Lecture 2

During this lecture you will learn about

- The *Least Mean Squares* algorithm (LMS)
- Convergence analysis of the LMS
- Equalizer (Kanalutjämnare)

Background

The method of the *Steepest descent* that was studies at the last lecture is a recursive algorithm for calculation of the Wiener filter when the statistics of the signals are known (knowledge about $\mathbf{R}$ och $\mathbf{p}$).

The problem is that this information is oftenly unknown!

LMS is a method that is based on the same principles as the method of the Steepest descent, but where the statistics is estimated continuously.

Since the statistics is estimated continuously, the LMS algorithm can adapt to changes in the signal statistics; The LMS algorithm is thus an adaptive filter.

Because of estimated statistics the gradient becomes noisy. The LMS algorithm belongs to a group of methods referred to as *stochastic gradient* methods, while the method of the Steepest descent belongs to the group *deterministic gradient* methods.

The Least Mean Square (LMS) algorithm

We want to create an algorithm that minimizes $E\{|e(n)|^2\}$, just like the SD, but based on unkown statistics.

A strategy that then can be used is to uses estimates of the autocorrelation matrix $\mathbf{R}$ and the cross correlationen vector $\mathbf{p}$. If instantaneous estimates are chosen,

$$\widehat{\mathbf{R}}(n) = \mathbf{u}(n)\mathbf{u}^H(n)$$
$$\widehat{\mathbf{p}}(n) = \mathbf{u}(n)d^*(n)$$

the resulting method is the *Least Mean Squares* algorithm.

The Least Mean Square (LMS) algorithm

For the SD, the update of the filter weights is given by

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

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

In the LMS we use the estimates $\widehat{\mathbf{R}}$ och $\widehat{\mathbf{p}}$ to calculate $\widehat{\boldsymbol{\nabla}} J(n)$. Thus, also the updated filter vector becomes an estimate. It is therefore denoted $\widehat{\mathbf{w}}(n)$;

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

The Least Mean Square (LMS) algorithm 5

For the LMS algorithm, the update equation of the filter weights becomes

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

where $e^*(n) = d^*(n) - \mathbf{u}^H(n)\widehat{\mathbf{w}}(n)$.

Compare with the corresponding equation for the SD:

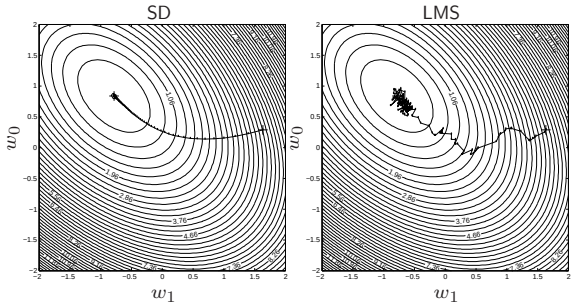
$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu E\{\mathbf{u}(n) e^*(n)\}$$

where $e^*(n) = d^*(n) - \mathbf{u}^H(n)\mathbf{w}(n)$.

The difference is thus that $E\{\mathbf{u}(n)e^*(n)\}$ is estimated as $\mathbf{u}(n)e^*(n)$. This leads to *gradient noise*.

The Least Mean Square (LMS) algorithm 6

Illustration of *gradient noise*. $\mathbf{R}$ och $\mathbf{p}$ in the example from Lecture 1. $\mu = 0.1$.



Because of the noise in the gradientm the LMS never reaches J_{min}, which the SD does.

Convergence analysis of the LMS, overview 7

Consider the update equation

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

We want to know how fast and towards what the algorithm converges,in terms of $J(n)$ and $\widehat{\mathbf{w}}(n)$.

Strategy:

1. Introduce the filter weight error vector $\boldsymbol{\epsilon}(n) = \widehat{\mathbf{w}}(n) - \mathbf{w}_o$.
2. Express the update equation in $\boldsymbol{\epsilon}(n)$.
3. Express $J(n)$ in $\boldsymbol{\epsilon}(n)$. This leads to $\mathbf{K}(n) = E\{\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^H(n)\}$.
4. Calculate the difference equation in $\mathbf{K}(n)$, which controls the convergence.
5. Make a variable change from $\mathbf{K}(n)$ to $\mathbf{X}(n)$ which converges equally fast.

1. The filter weight error vector 8

The LMS contains feedback, and just like for the SD there is a risk that the algorithm diverges, if the stepsize μ is not chosen appropriately.

In order to investigate the stability, the filter weight error vector is introduced $\boldsymbol{\epsilon}(n)$:

$$\boldsymbol{\epsilon}(n) = \widehat{\mathbf{w}}(n) - \mathbf{w}_o \qquad (\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{p} \quad \text{Wiener-Hopf})$$

Note that $\boldsymbol{\epsilon}(n)$ corresponds to $\mathbf{c}(n) = \mathbf{w}(n) - \mathbf{w}_o$ for the *Steepest descent*, but that $\boldsymbol{\epsilon}(n)$ is stochastic because of $\widehat{\mathbf{w}}(n)$.

2. Update equation in $\boldsymbol{\epsilon}(n)$

The update equation for $\boldsymbol{\epsilon}(n)$ at time $n+1$ can be expressed recursively as

$$\begin{aligned}\boldsymbol{\epsilon}(n+1) &= [\mathbf{I} - \mu\widehat{\mathbf{R}}(n)]\boldsymbol{\epsilon}(n) + \mu[\widehat{\mathbf{p}}(n) - \widehat{\mathbf{R}}(n)\mathbf{w}_o] \\ &= [\mathbf{I} - \mu\mathbf{u}(n)\mathbf{u}^H(n)]\boldsymbol{\epsilon}(n) + \mu\mathbf{u}(n)e_o^*(n)\end{aligned}$$

where $e_o^*(n) = d(n) - \mathbf{w}_o^H\mathbf{u}(n)$.

Compare to that of the SD:

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

3. Express $J(n)$ in $\boldsymbol{\epsilon}(n)$

The convergence analysis of the LMS is considerably more complicated than that for the SD. Therefore, two tools are needed (approximations).

These are

- *Independence theory*
- *Direct-averaging method*

3. Express $J(n)$ i $\boldsymbol{\epsilon}(n)$, cont.

Independence Theory

1. The input vectors of different time instants n,
   $\mathbf{u}(1), \mathbf{u}(2), \ldots, \mathbf{u}(n)$,
   is independet of each other (and thereby uncorrelated).
2. The input signal vector at time n is independent of the desired signal
   at all earlier time instants,
   $\mathbf{u}(n)$ idependent of $d(1), d(2), \ldots, d(n-1)$.
3. The desired signal at time n, $d(n)$,
   is independent of all earlier desired signals, $d(1), d(2), \ldots, d(n-1)$, but dependent of the input signal vector at time n, $\mathbf{u}(n)$.
4. The signals $d(n)$ and $\mathbf{u}(n)$ at the same time instant n is mutually normally distributed.

3. Express $J(n)$ in $\boldsymbol{\epsilon}(n)$, cont.

Direct-averaging

Direct-averaging means that in the update equation for $\boldsymbol{\epsilon}$,

$$\boldsymbol{\epsilon}(n+1) = [\mathbf{I} - \mu\mathbf{u}(n)\mathbf{u}^H(n)]\boldsymbol{\epsilon}(n) + \mu\mathbf{u}(n)e_o^*(n) \ ,$$

the instantaneous estimate $\widehat{\mathbf{R}}(n) = \mathbf{u}(n)\mathbf{u}^H(n)$ is substituted with the expectation over the ensemble, $\mathbf{R} = E\{\mathbf{u}(n)\mathbf{u}^H(n)\}$, such that

$$\boldsymbol{\epsilon}(n+1) = (\mathbf{I} - \mu\mathbf{R})\boldsymbol{\epsilon}(n) + \mu\mathbf{u}(n)e_o^*(n)$$

results.

3. Express $J(n)$ i $\boldsymbol{\epsilon}(n)$, cont.

For the SD we could show that

$$J(n) = J_{min} + [\mathbf{w}(n) - \mathbf{w}_o]^H \mathbf{R} [\mathbf{w}(n) - \mathbf{w}_o]$$

and that the eigenvalues of $\mathbf{R}$ controls the convergence speed.

For the LMS, $\boldsymbol{\epsilon}(n) = \widehat{\mathbf{w}}(n) - \mathbf{w}_o$,

$$\begin{aligned} J(n) &= E\{|e(n)|^2\} = E\{|d(n) - \widehat{\mathbf{w}}^H(n)\mathbf{u}(n)|^2\} \\ &= E\{|e_o(n) - \boldsymbol{\epsilon}^H(n)\mathbf{u}(n)|^2\} = \\ &\overset{i}{=} \underbrace{E\{|e_o(n)|^2\}}_{J_{min}} + E\{\boldsymbol{\epsilon}^H(n) \underbrace{\mathbf{u}(n)\mathbf{u}^H(n)}_{\widehat{\mathbf{R}}(n)} \boldsymbol{\epsilon}(n)\} \end{aligned}$$

Here, (i) indicates that the indpendence assupmtion is used. Since $\boldsymbol{\epsilon}$ and $\widehat{\mathbf{R}}$ are estimates, and in addition not independent, the expectation operator cannot just be "moved into $\widehat{\mathbf{R}}$.

3. Express $J(n)$ i $\boldsymbol{\epsilon}(n)$, cont.

The behavior of $J_{ex}(n) = J(n) - J_{min} = E\{\boldsymbol{\epsilon}^H(n)\widehat{\mathbf{R}}\boldsymbol{\epsilon}(n)\}$ determines the convergence.

Since the following holds for the trace of a matrix,

1. $\mathrm{tr}(scalar) = scalar$
2. $\mathrm{tr}(\mathbf{AB}) = \mathrm{tr}(\mathbf{BA})$
3. $E\{\mathrm{tr}(\mathbf{A})\} = \mathrm{tr}(E\{\mathbf{A}\})$

$J_{ex}(n)$ can be written as

$$\begin{aligned} J_{ex}(n) &\overset{1,2}{=} E\{\mathrm{tr}[\widehat{\mathbf{R}}(n)\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^H(n)]\} \\ &\overset{3}{=} \mathrm{tr}[E\{\widehat{\mathbf{R}}(n)\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^H(n)\}] \\ &\overset{i}{=} \mathrm{tr}[E\{\widehat{\mathbf{R}}(n)\}E\{\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^H(n)\}] \\ &= \mathrm{tr}[\mathbf{RK}(n)] \end{aligned}$$

The convergence thus depends on $\mathbf{K}(n) = E\{\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^H(n)\}$.

4. Difference equation of $\mathbf{K}(n)$

The update equation for the filter weight error is, as shown previously,

$$\boldsymbol{\epsilon}(n+1) = [\mathbf{I} - \mu\mathbf{u}(n)\mathbf{u}^H(n)]\boldsymbol{\epsilon}(n) + \mu\mathbf{u}(n)e_o^*(n)$$

The solution to this stochastic difference equation is for small μ close to the solution for

$$\boldsymbol{\epsilon}(n+1) = (\mathbf{I} - \mu\mathbf{R})\boldsymbol{\epsilon}(n) + \mu\mathbf{u}(n)e_o^*(n) \ ,$$

which is resulting from *Direct-averaging*.

This equation is still difficult to solve, but now the behavior of $\mathbf{K}(n)$ can be described by the following difference equation

$$\mathbf{K}(n+1) = (\mathbf{I} - \mu\mathbf{R})\mathbf{K}(n)(\mathbf{I} - \mu\mathbf{R}) + \mu^2 J_{min}\mathbf{R}$$

5. Changing of variables from $\mathbf{K}(n)$ to $\mathbf{X}(n)$

The variable changes

$$\mathbf{Q}^H\mathbf{RQ} = \boldsymbol{\Lambda}$$

$$\mathbf{Q}^H\mathbf{K}(n)\mathbf{Q} = \mathbf{X}(n)$$

where $\boldsymbol{\Lambda}$ is the diagonal eigenvalue matrix of $\mathbf{R}$, and where $\mathbf{X}(n)$ normally is not diagonal, yields

$$J_{ex}(n) = \mathrm{tr}(\mathbf{RK}(n)) = \mathrm{tr}(\boldsymbol{\Lambda}\mathbf{X}(n))$$

The convergence can be discribed by

$$\mathbf{X}(n+1) = (\mathbf{I} - \mu\boldsymbol{\Lambda})\mathbf{X}(n)(\mathbf{I} - \mu\boldsymbol{\Lambda}) + \mu^2 J_{min}\boldsymbol{\Lambda}$$

5. Changing of variables from $\mathbf{K}(n)$ to $\mathbf{X}(n)$, cont.

$J_{ex}(n)$ depends only on the diagonal elements of $\mathbf{X}(n)$ (because of the trace $\mathrm{tr}[\mathbf{\Lambda X}(n)]$). We can therefore write $J_{ex}(n) = \sum_{i=1}^{M} \lambda_i x_i(n)$.

The update of the diagonal elements is controlled by

$$x_i(n+1) = (1-\mu\lambda_i)^2 x_i(n) + \mu^2 J_{min}\lambda_i$$

This is an inhomogenous difference equation.This means that the solution will contain a transient part, and a stationary part that depends on J_{min} and μ. The cost function can therefore be written

$$J(n) = J_{min} + J_{ex}(\infty) + J_{trans}(n)$$

where $J_{min}+J_{ex}(\infty)$ och $J_{trans}(n)$ is the stationary and transient solutions, respectively.

At the best, the LMS can thus reach $J_{min}+J_{ex}(\infty)$.

Properties of the LMS

- Convergence for the same interval as the SD:
$$0 < \mu < \frac{2}{\lambda_{max}}$$
- $J_{ex}(\infty) = J_{min} \sum_{i=1}^{M} \frac{\mu\lambda_i}{2-\mu\lambda_i}$
- Misadjustment $\mathcal{M} = \frac{J_{ex}(\infty)}{J_{min}}$ is a measure of how close the the optimal solution the LMS (in mean-square sense).

Rules of thumb LMS

The eigenvalues of $\mathbf{R}$ are rarely known. Therefore, a set of rules of thumbs is commonly used, that is based on the *tap-input power*, $\sum_{k=0}^{M-1} E\{|u(n-k)|^2\} = Mr(0) = \mathrm{tr}(\mathbf{R})$.

- The stepsize $0 < \mu < \frac{2}{\sum_{k=0}^{M-1} E\{|u(n-k)|^2\}}$
- Misadjustment $\mathcal{M} \approx \frac{\mu}{2} \sum_{k=0}^{M-1} E\{|u(n-k)|^2\}$
- Average time constant $\tau_{mse,av} \approx \frac{1}{2\mu\lambda_{av}}$ där
  $\lambda_{av} = \frac{1}{M}\sum_{i=1}^{M}\lambda_i$.
  When λ_i is unknown, the following relationship is useful.
- Approximate relationship $\mathcal{M}$ and $\tau_{mse,av}$ $\quad \mathcal{M} \approx \frac{M}{4\tau_{mse,av}}$

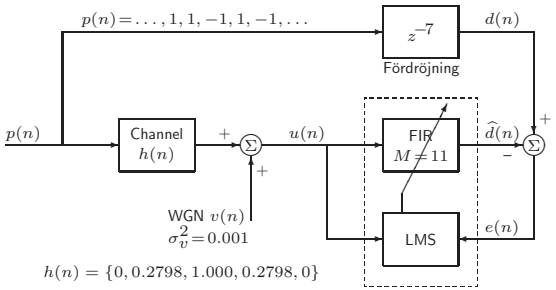
Here, $\tau_{mse,av}$ denotes the time it takes for $J(n)$ to decay a factor of e^{-1}.

Summary LMS

Summary of the iterations:

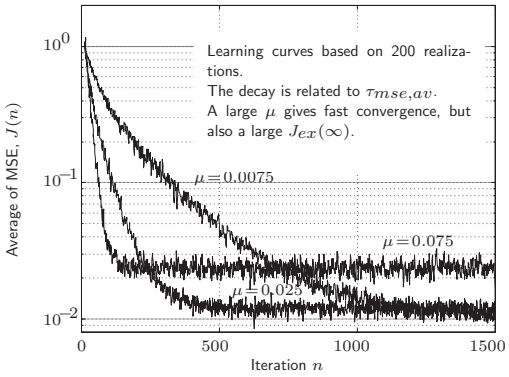
1. Set start values for the filter coefficients, $\hat{\mathbf{w}}(0)$
2. Calculate the error $e(n) = d(n) - \hat{\mathbf{w}}^H(n)\mathbf{u}(n)$
3. Update the filter coefficients
   $\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)[d^*(n) - \mathbf{u}^H(n)\hat{\mathbf{w}}(n)]$
4. Repeat steps 2 and 3

Example: Channel equalizer in training phase



Sent information in the form of a pn-sequence is affected by a channel ($h(n)$). An adaptive inverse filter is used to compensate for the channel such that the total effect can be approximated by a delay. White Gaussian noise is added during the transmission (WGN).

Example, cont: Learning Curve



The curves illustrates learning curves for two different μ.

What to read?

- Least Mean Squares, Haykin chap.5.1–5.3, 5.4 (see lecture), 5.5–5.7

Exercises: 4.1, 4.2, 1.2, 4.3, 4.5, (4.4)

Computer exercise, theme: Implement the LMS algorithm in Matlab, and generate the results in Haykin chap.5.7