Adaptive Signal Processing
by
Andreas Spanias,
Professor and Director SensIP Consortium
Department of Electrical Engineering
Arizona State University, Tempe, AZ 85287-5706
July 2007
Presented to
European Thematic Network, ERASMUS-TN,
DOCTORAL EDUCATION IN COMPUTING (DEC)
and the the University of Cyprus
spanias@asu.edu
Phone: 480 965 1837
http://www.eas.asu.edu/~spanias

Background in DSP

IIR Digital Filter Structure
\[ y(n) = \sum_{i=0}^{L} b_i x(n-i) - \sum_{i=0}^{M} a_i y(n-i) \]

FIR Digital Filter Structure
\[ y(n) = \sum_{i=0}^{L} b_i x(n-i) \]
Two i/p-o/p Equations for Digital Filters

One can compute the output using the convolution sum

\[ y(n) = \sum_{m=-\infty}^{\infty} h(m) x(n-m) = \sum_{m=-\infty}^{\infty} x(m) h(n-m) \]

or by using the difference equation

\[ y(n) = \sum_{i=0}^{L} b_i x(n-i) - \sum_{i=1}^{M} a_i y(n-i) \]

Remark: The impulse response \( h(n) \) can be determined by solving the difference equation.

From the Laplace Transform to the Z-Transform

The s-domain transfer function

\[ H_a(s) = \frac{1}{1+sRC} \]

becomes the z-domain transfer function

\[ H_d(z) = \frac{b_o}{1 + a_1 z^{-1}} \]

The Z-Transform - Definition

Given the signal: \( x(n) \)

its z-transform is

\[ X(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n} \]

For causal signals, i.e., \( x(n) = 0 \) for \( n < 0 \)

\[ X(z) = \sum_{n=0}^{\infty} x(n) z^{-n} \]

The Transfer Function

\[ y(n) = \sum_{i=0}^{L} b_i x(n-i) - \sum_{i=1}^{M} a_i y(n-i) \]

\[ X(z) \left( \sum_{i=1}^{L} b_i z^{-i} \right) = Y(z) \left( 1 + \sum_{i=1}^{M} a_i z^{-i} \right) \]

The transfer function \( H(z) \) is defined as:

\[ H(z) = \frac{Y(z)}{X(z)} = \frac{b_o}{1 + a_1 z^{-1} + \ldots + a_M z^{-M}} \]

Note that feedback terms are in the denominator.
The Frequency Response Function

The transfer function is

$$H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_L z^{-L}}{1 + a_1 z^{-1} + \cdots + a_M z^{-M}}$$

by evaluating on the unit circle, i.e. for $z = e^{j\Omega}$

$$H(e^{j\Omega}) = \frac{b_0 + b_1 e^{-j\Omega} + \cdots + b_L e^{-jL\Omega}}{1 + a_1 e^{-j\Omega} + \cdots + a_M e^{-jM\Omega}}$$

Poles and Zeros and Stability

$$H(z) = G(z) = \frac{(z - \zeta_1)(z - \zeta_2)\cdots(z - \zeta_L)}{(z - p_1)(z - p_2)\cdots(z - p_M)} = G(z) = \frac{\prod_{i=1}^{L}(z - \zeta_i)}{\prod_{i=1}^{M}(z - p_i)}$$

For stability of causal systems all the poles must be inside the unit circle, that is

$$|p_i| < 1 \quad \text{for all } i = 1, 2, \ldots, M$$

Random Signal Processing - Some Definitions

The mean is defined as:

$$\mu_x = \text{E}[x(n)] = \lim_{N \to \infty} \frac{1}{2N + 1} \sum_{n=-N}^{N} x(n)$$

Statistical expectation; for ergodic signals it is computed as a time average

The variance is a measure of dispersion from the mean

$$\sigma_x^2 = \text{E}[(x(n) - \mu_x)^2] = \text{E}[x^2(n)] - \mu_x^2$$
The Autocorrelation

\[ r_{xx}(m) = E[x(n+m)x(n)] = \lim_{N \to \infty} \frac{1}{2N+1} \sum_{n=-N}^{N} x(n+m)x(n) \]

Basic Properties:

- \( r_{xx}(-m) = r_{xx}(m) \)
- \( r_{xx}(0) \geq |r_{xx}(m)| \)

The filter acts as a correlator

The autocorrelation is a measure of predictability of the signal, i.e., a correlated signal would be one whose future values can be predicted from past values.

The Cross-correlation and the Cross-covariance

\[ r_{xy}(m) = E[x(n+m)y(n)] \]

The cross-correlation is a measure of similarity of two signals.

\[ r_{xy}(m) = r_{yx}(-m) \]

The Cross-correlation of the Output Process

\[ r_{yx}(m) = E[y(n+m)x(n)] \]

Another form

\[ r_{yx}(m) = h(m)*r_{xx}(m) \]
The channel is often modeled by a transfer function that can be determined by measuring the statistics of the signal at the receiver. Hence, by sending a training “white” sequence $x$ a cross-correlation $r_{xy}$ is measured and from that a channel impulse response

\[ r_{xy}(m) = \sum_{i=0}^{L} r_{xx}(m-i) b(i) \]

Calculating Autocorrelations
There are two kinds of autocorrelation estimates, i.e.,

Given an $N$-point data record $\{x(n), n = 0,1,2,\ldots,N-1\}$

a. Unbiased Estimates
\[ \hat{r}(m) = \frac{1}{N-|m|} \sum_{n=0}^{N-1} x(n+m)x(n) \]

b. Biased Estimates
\[ \tilde{r}(m) = \frac{1}{N} \sum_{n=0}^{N-1} x(n+m)x(n) \]

$-\left( N-1 \right) \leq m \leq N-1$

Preferred in most applications

ADAPTIVE FILTERING AND ITS APPLICATIONS
The idea is to make the filter $B(z)$ behave like $H(z)$.

To do this we try to get $B(z)$ to give the same output as $H(z)$.

System identification arises in many applications, such as adaptive noise and echo cancellation, channel equalization, active sound reduction, smart antennas, etc.

The key problems in adaptive system identification lie in the choice of the adaptive filtering algorithm. In particular, the following issues are of concern:

- presence of persistent and rich in frequency input (how good is the environment)
- adaptation speed (choice of adaptation gain)
- complexity of the algorithm (real time issues)
- quality of the final filter estimate (misadjustment)
Acoustic Echo in Telephones


The Adaptive Echo Canceller

Echo cancellation is achieved by subtracting a replica of the echo from the reflected signal s(n)

Smart (Adaptive) Antennas

Antenna Beamforming

References
The Adaptive Linear Combiner

This is essentially an FIR filter with adjustable coefficients.

\[
y(n) = d(n) - y(n) + e(n)
\]

In this system the filter coefficients are adjusted such that \( e(n) \) is minimized.

The Mean Square Error (MSE)

The Error

\[
e(n) = d(n) - y(n) + e(n)
\]

The MSE

\[
e = E[e^2(n)] = E[(d(n) - y(n))^2]
\]

Using vector notation, i.e.,

\[
E[e^2(n)] = E[d(n)^T d(n)] - 2E[d(n) x^T (n)] b + b^T E[x(n) x^T (n)] b
\]

where

\[p = E[d(n) x(n)] \text{ cross correlation vector}
\]

\[R = E[x(n) x^T (n)] \text{ autocorrelation matrix}
\]

The MSE Solution

Minimizing \( e \), i.e.,

\[
\nabla_e = \frac{\partial e}{\partial b} = 0
\]

we get

\[
b^0 = R^{-1} p
\]

where

\[
R = \begin{bmatrix}
r_{x}(0) & r_{x}(1) & r_{x}(2) & \cdots & r_{x}(l) \\
r_{x}(1) & r_{x}(0) & r_{x}(1) & \cdots & r_{x}(l-1) \\
r_{x}(2) & r_{x}(1) & r_{x}(0) & \cdots & r_{x}(l-2) \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
r_{x}(l) & r_{x}(l-1) & r_{x}(l-2) & \cdots & r_{x}(0)
\end{bmatrix}
\]

July 2007 Copyright 2007 (c) Andreas Spanias XII-31

The Adaptive Linear Combiner (Cont.)

\[
e(n) = d(n) - y(n)
\]

\[
e(n) = d(n) - \sum_{i=0}^{L} b_i x(n-i)
\]

\[
x(n) = \begin{bmatrix} x(n) \\ x(n-1) \\ x(n-2) \\ \vdots \\ x(n-L) \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}
\]

July 2007 Copyright 2007 (c) Andreas Spanias XII-30

The Mean Square Error (MSE)

The Error

\[
e(n) = d(n) - y(n) + e(n)
\]

The MSE

\[
e = E[e^2(n)] = E[(d(n) - y(n))^2]
\]

Using vector notation, i.e.,

\[
E[e^2(n)] = E[d(n)^T d(n)] - 2E[d(n) x^T (n)] b + b^T E[x(n) x^T (n)] b
\]

where

\[p = E[d(n) x(n)] \text{ cross correlation vector}
\]

\[R = E[x(n) x^T (n)] \text{ autocorrelation matrix}
\]
Remarks on the MSE Solution

Which implies that the error becomes uncorrelated with the inputs.

If the input is white noise then

\[ R = \sigma^2 \mathbf{I} \quad b^0 = \frac{p}{\sigma^2} \]

If \( R \) is a 2x2 matrix then the MSE can be described in terms of ellipses corresponding to constant MSE contours. MSE is minimum at the center of the ellipses.

Gradient Adaptive Filtering Algorithms

The Steepest Descent Algorithm (SDA)

\[ b(N+1) = b(N) - \mu \nabla_{\varepsilon}(N) \]

The SDA converges if

\[ 0 < \mu < \frac{1}{\lambda_{\text{max}}(R)} \]

The LMS is a steepest-descent algorithm where the gradient is calculated only from the present error sample.

The Newton Algorithm (NA)

\[ b(N+1) = b(N) - \mu R^{-1} \nabla_{\varepsilon}(N) \]

The NA converges if

\[ 0 < \mu < 1 \]

NA is faster than SDA in terms of convergence but requires a matrix inverse in every iteration.

Excess Mean Square Error and Misadjustment

This is defined at steady-state, i.e., after the adaptive transients vanish. The Excess MSE is due to noise in the adaptive process. The Excess MSE provides a measure of the difference between the actual and the optimal performance over time.

\[ \text{Excess MSE} = E[\varepsilon(k) - \varepsilon_{\text{min}}] \]

The misadjustment \( M \) is a normalized measure of the difference between the actual and the optimal performance.

\[ M = \frac{\text{Excess MSE}}{\varepsilon_{\text{min}}} \]
The LMS algorithm is due to Widrow. It is a steepest descent type of algorithm that uses an estimate of the gradient instead of the true gradient.

The Sequential LMS Algorithm

The SDA:

\[
\mathbf{b}(N+1) = \mathbf{b}(N) - \mu \hat{\nabla}_e(N)
\]

The LMS:

\[
\mathbf{b}(N+1) = \mathbf{b}(N) - \mu \nabla_e(N)
\]

where

\[
\hat{\nabla}_e(N) = -2e(N)\mathbf{x}(N)
\]

Convergence of the LMS

\[
\mathbf{b}(N+1) = \mathbf{b}(N) + 2\mu e(N)\mathbf{x}(N)
\]

It can be shown that

\[
\lim_{N \to \infty} E[b(N)] = b^0
\]

if

\[
0 < \mu < \frac{1}{\lambda_{\max}}
\]

A more conservative convergence condition

\[
0 < \mu < \frac{1}{\text{tr}(r)}
\]

Simulations LMS
Misadjustment of the LMS

It can be shown [Windrow and Stearns] that

\[ M \approx \mu \text{tr}(r) \]

Longest Learning time constant

\[ T_{\text{mse}} = \frac{1}{4\mu^2 \lambda_{\text{min}}} \]

Simulations LMS – Misadjustment and Excess MSE

Adaptive Filter Simulation Program (1)

Example 2.26 (file name: dsp2_26.m)

```matlab
clear
N=500; % total number of iterations
x=randn(N,1);%white Gaussian noise generated
Nf=256;
theta=(2*pi/Nf).*[0:Nf-1]; % precompute the set of discrete frequencies
L=2; %order of adaptive filter L=no. of coeff.-1
bhat(1:L+1)=0% initialized adaptive filter coefficients
mu=0.01 % step size
%********************************************************
% Fixed "unknown" filter
b=[1 1 1];
a=[1];
d = filter(b,a,x);% filter output
% Form output of the adaptive filter and calculate error
H=freqz(b,a,theta); % compute the frequency response of H(z)
%********************************************************
```
Example 2.26 (file name: dsp2_26.m) Page 2

```matlab
for n=L+1:N
    y(n)=0; %initialize
    for l=1:L+1
        y(n)=y(n)+bhat(l)*x(n-l+1); %compute output of adaptive filter
    end
    e(n)=d(n)-y(n); %compute error
    for l=1:L+1
        bhat(l)=bhat(l)+2*mu*e(n)*x(n-l+1); %adapt filter coefficients
    end
    jj=jj+1; %counter for plotting every 10 iterations
    if jj==10; %plot frequency responses every 10 iterations
        Hh=freqz(bhat,1,theta); %compute the frequency response (Hh(z))
        plot(theta,20*log10(abs(Hh)),theta,20*log10(abs(H))); %plot PSDs
        xlabel('frequency index')
        ylabel('Magnitude in (dB)')
        jj=0;
        pause; %press space bar to see the next frequency response
    end
end
plot(10*log10(e.*e+0.00000001)); %plot sample mse vs iterations
ylabel('MSE (dB)')
xlabel('Iteration')
```

Example 2.26a (file name: dsp2_26a.m)

```matlab
for n=L+1:N
    xv=x(n:-1:n-L);
    yh=bhat*xv; %compute output of adaptive filter
    e(n)=d(n)-yh; %compute error
    bhat=bhat+2*mu*e(n)*xv'; %adapt filter coefficients
    jj=jj+1; %counter for plotting every 10 iterations
    if jj==10; %plot frequency responses every 10 iterations
        Hh=freqz(bhat,1,theta); %compute the frequency response (Hh(z))
        plot(theta,20*log10(abs(Hh)),theta,20*log10(abs(H))); %plot PSDs
        xlabel('frequency index')
        ylabel('Magnitude in (dB)')
        jj=0;
        pause; %press space bar to see the next frequency response
    end
end
plot(10*log10(e.*e+0.00000001)); %plot sample mse vs iterations
ylabel('MSE (dB)')
xlabel('Iteration')
```

Vectorized Simulation of Adaptive Filter

```
for n=L+1:N
    yh=bhat*x(n:-1:n-L); %compute output of adaptive filter
    e(n)=d(n)-yh; %compute error
    bhat=bhat+2*mu*e(n)*x(n:-1:n-L)'; %adapt filter coefficients
    jj=jj+1; %counter for plotting every 10 iterations
    if jj==10; %plot frequency responses every 10 iterations
        Hh=freqz(bhat,1,theta); %compute the frequency response (Hh(z))
        plot(theta,20*log10(abs(Hh)),theta,20*log10(abs(H))); %plot PSDs
        xlabel('frequency index')
        ylabel('Magnitude in (dB)')
        jj=0;
        pause; %press space bar to see the next frequency response
    end
end
plot(10*log10(e.*e+0.00000001)); %plot sample mse vs iterations
ylabel('MSE (dB)')
xlabel('Iteration')
```

Adaptive Filter Simulation Program (2)

Adaptive Filter Simulation Program (3)

The simulation program gives the MSE curve as a function of iteration.

Block Algorithms
The block LMS algorithm, as opposed to the sequential LMS algorithm, minimizes a block (vector) of errors:

$$e(k) = E[e^{T}(k)e(k)]$$

or explicitly

$$[\begin{array}{c} e(k) \\ e(k+1) \\ e(k+2) \\ \vdots \\ e(k+N-1) \end{array}] = [\begin{array}{c} d(k) \\ d(k+1) \\ d(k+2) \\ \vdots \\ d(k+N-1) \end{array}]$$

The output vector can be written as

$$[\begin{array}{c} y(k) \\ y(k+1) \\ y(k+2) \\ \vdots \\ y(k+N-1) \end{array}] = [\begin{array}{c} x(k) \\ x(k-1) \\ x(k-2) \\ \vdots \\ x(k-N+1) \end{array}]b(k)$$

The block error

$$e(k) = d(k) - x(k)b(k)$$

The stability and misadjustment [Clark et al]

$$0 < \mu < \frac{1}{\lambda_{\text{max}}(R_{\beta})} \quad M \approx \mu tr(R_{\beta})$$

$$R_{\beta} = E[x_{\beta}(k)x_{\beta}^{T}(k)]$$

The Frequency Domain Adaptive Filter (FDAF)

This is due to [Dentino et al]

$$X_{\beta}(0)$$

$$\text{FFT}$$

$$E_{i}(0)$$

$$B_{i}(0)$$

$$\Sigma$$

$$D_{i}(0)$$

$$\text{FFT}$$

$$L = N - 1$$
The (FDAF) (Cont.)

The coefficient update expression for the FDAF is given by

\[ b(k + 1) = b(k) + 2\mu \sum_{d=0}^{H-1} (k) \hat{e}(k) \]

The stability condition is

\[ 0 < \mu < \frac{1}{\lambda_{\text{min}}(R_x)} \]

or

\[ 0 < \mu < \frac{1}{\max \{ \| X_k \| \} } \]

Remarks on the FDAF

The use of the FFT for fast convolution is associated with periodic (circular) as opposed to linear convolution.

The optimal solution for the FDAF is not the same as that of the LMS or the BLMS. In order to see the differences between the two solutions one has to examine the time domain equivalent of the FDAF.

The FDAF is a block frequency domain algorithm associated with a block time-domain circulant matrix.

The FLMS Algorithm

The FLMS algorithm is a frequency domain algorithm which is equivalent to the BLMS algorithm.

The FLMS uses 2N-point (augmented) FFTs to implement an Nth order BLMS adaptive filter in the frequency domain.

All the convolutions are done properly in the frequency domain using the overlap and save technique [Oppenheim].

In addition, the gradient is constrained such that it corresponds to that of the BLMS.
The Choice of the Convergence factor $\mu$

Much work has been done in choosing the convergence factor or the step size of gradient algorithms.

A procedure for optimizing the convergence factors was proposed by [Mikhael and Yassa]. Time-varying convergence factors for the LMS and the BLMS were proposed by [Mikhael and Wu].

Time-varying convergence factors for frequency domain algorithms were proposed by [Mikhael and Spanias].

Normalized convergence factors were also proposed by [Mansour and Gray].

Choice of $\mu$ for the LMS (Cont.)

It can be shown [Yassa] that

$$\mu_k = \frac{\sum_i^T (k) x_i(k)}{2 \sum_i^T (k) R x_i(k)}$$

Choice of $\mu$ for the BLMS

$$b(k+1) = b(k) - \mu_k x_d^H(k) \varepsilon(k)$$

$$\mu_k = \frac{e^T(k) x_d(k) x_d^T(k) \varepsilon(k)}{2 e^T(k) x_d(k) x_d^T(k) x_d(k) \varepsilon(k) \varepsilon(k)}$$

The last two equations describe the Optimum Block Algorithm (OBA)

Choice of $\mu$ for the FDAF

$$B(k+1) = B(k) - 2 \mu_k X_d^H(k) E(k)$$

$$\mu_k = \frac{\sum_{i=0}^{N-1} |e_i|^2 |x_i|^2}{\sum_{i=0}^{N-1} |e_i|^2 |x_i|^2}$$

Another frequency-depended gain is given below

$$\mu_k(t) = \frac{1}{L \sum_{j=k-L}^k |x_i(t)|^2}$$
Choice of $\mu$ for the FDAF and FLMS

In addition an individual $\mu_i$ for each coefficient was proposed by [Mikhael and Spanias]. This is given by

$$\mu_i(N+1) = \frac{1}{L} \sum_{j=L}^{N} |X_i(j)|^2$$

THE DATA STRUCTURE OF THE AUTOCORRELATION MATRIX AND THE RLS

$$R = \begin{bmatrix} x(n) \\ x(n-1) \\ x(n-2) \\ \vdots \\ x(n-L) \end{bmatrix}$$

$$\hat{R}(N) = \frac{N-1}{N} \hat{R}(N-1) + \frac{1}{N} x(N) x^T(N)$$

$$\tilde{R}(N) = \frac{N-1}{N} \tilde{R}(N-1) + \frac{1}{N} x(N) d(N)$$

THE WEIGHTED RLS (WRLS)

If the estimates of the autocorrelation matrix are modified such that a forgetting factor is introduced, i.e., current (recent) data is emphasized relative to older data then we get a modified time-recursive algorithm called the Weighted RLS (WRLS).

$$\hat{R}^\gamma(N+1) = \gamma \hat{R}^\gamma(N) + \frac{1}{\gamma} x(N+1) x^T(N+1)$$

$$P^\gamma(N+1) = \gamma P^\gamma(N) + \frac{1}{\gamma} x(N+1) d(N+1)$$

$$\left[ \hat{R}^\gamma(N+1) \right] = \gamma^2 \left[ \hat{R}^\gamma(N) \right] - \frac{\left[ \hat{R}^\gamma(N) \right] x(N) x^T(N) \left[ \hat{R}^\gamma(N) \right]^T}{\gamma^2 + x(N) \left[ \hat{R}^\gamma(N) \right] x^T(N)}$$
Simulations RLS vs LMS

IIR Adaptive Filtering

Inherent problems in IIR filtering is the stability of the filter during adaptation and the non-linearity of the optimization problem. There are basically two general techniques for adaptive filtering, namely, the equation error model and the output error model. The equation error model involves linear optimization and is stable during adaptation. The output error model involves non-linear optimization. Output error models are discussed by [Shunk]. In the following we describe the equation error model. The structure of the equation error model is given below:

The Equation Error Model (EEM)

The error equation for the EEM is given by

\[ e(n) = d(n) + \sum_{i=0}^{W} a_i d(n-i) - \sum_{j=0}^{M} b_j x(n-j) \]

Let us define the following (L+M+1) x 1 vectors

\[ y(n) = [ -x(n) \cdot x(n-1) \cdot x(n-2) \cdot x(d-1)] \]

and

\[ c(n) = [b_0(n) b_1(n-1) b_2(n) a_1(n) a_2(n) a_M(n)] \]

\[ e(n) = d(n) + y^T(n) c(n) \]
The Equation Error Model (Cont.)

The MSE

\[ \varepsilon = E[e^2(n)] = E[d^2(n)] - 2E[d(n)\hat{y}(n)] + \varepsilon^T E[y(n)y^T(n)]\varepsilon \]

Minimizing

\[ \nabla c = -\frac{\partial}{\partial c} = 0 \]

we get

\[ \nabla p = E[d(n)y(n)] \]

\[ \nabla e = \begin{bmatrix} \vdots \\ \vdots \\ \vdots \end{bmatrix} \]

\[ \nabla r = E[y(n)y^T(n)] \]

The LMS Algorithm for the EEM

\[ \hat{c}(k+1) = \hat{c}(k) - \mu \hat{\nabla}_e(k) \]

where

\[ \hat{\nabla}_e(k) = -2e(k)u(k) \]

Stable if

\[ 0 < \mu < \frac{1}{\lambda_{\text{max}}} \]

Misadjustment

\[ M \approx \mu \text{tr}(r) \]

The LMS Algorithm for Linear Prediction

\[ d(n) \]

\[ e(n) \]

\[ a(n+1) = a(n) + 2\mu d(n)e(n) \]

\[ e(n) = d(n) - \sum_{i=1}^{\infty} a_i(n)d(n-\ell) \]

WEB SITES AND REFERENCES
Web Site for JDSP PROGRAM

The JDSP program can be found at
http://jdsp.asu.edu

REFERENCES

DIGITAL SIGNAL PROCESSING


REFERENCES (2)

SPECTRAL ESTIMATION


ADAPTIVE FILTERS