mathbf{w}} + \underline{\mathbf{w}}^t \mathbf{R}_x \underline{\mathbf{w}}$$

$\Rightarrow$ Optimum: $\underline{\nabla} = \dfrac{\mathrm{d}J}{\mathrm{d}\underline{\mathbf{w}}} = -2(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}) = \underline{\mathbf{0}}$

$\Rightarrow$ Normal Equations $\boxed{\mathbf{R}_x \cdot \underline{\mathbf{w}} = \underline{\mathbf{r}}_{yx}}$

$\Rightarrow$ Wiener filter $\boxed{\underline{\mathbf{w}}_o = \mathbf{R}_x^{-1} \cdot \underline{\mathbf{r}}_{yx}}$

General expression: $\boxed{J = J_{min} + (\underline{\mathbf{w}} - \underline{\mathbf{w}}_o)^t \cdot \mathbf{R}_x \cdot (\underline{\mathbf{w}} - \underline{\mathbf{w}}_o)}$

Fixed weights: MMSE

$$J = E\{y^2[k]\} - \underline{\mathbf{w}}^t \underline{\mathbf{r}}_{yx} - \underline{\mathbf{r}}_{yx}^t \underline{\mathbf{w}} + \underline{\mathbf{w}}^t \mathbf{R}_x \underline{\mathbf{w}}$$

$$\Rightarrow \text{ Optimum: } \underline{\nabla} = \frac{\mathrm{d}J}{\mathrm{d}\underline{\mathbf{w}}} = -2(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}) = \underline{\mathbf{0}}$$

$\Rightarrow$ **Normal Equations** $\boxed{\mathbf{R}_x \cdot \underline{\mathbf{w}} = \underline{\mathbf{r}}_{yx}}$

$\Rightarrow$ **Wiener filter** $\boxed{\underline{\mathbf{w}}_o = \mathbf{R}_x^{-1} \cdot \underline{\mathbf{r}}_{yx}}$

General expression: $\boxed{J = J_{min} + (\underline{\mathbf{w}} - \underline{\mathbf{w}}_o)^t \cdot \mathbf{R}_x \cdot (\underline{\mathbf{w}} - \underline{\mathbf{w}}_o)}$

$$J_{min} = J_{\underline{\mathbf{w}}=\underline{\mathbf{w}}_o} = E\{e^2[k]\} = E\{y^2[k]\} - \underline{\mathbf{r}}_{yx}^t \mathbf{R}_x^{-1} \underline{\mathbf{r}}_{yx}$$

Fixed weights: MMSE

$$J = E\{y^2[k]\} - \underline{\mathbf{w}}^t \underline{\mathbf{r}}_{yx} - \underline{\mathbf{r}}_{yx}^t \underline{\mathbf{w}} + \underline{\mathbf{w}}^t \mathbf{R}_x \underline{\mathbf{w}}$$

$\Rightarrow$ Optimum: $\underline{\nabla} = \dfrac{\mathrm{d}J}{\mathrm{d}\underline{\mathbf{w}}} = -2(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}) = \underline{\mathbf{0}}$

$\Rightarrow$ Normal Equations $\boxed{\mathbf{R}_x \cdot \underline{\mathbf{w}} = \underline{\mathbf{r}}_{yx}}$

$\Rightarrow$ Wiener filter $\boxed{\underline{\mathbf{w}}_o = \mathbf{R}_x^{-1} \cdot \underline{\mathbf{r}}_{yx}}$

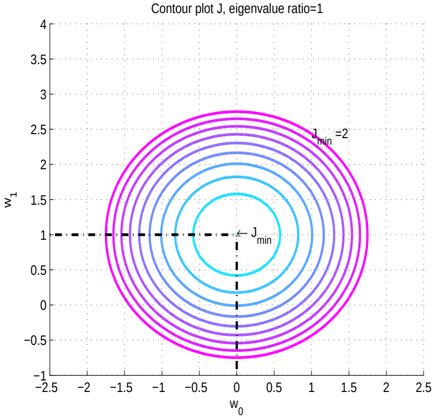
General expression: $\boxed{J = J_{min} + (\underline{\mathbf{w}} - \underline{\mathbf{w}}_o)^t \cdot \mathbf{R}_x \cdot (\underline{\mathbf{w}} - \underline{\mathbf{w}}_o)}$

$$J_{min} = J_{\underline{\mathbf{w}}=\underline{\mathbf{w}}_o} = E\{e^2[k]\} = E\{y^2[k]\} - \underline{\mathbf{r}}_{yx}^t \mathbf{R}_x^{-1} \underline{\mathbf{r}}_{yx}$$

From general expression $\Rightarrow J$ quadratic in $\underline{\mathbf{w}}$ thus $\underline{\mathbf{w}}_o$ really minimum

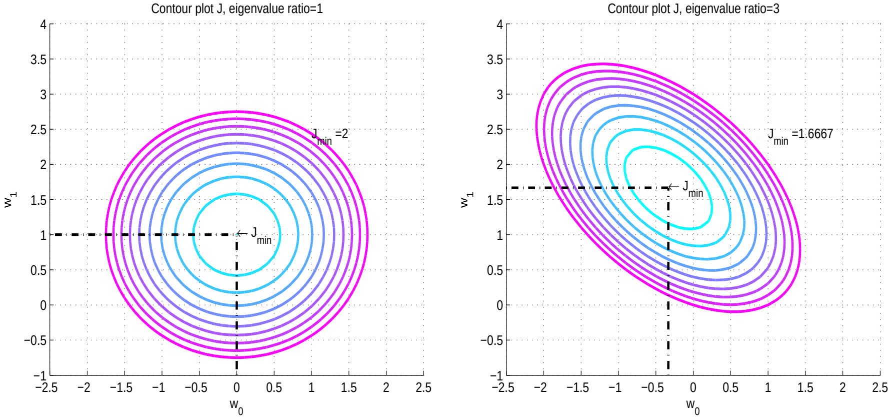
Fixed weights: MMSE

Contour plots $J = J_{min} + (\underline{\mathbf{w}} - \underline{\mathbf{w}}_o)^t \cdot \mathbf{R}_x \cdot (\underline{\mathbf{w}} - \underline{\mathbf{w}}_o)$



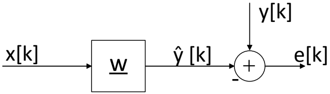
Fixed weights: MMSE

Contour plots $J = J_{min} + (\underline{\mathbf{w}} - \underline{\mathbf{w}}_o)^t \cdot \mathbf{R}_x \cdot (\underline{\mathbf{w}} - \underline{\mathbf{w}}_o)$

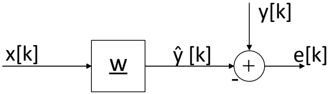


Eigenvalues: see Appendix

Least Squares (LS)



Least Squares (LS)



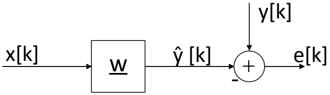
Different quadratic cost functions:

- Mean Square Error (MSE):

$$J_{mse} = E\{e^2[k]\} = E\{(y[k] - \underline{\mathbf{w}}^t \underline{\mathbf{x}}[k])^2\}$$

$\Rightarrow$ Minimum MSE (MMSE) = Wiener

Least Squares (LS)



Different quadratic cost functions:

- Mean Square Error (MSE):

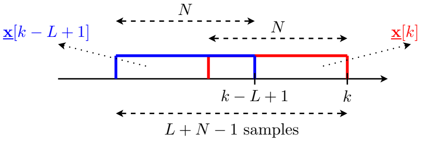
$$J_{mse} = E\{e^2[k]\} = E\{(y[k] - \underline{\mathbf{w}}^t \underline{\mathbf{x}}[k])^2\}$$

$\Rightarrow$ Minimum MSE (MMSE) = Wiener

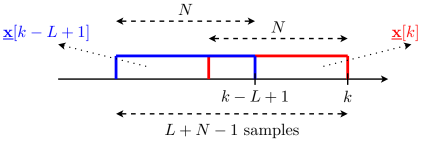
- **Least Square (LS):** If statistical information is not available $\Rightarrow$

  Use criterion based on data (thus without $E\{\cdot\}$)

Collect L (≥ 1) data vectors $\underline{\mathbf{x}}[k-i]$ (each of length N)

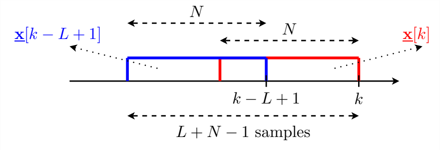


Collect L (≥ 1) data vectors $\underline{\mathbf{x}}[k-i]$ (each of length N)



Available data (for $i = 0, 1, \cdots, L-1$):

Collect $L\ (\geq 1)$ data vectors $\underline{\mathbf{x}}[k-i]$ (each of length N)



Available data (for $i = 0, 1, \cdots, L-1$):

- *Input signal samples/ vectors* $\underline{\mathbf{x}}[k-i]$

$$\underline{\mathbf{x}}[k-i] = (x[k-i], x[k-i-1], \cdots, x[k-i-N+1])^t$$

- *Reference signal samples:* $y[k-i]$

- *Residual signal samples:* $e[k-i] = y[k-i] - \underline{\mathbf{x}}^t[k-i] \cdot \underline{\mathbf{w}}$

Notation:

$$\underline{\mathbf{X}}[k] = \begin{pmatrix} \underline{\mathbf{x}}^t[k] \\ \underline{\mathbf{x}}^t[k-1] \\ \vdots \\ \underline{\mathbf{x}}^t[k-L+1] \end{pmatrix} \qquad \underline{\mathbf{w}} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{N-1} \end{pmatrix}$$

$$\underline{\mathbf{y}}[k] = \begin{pmatrix} y[k] \\ y[k-1] \\ \vdots \\ y[k-L+1] \end{pmatrix} \qquad \underline{\mathbf{e}}[k] = \begin{pmatrix} e[k] \\ e[k-1] \\ \vdots \\ e[k-L+1] \end{pmatrix}$$

Notation:

$$\mathbf{X}[k] = \begin{pmatrix} \underline{\mathbf{x}}^t[k] \\ \underline{\mathbf{x}}^t[k-1] \\ \vdots \\ \underline{\mathbf{x}}^t[k-L+1] \end{pmatrix} \qquad \underline{\mathbf{w}} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{N-1} \end{pmatrix}$$

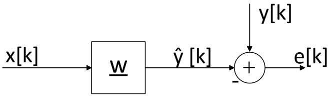
$$\underline{\mathbf{y}}[k] = \begin{pmatrix} y[k] \\ y[k-1] \\ \vdots \\ y[k-L+1] \end{pmatrix} \qquad \underline{\mathbf{e}}[k] = \begin{pmatrix} e[k] \\ e[k-1] \\ \vdots \\ e[k-L+1] \end{pmatrix}$$

Simplified notation (skip time indices):

$$\underline{\mathbf{e}} = \underline{\mathbf{y}} - \mathbf{X} \cdot \underline{\mathbf{w}}$$

Technische Universiteit Eindhoven University of Technology

LS



LS problem formulation:

$$\boxed{\underline{\mathbf{w}}_{ls,o} = \arg\min_{\underline{\mathbf{w}}} \left|\underline{\mathbf{y}} - \mathbf{X}\cdot\underline{\mathbf{w}}\right|^2}$$

LS

$$J_{ls} = \sum_{i=0}^{L-1} e^2[k-i] = \underline{\mathbf{e}}^t \cdot \underline{\mathbf{e}} = (\underline{\mathbf{y}}^t - \underline{\mathbf{w}}^t \mathbf{X}^t) \cdot (\underline{\mathbf{y}} - \mathbf{X}\underline{\mathbf{w}})$$

$$
\begin{aligned}
J_{ls} &= \sum_{i=0}^{L-1} e^2[k-i] = \underline{\mathbf{e}}^t \cdot \underline{\mathbf{e}} = (\underline{\mathbf{y}}^t - \underline{\mathbf{w}}^t \mathbf{X}^t) \cdot (\underline{\mathbf{y}} - \mathbf{X}\underline{\mathbf{w}}) \\
&= \underline{\mathbf{y}}^t \underline{\mathbf{y}} + \underline{\mathbf{w}}^t \mathbf{X}^t \mathbf{X} \underline{\mathbf{w}} - \underline{\mathbf{w}}^t \mathbf{X}^t \underline{\mathbf{y}} - \underline{\mathbf{y}}^t \mathbf{X} \underline{\mathbf{w}}
\end{aligned}
$$

$$
\begin{aligned}
J_{ls} &= \sum_{i=0}^{L-1} e^2[k-i] = \underline{\mathbf{e}}^t \cdot \underline{\mathbf{e}} = (\underline{\mathbf{y}}^t - \underline{\mathbf{w}}^t \mathbf{X}^t) \cdot (\underline{\mathbf{y}} - \mathbf{X}\underline{\mathbf{w}}) \\
&= \underline{\mathbf{y}}^t \underline{\mathbf{y}} + \underline{\mathbf{w}}^t \mathbf{X}^t \mathbf{X} \underline{\mathbf{w}} - \underline{\mathbf{w}}^t \mathbf{X}^t \underline{\mathbf{y}} - \underline{\mathbf{y}}^t \mathbf{X} \underline{\mathbf{w}}
\end{aligned}
$$

Minimum by setting gradient equal to zero:

$$
\frac{\mathrm{d}J_{ls}}{\mathrm{d}\underline{\mathbf{w}}} = \underline{\triangledown}_{ls} = -2(\mathbf{X}^t \underline{\mathbf{y}} - \mathbf{X}^t \mathbf{X} \cdot \underline{\mathbf{w}}) = \underline{\mathbf{0}}
$$

$$
\begin{aligned}
J_{ls} &= \sum_{i=0}^{L-1} e^2[k-i] = \underline{\mathbf{e}}^t \cdot \underline{\mathbf{e}} = (\underline{\mathbf{y}}^t - \underline{\mathbf{w}}^t \mathbf{X}^t) \cdot (\underline{\mathbf{y}} - \mathbf{X}\underline{\mathbf{w}}) \\
&= \underline{\mathbf{y}}^t \underline{\mathbf{y}} + \underline{\mathbf{w}}^t \mathbf{X}^t \mathbf{X} \underline{\mathbf{w}} - \underline{\mathbf{w}}^t \mathbf{X}^t \underline{\mathbf{y}} - \underline{\mathbf{y}}^t \mathbf{X} \underline{\mathbf{w}}
\end{aligned}
$$

Minimum by setting gradient equal to zero:

$$
\frac{\mathrm{d}J_{ls}}{\mathrm{d}\underline{\mathbf{w}}} = \underline{\bigtriangledown}_{ls} = -2(\mathbf{X}^t \underline{\mathbf{y}} - \mathbf{X}^t \mathbf{X} \cdot \underline{\mathbf{w}}) = \underline{\mathbf{0}}
$$

With $\overline{\mathbf{R}} = \mathbf{X}^t \mathbf{X}$ and $\underline{\mathbf{r}} = \mathbf{X}^t \underline{\mathbf{y}}$

$$
\begin{aligned}
J_{ls} &= \sum_{i=0}^{L-1} e^2[k-i] = \underline{\mathbf{e}}^t \cdot \underline{\mathbf{e}} = (\underline{\mathbf{y}}^t - \underline{\mathbf{w}}^t \mathbf{X}^t) \cdot (\underline{\mathbf{y}} - \mathbf{X}\underline{\mathbf{w}}) \\
&= \underline{\mathbf{y}}^t \underline{\mathbf{y}} + \underline{\mathbf{w}}^t \mathbf{X}^t \mathbf{X} \underline{\mathbf{w}} - \underline{\mathbf{w}}^t \mathbf{X}^t \underline{\mathbf{y}} - \underline{\mathbf{y}}^t \mathbf{X} \underline{\mathbf{w}}
\end{aligned}
$$

Minimum by setting gradient equal to zero:

$$
\frac{\mathrm{d}J_{ls}}{\mathrm{d}\underline{\mathbf{w}}} = \underline{\nabla}_{ls} = -2(\mathbf{X}^t \underline{\mathbf{y}} - \mathbf{X}^t \mathbf{X} \cdot \underline{\mathbf{w}}) = \underline{\mathbf{0}}
$$

With $\overline{\mathbf{R}} = \mathbf{X}^t \mathbf{X}$ and $\underline{\overline{\mathbf{r}}} = \mathbf{X}^t \underline{\mathbf{y}}$

$\Rightarrow$ Normal Equations $\boxed{\overline{\mathbf{R}}_x \cdot \underline{\mathbf{w}} = \underline{\overline{\mathbf{r}}}_{yx}}$

$\Rightarrow$ Wiener filter $\boxed{\underline{\mathbf{w}}_{ls,o} = \overline{\mathbf{R}}_x^{-1} \cdot \underline{\overline{\mathbf{r}}}_{yx}}$

LS: Correspondence with MMSE

Use time-averaging (ergodicity):

$$\begin{aligned}
\hat{\mathbf{R}}_x &= \frac{1}{L}\sum_{i=0}^{L-1} \underline{\mathbf{x}}[k-i]\cdot\underline{\mathbf{x}}^t[k-i] = \frac{1}{L}\mathbf{X}^t\cdot\mathbf{X} = \frac{1}{L}\overline{\mathbf{R}}_x \\
\underline{\hat{\mathbf{r}}}_{yx} &= \frac{1}{L}\sum_{i=0}^{L-1} \underline{\mathbf{x}}[k-i]\cdot y[k-i] = \frac{1}{L}\mathbf{X}^t\cdot\underline{\mathbf{y}} = \frac{1}{L}\underline{\overline{\mathbf{r}}}_{yx}
\end{aligned}$$

TU/e Technische Universiteit Eindhoven University of Technology

LS: Correspondence with MMSE

Use time-averaging (ergodicity):

$$\hat{\mathbf{R}}_x = \frac{1}{L}\sum_{i=0}^{L-1} \underline{\mathbf{x}}[k-i]\cdot\underline{\mathbf{x}}^t[k-i] = \frac{1}{L}\mathbf{X}^t\cdot\mathbf{X} = \frac{1}{L}\overline{\mathbf{R}}_x$$

$$\hat{\underline{\mathbf{r}}}_{yx} = \frac{1}{L}\sum_{i=0}^{L-1} \underline{\mathbf{x}}[k-i]\cdot y[k-i] = \frac{1}{L}\mathbf{X}^t\cdot\underline{\mathbf{y}} = \frac{1}{L}\overline{\underline{\mathbf{r}}}_{yx}$$

with $\hat{\mathbf{R}}_x$ estimate of $\mathbf{R}_x$ and $\hat{\underline{\mathbf{r}}}_{yx}$ estimate of $\underline{\mathbf{r}}_{yx}$

$$\Rightarrow \quad \hat{\underline{\mathbf{w}}}_{mmse} = \left(\frac{1}{L}\overline{\mathbf{R}}_x\right)^{-1}\cdot\left(\frac{1}{L}\overline{\underline{\mathbf{r}}}_{yx}\right) = \overline{\mathbf{R}}_x^{-1}\cdot\overline{\underline{\mathbf{r}}}_{yx} = \underline{\mathbf{w}}_{ls}$$

LS: Correspondence with MMSE

Use time-averaging (ergodicity):

$$\hat{\mathbf{R}}_x = \frac{1}{L}\sum_{i=0}^{L-1} \underline{\mathbf{x}}[k-i]\cdot\underline{\mathbf{x}}^t[k-i] = \frac{1}{L}\mathbf{X}^t\cdot\mathbf{X} = \frac{1}{L}\overline{\mathbf{R}}_x$$

$$\hat{\underline{\mathbf{r}}}_{yx} = \frac{1}{L}\sum_{i=0}^{L-1} \underline{\mathbf{x}}[k-i]\cdot y[k-i] = \frac{1}{L}\mathbf{X}^t\cdot\underline{\mathbf{y}} = \frac{1}{L}\overline{\underline{\mathbf{r}}}_{yx}$$

with $\hat{\mathbf{R}}_x$ estimate of $\mathbf{R}_x$ and $\hat{\underline{\mathbf{r}}}_{yx}$ estimate of $\underline{\mathbf{r}}_{yx}$

$$\Rightarrow \quad \hat{\underline{\mathbf{w}}}_{mmse} = \left(\frac{1}{L}\overline{\mathbf{R}}_x\right)^{-1}\cdot\left(\frac{1}{L}\overline{\underline{\mathbf{r}}}_{yx}\right) = \overline{\mathbf{R}}_x^{-1}\cdot\overline{\underline{\mathbf{r}}}_{yx} = \underline{\mathbf{w}}_{ls}$$

Finally note that for ergodic processes:

$$\lim_{L\to\infty}\frac{1}{L}\overline{\mathbf{R}}_x = \mathbf{R}_x\ ;\ \lim_{L\to\infty}\frac{1}{L}\overline{\underline{\mathbf{r}}}_{yx} = \underline{\mathbf{r}}_{yx}\ ;\ \lim_{L\to\infty}\underline{\mathbf{w}}_{ls} = \underline{\mathbf{w}}_{mmse}$$

Steepest gradient descent (GD)

Problem: Optimal Wiener involves $\mathbf{R}_x^{-1}$

Steepest gradient descent (GD)

Problem: Optimal Wiener involves $\mathbf{R}_x^{-1}$

To avoid this inversion, estimate optimum *iteratively*

Steepest gradient descent (GD)

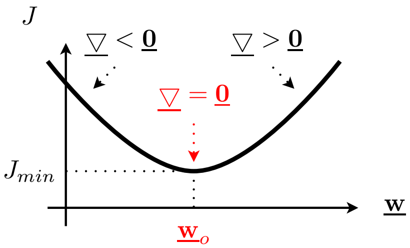
Problem: Optimal Wiener involves $\mathbf{R}_x^{-1}$

To avoid this inversion, estimate optimum *iteratively*

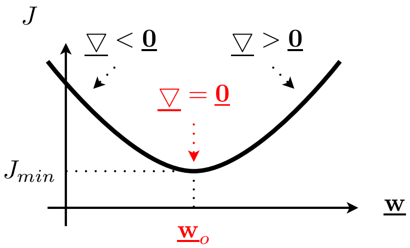
Goal: Decrease J each new iteration

Technische Universiteit
Eindhoven
University of Technology

GD

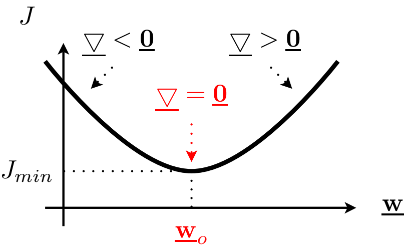

J
$\underline{\nabla} < \underline{\mathbf{0}}$
$\underline{\nabla} > \underline{\mathbf{0}}$
$\underline{\nabla} = \underline{\mathbf{0}}$
J_{min}
$\underline{\mathbf{w}}$
$\underline{\mathbf{w}}_o$

GD



GD principle: Update in negative gradient direction

$\Leftrightarrow \underline{\mathbf{w}} \doteq \underline{\mathbf{w}} - \alpha\underline{\nabla}$ with adaptation constant $\alpha \geq 0$

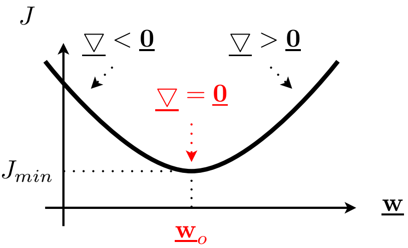


GD principle: Update in negative gradient direction

$\Leftrightarrow \underline{\mathbf{w}} \doteq \underline{\mathbf{w}} - \alpha \underline{\nabla}$ with adaptation constant $\alpha \geq 0$

With $\underline{\nabla} = -2(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}[k])$

GD

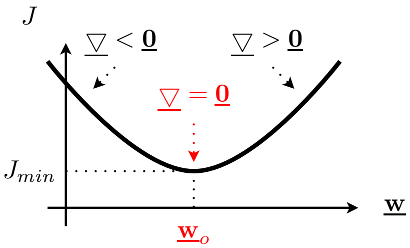


GD principle: Update in negative gradient direction

$\Leftrightarrow \underline{\mathbf{w}} \doteq \underline{\mathbf{w}} - \alpha \underline{\nabla}$ with adaptation constant $\alpha \geq 0$

With $\underline{\nabla} = -2(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}[k]) \Rightarrow$ GD algorithm:

$$\boxed{\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}[k])}$$



GD principle: Update in negative gradient direction

$\Leftrightarrow \underline{\mathbf{w}} \doteq \underline{\mathbf{w}} - \alpha \underline{\nabla}$ with adaptation constant $\alpha \geq 0$

With $\underline{\nabla} = -2(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}[k]) \Rightarrow$ **GD algorithm:**

$$\boxed{\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}[k])}$$

Notes: 1) No matrix inversion needed! 2) Usually $\underline{\mathbf{w}}[0] = \underline{\mathbf{0}}$

GD

GD converges to Wiener solution:

$$\boxed{\lim_{k \to \infty} \underline{\mathbf{w}}[k] \simeq \mathbf{R}_x^{-1} \cdot \underline{\mathbf{r}}_{yx}}$$

Technische Universiteit Eindhoven University of Technology

GD converges to Wiener solution:

$$\boxed{\lim_{k\to\infty} \underline{\mathbf{w}}[k] \simeq \mathbf{R}_x^{-1} \cdot \underline{\mathbf{r}}_{yx}}$$

’Proof’:

For $k \to \infty$ we have:

$$\underline{\mathbf{w}}[k+1] \simeq \underline{\mathbf{w}}[k] \simeq \underline{\mathbf{w}}[\infty]$$

GD

GD converges to Wiener solution:

$$\boxed{\lim_{k\to\infty} \underline{\mathbf{w}}[k] \simeq \mathbf{R}_x^{-1} \cdot \underline{\mathbf{r}}_{yx}}$$

'Proof':

For $k \to \infty$ we have:

$$\begin{array}{rcl} & & \underline{\mathbf{w}}[k+1] \simeq \underline{\mathbf{w}}[k] \simeq \underline{\mathbf{w}}[\infty] \\ \text{GD} & \Rightarrow & \underline{\mathbf{w}}[\infty] \simeq \underline{\mathbf{w}}[\infty] + 2\alpha(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}[\infty]) \end{array}$$

GD converges to Wiener solution:

$$\boxed{\lim_{k\to\infty} \underline{\mathbf{w}}[k] \simeq \mathbf{R}_x^{-1} \cdot \underline{\mathbf{r}}_{yx}}$$

'Proof':

For $k \to \infty$ we have:

$$\begin{aligned} & \underline{\mathbf{w}}[k+1] \simeq \underline{\mathbf{w}}[k] \simeq \underline{\mathbf{w}}[\infty] \\ \text{GD} \quad \Rightarrow \quad & \underline{\mathbf{w}}[\infty] \simeq \underline{\mathbf{w}}[\infty] + 2\alpha(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}[\infty]) \\ \Rightarrow \quad & \underline{\mathbf{w}}[\infty] \simeq \mathbf{R}_x^{-1} \cdot \underline{\mathbf{r}}_{yx} \end{aligned}$$

GD

GD converges to Wiener solution:

$$\boxed{\lim_{k\to\infty} \underline{\mathbf{w}}[k] \simeq \mathbf{R}_x^{-1} \cdot \underline{\mathbf{r}}_{yx}}$$

'Proof':

For $k \to \infty$ we have:

$$\begin{aligned} & \underline{\mathbf{w}}[k+1] \simeq \underline{\mathbf{w}}[k] \simeq \underline{\mathbf{w}}[\infty] \\ \text{GD} \quad \Rightarrow \quad & \underline{\mathbf{w}}[\infty] \simeq \underline{\mathbf{w}}[\infty] + 2\alpha(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}[\infty]) \\ \Rightarrow \quad & \underline{\mathbf{w}}[\infty] \simeq \mathbf{R}_x^{-1} \cdot \underline{\mathbf{r}}_{yx} \end{aligned}$$

For exact proof we need **stability analysis**

Stability GD

Define difference weight vector: $\underline{\mathbf{d}}[k] = \underline{\mathbf{w}}[k] - \underline{\mathbf{w}}_o$

Stability GD

Define difference weight vector: $\underline{\mathbf{d}}[k] = \underline{\mathbf{w}}[k] - \underline{\mathbf{w}}_o$

$$\underline{\mathbf{w}}[k+1] \quad = \quad \underline{\mathbf{w}}[k] + 2\alpha(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}[k])$$

Stability GD

Define difference weight vector: $\underline{\mathbf{d}}[k] = \underline{\mathbf{w}}[k] - \underline{\mathbf{w}}_o$

$$\begin{aligned} \underline{\mathbf{w}}[k+1] &= \underline{\mathbf{w}}[k] + 2\alpha(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}[k]) \\ \underline{\mathbf{w}}[k+1] - \underline{\mathbf{w}}_o &= (\mathbf{I} - 2\alpha\mathbf{R}_x) \cdot \underline{\mathbf{w}}[k] - \underline{\mathbf{w}}_o + 2\alpha\underline{\mathbf{r}}_{yx} \end{aligned}$$

Stability GD

Define difference weight vector: $\underline{\mathbf{d}}[k] = \underline{\mathbf{w}}[k] - \underline{\mathbf{w}}_o$

$$\begin{aligned}
\underline{\mathbf{w}}[k+1] &= \underline{\mathbf{w}}[k] + 2\alpha(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x\underline{\mathbf{w}}[k]) \\
\underline{\mathbf{w}}[k+1] - \underline{\mathbf{w}}_o &= (\mathbf{I} - 2\alpha\mathbf{R}_x)\cdot\underline{\mathbf{w}}[k] - \underline{\mathbf{w}}_o + 2\alpha\underline{\mathbf{r}}_{yx} \\
\Rightarrow\ \underline{\mathbf{d}}[k+1] &= (\mathbf{I} - 2\alpha\mathbf{R}_x)\cdot\underline{\mathbf{d}}[k]
\end{aligned}$$

Stability GD

Define difference weight vector: $\underline{\mathbf{d}}[k] = \underline{\mathbf{w}}[k] - \underline{\mathbf{w}}_o$

$$\begin{aligned}
\underline{\mathbf{w}}[k+1] &= \underline{\mathbf{w}}[k] + 2\alpha(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x\underline{\mathbf{w}}[k]) \\
\underline{\mathbf{w}}[k+1] - \underline{\mathbf{w}}_o &= (\mathbf{I} - 2\alpha\mathbf{R}_x)\cdot\underline{\mathbf{w}}[k] - \underline{\mathbf{w}}_o + 2\alpha\underline{\mathbf{r}}_{yx} \\
\Rightarrow\ \underline{\mathbf{d}}[k+1] &= (\mathbf{I} - 2\alpha\mathbf{R}_x)\cdot\underline{\mathbf{d}}[k]
\end{aligned}$$

Recursion:

$$\underline{\mathbf{d}}[k] = (\mathbf{I} - 2\alpha\mathbf{R}_x)\cdot\underline{\mathbf{d}}[k-1] = \cdots = (\mathbf{I} - 2\alpha\mathbf{R}_x)^k\cdot\underline{\mathbf{d}}[0]$$

Stability GD

Define difference weight vector: $\underline{\mathbf{d}}[k] = \underline{\mathbf{w}}[k] - \underline{\mathbf{w}}_o$

$$\begin{aligned}
\underline{\mathbf{w}}[k+1] &= \underline{\mathbf{w}}[k] + 2\alpha(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x\underline{\mathbf{w}}[k]) \\
\underline{\mathbf{w}}[k+1] - \underline{\mathbf{w}}_o &= (\mathbf{I} - 2\alpha\mathbf{R}_x)\cdot\underline{\mathbf{w}}[k] - \underline{\mathbf{w}}_o + 2\alpha\underline{\mathbf{r}}_{yx} \\
\Rightarrow\ \underline{\mathbf{d}}[k+1] &= (\mathbf{I} - 2\alpha\mathbf{R}_x)\cdot\underline{\mathbf{d}}[k]
\end{aligned}$$

Recursion:

$$\underline{\mathbf{d}}[k] = (\mathbf{I} - 2\alpha\mathbf{R}_x)\cdot\underline{\mathbf{d}}[k-1] = \cdots = (\mathbf{I} - 2\alpha\mathbf{R}_x)^k\cdot\underline{\mathbf{d}}[0]$$

Converges if: $\lim_{k\to\infty}(\mathbf{I} - 2\alpha\mathbf{R}_x)^k = \mathbf{0}$

Stability GD

Define difference weight vector: $\underline{\mathbf{d}}[k] = \underline{\mathbf{w}}[k] - \underline{\mathbf{w}}_o$

$$\begin{aligned} \underline{\mathbf{w}}[k+1] &= \underline{\mathbf{w}}[k] + 2\alpha(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}[k]) \\ \underline{\mathbf{w}}[k+1] - \underline{\mathbf{w}}_o &= (\mathbf{I} - 2\alpha\mathbf{R}_x) \cdot \underline{\mathbf{w}}[k] - \underline{\mathbf{w}}_o + 2\alpha\underline{\mathbf{r}}_{yx} \\ \Rightarrow \underline{\mathbf{d}}[k+1] &= (\mathbf{I} - 2\alpha\mathbf{R}_x) \cdot \underline{\mathbf{d}}[k] \end{aligned}$$

Recursion:

$$\underline{\mathbf{d}}[k] = (\mathbf{I} - 2\alpha\mathbf{R}_x) \cdot \underline{\mathbf{d}}[k-1] = \cdots = (\mathbf{I} - 2\alpha\mathbf{R}_x)^k \cdot \underline{\mathbf{d}}[0]$$

Converges if: $\lim_{k\to\infty} (\mathbf{I} - 2\alpha\mathbf{R}_x)^k = \mathbf{0}$

Note:

When stable $\Rightarrow \underline{\mathbf{d}}[\infty] = \underline{\mathbf{0}} \Rightarrow \underline{\mathbf{w}}[\infty] \simeq$ Wiener

Stability GD

How do weights converge:

<u>How</u> do weights converge:

Use eigenvalue decomposition (see Appendix):

How do weights converge:

Use eigenvalue decomposition (see Appendix):

With $\mathbf{Q}^h \cdot \mathbf{Q} = \mathbf{Q} \cdot \mathbf{Q}^h = \mathbf{I}$ and $\mathbf{R}_x = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^h$

Stability GD

<u>How</u> **do weights converge:**

Use eigenvalue decomposition (see Appendix):

With $\mathbf{Q}^h \cdot \mathbf{Q} = \mathbf{Q} \cdot \mathbf{Q}^h = \mathbf{I}$ and $\mathbf{R}_x = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^h$

$$\begin{aligned} \Rightarrow \ (\mathbf{I} - 2\alpha\mathbf{R}_x)^k &= (\mathbf{Q}\mathbf{Q}^h - 2\alpha\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^h)^k \\ &= \mathbf{Q}(\mathbf{I} - 2\alpha\mathbf{\Lambda})^k\mathbf{Q}^h \end{aligned}$$

<u>How</u> do weights converge:

Use eigenvalue decomposition (see Appendix):

With $\mathbf{Q}^h \cdot \mathbf{Q} = \mathbf{Q} \cdot \mathbf{Q}^h = \mathbf{I}$ and $\mathbf{R}_x = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^h$

$$\Rightarrow \; (\mathbf{I} - 2\alpha\mathbf{R}_x)^k = (\mathbf{Q}\mathbf{Q}^h - 2\alpha\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^h)^k = \mathbf{Q}(\mathbf{I} - 2\alpha\mathbf{\Lambda})^k\mathbf{Q}^h$$

Change of variables: $\underline{\mathbf{D}}[k] = \mathbf{Q}^h \cdot \underline{\mathbf{d}}[k]$

How do weights converge:

Use eigenvalue decomposition (see Appendix):

With $\mathbf{Q}^h \cdot \mathbf{Q} = \mathbf{Q} \cdot \mathbf{Q}^h = \mathbf{I}$ and $\mathbf{R}_x = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^h$

$$\Rightarrow \; (\mathbf{I} - 2\alpha\mathbf{R}_x)^k = (\mathbf{Q}\mathbf{Q}^h - 2\alpha\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^h)^k = \mathbf{Q}(\mathbf{I} - 2\alpha\mathbf{\Lambda})^k\mathbf{Q}^h$$

Change of variables: $\underline{\mathbf{D}}[k] = \mathbf{Q}^h \cdot \underline{\mathbf{d}}[k]$

$$\underline{\mathbf{d}}[k] = (\mathbf{I} - 2\alpha\mathbf{R}_x)^k\underline{\mathbf{d}}[0] \quad \Rightarrow \quad \underline{\mathbf{D}}[k] = (\mathbf{I} - 2\alpha\mathbf{\Lambda})^k\underline{\mathbf{D}}[0]$$

<u>How</u> do weights converge:

Use eigenvalue decomposition (see Appendix):

With $\mathbf{Q}^h \cdot \mathbf{Q} = \mathbf{Q} \cdot \mathbf{Q}^h = \mathbf{I}$ and $\mathbf{R}_x = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^h$

$$\Rightarrow \quad (\mathbf{I} - 2\alpha\mathbf{R}_x)^k = (\mathbf{Q}\mathbf{Q}^h - 2\alpha\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^h)^k$$
$$= \mathbf{Q}(\mathbf{I} - 2\alpha\mathbf{\Lambda})^k\mathbf{Q}^h$$

Change of variables: $\underline{\mathbf{D}}[k] = \mathbf{Q}^h \cdot \underline{\mathbf{d}}[k]$

$$\underline{\mathbf{d}}[k] = (\mathbf{I} - 2\alpha\mathbf{R}_x)^k\underline{\mathbf{d}}[0] \quad \Rightarrow \quad \underline{\mathbf{D}}[k] = (\mathbf{I} - 2\alpha\mathbf{\Lambda})^k\underline{\mathbf{D}}[0]$$

Recursion stable if: $\lim_{k\to\infty} (\mathbf{I} - 2\alpha\mathbf{\Lambda})^k = \mathbf{0}$

Stability GD

Recursion stable if: $\lim_{k\to\infty} (\mathbf{I} - 2\alpha\mathbf{\Lambda})^k = \mathbf{0}$

Recursion stable if: $\lim_{k\to\infty}(\mathbf{I} - 2\alpha\mathbf{\Lambda})^k = \mathbf{0}$

Both matrices $\mathbf{I}$ and $\mathbf{\Lambda}$ diagonal

Stability GD

Recursion stable if: $\lim_{k\to\infty} (\mathbf{I} - 2\alpha\mathbf{\Lambda})^k = \mathbf{0}$

Both matrices $\mathbf{I}$ and $\mathbf{\Lambda}$ diagonal $\Rightarrow$ Stable if:

$$|1 - 2\alpha\lambda_i| < 1 \quad \Leftrightarrow \quad 0 < \alpha < \frac{1}{\lambda_i} \quad \text{for } i = 0, 1, \cdots, N-1$$

Stability GD

Recursion stable if: $\lim_{k\to\infty} (\mathbf{I} - 2\alpha\mathbf{\Lambda})^k = \mathbf{0}$

Both matrices $\mathbf{I}$ and $\mathbf{\Lambda}$ diagonal $\Rightarrow$ Stable if:

$$|1 - 2\alpha\lambda_i| < 1 \quad \Leftrightarrow \quad 0 < \alpha < \frac{1}{\lambda_i} \quad \text{for } i = 0, 1, \cdots, N-1$$

Thus GD algorithm stable if: $\boxed{0 < \alpha < \frac{1}{\lambda_{max}}}$

Stability GD

Recursion stable if: $\lim_{k\to\infty} (\mathbf{I} - 2\alpha\mathbf{\Lambda})^k = \mathbf{0}$

Both matrices $\mathbf{I}$ and $\mathbf{\Lambda}$ diagonal $\Rightarrow$ Stable if:

$$|1 - 2\alpha\lambda_i| < 1 \quad \Leftrightarrow \quad 0 < \alpha < \frac{1}{\lambda_i} \quad \text{for } i = 0, 1, \cdots, N-1$$

Thus GD algorithm stable if: $\boxed{0 < \alpha < \frac{1}{\lambda_{max}}}$

For adaptation constant α in this region:

$$\lim_{k\to\infty} \underline{\mathbf{w}}[k] = \underline{\mathbf{w}}_o = \mathbf{R}_x^{-1} \cdot \underline{\mathbf{r}}_{yx}$$

Recursion stable if: $\lim_{k\to\infty} (\mathbf{I} - 2\alpha\mathbf{\Lambda})^k = \mathbf{0}$

Both matrices $\mathbf{I}$ and $\mathbf{\Lambda}$ diagonal $\Rightarrow$ Stable if:

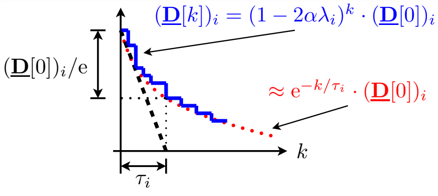
$$|1 - 2\alpha\lambda_i| < 1 \quad \Leftrightarrow \quad 0 < \alpha < \frac{1}{\lambda_i} \quad \text{for } i = 0, 1, \cdots, N-1$$

Thus GD algorithm stable if: $\boxed{0 < \alpha < \frac{1}{\lambda_{max}}}$

For adaptation constant α in this region:

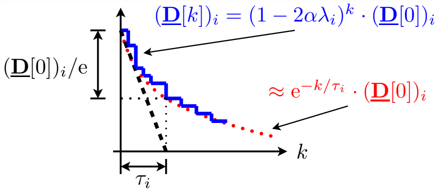
$$\lim_{k\to\infty} \underline{\mathbf{w}}[k] = \underline{\mathbf{w}}_o = \mathbf{R}_x^{-1} \cdot \underline{\mathbf{r}}_{yx}$$

$$J_{\underline{\mathbf{w}}=\underline{\mathbf{w}}_o} = E\{e^2[k]\} = J_{min} = E\{y^2\} - \underline{\mathbf{r}}_{yx}^t \mathbf{R}_x^{-1} \underline{\mathbf{r}}_{yx}$$



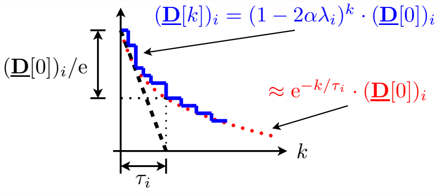
$(\underline{\mathbf{D}}[k])_i = (1 - 2\alpha\lambda_i)^k \cdot (\underline{\mathbf{D}}[0])_i$
$(\underline{\mathbf{D}}[0])_i/\mathrm{e}$
$\approx \mathrm{e}^{-k/\tau_i} \cdot (\underline{\mathbf{D}}[0])_i$
k
τ_i

Convergence GD



$$\mathrm{e}^{-k/\tau_i} \cdot (\underline{\mathbf{D}}[0])_i \approx (1 - 2\alpha\lambda_i)^k \cdot (\underline{\mathbf{D}}[0])_i \quad \Rightarrow$$

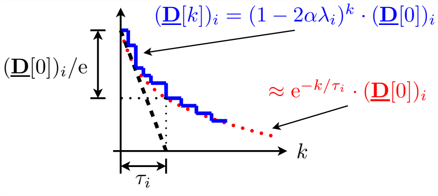
Convergence GD



$$\mathsf{e}^{-k/\tau_i} \cdot (\underline{\mathbf{D}}[0])_i \approx (1 - 2\alpha\lambda_i)^k \cdot (\underline{\mathbf{D}}[0])_i \quad \Rightarrow$$

Average behavior: $\tau_{av,i} = \dfrac{-1}{\ln(1 - 2\alpha\lambda_i)}$

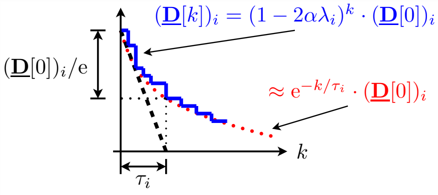
Convergence GD



$$\mathrm{e}^{-k/\tau_i} \cdot (\underline{\mathbf{D}}[0])_i \approx (1-2\alpha\lambda_i)^k \cdot (\underline{\mathbf{D}}[0])_i \quad \Rightarrow$$

Average behavior: $\tau_{av,i} = \dfrac{-1}{\ln(1-2\alpha\lambda_i)}$ $\Rightarrow$ For small α $\boxed{\tau_{av,i} \approx \dfrac{1}{2\alpha\lambda_i}}$

Convergence GD



$$\mathrm{e}^{-k/\tau_i} \cdot (\underline{\mathbf{D}}[0])_i \approx (1-2\alpha\lambda_i)^k \cdot (\underline{\mathbf{D}}[0])_i \quad \Rightarrow$$

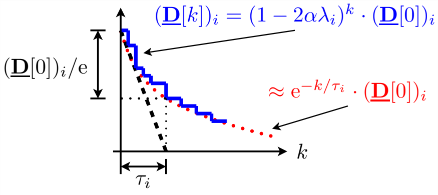
Average behavior: $\tau_{av,i} = \dfrac{-1}{\ln(1-2\alpha\lambda_i)}$ $\Rightarrow$ For small α $\boxed{\tau_{av,i} \approx \dfrac{1}{2\alpha\lambda_i}}$

Note:

Overall time constant depends on eigenvalue spread $\Gamma_x = \lambda_{max}/\lambda_{min}$.
Thus, the larger Γ_x the longer it takes for adaptation.

Technische Universiteit Eindhoven University of Technology

Convergence GD



$$\mathrm{e}^{-k/\tau_i} \cdot (\underline{\mathbf{D}}[0])_i \approx (1-2\alpha\lambda_i)^k \cdot (\underline{\mathbf{D}}[0])_i \quad \Rightarrow$$

Average behavior: $\tau_{av,i} = \dfrac{-1}{\ln(1-2\alpha\lambda_i)}$ $\Rightarrow$ For small α $\boxed{\tau_{av,i} \approx \dfrac{1}{2\alpha\lambda_i}}$

Note:

Overall time constant depends on eigenvalue spread $\Gamma_x = \lambda_{max}/\lambda_{min}$.
Thus, the larger Γ_x the longer it takes for adaptation.

Q: What happens for white noise?

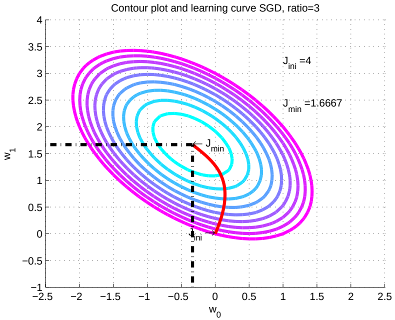
Convergence GD

Example with $\Gamma_x = \lambda_{max}/\lambda_{min} = 3$

Convergence GD

Example with $\Gamma_x = \lambda_{max}/\lambda_{min} = 3$

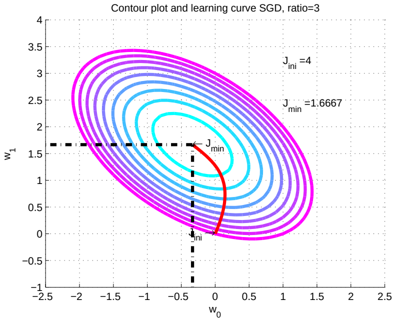
Learning curve in contour plot J



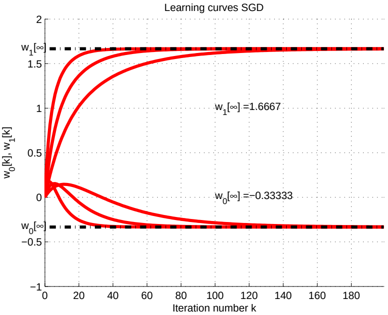
Convergence GD

Example with $\Gamma_x = \lambda_{max}/\lambda_{min} = 3$

Learning curve in contour plot J



Learning curves for different α



Least Mean Square (LMS)

Motivation: GD not practical. Gradient assumes known $\mathbf{R}_x$ and $\underline{\mathbf{r}}_{yx}$

Least Mean Square (LMS)

Motivation: GD not practical. Gradient assumes known $\mathbf{R}_x$ and $\underline{\mathbf{r}}_{yx}$

LMS principle: Use instantaneous estimate of gradient:

$$\begin{aligned}
\underline{\hat{\bigtriangledown}}[k] &= -2\left(y[k]\underline{\mathbf{x}}[k] - \underline{\mathbf{x}}[k]\underline{\mathbf{x}}^t[k]\underline{\mathbf{w}}[k]\right) \\
&= -2\underline{\mathbf{x}}[k]\left(y[k] - \underline{\mathbf{x}}^t[k]\underline{\mathbf{w}}[k]\right) = -2\underline{\mathbf{x}}[k]e[k]
\end{aligned}$$

Least Mean Square (LMS)

Motivation: GD not practical. Gradient assumes known $\mathbf{R}_x$ and $\underline{\mathbf{r}}_{yx}$

LMS principle: Use instantaneous estimate of gradient:

$$\begin{aligned}\underline{\hat{\nabla}}[k] &= -2\left(y[k]\underline{\mathbf{x}}[k] - \underline{\mathbf{x}}[k]\underline{\mathbf{x}}^t[k]\underline{\mathbf{w}}[k]\right) \\ &= -2\underline{\mathbf{x}}[k]\left(y[k] - \underline{\mathbf{x}}^t[k]\underline{\mathbf{w}}[k]\right) = -2\underline{\mathbf{x}}[k]e[k]\end{aligned}$$

With $\underline{\mathbf{w}} \doteq \underline{\mathbf{w}} - \alpha\underline{\hat{\nabla}} \Rightarrow$ LMS algorithm (Widrow, 1975):

$$\boxed{\begin{aligned} k = 0 \quad &: \quad \underline{\mathbf{w}}[0] = \underline{\mathbf{0}} \quad \text{(usually)} \\ k > 0 \quad &: \quad \hat{y}[k] = \underline{\mathbf{w}}^t[k] \cdot \underline{\mathbf{x}}[k] \\ & \quad\;\; e[k] = y[k] - \hat{y}[k] \\ & \quad\;\; \color{red}{\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha\underline{\mathbf{x}}[k]e[k]} \end{aligned}}$$

Least Mean Square (LMS)

Motivation: GD not practical. Gradient assumes known $\mathbf{R}_x$ and $\underline{\mathbf{r}}_{yx}$

LMS principle: Use instantaneous estimate of gradient:

$$\begin{aligned}\hat{\underline{\nabla}}[k] &= -2\left(y[k]\underline{\mathbf{x}}[k] - \underline{\mathbf{x}}[k]\underline{\mathbf{x}}^t[k]\underline{\mathbf{w}}[k]\right) \\ &= -2\underline{\mathbf{x}}[k]\left(y[k] - \underline{\mathbf{x}}^t[k]\underline{\mathbf{w}}[k]\right) = -2\underline{\mathbf{x}}[k]e[k]\end{aligned}$$

With $\underline{\mathbf{w}} \doteq \underline{\mathbf{w}} - \alpha\hat{\underline{\nabla}} \Rightarrow$ LMS algorithm (Widrow, 1975):

$$\boxed{\begin{array}{lll} k=0 & : & \underline{\mathbf{w}}[0] = \underline{\mathbf{0}} \quad \text{(usually)} \\ k>0 & : & \hat{y}[k] = \underline{\mathbf{w}}^t[k]\cdot\underline{\mathbf{x}}[k] \\ & & e[k] = y[k] - \hat{y}[k] \\ & & \underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha\underline{\mathbf{x}}[k]e[k] \end{array}}$$

Note: $\underline{\mathbf{w}}^t[k]\cdot\underline{\mathbf{x}}[k]$ is "convolution" and $\underline{\mathbf{x}}[k]e[k]$ "correlation"

Normalized LMS

- Convergence of LMS strongly depends on α; "optimal" choice depends on amplitude of signals.

Technische Universiteit Eindhoven University of Technology

Normalized LMS

- Convergence of LMS strongly depends on α; "optimal" choice depends on amplitude of signals.
- **NLMS:** LMS with normalization by $\sigma_x^2 = E\{x^2[k]\}$:

$$\boxed{\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + \frac{2\alpha}{\sigma_x^2}\underline{\mathbf{x}}[k]e[k]}$$

Normalized LMS

- Convergence of LMS strongly depends on α; "optimal" choice depends on amplitude of signals.
- **NLMS:** LMS with normalization by $\sigma_x^2 = E\{x^2[k]\}$:

$$\boxed{\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + \frac{2\alpha}{\sigma_x^2}\underline{\mathbf{x}}[k]e[k]}$$

In practice $\hat{\sigma}_x^2[k] \Rightarrow$ time-varying step size. E.g.:

- $\hat{\sigma}_x^2[k] = \beta\hat{\sigma}_x^2[k-1] + (1-\beta)\frac{\underline{\mathbf{x}}^t[k]\underline{\mathbf{x}}[k]}{N}$ with $0 < \beta < 1$
- $\hat{\sigma}_x^2[k] = \frac{\underline{\mathbf{x}}^t[k]\underline{\mathbf{x}}[k]}{N} + \epsilon$ with ϵ some small constant

Newton

Convergence gradient based algorithms depends on coloration input:

$$\underline{\nabla} = -2\left(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}[k]\right)$$

Technische Universiteit
Eindhoven
University of Technology

Newton

Convergence gradient based algorithms depends on coloration input:

$$\underline{\nabla} = -2\left(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}[k]\right)$$

Solution Newton: $\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] - \alpha \mathbf{R}_x^{-1} \underline{\nabla} \Rightarrow$

$$\boxed{\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha \mathbf{R}_x^{-1} \cdot \left(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}[k]\right)}$$

Newton

Convergence gradient based algorithms depends on coloration input:

$$\underline{\nabla} = -2\left(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x\underline{\mathbf{w}}[k]\right)$$

Solution Newton: $\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] - \alpha\mathbf{R}_x^{-1}\underline{\nabla} \Rightarrow$

$$\boxed{\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha\mathbf{R}_x^{-1}\cdot\left(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x\underline{\mathbf{w}}[k]\right)}$$

Convergence Newton:

$$\underline{\mathbf{d}}[k+1] = \left(\mathbf{I} - 2\alpha\mathbf{R}_x^{-1}\mathbf{R}_x\right)\underline{\mathbf{d}}[k] = (1-2\alpha)\underline{\mathbf{d}}[k] \quad \Rightarrow \text{ Convergence } 0 < \alpha < 1$$

Newton

Convergence gradient based algorithms depends on coloration input:

$$\underline{\nabla} = -2\left(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}[k]\right)$$

Solution Newton: $\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] - \alpha \mathbf{R}_x^{-1} \underline{\nabla} \Rightarrow$

$$\boxed{\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha \mathbf{R}_x^{-1} \cdot \left(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}[k]\right)}$$

Convergence Newton:

$$\underline{\mathbf{d}}[k+1] = \left(\mathbf{I} - 2\alpha \mathbf{R}_x^{-1} \mathbf{R}_x\right) \underline{\mathbf{d}}[k] = (1-2\alpha)\underline{\mathbf{d}}[k] \quad \Rightarrow \text{ Convergence } 0 < \alpha < 1$$

Notes:

- $\mathbf{R}_x^{-1}$ causes whitening of input process

Newton

Convergence gradient based algorithms depends on coloration input:

$$\underline{\nabla} = -2\left(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}[k]\right)$$

Solution Newton: $\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] - \alpha \mathbf{R}_x^{-1} \underline{\nabla} \Rightarrow$

$$\boxed{\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha \mathbf{R}_x^{-1} \cdot \left(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}[k]\right)}$$

Convergence Newton:

$$\underline{\mathbf{d}}[k+1] = \left(\mathbf{I} - 2\alpha \mathbf{R}_x^{-1} \mathbf{R}_x\right) \underline{\mathbf{d}}[k] = (1-2\alpha)\underline{\mathbf{d}}[k] \quad \Rightarrow \text{ Convergence } 0 < \alpha < 1$$

Notes:

- $\mathbf{R}_x^{-1}$ causes whitening of input process
- All weights have same convergence (in contrast to LMS, GD)

Newton

Convergence gradient based algorithms depends on coloration input:

$$\underline{\nabla} = -2\left(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}[k]\right)$$

Solution Newton: $\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] - \alpha \mathbf{R}_x^{-1} \underline{\nabla} \Rightarrow$

$$\boxed{\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha \mathbf{R}_x^{-1} \cdot \left(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}[k]\right)}$$

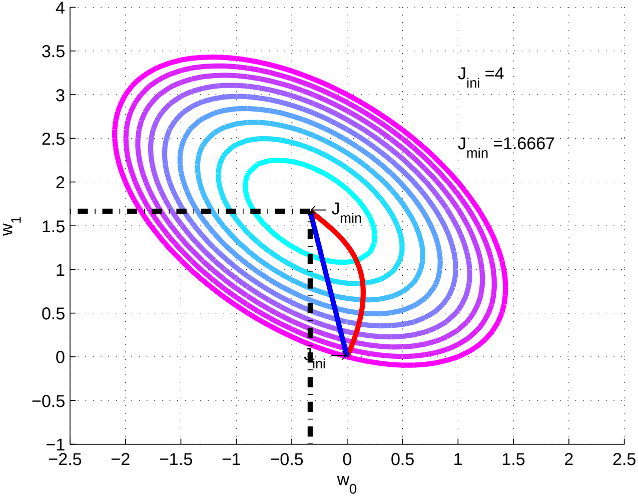
Convergence Newton:

$$\underline{\mathbf{d}}[k+1] = \left(\mathbf{I} - 2\alpha \mathbf{R}_x^{-1} \mathbf{R}_x\right) \underline{\mathbf{d}}[k] = (1-2\alpha)\underline{\mathbf{d}}[k] \quad \Rightarrow \text{ Convergence } 0 < \alpha < 1$$

Notes:

- $\mathbf{R}_x^{-1}$ causes whitening of input process
- All weights have same convergence (in contrast to LMS, GD)
- Newton $\equiv$ GD with white noise input!

Learning curves in contour plot: Newton vs. GD



Newton: Practical problem

Autocorrelation matrix $\mathbf{R}_x$:

Newton: Practical problem

Autocorrelation matrix $\mathbf{R}_x$:

- (In general) not known in advance
- May change during time (non-stationary process)
- Inversion is expensive (many MIPS)

Technische Universiteit Eindhoven University of Technology

Newton: Practical problem

Autocorrelation matrix $\mathbf{R}_x$:

- (In general) not known in advance
- May change during time (non-stationary process)
- Inversion is expensive (many MIPS)

$\Rightarrow$ Complexity Newton algorithm huge

$\Rightarrow$ Need for efficient solution with estimate of $\mathbf{R}_x$

$\Rightarrow$ Different algorithms, e.g. RLS.

Recursive Least Squares (RLS)

For data block length L fixed, Least Squares problem becomes:

$$\min_{\underline{\mathbf{w}}[k]} |\underline{\mathbf{y}}[k] - \mathbf{X}[k] \cdot \underline{\mathbf{w}}[k]|^2 \quad \Rightarrow \quad \underline{\mathbf{w}}_{LS}[k] = \left(\mathbf{X}^t[k]\mathbf{X}[k]\right)^{-1} \left(\mathbf{X}^t[k]\underline{\mathbf{y}}[k]\right)$$

Technische Universiteit
Eindhoven
University of Technology

Recursive Least Squares (RLS)

For data block length L fixed, Least Squares problem becomes:

$$\min_{\underline{\mathbf{w}}[k]} |\underline{\mathbf{y}}[k] - \mathbf{X}[k] \cdot \underline{\mathbf{w}}[k]|^2 \quad \Rightarrow \quad \underline{\mathbf{w}}_{LS}[k] = \left(\mathbf{X}^t[k]\mathbf{X}[k]\right)^{-1} \left(\mathbf{X}^t[k]\underline{\mathbf{y}}[k]\right)$$

RLS: Find efficient recursive solution for LS problem from $k \rightarrow k+1$

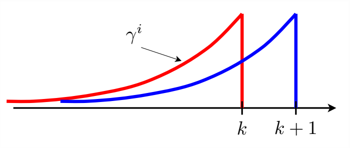
Recursive Least Squares (RLS)

For data block length L fixed, Least Squares problem becomes:

$$\min_{\underline{\mathbf{w}}[k]} |\underline{\mathbf{y}}[k] - \mathbf{X}[k] \cdot \underline{\mathbf{w}}[k]|^2 \quad \Rightarrow \quad \underline{\mathbf{w}}_{LS}[k] = \left(\mathbf{X}^t[k]\mathbf{X}[k]\right)^{-1} \left(\mathbf{X}^t[k]\underline{\mathbf{y}}[k]\right)$$

RLS: Find efficient recursive solution for LS problem from $k \rightarrow k+1$

Use exponential sliding window: Scale down data by factor γ



Forgetting factor : $0 < \gamma < 1$

'Memory' : $\dfrac{1}{1-\gamma}$

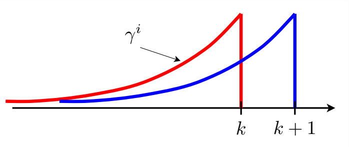
Recursive Least Squares (RLS)

For data block length L fixed, Least Squares problem becomes:

$$\min_{\underline{\mathbf{w}}[k]} |\underline{\mathbf{y}}[k] - \mathbf{X}[k] \cdot \underline{\mathbf{w}}[k]|^2 \quad \Rightarrow \quad \underline{\mathbf{w}}_{LS}[k] = \left(\mathbf{X}^t[k]\mathbf{X}[k]\right)^{-1} \left(\mathbf{X}^t[k]\underline{\mathbf{y}}[k]\right)$$

RLS: Find efficient recursive solution for LS problem from $k \rightarrow k+1$

Use exponential sliding window: Scale down data by factor γ



Forgetting factor : $0 < \gamma < 1$

'Memory' : $\dfrac{1}{1-\gamma}$

$$\mathbf{X}[k] = \begin{pmatrix} \gamma^0 \underline{\mathbf{x}}^t[k] \\ \cdots \\ \gamma^i \underline{\mathbf{x}}^t[k-i] \\ \cdots \\ \gamma^k \underline{\mathbf{x}}^t[0] \end{pmatrix} \text{ and } \underline{\mathbf{y}}[k] = \begin{pmatrix} \gamma^0 y[k] \\ \cdots \\ \gamma^i y[k-i] \\ \cdots \\ \gamma^k y[0] \end{pmatrix}$$

Technische Universiteit Eindhoven University of Technology

RLS algorithm

RLS algorithm

Initialization: $\underline{\bar{\mathbf{r}}}_{yx}[0] = \underline{\mathbf{0}}$; $\overline{\mathbf{R}}_x^{-1}[0] = \delta^{-1}\mathbf{I}$ with δ large

RLS algorithm

Initialization: $\underline{\bar{\mathbf{r}}}_{yx}[0] = \underline{\mathbf{0}}$; $\overline{\mathbf{R}}_x^{-1}[0] = \delta^{-1}\mathbf{I}$ with δ large

For $k \geq 0$:

RLS algorithm

Initialization: $\underline{\bar{\mathbf{r}}}_{yx}[0] = \underline{\mathbf{0}}$; $\overline{\mathbf{R}}_x^{-1}[0] = \delta^{-1}\mathbf{I}$ with δ large

For $k \geq 0$:

$$\overline{\mathbf{R}}_x^{-1}[k+1] \quad = \quad \gamma^{-2}\left(\overline{\mathbf{R}}_x^{-1}[k] - \underline{\mathbf{g}}[k+1] \cdot \underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}_x^{-1}[k]\right)$$

RLS algorithm

Initialization: $\underline{\overline{\mathbf{r}}}_{yx}[0] = \underline{\mathbf{0}}$; $\overline{\mathbf{R}}_x^{-1}[0] = \delta^{-1}\mathbf{I}$ with δ large

For $k \geq 0$:

$$\begin{aligned}
\underline{\mathbf{g}}[k+1] &= \frac{\overline{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]}{\gamma^2 + \underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]} \\
\overline{\mathbf{R}}_x^{-1}[k+1] &= \gamma^{-2}\left(\overline{\mathbf{R}}_x^{-1}[k] - \underline{\mathbf{g}}[k+1] \cdot \underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}_x^{-1}[k]\right)
\end{aligned}$$

RLS algorithm

Initialization: $\underline{\overline{\mathbf{r}}}_{yx}[0] = \underline{\mathbf{0}}$; $\overline{\mathbf{R}}_x^{-1}[0] = \delta^{-1}\mathbf{I}$ with δ large

For $k \geq 0$:

$$\begin{aligned}
\underline{\mathbf{g}}[k+1] &= \frac{\overline{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]}{\gamma^2 + \underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]} \\
\overline{\mathbf{R}}_x^{-1}[k+1] &= \gamma^{-2}\left(\overline{\mathbf{R}}_x^{-1}[k] - \underline{\mathbf{g}}[k+1]\cdot\underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}_x^{-1}[k]\right) \\
\underline{\overline{\mathbf{r}}}_{yx}[k+1] &= \gamma^2\underline{\overline{\mathbf{r}}}_{yx}[k] + \underline{\mathbf{x}}^t[k+1]\cdot y[k+1]
\end{aligned}$$

RLS algorithm

Initialization: $\underline{\overline{\mathbf{r}}}_{yx}[0] = \underline{\mathbf{0}}$; $\overline{\mathbf{R}}_x^{-1}[0] = \delta^{-1}\mathbf{I}$ with δ large

For $k \geq 0$:

$$\begin{aligned}
\underline{\mathbf{g}}[k+1] &= \frac{\overline{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]}{\gamma^2 + \underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]} \\
\overline{\mathbf{R}}_x^{-1}[k+1] &= \gamma^{-2}\left(\overline{\mathbf{R}}_x^{-1}[k] - \underline{\mathbf{g}}[k+1]\cdot\underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}_x^{-1}[k]\right) \\
\underline{\overline{\mathbf{r}}}_{yx}[k+1] &= \gamma^2\underline{\overline{\mathbf{r}}}_{yx}[k] + \underline{\mathbf{x}}^t[k+1]\cdot y[k+1] \\
\underline{\mathbf{w}}[k+1] &= \overline{\mathbf{R}}_x^{-1}[k+1]\cdot\underline{\mathbf{r}}_{yx}[k+1]
\end{aligned}$$

RLS algorithm

Initialization: $\underline{\bar{\mathbf{r}}}_{yx}[0] = \underline{\mathbf{0}}$; $\overline{\mathbf{R}}_x^{-1}[0] = \delta^{-1}\mathbf{I}$ with δ large

For $k \geq 0$:

$$
\begin{aligned}
\underline{\mathbf{g}}[k+1] &= \frac{\overline{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]}{\gamma^2 + \underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]} \\
\overline{\mathbf{R}}_x^{-1}[k+1] &= \gamma^{-2}\left(\overline{\mathbf{R}}_x^{-1}[k] - \underline{\mathbf{g}}[k+1]\cdot\underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}_x^{-1}[k]\right) \\
\underline{\bar{\mathbf{r}}}_{yx}[k+1] &= \gamma^2\underline{\bar{\mathbf{r}}}_{yx}[k] + \underline{\mathbf{x}}^t[k+1]\cdot y[k+1] \\
\underline{\mathbf{w}}[k+1] &= \overline{\mathbf{R}}_x^{-1}[k+1]\cdot\underline{\mathbf{r}}_{yx}[k+1]
\end{aligned}
$$

Notes:

- $\underline{\mathbf{w}}[\infty] \rightarrow \underline{\mathbf{w}}_o$

RLS algorithm

Initialization: $\underline{\bar{\mathbf{r}}}_{yx}[0] = \underline{\mathbf{0}}$; $\overline{\mathbf{R}}_x^{-1}[0] = \delta^{-1}\mathbf{I}$ with δ large

For $k \geq 0$:

$$\begin{aligned}
\underline{\mathbf{g}}[k+1] &= \frac{\overline{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]}{\gamma^2 + \underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]} \\
\overline{\mathbf{R}}_x^{-1}[k+1] &= \gamma^{-2}\left(\overline{\mathbf{R}}_x^{-1}[k] - \underline{\mathbf{g}}[k+1]\cdot\underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}_x^{-1}[k]\right) \\
\underline{\bar{\mathbf{r}}}_{yx}[k+1] &= \gamma^2\underline{\bar{\mathbf{r}}}_{yx}[k] + \underline{\mathbf{x}}^t[k+1]\cdot y[k+1] \\
\underline{\mathbf{w}}[k+1] &= \overline{\mathbf{R}}_x^{-1}[k+1]\cdot\underline{\mathbf{r}}_{yx}[k+1]
\end{aligned}$$

Notes:

- $\underline{\mathbf{w}}[\infty] \rightarrow \underline{\mathbf{w}}_o$
- Complexity $O(N^2)$ per time update

RLS algorithm

Initialization: $\underline{\overline{\mathbf{r}}}_{yx}[0] = \underline{\mathbf{0}}$; $\overline{\mathbf{R}}_x^{-1}[0] = \delta^{-1}\mathbf{I}$ with δ large

For $k \geq 0$:

$$\begin{aligned}
\underline{\mathbf{g}}[k+1] &= \frac{\overline{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]}{\gamma^2 + \underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]} \\
\overline{\mathbf{R}}_x^{-1}[k+1] &= \gamma^{-2}\left(\overline{\mathbf{R}}_x^{-1}[k] - \underline{\mathbf{g}}[k+1]\cdot\underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}_x^{-1}[k]\right) \\
\underline{\overline{\mathbf{r}}}_{yx}[k+1] &= \gamma^2\underline{\overline{\mathbf{r}}}_{yx}[k] + \underline{\mathbf{x}}^t[k+1]\cdot y[k+1] \\
\underline{\mathbf{w}}[k+1] &= \overline{\mathbf{R}}_x^{-1}[k+1]\cdot\underline{\mathbf{r}}_{yx}[k+1]
\end{aligned}$$

Notes:

- $\underline{\mathbf{w}}[\infty] \rightarrow \underline{\mathbf{w}}_o$
- Complexity $O(N^2)$ per time update
- Window length increases when time increases!

RLS algorithm

Initialization: $\underline{\bar{\mathbf{r}}}_{yx}[0] = \underline{\mathbf{0}}$; $\overline{\mathbf{R}}_x^{-1}[0] = \delta^{-1}\mathbf{I}$ with δ large

For $k \geq 0$:

$$
\begin{aligned}
\underline{\mathbf{g}}[k+1] &= \frac{\overline{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]}{\gamma^2 + \underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]} \\
\overline{\mathbf{R}}_x^{-1}[k+1] &= \gamma^{-2}\left(\overline{\mathbf{R}}_x^{-1}[k] - \underline{\mathbf{g}}[k+1]\cdot\underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}_x^{-1}[k]\right) \\
\underline{\bar{\mathbf{r}}}_{yx}[k+1] &= \gamma^2\underline{\bar{\mathbf{r}}}_{yx}[k] + \underline{\mathbf{x}}^t[k+1]\cdot y[k+1] \\
\underline{\mathbf{w}}[k+1] &= \overline{\mathbf{R}}_x^{-1}[k+1]\cdot\underline{\mathbf{r}}_{yx}[k+1]
\end{aligned}
$$

Notes:

- $\underline{\mathbf{w}}[\infty] \to \underline{\mathbf{w}}_o$
- Complexity $O(N^2)$ per time update
- Window length increases when time increases!
- Exhibits unstable roundoff error accumulation

RLS algorithm

Initialization: $\underline{\bar{\mathbf{r}}}_{yx}[0] = \underline{\mathbf{0}}$; $\overline{\mathbf{R}}_x^{-1}[0] = \delta^{-1}\mathbf{I}$ with δ large

For $k \geq 0$:

$$\begin{aligned}
\underline{\mathbf{g}}[k+1] &= \frac{\overline{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]}{\gamma^2 + \underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]} \\
\overline{\mathbf{R}}_x^{-1}[k+1] &= \gamma^{-2}\left(\overline{\mathbf{R}}_x^{-1}[k] - \underline{\mathbf{g}}[k+1]\cdot\underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}_x^{-1}[k]\right) \\
\underline{\bar{\mathbf{r}}}_{yx}[k+1] &= \gamma^2\underline{\bar{\mathbf{r}}}_{yx}[k] + \underline{\mathbf{x}}^t[k+1]\cdot y[k+1] \\
\underline{\mathbf{w}}[k+1] &= \overline{\mathbf{R}}_x^{-1}[k+1]\cdot\underline{\mathbf{r}}_{yx}[k+1]
\end{aligned}$$

Notes:

- $\underline{\mathbf{w}}[\infty] \rightarrow \underline{\mathbf{w}}_o$
- Complexity $O(N^2)$ per time update
- Window length increases when time increases!
- Exhibits unstable roundoff error accumulation
- RLS is basis for many practical algorithms

RLS algorithm

Initialization: $\underline{\bar{\mathbf{r}}}_{yx}[0] = \underline{\mathbf{0}}$; $\overline{\mathbf{R}}_x^{-1}[0] = \delta^{-1}\mathbf{I}$ with δ large

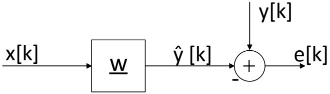
For $k \geq 0$:

$$\begin{aligned}
\underline{\mathbf{g}}[k+1] &= \frac{\overline{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]}{\gamma^2 + \underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]} \\
\overline{\mathbf{R}}_x^{-1}[k+1] &= \gamma^{-2}\left(\overline{\mathbf{R}}_x^{-1}[k] - \underline{\mathbf{g}}[k+1]\cdot\underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}_x^{-1}[k]\right) \\
\underline{\bar{\mathbf{r}}}_{yx}[k+1] &= \gamma^2\underline{\bar{\mathbf{r}}}_{yx}[k] + \underline{\mathbf{x}}^t[k+1]\cdot y[k+1] \\
\underline{\mathbf{w}}[k+1] &= \overline{\mathbf{R}}_x^{-1}[k+1]\cdot\underline{\mathbf{r}}_{yx}[k+1]
\end{aligned}$$

Notes:

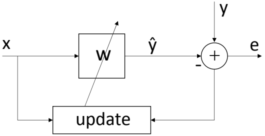
- $\underline{\mathbf{w}}[\infty] \rightarrow \underline{\mathbf{w}}_o$
- Complexity $O(N^2)$ per time update
- Window length increases when time increases!
- Exhibits unstable roundoff error accumulation
- RLS is basis for many practical algorithms
- Decorrelation takes place in algorithm

Summary



	MMSE	LS
Auto correlation	$\mathbf{R}_x = E\{\underline{\mathbf{x}}[k] \cdot \underline{\mathbf{x}}^t[k]\}$	$\overline{\mathbf{R}}_x = \mathbf{X}^t \cdot \mathbf{X}$
Cross correlation	$\underline{\mathbf{r}}_{yx} = E\{y[k] \cdot \underline{\mathbf{x}}[k]\}$	$\underline{\overline{\mathbf{r}}}_{yx} = \mathbf{X}^t \cdot \underline{\mathbf{y}}$
Error J	$E\{e^2[k]\}$	$\sum_{i=0}^{L-1} e^2[k-i]$
Criterion	$\min_{\underline{\mathbf{w}}}\{E\{e^2[k]\}\}$	$\min_{\underline{\mathbf{w}}} \|\underline{\mathbf{y}} - \mathbf{X} \cdot \underline{\mathbf{w}}\|^2$
Opt. solution $\underline{\mathbf{w}}_o$	$\mathbf{R}_x^{-1} \cdot \underline{\mathbf{r}}_{yx}$	$\overline{\mathbf{R}}_x^{-1} \cdot \underline{\overline{\mathbf{r}}}_{yx}$
Min. error J_{min}	$E\{y^2\} - \underline{\mathbf{r}}_{yx}^t \mathbf{R}_x^{-1} \underline{\mathbf{r}}_{yx}$	$\underline{\mathbf{y}}^t \underline{\mathbf{y}} - \underline{\overline{\mathbf{r}}}_{yx}^t \overline{\mathbf{R}}_x^{-1} \underline{\overline{\mathbf{r}}}_{yx}$

Summary



Simple adaptive algorithms (no decorrelation):

$$\text{GD} \quad : \quad \underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x\underline{\mathbf{w}}[k])$$

$$\text{(N)LMS} \quad : \quad \underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + \frac{2\alpha}{\hat{\sigma}_x^2}\underline{\mathbf{x}}[k]e^*[k]$$

Algorithms with improved convergence:

LMS/Newton : $\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha \mathbf{R}_x^{-1}\underline{\mathbf{x}}[k]r[k]$

Newton : $\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha \mathbf{R}_x^{-1} \cdot \left(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x \underline{\mathbf{w}}[k]\right)$

RLS :
$$\underline{\mathbf{g}}[k+1] = \frac{\overline{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]}{\gamma^2 + \underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]}$$
$$\overline{\mathbf{R}}_x^{-1}[k+1] = \gamma^{-2}\left(\overline{\mathbf{R}}_x^{-1}[k] - \underline{\mathbf{g}}[k+1] \cdot \underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}_x^{-1}[k]\right)$$
$$\underline{\overline{\mathbf{r}}}_{yx}[k+1] = \gamma^2 \underline{\overline{\mathbf{r}}}_{yx}[k] + \underline{\mathbf{x}}^t[k+1] \cdot y[k+1]$$
$$\underline{\mathbf{w}}[k+1] = \overline{\mathbf{R}}_x^{-1}[k+1] \cdot \underline{\mathbf{r}}_{yx}[k+1]$$

Technische Universiteit Eindhoven University of Technology

Appendix
Optimum Linear Filters &
Adaptive Signal Processing

Content appendix

- Eigenvalue problem

Eigenvalue problem

Procedure: With eigenvalues λ_i and eigenvectors $\underline{\mathbf{q}}_i$:

$$\mathbf{R} \cdot \underline{\mathbf{q}}_i = \lambda_i \cdot \underline{\mathbf{q}}_i \;\; \Rightarrow \;\; (\mathbf{R} - \lambda_i \mathbf{I}) \cdot \underline{\mathbf{q}}_i = \underline{\mathbf{0}} \text{ for } i = 0, 1, \cdots, N-1$$

Eigenvalue problem

Procedure: With eigenvalues λ_i and eigenvectors $\underline{\mathbf{q}}_i$:

$$\mathbf{R} \cdot \underline{\mathbf{q}}_i = \lambda_i \cdot \underline{\mathbf{q}}_i \quad \Rightarrow \quad (\mathbf{R} - \lambda_i \mathbf{I}) \cdot \underline{\mathbf{q}}_i = \underline{\mathbf{0}} \ \text{ for } i = 0, 1, \cdots, N-1$$

With $\mathbf{Q} = (\underline{\mathbf{q}}_0, \cdots, \underline{\mathbf{q}}_{N-1})$ and $\mathbf{\Lambda} = diag\{\lambda_0, \cdots, \lambda_{N-1}\}$

$$\mathbf{R} \cdot \mathbf{Q} = \mathbf{Q} \cdot \mathbf{\Lambda} \quad \Rightarrow \quad \mathbf{R} = \mathbf{Q} \mathbf{\Lambda} \mathbf{Q}^{-1}$$

Eigenvalue problem

Procedure: With eigenvalues λ_i and eigenvectors $\underline{\mathbf{q}}_i$:

$$\mathbf{R} \cdot \underline{\mathbf{q}}_i = \lambda_i \cdot \underline{\mathbf{q}}_i \quad \Rightarrow \quad (\mathbf{R} - \lambda_i \mathbf{I}) \cdot \underline{\mathbf{q}}_i = \underline{\mathbf{0}} \text{ for } i = 0, 1, \cdots, N-1$$

With $\mathbf{Q} = (\underline{\mathbf{q}}_0, \cdots, \underline{\mathbf{q}}_{N-1})$ and $\mathbf{\Lambda} = diag\{\lambda_0, \cdots, \lambda_{N-1}\}$

$$\mathbf{R} \cdot \mathbf{Q} = \mathbf{Q} \cdot \mathbf{\Lambda} \quad \Rightarrow \quad \mathbf{R} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1}$$

Property: *Eigenvectors* $\underline{\mathbf{q}}_i$ *orthogonal* $\Rightarrow$

$$\mathbf{Q}^h \cdot \mathbf{Q} = \mathbf{Q} \cdot \mathbf{Q}^h = c \cdot \mathbf{I} \quad \text{with} \quad c \quad \text{some constant}$$

Eigenvalue problem

Procedure: With eigenvalues λ_i and eigenvectors $\underline{\mathbf{q}}_i$:

$$\mathbf{R} \cdot \underline{\mathbf{q}}_i = \lambda_i \cdot \underline{\mathbf{q}}_i \quad \Rightarrow \quad (\mathbf{R} - \lambda_i \mathbf{I}) \cdot \underline{\mathbf{q}}_i = \underline{\mathbf{0}} \text{ for } i = 0, 1, \cdots, N-1$$

With $\mathbf{Q} = (\underline{\mathbf{q}}_0, \cdots, \underline{\mathbf{q}}_{N-1})$ and $\mathbf{\Lambda} = diag\{\lambda_0, \cdots, \lambda_{N-1}\}$

$$\mathbf{R} \cdot \mathbf{Q} = \mathbf{Q} \cdot \mathbf{\Lambda} \quad \Rightarrow \quad \mathbf{R} = \mathbf{Q} \mathbf{\Lambda} \mathbf{Q}^{-1}$$

Property: *Eigenvectors* $\underline{\mathbf{q}}_i$ *orthogonal* $\Rightarrow$

$$\mathbf{Q}^h \cdot \mathbf{Q} = \mathbf{Q} \cdot \mathbf{Q}^h = c \cdot \mathbf{I} \quad \text{with} \quad c \quad \text{some constant}$$

Main result:

Diagonalization: $\boxed{\mathbf{Q}^h \mathbf{R} \mathbf{Q} = \mathbf{\Lambda} \quad \Leftrightarrow \quad \mathbf{R} = \mathbf{Q} \mathbf{\Lambda} \mathbf{Q}^h}$

Technische Universiteit Eindhoven University of Technology

Example MA(1):

$x[k] = i[k] + ai[k-1]$ with $E\{i[k]\} = 0$ and $E\{i^2[k]\} = \sigma_i^2 \Rightarrow$

Eigenvalue problem

Example MA(1):

$x[k] = i[k] + ai[k-1]$ with $E\{i[k]\} = 0$ and $E\{i^2[k]\} = \sigma_i^2 \Rightarrow$

$\rho[0] = (1+a^2)\sigma_i^2$; $\rho[1] = \rho[-1] = a\sigma_i^2$; $\rho[\tau] = 0$ for $|\tau| \geq 2$

Eigenvalue problem

Example MA(1):

$x[k] = i[k] + ai[k-1]$ with $E\{i[k]\} = 0$ and $E\{i^2[k]\} = \sigma_i^2 \Rightarrow$

$\rho[0] = (1+a^2)\sigma_i^2$; $\rho[1] = \rho[-1] = a\sigma_i^2$; $\rho[\tau] = 0$ for $|\tau| \geq 2$

Eigenvalues problem det $(\mathbf{R} - \lambda\mathbf{I}) = 0$ for $N = 2$ (with $\gamma = \rho[1]/\rho[0]$):

$$\mathbf{\Lambda} = \begin{pmatrix} \lambda_0 & 0 \\ 0 & \lambda_1 \end{pmatrix} = \begin{pmatrix} 1+\gamma & 0 \\ 0 & 1-\gamma \end{pmatrix} \; ; \; \mathbf{Q} = \left(\underline{\mathbf{q}}_0, \underline{\mathbf{q}}_1\right) = c \cdot \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Eigenvalue problem

Example MA(1):

$x[k] = i[k] + ai[k-1]$ with $E\{i[k]\} = 0$ and $E\{i^2[k]\} = \sigma_i^2 \Rightarrow$

$\rho[0] = (1+a^2)\sigma_i^2$; $\rho[1] = \rho[-1] = a\sigma_i^2$; $\rho[\tau] = 0$ for $|\tau| \geq 2$

Eigenvalues problem $\det(\mathbf{R} - \lambda\mathbf{I}) = 0$ for $N = 2$ (with $\gamma = \rho[1]/\rho[0]$):

$$\mathbf{\Lambda} = \begin{pmatrix} \lambda_0 & 0 \\ 0 & \lambda_1 \end{pmatrix} = \begin{pmatrix} 1+\gamma & 0 \\ 0 & 1-\gamma \end{pmatrix} \;\; ; \;\; \mathbf{Q} = \left(\underline{\mathbf{q}}_0, \underline{\mathbf{q}}_1\right) = c \cdot \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Notes:

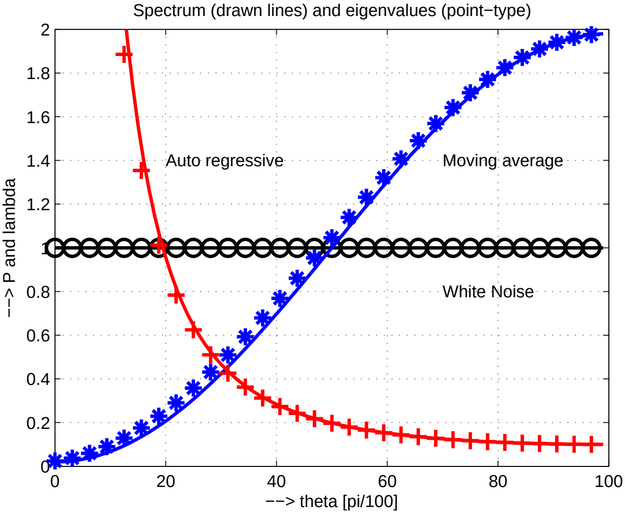
- Vector $\underline{\mathbf{q}}_0$ orthogonal to $\underline{\mathbf{q}}_1$ since $\underline{\mathbf{q}}_0^t \cdot \underline{\mathbf{q}}_1 = 0$
- For white noise ($a = 0$): $\mathbf{\Lambda} = \mathbf{I}$
- For MA(1) with $N > 2$: $\mathbf{R}$ is tri-diagonal

Eigenvalue problem

Example: Eigenvalues and psd for white noise, MA(1) and AR(1)

Eigenvalue problem

Example: Eigenvalues and psd for white noise, MA(1) and AR(1)



Example: Eigenvalues and psd for white noise, MA(1) and AR(1)