(Lecture 13) Least Mean Square Adaptive Filters.

Consider steepest descent algorithm:

\[ \omega(n+1) = \omega(n) + \mu \left( -\frac{\partial J(\omega)}{\partial \omega} \right) \]

\[ = \omega(n) + \mu (p - P \omega(n)) \]

where, \( P \) is an estimate of the cross correlation vector:

\[ P = E \{ y(n) d(n) \} \]

\( P \) is an estimate of the correlation matrix:

\[ P = E \{ y(n) y^H(n) \} \]

In practice the estimates of cross correlation vector and correlation matrix are obtained through a time domain averaging (assuming ergodicity of the processes). In other words,

\[ P = \frac{1}{N} \sum_{n=0}^{N-1} y(n) d(n) \]

\[ P = \frac{1}{N} \sum_{n=0}^{N-1} y(n) y^H(n) \]

As \( N \) becomes larger, both estimates become more accurate, but the memory requirements placed on the algorithm become larger.

For the LMS algorithm the estimates of \( P \) and \( R \) are obtained as:

\[ \hat{P}(n) = y(n) d^H(n) \quad \text{in other words:} \quad N = 1 \]

\[ \hat{R}(n) = y(n) y^H(n) \]

Therefore, instantaneous estimate of the gradient is given as:

\[ \frac{\partial J(\omega)}{\partial \omega} \bigg|_{\omega = \omega(n)} = - \hat{P}(n) - \hat{R}(n) \omega(n) \]

\[ = - y(n) d(n) + y(n) y^H(n) \omega(n) \]
\[ \nabla J(n) = -\mu(n) \cdot [P(n-1) - \mu^H(n) \cdot \omega(n)] = \\
= -\mu(n) \cdot [P(n-1) - \omega^H(n) \cdot \mu(n)]^* = \\
= -\mu(n) \cdot \epsilon^*(n) \]

\[ \nabla J(n) = \text{ stochastic gradient} \]

Substituting the estimate of the gradient in Steepest descent equations one obtains:

\[ \omega(n+1) = \omega(n) - \mu \frac{\partial J(\omega)}{\partial \omega} \bigg|_{\omega=\omega(n)} = \\
= \omega(n) - \mu \nabla J(n) = \omega(n) + \mu \mu(n) \cdot \epsilon^*(n) \]

One identifies three steps in the LMS algorithm:

1° Filtering,

\[ y(n) = \omega^H(n) \cdot \mu(n) \]

2° Estimation of error

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

3° Update of tap-weights

\[ \omega(n+1) = \omega(n) + \mu(n) \cdot \epsilon^*(n) \]
Signal flow graph for LMS algorithm:

1) $M$ - multiplications, $H-1$ - addition
2) $I$ - addition
3) $I$ - multiplications
4) $M$ - multiplications
5) $M$ - additions

$2M+1$ multiplications, $2M$ - additions per each iteration.

Therefore, the complexity of the algorithm is $O(M)$ - increase linearly with an increase of the filter's size. From the practical standpoint a very desirable property.

- Instantaneous estimates of $P$ & $P$ have large variances
- Due to its recursive nature, the algorithm is self-correcting
- LMS can be applied in:
  - Stochastic environment (same as steepest descent)
  - Deterministic environment (from linear regression = Adaline - neural network)
  - Non-stochastic environment (tracking applications)
Stability of the LMS algorithm

LMS algorithm is built around linear combiner.
LMS filter is a complex and nonlinear estimator.

Consider the update equation for LMS filter:

\[ \omega(n+1) = \omega(n) + \mu y(n) e^*(n) \]

Assume \( \omega(0) = 0 \). Then

\[ \omega(1) = \omega(0) + \mu y(0) e^*(0) = \mu y(0) e^*(0) \]

\[ \omega(2) = \omega(1) + \mu y(1) e^*(1) = \mu y(0) e^*(0) + \mu y(1) e^*(1) \]

\[ = \mu \sum_{i=0}^{n-1} y(i) e^*(i) \]

\[ \omega(n) = \mu \sum_{i=1}^{n-1} y(i) e^*(i) = \mu \sum_{i=0}^{n-1} y(i) \left[ d(i) - \omega^H(i) y(i) \right]^* \]

The output of the filter:

\[ y(n) = \omega(n) u(n) = \left( \mu \sum_{i=0}^{n-1} y(i) \left[ d(i) - \omega^H(i) y(i) \right]^* \right) u(n) \]

\[ = \frac{1}{2} \left[ \begin{bmatrix} y(1) \end{bmatrix} \sum_{i=0}^{n-1} y(i) \right] \Rightarrow \text{nonlinear function of an input sequence} \]

Since output is a nonlinear function of the input sequence, the superposition does not apply. In other words:

\[ u_1(n) \rightarrow y_1(n), \quad u_2(n) \rightarrow y_2(n), \quad i = 1, 2 \]

\[ y_1(n) \neq y_2(n) \]
For LMS algorithm does not apply.

\[ u(n) = u_1(n) + u_2(n) \neq y_1(n) + y_2(n) \]

- Due to nonlinearity analysis of convergence is extremely complex.
- Shall open research issue
- Only recently a bound on \( \mu \) has been derived (necessary)

\[ 0 < \mu < \frac{2}{M \cdot S(w)} \text{, for all } \omega \text{ and } \text{range } H \]

That is

\[ 0 < \mu < \frac{2}{M \cdot \text{Smax}} \text{, Smax maximum value of } u(n) \text{, PSD } (x) \]

where \( S(w) = \text{DFT} \{ u(n) \} \)

\[ S(w) = \sum_{k=1}^{M} \mu(k) e^{-j\frac{2\pi k}{M}} \text{ (Bhulkade result)} \]

- Practical applications of the result in (x) are limited
- Usually \( \mu \) is set on a basis of experimentation (previous experience)
- One commonly used guideline

\[ \mu = 0.1 \cdot \frac{2}{M \cdot E[u(n)^2]} = 0.1 \cdot \frac{2}{M \cdot P_u} \]

where \( M \) - number of taps.
\( P_u \) - power of the signal

One recognizes, \( M \cdot P_u \approx \text{max} \{ P \} \)
Some applications of the LMS algorithm

- Evaluation of complex communication channel
- Adaptive deconvolution
- Instantaneous frequency measurements
- Adaptive multi-pole cancellation of sinusoidal signal
- Adaptive tone enhancer
- Adaptive beamforming

Homework problem

1) Demonstrate that

\[
\hat{\theta}(n) = \frac{\epsilon(n)}{|\epsilon(n)|^2}
\]

\[\epsilon(n)\] - Instantaneous error