Note Title 15-09-2014

Stochastic Gradient Algorithms

Study LMS (Least Mean Squares) algorithm and its variants.

Recall the optimal estimation problem

X is random variable
$\underline{Y}$ is random vector ($M \times 1$) — Size. — Zero Mean

$$\hat{X}_{LMMSE} = \underline{w}_{opt}^{*} \underline{Y} \quad \text{where} \quad \underline{w}_{opt} = R_Y^{-1} R_{YX} \rightarrow (*)$$

SD A: $$\underline{w}(n+1) = \underline{w}(n) + \mu \left(R_{YX} - R_Y \underline{w}(n) \right) \rightarrow (**)$$

To find optimal filter directly (*) or iteratively (**), we need the second order statistics R_Y, R_{YX}.

In many situations,

(a) R_Y, R_{YX} may not be known

or

(b) R_Y, R_{YX} may change over time (non stationary)

Problem Formulation

We are given realizations of X and $\underline{Y}$

$$\{x(0), x(1), \ldots\}, \{\underline{y}(0), \underline{y}(1), \ldots\}$$

$x(n) \rightarrow$ outcome of X at time instant n

$\underline{y}(n) \rightarrow$ outcome of $\underline{Y}$ at time n

Using these outcomes (data), we want to find / learn the optimal filter.

Approach:

→ Approximate $R_{\underline{y}}$ & $R_{\underline{y}x}$ with instantaneous values

→ Learn optimal filter iteratively

→ Avoid the need for apriori statistical knowledge

→ Provide a tracking mechanism

LMS Algorithm -

$$R_y = E\{\underline{y}\,\underline{y}^*\}$$

$$\approx \frac{1}{N}\sum_{n=1}^{N} \underline{y}(n)\,\underline{y}^*(n)$$

$$R_{yx} = E\{\underline{y}\,x^*\}$$

$$\approx \frac{1}{N}\sum_{n=1}^{N} \underline{y}(n)\,x^*(n)$$

At time instant 'n'

Instananeous approximation $\hat{R}_y(n) = \underline{y}(n)\,\underline{y}^*(n)$

$$\hat{R}_{yx}(n) = \underline{y}(n)\,x^*(n)$$

Apply these approximations in SDA.

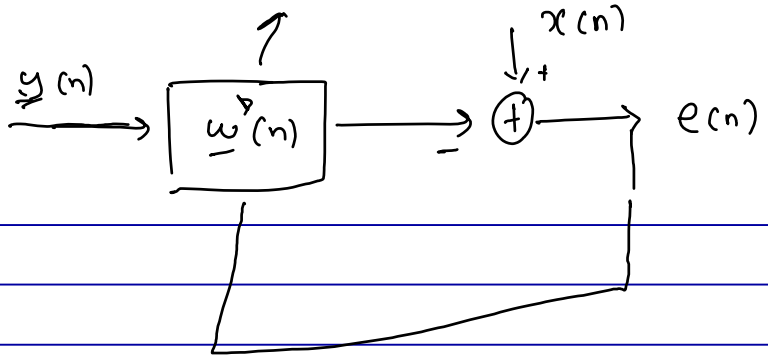
$$\underline{w}(n+1) = \underline{w}(n) + \mu\left(R_{yx}(n) - \hat{R}_y(n)\underline{w}(n)\right)$$

$$= \underline{w}(n) + \mu\left(\underline{y}(n)\,x^*(n) - \underline{y}(n)\,\underline{y}^*(n)\,\underline{w}(n)\right)$$

(Widrow & Hoff)

LMS $\underline{w}(n+1) = \underline{w}(n) + \mu\,\underline{y}(n)\left[x^*(n) - \underline{y}^*(n)\,\underline{w}(n)\right]$

$e^*(n)$



Remarks

- $\mu > 0$ (usually very small)
- Computationally simple algorithm
  (2M complex multiplications)
- * Widely used algorithm

x ——————— x

16/9 Least Perturbation Property of LMS

- LMS can be regarded as an exact solution to a local optimization problem
- Want to get an update $\underline{w}(n+1)$, from $\underline{w}(n)$, $\underline{y}(n)$, $x(n)$

apriori error $e_a(n) = x(n) - \underline{w}^H(n)\,\underline{y}(n)$

posterior error $e_p(n) = x(n) - \underline{w}^H(n+1)\,\underline{y}(n)$

↓

after filter update

Consider the optimization problem

$$\min \| \underline{w}(n+1) - \underline{w}(n) \|^2 \quad \text{subject to the}$$

Constraint $e_p(n) = \left(1 - \mu \| \underline{y}(n) \|^2\right) e_a(n)$

This constraint is relevant only if

$$\left| 1 - \mu \| \underline{y}(n) \|^2 \right| < 1 \quad \forall n$$

In this case $|e_p(n)| < |e_a(n)|$

(meaning $\underline{w}^*(n+1)\,\underline{y}(n)$ is a better estimate of $x(n)$ than $\underline{w}^*(n)\,\underline{y}(n)$)

Solve the above optimization problem

$$e_a(n) - e_p(n) = \underbrace{\left(\underline{w}(n+1) - \underline{w}(n)\right)^*}_{\delta \underline{w}} \underline{y}(n)$$

from the constraint $= \mu \| \underline{y}(n) \|^2 e_a(n)$

$\| \underline{u} \|^2 = \underline{u}^* \underline{u}$

(local) optimization problem is

at time n $\min \| \delta \underline{w} \|^2$ such that

$$(*) \longrightarrow \left(\delta \underline{w}\right)^* \underline{y}(n) = \mu \| \underline{y}(n) \|^2 e_a(n).$$

There are infinite solutions for (*)

Consider $\delta\underline{w}_0 = \mu e_a^*(n)\, \underline{y}(n)$

This is a solution for (*).

$$\delta\underline{w}_0^* \underline{y}(n) = \mu e_a(n)\, \underline{y}^*(n)\, \underline{y}(n)$$

$$= \mu e_a(n) \|\underline{y}(n)\|^2$$

Let $\delta\underline{w}_0 + \underline{z}$ be any other solution for (*)

So $(\delta\underline{w}_0 + \underline{z})^* \underline{y}(n) = \mu e_a(n) \|\underline{y}(n)\|^2$

$\delta\underline{w}_0^* \underline{y}(n) + \underline{z}^* \underline{y}(n) = \mu e_a(n) \|\underline{y}(n)\|^2$

$\Rightarrow \underline{z}^* \underline{y}(n) = 0$

Now, $\|\delta\underline{w}_0 + \underline{z}\|^2 = (\delta\underline{w}_0 + \underline{z})^* (\delta\underline{w}_0 + \underline{z})$

$= \delta\underline{w}_0^* \delta\underline{w}_0 + \underline{z}^* \underline{z} + \delta\underline{w}_0^* \underline{z} + \underline{z}^* \delta\underline{w}_0$

$= \|\delta\underline{w}_0\|^2 + \|\underline{z}\|^2 + \mu e_a(n) \underbrace{\underline{y}^*(n)\, \underline{z}}_{0} + \underbrace{\underline{z}^*\, \underline{y}(n)}_{0} \mu e_a(n)$

$= \|\delta\underline{w}_0\|^2 + \|\underline{z}\|^2$

$\geq \|\delta\underline{w}_0\|^2$

So $\underline{w}_0$ is the min norm solutn satisfying constraint (*)

$$\underline{w}_0 = \mu e_a^*(n)\, \underline{y}(n)$$

$$\underline{w}_{n+1} - \underline{w}(n) = \mu e_a^*(n)\, \underline{y}(n)$$

$$\underline{w}(n+1) = \underline{w}(n) + \mu \underline{y}(n) \left[x^*(n) - \underline{y}^H(n)\, \underline{w}(n) \right]$$

↓

LMS update.

Applicatins of LMS

1. Channel estimatin / System Identificatn
2. Channel Equalizatin

Channel Estimatin Problem.

- Channel is modelled as FIR filter (finite impulse response)

$$x(n) = \sum_{l=0}^{L} h_l^* y(n-l) + v(n)$$

$v(n)$ ↓ measurement noise

$\{h_0, h_1, \ldots, h_L\}$ are unknown filter coefficients.

Define vectors $\underline{y}(n) = \begin{bmatrix} y(n) \\ y(n-1) \\ \vdots \\ y(n-L) \end{bmatrix}$

$L \rightarrow$ length of FIR filter.

$$\underline{h} = \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_L \end{bmatrix}$$

$$x(n) = \underline{h}^* \underline{y}(n) + v(n)$$

17/4

Send a random sequence $\{r(n)\}$ thro' the system

Corresponding output

$$X(n) = \underline{h}^* \underline{r}(n) + V(n)$$

Assume that noise $V(n)$ is uncorrelated with input $r(n)$

Find $\underline{w}$ such that

$$E\left| X(n) - \underline{w}^* \underline{r}(n) \right|^2 \text{ is minimum}$$

$$R_r(n) = E\{ \underline{r}(n) \underline{r}^*(n) \} = R_r$$

$$R_{rx}(n) = E\{ r(n) x^*(n) \} = R_{rx}$$

} for (wide-sense) stationary sequences

Assuming stationarity.

$$\boxed{R_r \underline{w}_{opt} = R_{rx}}$$

$$R_{yx} = E\{ \underline{y}(n)\, x^*(n) \}$$

$$= E\{ \underline{y}(n) \cdot (\underline{h}^H \underline{y}(n) + v(n))^* \}$$

$$= E\{ \underline{y}(n)\, \underline{y}^H(n)\, \underline{h} + \underbrace{\underline{y}(n)\, v^*(n)}_{\underline{0}} \}$$

$\underline{0}$ (noise is uncorrelated with input)

$$= R_y \underline{h}$$

$$\boxed{\underline{h} = R_y^{-1} R_{yx}} = \underline{w}_{opt}$$

$\Rightarrow$ direct computation of optimal filter

We know that SDA can be used to compute $\underline{w}_{opt}$ iteratively (without computing R_y^{-1} explicitly)

start with $\underline{w}(0)$ (initial vector)

SDA:

$$\underline{w}(n+1) = \underline{w}(n) + \mu (R_{yx} - R_y \underline{w}(n))$$

Suppose R_y & R_{yx} are unknown, then we can use LMS algorithm to "learn" the optimal filter (which is same as impulse response of system)

LMS: Instantaneous Approximations

$$R_y = \underline{y}(n)\, \underline{y}^H(n)$$

$$R_{yx} = \underline{y}(n)\, \underline{x}^*(n)$$

$$\underline{w}(n+1) = \underline{w}(n) + \mu\, \underline{y}(n) \left[x^*(n) - \underline{y}^*(n)\, \underline{w}(n) \right]$$

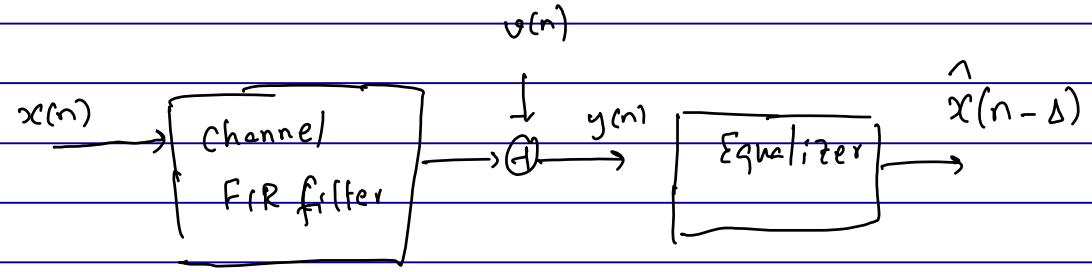
$\Downarrow$

After convergence $\underline{w}(n) \simeq \underline{h}$

$\{y(n)\}$ is known training signal sent thru system.

$\{x(n)\}$ is corresponding output

Channel Equalization.



$x(n)$ is input to channel

$y(n)$ is channel output + noise

From $y(n)$, we want to recover $x(n)$

$\Delta \rightarrow$ is the delay introduced due to filter & equalizer.

$X(n) \rightarrow$ random sequence sent thru channel

$V(n) \rightarrow$ noise uncorrelated with $X(n)$

$$Y(n) = \sum_{i=0}^{L} h_i^* X(n-i) + V(n)$$

Let us collect M observations of $\{Y(n)\}$

$$\underline{Y}(n) = \begin{bmatrix} Y(n) \\ Y(n-1) \\ \vdots \\ Y(n-M+1) \end{bmatrix}$$

$M \times 1$

$$\begin{bmatrix} Y(n) \\ Y(n-1) \\ \vdots \\ Y(n-M+1) \end{bmatrix} = \begin{bmatrix} h_0^* & h_1^* & \cdots & h_L^* & 0 & 0 & \cdots & 0 \\ 0 & h_0^* & h_1^* & \cdots & h_L^* & 0 & \cdots & 0 \\ & & \ddots & & & \ddots & & \\ 0 & \cdots & 0 & h_0^* & h_1^* & \cdots & h_L^* & \end{bmatrix} \begin{bmatrix} X(n) \\ X(n-1) \\ \vdots \\ X(n-M-L+1) \end{bmatrix} + \begin{bmatrix} V(n) \\ \vdots \\ V(n-M+1) \end{bmatrix}$$

$$\underline{Y}(n) = H \underline{X}(n) + \underline{V}(n)$$

From $\underline{Y}(n)$, Let us recover $\underline{X}(n)$

$\hat{\underline{X}}(n)$ is recovered from LMMSE estimate of $\underline{X}(n)$ from $\underline{Y}(n)$

$$\hat{\underline{X}}_{\text{LMMSE}}(n) = R_{XY}(n) \cdot R_Y^{-1}(n) \, \underline{Y}(n)$$

Assume $\{X(n), V(n)\}$ are stationary & uncorrelated.

$$R_Y = E\{ \underline{Y}(n) \underline{Y}(n)^* \}$$

$$= H R_x H^* + R_v$$

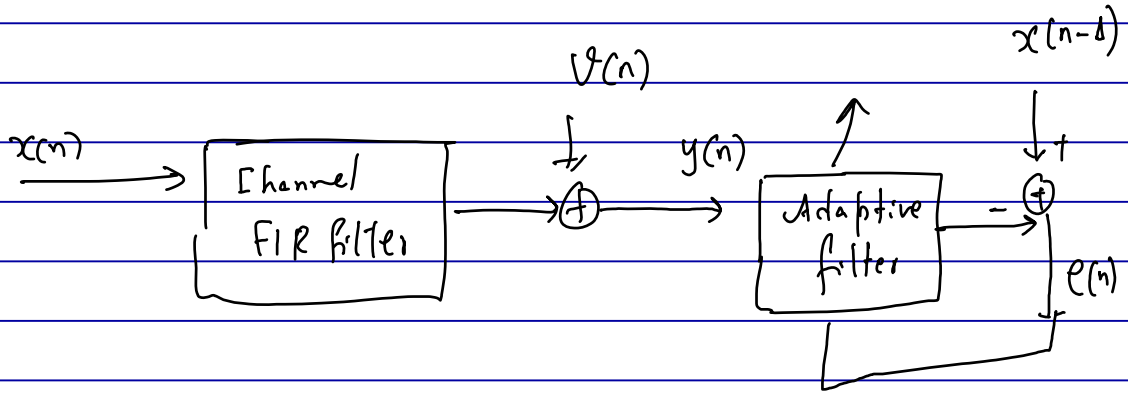
$$R_{YX} = H R_x = R_{XY}^*$$

$$\hat{\underline{X}}_{LMMSE}(n) = (H R_x)^* [H R_x H^* + R_v]^{-1} \underline{Y}(n)$$

linear MMSE equalizer.

Adaptive implementation of Equalizer

Send a training signal $\{x(n)\}$ thru channel.



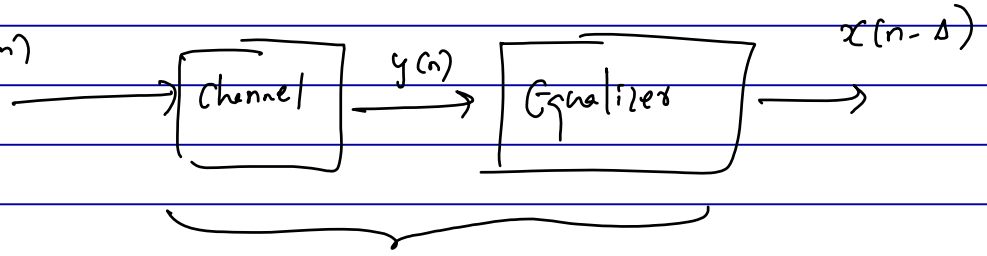
Start with weight vector $\underline{w}(0)$ for adaptive filter.

LMS update:

$$\underline{w}(n+1) = \underline{w}(n) + \mu \underline{y}(n) [x^*(n-\Delta) - \underline{y}^*(n) \underline{w}(n)]$$

After convergence, $\underline{w}(n)$ will be close to $\underline{w}_{opt}$.

Then $\underline{w}^T(n)\,\underline{y}(n)$ will be close to $x(n-\Delta)$



$\approx z^{-\Delta}$ (z transform)

Once the training phase is over,

we can send (unknown) inputs thru channel

& use the equalizer to get

back the inputs.

Normalized LMS algorithm

We had steepest descent algorithm

and used instantaneous approximation to get LMS

SDA: $\underline{w}(n+1) = \underline{w}(n) + \mu\left(R_{yx} - R_y\,\underline{w}(n)\right)$

Newton's method:

$$\underline{w}(n+1) = \underline{w}(n) + \mu\, R_y^{-1}\left(R_{yx} - R_y\,\underline{w}(n)\right)$$

Newton's method with regularization:

- If R_y is not invertible or R_y is close to a singular matrix

Then Replace R_y^{-1} with $\left(R_y + \varepsilon I\right)^{-1}$

$\varepsilon > 0$

$$\underline{w}(n+1) = \underline{w}(n) + \mu \left(R_y + \varepsilon I\right)^{-1} \left(R_{yx} - R_y \underline{w}(n)\right)$$

We use instantaneous approximation on this

Newton's method with regularization.

$$R_y = \underline{y}(n)\,\underline{y}^*(n) \qquad R_{yx} = \underline{y}(n)\,x^*(n)$$

$$\underline{w}(n+1) = \underline{w}(n) + \mu \left[\varepsilon I + \underline{y}(n)\underline{y}^*(n)\right]^{-1} \left(\underline{y}(n)x^*(n) - \underline{y}(n)\underline{y}^*(n)\underline{w}(n)\right)$$

Using Matrix inversion lemma.

$$\left[\varepsilon I + \underline{y}(n)\underline{y}^*(n)\right]^{-1} = \varepsilon^{-1} I - \frac{\varepsilon^{-2}\,\underline{y}(n)\underline{y}^*(n)}{1 + \varepsilon^{-1}\|\underline{y}(n)\|^2}$$

$$\underline{w}(n+1) = \underline{w}(n) + \mu \left(\varepsilon^{-1} I - \frac{\varepsilon^{-2}\,\underline{y}(n)\underline{y}^*(n)}{1 + \varepsilon^{-1}\|\underline{y}(n)\|^2}\right) \underline{y}(n)\left[x^*(n) - \underline{y}^*(n)\underline{w}(n)\right]$$

$$= \underline{w}(n) + \mu \left(\varepsilon^{-1}\underline{y}(n) - \frac{\varepsilon^{-2}\,\underline{y}(n)\|\underline{y}(n)\|^2}{1 + \varepsilon^{-1}\|\underline{y}(n)\|^2}\right)\left(x^*(n) - \underline{y}^*(n)\underline{w}(n)\right)$$

$$\underline{w}(n+1) = \underline{w}(n) + \frac{\mu}{\varepsilon + \|\underline{y}(n)\|^2} \underline{y}(n) \left(x^*(n) - \underline{y}^H(n)\, \underline{w}(n) \right)$$

$\Downarrow$

Normalized LMS. (NLMS)

This can be considered as a Variable step size LMS with $\mu(n) = \frac{\mu}{\varepsilon + \|\underline{y}(n)\|^2}$

× ——— ×

Least Perturbation property of ε-NLMS

Similar to the LMS.

We can verify that ε-NLMS update is the solution to the following optimization problem.

$\min \|\underline{w}(n+1) - \underline{w}(n)\|^2$ subject to the constraint that

$$e_p(n) = \left(1 - \frac{\mu \|\underline{y}(n)\|^2}{\varepsilon + \|\underline{y}(n)\|^2} \right) e_a(n)$$

$$e_a(n) = x(n) - \underline{w}^H(n)\, \underline{y}(n)$$

$$e_p(n) = x(n) - \underline{w}^H(n+1)\, \underline{y}(n)$$

We need μ such that

$$\left| 1 - \frac{\mu \|\underline{y}(n)\|^2}{\varepsilon + \|\underline{y}(n)\|^2} \right| < 1$$

Equivalently

$$0 < \mu < \frac{2\left(\varepsilon + \|\underline{y}(n)\|^2\right)}{\|\underline{y}(n)\|^2}$$

It is easy to see that choosing $0 < \mu < 2$ the above condition is satisfied irrespective of $\|\underline{y}(n)\|^2$

× ——— ×

Various cost functions

- So far we have studied MSE cost.

$$\min_{\underline{w}} J(\underline{w}) = E\{|X - \underline{w}^T \underline{Y}|^2\}$$

$-\nabla J(\underline{w})$

* Direct computation $\underline{w}_{opt} = R_Y^{-1} R_{YX}$

Compute $\nabla_{\underline{w}} J(\underline{w})$ →

* SDA : $\underline{w}(n+1) = \underline{w}(n) + \mu\left(R_{YX} - R_Y \underline{w}(n)\right)$

* LMS : $\underline{w}(n+1) = \underline{w}(n) + \mu\, \underline{y}(n)\left[x(n) - \underline{y}^T(n)\underline{w}(n)\right]$

instantaneous approximation for R_Y, R_{YX}

Variants of LMS

* **Signed - LMS.**

$J(\underline{w})$

$$\min_{\underline{w}} \; E\{ |X - \underline{w}^H \underline{Y}| \}$$

$\Downarrow$

absolute error is cost.

$$\underline{w}(n+1) = \underline{w}(n) + \mu \underline{y}(n) \cdot \text{Sign}\left(x^*(n) - \underline{y}^H(n) \underline{w}(n) \right)$$

where $\text{Sign}(a + jb) = \text{Sign}(a) + j\,\text{Sign}(b)$

* **Leaky LMS.**

$$\min_{\underline{w}} \; \alpha \|\underline{w}\|^2 + E\{|X - \underline{w}^H \underline{Y}|^2\}$$

for some $\alpha > 0$.

$$\underline{w}(n+1) = (1 - \mu\alpha)\, \underline{w}(n) + \mu \underline{y}(n) \left[x^*(n) - \underline{y}^H(n) \underline{w}(n) \right]$$

22/9

* **Least Mean Fourth (LMF) Algorithm.**

$$\text{Cost} \quad J(\underline{w}) = E\{ |X - \underline{w}^H \underline{Y}|^4 \}$$

corresponding update equation is

$$\underline{w}(n+1) = \underline{w}(n) + \mu \underline{y}(n) \left[e^*(n) |e(n)|^2 \right]$$

where $e(n) = x(n) - \underline{w}^H(n)\,\underline{y}(n)$

* Least mean - mixed norm [LMMN] Algorithm.

let $E = X - \underline{w}^H \underline{Y}$

$$J(\underline{w}) = E\left\{ \delta |E|^2 + \frac{(1-\delta)|E|^4}{2} \right\}$$

$0 \leq \delta \leq 1$

update equation

$$\underline{w}(n+1) = \underline{w}(n) + \mu\, \underline{y}(n)\, e^*(n)\left[\delta + (1-\delta)|e(n)|^2\right]$$

where $e(n) = x(n) - \underline{w}^H(n)\,\underline{y}(n)$

Blind Algorithms.

- So far, adaptive filters we saw need training data $\{x(0), x(1), \ldots\}$ & $\{y(0), y(1), \ldots,\}$
- In some applications, you may not have access to $\{x(n)\}$
- In this case, you can still use some properties you know (apriorily) about $x(n)$ to train filter.

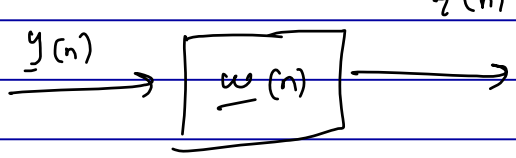
× ——————— ×

Constant Modulus Algorithms. (CMA)

* In digital communication, the symbols $\{x(n)\}$ sent thru channel from transmitter are from a specific constellation. Note that if constellation is PSK, then all symbols $\{x(n)\}$ have same modulus.

CMA algorithms exploit this constant modulus property of $\{x(n)\}$ to blindly train the adaptive filter

* CMA 1



$$\min_{\underline{w}} \; J(\underline{w}) = E\left\{ \left| \gamma - |z| \right|^2 \right\}$$

$$z = \underline{w}^H \underline{y}$$

where $\gamma > 0$

& real positive constant.

Update equation:

$$\underline{w}(n+1) = \underline{w}(n) + \mu\, \underline{y}(n) \left[\gamma \frac{z(n)}{|z(n)|} - z(n) \right]$$

Note that $\underline{w}(n+1) = \underline{w}(n)$

if $|z(n)| = \gamma$

ε- Normalized CMA

$$\underline{w}(n+1) = \underline{w}(n) + \frac{\mu}{\varepsilon + \|\underline{y}(n)\|^2}\, \underline{y}(n) \left[\gamma \frac{z(n)}{|z(n)|} - z(n) \right]$$

- CMA 2

$$J(\underline{w}) = E\left\{ \left| \gamma - |z|^2 \right|^2 \right\}$$

$$z = \underline{w}^H \underline{y}$$

Update equation is

$$\underline{w}(n+1) = \underline{w}(n) + \mu\, \underline{y}(n)\, z(n) \left[\gamma - |z(n)|^2 \right]$$

$$z(n) = \underline{w}^H(n)\, \underline{y}(n).$$

Other Variants of CMA.

* Reduced Constellation Algorithm (RCA)

$$J(\underline{w}) = E\left\{ \left| \underline{w}^H \underline{y} - \gamma\, \text{sign}(\underline{w}^H \underline{y}) \right|^2 \right\}$$

Update Equation

$$\underline{w}(n+1) = \underline{w}(n) + \mu\, \underline{y}(n) \left[\gamma\, \text{Sign}(z(n)) - z(n) \right]$$

$$z(n) = \underline{w}^H(n)\, \underline{y}(n)$$

* Multi-Modulus Algorithm (MMA)

$$J(\underline{w}) = E\left\{ \left[\left(\text{Re}\{\underline{w}^H \underline{y}\} \right)^2 - \gamma \right]^2 + \left[\left(\text{Im}(\underline{w}^H \underline{y}) \right)^2 - \gamma \right]^2 \right\}$$

Update equation:

$$z(n) = \underline{w}(n)^{*} \underline{y}(n)$$

$$a(n) = \mathrm{Re}\,(z(n))$$

$$b(n) = \mathrm{Im}\,(z(n))$$

$$e(n) = a(n)\left[\gamma - a^2(n)\right] + j\, b(n)\left[\gamma - b^2(n)\right]$$

$$\underline{w}(n+1) = \underline{w}(n) + \mu\, \underline{y}(n)\, e^{*}(n)$$

Portions for quiz 2

Affine Projection Algorithms.

- We go back to the MSE cost

$$J(\underline{w}) = E\left\{ |X - \underline{w}^{H} \underline{Y}|^2 \right\}$$

LMS Approximation

$$R_Y = \hat{R}_Y(n) = \underline{y}(n)\, \underline{y}^{H}(n)$$

Recall Newton method with regularization

$$(*) \quad \underline{w}(n+1) = \underline{w}(n) + \mu \left(\varepsilon \underline{\underline{I}} + R_Y \right)^{-1} \left[R_{YX} - R_Y \underline{w}(n) \right]$$

At n^{th} instant, we use past K realizations $\{ \underline{y}(n), \underline{y}(n-1), \ldots, \underline{y}(n-k+1) \}$

$\{ x(n), x(n-1), \ldots, x(n-k+1) \}$

to estimate R_y and R_{yx}

$$\hat{R}_y = \frac{1}{k} \sum_{i=n-k+1}^{n} \underline{y}(i)\, \underline{y}^*(i)$$

$$\hat{R}_{yx} = \frac{1}{k} \sum_{i=n-k+1}^{n} \underline{y}(i)\, x^*(i)$$

Using these estimates in Newtons method, we get affine projection algorithms.

23/9

Let us define matrix

$$\underset{M \times k}{Y_n} = [\underline{y}(n)\ \underline{y}(n-1) \ \ldots\ \underline{y}(n-k+1)]$$

$$\underset{k \times 1}{\underline{x}_n} = [x(n)\ x(n-1) \ \ldots\ x(n-k+1)]$$

$$Y_n Y_n^* = [\underline{y}(n) \ \ldots\ \underline{y}(n-k+1)] \begin{bmatrix} \underline{y}^*(n) \\ \underline{y}^*(n-1) \\ \vdots \\ \underline{y}^*(n-k+1) \end{bmatrix}$$

$$= \underline{y}(n)\underline{y}^*(n) + \underline{y}(n-1)\underline{y}^*(n-1) + \cdots + \underline{y}(n-k+1)\underline{y}^*(n-k+1)$$

$$\hat{R}_Y = \frac{\underline{Y}_n \underline{Y}_n^*}{k}$$

$$\hat{R}_{YX} = \frac{1}{k}\underline{Y}_n \underline{X}_n^*$$

} using these approximations in Newton's method

Take ε' in (*) as ε/k

Newtons method becomes

$$\underline{w}(n+1) = \underline{w}(n) + \mu\left(\frac{\varepsilon}{k}\underline{I} + \frac{1}{k}\underline{Y}_n\underline{Y}_n^*\right)^{-1}$$

$$\left(\frac{1}{k}\underline{Y}_n\underline{X}_n^* - \frac{1}{k}\underline{Y}_n\underline{Y}_n^*\underline{w}(n)\right)$$

ie)

$$\underline{w}(n+1) = \underline{w}(n) + \mu\left(\varepsilon\underline{I} + \underline{Y}_n\underline{Y}_n^*\right)^{-1}\left(\underline{Y}_n\underline{X}_n^* - \underline{Y}_n\underline{Y}_n^*\underline{w}(n)\right)$$

$\Downarrow$

ε – Affine projection algorithm (APA)

$\varepsilon\underline{I} + \underline{Y}_n\underline{Y}_n^* \rightarrow M \times M$ matrix

Using Matrix inversion lemma, we can write this inverse in terms of $k \times k$ inverse.

(Typically $k \leq M$)

$$(A+BCD)^{-1} = A^{-1} - A^{-1}B\left(C^{-1} + DA^{-1}B\right)^{-1} DA^{-1}$$

$$\left(\varepsilon I + Y_n Y_n^*\right)^{-1} = ?$$

$$A = \varepsilon I \qquad B = Y_n \qquad C = I \qquad D = Y_n^*$$

$$\left(\varepsilon I + Y_n Y_n^*\right)^{-1} = \frac{1}{\varepsilon} I - \frac{1}{\varepsilon} I\, Y_n \left(I + \frac{Y_n^* I Y_n}{\varepsilon}\right)^{-1} \cdot \left(\frac{Y_n^* I}{\varepsilon}\right)$$

$$= \frac{1}{\varepsilon}\left(I - Y_n \underbrace{\left(\varepsilon I + Y_n^* Y_n\right)^{-1}}_{k \times k \text{ matrix}} Y_n^*\right)$$

$$\left(\varepsilon I + Y_n Y_n^*\right)^{-1} Y_n = \frac{1}{\varepsilon}\left(I - Y_n \left(\varepsilon I + Y_n^* Y_n\right)^{-1} Y_n^*\right) Y_n$$

$$= \frac{Y_n}{\varepsilon}(I) - \frac{Y_n}{\varepsilon}\left(\varepsilon I + Y_n^* Y_n\right)^{-1} Y_n^* Y_n$$

$$= \frac{Y_n}{\varepsilon}\underbrace{\left(\varepsilon I + Y_n^* Y_n\right)^{-1}\left(\varepsilon I + Y_n^* Y_n\right)}$$

$$- \frac{Y_n}{\varepsilon}\left(\varepsilon I + Y_n^* Y_n\right)^{-1} Y_n^* Y_n$$

$$= \frac{Y_n}{\varepsilon}\left(\varepsilon I + Y_n^* Y_n\right)^{-1}\left(\varepsilon I + Y_n^* Y_n - Y_n^* Y_n\right)$$

$$= \underline{Y}_n \left(\varepsilon \underline{I} + \underline{Y}_n^* \underline{Y}_n \right)^{-1}$$

Update eqn of ε-APA becomes

$$\underline{w}(n+1) = \underline{w}(n) + \mu \underline{Y}_n \left(\varepsilon \underline{I} + \underline{Y}_n^* \underline{Y}_n \right)^{-1} \left(\underline{x}_n^* - \underline{Y}_n^* \underline{w}(n) \right)$$

↓

involves $k \times k$ matrix inversion

which requires ($O(k^3)$ operations

addition & multiplications)

For $k=1$, we get ε-NLMS

$$\underline{w}(n+1) = \underline{w}(n) + \mu \frac{\underline{y}(n)}{\varepsilon + \| \underline{y}(n) \|^2} \left(x^*(n) - \underline{y}^*(n) \underline{w}(n) \right)$$

* ε-APA uses vector valued estimation error

$$\left(\underline{x}_n^* - \underline{Y}_n^* \underline{w}(n) \right)$$

* We are reusing data in update eqn.

* k is referred as order of ε-APA.

Least Perturbation Property.

Similar to (LMS, ε-NLMS) the ε-APA can also be seen as an exact solution to a local optimization problem

Let us define apriori & posteriori estimation error vectors.

$$\underline{e}_a(n) = \underline{x}'(n) - \underline{Y}_n' \underline{w}(n)$$

$$\underline{e}_p(n) = \underline{x}'(n) - \underline{Y}_n' \underline{w}(n+1)$$

ε-APA update equation is the solution for the following optimization problem.

$$\min_{\underline{w}(n+1)} \| \underline{w}(n+1) - \underline{w}(n) \|^2 \quad \text{subject to}$$

The constraint $\underline{e}_p(n) = \left(\underline{I} - \mu \underline{Y}_n \underline{Y}_n' \left(\varepsilon \underline{I} + \underline{Y}_n' \underline{Y}_n \right)^{-1} \right) \underline{e}_a(n)$

Variation of APA.

Partial Rank Algorithm.

* Weight vector is updated once out of k iterations

$$\underline{w}(n+1) = \underline{w}(n) + \mu \underline{\gamma}_n \left(\varepsilon \underline{I} + \underline{\gamma}_n^* \underline{\gamma}_n\right)^{-1} \left(\underline{x}_n^* - \underline{\gamma}_n^* \underline{w}(n)\right)$$

$$\underline{w}(n+1) = w(n+2) = w(n+3) = \cdots = w(n+k)$$

$\underline{w}(n+1+k)$ is computed as per update eqn.

× ——————— ×

24/9

Recursive Least Squares (RLS)

- RLS uses more sophisticated approximation of Covariance matrix R_y using all the past data
- RLS is exact solution for a least squares estimation problem. Will see more details later Right now we see RLS as stochastic Gradient method.

Newton's Method (with regularization)

$$\underline{w}(n+1) = \underline{w}(n) + \mu_n \underbrace{\left(\varepsilon_n \underline{I} + R_y\right)^{-1}}_{\text{Gradient term}} \left(R_{yx} - R_y \underline{w}(n)\right)$$

For Gradient term,

$$R_{yx} - R_y \underline{w}(n) \approx \underline{y}(n)\, x^*(n) - \underline{y}(n)\, \underline{y}^*(n)\, \underline{w}(n)$$

For computing $\left(\mathcal{E}_n I + R_y\right)^{-1}$ we use the following approximation for R_y.

$$R_y \approx \frac{1}{n+1} \sum_{i=0}^{n} \lambda^{n-i}\, \underline{y}(i)\, \underline{y}^*(i) = \hat{R}_y$$

We choose $0 \ll \lambda < 1$

This helps in providing higher weightage to recent data in averaging.

$\lambda = 1 \Rightarrow$ all data gets equal weightage

$\lambda \rightarrow$ forgetting factor

Choose $\mu_n = \frac{1}{n+1}$ $\qquad \mathcal{E}_n = \frac{\lambda^{n+1} \cdot \mathcal{E}}{n+1}$.

with these choices, newton method approximation becomes

$$\underline{w}(n+1) = \underline{w}(n) + \frac{1}{n+1} \left(\frac{\lambda^{n+1}}{n+1} \mathcal{E} I + \frac{1}{n+1} \sum_{i=0}^{n} \lambda^{n-i}\, \underline{y}(i)\, \underline{y}^*(i) \right)^{-1} \left(\underline{y}(n)\, x^*(n) - \underline{y}(n)\, \underline{y}^*(n)\, \underline{w}(n) \right)$$

$$= \underline{w}(n) + \left(\lambda^{n+1}\varepsilon I + \sum_{i=0}^{n} \lambda^{n-i}\, \underline{y}(i)\,\underline{y}^{*}(i)\right)^{-1}$$

$$\left(\underline{y}(n)\,x^{*}(n) - \underline{y}(n)\,\underline{y}^{*}(n)\,\underline{w}(n)\right)$$

$$\Phi_n = \lambda^{n+1}\varepsilon I + \sum_{i=0}^{n} \lambda^{n-i}\, \underline{y}(i)\,\underline{y}^{*}(i)$$

We want Φ_n^{-1}. Let $P_n = \Phi_n^{-1}$

We will compute P_n using matrix inversion lemma.

First write Φ_n using a recursion

$$\Phi_n = \lambda\left(\lambda^{n}\varepsilon I + \sum_{i=0}^{n-1} \lambda^{n-1-i}\, \underline{y}(i)\,\underline{y}^{*}(i)\right)$$

$$+ \underline{y}(n)\,\underline{y}^{*}(n)$$

$$= \lambda\,\Phi_{n-1} + \underline{y}(n)\,\underline{y}^{*}(n)$$

$$\Phi_n^{-1} = P_n = \left(\lambda\,\Phi_{n-1} + \underline{y}(n)\,\underline{y}^{*}(n)\right)^{-1}$$

using Matrix inversion lemma

$$= \lambda^{-1}\Phi_{n-1}^{-1} - \lambda^{-2}\Phi_{n-1}^{-1}\,\underline{y}(n)\cdot$$

$$\left(1 + \lambda^{-1}\,\underline{y}^{*}(n)\,\Phi_{n-1}^{-1}\,\underline{y}(n)\right)^{-1}$$

$$\underline{y}^{*}(n)\,\Phi_{n-1}^{-1}$$

Note

$\phi_{n-1}^{-1} = P_{n-1}$

$$P_n = \lambda^{-1}\left[P_{n-1} - \frac{\lambda^{-1} P_{n-1}\, \underline{y}(n)\, \underline{y}^H(n)\, P_{n-1}}{1 + \lambda^{-1}\, \underline{y}^H(n)\, P_{n-1}\, \underline{y}(n)} \right]$$

Initialization $P_{-1} = \varepsilon^{-1} I$

RLS algorithm:

$$\underline{w}(n+1) = \underline{w}(n) + P_n\, \underline{y}(n) \left[x^*(n) - \underline{y}^H(n)\, \underline{w}(n) \right]$$

$$P_n = \lambda^{-1}\left[P_{n-1} - \frac{\lambda^{-1} P_{n-1}\, \underline{y}(n)\, \underline{y}^H(n)\, P_{n-1}}{1 + \lambda^{-1}\, \underline{y}^H(n)\, P_{n-1}\, \underline{y}(n)} \right]$$

$P_{-1} = \varepsilon^{-1} I$ and $0 < \lambda < 1$

Using Matrix inversion lemma, we have converted a matrix inversion $\left(\phi_n^{-1}\right)$ into a scalar inversion $\left[1 + \lambda^{-1}\, \underline{y}^H(n)\, P_{n-1}\, \underline{y}(n)\right]^{-1}$