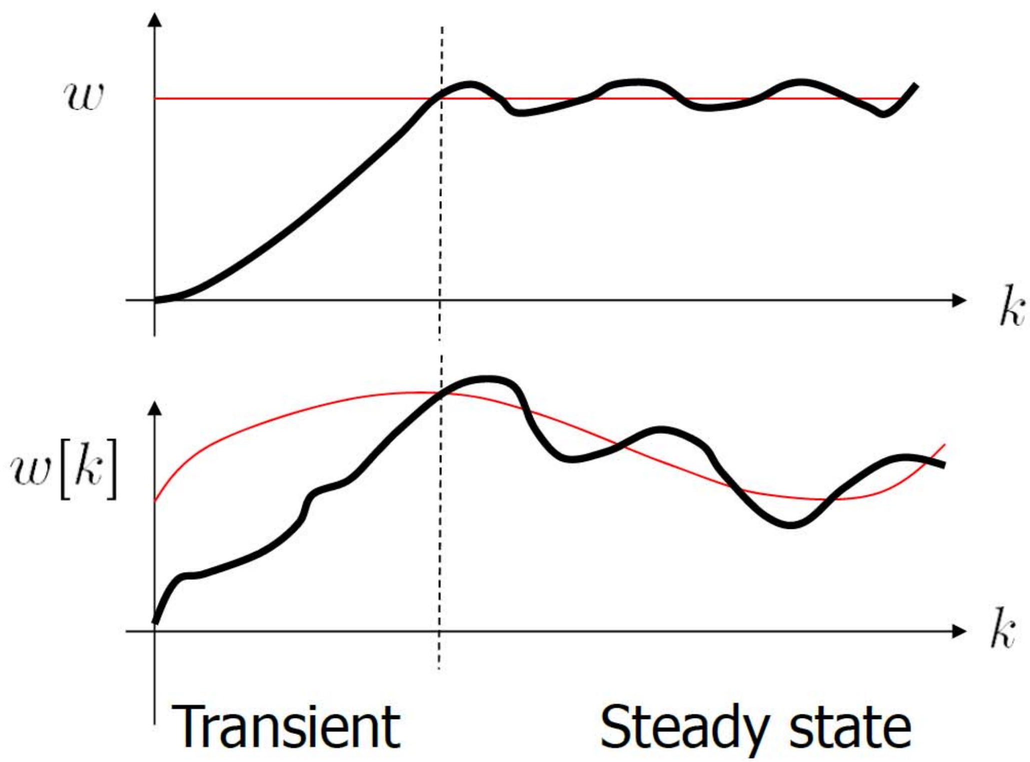
Advanced Digital Signal Processing (ADSP)

徐林

上海科技大学
ShanghaiTech University

Convergence LMS

Acquisition and tracking:



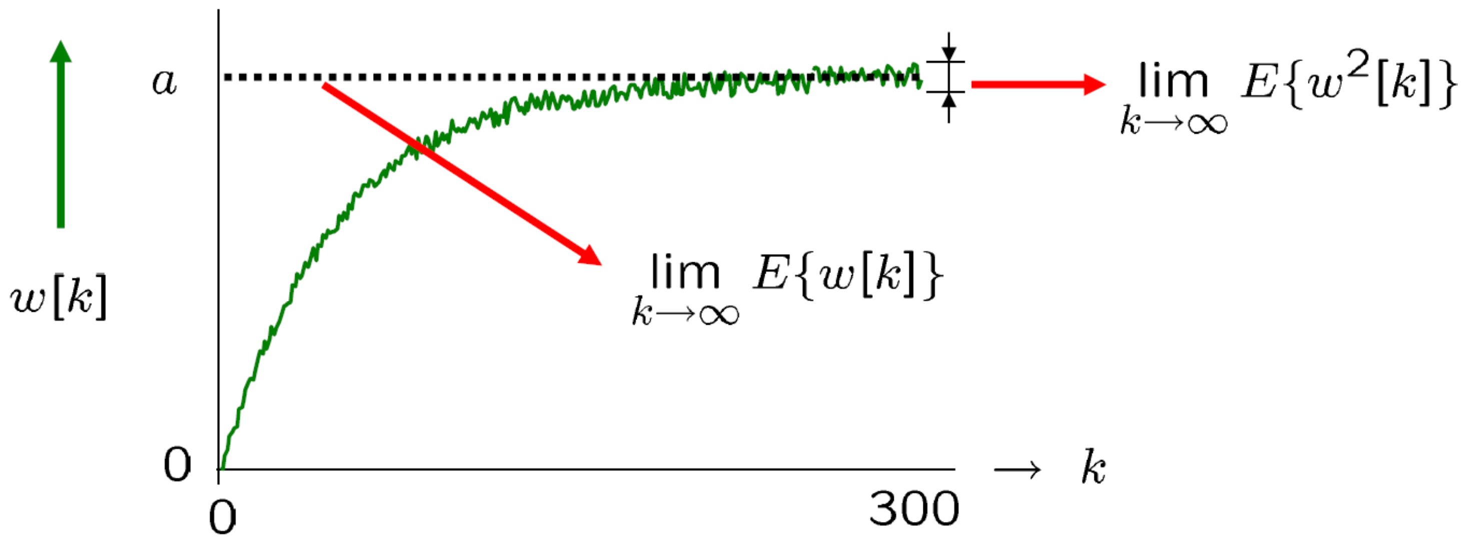
Stationary case

Non stationary case

Convergence LMS

Consequence of not using $E\{\cdot\}$?

Example: LMS, $N = 1$, $w[0] = 0$ $w_o = a$



Questions about convergence:

$$\lim_{k\to\infty} E\{w[k]\} = w_o = a \text{ and } \lim_{k\to\infty} E\{w^2[k]\} < \infty?$$

ADSP

Convergence LMS

First compare difference with (optimal) Wiener weights:

$$\underline{\mathbf{d}}[k] = \underline{\mathbf{w}}[k] - \underline{\mathbf{w}}_o \quad \text{with} \quad \underline{\mathbf{w}}_o = \mathbf{R}_x^{-1} \cdot \underline{\mathbf{r}}_{ex}$$

$$\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha \left(\underline{\mathbf{x}}[k] e[k] - \underline{\mathbf{x}}[k] \underline{\mathbf{x}}^t[k] \underline{\mathbf{w}}[k] \right)$$

$$\underline{\mathbf{w}}[k+1] - \underline{\mathbf{w}}_o = \left(\mathbf{I} - 2\alpha \underline{\mathbf{x}}[k] \underline{\mathbf{x}}^t[k] \right) \underline{\mathbf{w}}[k] - \underline{\mathbf{w}}_o + 2\alpha \underline{\mathbf{x}}[k] e[k]$$

$$\underline{\mathbf{d}}[k+1] = \left(\mathbf{I} - 2\alpha \underline{\mathbf{x}}[k] \underline{\mathbf{x}}^t[k] \right) \underline{\mathbf{d}}[k] + 2\alpha \underline{\mathbf{x}}[k] r_{min}[k]$$

with $r_{min}[k] = e[k] - \underline{\mathbf{x}}^t[k] \underline{\mathbf{w}}_o$

Convergence LMS

Convergence in the mean:

$$
\begin{aligned}
E\{\underline{\mathbf{d}}[k+1]\} &= E\left\{\left(\mathbf{I} - 2\alpha\underline{\mathbf{x}}[k]\underline{\mathbf{x}}^t[k]\right)\underline{\mathbf{d}}[k]\right\} \\
&\quad + 2\alpha\left(\underbrace{E\{\underline{\mathbf{x}}[k]e[k]\} - E\{\underline{\mathbf{x}}[k]\underline{\mathbf{x}}^t[k]\}\underline{\mathbf{w}}_o}_{0}\right)
\end{aligned}
$$

With independence assumption:

$$E\{\underline{\mathbf{x}}[k]\underline{\mathbf{x}}^t[k]\underline{\mathbf{d}}[k]\} \approx E\{\underline{\mathbf{x}}[k]\underline{\mathbf{x}}^t[k]\} \cdot E\{\underline{\mathbf{d}}[k]\}$$

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

Average convergence behaviour LMS same as SGD

$$0 < \alpha < 1/\lambda_{max} \; : \quad \lim_{k\to\infty} E\{\underline{\mathbf{w}}[k]\} = \underline{\mathbf{w}}_o \; ; \; \tau_{av,i} \approx 1/2\alpha\lambda_i$$

$\boxed{\Rightarrow \text{ Depends on coloration input process!}}$

Convergence LMS

Mean-square convergence:

$$J_{LMS} = E\{r^2\} = E\{(e - \underline{\mathbf{w}}^t\underline{\mathbf{x}})(e - \underline{\mathbf{x}}^t\underline{\mathbf{w}})\}$$

With $\underline{\mathbf{d}} = \underline{\mathbf{w}} - \underline{\mathbf{w}}_o$ or $\underline{\mathbf{w}} = w_o + \underline{\mathbf{d}} \quad \Rightarrow$

$$J_{LMS} = E\{\left((e - \underline{\mathbf{w}}_o^t\underline{\mathbf{x}}) - \underline{\mathbf{d}}^t\underline{\mathbf{x}}\right)\left((e - \underline{\mathbf{x}}^t\underline{\mathbf{w}}_o) - \underline{\mathbf{x}}^t\underline{\mathbf{d}}\right)\}$$

$$J_{LMS} = E\{(e - \underline{\mathbf{w}}_o^t\underline{\mathbf{x}})(e - \underline{\mathbf{x}}^t\underline{\mathbf{w}}_o)\} + E\{\underline{\mathbf{d}}^t\underline{\mathbf{x}}\underline{\mathbf{x}}^t\underline{\mathbf{d}}\}$$
$$-E\{\underline{\mathbf{d}}^t\underline{\mathbf{x}}(e - \underline{\mathbf{x}}^t\underline{\mathbf{w}}_o)\} - E\{(e - \underline{\mathbf{w}}_o^t\underline{\mathbf{x}})\underline{\mathbf{x}}^t\underline{\mathbf{d}}\}$$

Independence assumption $\Rightarrow$

$$E\{\underline{\mathbf{d}}^t\underline{\mathbf{x}}(e - \underline{\mathbf{x}}^t\underline{\mathbf{w}}_o)\} \approx E\{\underline{\mathbf{d}}^t\} \cdot \left(E\{\underline{\mathbf{x}}e\} - E\{\underline{\mathbf{x}}\underline{\mathbf{x}}^t\}\underline{\mathbf{w}}_o\right)$$
$$= E\{\underline{\mathbf{d}}^t\} \cdot (\underline{\mathbf{r}}_{ex} - \mathbf{R}_x\underline{\mathbf{w}}_o) = 0$$

Similar $E\{(e - \underline{\mathbf{w}}_o^t\underline{\mathbf{x}})\underline{\mathbf{x}}^t\underline{\mathbf{d}}\} \rightarrow 0$

Convergence LMS

Compare with MMSE expression

$$\Rightarrow \; J_{LMS} \; \approx \; \underbrace{E\{(e - \underline{\mathbf{w}}_o^t \underline{\mathbf{x}})(e - \underline{\mathbf{x}}^t \underline{\mathbf{w}}_o)\}}_{\text{Fixed}} + \underbrace{E\{\underline{\mathbf{d}}^t \underline{\mathbf{x}}\underline{\mathbf{x}}^t \underline{\mathbf{d}}\}}_{\text{Adaptive}}$$

Fixed *Adaptive*

Wiener error: $J_{min} = E\{(e - \underline{\mathbf{w}}_o^t \underline{\mathbf{x}})(e - \underline{\mathbf{x}}^t \underline{\mathbf{w}}_o)\} = E\{r_{min}^2\}$

Excess error:
$$\begin{aligned} J_{ex} \; &= \; E\{\underline{\mathbf{d}}^t \underline{\mathbf{x}}\underline{\mathbf{x}}^t \underline{\mathbf{d}}\} \approx E\{\underline{\mathbf{d}}^t E\{\underline{\mathbf{x}}\underline{\mathbf{x}}^t\} \underline{\mathbf{d}}\} \\ &= \; E\{\underline{\mathbf{d}}^t \mathbf{R}_x \underline{\mathbf{d}}\} \end{aligned}$$

Dynamic behaviour adaptive filter: $\tilde{J}[k] = \dfrac{J_{ex}[k]}{J_{min}[k]}$

Depends on coloration input process!

Convergence LMS

Conclusion convergence LMS:

Convergence gradient based algorithms, like SGD heavily relates on corolation input process and initiation of adaptive weights!

This also follows from rewriting gradient as:

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

$\Rightarrow$ Gradient (=update) depends on coloration input

Solution:

Alternative that decorrelates input $\rightarrow$ Newton

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

ADSP

Newton

Principle: Undo coloration effect SGD

SGD:

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

Newton:

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

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

Note: General update rule $\underline{\mathbf{w}} = \underline{\mathbf{w}} - \alpha\underline{\mathbf{U}}$

$\underline{\mathbf{U}}$ must be such that each iteration J decreases

Newton

Convergence Newton algorithm:

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

Conclusion:

For $|1-2\alpha| < 1 \;\; \Leftrightarrow 0 < \alpha < 1$

$$\lim_{k\to\infty} E\{\underline{\mathbf{d}}[k]\} = \underline{\mathbf{0}} \;\; \Leftrightarrow \;\; \lim_{k\to\infty} E\{\underline{\mathbf{w}}[k]\} = \underline{\mathbf{w}}_o = \mathbf{R}_x^{-1}\underline{\mathbf{r}}_{ex}$$

也可以不同期望，同期望为了和LMS对比

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

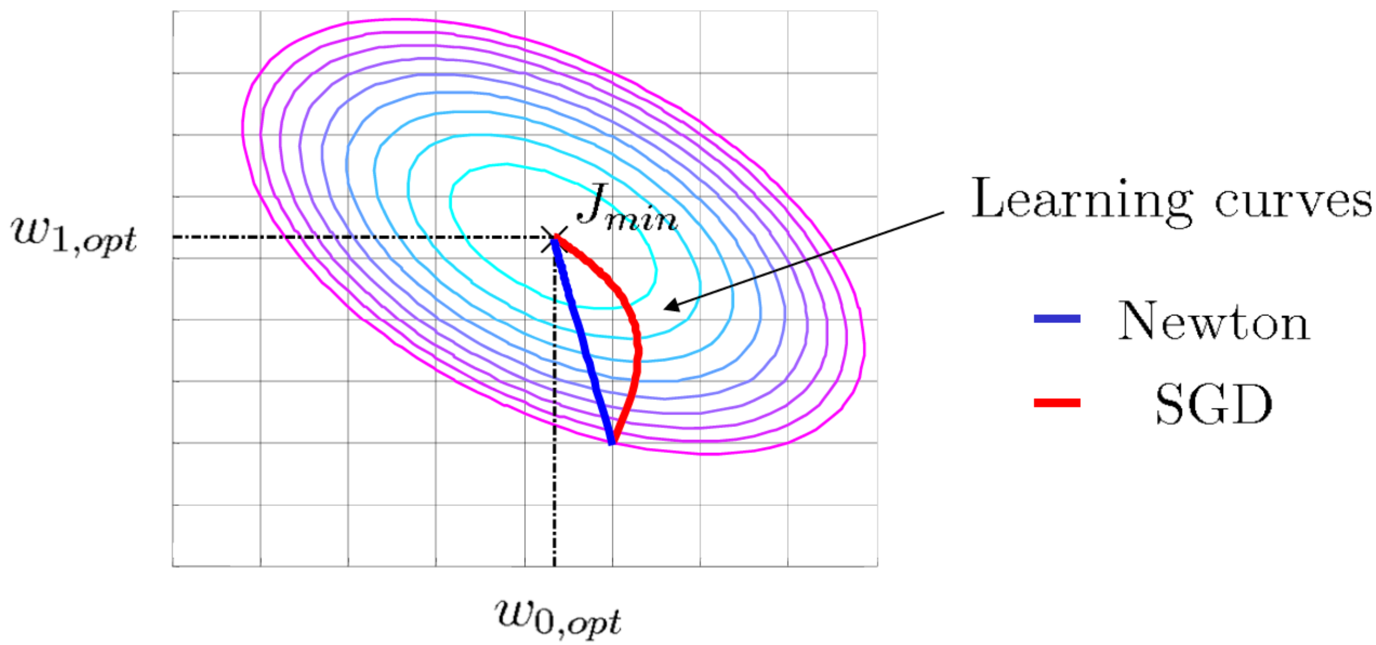
白化

- All weights have same convergence!
- Equivalent to SGD with white noise input!

Newton

Learning curves Newton vs. SGD in contour plot

Coloured input process with $\Gamma_x = \lambda_{max}/\lambda_{min} = 3$



Note: SGD-curve each iteration orthogonal to contourplot J

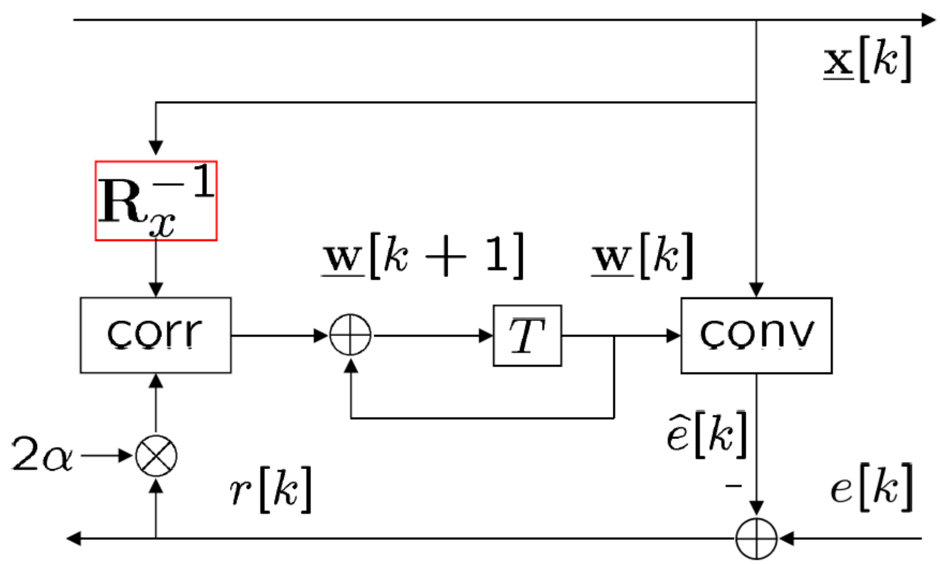
Newton-curve point each iteration towards J_{min}

Newton

Another view:

By replacing $\underline{\nabla} \rightarrow \underline{\hat{\nabla}}_{LMS} = \underline{\mathbf{x}}[k]r[k]$

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



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

Newton

Practical problems Newton: 收敛快.精.需要导数的

每一步计算复杂

Autocorrelation matrix $\mathbf{R}_x$:

- Not known in general
- May change in time (non-stationary process)
- Inversion is very expensive (many MIPS)

Complexity Newton: Huge

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

$\Rightarrow$ RLS; FDAF; etc.

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

RLS

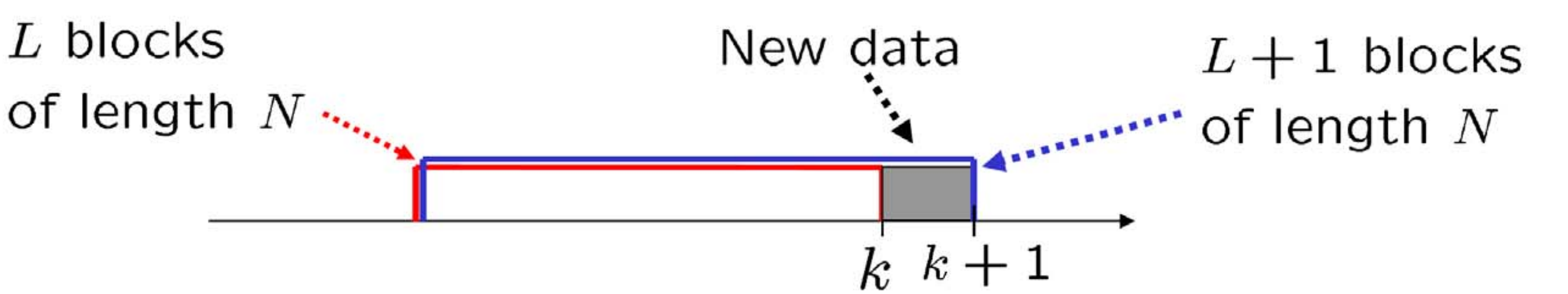
Recursive Least Squares

For L fixed, least squares problem becomes:

$$\min_{\underline{\mathbf{w}}} |\underline{\mathbf{e}} - \mathbf{X} \cdot \underline{\mathbf{w}}|^2 \Rightarrow \underline{\mathbf{w}}_{LS} = (\mathbf{X}^t \mathbf{X})^{-1} \cdot (\mathbf{X}^t \underline{\mathbf{e}})$$

RLS concept for $k \to k+1$:

Find recursive (=adaptive) solution for LS problem



RLS

Current solution (time k) :

Based on L data vectors, each of length N

$$\begin{aligned}\underline{\mathbf{w}}^{L}_{LS}[k] &= (\overline{\mathbf{R}}^{L}_{x}[k])^{-1}\cdot \underline{\overline{\mathbf{r}}}^{L}_{ex}[k]\\ &= \Big((\mathbf{X}^{L}[k])^{t}\mathbf{X}^{L}[k]\Big)^{-1}\cdot (\mathbf{X}^{L}[k])^{t}\underline{\mathbf{e}}^{L}[k]\end{aligned}$$

$$\mathbf{X}^{L}[k] = \begin{pmatrix} \underline{\mathbf{x}}^{t}[k] \\ \underline{\mathbf{x}}^{t}[k-1] \\ \vdots \\ \underline{\mathbf{x}}^{t}[k-L+1] \end{pmatrix} \quad \underline{\mathbf{e}}^{L}[k] = \begin{pmatrix} e[k] \\ e[k-1] \\ \vdots \\ e[k-L+1] \end{pmatrix}$$

Similar result for L+1

RLS

Compute solution at time $k+1$:

$$\begin{aligned}\underline{\mathbf{w}}_{LS}^{L+1}[k+1] &= (\overline{\mathbf{R}}_x^{L+1}[k+1])^{-1} \cdot \underline{\overline{\mathbf{r}}}_{ex}^{L+1}[k+1] \\ &= \left((\mathbf{X}^{L+1}[k+1])^t \mathbf{X}^{L+1}[k+1]\right)^{-1} \cdot \\ &\quad \cdot (\mathbf{X}^{L+1}[k+1])^t \underline{\mathbf{e}}^{L+1}[k+1]\end{aligned}$$

With $\quad \mathbf{X}^{L+1}[k+1] = \begin{pmatrix} \underline{\mathbf{x}}^t[k+1] \\ \underline{\mathbf{x}}^t[k] \\ \vdots \\ \underline{\mathbf{x}}^t[k-L+1] \end{pmatrix}$

$$\underline{\mathbf{e}}^{L+1}[k+1] = (e[k+1], e[k], \cdots, e[k-L+1])^t$$

RLS

Observe:

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

$$\overline{\underline{\mathbf{r}}}_{ex}^{L+1}[k+1] = \sum_{i=0}^{L} \underline{\mathbf{x}}[k+1-i]e[k+1-i]$$
$$= \overline{\underline{\mathbf{r}}}_{ex}^{L}[k] + \underline{\mathbf{x}}[k+1]e[k+1]$$

From matrix inversion lemma (see Appendix): $\Rightarrow$

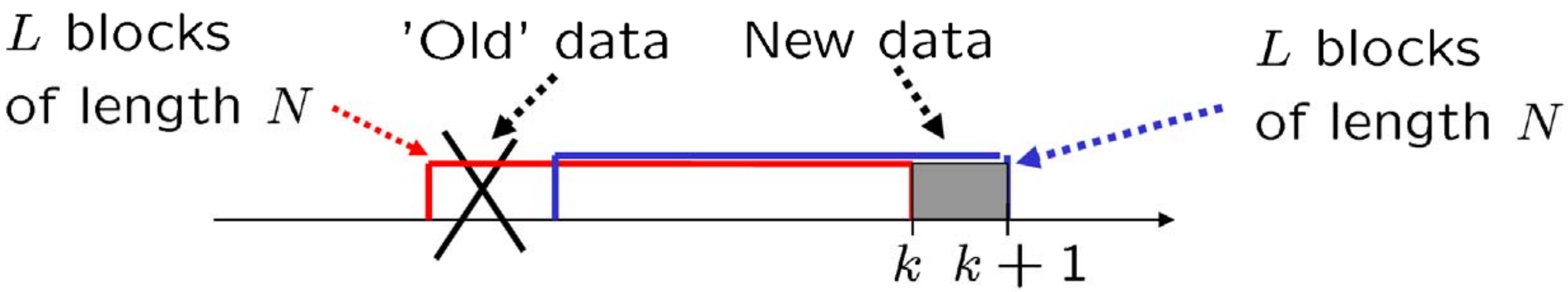
$$\overline{\mathbf{R}}_x^{-1}[k+1] = \overline{\mathbf{R}}_x^{-1}[k] - \frac{\overline{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]\underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}_x^{-1}[k]}{1+\underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]}$$

Finally: $\underline{\mathbf{w}}[k+1] = \overline{\mathbf{R}}_x^{-1}[k+1] \cdot \overline{\underline{\mathbf{r}}}_{ex}[k+1]$

RLS

For adaptivity $\rightarrow$ more effective data window

Sliding window: Keep window length L constant



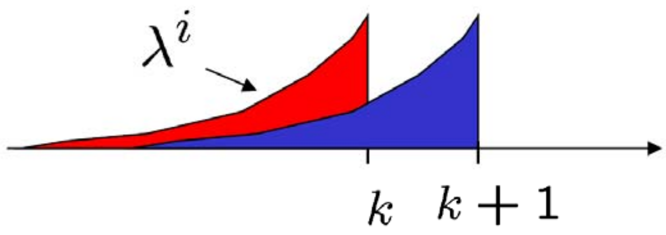
Note: Now we can write for autocorrelation

$$\overline{\mathbf{R}}_x[k+1] = \overline{\mathbf{R}}_x[k] - \underline{\mathbf{x}}[k-L+1] \cdot \underline{\mathbf{x}}^t[k-L+1] + \underline{\mathbf{x}}[k+1] \cdot \underline{\mathbf{x}}^t[k+1]$$

$\Rightarrow$ Still very complex

RLS

Exponential window: Scale down data by factor λ



$0 < \lambda < 1$: forget factor

$\dfrac{1}{1-\lambda}$: 'memory' of algorithm

$$\mathbf{X}[k] = \begin{pmatrix} \lambda^0 \underline{\mathbf{x}}^t[k] \\ \lambda^1 \underline{\mathbf{x}}^t[k-1] \\ \vdots \\ \lambda^k \underline{\mathbf{x}}^t[0] \end{pmatrix} \quad \underline{\mathbf{e}}[k] = \begin{pmatrix} \lambda^0 e[k] \\ \lambda^1 e[k-1] \\ \vdots \\ \lambda^k e[0] \end{pmatrix}$$

$$J[k] = \sum_{i=0}^{k} \lambda^i r^2[k-i] = \left(\underline{\mathbf{e}}^t[k] - \underline{\mathbf{w}}^t[k]\mathbf{X}^t[k]\right)\left(\underline{\mathbf{e}}[k] - \mathbf{X}[k]\underline{\mathbf{w}}[k]\right)$$

? 此处存疑

RLS

Observe:

$$\overline{\mathbf{R}}_x[k+1] = \lambda^2\overline{\mathbf{R}}_x[k] + \underline{\mathbf{x}}[k+1]\cdot\underline{\mathbf{x}}^t[k+1]$$

$$\overline{\underline{\mathbf{r}}}_{ex}[k+1] = \lambda^2\overline{\underline{\mathbf{r}}}_{ex}[k] + \underline{\mathbf{x}}^t[k+1]e[k+1]$$

From matrix inversion theorem (see Appendix):

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

with gain vector: $\underline{\mathbf{g}}[k+1] = \dfrac{\overline{\mathbf{R}}^{-1}[k]\underline{\mathbf{x}}[k+1]}{\lambda^2 + \underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}^{-1}[k]\underline{\mathbf{x}}[k+1]}$

New weight vector:

$$\underline{\mathbf{w}}[k+1] = \overline{\mathbf{R}}_x^{-1}[k+1]\cdot\overline{\underline{\mathbf{r}}}_{ex}[k+1]$$

RLS

RLS algorithm

Initialization:

$$\underline{\overline{\mathbf{r}}}_{ex}[0] = \underline{\mathbf{0}} \; ; \; \overline{\mathbf{R}}_x^{-1}[0] = \delta^{-1}\mathbf{I} \quad \text{with } \delta \text{ small}$$

For $k \geq 0$:

$$\underline{\mathbf{g}}[k+1] = \frac{\overline{\mathbf{R}}^{-1}[k]\underline{\mathbf{x}}[k+1]}{\lambda^2 + \underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}^{-1}[k]\underline{\mathbf{x}}[k+1]}$$

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

$$\underline{\overline{\mathbf{r}}}_{ex}[k+1] = \lambda^2\underline{\overline{\mathbf{r}}}_{ex}[k] + \underline{\mathbf{x}}[k+1]e[k+1]$$

$$\underline{\mathbf{w}}[k+1] = \overline{\mathbf{R}}_x^{-1}[k+1] \cdot \underline{\overline{\mathbf{r}}}_{ex}[k+1]$$

上海科技大学 ShanghaiTech University

ADSP

RLS

Compare RLS with "LMS/Newton"

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

Update RLS can be rewritten as (see Appendix):

[No Title]

$$\begin{aligned} \underline{\mathbf{w}}[k+1] &= \underline{\mathbf{w}}[k] + \underline{\mathbf{g}}[k+1]\left(e[k+1] - \underline{\mathbf{x}}^t[k+1]\underline{\mathbf{w}}[k]\right) \\ &= \underline{\mathbf{w}}[k] + \underline{\mathbf{g}}[k+1] r[k+1] \end{aligned}$$

with gain vector:

$$\underline{\mathbf{g}}[k+1] = \frac{\overline{\mathbf{R}}^{-1}[k]\underline{\mathbf{x}}[k+1]}{\lambda^2 + \underline{\mathbf{x}}^t[k+1]\overline{\mathbf{R}}^{-1}[k]\underline{\mathbf{x}}[k+1]}$$

RLS

Notes on RLS:

$\underline{\mathbf{w}}[\infty] = \underline{\mathbf{w}}_o$

Complexity: $O(N^2)$ per time update

Window length increases when time increases!

Exhibits unstable roundoff error accumulation 通过平均二方法消除累积误差

RLS basis for many practical algorithms

Decorrelation takes place in algorithm

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

去白化

FDAF

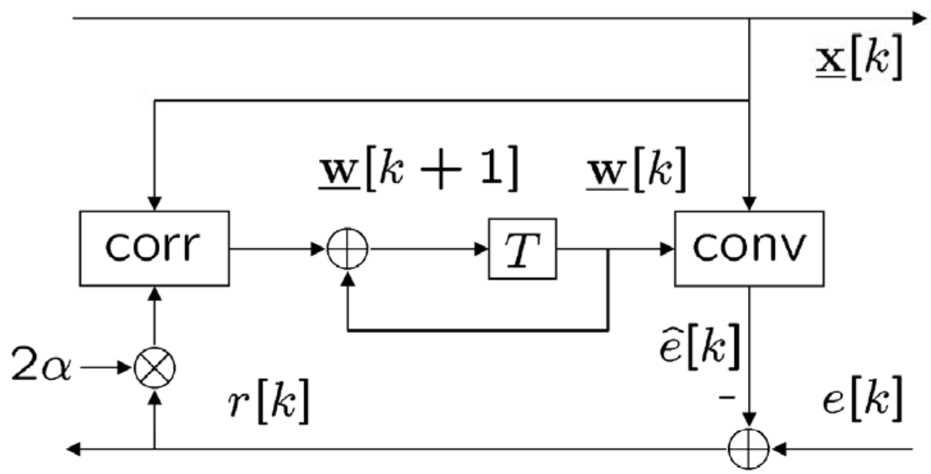
Frequency Domain Adaptive Filter

Alternative for LMS/Newton and RLS

FDAF

Frequency Domain Adaptive Filter

First translate LMS to frequency domain:



LMS weight update:

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

Filter output:

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

Part B: Adaptive signal processing

上海科技大学 ShanghaiTech University

ADSP

F是一个矩阵，左乘F 就是求傅利叶

matlab有自带命令 求F^{-1}

FDAF

Apply filter operation in frequency domain:

矩阵 ←
$$\mathbf{F}\cdot\underline{\mathbf{x}}[k] = \underline{\mathbf{X}}[k] = \left(X_0[k], X_1[k], \cdots, X_{N-1}[k]\right)^t$$

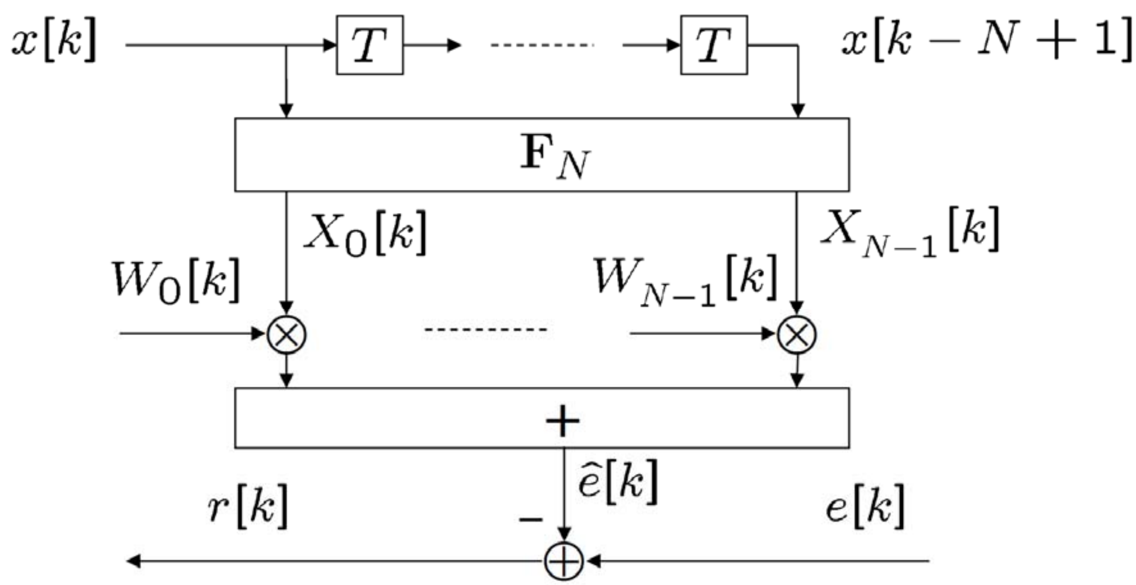
$$\mathbf{F}^{-1}\cdot\underline{\mathbf{w}}[k] = \underline{\mathbf{W}}[k] = \left(W_0[k], W_1[k], \cdots, W_{N-1}[k]\right)^t$$

$$Note:\ \mathbf{F}^{-1} = \frac{1}{N}\mathbf{F}^*$$

Filter output:

$$\begin{aligned}
\hat{e}[k] &= \sum_{i=0}^{N-1} x[k-i]w_i[k] = \underline{\mathbf{x}}^t[k]\cdot\underline{\mathbf{w}}[k] \\
&= \underline{\mathbf{x}}^t[k]\mathbf{F}\cdot\mathbf{F}^{-1}\underline{\mathbf{w}}[k] = (\mathbf{F}\underline{\mathbf{x}}[k])^t\cdot\left(\mathbf{F}^{-1}\underline{\mathbf{w}}[k]\right) \\
&= \underline{\mathbf{X}}^t[k]\cdot\underline{\mathbf{W}}[k] = \sum_{l=0}^{N-1} X_l[k]W_l[k]
\end{aligned}$$

FDAF



Notes:

- Weights perfom inverse transform
- Use DFT symmetry to reduce complexity
- Separate frequency bins 'uncorrelated' (large N)

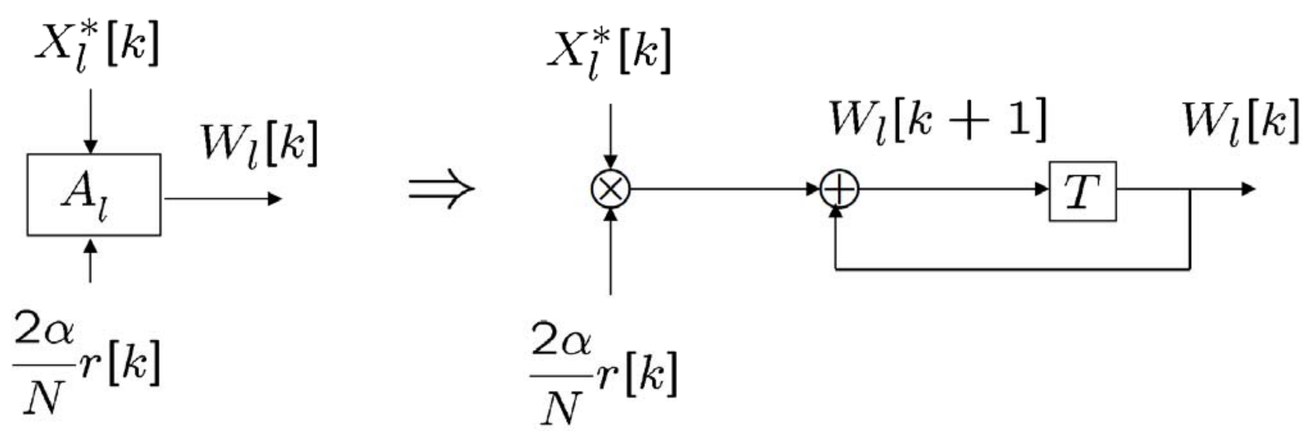
FDAF

Apply LMS update in frequency domain:

Multiply both sides by $\mathbf{F}^{-1}$ $\Rightarrow$

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

$$\underline{\mathbf{W}}[k+1] = \underline{\mathbf{W}}[k] + \frac{2\alpha}{N}\underline{\mathbf{X}}^*[k]r[k]$$



FDAF

Improve convergence properties easily by

Decorrelation by power normalization:

FDAF algorithm:

$$\underline{\mathbf{W}}[k+1] = \underline{\mathbf{W}}[k] + \frac{2\alpha}{N}\mathbf{P}^{-1}\underline{\mathbf{X}}^*[k]r[k]$$

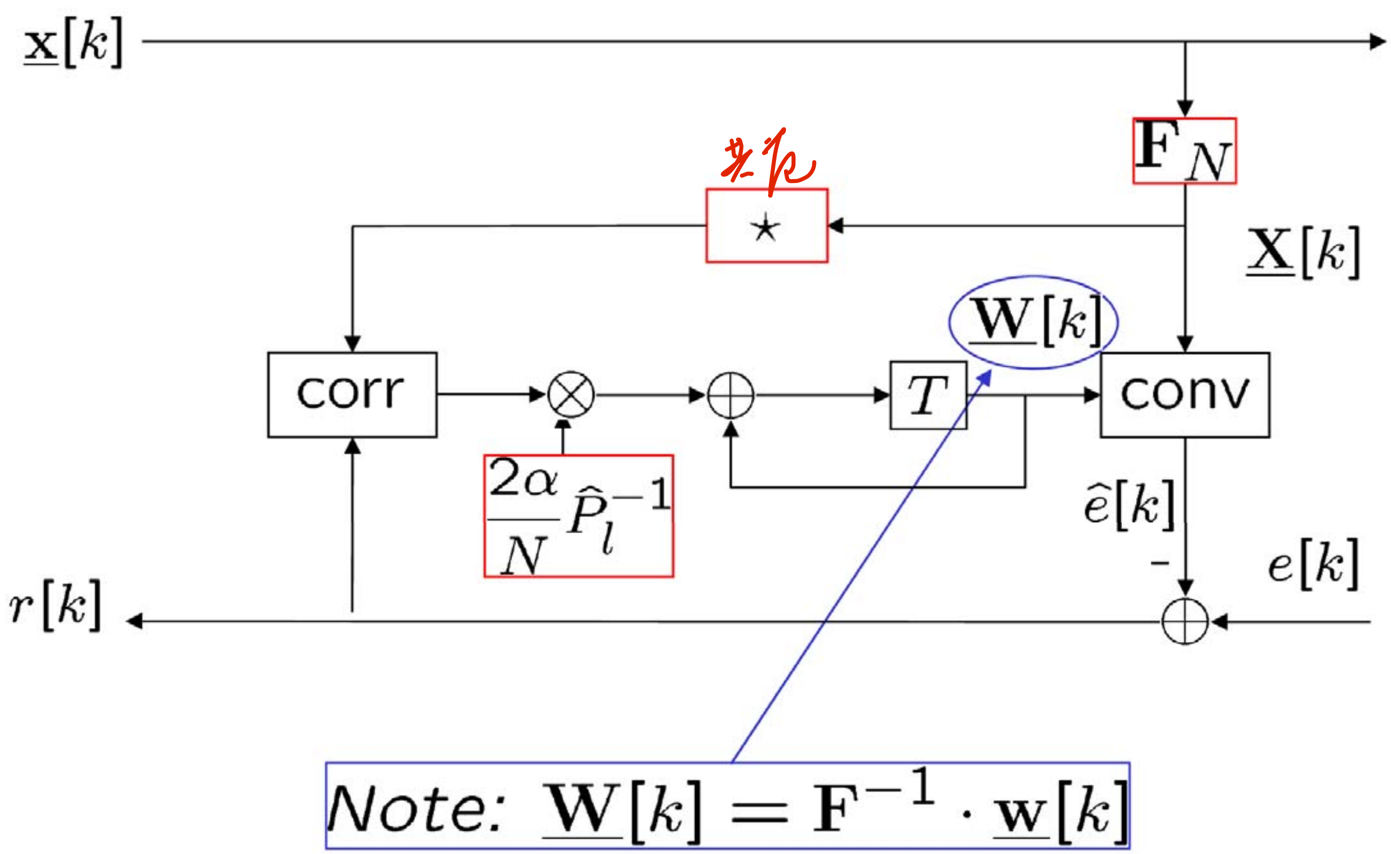
$$\mathbf{P} = diag\{\underline{\mathbf{P}}\} \text{ with } P_l = \frac{1}{N}E\{|X_l[k]|^2\}$$

In practice (e.g.):

$$\widehat{P}_l[k+1] = \beta\widehat{P}_l[k] + (1-\beta)\frac{|X_l[k]|^2}{N}\ \forall l$$

FDAF

Simplified realization scheme:



FDAF

Average behaviour FDAF:

With $\underline{\mathbf{D}} = \mathbf{F}^{-1}\underline{\mathbf{d}} = \mathbf{F}^{-1}(\underline{\mathbf{w}} - \underline{\mathbf{w}}_o) = \underline{\mathbf{W}} - \underline{\mathbf{W}}_o$

FDAF can be rewritten as:

$$\underline{\mathbf{D}}[k+1] = \left(\mathbf{I} - \frac{2\alpha}{N}\mathbf{P}^{-1}\underline{\mathbf{X}}^*[k]\underline{\mathbf{X}}^t[k]\right)\underline{\mathbf{D}}[k] + \frac{2\alpha}{N}\mathbf{P}^{-1}\underline{\mathbf{X}}^*[k]r_{min}[k]$$

Different bins uncorrelated: $\Rightarrow \dfrac{E\{\underline{\mathbf{X}}^*[k]\underline{\mathbf{X}}^t[k]\}}{N} \approx \mathbf{P}$

Thus $E\{\underline{\mathbf{D}}[k+1]\} \approx (1 - 2\alpha)\, E\{\underline{\mathbf{D}}[k]\}$

FDAF

Notes FDAF:

- Both update algorithm and filter are transformed
- FFT (or DFT) is fixed transform --> easy but not exact

Conclusions FDAF:

$$\lim_{k\to\infty} E\{\underline{\mathbf{D}}[k]\} = \underline{\mathbf{0}} \quad \Leftrightarrow \quad \lim_{k\to\infty} E\{\underline{\mathbf{W}}[k]\} = \underline{\mathbf{W}}_o = \mathbf{F}^{-1}\underline{\mathbf{w}}_o$$

循环外
只求一次逆

$\mathbf{P}_x^{-1}$: update of each bin is power normalized

All weights have (in average) similar convergence!

Equivalent to NLMS with white noise input!

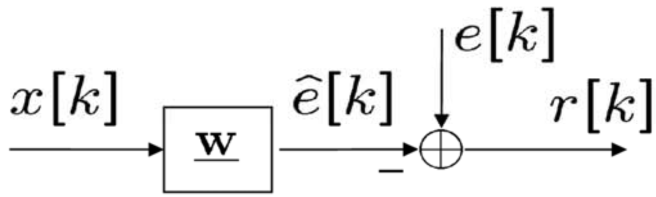
上海科技大学 ShanghaiTech University

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

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	$\underline{\mathbf{r}}_{ex} = E\{e[k] \cdot \underline{\mathbf{x}}[k]\}$	$\underline{\overline{\mathbf{r}}}_{ex} = \mathbf{X}^t \cdot \underline{\mathbf{e}}$
Error J	$E\{r^2[k]\}$	$\sum_{i=0}^{L-1} r^2[k-i]$
Criterion	$\min_{\underline{\mathbf{w}}}\{E\{r^2[k]\}\}$	$\min_{\underline{\mathbf{w}}} \lvert \underline{\mathbf{e}} - \mathbf{X} \cdot \underline{\mathbf{w}} \rvert^2$
Opt. solution $\underline{\mathbf{w}}_o$	$\mathbf{R}_x^{-1} \cdot \underline{\mathbf{r}}_{ex}$	$\overline{\mathbf{R}}_x^{-1} \cdot \underline{\overline{\mathbf{r}}}_{ex}$
Min. error J_{min}	$E\{e^2\} - \underline{\mathbf{r}}_{ex}^t \mathbf{R}_x^{-1} \underline{\mathbf{r}}_{ex}$	$\underline{\mathbf{e}}^t \underline{\mathbf{e}} - \underline{\overline{\mathbf{r}}}_{ex}^t \overline{\mathbf{R}}_x^{-1} \underline{\overline{\mathbf{r}}}_{ex}$

Summary

$$\text{Set of constraints: } \mathbf{C}^t \cdot \underline{\mathbf{w}} = \underline{\mathbf{f}}$$

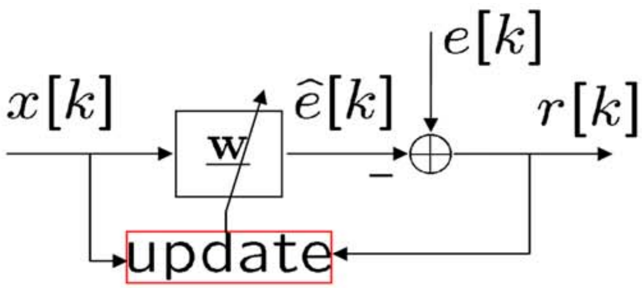
$$\text{Solution for } N \geq M\text{: } \underline{\mathbf{w}}^c = \mathbf{C}\left(\mathbf{C}^t\mathbf{C}\right)^{-1}\underline{\mathbf{f}}$$

Solution for $N > M$ with MMSE:

$$\underline{\mathbf{w}}_o^c = \underline{\mathbf{w}}_o + \mathbf{R}_x^{-1}\mathbf{C}\left(\mathbf{C}^t\mathbf{R}_x^{-1}\mathbf{C}\right)^{-1}\left(\underline{\mathbf{f}} - \mathbf{C}^t\underline{\mathbf{w}}_o\right)$$

$$\text{Equivalently: } \underline{\mathbf{w}}_o^c = \mathbf{R}_x^{-1}\mathbf{C}\left(\mathbf{C}^t\mathbf{R}_x^{-1}\mathbf{C}\right)^{-1}\underline{\mathbf{f}}$$

Summary



Simple adaptive algorithms (no decorrelation):

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

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

Constrained LMS : $\mathrm{C}^t \cdot \underline{\mathrm{w}} = \underline{\mathrm{f}}$

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

$$\tilde{\mathbf{P}} = \mathbf{I} - \mathbf{C}\left(\mathbf{C}^t\mathbf{C}\right)^{-1}\mathbf{C}^t \text{ and } \underline{\mathbf{w}}[0] = \mathbf{C}\left(\mathbf{C}^t\mathbf{C}\right)^{-1}\underline{\mathbf{f}}$$

Summary

Algorithms with improved convergence:

$$
\begin{aligned}
\text{"LMS/Newton"} &: \quad \underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha \mathbf{R}_x^{-1} \cdot \underline{\mathbf{x}}[k] r[k] \\
\text{Newton} &: \quad \underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha \mathbf{R}_x^{-1} \cdot (\underline{\mathbf{r}}_{ex} - \mathbf{R}_x \underline{\mathbf{w}}[k]) \\
\text{RLS} &: \quad \underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + \underline{\mathbf{g}}[k+1] r[k+1] \\
\text{with} & \quad \underline{\mathbf{g}}[k+1] = \frac{\overline{\mathbf{R}}^{-1}[k] \underline{\mathbf{x}}[k+1]}{\lambda^2 + \underline{\mathbf{x}}^t[k+1] \overline{\mathbf{R}}^{-1}[k] \underline{\mathbf{x}}[k+1]} \\
\text{FDAF} &: \quad \underline{\mathbf{W}}[k+1] = \underline{\mathbf{W}}[k] + \frac{2\alpha}{N} \mathbf{P}^{-1} \underline{\mathbf{X}}^*[k] r[k] \\
\text{etc.} &
\end{aligned}
$$

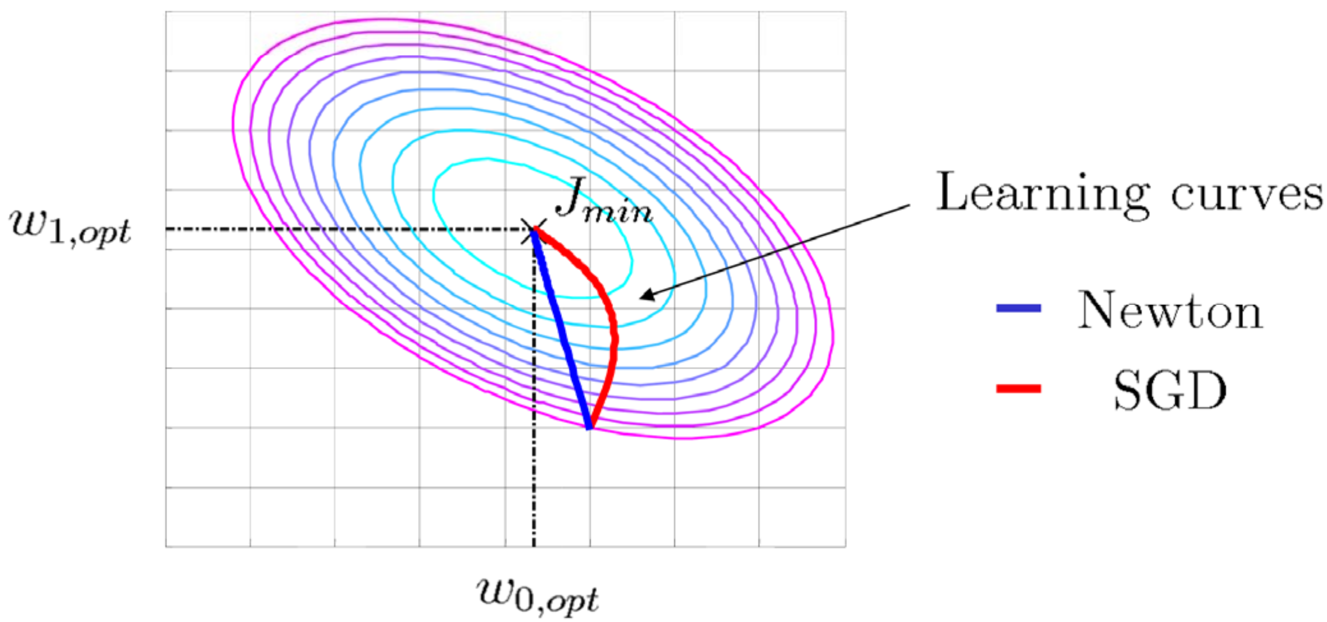
上海科技大学 ShanghaiTech University

ADSP

Summary

Learning curves Newton vs. SGD in contour plot

Coloured input process with $\Gamma_x = \lambda_{max}/\lambda_{min} = 3$



Note: SGD-curve each iteration orthogonal to contourplot J

Newton-curve point each iteration towards J_{min}