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 $\sigma_v^2$
- 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
  - $r_{yy}(k) = E[y(n + k)y(n)]$
  - $r_{yy}(k) = E\left[(x(n + k) + ax(n + k - 1) + bx(n + k - 2)) \cdot (x(n) + ax(n - 1) + bx(n - 2))\right]$
  - $r_{yy}(k) = E[x(n + k)x(n)] + E[ax(n + k)x(n - 1)] + ...$
  - $r_{yy}(k) = r_{xx}(k) + ar_{xx}(k - 1) + ...$
  - $r_{yy}(k) = \begin{cases} (1 + a^2 + b^2)\sigma_x^2 & k = 0 \\ (a + ab)\sigma_x^2 & k = \pm 1 \\ b\sigma_x^2 & k = \pm 2 \\ 0 & \text{else} \end{cases}$
Practical Estimation

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

- Use Matlab (\texttt{mean.m}, \texttt{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

- \( y(n) = \sum_{l=0}^{L-1} w_l(n)x(n - l) \)
  - Notice time-varying weights

- In vector form
  - \( y(n) = w^T(n)x(n) = x^T(n)w(n) \)
  - \( x(n) = [x(n), x(n - 1), \ldots, x(n - L + 1)]^T \)
  - \( w(n) = [w_0(n), w_1(n), \ldots, w_{L-1}(n)]^T \)

- Error signal
  - \( e(n) = d(n) - y(n) = d(n) - w^T(n)x(n) \)
Performance Function

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

- Toeplitz matrix – symmetric across main diagonal

\[
\begin{bmatrix}
  r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(L-1) \\
  r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(L-2) \\
  \vdots & \vdots & \ddots & \vdots \\
  r_{xx}(L-1) & r_{xx}(L-2) & \cdots & r_{xx}(0)
\end{bmatrix}
\]
Steepest Descent Optimization

• Error function is a quadratic surface
  \[ \xi(n) = E[d^2(n)] - 2p^T w(n) + w^T(n)Rw(n) \]

• 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
  ▫ \[ w(n + 1) = w(n) - \frac{\mu}{2} \nabla \xi(n) \]
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 \( L, \mu, \text{ and } w(0) \)
     - \( L \) – filter length
     - \( \mu \) – step size (small e.g. 0.01)
     - \( w(0) \) – initial filter weights
  2. Compute filter output
     - \( y(n) = w^T(n)x(n) \)
  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
LMS Stability

- Convergence of LMS algorithm
  - $0 < \mu < 2/\lambda_{max}$
    - $\lambda_{max}$ - largest eigenvalue of autocorrelation matrix $R$
    - Not easy to compute eigenvalues

- Eigenvalue approximation
  - $0 < \mu < 2/LP_x$
    - $L$ – length of data window, filter length
    - $P_x = r_{xx}(0) = E[x^2(n)]$

- Step size is inversely proportional to filter length
  - Smaller $\mu$ for higher order filters

- Step size inversely proportional to input signal power
  - Larger $\mu$ for lower power signal
Convergence Speed

- Convergence of filter weights is defined by the time $\tau_{MSE}$ to go from initial MSE to min
  - Plot of MSE vs. time is known as the learning curve
- Convergence time related to the minimum eigenvalue of $R$
  - $\tau_{MSE} \cong \frac{1}{\mu \lambda_{min}}$
    - Smaller step size results in longer convergence time
- In practice, weights will not converge to a fixed optimum value but will vary around it
Example 6.7

- \( sd = 12357; \) \( \text{rng}(sd); \) % Set seed value
- \( x = \text{randn}(1,128); \) % Reference signal \( x(n) \)
- \( b = [0.1,0.2,0.4,0.2,0.1]; \) % An FIR filter to be identified
- \( d = \text{filter}(b,1,x); \) % Desired signal \( d(n) \)
- \( \mu = 0.05; \) % Step size \( \mu \)
- \( h = \text{adaptfilt.lms}(5,\mu); \) % LMS algorithm
- \([y,e] = \text{filter}(h,x,d); \) % Adaptive filtering
- \( n = 1:128; \)
- \( h1=\text{figure}; \)
- \( \text{hold all}; \)
- \( \text{plot}(n,d,\'\-',\'linewidth', 3); \)
- \( \text{plot}(n,y,\'\-', \'linewidth', 3); \)
- \( \text{plot}(n,e,\'\--', \'linewidth', 2); \)
- \( \text{axis}([1 128 -inf inf]); \)
- \( \text{xlabel('Time index, n'}); \)
- \( \text{ylabel('Amplitude'}); \)
- \( \text{legend('d[n]', 'y[n]', 'e[n]');} \)
- \( [b; h.\text{coefficients}] \)

- **Coefficients**
  - \( b = [0.1000 \ 0.2000 \ 0.4000 \ 0.2000 \ 0.1000] \)
  - \( w = [0.1005 \ 0.1999 \ 0.3996 \ 0.1995 \ 0.0996] \)
Practical Applications

- Four classes of adaptive filtering applications
  - System identification
  - Prediction
  - Noise cancellation
  - Inverse modeling

- Differences based on configuration of control signals $x(n), d(n), y(n), e(n)$
System Identification

- Given an unknown system, try to determine (identify) coefficients

- Excite unknown system and adaptive system with same input
  - Input signal: white noise
  - Reference signal: output of unknown system
  - Error is difference between adaptive filter and the output of unknown system

Figure 6.7 Adaptive system identification using the LMS algorithm
Prediction

- Linear predictor estimates signal values at future times
- Reference signal: signal of interest
- Input signal: delayed reference signal
- Error is difference between current sample and predicted sample (using past samples)
  - Leverage correlation between samples
- Broadband output: noise component
- Narrowband output: signal of interest (high correlation)
Example 6.9

- \( F_s = 1000 \);
- \( f_0 = 150 \);
- \( L = 64 \);
- \( N = 256 \);
- \( A = \sqrt{2} \);
- \( w_0 = 2\pi f_0 / F_s \);
- \( n = [0: N-1] \);
- \( s_n = A \sin(w_0 \cdot n) \);
- \( v_n = 0.1 \cdot (\text{rand}(1, N) - 0.5) \cdot \sqrt{12} \);
- \( x = s_n + v_n \);
- \( d = [0, x(2:256)] \);
- \( \mu = 0.001 \);
- \( h = \text{adaptfilt.lms}(L, \mu) \);
- \([y, e] = \text{filter}(h, x, d) \);
- \( h1 = \text{figure} \);
- \( \text{hold on} \);
- \( \text{plot}(n, x, '-', 'linewidth', 2); \)
- \( \text{plot}(n, y, '-', 'linewidth', 2); \)
- \( \text{plot}(n, e, '--', 'linewidth', 2); \)
- \( \text{axis}([1 N -\infty \infty]); \)
- \( \text{xlabel('Time index, n')}; \)
- \( \text{ylabel('Amplitude')}; \)
- \( \text{legend('x[n]', 'y[n]', 'e[n]')} \);
Noise Cancellation

- Remove (cancel) noise components embedded in a primary signal
  - E.g. background noise in speech signal

- Flip idea of reference and input signals
  - Reference signal: primary signal + noise
    - Close to primary source
  - Input signal: noise signal
    - Far from primary source to measure noise
  - Adaptive filter tracks correlated noise
    - Error signal is the desired cleaned primary signal
Example 6.10

- Fs = 1000;
- f0 = 110;
- L = 3;
- N = 128;
- w0 = 2*pi*f0/Fs;
- pz = [0.1, 0.3, 0.2]; % Define noise path
- n = [0:N-1]; % Time index
- sd = 12357; rng(sd); % Set seed value

- sn = 0.5*sin(w0*n); % Sine sequence
- xn = 2.5*(rand(1,N)-0.5); % Zero-mean white noise
- xpn = filter(pz, 1, xn); % Generate x'(n)
- dn = sn+xpn; % Sinewave embedded in white noise

- mu = 0.025; % Step size mu
- h = adaptfilt.lms(L,mu); % LMS algorithm
- [y,e] = filter(h,xn,dn); % Adaptive filtering

- hold all;
- plot(n,dn,'-', 'linewidth', 2);
- plot(n,sn,'-', 'linewidth', 2);
- plot(n,e,'--', 'linewidth', 2);
- axis([1 N -inf inf]);
- xlabel('Time index, n');
- ylabel('Amplitude');
- legend('d[n] - noisy signal', 's[n] primary', 'e[n] - output');
Inverse Modeling

- Method to estimate the inverse of an unknown system
  - E.g. a communication channel is unknown but its distortion needs to be corrected

- Reference signal: a known training signal
- Input signal: training signal after going through unknown system

*Figure 6.14* An adaptive channel equalizer as an example of inverse modeling