SGN 21006 Advanced Signal Processing: Lecture 5 Stochastic gradient based adaptation: Least Mean Square (LMS) Algorithm

Ioan Tabus

Department of Signal Processing
Tampere University of Technology
Finland

LMS algorithm derivation based on the Steepest descent (SD) algorithm

Steepest descent search algorithm (from last lecture)

Given
- the autocorrelation matrix $R = E\underline{u}(n)\underline{u}^T(n)$
- the cross-correlation vector $\underline{p}(n) = E\underline{u}(n)d(n)$

Initialize the algorithm with an arbitrary parameter vector $\underline{w}(0)$.
Iterate for $n = 0, 1, 2, 3, \ldots, n_{max}$

$$\underline{w}(n+1) = \underline{w}(n) + \mu[\underline{p} - R\underline{w}(n)] \qquad \text{(Equation } \mathbf{SD} - \underline{p}, R\text{)}$$

We have shown that adaptation equation ($\mathbf{SD} - \underline{p}, R$) can be written in an equivalent form as (see also the Figure with the implementation of SD algorithm)

$$\underline{w}(n+1) = \underline{w}(n) + \mu[Ee(n)\underline{u}(n)] \qquad \text{(Equation } \mathbf{SD} - \underline{u}, e\text{)}$$

In order to simplify the algorithm, instead the true gradient of the criterion

$$\nabla_{\underline{w}(n)} J(n) = -2E\underline{u}(n)e(n)$$

LMS algorithm will use an immediately available approximation

$$\hat{\nabla}_{\underline{w}(n)} J(n) = -2\underline{u}(n)e(n)$$

LMS algorithm

Using **the noisy gradient**, the adaptation will carry on the equation

$$\underline{w}(n+1) = \underline{w}(n) - \frac{1}{2}\mu \hat{\nabla}_{\underline{w}(n)} J(n) = \underline{w}(n) + \mu \underline{u}(n) e(n)$$

In order to gain new information at each time instant about the gradient estimate, the procedure will go through all data set $\{(d(1), u(1)), (d(2), u(2)), \ldots\}$, many times if needed.

LMS algorithm

Given

- the (correlated) input signal samples $\{u(1), u(2), u(3), \ldots\}$, generated randomly;
- the desired signal samples $\{d(1), d(2), d(3), \ldots\}$ correlated with $\{u(1), u(2), u(3), \ldots\}$

1 Initialize the algorithm with an arbitrary parameter vector $\underline{w}(0)$, for example $\underline{w}(0) = 0$.

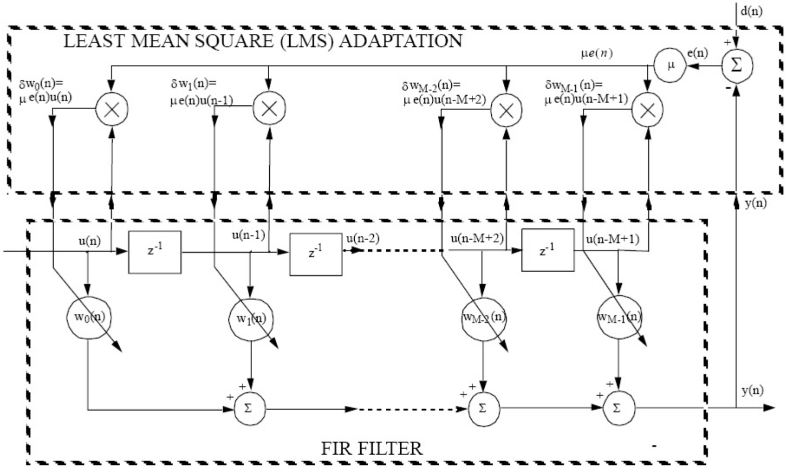
2 Iterate for $n = 0, 1, 2, 3, \ldots, n_{max}$

2.0	Read /generate a new data pair,	$(\underline{u}(n), d(n))$
2.1	(Filter output)	$y(n) = \underline{w}(n)^T \underline{u}(n) = \sum_{i=0}^{M-1} w_i(n) u(n-i)$
2.2	(Output error)	$e(n) = d(n) - y(n)$
2.3	(Parameter adaptation)	$\underline{w}(n+1) = \underline{w}(n) + \mu \underline{u}(n) e(n)$

$$\begin{bmatrix} w_0(n+1) \\ w_1(n+1) \\ \vdots \\ w_{M-1}(n+1) \end{bmatrix} = \begin{bmatrix} w_0(n) \\ w_1(n) \\ \vdots \\ w_{M-1}(n) \end{bmatrix} + \mu e(n) \begin{bmatrix} u(n) \\ u(n-1) \\ \vdots \\ u(n-M+1) \end{bmatrix}$$

□

Schematic view of LMS algorithm



Stability analysis of LMS algorithm

SD algorithm is guaranteed to converge to Wiener optimal filter if the value of μ is selected properly (see last Lecture)

$$\underline{w}(n) \to \underline{w}_o$$

$$J(\underline{w}(n)) \to J(\underline{w}_o)$$

The iterations are deterministic : starting from a given $\underline{w}(0)$, all the iterations $\underline{w}(n)$ are perfectly determined.
LMS iterations are not deterministic: the values $w(n)$ depend on the realization of the data $d(1), \ldots, d(n)$ and $u(1), \ldots, u(n)$. *Thus,* $\underline{w}(n)$ *is now a random variable.*
The convergence of LMS can be analyzed from following perspectives:

- Convergence of parameters $\underline{w}(n)$ in the mean:

$$E\underline{w}(n) \to \underline{w}_o$$

- Convergence of the criterion $J(\underline{w}(n))$ (in the mean square of the error)

$$\underline{J}(\underline{w}(n)) \to \underline{J}(\underline{w}_\infty)$$

Convergence of average parameter vector $E\underline{w}(n)$

We will subtract from the adaptation equation

$$\underline{w}(n+1) = \underline{w}(n) + \mu\underline{u}(n)e(n) = \underline{w}(n) + \mu\underline{u}(n)(d(n) - \underline{w}(n)^T\underline{u}(n))$$

the vector $\underline{w}_o$ and we will denote $\underline{\varepsilon}(n) = \underline{w}(n) - \underline{w}_o$

$$\begin{aligned} \underline{w}(n+1) - \underline{w}_o &= \underline{w}(n) - \underline{w}_o + \mu\underline{u}(n)(d(n) - \underline{w}(n)^T\underline{u}(n)) \\ \underline{\varepsilon}(n+1) &= \underline{\varepsilon}(n) + \mu\underline{u}(n)(d(n) - \underline{w}_o^T\underline{u}(n)) + \mu\underline{u}(n)(\underline{u}(n)^T\underline{w}_o - \underline{u}(n)^T\underline{w}(n)) \\ &= \underline{\varepsilon}(n) + \mu\underline{u}(n)e_o(n) - \mu\underline{u}(n)\underline{u}(n)^T\underline{\varepsilon}(n) = (I - \mu\underline{u}(n)\underline{u}(n)^T)\underline{\varepsilon}(n) + \mu\underline{u}(n)e_o(n) \end{aligned}$$

Taking the expectation of $\underline{\varepsilon}(n+1)$ using the last equality we obtain

$$E\underline{\varepsilon}(n+1) = E(I - \mu\underline{u}(n)\underline{u}(n)^T)\underline{\varepsilon}(n) + E\mu\underline{u}(n)e_o(n)$$

and now assuming the statistical independence of $\underline{u}(n)$ and $\underline{\varepsilon}(n)$, we have

$$E\underline{\varepsilon}(n+1) = (I - \mu E[\underline{u}(n)\underline{u}(n)^T])E[\underline{\varepsilon}(n)] + \mu E[\underline{u}(n)e_o(n)]$$

Using the principle of orthogonality which states that $E[\underline{u}(n)e_o(n)] = 0$, the last equation becomes

$$E[\underline{\varepsilon}(n+1)] = (I - \mu E[\underline{u}(n)\underline{u}(n)^T])E[\underline{\varepsilon}(n)] = (I - \mu R)E[\underline{\varepsilon}(n)]$$

Reminding the equation

$$\underline{c}(n+1) = (I - \mu R)\underline{c}(n)$$

which was used in the analysis of SD algorithm stability, and identifying now $\underline{c}(n)$ with $E\underline{\varepsilon}(n)$, we have the stability condition on next page.

Convergence of average parameter vector $E\underline{w}(n)$

The mean $E\underline{\varepsilon}(n)$ converges to zero, and consequently $E\underline{w}(n)$ converges to $\underline{w}_o$

iff

$0 < \mu < \dfrac{2}{\lambda_{max}}$ (*STABILITYCONDITION*!) where λ_{max} is the largest eigenvalue of the matrix $R = E[\underline{u}(n)\underline{u}(n)^T]$.

Stated in words, LMS is convergent in mean, if the stability condition is met.
The convergence property explains the behavior of the first order characterization of $\underline{\varepsilon}(n) = \underline{w}(n) - \underline{w}_o$.

Convergence of the criterion $J(\underline{w}(n))$

Define the criterion J_∞ as the value of criterion $J(\underline{w}(n)) = J(n)$ when $n \to \infty$.
Denote the optimal mean square error (of the Wiener filter) J_o.
It can be shown that in the case of LMS adaptation one can obtain an approximate expression

$$J_\infty = J_o + \sum_{i=1}^{M} \lambda_i \frac{\mu J_o}{2 - \mu\lambda_i}$$

For $\mu\lambda_i \ll 2$

$$J_\infty = J_o + \mu J_o \sum_{i=1}^{M} \lambda_i/2 = J_o(1 + \mu \sum_{i=1}^{M} \lambda_i/2) = J_o(1 + \mu tr(R)/2)$$

$$\boxed{J_\infty = J_o(1 + \mu M r(0)/2) = J_o(1 + \frac{\mu M}{2} \cdot \text{Power of the input})}$$

The steady state mean square error J_∞ is close to J_o if μ is small enough.

Learning curves

The statistical performance of adaptive filters is studied using learning curves, averaged over many realizations, or ensemble-averaged.

- **The mean square error MSE learning curve** Take an ensemble average of the squared estimation error $e(n)^2$

$$J(n) = Ee^2(n) \tag{1}$$

- **The mean-square deviation (MSD) learning curve** Take an ensemble average of the squared error deviation $||\underline{\varepsilon}(n)||^2$ where $\underline{\varepsilon} = \underline{w}(n) - \underline{w}_o$ is the deviation of the parameters of the adaptive filter, from the optimal parameters

$$\mathcal{D}(n) = E||\underline{\varepsilon}(n)||^2 \tag{2}$$

- **The excess mean-square-error**

$$J_{ex}(n) = J(n) - J_{min} \tag{3}$$

where J_{min} is the MSE error of the optimal Wiener filter.

Results of the small step size theory

The statistical performance of adaptive filters is studied using learning curves, averaged over many realizations, or ensemble-averaged.

- **Connection between MSE and MSD**

$$\lambda_{min}\mathcal{D}(n) \leq J_{ex}(n) \leq \lambda_{max}\mathcal{D}(n) \tag{4}$$

$$J_{ex}(n)/\lambda_{min} \leq \mathcal{D}(n) \leq J_{ex}(n)/\lambda_{max} \tag{5}$$

It is therefore enough to study the transient behavior of $J_{ex}(n)$, since $\mathcal{D}(n)$ follows its evolutions.

- **The condition for stability**

$$0 < \mu < \frac{2}{\lambda_{max}} \tag{6}$$

- **The excess mean-square-error** converges to

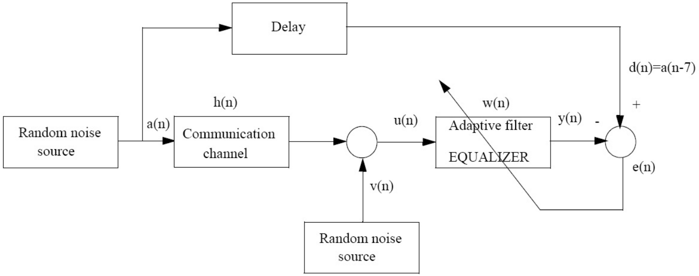
$$J_{ex}(\infty) = \frac{\mu J_{min}}{2}\sum_{k=1}^{M}\lambda_k \tag{7}$$

- **The misadjustment**

$$\mathcal{M} = \frac{J_{ex}(\infty)}{J_{min}} = \frac{\mu}{2}\sum_{k=1}^{M}\lambda_k = \frac{\mu}{2}tr(R) = \frac{\mu}{2}Mr(0) \tag{8}$$

Application of LMS algorithm : Adaptive Equalization

Block diagram of adaptive equalizer experiment



Modelling the communication channel

We assume the impulse response of the channel in the form

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

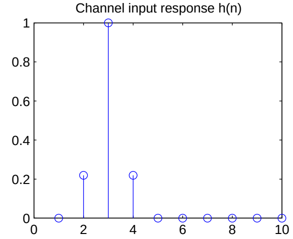
The filter input signal will be

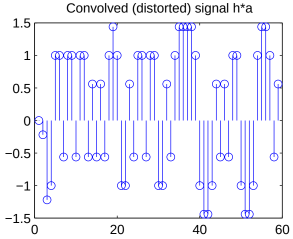
$$u(n) = (h * a)(n) = \sum_{k=1}^{3} h(k)a(n-k) + v(n)$$

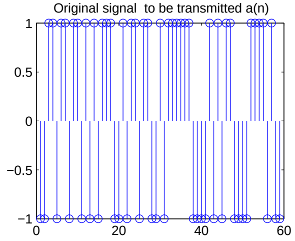
where $\sigma_v^2 = 0.001$
Selecting the filter structure
The filter has $M = 11$ delays units (taps).
The weights (parameter) of the filter are symmetric with respect to the middle tap ($n = 5$).
The channel input is delayed 7 units to provide the desired response to the equalizer.
Correlation matrix of the Equalizer input
Since $u(n) = \sum_{k=1}^{3} h(k)a(n-k) + v(n)$ is a MA process, the correlation function will be

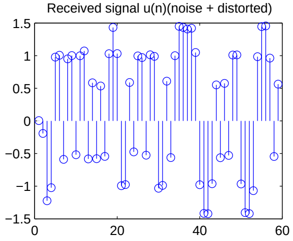
$$\begin{aligned} r(0) &= h(1)^2 + h(2)^2 + h(3)^2 + \sigma_v^2 \\ r(1) &= h(1)h(2) + h(2)h(3) \\ r(2) &= h(1)h(3) \\ r(3) = r(4) &= \ldots = 0 \end{aligned}$$

Sample signals in adaptive equalizer experiment









Effect of the parameter W on the eigenvalue spread

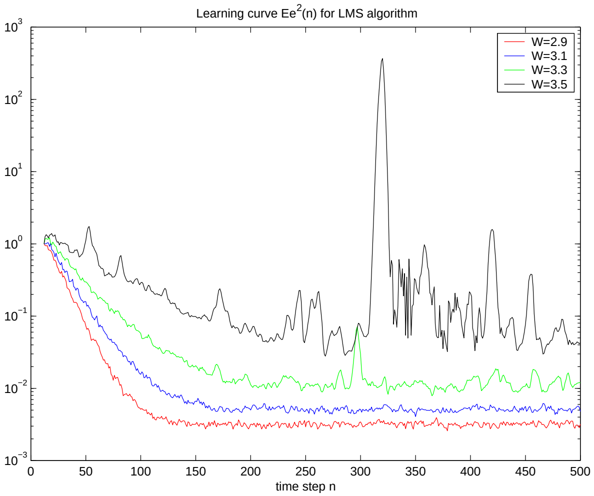
We define the eigenvalue spread $\chi(R)$ of a matrix as the ratio of the maximum eigenvalue over the minimum eigenvalue

W	2.9	3.1	3.3	3.5
$r(0)$	1.0973	1.1576	1.2274	1.3032
$r(1)$	0.4388	0.5596	0.6729	0.7775
$r(2)$	0.0481	0.0783	0.1132	0.1511
λ_{min}	0.3339	0.2136	0.1256	0.0656
λ_{max}	2.0295	2.3761	2.7263	3.0707
$\chi(R) = \lambda_{max}/\lambda_{min}$	6.0782	11.1238	21.7132	46.8216

Experiment 1: Effect of eigenvalue spread

- The step size was kept constant at $\mu = 0.075$
- The eigenvalue spread were taken [6.0782 11.1238 21.7132 46.8216] (see previous table), thus varying in a wide range
- for small eigenvalue spread, $\chi(R) = 6.07$, the convergence is the fastest, and the best steady state average squared error. The convergence time is about 80 iterations. The steady state average squared error is about 0.003.
- for small eigenvalue spread, $\chi(R) = 46.8$, the convergence is the slowest, and the worst steady state average squared error. The convergence time is about 200 iterations. The steady state average squared error is about 0.04.

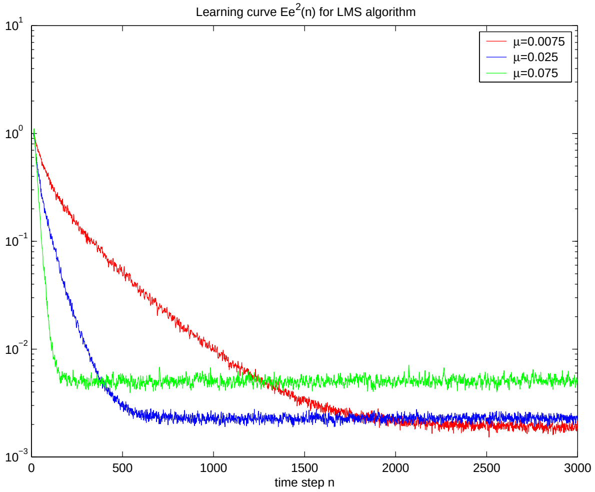
Learning curves for $\mu = 0.075$, $W = [2.9\ 3.1\ 3.3\ 3.5]$



Experiment 2: Effect of step size

- The eigenvalue spread was kept constant at $\chi = 11.12$
- The step size were taken [0.0075 0.025 0.075], thus varying in a range 1:10
- for smallest step sizes, $\mu = 0.0075$, the convergence is the slowest, and the best steady state average squared error. The convergence time is about 2300 iterations. The steady state average squared error is about 0.001.
- for large step size, $\mu = 0.075$, the convergence is the fastest, and the worst steady state average squared error. The convergence time is about 100 iterations. The steady state average squared error is about 0.005.

Learning curves for $\mu = [0.0075\ 0.025\ 0.075]$; $W = 3.1$



Summary

- LMS is simple to implement.
- LMS does not require preliminary modelling.
- Main disadvantage: slow rate of convergence.
- Convergence speed is affected by two factors: the step size and the eigenvalue spread of the correlation matrix.
- **The condition for stability** is

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

- For LMS filters with filter length M moderate to large, the convergence condition on the step size is

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

  where S_{max} is the maximum value of the power spectral density of the tap inputs.

Normalized LMS Algorithm

Modify at time n the parameter vector from $\underline{w}(n)$ to $\underline{w}(n+1)$

- fulfilling the constraint
$$\underline{w}^T(n+1)\underline{u}(n) = d(n)$$
- with the "least modification" of $\underline{w}(n)$, i.e. with the least Euclidian norm of the difference
$$\underline{w}(n+1) - \underline{w}(n) = \delta\underline{w}(n+1)$$

Thus we have to minimize

$$\|\delta\underline{w}(n+1)\|^2 = \sum_{k=0}^{M-1}(w_k(n+1) - w_k(n))^2$$

under the constraint

$$\underline{w}^T(n+1)\underline{u}(n) = d(n)$$

The solution can be obtained by Lagrange multipliers method:

$$\begin{aligned} J(\underline{w}(n+1), \lambda) &= \|\delta\underline{w}(n+1)\|^2 + \lambda\left(d(n) - \underline{w}^T(n+1)\underline{u}(n)\right) \\ &= \sum_{k=0}^{M-1}(w_k(n+1) - w_k(n))^2 + \lambda\left(d(n) - \sum_{i=0}^{M-1} w_i(n+1)u(n-i)\right) \end{aligned}$$

To obtain the minimum of $J(\underline{w}(n+1), \lambda)$ we equate to zero the partial derivatives $\frac{\partial J(\underline{w}(n+1), \lambda)}{\partial w_j(n+1)} = 0$:

$$\begin{aligned} \frac{\partial}{\partial w_j(n+1)}\left[\sum_{k=0}^{M-1}(w_k(n+1) - w_k(n))^2 + \lambda\left(d(n) - \sum_{i=0}^{M-1} w_i(n+1)u(n-i)\right)\right] &= 0 \\ 2(w_j(n+1) - w_j(n)) - \lambda u(n-j) &= 0 \end{aligned}$$

Normalized LMS Algorithm

From

$$2(w_j(n+1) - w_j(n)) - \lambda u(n-j) \quad = \quad 0$$

we have

$$w_j(n+1) = w_j(n) + \frac{1}{2}\lambda u(n-j)$$

where λ will result from

$$\begin{aligned}
d(n) &= \sum_{i=0}^{M-1} w_i(n+1)u(n-i) \\
d(n) &= \sum_{i=0}^{M-1} (w_i(n) + \frac{1}{2}\lambda u(n-i))u(n-i) \\
d(n) &= \sum_{i=0}^{M-1} w_i(n)u(n-i) + \frac{1}{2}\lambda \sum_{i=0}^{M-1} (u(n-i))^2 \\
\lambda &= \frac{2(d(n) - \sum_{i=0}^{M-1} w_i(n)u(n-i))}{\sum_{i=0}^{M-1}(u(n-i))^2} \\
\lambda &= \frac{2e(n)}{\sum_{i=0}^{M-1}(u(n-i))^2}
\end{aligned}$$

Normalized LMS Algorithm

Thus, the minimum of the criterion $J(\underline{w}(n+1), \lambda)$ will be obtained using the adaptation equation

$$w_j(n+1) = w_j(n) + \frac{e(n)}{\sum_{i=0}^{M-1}(u(n-i))^2} u(n-j)$$

In order to add an extra freedom degree to the adaptation strategy, one constant, $\tilde{\mu}$, controlling the step size will be introduced:

$$w_j(n+1) = w_j(n) + \tilde{\mu}\frac{1}{\sum_{i=0}^{M-1}(u(n-i))^2} e(n)u(n-j) = w_j(n) + \frac{\tilde{\mu}}{\|\underline{u}(n)\|^2} e(n)u(n-j)$$

To overcome the possible numerical difficulties when $\|\underline{u}(n)\|$ is very close to zero, a constant $a > 0$ is used:

$$\boxed{w_j(n+1) = w_j(n) + \frac{\tilde{\mu}}{a + \|\underline{u}(n)\|^2} e(n)u(n-j)}$$

This is the updating equation used in the Normalized LMS algorithm.

Normalized LMS Algorithm

The principal characteristics of the Normalized LMS algorithm are the following:

- The adaptation constant $\tilde{\mu}$ is dimensionless, whereas in LMS, the adaptation has the dimensioning of a inverse power.
- Setting

  $$\mu(n) = \frac{\tilde{\mu}}{a + \|\underline{u}(n)\|^2}$$

  we may view Normalized LMS algorithm as a LMS algorithm with *data- dependent adaptation step size*.
- Considering the approximate expression

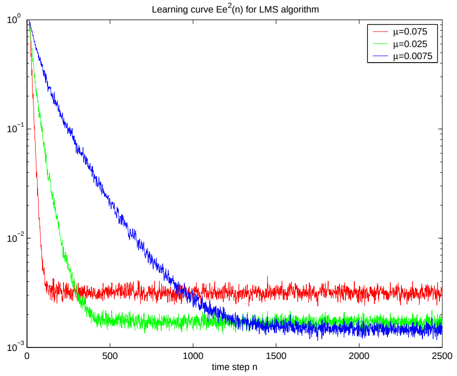
  $$\mu(n) = \frac{\tilde{\mu}}{a + ME(u(n))^2}$$

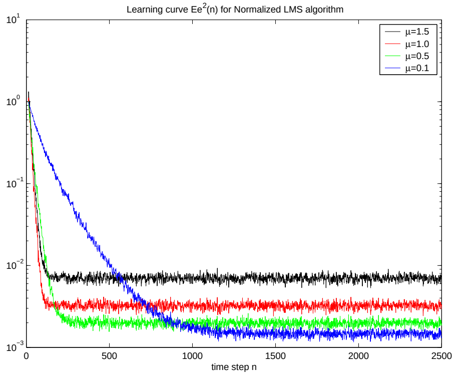
  the normalization is such that :
  * the effect of large fluctuations in the power levels of the input signal is compensated at the adaptation level.
  * the effect of large input vector length is compensated, by reducing the step size of the algorithm.
- This algorithm was derived based on an intuitive principle:

  In the light of new input data, the parameters of an adaptive system should only be disturbed in a minimal fashion.
- The Normalized LMS algorithm is convergent in mean square sense if

  $$0 < \tilde{\mu} < 2$$

Comparison of LMS and NLMS in the channel equalization example





Comparison of LMS and NLMS in the channel equalization example

- The LMS was run with three different step-sizes: $\mu = [0.075; 0.025; 0.0075]$
- The NLMS was run with four different step-sizes: $\tilde{\mu} = [1.5; 1.0; 0.5; 0.1]$
  - With the step-size $\tilde{\mu} = 1.5$, NLMS behaved definitely worse than with step-size $\tilde{\mu} = 1.0$ (slower, and with a higher steady state square error). So $\tilde{\mu} = 1.5$ is further ruled out
  - Each of the three step-sizes $\tilde{\mu} = [1.0; 0.5; 0.1]$ was interesting: on one hand, the larger the step-size, the faster the convergence. But on the other hand, the smaller the step-size, the better the steady state square error. So each step-size may be a useful tradeoff between convergence speed and stationary MSE (not both can be very good simultaneously).
- LMS with $\mu = 0.0075$ and NLMS with $\tilde{\mu} = 0.1$ achieved a similar (very good) average steady state square error. However, NLMS was faster.
- LMS with $\mu = 0.075$ and NLMS with $\tilde{\mu} = 1.0$ had a similar convergence speed (very good). However, NLMS achieved a lower steady state average square error.
- To conclude: NLMS offers better trade-offs than LMS.
- The computational complexity of NLMS is slightly higher than that of LMS.

LMS Algorithm with Time Variable Adaptation Step

Heuristics of the method: We combine the benefits of two different situations:

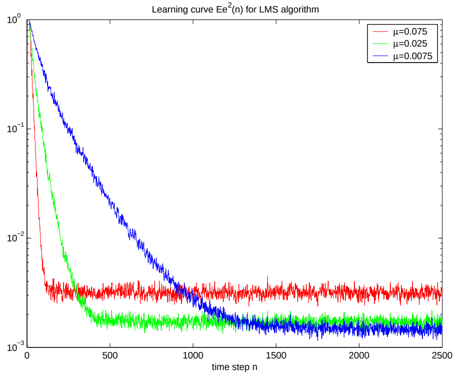
- The convergence time constant is small for large μ.
- The mean-square error in steady state is low for small μ.

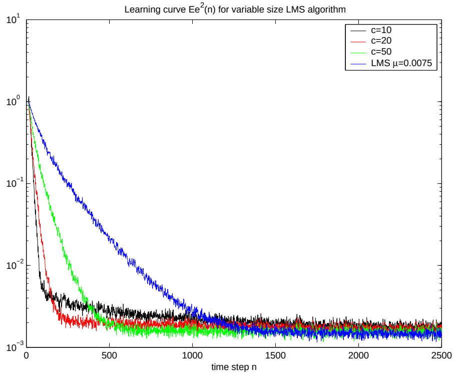
Therefore, in the initial adaptation stages μ is kept large, then it is monotonically reduced, such that in the final adaptation stage it is very small. There are many receipts of cooling down an adaptation process.

- Monotonically decreasing the step size
$$\mu(n) = \frac{1}{n+c}$$
Disadvantage for non-stationary data: the algorithm will not react anymore to changes in the optimum solution, for large values of n.
- Variable Step algorithm:
$$\underline{w}(n+1) = \underline{w}(n) + M(n)\underline{u}(n)e(n)$$
where
$$M(n) = \begin{bmatrix} \mu_0(n) & 0 & . & 0 \\ 0 & \mu_1(n) & . & 0 \\ 0 & 0 & . & 0 \\ 0 & 0 & . & \mu_{M-1}(n) \end{bmatrix}$$
or componentwise
$$w_i(n+1) = w_i(n) + \mu_i(n)u(n-i)e(n) \quad i = 0, 1, \ldots, M-1$$
* each filter parameter $w_i(n)$ is updated using an independent adaptation step $\mu_i(n)$.
* the time variation of $\mu_i(n)$ is ad-hoc selected as
  - if m_1 successive identical signs of the gradient estimate, $-e(n)u(n-i)$,are observed, then $\mu_i(n)$ is increased $\mu_i(n) = c_1\mu_i(n-m_1)$ $(c_1 > 1)$(the algorithm is still far of the optimum, is better to accelerate)
  - if m_2 successive changes in the sign of gradient estimate, $-e(n)u(n-i)$, are observed, then $\mu_i(n)$ is decreased $\mu_i(n) = \mu_i(n-m_2)/c_2$ $(c_2 > 1)$ (the algorithm is near the optimum, is better to decelerate; by decreasing the step-size, so that the steady state error will finally decrease).

Comparison of LMS and variable size LMS

Comparison of LMS and variable size LMS ($\tilde{\mu} = \frac{\mu}{0.01n+c}$) within the channel equalization example (channel equalization) with $c = [10; 20; 50]$





Lowpass filtering the noisy gradient

- Lowpass filtering the noisy gradient
  Let us rename the noisy gradient $\underline{g}(n) = \hat{\nabla}_{\underline{w}} J$

$$\begin{aligned} \underline{g}(n) &= \hat{\nabla}_{\underline{w}} J = -2\underline{u}(n)e(n) \\ g_i(n) &= -2e(n)u(n-i) \end{aligned}$$

  Passing the signals $g_i(n)$ through low pass filters will prevent the large fluctuations of direction during adaptation process.

$$b_i(n) = LPF(g_i(n))$$

  where LPF denotes a low pass filtering operation. The updating process will use the filtered noisy gradient

$$\underline{w}(n+1) = \underline{w}(n) - \mu \underline{b}(n)$$

  The following versions are well known:

- Averaged LMS algorithm
  When LPF is the filter with impulse response
  $h(0) = \frac{1}{N}, \ldots, h(N-1) = \frac{1}{N}, h(N) = h(N+1) = \ldots = 0$ we obtain simply the average of gradient components:

$$\underline{w}(n+1) = \underline{w}(n) + \frac{\mu}{N} \sum_{j=n-N+1}^{n} e(j)\underline{u}(j)$$

Momentum LMS algorithm

- Momentum LMS algorithm
  When LPF is an IIR filter of first order $h(0) = 1 - \gamma$, $h(1) = \gamma h(0)$, $h(2) = \gamma^2 h(0), \ldots$ then,

$$\begin{aligned} b_i(n) &= LPF(g_i(n)) = \gamma b_i(n-1) + (1-\gamma) g_i(n) \\ \underline{b}(n) &= \gamma \underline{b}(n-1) + (1-\gamma)\underline{g}(n) \end{aligned}$$

The resulting algorithm can be written as a second order recursion:

$$\begin{aligned} \underline{w}(n+1) &= \underline{w}(n) - \mu \underline{b}(n) \\ \gamma \underline{w}(n) &= \gamma \underline{w}(n-1) - \gamma\mu \underline{b}(n-1) \\ \underline{w}(n+1) - \gamma \underline{w}(n) &= \underline{w}(n) - \gamma \underline{w}(n-1) - \mu \underline{b}(n) + \gamma\mu \underline{b}(n-1) \\ \underline{w}(n+1) &= \underline{w}(n) + \gamma(\underline{w}(n) - \underline{w}(n-1)) - \mu(\underline{b}(n) - \gamma \underline{b}(n-1)) \\ \underline{w}(n+1) &= \underline{w}(n) + \gamma(\underline{w}(n) - \underline{w}(n-1)) - \mu(1-\gamma)\underline{g}(n) \\ \underline{w}(n+1) &= \underline{w}(n) + \gamma(\underline{w}(n) - \underline{w}(n-1)) + 2\mu(1-\gamma) e(n)\underline{u}(n) \end{aligned}$$

$$\boxed{\underline{w}(n+1) - \underline{w}(n) = \gamma(\underline{w}(n) - \underline{w}(n-1)) + \tilde{\mu}(1-\gamma) e(n)\underline{u}(n)}$$

Drawback: The convergence rate may decrease.
Advantages: The momentum term keeps the algorithm active even in the regions close to minimum. For nonlinear criterion surfaces this helps in avoiding local minima (as in neural network learning by backpropagation)

Nonlinear smoothing of LMS gradient estimates

- If there is an impulsive interference in either $d(n)$ or $u(n)$, the performances of LMS algorithm will drastically degrade (sometimes even leading to instability).
  Smoothing the noisy gradient components using a nonlinear filter provides a potential solution.
- The Median LMS Algorithm
  Computing the median of window size $N+1$, for each component of the gradient vector, will smooth out the effect of impulsive noise. The adaptation equation can be implemented as

$$w_i(n+1) = w_i(n) - \mu \cdot med\left((e(n)u(n-i)), (e(n-1)u(n-1-i)), \ldots, (e(n-N)u(n-N-i))\right)$$

* Experimental evidence shows that the smoothing effect in impulsive noise environment is very strong.
* If the environment is not impulsive, the performances of Median LMS are comparable with those of LMS, thus the extra computational cost of Median LMS is not worth.
* However, the convergence rate may be slower than in LMS, and there are reports of instability occurring whith Median LMS.