Neural networks: Least mean squares algorithm (LMS)

Prof. Sven Lončarić

Faculty of Electrical Engineering and Computing

sven.loncaric@fer.hr

http://www.fer.hr/ipg

Overview of topics

- Introduction
- Wiener-Hopf equation
- Steepest descent method
- Least mean squares (LMS) learning algorithm
- Learning curve
- Discussion

Introduction

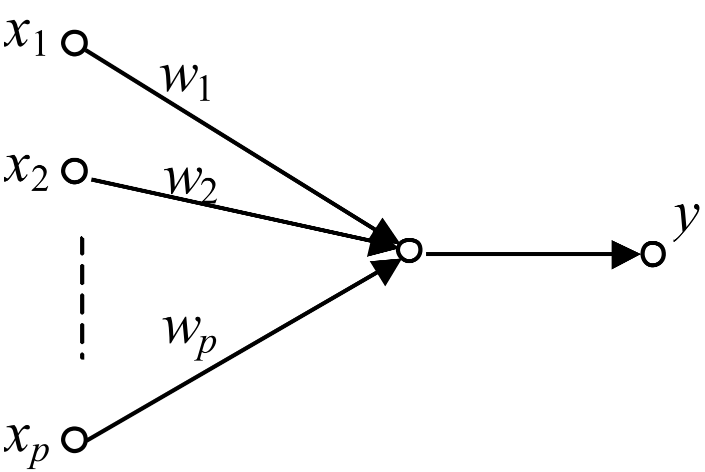
- In this section we talk about a class of neural networks with a single neuron that work in linear mode (there is no nonlinear activation function)
- Such networks are important because:
  - Linear adaptive filters are well studied and applied in communications, control, and biomedicine
  - They represent a first step towards multilayer neural networks

Introduction

- LMS = least mean squares
- LMS is a learning algorithm
- LMS algorithm was developed by Widrow and Hoff, 1960
- LMS algorithm is used in various applications of adaptive signal processing including:
  - Adaptive equalization of communication channels,
  - Echo cancellation on phone lines,
  - Adaptive signal detection in presence of noise

Optimal filtering problem

- Assume there are p sensors located in space
- Let $x_1, x_2, \ldots, x_p$ be signals acquired by the sensors and multipled by weights $w_1, w_2, \ldots, w_p$
- We need to determine $w_1, w_2, \ldots, w_p$ to minimize difference between obtained response y and desired response d in the sense of the mean square error



$$y = \sum_{k=1}^{p} w_k x_k$$

Optimal filtering problem

- Error signal is defined as:

  $$e = d - y$$

- Let d be a random variable
- Let input values x_k be random variables – such a sequence of random variables is a random process
- In that case y and e are also random variables

Optimal filtering problem

- Mean square error is used as an error measure:

  $$J = 1/2\ E\ [e^2]$$

  where E is the statistical expectation operator
- The optimal filtering problem is:
  Determine the optimal set of weights $w_1, w_2, \ldots, w_p$ for which the mean square error J is minimal
- In signal processing, the solution to this problem is called Wiener filter

Optimal filtering problem

- Expression for mean square error can be written as:

$$J = \frac{1}{2}E\left[d^2\right] - E\left[\sum_{k=1}^{p} w_k x_k d\right] + \frac{1}{2}E\left[\sum_{j=1}^{p}\sum_{k=1}^{p} w_j w_k x_j x_k\right]$$

- Furthermore the expression can be simplified as:

$$J = \frac{1}{2}E\left[d^2\right] - \sum_{k=1}^{p} w_k E\left[x_k d\right] + \frac{1}{2}\sum_{j=1}^{p}\sum_{k=1}^{p} w_j w_k E\left[x_j x_k\right]$$

where weights w_i as constants are extracted in front of the expectation operator

Optimal filtering problem

- Let us introduce the following notation:
- Mean square error of desired output *d*:

$$r_d = E\left[d^2\right]$$

- Crosscorrelation function of desired output *d* and signal x_k:

$$r_{dx}(k) = E\left[dx_k\right], \quad k = 1,2,\ldots,p$$

- Autocorrelation function of input signal:

$$r_x(j,k) = E\left[x_j x_k\right], \quad j,k = 1,2,\ldots,p$$

Optimal filtering problem

- Using this notation we can write the expression for the mean square error as:

$$J = \frac{1}{2} r_d - \sum_{k=1}^{p} w_k r_{dx}(k) + \frac{1}{2} \sum_{j=1}^{p} \sum_{k=1}^{p} w_j w_k r_x(j,k)$$

- Multidimensional representation of *J* as a function of weights w_1, w_2, …, w_p as free parameters is called error surface
- Error surface is a second order function of weights that has one global minimum

Error minimization

- To determine the optimum of error function J which depends on p variables w_1, w_2, …, w_p we need to calculate partial derivatives of J with respect to w_1, w_2, …, w_p and equate them to zero
- The partial derivatives of error function J are given by:

$$\frac{\partial J}{\partial w_k} = -r_{dx}(k) + \sum_{j=1}^{p} w_j r_x(j,k), \quad k = 1,2,\ldots,p$$

Wiener-Hopf equations

- By equating the partial derivatives to zero we obtain Wiener-Hopf equations:

$$\sum_{j=1}^{p} w_j r_x(j,k) = r_{dx}(k), \quad k = 1,2,\ldots,p$$

- Weights w_1, w_2, …, w_p that satisfy Wiener-Hopf equations define the filter called Wiener filter
  - To determine unknown weights we need to solve the linear system of Wiener-Hopf equations, so it is necessary to calculate inverse matrix of dimension $p \times p$
  - Iterative numerical methods can be used to avoid computation of the inverse matrix

Steepest descent method

- Steepest descent method may be used to iteratively determine weights values by moving across the error surface towards the global minimum
- Error correction in iteration *n* is equal to product of constant η and gradient:

$$\Delta w_k(n) = -\eta \frac{\partial J}{\partial w_k}, \quad k = 1,2,\ldots,p$$

where η is a positive constant determining the learning rate

Steepest descent method

- Given the old weight value in iteration *n* the new value is calculated as:

$$w_k(n+1) = w_k(n) + \Delta w_k(n), \quad k = 1,2,\ldots,p$$

- If we use the previously derived expression for gradient, we obtain the final expression:

$$w_k(n+1) = w_k(n) + \eta\left[r_{dx}(k) - \sum_{j=1}^{p} w_j(n) r_x(j,k)\right], \quad k = 1,2,\ldots,p$$

Least mean squares (LMS) algorithm

- The problem of determining weights of the Wiener filter is that we need to know cross-correlation function $r_{dx}(k)$ and autocorrelation function $r_x(j,k)$
- LMS algorithm is a special case of the previously derived steepest descent algorithm
- LMS algoritm uses the following estimations of autocorrelation and cross-correlation functions:

$$\hat{r}(j,k;n) = x_j(n)x_k(n)$$
$$\hat{r}_{dx}(k;n) = x_k(n)d(n)$$

LMS algorithm

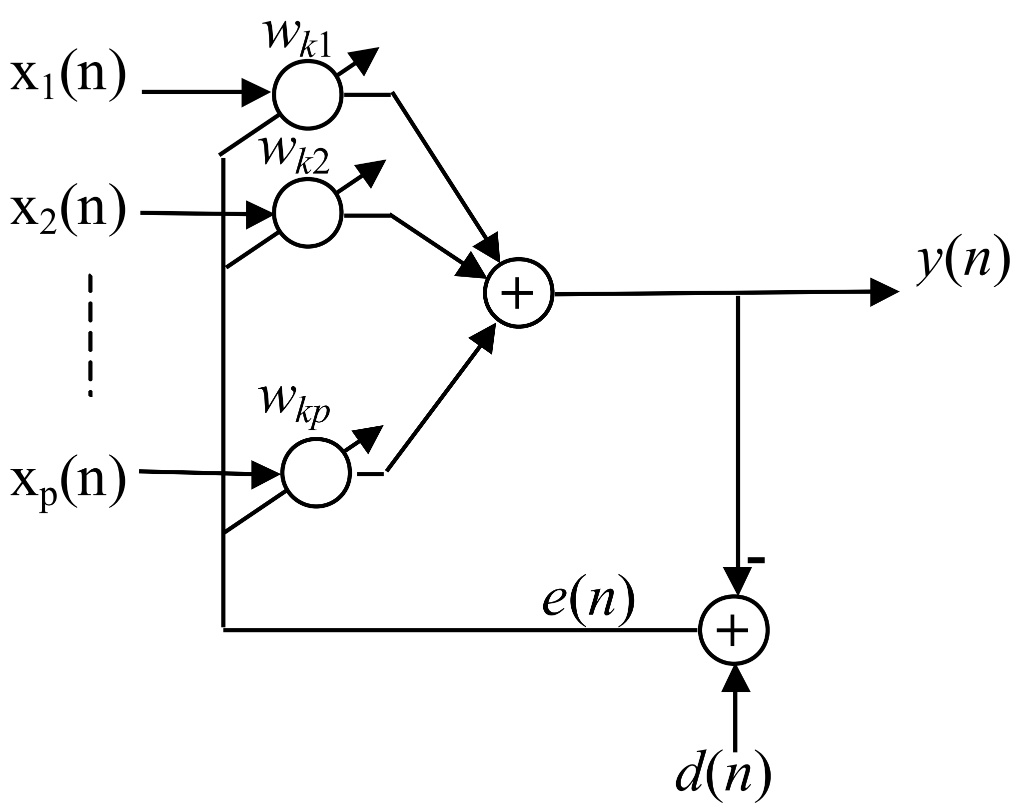
- If we substitute the estimations into previously derived expression we obtain the LMS learning algorithm (aka delta learning rule or Widrow-Hoff learning rule) :

$$
\begin{aligned}
w_k(n+1) &= w_k(n) + \eta\left[x_k(n)d(n) - \sum_{j=1}^{p} w_j(n)x_j(n)x_k(n) \right] \\
&= w_k(n) + \eta\left[d(n) - \sum_{j=1}^{p} w_j(n)x_j(n) \right] x_k(n) \\
&= w_k(n) + \eta\left[d(n) - y(n) \right] x_k(n), \quad k = 1,2,\ldots,p
\end{aligned}
$$

$$\Delta w_k(n) = \eta e(n) x_k(n)$$

An adaptive filter using LMS algorithm

- Figure shows an adaptive filter
- Weights are calculated in real time using LMS algorithm



LMS algorithm - summary

1. Initialization. Set initial weight values

$$w_k(1) = 0, \quad k = 1,2,\ldots,p$$

2. Filtering. For moments *n*=1, 2, ... calculate

$$y(n) = \sum_{1=1}^{p} w_j(n) x_j(n)$$

$$e(n) = d(n) - y(n)$$

$$w_k(n+1) = w_k(n) + \eta e(n) x_k(n), \quad k = 1,2,\ldots,p$$

Learning curve

- Learning curve is a plot of $J(n)$ with respect to iteration n
- Learning curve shows characteristics of the learning process
- For an LMS algorithm one characteristic value is the steady-state error value $J(\infty)$
- The steady-state error value is always larger than stationary error $J_{\min}$ for a given Wiener filter:
- Difference of these two errors is called residual error:

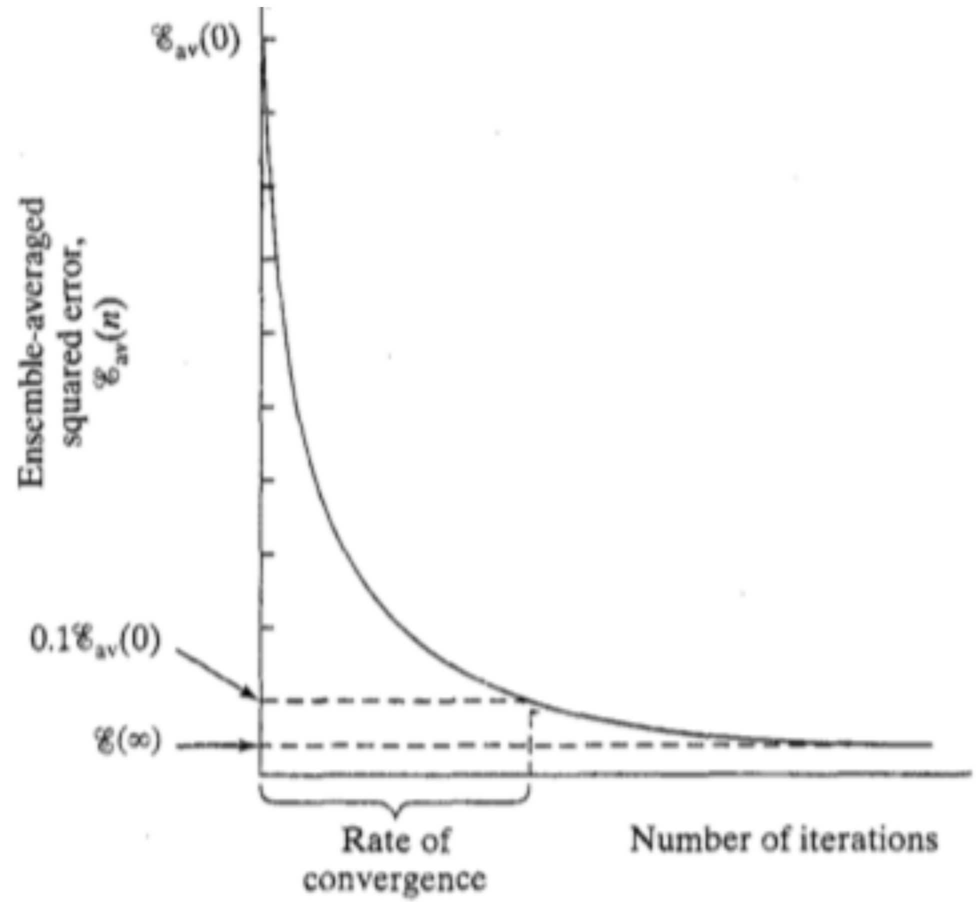
$$J_{\text{ex}} = J(\infty) - J_{\min}$$

Learning curve misadjustment

- Quotient of the residual error and Wiener error is called misadjustment :
$$M = \frac{J_{\mathrm{ex}}}{J_{\min}}$$
- Misadjustment is expressed in percentages (for example misadjustment of 10% is considered acceptable in practice)

Learning curve settling time

- Another important feature of LMS algorithm is the settling time
  - Settling time can be defined as average time constant of the exponential curve that approximates learning curve *J*(*n*)



Learning curve

- Misadjustment is proportional to learning rate η
- Settling time is inverse proportional to learning rate η
- These are contradictory requirements because if we reduce learning rate η to reduce misadjustment then settling time increases
- During design it is necessary to carefully select learning rate η to satisfy all requirements

Variable learning rate

- The simplest learning is with constant learning rate η, which does not change with iteration indeks n:

  $\eta(n)=\eta_{0,}$ for each n

- Convergence of LMS algorithm can be improved if a variable learning rate is used, e.g.:

  $\eta(n) = c / n,$ where c is a constant

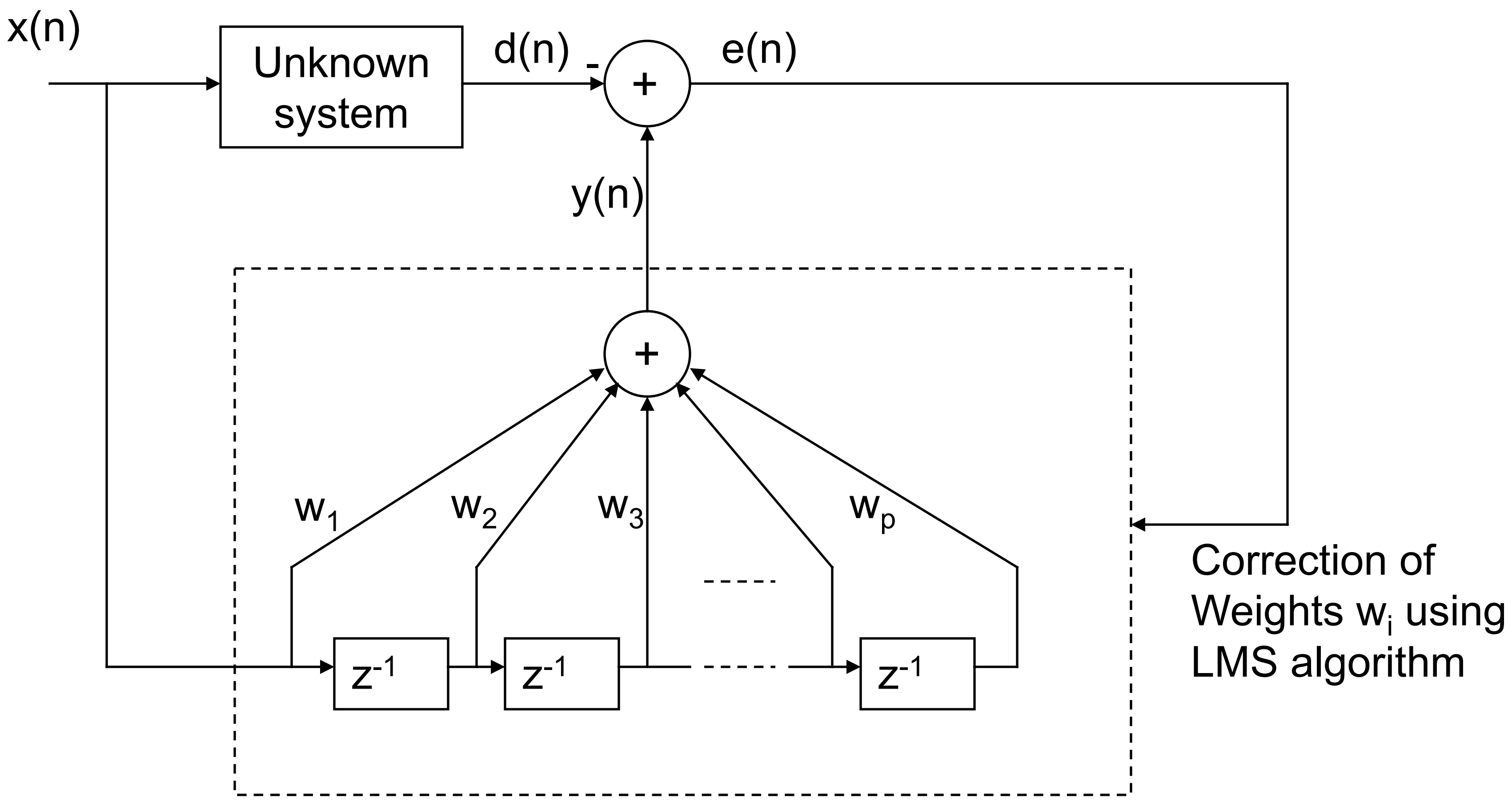
- Due to large values for small n, a modified expression is used:

$$\eta(n) = \frac{\eta_0}{1 + (n / \tau)}$$

Applications of LMS algorithm

- LMS algorithm has numerous applications including:
  - System identification
  - Adaptive noise control (noise cancellation)

Example: System identification

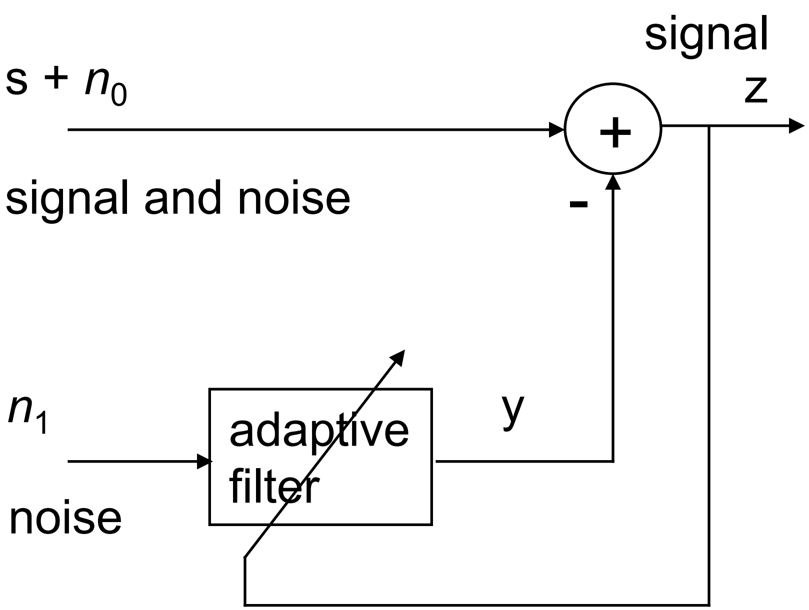


System model (adaptive FIR filter)

Example: Adaptive noise cancellation

- Applications:
- Noise cancellation in pilot cockpit
  - Jet engine noise up to 140 dB, pilot speech 30 dB
  - Speech communication is not possible without noise cancellation
  - Engine noise is not constant but depends on the flight regime
  - Noise cancellation must be adaptive with respect to noise
- Noise cancellation headphones

Adaptive noise cancellation



- s is signal, n_0 is noise which is not correlated to signal s
- n_1 is referent noise correlated to noise n_0
- y is estimate of noise n_0
- E.g. n_1 is engine noise recorded outside of the cockpit and s + n_0 is a noisy speech signal in the cockpit

Adaptive noise cancellation

- Signal *y* is not correlated to signal s
- Noise n_0 is not correlated to signal s
- Then it holds that:

$$z = s + n_0 - y$$

$$E[z^2] = E[s^2] + E[(n_0 - y)^2]$$

- When adaptive filter is changed to minimize $E[z^2]$, then we also minimize $E[(n_0 - y)^2]$

Adaptive noise cancellation

- Hence, we can use signal *z* as the error signal in the adaptive filter that learns using LMS algorithm
- For adaptation we use LMS learning rule:

$$\Delta w_k(i) = \eta z(i) n_1(i)$$

Discussion

- LMS algorithm is a widely accepted algorithm in adaptive signal processing due to its properties:
  - Simplicity of implementation
  - Works well in unknown environment
  - Can adapt to variability of statistical properties of non-stationary signals