Professor Brendan Morris, SEB 3216, brendan.morris@unlv.edu

EE482: Digital Signal Processing Applications

Spring 2014
TTh 14:30-15:45 CBC C222

Lecture 11
Adaptive Filtering
14/03/04

http://www.ee.unlv.edu/~b1morris/ee482/

Outline

- Random Processes
- Adaptive Filters
- LMS Algorithm

Adaptive Filtering

- FIR and IIR filters are designed for linear time-invariant signals

- How can we handle signals when the characteristics are unknown or changing?

- Need ways to update filter coefficients automatically and continually
  - Track time-varying signals and systems

Random Processes

- Real-world signals are time varying and have randomness in nature
  - E.g. speech, music, noise

- Need to characterize a signal even if full deterministic mathematical definition does not exist

- Random process – sequence of random variables

Autocorrelation

- Specifies statistical relationship of signal at different time lags $(n - k)$
  - $r_{xx}(n, k) = E[x(n), x(k)]$
  - Similarity of observations as a function of the time lag between them
- Mathematical tool for detecting signals
  - Repeating patterns (noise in sinusoid)
  - Measuring time-delay between signals
    - Radar, sonar, lidar
  - Estimation of impulse response
  - Etc.

Wide Sense Stationary (WSS) Process

- Random process statistics do not change with time
- Mean independent of time
  - $E[x(n)] = m_x$
- Autocorrelation only depends only on time lag
  - $r_{xx}(k) = E[x(n+k)x(n)]$
- WSS autocorrelation properties
  - Even function
    - $r_{xx}(-k) = r_{xx}(k)$
  - Bounded by 0 time lag
    - $|r_{xx}(k)| \leq r_{xx}(0) = E[x^2(n)]$
      - Zero mean process: $E[x^2(n)] = \sigma_x^2$
- Cross-correlation
  - $r_{xy}(k) = E[x(n+k)y(n)]$

Expected Value

- Value of random variable "expected" if random variable process repeated infinite number of times
  - Weighted average of all possible values
- Expectation operator
  - $E[.] = \int_{-\infty}^{\infty} .\, f(x)dx$
  - $f(x)$ – probability density function of random variable X

White Noise

- $v(n)$ with zero mean and variance σ_v^2
- Very popular random signal
  - Typical noise model
- Autocorrelation
  - $r_{vv}(k) = \sigma_v^2 \delta(k)$
  - Statistically uncorrelated except at zero time lag
- Power spectrum
  - $P_{vv}(\omega) = \sigma_v^2, \quad |\omega| \leq \pi$
  - Uniformly distributed over entire frequency range

Example 6.2

- Second-order FIR filter with white noise input
  - $y(n) = x(n) + ax(n-1) + bx(n-2)$
- Mean
  - $E[y(n)] = E[x(n) + ax(n-1) + bx(n-2)]$
  - $E[y(n)] = E[x(n)] + aE[x(n-1)] + bE[x(n-2)]$
  - $E[y(n)] = 0 + a \cdot 0 + b \cdot 0 = 0$
- Autocorrelation
  - $r_{yy}(k) = E[y(n+k)y(n)]$
  - $r_{yy}(k) = E\left[\begin{matrix}\big(x(n+k) + ax(n+k-1) + bx(n+k-2)\big) \cdot \\ \big(x(n) + ax(n-1) + bx(n-2)\big)\end{matrix}\right]$
  - $r_{yy}(k) = E[x(n+k)x(n)] + E[ax(n+k)x(n-1)] + \ \ldots$
  - $r_{yy}(k) = r_{xx}(k) + ar_{xx}(k-1) + \cdots$
  - $r_{yy}(k) = \begin{cases} (1 + a^2 + b^2)\sigma_x^2 & k = 0 \\ (a + ab)\sigma_x^2 & k = \pm 1 \\ b\sigma_x^2 & k = \pm 2 \\ 0 & else \end{cases}$

Practical Estimation

- Practical applications have finite length sequences
- Sample mean
  - $\overline{m_x} = \frac{1}{N}\sum_{n=0}^{N-1} x(n)$
- Sample autocorrelation
  - $\overline{r_{xx}}(k) = \frac{1}{N-k}\sum_{n-0}^{N-k-1} x(n+k)x(n)$
  - Only produces a good estimate of lags $< 10\%$ of N

- Use Matlab (`mean.m`, `xcorr.m`, etc.) to calculate

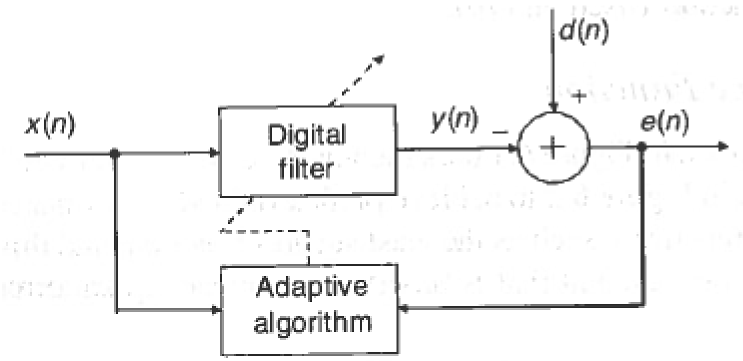
Adaptive Filters

- Signal characteristics in practical applications are time varying and/or unknown
- Must modify filter coefficients adaptively in an automated fashion to meet objectives

- Example: Channel equalization
  - High-speed data communication via media channel (e.g. wireless network)
  - Channel equalization compensates for channel distortion (e.g. path from wifi router and computer)
  - Channel must be continually tracked and characterized to compensate for distortion (e.g. moving around a room)

General Adaptive Filter

- Two components
  - Digital filter – defined by coefficients
  - Adaptive algorithm – automatically update filter coefficients (weights)



- Adaption occurs by comparing filtered signal $y(n)$ with a desired (reference) signal $d(n)$
  - Minimize error $e(n)$ using a cost function (e.g. mean-square error)
  - Continually lower error and get $y(n)$ closer to $d(n)$

FIR Adaptive Filter

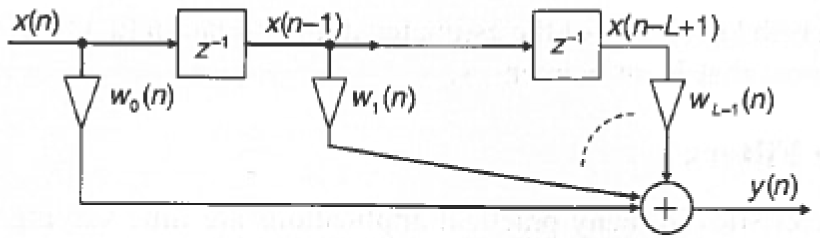


Figure 6.2 Block diagram of time-varying FIR filter for adaptive filtering

- $y(n) = \sum_{l=0}^{L-1} w_l(n)x(n-l)$
  - Notice time-varying weights
- In vector form
  - $y(n) = \boldsymbol{w}^T(n)\boldsymbol{x}(n) = \boldsymbol{x}^T(n)\boldsymbol{w}(n)$
  - $\boldsymbol{x}(n) = [x(n), x(n-1), \dots, x(n-L+1)]^T$
  - $\boldsymbol{w}(n) = [w_0(n), w_1(n), \dots, w_{L-1}(n)]^T$
- Error signal
  - $e(n) = d(n) - y(n) = d(n) - \boldsymbol{w}^T(n)\boldsymbol{x}(n)$

Performance Function

- Use mean-square error (MSE) cost function
- $\xi(n) = E[e^2(n)]$
- $\xi(n) = E[d^2(n)] - 2\boldsymbol{p}^T\boldsymbol{w}(n) + \boldsymbol{w}^T(n)\boldsymbol{R}\boldsymbol{w}(n)$
  - $\boldsymbol{p} = E[d(n)\boldsymbol{x}(n)] = [r_{dx}(0), r_{dx}(1), \dots, r_{dx}(L-1)]^T$
  - $\boldsymbol{R}$ – autocorrelation matrix
    - $\boldsymbol{R} = E[\boldsymbol{x}(n)\boldsymbol{x}^T(n)]$

$$= \begin{bmatrix} r_{xx}(0) & r_{xx}(1) & \dots & r_{xx}(L-1) \\ r_{xx}(1) & r_{xx}(0) & \dots & r_{xx}(L-2) \\ \vdots & \dots & \ddots & \vdots \\ r_{xx}(L-1) & r_{xx}(L-2) & \dots & r_{xx}(0) \end{bmatrix}, \qquad (6.22)$$

    - Toeplitz matrix – symmetric across main diagonal

Steepest Descent Optimization

- Error function is a quadratic surface
  - $\xi(n) = E[d^2(n)] - 2\boldsymbol{p}^T\boldsymbol{w}(n) + \boldsymbol{w}^T(n)\boldsymbol{R}\boldsymbol{w}(n)$
- Therefore gradient decent search techniques can be used
  - Gradient points in direction of greatest change
- Iterative optimization to "step" toward the bottom of error surface
  - $w(n+1) = w(n) - \frac{\mu}{2}\nabla\xi(n)$

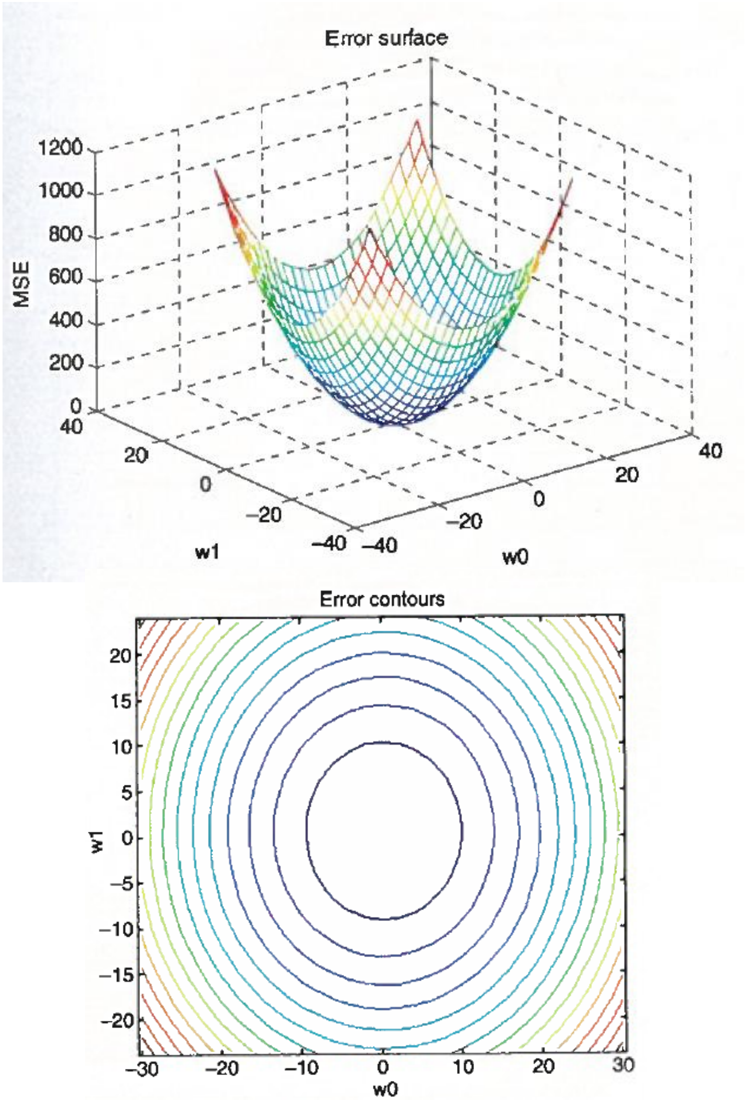


Figure 6.4 Examples of error surface (top) and error contours (bottom), $L=2$

LMS Algorithm

- Practical applications do not have knowledge of $d(n), x(n)$
  - Cannot directly compute MSE and gradient
  - Stochastic gradient algorithm
- Use instantaneous squared error to estimate MSE
  - $\hat{\xi}(n) = e^2(n)$
- Gradient estimate
  - $\nabla\hat{\xi}(n) = 2[\nabla e(n)]e(n)$
    - $e(n) = d(n) - w^T(n)x(n)$
  - $\nabla\hat{\xi}(n) = -2x(n)e(n)$
- Steepest descent algorithm
  - $w(n+1) = w(n) + \mu x(n)e(n)$

- LMS Steps

1. Set $L, \mu,$ and $\boldsymbol{w}(0)$
   - L – filter length
   - μ – step size (small e.g. 0.01)
   - $\boldsymbol{w}(0)$ – initial filter weights
2. Compute filter output
   - $y(n) = \boldsymbol{w}^T(n)\boldsymbol{x}(n)$
3. Compute error signal
   - $e(n) = d(n) - y(n)$
4. Update weight vector
   - $w_l(n+1) = w_l(n) + \mu x(n-l)e(n),$
     $l = 0,1, \ldots L-1$

- Notice this requires a reference signal