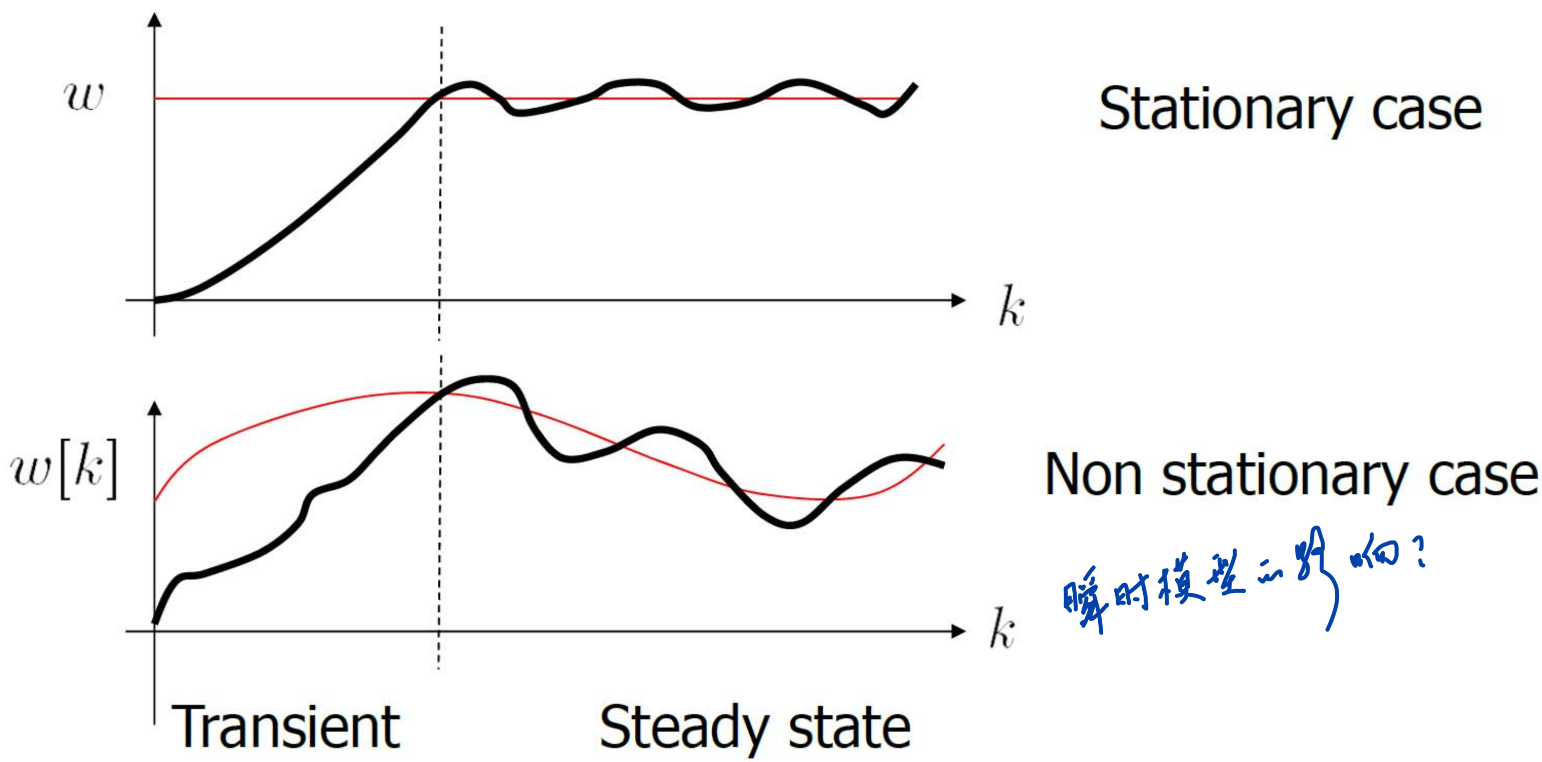
Advanced Digital Signal Processing (ADSP)

徐林

上海科技大学
ShanghaiTech University

Convergence LMS

Acquisition and tracking:



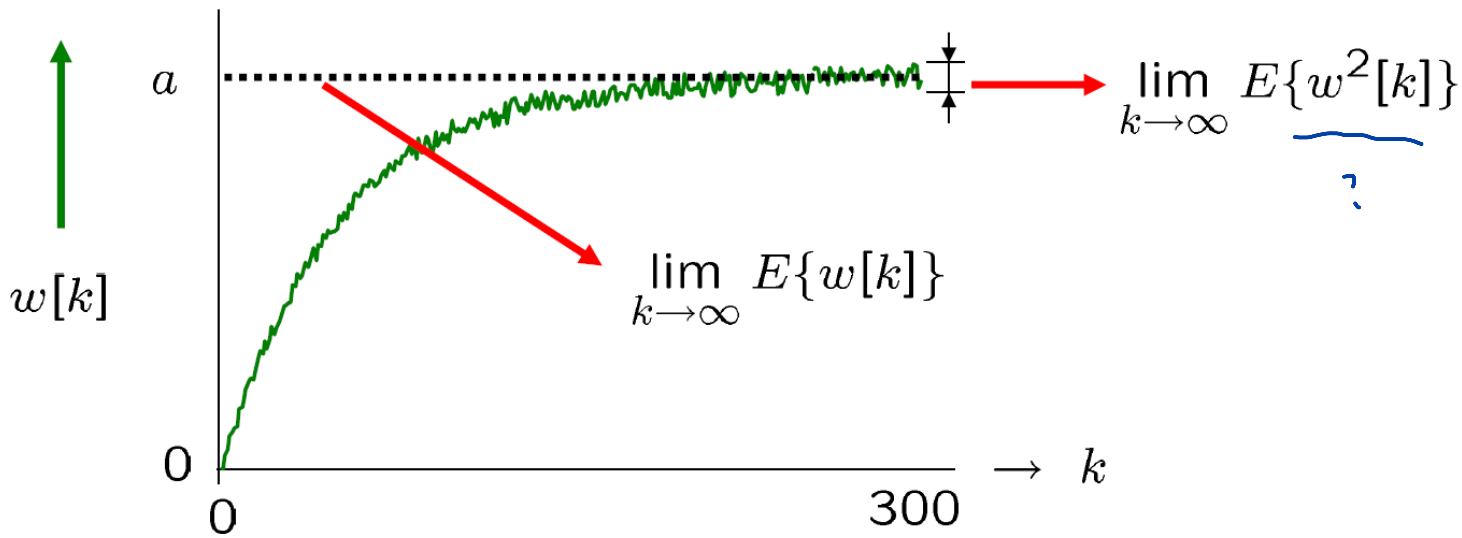
瞬时模型不好啊？

ADSP

Convergence LMS

Consequence of not using $E\{\cdot\}$? 不使用E[·]的推论

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



Questions about convergence:

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

Convergence LMS

First compare difference with (optimal) Wiener weights:

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

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

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

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

$- 2\alpha \underline{x}[k] \underline{x}^t[k] \underline{w}_o + 2\alpha \underline{x}[k] e[k]$

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

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

$\hat{e}_{min}$

ADSP

Convergence LMS

迭代：SGD: $\underline{d}[k+1] = (I - 2\alpha R_x)\underline{d}[k]$; $\underline{d}[k] = (I - 2\alpha R_x)^k \underline{d}[0]$

Convergence in the mean: Why use mean $E[\cdot]$?

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

0

$r_{min} \perp \underline{x}$

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

With independence assumption:

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

$k \to \infty$ $(1 - 2\alpha\lambda_i)^k = 0$

$\Rightarrow 0 < |1 - 2\alpha\lambda_i| < 1$

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

for all i: $\alpha < \frac{1}{\lambda_i} \Rightarrow \alpha < \frac{1}{\lambda_{max}}$

开始迭代：$E\{\underline{d}[k]\} = (I - 2\alpha R_x)^k E\{\underline{d}[0]\}$

Average convergence behaviour LMS same as SGD

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

$\Rightarrow$ Depends on coloration input process!

$\ln(e^{-\frac{k}{\tau_i}}) = \ln(1 - 2\alpha\lambda_i)^k$

白信号和有色信号不同点在于？

$\Rightarrow \tau_i = -\frac{1}{\ln(1-2\alpha\lambda_i)} \sim \frac{1}{2\alpha\lambda_i}$

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 $\underline{\mathbf{w}} = w_o + \underline{\mathbf{d}} \;\;\Rightarrow$

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

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

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

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

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: $J_{ex} \; = \; E\{\underline{\mathbf{d}}^t \underline{\mathbf{x}\mathbf{x}}^t \underline{\mathbf{d}}\} \approx E\{\underline{\mathbf{d}}^t E\{\underline{\mathbf{x}\mathbf{x}}^t\} \underline{\mathbf{d}}\}$

$$= \; E\{\underline{\mathbf{d}}^t \mathbf{R}_x \underline{\mathbf{d}}\}$$

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

Depends on coloration input process!

ADSP

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:

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

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

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

Solution:

Alternative that decorrelates input $\rightarrow$ Newton

输入去相关

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

Newton

取消彩色之影响

Principle: Undo coloration effect SGD

SGD:

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

Newton:

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

白化

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

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

上海科技大学 ShanghaiTech University

ADSP

Newton

Convergence Newton algorithm: 消除了R_x对收敛的影响

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

Conclusion:

$\underline{d}[k] = (1-2\alpha)^k \underline{d}[0]$

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

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

用E{·}是为了和"LMS"保持一致进行对比

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

w的每一个元素

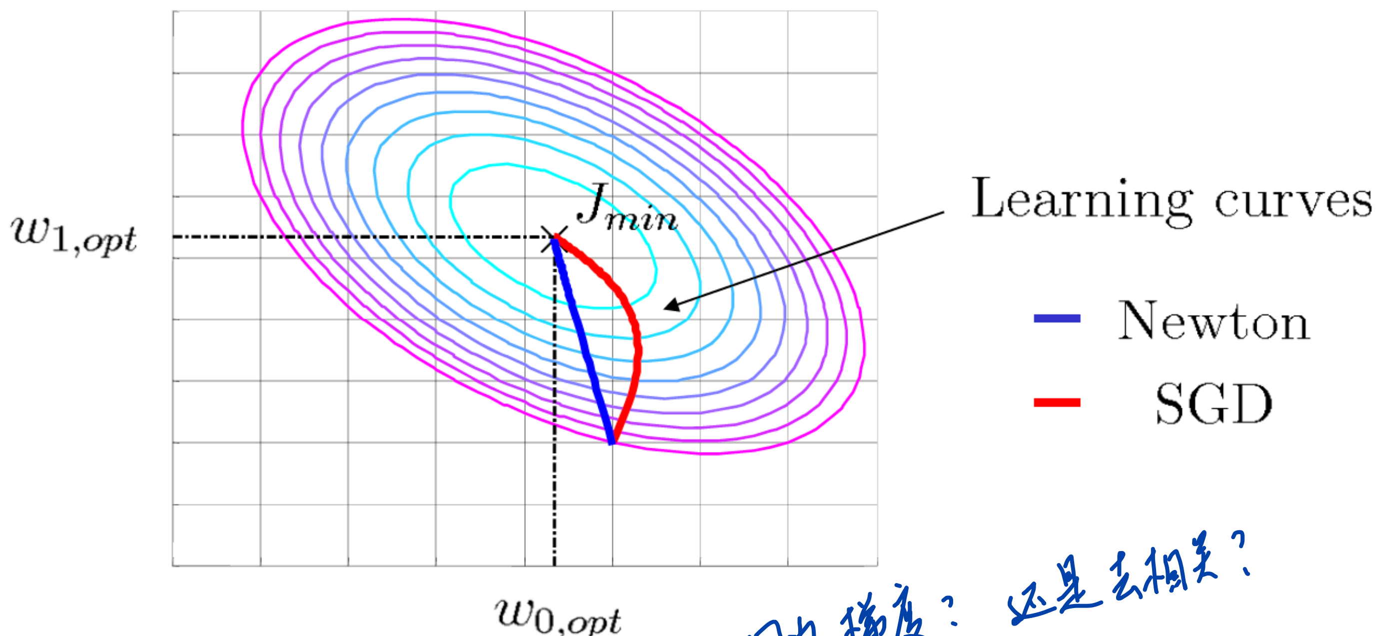
- All weights have same convergence!
- Equivalent to SGD with white noise input!

R_x的特征值相等?

Newton

Learning curves Newton vs. SGD in contour plot

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



因为梯度？还是去相关？

Note: SGD-curve each iteration orthogonal to contourplot J

Newton-curve point each iteration towards J_{min}

白化改变了梯度？

① first view: $\underline{\nabla} = \underline{\nabla}_{SGD} = -2(\underline{r}_{ex} - R_x \underline{w})$

ADSP

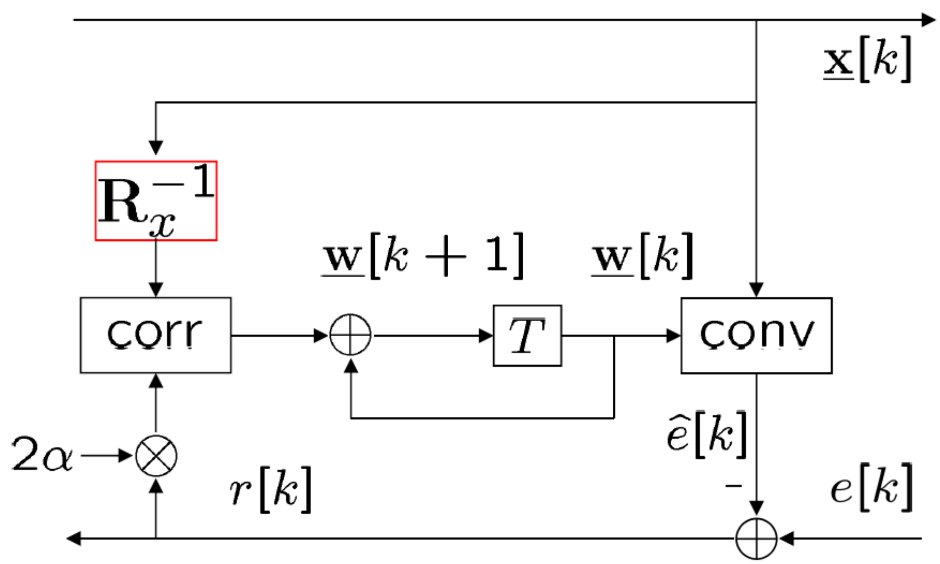
Newton

② *Another view:* $\hat{\underline{\nabla}}_{LMS} = -2\,\underline{x}[k]\,r[k]$

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

白化

$\Rightarrow$ "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

收敛快（需要的步数少）
每一步计算复杂

Practical problems Newton:

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.

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) 递归最小二乘
- Frequency Domain Adaptive Filter (FDAF)
- Summary

RLS

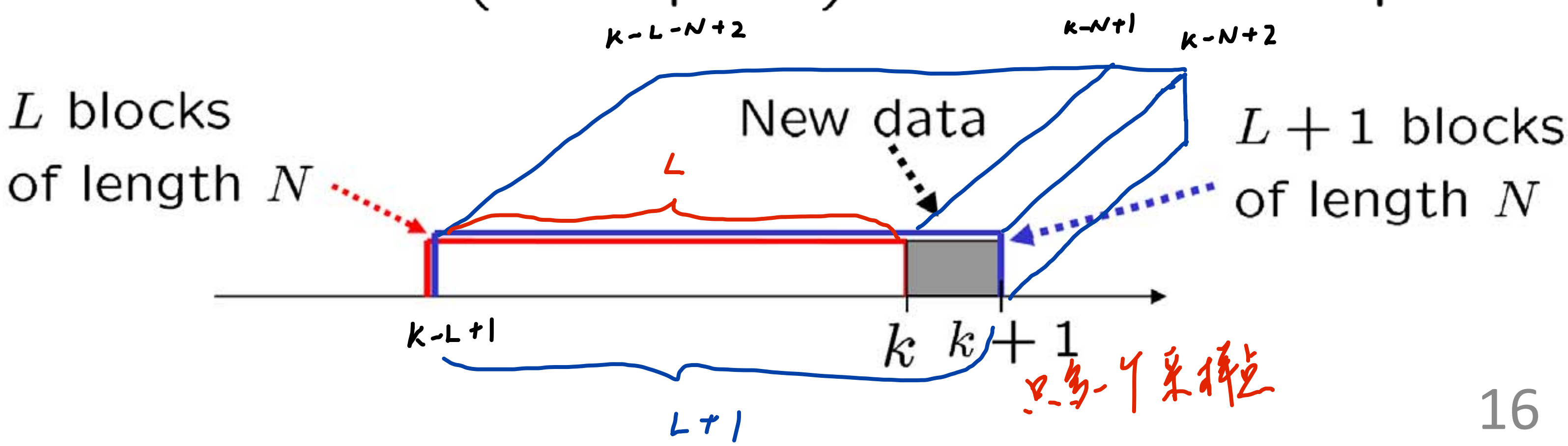
Recursive Least Squares

For L fixed, least squares problem becomes:

$$\min_{\underline{\mathbf{w}}} |\underline{\mathbf{e}} - \mathbf{X}\cdot\underline{\mathbf{w}}|^2 \;\Rightarrow\; \underline{\mathbf{w}}_{LS} = \underbrace{(\mathbf{X}^t\mathbf{X})^{-1}}_{\bar{R}_x^{-1}} \cdot \underbrace{(\mathbf{X}^t\underline{\mathbf{e}})}_{\bar{r}_{ex}}$$

RLS concept for $k \to k+1$: 每次迭代使用"L+1"blocks数据 (L+1)·N
除了 k=0时?

Find recursive (=adaptive) solution for LS problem



RLS

Current solution (time k) :

Based on L data vectors, each of length N

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

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

Similar result for L+1

"RLS" 原"LS"中求迭代

Compute solution at time $k+1$:

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

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

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

RLS

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

Observe:

$$\overline{\mathbf{R}}_x^{L+1}[k+1] = \sum_{i=0}^{L} \underline{\mathbf{x}}[k+1-i]\underline{\mathbf{x}}^t[k+1-i] \quad \text{共}(L+1)\text{项}$$

$X^{L+1}[k+1]^t X^{L+1}[k+1]$

$$\overset{\sum_{i=1}^{L}}{} = \overline{\mathbf{R}}_x^{L}[k] + \underbrace{\underline{\mathbf{x}}[k+1]\cdot\underline{\mathbf{x}}^t[k+1]}_{N\times N\ \text{矩阵}} \quad \to i=0$$

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

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

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

$N\cdot(L+1)\cdot(L+1)\cdot N$?

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

$N(L+1)\cdot(L+1)\cdot N$?

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

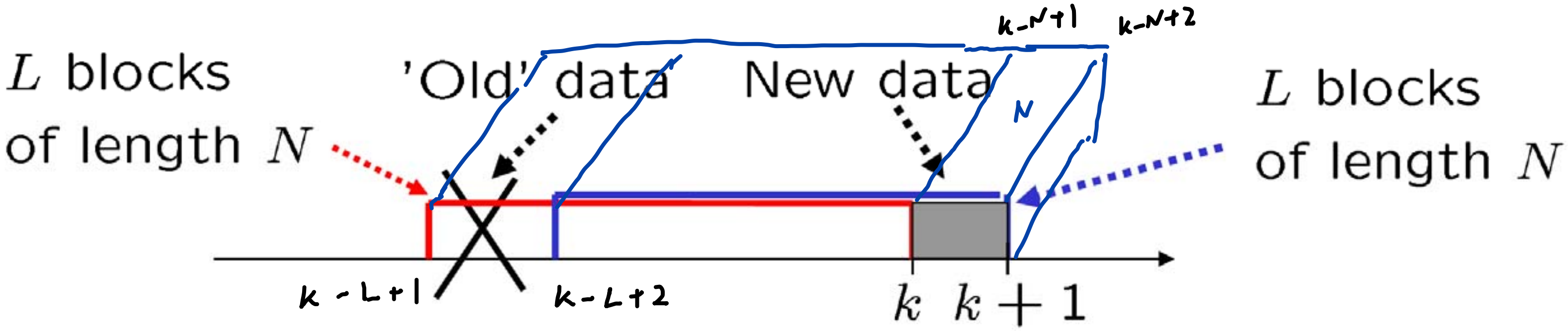
RLS

For adaptivity $\rightarrow$ more effective data window

每次迭代使用"L" blocks 数据

$L \cdot N$

Sliding window: Keep window length L constant



Note: Now we can write for autocorrelation

$N \cdot L \cdot L \cdot N$

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

$$\bar{R}_x^{L+1}[k+1] = \bar{R}_x^{L}[k] + \underline{x}[k+1]\,\underline{x}^t[k+1]$$

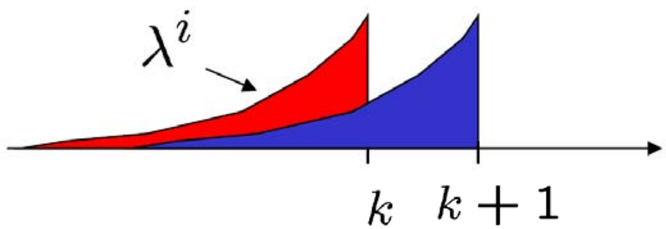
$N \cdot (L+1) \cdot (L+1) \cdot N$

$\Rightarrow$ Still very complex

RLS

Exponential window: Scale down data by factor λ

每次迭代使用数据量不同，第 i 次使用 (i+1) blocks 数据



$0 < \lambda < 1$: forget factor $(i+1)\cdot N$

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

$$\mathbf{X}[k] = \begin{pmatrix} \lambda^0 \underline{\mathbf{x}}^t[k] \\ \lambda^1 \underline{\mathbf{x}}^t[k-1] \\ \vdots \\ \underbrace{\lambda^k \underline{\mathbf{x}}^t[0]} \end{pmatrix}_{(k+1)\cdot N\ ?} \quad \underline{\mathbf{e}}[k] = \begin{pmatrix} \lambda^0 e[k] \\ \lambda^1 e[k-1] \\ \vdots \\ \lambda^k e[0] \end{pmatrix}_{(k+1)\cdot 1}$$

所有累积量均含于其中

$\lambda^{k-1} \underline{x}^t[1]$?

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

$$\bar{R}_x[k] = \sum_{i=0}^{k} \lambda^i \underline{x}[k-i] \cdot \lambda^i \underline{x}^t[k-i] = \sum_{i=0}^{k} \lambda^{2i} \underline{x}[k-i]\, \underline{x}^t[k-i]$$

$\bar{R}_x[k] = \sum_{i=0}^{k} \lambda^{2i} \underline{x}[k-i]\, \underline{x}^t[k-i]$

RLS

$\bar{R}_x[k+1] = \sum_{i=0}^{k+1} \lambda^{2i} \underline{x}[k+1-i]\, \underline{x}^t[k+1-i] = \sum_{i=1}^{k+1} [\cdot] + \sum_{i=0}^{0} \cdots$

$= \lambda^2 \bar{R}_x[k] + \lambda^0 \underline{x}[k+1]\, \underline{x}^t[k+1]$

Observe:

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

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

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:

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

RLS

RLS algorithm

Initialization:

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

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

误差的累积

通过平均的方法

消除累积误差

RLS basis for many practical algorithms

Decorrelation takes place in algorithm

去相关

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

FDAF

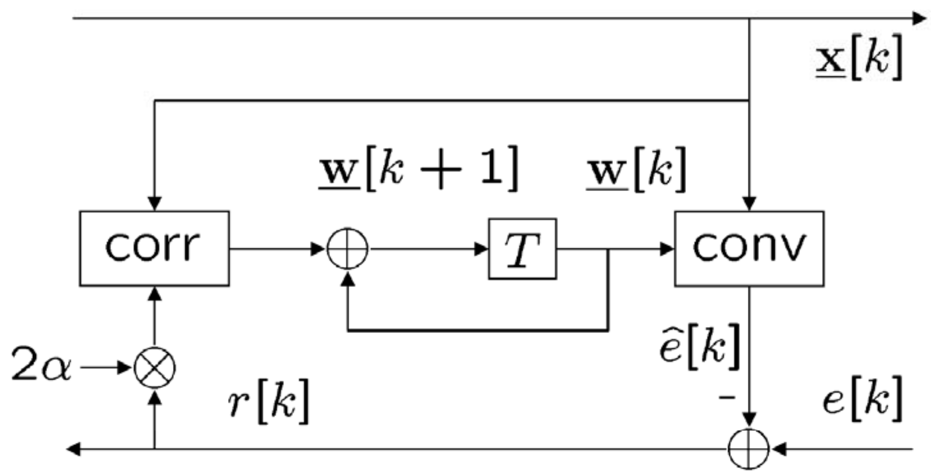
Frequency Domain Adaptive Filter

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:

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

FDAF

Apply filter operation in frequency domain:

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

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

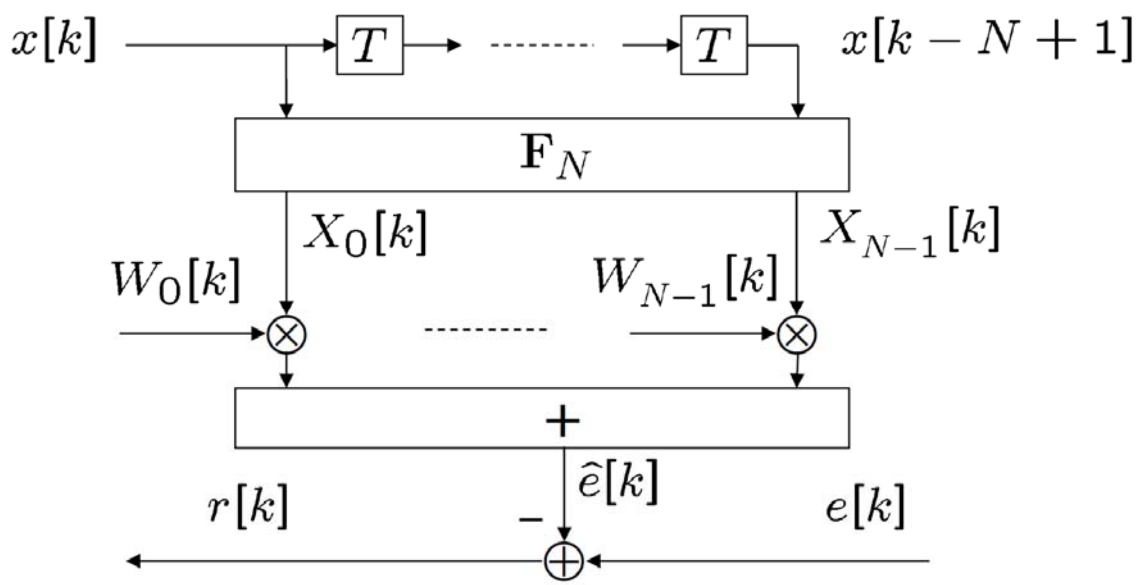
$$Note:\ \mathbf{F}^{-1} = \frac{1}{N}\mathbf{F}^*$$

Filter output:

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

ADSP

FDAF



Notes:

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

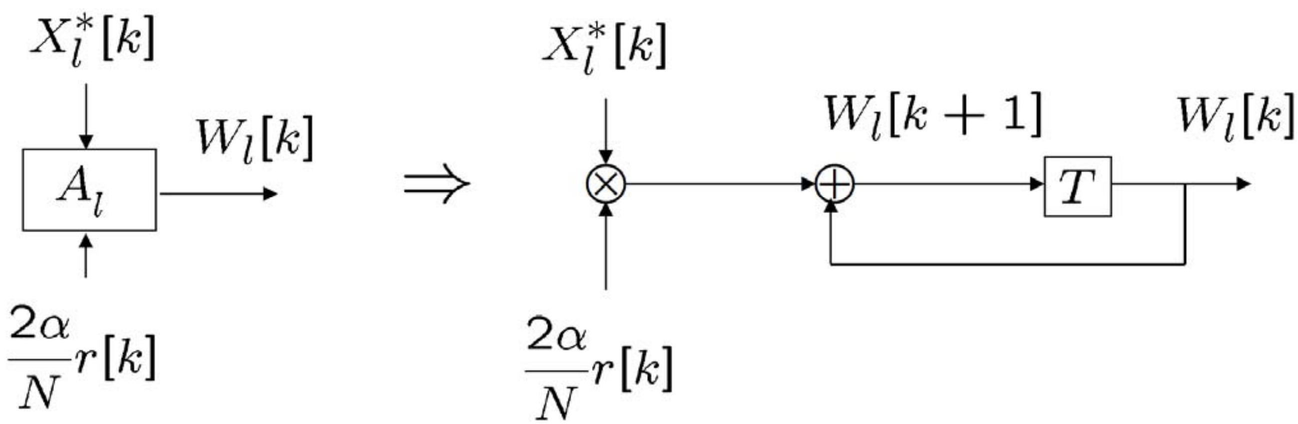
FDAF

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:

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

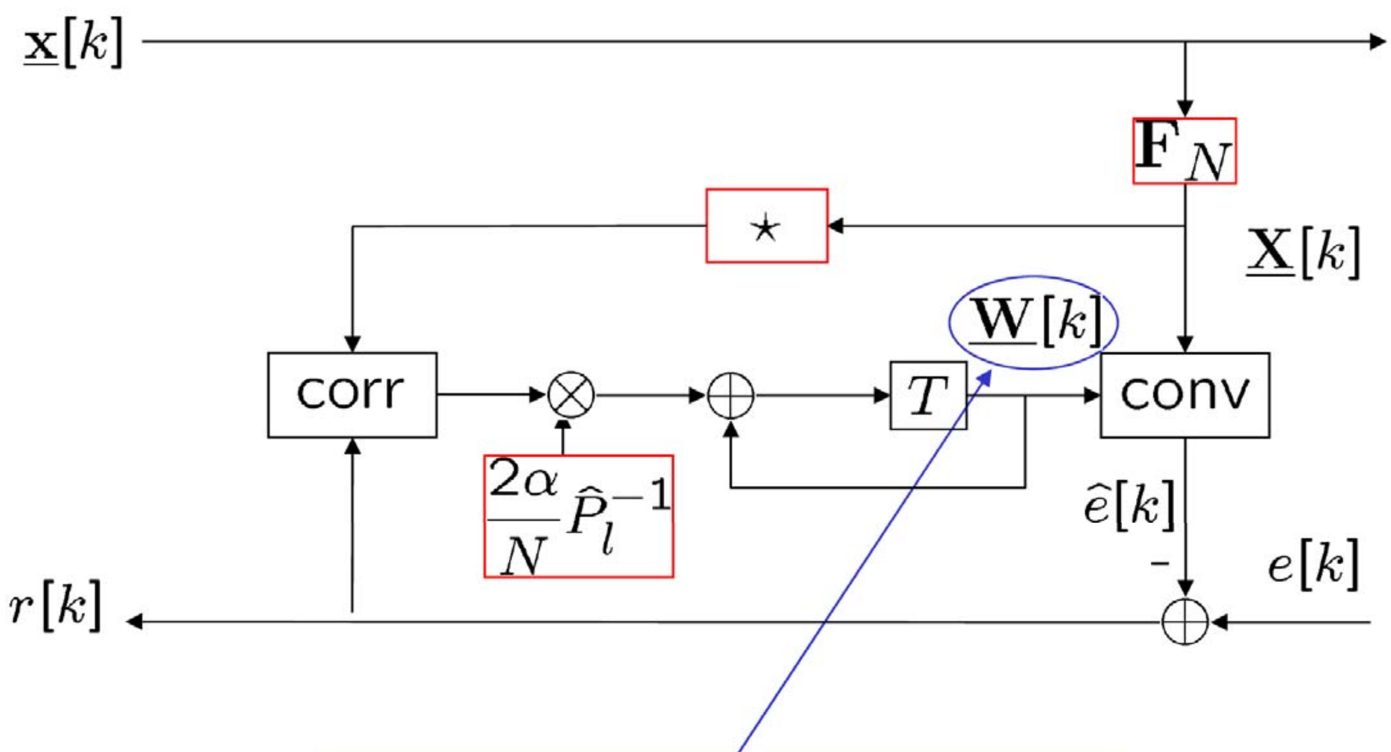
$$\mathbf{P} = diag\{\underline{\mathbf{P}}\} \text{ with } P_l = \frac{1}{N}E\{|X_l[k]|^2\}$$

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:



Note: $\underline{\mathbf{W}}[k] = \mathbf{F}^{-1} \cdot \underline{\mathbf{w}}[k]$

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:

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

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

All weights have (in average) similar convergence!

Equivalent to NLMS with white noise input!

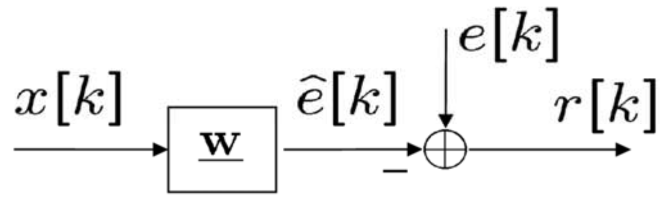
上海科技大学
ShanghaiTech University

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}	$E\{e^2\} - \underline{\mathbf{r}}_{ex}^t \mathbf{R}_x^{-1} \underline{\mathbf{r}}_{ex}$	$\underline{\mathbf{e}}^t \underline{\mathbf{e}} - \underline{\overline{\mathbf{r}}}_{ex}^t \overline{\mathbf{R}}_x^{-1} \underline{\overline{\mathbf{r}}}_{ex}$

ADSP

Summary

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

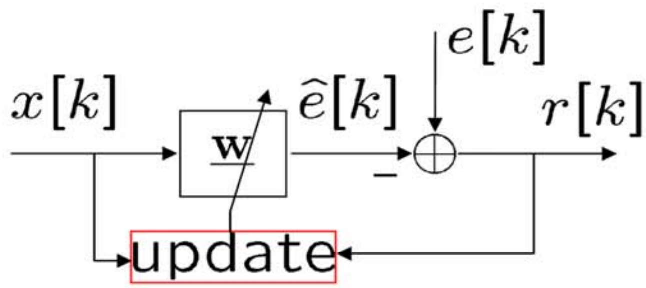
Solution for $N \geq M$: $\underline{\mathbf{w}}^c = \mathbf{C}\left(\mathbf{C}^t\mathbf{C}\right)^{-1}\underline{\mathbf{f}}$

Solution for $N > M$ with MMSE:

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

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:

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

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

RLS : $\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + \underline{\mathbf{g}}[k+1] r[k+1]$

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

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

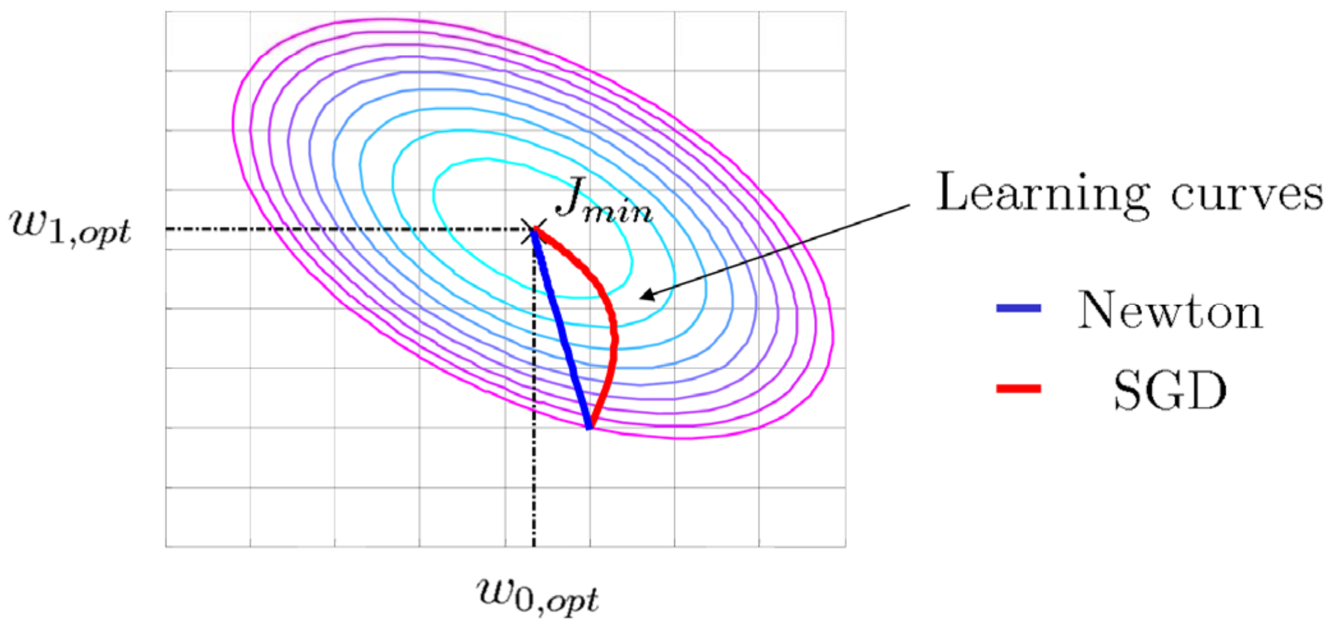
etc.

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}