Stability SGD

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

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

$$\underline{\mathbf{w}}[k+1] - \underline{\mathbf{w}}_o = \left(\mathbf{I} - 2\alpha\mathbf{R}_x\right)\underline{\mathbf{w}}[k] - \underline{\mathbf{w}}_o + 2\alpha\underline{\mathbf{r}}_{ex}$$

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

Recursion:

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

Stable iff: $\lim_{k\to\infty} \left(\mathbf{I} - 2\alpha\mathbf{R}_x\right)^k = \mathbf{0}$

<u>Note:</u> If stable $\Rightarrow$ $\underline{\mathbf{d}}[\infty] = 0$ $\Rightarrow$ $\underline{\mathbf{w}}[\infty]$ = Wiener

Stability SGD

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 $\mathbf{R}_x = \mathbf{Q} \boldsymbol{\Lambda} \mathbf{Q}^h$

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

Change of variables:

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

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

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

Stability SGD

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

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

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

SGD algorithm stable if: $0 < \alpha < \dfrac{1}{\lambda_{max}}$

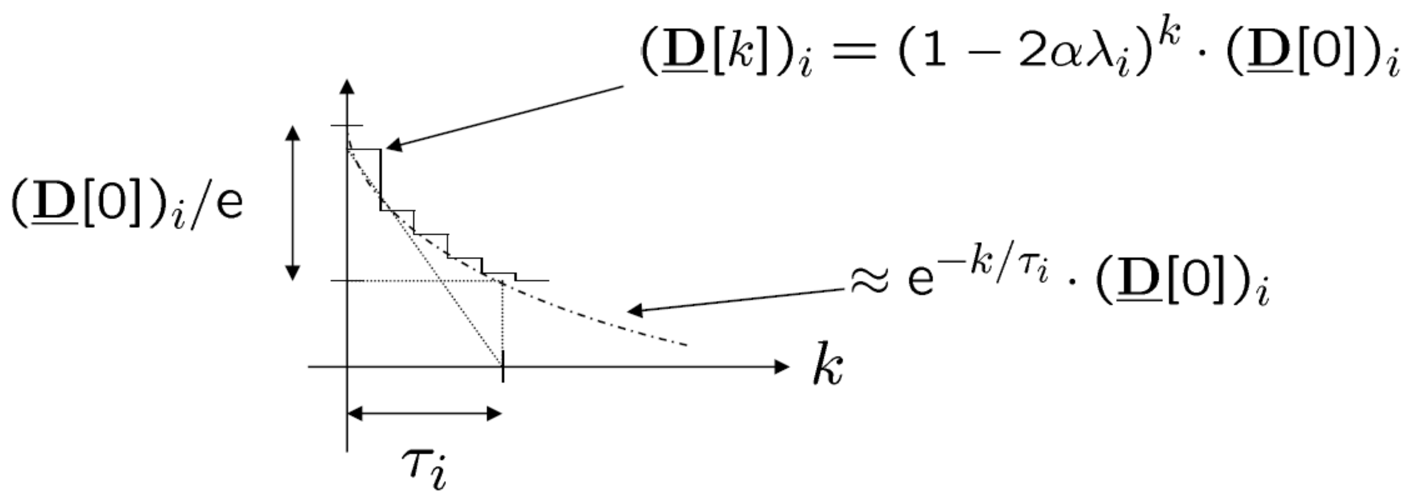
For adaptation constant α in this region:

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

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

Convergence rate SGD

Behaviour coefficient i of $\underline{\mathbf{D}}[k] = (\mathbf{I} - 2\alpha\mathbf{\Lambda})^k \, \underline{\mathbf{D}}[0]$:



Time constant follows from:

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

Convergence rate SGD

$\Rightarrow$ Time constant average weights behaviour:

$$\tau_i = \frac{-1}{\ln(1-2\alpha\lambda_i)} \quad \text{for small } \alpha : \quad \tau_i \approx \frac{1}{2\alpha\lambda_i}$$

Similar derivation for MMSE: $\tau_{mmse,i} \approx \dfrac{1}{4\alpha\lambda_i}$

Notes on overall time constant τ_{av}: $\dfrac{1}{\alpha\lambda_{\min}}$ stable : $\alpha < \dfrac{1}{\lambda_{max}}$

- Depends on eigenvalue spread $\Gamma_x = \lambda_{max}/\lambda_{min}$

$w \rightarrow w_0$ 的最短收敛时间

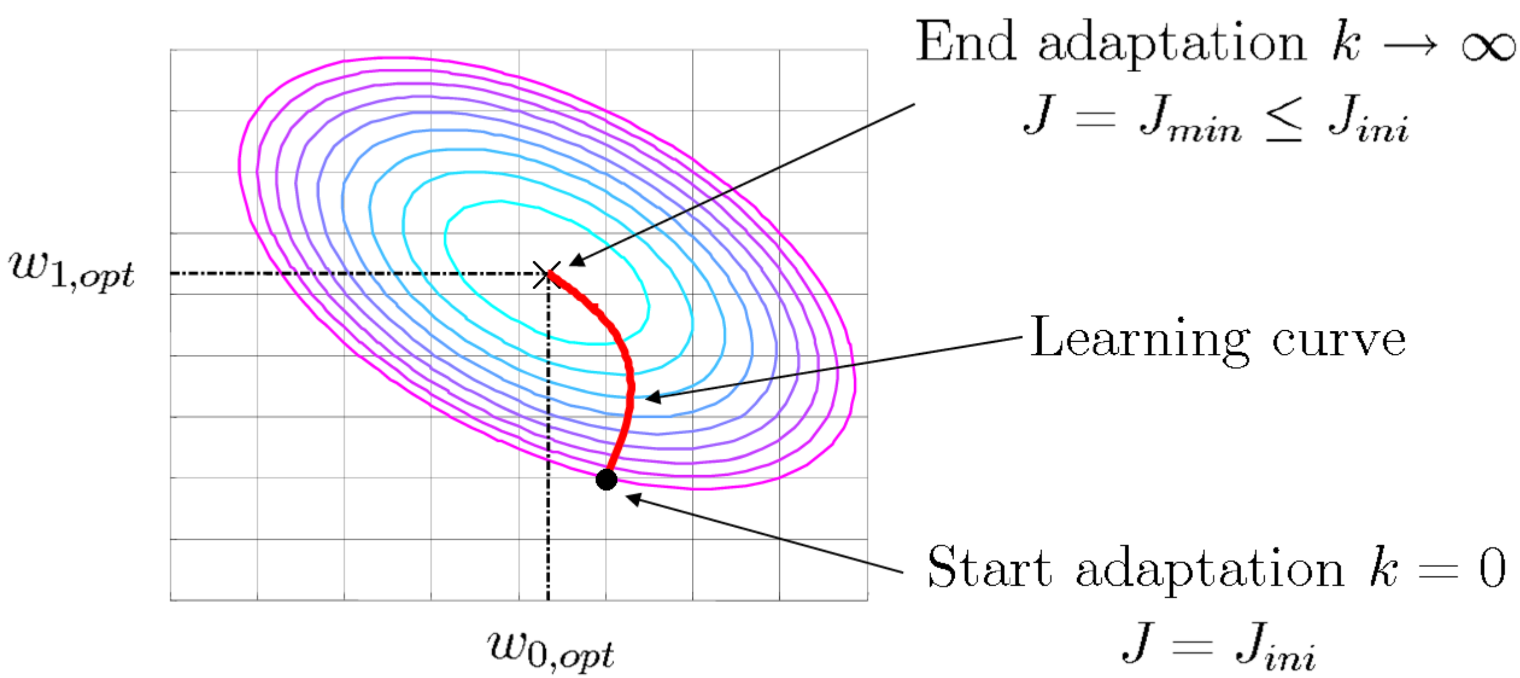
Thus, the larger Γ_x the longer it takes for adaptation

Q: What happens for white noise input process?

Convergence rate SGD

Learning curve in contour plot J

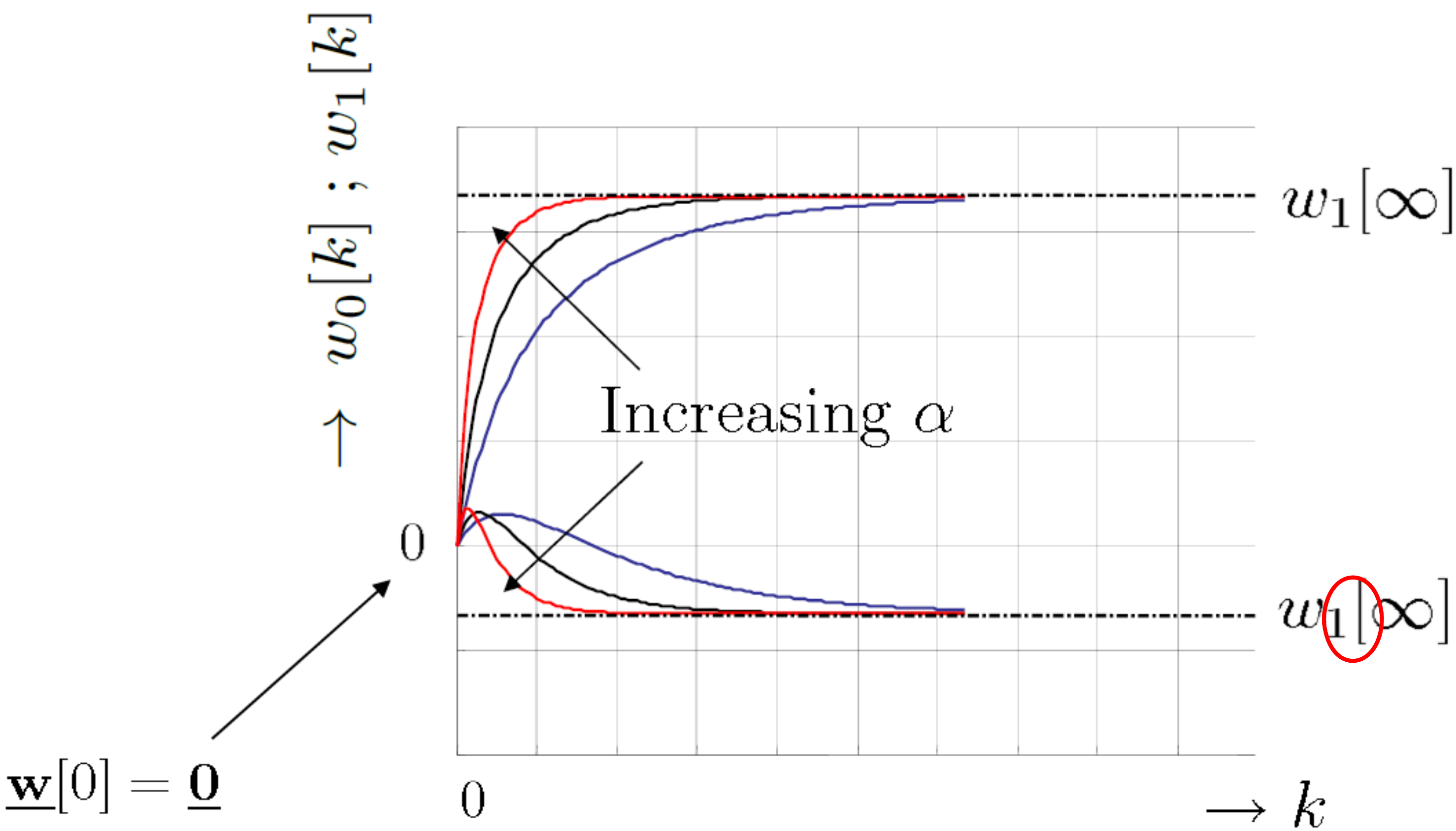
$$\Gamma_x = \frac{\lambda_{max}}{\lambda_{min}} = 3$$



ADSP

Convergence rate SGD

Learning curves for different α



上海科技大学 ShanghaiTech University

ADSP Part B: Adaptive signal processing

Focus on single channel adaptive algorithms using FIR structure

- Applications Adaptive Algorithms
- Minimum Mean Square Error (MMSE)
- Constrained MMSE
- Least Square (LS)
- Steepest Gradient Descent (SGD)
- Three LMS variants: NLMS, Complex LMS, Constrained LMS
- Newton
- Recursive Least Squares (RLS)
- Frequency Domain Adaptive Filter (FDAF)
- Summary

LMS

基于梯度下降

最小均方

LMS: Least Mean Square algorithm

Motivation: SGD not practical since gradient

assumes **known** autocorrelation $\mathbf{R}_x = E\{\underline{\mathbf{x}}[k]\underline{\mathbf{x}}^t[k]\}$

and cross correlation $\underline{\mathbf{r}}_{ex} = E\{e[k]\underline{\mathbf{x}}[k]\}$

瞬时估计可靠吗？

LMS principle: Use *instantaneous* estimate gradient

通过估计量来解决统计量问题

$\underline{\nabla}[k] = -2(\underline{r}_{ex} - R_x \underline{w})$

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

LMS

通过迭代解决求逆问题

$$\underline{\mathbf{w}} := \underline{\mathbf{w}} - \alpha \hat{\underline{\nabla}} \quad \Rightarrow$$

既有估计又有迭代，所以解决了统计量和求逆问题

Least Mean Square (LMS) algorithm (Widrow, 1975):

$$k = 0: \quad \underline{\mathbf{w}}[0] = \underline{\mathbf{0}} \ \text{(usually)}$$

$$k > 0: \quad \hat{e}[k] = \underline{\mathbf{w}}^t[k] \cdot \underline{\mathbf{x}}[k]$$

$$r[k] = e[k] - \hat{e}[k]$$

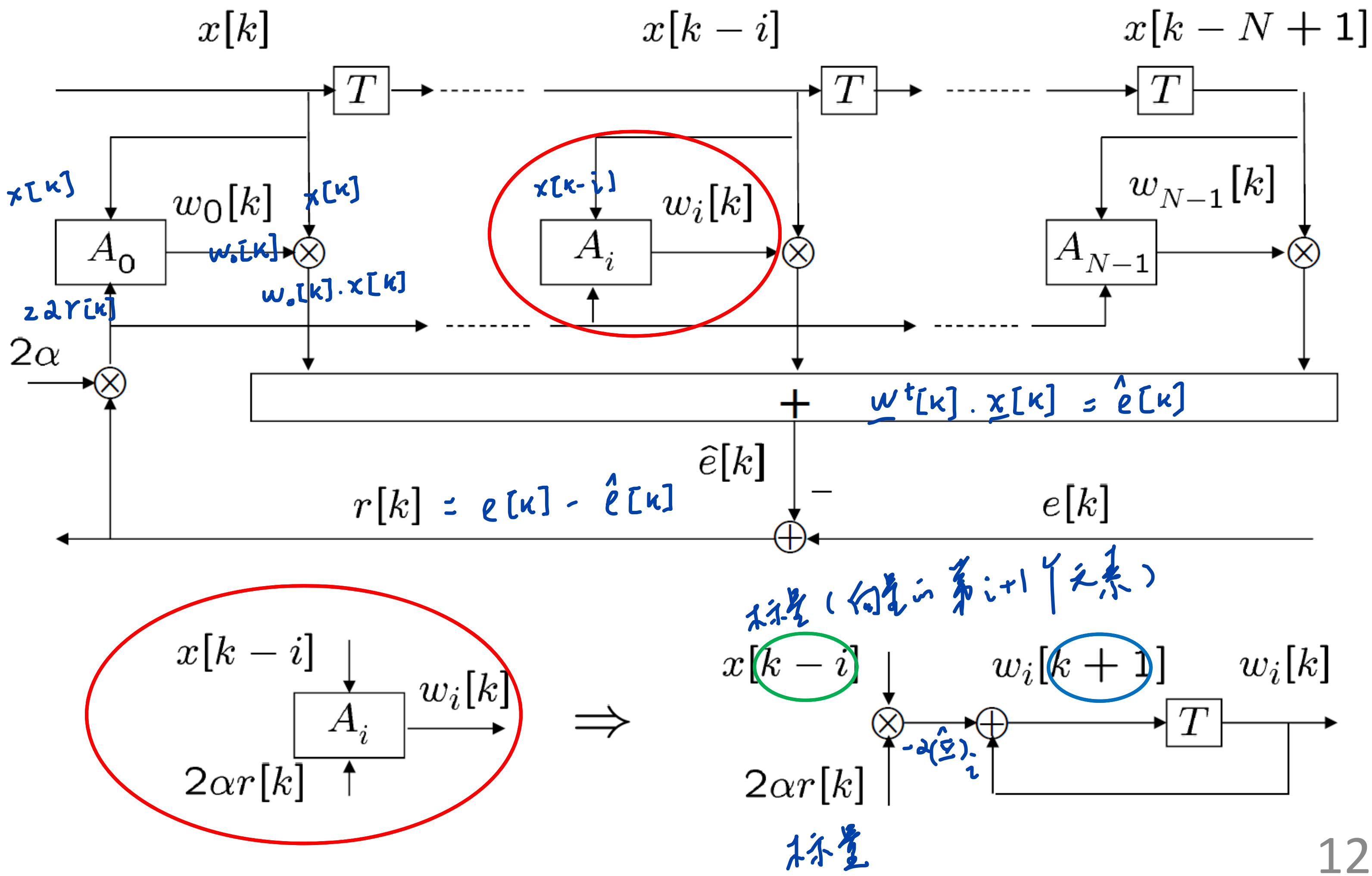
$$\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha \underline{\mathbf{x}}[k] r[k]$$

"*convolution*"

"*cross correlation*"

LMS

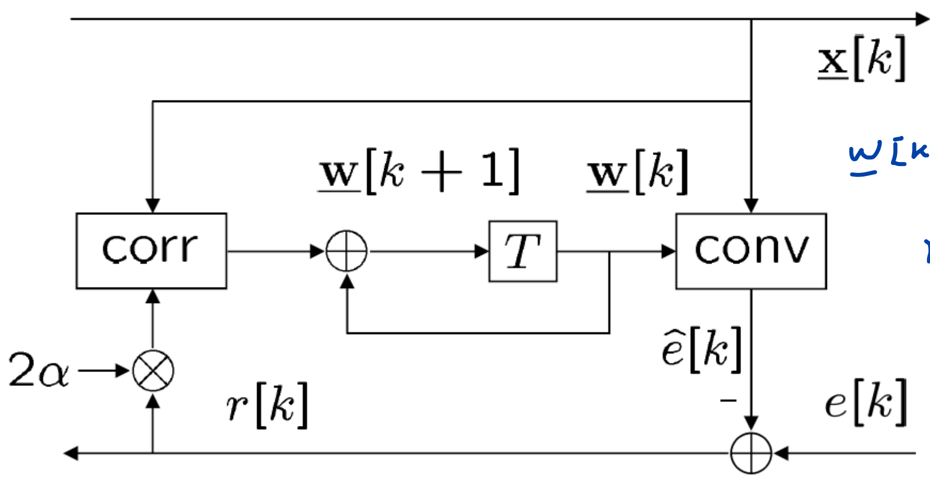
Realization scheme:



ADSP

LMS

Simplified realization scheme:



$\underline{w}[k+1] = \underline{w}[k] + 2\alpha\, \underline{x}[k]\, r[k]$

$r[k] = e[k] - \underline{w}^t[k]\, \underline{x}[k]$

Notes LMS:

Simple and robust algorithm (Complexity $O(2N)$)

LMS tries to "decorrelate" signals x and r 当 r与 $\underline{x}$ 正交时，

去相关 去相关，$\underline{w}[k+1] = \underline{w}[k]$

In contrast to SGD: 收敛

Weights fluctuate around optimal values

因为使用了瞬时估计值 $\underline{x}[k]\, \underline{x}^t[k]$ 与 $e[k]\, \underline{x}[k]$

Three ~~There~~ LMS variants

- **NLMS:**

  LMS with normalization of input signal variance

- **Complex-LMS:**

  LMS for complex signals and weights

  → Similar as before with MMSE

- **Constrained-LMS:**

  LMS with constrained updating

ADSP

NLMS

In LMS convergence properties depends on σ_x^2

$\Rightarrow$ May cause problems with non-stationary input

对于非平稳信号，instantaneous 模型有问题

LMS principle: Use *instantaneous* estimate gradient

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

$$= -2\underline{\mathbf{x}}[k]\left(e[k] - \underline{\mathbf{x}}^t[k]\underline{\mathbf{w}}[k]\right)$$

$$= -2\underline{\mathbf{x}}[k]\left(e[k] - \hat{e}[k]\right)$$

$$= -2\underline{\mathbf{x}}[k]r[k]$$

ADSP

NLMS

In LMS convergence properties depends on σ_x^2

$\Rightarrow$ May cause problems with non-stationary input

<u>Solution:</u> Normalized LMS (NLMS) 归一化，变步长

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

zero-mean ? In practice: $\hat{\sigma}_x^2[k] \quad \Rightarrow \quad$ time-varying step size

1×N×N×1

e.g. 1: $\hat{\sigma}_x^2[k] = \beta\hat{\sigma}_x^2[k-1] + (1-\beta)\left(\frac{\underline{\mathbf{x}}^t[k]\underline{\mathbf{x}}[k]}{N}\right) \quad 0 < \beta < 1$

e.g. 2: $\hat{\sigma}_x^2 = \frac{\underline{\mathbf{x}}^t[k]\underline{\mathbf{x}}[k]}{N} + \varepsilon$ with ε small constant

避免出现 0

ADSP

Complex LMS

LMS update rule: $\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] - \alpha \underline{\hat{\nabla}}$

with $\underline{\hat{\nabla}}$ estimate of $\underline{\nabla}$ 'without' $E\{\cdot\}$ $\Rightarrow$

(see slides Complex MMSE)

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

负梯度

$$\boxed{\Rightarrow \text{ LMS: } \underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha\underline{\mathbf{x}}[k]r^*[k]}$$

Rule of thump C-(N)LMS:

Complex $\underline{\mathbf{w}}$, $\underline{\mathbf{x}}$ and replace r by r^* in (N)LMS

Constrained LMS

Derivation LMS algorithm with **constraints**:

$$\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] - \alpha\hat{\underline{\nabla}}[k]$$

From constrained MMSE slides it follows:

$$\begin{aligned}\underline{\nabla}[k] &= -2\underline{\mathbf{r}}_{ex} + 2\mathbf{R}_x\underline{\mathbf{w}}[k] + \mathbf{C}\underline{\lambda} \\ &= -2E\{\underline{\mathbf{x}}[k]e[k]\} + 2E\{\underline{\mathbf{x}}[k]\underline{\mathbf{x}}^t[k]\}\underline{\mathbf{w}}[k] + \mathbf{C}\underline{\lambda}\end{aligned}$$

LMS estimate of gradient (no $E\{\ \}$):

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

Constrained LMS

$$\Rightarrow \quad \underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha \underline{\mathbf{x}}[k] r[k] - \alpha \mathbf{C} \underline{\lambda}$$

Furthermore: $\mathbf{C}^t \underline{\mathbf{w}}[k+1] = \underline{\mathbf{f}}$ 当前值

$$\Rightarrow \quad \alpha \underline{\lambda} = \left(\mathbf{C}^t \mathbf{C}\right)^{-1} \left(\mathbf{C}^t \underline{\mathbf{w}}[k] + 2\alpha \mathbf{C}^t \underline{\mathbf{x}}[k] r[k] - \underline{\mathbf{f}}\right)$$

Final result:

$$\boxed{\underline{\mathbf{w}}[k+1] = \tilde{\mathbf{P}} \cdot \{\underline{\mathbf{w}}[k] + 2\alpha \underline{\mathbf{x}}[k] r[k]\} + \mathbf{C}\left(\mathbf{C}^t \mathbf{C}\right)^{-1} \underline{\mathbf{f}}}$$

TP_tmp

with projection matrix $\tilde{\mathbf{P}}$ (see Appendix):

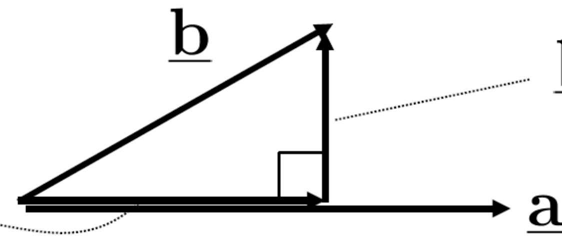
$$\tilde{\mathbf{P}} = \mathbf{I} - \mathbf{C}\left(\mathbf{C}^t \mathbf{C}\right)^{-1} \mathbf{C}^t$$ 垂直于向 C 投影

Note: Initial guess $\underline{\mathbf{w}}[0] = \mathbf{C}\left(\mathbf{C}^t \mathbf{C}\right)^{-1} \underline{\mathbf{f}}$

已知条件可以计算

ADSP

Projection Matrix



$$\underline{\hat{b}} = \alpha \cdot \underline{a}$$

$$\underline{b}^{\perp} = \underline{b} - \underline{\hat{b}}$$

$(\alpha \cdot \underline{a}^h) \cdot (\underline{b} - \alpha \underline{a}) = 0$

$\alpha \underline{a}^h \cdot \underline{b} = \alpha^2 \underline{a}^h \underline{a}$

$\alpha = (\underline{a}^h \underline{a})^{-1} \underline{a}^h \cdot \underline{b}$

$$\underline{\hat{b}}^h \cdot \underline{b}^{\perp} = 0 \;\Rightarrow\; \alpha = \left(\underline{a}^h \underline{a}\right)^{-1} \underline{a}^h \cdot \underline{b}$$

[No Title]

$$\Rightarrow\; \underline{\hat{b}} = \left\{\underline{a} \left(\underline{a}^h \underline{a}\right)^{-1} \underline{a}^h\right\} \cdot \underline{b}$$

标量

$\underline{\hat{b}} = \alpha \cdot \underline{a} = \underline{a} \cdot \alpha$

$= \{\underline{a} (\underline{a}^h \underline{a})^{-1} \underline{a}^h\} \underline{b}$

with projection matrix $\mathbf{P}_a = \underline{a}\left(\underline{a}^h \underline{a}\right)^{-1} \underline{a}^h$

$$\Rightarrow\; \underline{\hat{b}} = \mathbf{P}_a \cdot \underline{b} \text{ and } \underline{b}^{\perp} = (\mathbf{I} - \mathbf{P}_a)\, \underline{b}$$

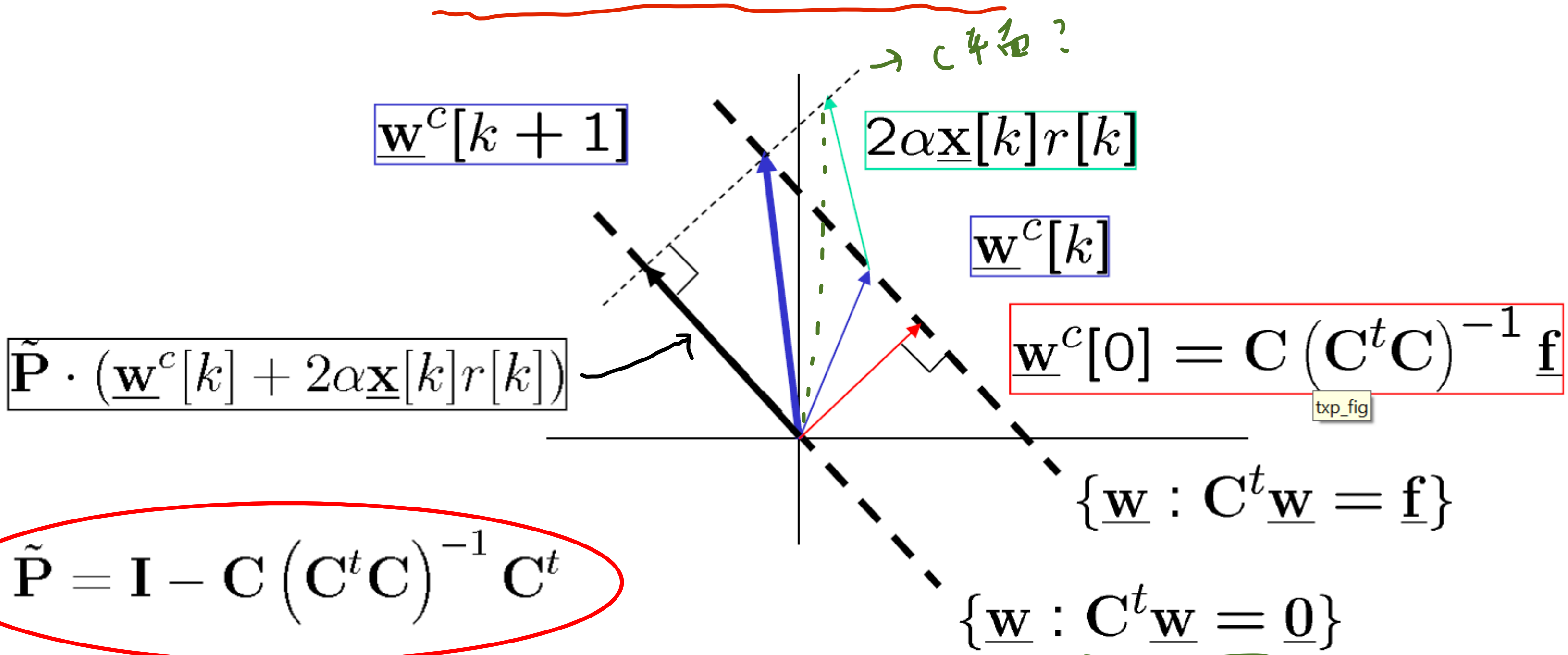
ADSP

Constrained LMS

Geometrical explanation constrained LMS

Rewrite update equation: 此项 垂直于 $(\underline{w}^c[k] + 2\alpha \underline{x}[k] r[k])$ 向 C 投影

$$\underline{\mathbf{w}}^c[k+1] = \tilde{\mathbf{P}} \cdot (\underline{\mathbf{w}}^c[k] + 2\alpha \underline{\mathbf{x}}[k] r[k]) + \underline{\mathbf{w}}^c[0]$$



$$\tilde{\mathbf{P}} = \mathbf{I} - \mathbf{C}\left(\mathbf{C}^t\mathbf{C}\right)^{-1}\mathbf{C}^t$$