FSAN/ELEG815: Statistical Learning

Gonzalo R. Arce

Department of Electrical and Computer Engineering
University of Delaware

VII: Adaptive Optimization (Steepest Descent and LMS Algorithms)

Outline of the Course

1. Review of Probability
2. Stationary processes
3. Eigen Analysis, Singular Value Decomposition (SVD) and Principal Component Analysis (PCA)
4. The Learning Problem
5. Training vs Testing
6. The Wiener Filter
7. Adaptive Optimization: Steepest descent and the LMS algorithm
8. Overfitting and Regularization
9. Logistic, Ridge and Lasso regression.
10. Neural Networks
11. Matrix Completion

Adaptive Optimization and Filtering Methods

Motivation

Adaptive optimization and filtering methods are appropriate, advantageous, or necessary when:

- Signal statistics are not known *a priori* and must be "learned" from observed or representative samples
- Signal statistics evolve over time
- Time or computational restrictions dictate that simple, if repetitive, operations be employed rather than solving more complex, closed form expressions
- To be considered are the following algorithms:
  - Steepest Descent (SD) – deterministic
  - Least Means Squared (LMS) – stochastic
  - Recursive Least Squares (RLS) – deterministic

Definition (Steepest Descent (SD))

Steepest descent, also known as gradient descent, it is an iterative technique for finding the local minimum of a function.

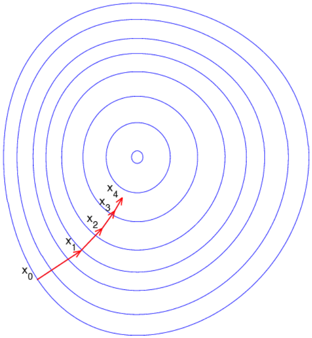
Approach: Given an arbitrary starting point, the current location (value) is moved in steps proportional to the negatives of the gradient at the current point.

- SD is an old, deterministic method, that is the basis for stochastic gradient based methods
- SD is a feedback approach to finding local minimum of an error performance surface
- The error surface must be known *a priori*
- In the MSE case, SD converges converges to the optimal solution, $\mathbf{w}_0 = \mathbf{R}^{-1}\mathbf{p}$, without inverting a matrix

  Question: Why in the MSE case does this converge to the global minimum rather than a local minimum?

Example

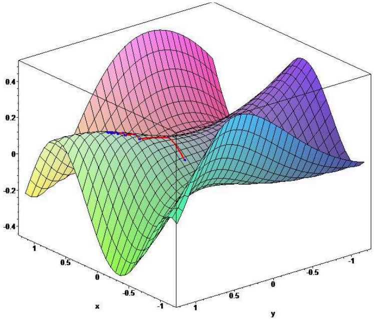
Consider a well structured cost function with a single minimum. The optimization proceeds as follows:



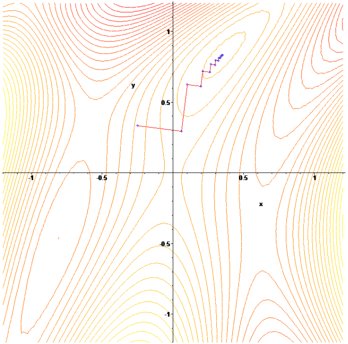
Contour plot showing that evolution of the optimization

Example

Consider a gradient ascent example in which there are multiple minima/maxima

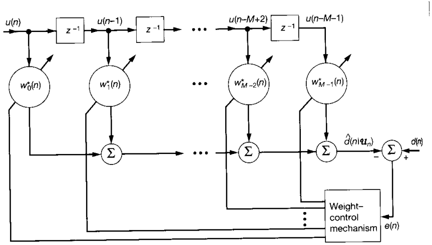


Surface plot showing the multiple minima and maxima



Contour plot illustrating that the final result depends on starting value

To derive the approach, consider the FIR case:



$\{x(n)\}$ – the WSS input samples
$\{d(n)\}$ – the WSS desired output
$\{\hat{d}(n)\}$ – the estimate of the desired signal given by

$$\hat{d}(n) = \mathbf{w}^H(n)\mathbf{x}(n)$$

where

$$\mathbf{x}(n) = [x(n), x(n-1), \cdots, x(n-M+1)]^T \qquad \text{[obs. vector]}$$
$$\mathbf{w}(n) = [w_0(n), w_1(n), \cdots, w_{\text{M-1}}(n)]^T \qquad \text{[time indexed filter coefs.]}$$

Then similarly to previously considered cases

$$e(n) = d(n) - \hat{d}(n) = d(n) - \mathbf{w}^H(n)\mathbf{x}(n)$$

and the MSE at time n is

$$\begin{aligned} J(n) &= E\{|e(n)|^2\} \\ &= \sigma_d^2 - \mathbf{w}^H(n)\mathbf{p} - \mathbf{p}^H\mathbf{w}(n) + \mathbf{w}^H(n)\mathbf{R}\mathbf{w}(n) \end{aligned}$$

where

σ_d^2 – variance of desired signal

$\mathbf{p}$ – cross-correlation between $\mathbf{x}(n)$ and $d(n)$

$\mathbf{R}$ – correlation matrix of $\mathbf{x}(n)$

Note: The weight vector and cost function are time indexed (functions of time)

When $\mathbf{w}(n)$ is set to the (optimal) Wiener solution,

$$\mathbf{w}(n) = \mathbf{w}_0 = \mathbf{R}^{-1}\mathbf{p}$$

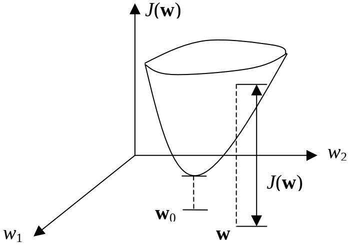
and

$$J(n) = J_{\min} = \sigma_d^2 - \mathbf{p}^H \mathbf{w}_0$$

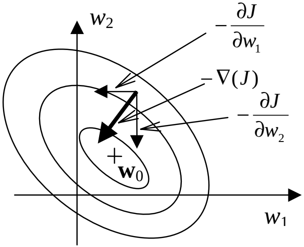
- Use the method of steepest descent to iteratively find $\mathbf{w}_0$.
- The optimal result is achieved since the cost function is a second order polynomial with a single unique minimum

Example

Let $M = 2$. The MSE is a bowl–shaped surface, which is a function of the 2-D space weight vector $\mathbf{w}(n)$



Surface Plot



Contour Plot

Imagine dropping a marble at any point on the bowl-shaped surface.
The ball will reach the minimum point by going through the path of steepest descent.

Observation: Set the direction of filter update as: $-\nabla J(n)$

Resulting Update:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{1}{2}\mu[-\nabla J(n)]$$

or, since $\nabla J(n) = -2\mathbf{p} + 2\mathbf{R}\mathbf{w}(n)$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu[\mathbf{p} - \mathbf{R}\mathbf{w}(n)] \quad n = 0, 1, 2, \cdots$$

where $\mathbf{w}(0) = \mathbf{0}$ (or other appropriate value) and μ is the step size

Observation: SD uses feedback, which makes it possible for the system to be unstable

- Bounds on the step size guaranteeing stability can be determined with respect to the eigenvalues of $\mathbf{R}$ (Widrow, 1970)

Convergence Analysis

Define the error vector for the tap weights as

$$\mathbf{c}(n) = \mathbf{w}(n) - \mathbf{w}_0$$

Then using $\mathbf{p} = \mathbf{R}\mathbf{w}_0$ in the update,

$$\begin{aligned}
\mathbf{w}(n+1) &= \mathbf{w}(n) + \mu[\mathbf{p} - \mathbf{R}\mathbf{w}(n)] \\
&= \mathbf{w}(n) + \mu[\mathbf{R}\mathbf{w}_0 - \mathbf{R}\mathbf{w}(n)] \\
&= \mathbf{w}(n) - \mu\mathbf{R}\mathbf{c}(n)
\end{aligned}$$

and subtracting $\mathbf{w}_0$ from both sides

$$\begin{aligned}
\mathbf{w}(n+1) - \mathbf{w}_0 &= \mathbf{w}(n) - \mathbf{w}_0 - \mu\mathbf{R}\mathbf{c}(n) \\
\Rightarrow \mathbf{c}(n+1) &= \mathbf{c}(n) - \mu\mathbf{R}\mathbf{c}(n) \\
&= [\mathbf{I} - \mu\mathbf{R}]\mathbf{c}(n)
\end{aligned}$$

Using the Unitary Similarity Transform

$$\mathbf{R} = \mathbf{Q}\mathbf{\Omega}\mathbf{Q}^H$$

we have

$$\begin{aligned} \mathbf{c}(n+1) &= [\mathbf{I} - \mu\mathbf{R}]\mathbf{c}(n) \\ &= [\mathbf{I} - \mu\mathbf{Q}\mathbf{\Omega}\mathbf{Q}^H]\mathbf{c}(n) \\ \Rightarrow \mathbf{Q}^H\mathbf{c}(n+1) &= [\mathbf{Q}^H - \mu\mathbf{Q}^H\mathbf{Q}\mathbf{\Omega}\mathbf{Q}^H]\mathbf{c}(n) \\ &= [\mathbf{I} - \mu\mathbf{\Omega}]\mathbf{Q}^H\mathbf{c}(n) \qquad (*) \end{aligned}$$

Define the transformed coefficients as

$$\begin{aligned} \mathbf{v}(n) &= \mathbf{Q}^H\mathbf{c}(n) \\ &= \mathbf{Q}^H(\mathbf{w}(n) - \mathbf{w}_0) \end{aligned}$$

Then $(*)$ becomes

$$\mathbf{v}(n+1) = [\mathbf{I} - \mu\mathbf{\Omega}]\mathbf{v}(n)$$

Consider the initial condition of $\mathbf{v}(n)$

$$\begin{aligned}\mathbf{v}(0) &= \mathbf{Q}^H(\mathbf{w}(0)-\mathbf{w}_0)\\ &= -\mathbf{Q}^H\mathbf{w}_0 \qquad\qquad [\text{if } \mathbf{w}(0)=\mathbf{0}]\end{aligned}$$

Consider the k^{th} term (mode) in

$$\mathbf{v}(n+1) = [\mathbf{I}-\mu\mathbf{\Omega}]\mathbf{v}(n)$$

- Note $[\mathbf{I}-\mu\mathbf{\Omega}]$ is diagonal
- Thus all modes are independently updated
- The update for the k^{th} term can be written as

$$v_k(n+1) = (1-\mu\lambda_k)v_k(n) \quad k=1,2,\cdots,M$$

or using recursion

$$v_k(n) = (1-\mu\lambda_k)^n v_k(0)$$

Observation: Conversion to the optimal solution requires

$$\begin{aligned}
\lim_{n\to\infty} \mathbf{w}(n) &= \mathbf{w}_0 \\
\Rightarrow \lim_{n\to\infty} \mathbf{c}(n) &= \lim_{n\to\infty} \mathbf{w}(n) - \mathbf{w}_0 = \mathbf{0} \\
\Rightarrow \lim_{n\to\infty} \mathbf{v}(n) &= \lim_{n\to\infty} \mathbf{Q}^H \mathbf{c}(n) = \mathbf{0} \\
\Rightarrow \lim_{n\to\infty} v_k(n) &= 0 \quad k = 1, 2, \cdots, M \qquad (*)
\end{aligned}$$

Result: According to the recursion

$$v_k(n) = (1 - \mu\lambda_k)^n v_k(0)$$

the limit in $(*)$ holds if and only if

$$|1 - \mu\lambda_k| < 1 \quad \text{for all } k$$

Thus since the eigenvalues are nonnegative, $0 < \mu\lambda_{\max} < 2$, or

$$0 < \mu < \frac{2}{\lambda_{\max}}$$

Observation: The k^{th} mode has geometric decay

$$v_k(n) = (1-\mu\lambda_k)^n v_k(0)$$

- The rate of decay it is characterized by the time it takes to decay to e^{-1} of the initial value
- Let τ_k denote this time for the k^{th} mode

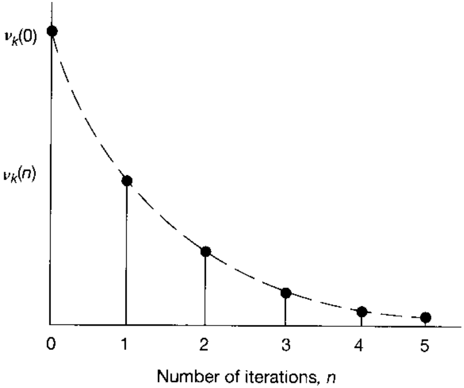
$$\begin{aligned} v_k(\tau_k) &= (1-\mu\lambda_k)^{\tau_k} v_k(0) = e^{-1} v_k(0) \\ \Rightarrow e^{-1} &= (1-\mu\lambda_k)^{\tau_k} \\ \Rightarrow \tau_k &= \frac{-1}{\ln(1-\mu\lambda_k)} \approx \frac{1}{\mu\lambda_k} \quad \text{for} \quad \mu \ll 1 \end{aligned}$$

Result: The overall rate of decay is

$$\frac{-1}{\ln(1-\mu\lambda_{\max})} \leq \tau \leq \frac{-1}{\ln(1-\mu\lambda_{\min})}$$

Example

Consider the typical behavior of a single mode



Error Analysis

Recall that

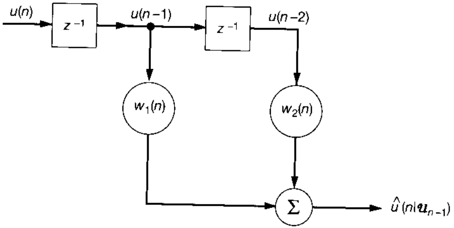
$$\begin{aligned} J(n) &= J_{\min} + (\mathbf{w}(n) - \mathbf{w}_0)^H \mathbf{R} (\mathbf{w}(n) - \mathbf{w}_0) \\ &= J_{\min} + (\mathbf{w}(n) - \mathbf{w}_0)^H \mathbf{Q} \boldsymbol{\Omega} \mathbf{Q}^H (\mathbf{w}(n) - \mathbf{w}_0) \\ &= J_{\min} + \mathbf{v}(n)^H \boldsymbol{\Omega} \mathbf{v}(n) \\ &= J_{\min} + \sum_{k=1}^{M} \lambda_k |v_k(n)|^2 \qquad \text{[sub in } v_k(n) = (1-\mu\lambda_k)^n v_k(0)] \\ &= J_{\min} + \sum_{k=1}^{M} \lambda_k (1-\mu\lambda_k)^{2n} |v_k(0)|^2 \end{aligned}$$

Result: If $0 < \mu < \frac{2}{\lambda_{\max}}$,
then

$$\lim_{n \to \infty} J(n) = J_{\min}$$

Example

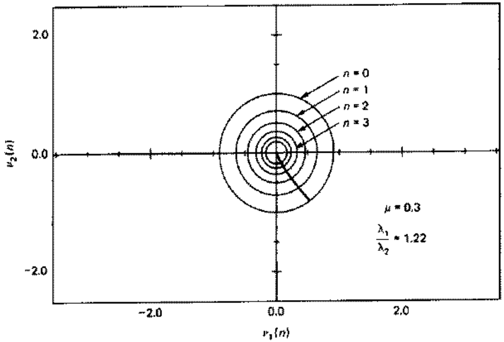
Consider a two–tap predictor for real–valued input

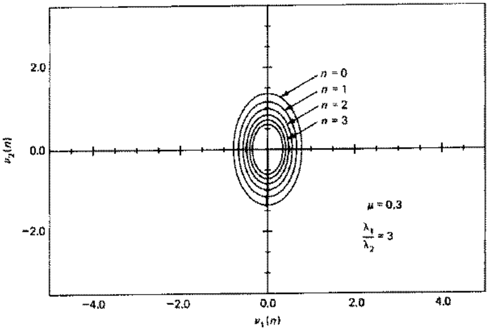


Analyzed the effects of the following cases:

- Varying the eigenvalue spread $\chi(\mathbf{R}) = \frac{\lambda_{\max}}{\lambda_{\min}}$ while keeping μ fixed
- Varying μ and keeping the eigenvalue spread $\chi(\mathbf{R})$ fixed

SD loci plots (with shown $J(n)$ contours) as a function of $[v_1(n), v_2(n)]$ for step-size $\mu = 0.3$

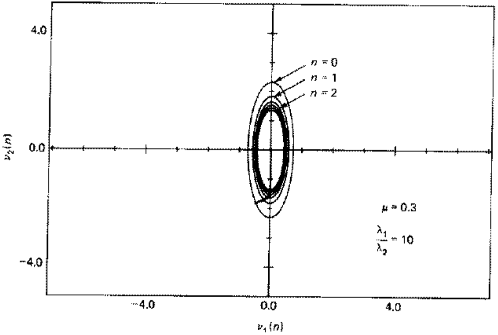


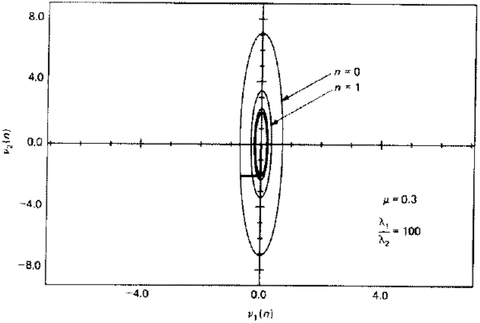


- Eigenvalue spread: $\chi(\mathbf{R}) = 1.22$
- Small eigenvalue spread $\Rightarrow$ modes converge at a similar rate

- Eigenvalue spread: $\chi(\mathbf{R}) = 3$
- Moderate eigenvalue spread $\Rightarrow$ modes converge at moderately similar rates

SD loci plots (with shown $J(n)$ contours) as a function of $[v_1(n), v_2(n)]$ for step-size $\mu = 0.3$

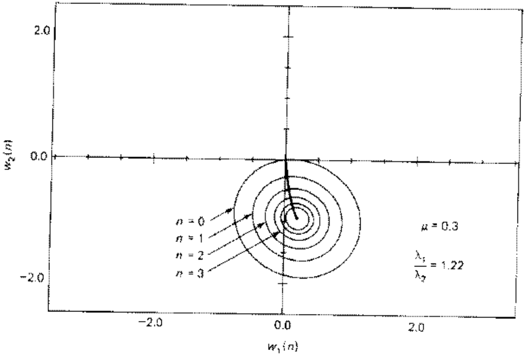


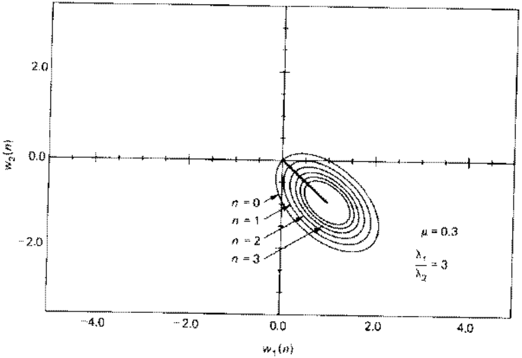


- Eigenvalue spread: $\chi(\mathbf{R}) = 10$
- Large eigenvalue spread $\Rightarrow$ modes converge at different rates

- Eigenvalue spread: $\chi(\mathbf{R}) = 100$
- Very large eigenvalue spread $\Rightarrow$ modes converge at very different rates
- Principle direction convergence is fastest

SD loci plots (with shown $J(n)$ contours) as a function of $[w_1(n), w_2(n)]$ for step-size $\mu = 0.3$

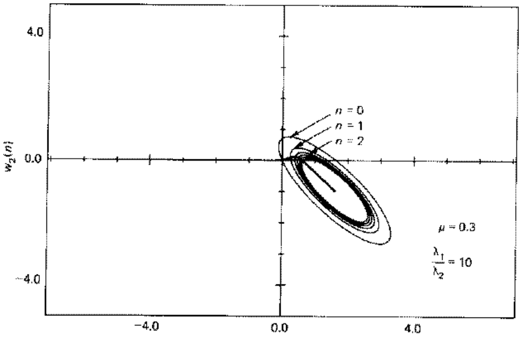


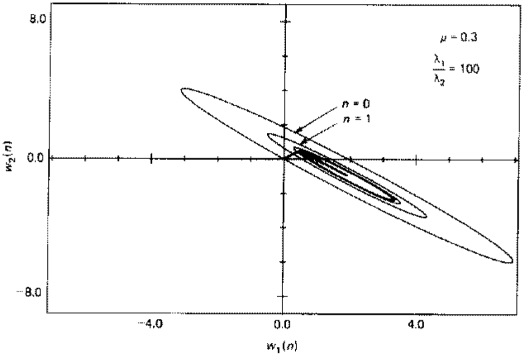


- Eigenvalue spread: $\chi(\mathbf{R}) = 1.22$
- Small eigenvalue spread $\Rightarrow$ modes converge at a similar rate

- Eigenvalue spread: $\chi(\mathbf{R}) = 3$
- Moderate eigenvalue spread $\Rightarrow$ modes converge at moderately similar rates

SD loci plots (with shown $J(n)$ contours) as a function of $[w_1(n), w_2(n)]$ for step-size $\mu = 0.3$

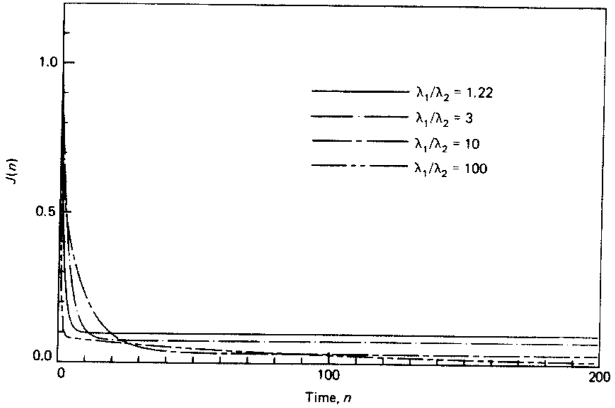




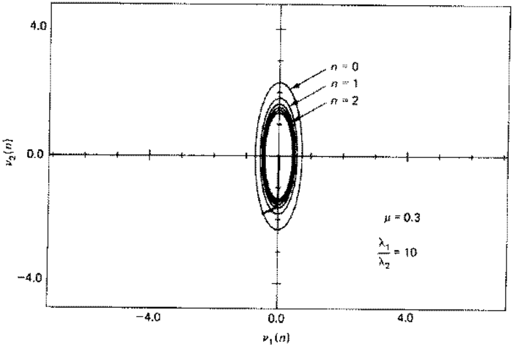
- Eigenvalue spread: $\chi(\mathbf{R}) = 10$
- Large eigenvalue spread $\Rightarrow$ modes converge at different rates

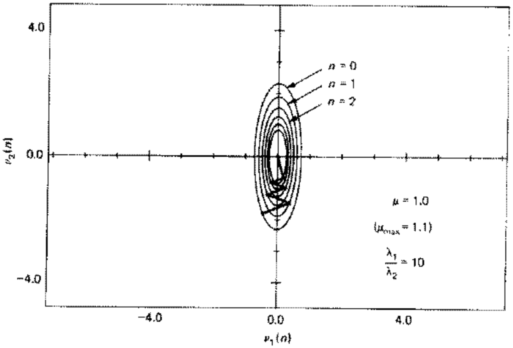
- Eigenvalue spread: $\chi(\mathbf{R}) = 100$
- Very large eigenvalue spread $\Rightarrow$ modes converge at very different rates
- Principle direction convergence is fastest

Learning curves of steepest-descent algorithm with step-size parameter $\mu = 0.3$ and varying eigenvalue spread.



SD loci plots (with shown $J(n)$ contours) as a function of $[v_1(n), v_2(n)]$ with $\chi(\mathbf{R}) = 10$ and varying step–sizes

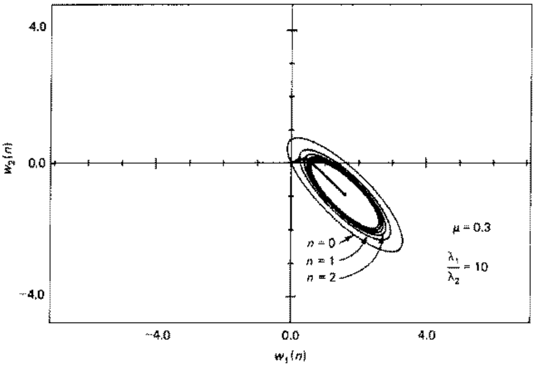


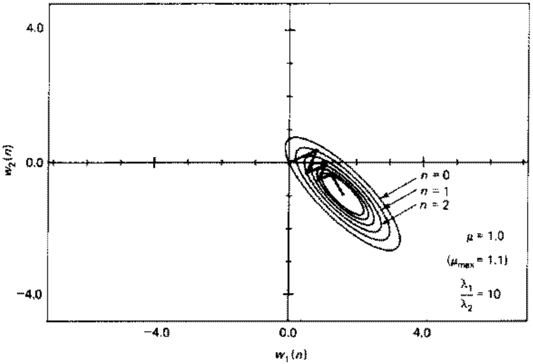


- Step–sizes: $\mu = 0.3$
- This is over–damped $\Rightarrow$ slow convergence

- Step–sizes: $\mu = 1$
- This is under–damped $\Rightarrow$ fast (erratic) convergence

SD loci plots (with shown $J(n)$ contours) as a function of $[w_1(n), w_2(n)]$ with $\chi(\mathbf{R}) = 10$ and varying step–sizes



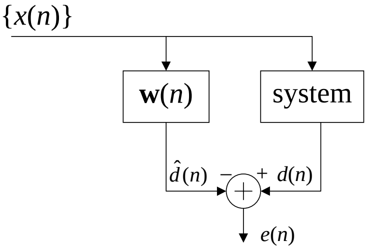


- Step–sizes: $\mu = 0.3$
- This is over–damped $\Rightarrow$ slow convergence

- Step–sizes: $\mu = 1$
- This is under–damped $\Rightarrow$ fast (erratic) convergence

Example

Consider a system identification problem



Suppose $M = 2$ and

$$\mathbf{R_x} = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix} \quad \mathbf{P} = \begin{bmatrix} 0.8 \\ 0.5 \end{bmatrix}$$

From eigen analysis we have

$$\lambda_1 = 1.8, \lambda_2 = 0.2 \Rightarrow \mu < \frac{2}{1.8}$$

also

$$\mathbf{q}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \mathbf{q}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

and

$$\mathbf{Q} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Also,

$$\mathbf{w}_0 = \mathbf{R}^{-1}\mathbf{p} = \begin{bmatrix} 1.11 \\ -0.389 \end{bmatrix}$$

Thus

$$\mathbf{v}(n) = \mathbf{Q}^H[\mathbf{w}(n) - \mathbf{w}_0]$$

Noting that

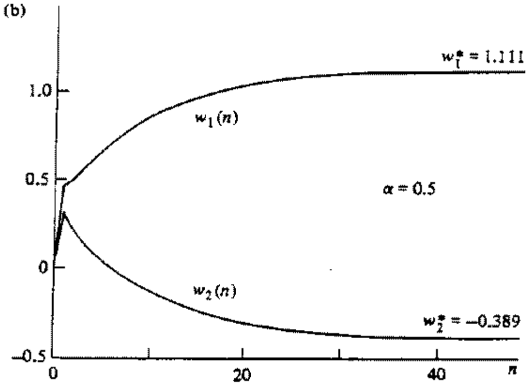
$$\mathbf{v}(0) = -\mathbf{Q}^H\mathbf{w}_0 = -\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 1.11 \\ -0.389 \end{bmatrix} = \begin{bmatrix} 0.51 \\ 1.06 \end{bmatrix}$$

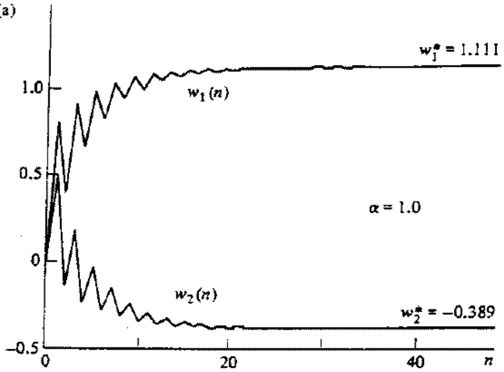
and

$$v_1(n) = (1 - \mu(1.8))^n 0.51$$

$$v_2(n) = (1 - \mu(0.2))^n 1.06$$

SD convergence properties for two μ values





- Step–sizes: $\mu = 0.5$
- This is over–damped $\Rightarrow$ slow convergence

- Step–sizes: $\mu = 1$
- This is under–damped $\Rightarrow$ fast (erratic) convergence

Least Mean Squares (LMS) Algorithm

Definition

Motivation: The error performance surface used by the SD method is not always known *a priori*

Solution: Use estimated values. We will use the following instantaneous estimates

$$\hat{\mathbf{R}}(n) = \mathbf{x}(n)\mathbf{x}^H(n)$$

$$\hat{\mathbf{p}}(n) = \mathbf{x}(n)d^*(n)$$

Result: The estimates are RVs and thus this leads to a stochastic optimization

Historical Note: Invented in 1960 by Stanford University professor Bernard Widrow and his first Ph.D. student, Ted Hoff

Recall the SD update

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{1}{2}\mu[-\nabla(J(n))]$$

where the gradient of the error surface at $\mathbf{w}(n)$ was shown to be

$$\nabla(J(n)) = -2\mathbf{p} + 2\mathbf{R}\mathbf{w}(n)$$

Using the instantaneous estimates,

$$\begin{aligned}
\hat{\nabla}(J(n)) &= -2\mathbf{x}(n)d^*(n) + 2\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{w}(n) \\
&= -2\mathbf{x}(n)[d^*(n) - \mathbf{x}^H(n)\mathbf{w}(n)] \\
&= -2\mathbf{x}(n)[d^*(n) - \hat{d}^*(n)] \\
&= -2\mathbf{x}(n)e^*(n)
\end{aligned}$$

where $e^*(n)$ is the complex conjugate of the estimate error.

Utilizing $\nabla(J(n)) = -2\mathbf{x}(n)e^*(n)$ in the update

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) + \frac{1}{2}\mu[-\nabla(J(n))] \\ &= \mathbf{w}(n) + \mu\mathbf{x}(n)e^*(n) \qquad \text{[LMS Update]}\end{aligned}$$

- The LMS algorithm belongs to the family of stochastic gradient algorithms
- The update is extremely simple
- Although the instantaneous estimates may have large variance, the LMS algorithm is recursive and effectively averages these estimates
- The simplicity and good performance of the LMS algorithm make it the benchmark against which other optimization algorithms are judged

Convergence Analysis

Independence Theorem

The following conditions hold:

1. The vectors $\mathbf{x}(1), \mathbf{x}(2), \cdots, \mathbf{x}(n)$ are statistically independent
2. $\mathbf{x}(n)$ is independent of $d(1), d(2), \cdots, d(n-1)$
3. $d(n)$ is statistically dependent on $\mathbf{x}(n)$, but is independent of $d(1), d(2), \cdots, d(n-1)$
4. $\mathbf{x}(n)$ and $d(n)$ are mutually Gaussian

- The independence theorem is invoked in the LMS algorithm analysis
- The independence theorem is justified in some cases, e.g., beamforming where we receive independent vector observations
- In other cases it is not well justified, but allows the analysis to proceeds (i.e., when all else fails, invoke simplifying assumptions)

We will invoke the independence theorem to show that $\mathbf{w}(n)$ converges to the optimal solution in the mean

$$\lim_{n\to\infty} E\{\mathbf{w}(n)\} = \mathbf{w}_0$$

To prove this, evaluate the update

$$\begin{aligned}
\mathbf{w}(n+1) &= \mathbf{w}(n) + \mu\mathbf{x}(n)e^*(n) \\
\Rightarrow \mathbf{w}(n+1) - \mathbf{w}_0 &= \mathbf{w}(n) - \mathbf{w}_0 + \mu\mathbf{x}(n)e^*(n) \\
\Rightarrow \mathbf{c}(n+1) &= \mathbf{c}(n) + \mu\mathbf{x}(n)(d^*(n) - \mathbf{x}^H(n)\mathbf{w}(n)) \\
&= \mathbf{c}(n) + \mu\mathbf{x}(n)d^*(n) - \mu\mathbf{x}(n)\mathbf{x}^H(n)[\mathbf{w}(n) - \mathbf{w}_0 + \mathbf{w}_0] \\
&= \mathbf{c}(n) + \mu\mathbf{x}(n)d^*(n) - \mu\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{c}(n) \\
&\quad -\mu\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{w}_0 \\
&= [\mathbf{I} - \mu\mathbf{x}(n)\mathbf{x}^H(n)]\mathbf{c}(n) + \mu\mathbf{x}(n)[d^*(n) - \mathbf{x}^H(n)\mathbf{w}_0] \\
&= [\mathbf{I} - \mu\mathbf{x}(n)\mathbf{x}^H(n)]\mathbf{c}(n) + \mu\mathbf{x}(n)e_0^*(n)
\end{aligned}$$

Take the expectation of the update noting that

- $\mathbf{w}(n)$ is based on past inputs and desired values

$\Rightarrow$ $\mathbf{w}(n)$, and consequently $\mathbf{c}(n)$), are independent of $\mathbf{x}(n)$ (Independence Theorem)

Thus

$$\begin{aligned}
\mathbf{c}(n+1) &= [\mathbf{I}-\mu\mathbf{x}(n)\mathbf{x}^H(n)]\mathbf{c}(n)+\mu\mathbf{x}(n)e_0^*(n) \\
\Rightarrow E\{\mathbf{c}(n+1)\} &= (\mathbf{I}-\mu\mathbf{R})E\{\mathbf{c}(n)\}+\mu\underbrace{E\{\mathbf{x}(n)e_0^*(n)\}}_{=0 \text{ why?}} \\
&= (\mathbf{I}-\mu\mathbf{R})E\{\mathbf{c}(n)\}
\end{aligned}$$

Using arguments similar to the SD case we have

$$\lim_{n\to\infty} E\{\mathbf{c}(n)\} = 0 \quad \text{if} \quad 0<\mu<\frac{2}{\lambda_{\max}}$$

or equivalently

$$\lim_{n\to\infty} E\{\mathbf{w}(n)\} = \mathbf{w}_0 \quad \text{if} \quad 0<\mu<\frac{2}{\lambda_{\max}}$$

Noting that $\sum_{i=1}^{M} \lambda_i = \text{trace}[\mathbf{R}]$

$$\Rightarrow \lambda_{\max} \leq \text{trace}[\mathbf{R}] = Mr(0) = M\sigma_x^2$$

Thus a more conservative bound (and one easier to determine) is

$$0 < \mu < \frac{2}{M\sigma_x^2}$$

- Convergence in the mean

$$\lim_{n\to\infty} E\{\mathbf{w}(n)\} = \mathbf{w}_0$$

  is a weak condition that says nothing about the variance, which may even grow
- A stronger condition is convergence in the mean square, which says

$$\lim_{n\to\infty} E\{|\mathbf{c}(n)|^2\} = \text{constant}$$

Proving convergence in the mean square is equivalent to showing that

$$\lim_{n\to\infty} J(n) = \lim_{n\to\infty} E\{|e(n)|^2\} = \text{constant}$$

To evaluate the limit, write $e(n)$ as

$$\begin{aligned} e(n) &= d(n) - \hat{d}(n) = d(n) - \mathbf{w}^H(n)\mathbf{x}(n) \\ &= d(n) - \mathbf{w}_0^H\mathbf{x}(n) - [\mathbf{w}^H(n) - \mathbf{w}_0^H]\mathbf{x}(n) \\ &= e_0(n) - \mathbf{c}^H(n)\mathbf{x}(n) \end{aligned}$$

Thus

$$\begin{aligned} J(n) &= E\{|e(n)|^2\} \\ &= E\left\{\left(e_0(n) - \mathbf{c}^H(n)\mathbf{x}(n)\right)\left(e_0^*(n) - \mathbf{x}^H(n)\mathbf{c}(n)\right)\right\} \\ &= J_{\min} + \underbrace{E\{\mathbf{c}^H(n)\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{c}(n)\}}_{J_{\text{ex}}(n)} \quad \text{[Cross terms} \to 0\text{, why?]} \\ &= J_{\min} + J_{\text{ex}}(n) \end{aligned}$$

Consider again

$$J_{\text{ex}}(n) = \sum_{i=1}^{M} \lambda_i s_i(n)$$

Taking the limit and utilizing $s_i(n) = \frac{\mu J_{\text{min}}}{2-\mu\lambda_i}$,

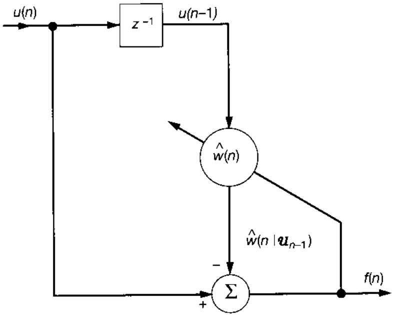
$$\lim_{n\to\infty} J_{\text{ex}}(n) = J_{\text{min}} \sum_{i=1}^{M} \frac{\mu\lambda_i}{2-\mu\lambda_i}$$

The LMS misadjustment is defined as

$$MA = \frac{\lim_{n\to\infty} J_{\text{ex}}(n)}{J_{\text{min}}} = \sum_{i=1}^{M} \frac{\mu\lambda_i}{2-\mu\lambda_i}$$

Note: A misadjustment at 10% or less is generally considered acceptable.

Example



- This is a one tap predictor

$$\hat{x}(n) = w(n)x(n-1)$$

- Take the underlying process to be a real order one AR process

$$x(n) = -ax(n-1) + v(n)$$

The weight update is

$$\begin{aligned} w(n+1) &= w(n) + \mu x(n-1)e(n) \quad \text{[LMS update for obs. } x(n-1)\text{]} \\ &= w(n) + \mu x(n-1)[x(n) - w(n)x(n-1)] \end{aligned}$$

Since

$$x(n) = -ax(n-1) + v(n) \qquad \text{[AR model]}$$

and

$$\begin{aligned} \hat{x}(n) &= w(n)x(n-1) \qquad \text{[one tap predictor]} \\ \Rightarrow w_0 &= -a \end{aligned}$$

Note that

$$E\{x(n-1)e_o(n)\} = E\{x(n-1)v(n)\} = 0$$

proves the optimality

- Set $\mu = 0.05$ and consider two cases

a	σ_x^2
-0.99	0.93627
0.99	0.995

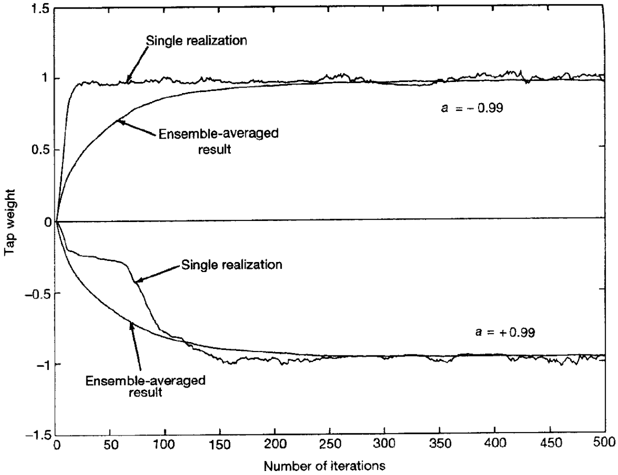


Figure: Transient behavior of adaptive first-order predictor weight $\hat{w}(n)$ for $\mu = 0.05$.

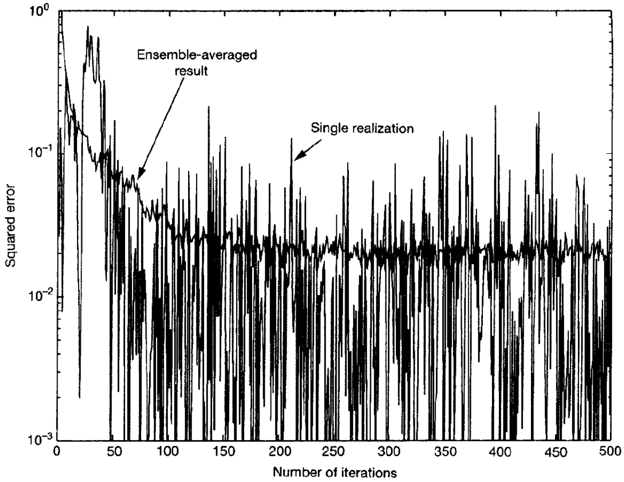


Figure: Transient behavior of adaptive first-order predictor squared error for $\mu = 0.05$.

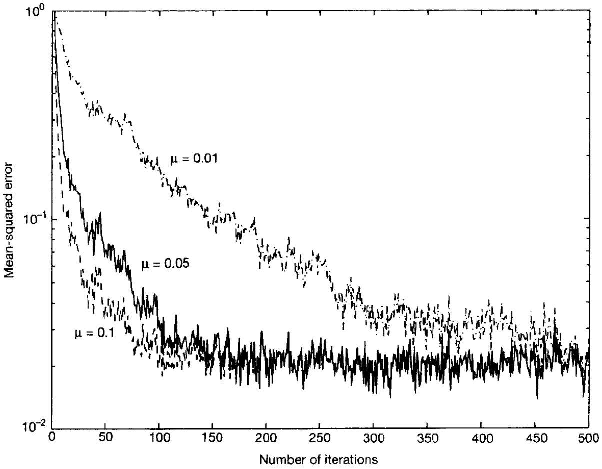


Figure: Mean-squared error learning curves for an adaptive first-order predictor with varying step-size parameter μ.

Consider the expected trajectory of $w(n)$. Recall

$$\begin{aligned} w(n+1) &= w(n)+\mu x(n-1)e(n) \\ &= w(n)+\mu x(n-1)[x(n)-w(n)x(n-1)] \\ &= [1-\mu x(n-1)x(n-1)]w(n)+\mu x(n-1)x(n) \end{aligned}$$

In this example, $x(n)=-ax(n-1)+v(n)$. Substituting in:

$$\begin{aligned} w(n+1) &= [1-\mu x(n-1)x(n-1)]w(n)+\mu x(n-1)[-ax(n-1) \\ &\quad +v(n)] \\ &= [1-\mu x(n-1)x(n-1)]w(n)-\mu a x(n-1)x(n-1) \\ &\quad +\mu x(n-1)v(n) \end{aligned}$$

Taking the expectation and invoking the dependence theorem

$$E\{w(n+1)\}=(1-\mu\sigma_x^2)E\{w(n)\}-\mu\sigma_x^2 a$$

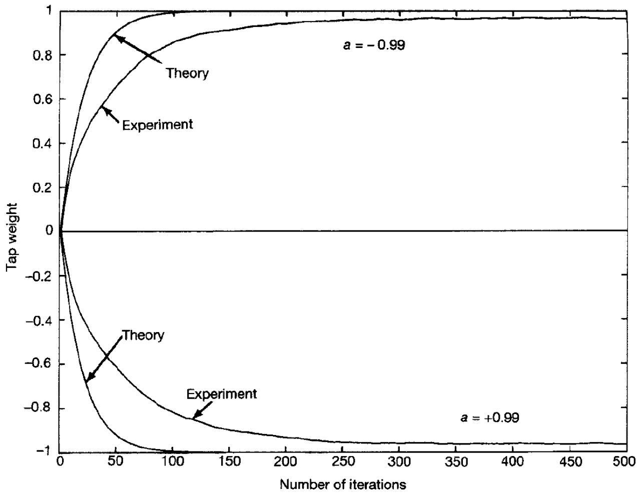


Figure: Comparison of experimental results with theory, based on $\hat{w}(n)$.

Next, derive a theoretical expression for $J(n)$.

Note that the initial value of $J(n)$ is

$$J(0) = E\{(x(0) - w(0)x(-1))^2\} = E\{(x(0))^2\} = \sigma_x^2$$

and the final value is

$$\begin{aligned} J(\infty) &= J_{\text{min}} + J_{\text{ex}} \\ &= E\{(x(n) - w(n)x(n-1))^2\} + J_{\text{ex}} \\ &= E\{(v(n))^2\} + J_{\text{ex}} \\ &= \sigma_v^2 + J_{\text{min}} \frac{\mu\lambda_1}{2 - \mu\lambda_1} \end{aligned}$$

Note $\lambda_1 = \sigma_x^2$. Thus,

$$\begin{aligned} J(\infty) &= \sigma_v^2 + \sigma_v^2 \left(\frac{\mu\sigma_x^2}{2 - \mu\sigma_x^2} \right) \\ &= \sigma_v^2 \left(1 + \frac{\mu\sigma_x^2}{2 - \mu\sigma_x^2} \right) \end{aligned}$$

And if μ is small

$$
\begin{aligned}
J(\infty) &= \sigma_v^2\left(1+\frac{\mu\sigma_x^2}{2-\mu\sigma_x^2}\right)\\
&\approx \sigma_v^2\left(1+\frac{\mu\sigma_x^2}{2}\right)
\end{aligned}
$$

Putting all the components together:

$$J(n) = \underbrace{[\sigma_x^2 - \sigma_v^2(1+\frac{\mu}{2}\sigma_x^2)]}_{J(0)-J(\infty)}\underbrace{(1-\mu\sigma_x^2)^{2n}}_{1\to 0} + \underbrace{\sigma_v^2(1+\frac{\mu}{2}\sigma_x^2)}_{J(\infty)}$$

Also, the time constant is

$$\tau = -\frac{1}{2\ln(1-\mu\lambda_1)} = -\frac{1}{2\ln(1-\mu\sigma_x^2)} \approx \frac{1}{2\mu\sigma_x^2}$$

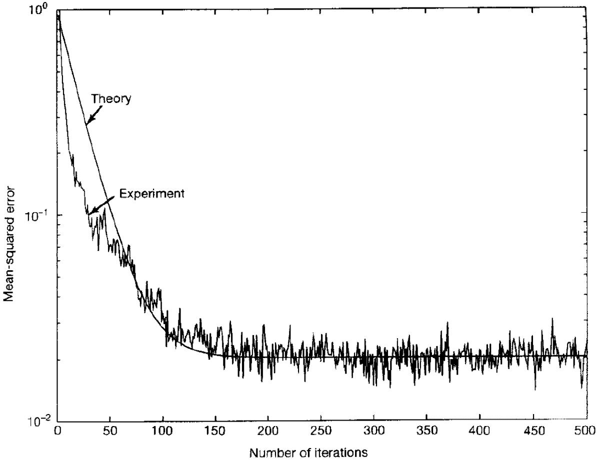
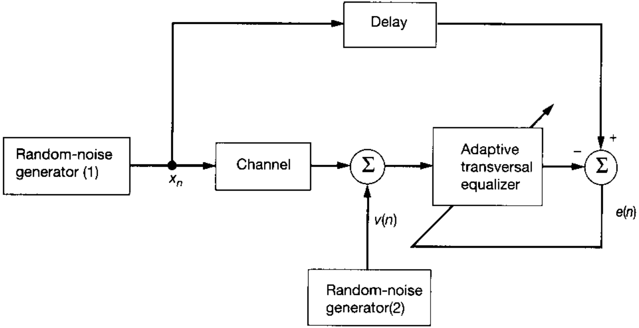


Figure: Comparison of experimental results with theory for the adaptive predictor, based on the mean-square error for $\mu = 0.001$.

Example (Adaptive Equalization)

Objective: Pass a known signal through an unknown channel to invert the effects the channel and noise have on the signal



- The signal is a Bernoulli sequence

$$x_n = \begin{cases} +1 & \text{with probability } 1/2 \\ -1 & \text{with probability } 1/2 \end{cases}$$

- The additive noise is $\sim N(0, 0.001)$
- The channel has a raised cosine response

$$h_n = \begin{cases} \frac{1}{2}\left[1+\cos\left(\frac{2\pi}{w}(n-2)\right)\right] & n = 1,2,3 \\ 0 & \text{otherwise} \end{cases}$$

$\Rightarrow$ w controls the eigenvalue spread $\chi(\mathbf{R})$

$\Rightarrow$ h_n is symmetric about $n = 2$ and thus introduces a delay of 2

- We will use an $M = 11$ tap filter, which is symmetric about $n = 5$

$\Rightarrow$ Introduce a delay of 5

- Thus an overall delay of $\delta = 5 + 2 = 7$ is added to the system

Channel response and Filter response

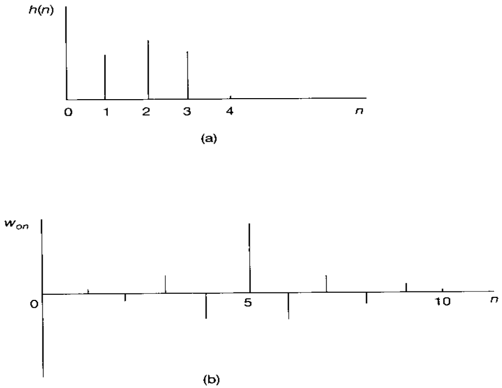


Figure: (a) Impulse response of channel; (b) impulse response of optimum transversal equalizer.

Consider three w values

TABLE 9.1 SUMMARY OF PARAMETERS FOR THE EXPERIMENT ON ADAPTIVE EQUALIZATION

W	2.9	3.1	3.3	3.5
$r(0)$	1.0963	1.1568	1.2264	1.3022
$r(1)$	0.4388	0.5596	0.6729	0.7774
$r(2)$	0.0481	0.0783	0.1132	0.1511
$\lambda_{\min}$	0.3339	0.2136	0.1256	0.0656
$\lambda_{\max}$	2.0295	2.3761	2.7263	3.0707
$\chi(\mathbf{R}) = \lambda_{\max}/\lambda_{\min}$	6.0782	11.1238	21.7132	46.8216

Note the step size is bound by the $w = 3.5$ case

$$\mu \leq \frac{2}{Mr(0)} = \frac{2}{11(1.3022)} = 0.14$$

Choose $\mu = 0.075$ in all cases.

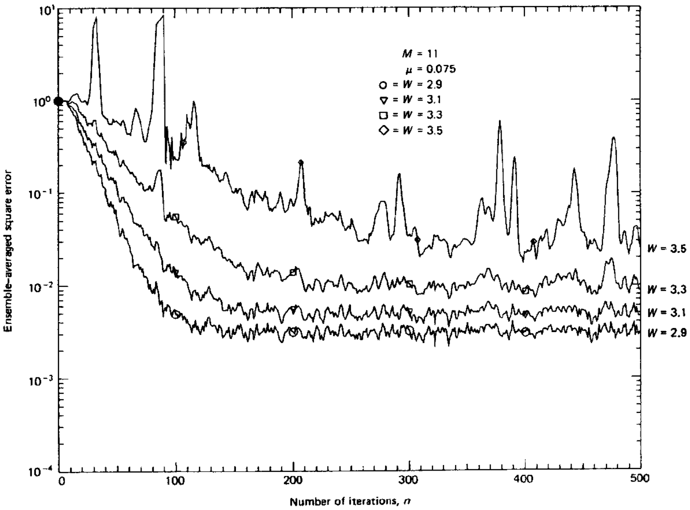
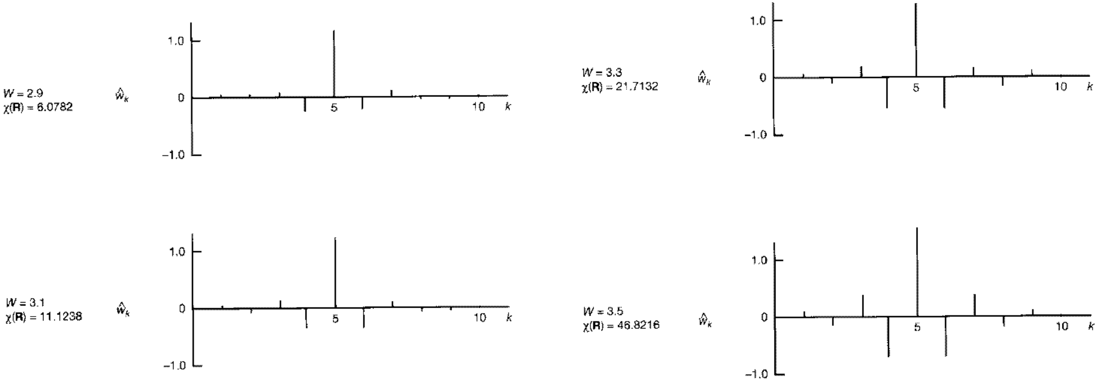


Figure: Learning curves of the LMS algorithm for an adaptive equalizer with number of taps $M = 11$, step-size parameter $\mu = 0.075$, and varying eigenvalue spread $\chi(\mathbf{R})$.

Ensemble-average impulse response of the adaptive equalizer (after 1000 iterations) for each of four different eigenvalue spreads.



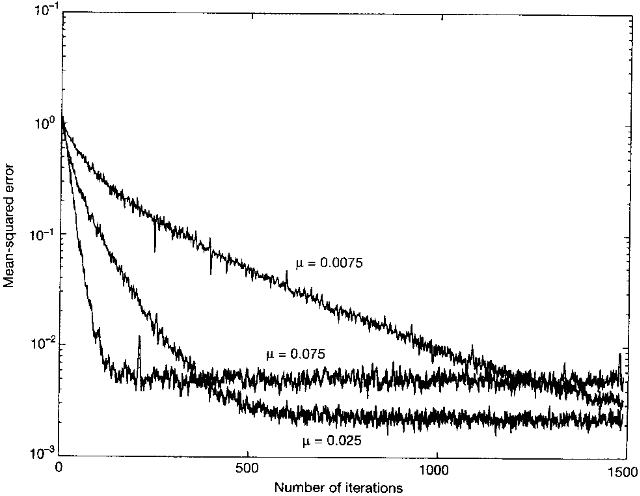


Figure: Learning curves of the LMS algorithm for an adaptive equalizer with the number of taps $M = 11$, fixed eigenvalue spread, and varying step-size parameter μ.

Normalized LMS Algorithm

Observation: The LMS correction is proportional to $\mu\mathbf{x}(n)e^*(n)$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\mathbf{x}(n)e^*(n)$$

- If $\mathbf{x}(n)$ is large, the LMS update suffers from gradient noise amplification
- The normalized LMS algorithm seeks to avoid gradient noise amplification
  - The step size is made time varying, $\mu(n)$, and optimized to minimize the next step error

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) + \frac{1}{2}\mu(n)[-\nabla J(n)] \\ &= \mathbf{w}(n) + \mu(n)[\mathbf{p} - \mathbf{R}\mathbf{w}(n)]\end{aligned}$$

Choose $\mu(n)$, such that $\mathbf{w}(n+1)$ produces the minimum MSE,

$$J(n+1) = E\{|e(n+1)|^2\}$$

Let $\nabla(n) \equiv \nabla J(n)$ and note

$$e(n+1) = d(n+1) - \mathbf{w}^H(n+1)\mathbf{x}(n+1)$$

Objective: Choose $\mu(n)$ such that it minimizes $J(n+1)$

- The optimal step size, $\mu_0(n)$, will be a function of $\mathbf{R}$ and $\nabla(n)$.
  - $\Rightarrow$ Use instantaneous estimates of these values
- To determine $\mu_0(n)$, expand $J(n+1)$

$$\begin{aligned} J(n+1) &= E\{e(n+1)e^*(n+1)\} \\ &= E\{(d(n+1) - \mathbf{w}^H(n+1)\mathbf{x}(n+1)) \\ &\quad (d^*(n+1) - \mathbf{x}^H(n+1)\mathbf{w}(n+1))\} \\ &= \sigma_d^2 - \mathbf{w}^H(n+1)\mathbf{p} - \mathbf{p}^H\mathbf{w}(n+1) \\ &\quad + \mathbf{w}^H(n+1)\mathbf{R}\mathbf{w}(n+1) \end{aligned}$$

Now use the fact that $\mathbf{w}(n+1) = \mathbf{w}(n) - \frac{1}{2}\mu(n)\nabla(n)$

$$
\begin{aligned}
J(n+1) &= \sigma_d^2 - \mathbf{w}^H(n+1)\mathbf{p} - \mathbf{p}^H\mathbf{w}(n+1) \\
&\quad + \mathbf{w}^H(n+1)\mathbf{R}\mathbf{w}(n+1) \\
&= \sigma_d^2 - \left[\mathbf{w}(n) - \frac{1}{2}\mu(n)\nabla(n)\right]^H \mathbf{p} \\
&\quad - \mathbf{p}^H\left[\mathbf{w}(n) - \frac{1}{2}\mu(n)\nabla(n)\right] \\
&\quad + \underbrace{\left[\mathbf{w}(n) - \frac{1}{2}\mu(n)\nabla(n)\right]^H \mathbf{R}\left[\mathbf{w}(n) - \frac{1}{2}\mu(n)\nabla(n)\right]}
\end{aligned}
$$

$$
\begin{aligned}
&= \mathbf{w}^H(n)\mathbf{R}\mathbf{w}(n) - \frac{1}{2}\mu(n)\mathbf{w}^H(n)\mathbf{R}\nabla(n) \\
&\quad - \frac{1}{2}\mu(n)\nabla^H(n)\mathbf{R}\mathbf{w}(n) + \frac{1}{4}\mu^2(n)\nabla^H(n)\mathbf{R}\nabla(n)
\end{aligned}
$$

$$
\begin{aligned}
J(n+1) &= \sigma_d^2 - \left[\mathbf{w}(n) - \frac{1}{2}\mu(n)\nabla(n)\right]^H \mathbf{p} \\
&\quad -\mathbf{p}^H \left[\mathbf{w}(n) - \frac{1}{2}\mu(n)\nabla(n)\right] \\
&\quad +\mathbf{w}^H(n)\mathbf{R}\mathbf{w}(n) - \frac{1}{2}\mu(n)\mathbf{w}^H(n)\mathbf{R}\nabla(n) \\
&\quad -\frac{1}{2}\mu(n)\nabla^H(n)\mathbf{R}\mathbf{w}(n) + \frac{1}{4}\mu^2(n)\nabla^H(n)\mathbf{R}\nabla(n)
\end{aligned}
$$

Differentiating with respect to $\mu(n)$,

$$
\begin{aligned}
\frac{\partial J(n+1)}{\partial \mu(n)} &= \frac{1}{2}\nabla^H(n)\mathbf{p} + \frac{1}{2}\mathbf{p}^H\nabla(n) - \frac{1}{2}\mathbf{w}^H\mathbf{R}\nabla(n) \\
&\quad -\frac{1}{2}\nabla^H(n)\mathbf{R}\mathbf{w}(n) + \frac{1}{2}\mu(n)\nabla^H(n)\mathbf{R}\nabla(n) \qquad (*)
\end{aligned}
$$

Setting $(*)$ equal to 0

$$\begin{aligned}\mu_0(n)\nabla^H(n)\mathbf{R}\nabla(n) &= \mathbf{w}^H(n)\mathbf{R}\nabla(n)-\mathbf{p}^H\nabla(n)\\&\quad+\nabla^H(n)\mathbf{R}\mathbf{w}(n)-\nabla^H(n)\mathbf{p}\end{aligned}$$

$$\begin{aligned}\Rightarrow \mu_0(n) &= \frac{\mathbf{w}^H(n)\mathbf{R}\nabla(n)-\mathbf{p}^H\nabla(n)+\nabla^H(n)\mathbf{R}\mathbf{w}(n)-\nabla^H(n)\mathbf{p}}{\nabla^H(n)\mathbf{R}\nabla(n)}\\
&= \frac{[\mathbf{w}^H(n)\mathbf{R}-\mathbf{p}^H]\nabla(n)+\nabla^H(n)[\mathbf{R}\mathbf{w}(n)-\mathbf{p}]}{\nabla^H(n)\mathbf{R}\nabla(n)}\\
&= \frac{[\mathbf{R}\mathbf{w}(n)-\mathbf{p}]^H\nabla(n)+\nabla^H(n)[\mathbf{R}\mathbf{w}(n)-\mathbf{p}]}{\nabla^H(n)\mathbf{R}\nabla(n)}\\
&= \frac{\frac{1}{2}\nabla^H(n)\nabla(n)+\frac{1}{2}\nabla^H(n)\nabla(n)}{\nabla^H(n)\mathbf{R}\nabla(n)}\\
&= \frac{\nabla^H(n)\nabla(n)}{\nabla^H(n)\mathbf{R}\nabla(n)}\end{aligned}$$

Using instantaneous estimates

$$\begin{aligned}
\hat{\mathbf{R}} &= \mathbf{x}(n)\mathbf{x}^H(n) \quad \text{and} \quad \hat{\mathbf{p}} = \mathbf{x}(n)d^*(n) \\
\Rightarrow \hat{\nabla}(n) &= 2[\hat{\mathbf{R}}\mathbf{w}(n) - \hat{\mathbf{p}}] \\
&= 2[\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{w}(n) - \mathbf{x}(n)d^*(n)] \\
&= 2[\mathbf{x}(n)(\hat{d}^*(n) - d^*(n))] \\
&= -2\mathbf{x}(n)e^*(n)
\end{aligned}$$

Thus

$$\begin{aligned}
\mu_0(n) &= \frac{\nabla^H(n)\nabla(n)}{\nabla^H(n)\mathbf{R}\nabla(n)} = \frac{4\mathbf{x}^H(n)e(n)\mathbf{x}(n)e^*(n)}{2\mathbf{x}^H(n)e(n)\mathbf{x}(n)\mathbf{x}^H(n)2\mathbf{x}(n)e^*(n)} \\
&= \frac{|e(n)|^2\mathbf{x}^H(n)\mathbf{x}(n)}{|e(n)|^2(\mathbf{x}^H(n)\mathbf{x}(n))^2} \\
&= \frac{1}{\mathbf{x}^H(n)\mathbf{x}(n)} = \frac{1}{||\mathbf{x}(n)||^2}
\end{aligned}$$

Result: The NLMS update is

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \underbrace{\frac{\tilde{\mu}}{||\mathbf{x}(n)||^2}}_{\mu(n)} \mathbf{x}(n)e^*(n)$$

- $\tilde{\mu}$ is introduced to scale the update
- To avoid problems when $||\mathbf{x}(n)||^2 \approx 0$ we add an offset

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\tilde{\mu}}{a + ||\mathbf{x}(n)||^2} \mathbf{x}(n)e^*(n)$$

where $a > 0$

Objective: Analyze the NLMS convergence

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\tilde{\mu}}{||\mathbf{x}(n)||^2}\mathbf{x}(n)e^*(n)$$

Substituting $e(n) = d(n) - \mathbf{w}^H(n)\mathbf{x}(n)$

$$\begin{aligned} \mathbf{w}(n+1) &= \mathbf{w}(n) + \frac{\tilde{\mu}}{||\mathbf{x}(n)||^2}\mathbf{x}(n)[d^*(n) - \mathbf{x}^H(n)\mathbf{w}(n)] \\ &= \left[\mathbf{I} - \tilde{\mu}\frac{\mathbf{x}(n)\mathbf{x}^H(n)}{||\mathbf{x}(n)||^2}\right]\mathbf{w}(n) + \tilde{\mu}\frac{\mathbf{x}(n)d^*(n)}{||\mathbf{x}(n)||^2} \end{aligned}$$

Objective: Compare the NLMS and LMS algorithms:

- NLMS:

$$\mathbf{w}(n+1) = \left[\mathbf{I} - \tilde{\mu}\frac{\mathbf{x}(n)\mathbf{x}^H(n)}{||\mathbf{x}(n)||^2}\right]\mathbf{w}(n) + \tilde{\mu}\frac{\mathbf{x}(n)d^*(n)}{||\mathbf{x}(n)||^2}$$

- LMS:

$$\mathbf{w}(n+1) = [\mathbf{I} - \mu\mathbf{x}(n)\mathbf{x}^H(n)]\mathbf{w}(n) + \mu\mathbf{x}(n)d^*(n)$$

By observation, we see the following corresponding terms

LMS	NLMS
μ	$\tilde{\mu}$
$\mathbf{x}(n)\mathbf{x}^H(n)$	$\frac{\mathbf{x}(n)\mathbf{x}^H(n)}{\vert\vert\mathbf{x}(n)\vert\vert^2}$
$\mathbf{x}(n)d^*(n)$	$\frac{\mathbf{x}(n)d^*(n)}{\vert\vert\mathbf{x}(n)\vert\vert^2}$

LMS	NLMS
μ	$\tilde{\mu}$
$\mathbf{x}(n)\mathbf{x}^H(n)$	$\frac{\mathbf{x}(n)\mathbf{x}^H(n)}{\|\|\mathbf{x}(n)\|\|^2}$
$\mathbf{x}(n)d^*(n)$	$\frac{\mathbf{x}(n)d^*(n)}{\|\|\mathbf{x}(n)\|\|^2}$

- LMS case:

$$0 < \mu < \frac{2}{\text{trace}[E\{\mathbf{x}(n)\mathbf{x}^H(n)\}]} = \frac{2}{\text{trace}[\mathbf{R}]}$$

  guarantees stability
- By analogy,

$$0 < \tilde{\mu} < \frac{2}{\text{trace}\left[E\left\{\frac{\mathbf{x}(n)\mathbf{x}^H(n)}{||\mathbf{x}(n)||^2}\right\}\right]}$$

  guarantees stability of the NLMS

To analyze the bound, make the following approximation

$$E\left\{\frac{\mathbf{x}(n)\mathbf{x}^H(n)}{||\mathbf{x}(n)||^2}\right\} \approx \frac{E\{\mathbf{x}(n)\mathbf{x}^H(n)\}}{E\{||\mathbf{x}(n)||^2\}}$$

Then

$$\begin{aligned}
\text{trace}\left[E\left\{\frac{\mathbf{x}(n)\mathbf{x}^H(n)}{||\mathbf{x}(n)||^2}\right\}\right] &= \frac{\text{trace}[E\{\mathbf{x}(n)\mathbf{x}^H(n)\}]}{E\{||\mathbf{x}(n)||^2\}} \\
&= \frac{E\{\text{trace}[\mathbf{x}(n)\mathbf{x}^H(n)]\}}{E\{||\mathbf{x}(n)||^2\}} \\
&= \frac{E\{\text{trace}[\mathbf{x}^H(n)\mathbf{x}(n)]\}}{E\{||\mathbf{x}(n)||^2\}} \\
&= \frac{E\{\text{trace}[||\mathbf{x}(n)||^2]\}}{E\{||\mathbf{x}(n)||^2\}} \\
&= 1
\end{aligned}$$

Thus

$$0 < \tilde{\mu} < \frac{2}{\text{trace}\left[E\left\{\frac{\mathbf{x}(n)\mathbf{x}^H(n)}{||\mathbf{x}(n)||^2}\right\}\right]} = 2$$

Final Result: The NLMS update

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \tilde{\mu}\frac{\mathbf{x}(n)}{||\mathbf{x}(n)||^2}e^*(n)$$

will converge if $0 < \tilde{\mu} < 2$

Note:

- The NLMS has a simpler convergence criterion than the LMS
- The NLMS generally converges faster than the LMS algorithm