02471 Machine Learning for Signal Processing

Week 4: Adaptive Linear Filtering with LMS

This exercise is based on S. Theodoridis: Machine Learning, A Bayesian and Optimization Perspective 2nd edition, section(s) 2.6, 5.1–5.4 (excluding 5.3.1), 5.6.

The objective of the exercise is to expand on the linear filtering exercise from week 3 and get acquainted with adaptive (online learning) filtering.

Overview

Adaptive filters are used in a wide range of applications where the characteristics of the signals or the system is not known a priori, or is known to be time varying.

In this exercise we will consider the adaptive finite impulse response (FIR) filter. The adaptation strategy will be both the Least-Mean-Square (LMS) and Normalized Least-Mean-Square (NLMS) algorithms.

The exercise have the following structure:

4.1 will review the wiener filter setup and find the solution given values for the correlation functions.

4.2 will consider the system identification setup, first applying a Wiener filter, and then investigate how the adaptive algorithms LMS and NLMS performs.

4.3 will consider an audio signal (from last week) contaminated with noise and use adaptive filtering to suppress the noise.

The exercises should be completed in order, but, if you would like to see the adaptive filter in action before deriving the results, consider starting with 4.3.

Notation

- $r(k)$ refers to a correlation between two signal (called cross-correlation), or the correlation between the signal itself (called auto-correlation). The function $\hat{r}(k)$ denotes the estimate of $r(k)$ from data.
- $r_{xy}(k) = \mathbb{E}[x(n)y(n-k)] = \mathbb{E}[\mathrm{x}_n \mathrm{y}_{n-k}]$ refers to the cross-correlation between the two stochastic processes $x(n)$ and $y(n)$. This is often in DSP literature denoted as $\gamma_{xy}(m)$.
- $r_x(k) = \mathbb{E}[x(n)x(n-k)] = \mathbb{E}[\mathrm{x}_n \mathrm{x}_{n-k}]$ refers to the autocorrelation of x_n. This function is also often written as $r_{xx}(k)$ or $\gamma_{xx}(m)$.

Code

The code can be found in the .m and .py files named in the same way as exercises, ie. the code for exercise 4.x.y is in the file 4_x_y.m (or .py).

For coding exercises that requires implementation we will usually write `complete this line` where the implementing should be done.

Solutions

The solution is provided for all derivation exercises, and often hints are provided at the end of the document. If you get stuck, take a look at the hints, and if you are still stuck, take a look in the solution to see the approach being taken. Then try to do it on your own.

Solutions are also provided for some coding exercises. If you get stuck, take a look at the solution, and then try to implement it on your own.

4.1 Wiener Filter review

In order to solve the subsequent exercises we will briefly review the Wiener filter. Make sure to read section 4.2 and 4.5 in the book before completing this exercise.

This problem considers a digital system for measurement and analysis of the analog signal $s_a(t)$. The measurement and A/D conversion is carried out with appropriate sampling frequency, so it can be assumed that the required spectral limitation of the input signal is appropriate.

During the conversion process, the signal is contaminated by a zero–mean noise sequence which is assumed to be uncorrelated with the source signal $s_a(t)$. The available digital signal is then given by

$$\mathrm{u}_n = \mathrm{s}_n + \mathrm{v}_n$$

where s_n is the sampled source signal $s_a(nT)$ with an appropriate sampling period $T > 0$, and v_n is the uncorrelated noise signal. We assume that all signals are stochastic and wide-sense stationary.

For entire exercise 4.1, we assume a FIR Wiener filter of length 2.

Exercise 4.1.1

Write up the normal equations for the Wiener filter using correlations functions for the input signal u_n and desired signal d_n on the form

$$\begin{bmatrix} r_?(?) & \cdots & r_?(?) \\ \vdots & \ddots & \vdots \\ r_?(?) & \cdots & r_?(?) \end{bmatrix} \begin{bmatrix} w_? \\ \vdots \\ w_? \end{bmatrix} = \begin{bmatrix} r_?(?) \\ \vdots \\ r_?(?) \end{bmatrix}$$

$$\Sigma_? \boldsymbol{w} = \boldsymbol{p}$$

Exercise 4.1.2

Determine expressions for the correlation functions $r_u(k)$ and $r_{du}(k)$ in terms of $r_s(k)$ and $r_v(k)$.

Exercise 4.1.3

Now assume that the follow correlations are estimated

k	0	1	2
$r_v(k)$	0.6	-0.1	-0.1
$r_s(k)$	0.6	-0.3	0.2

Determine the values of the filter coefficients.

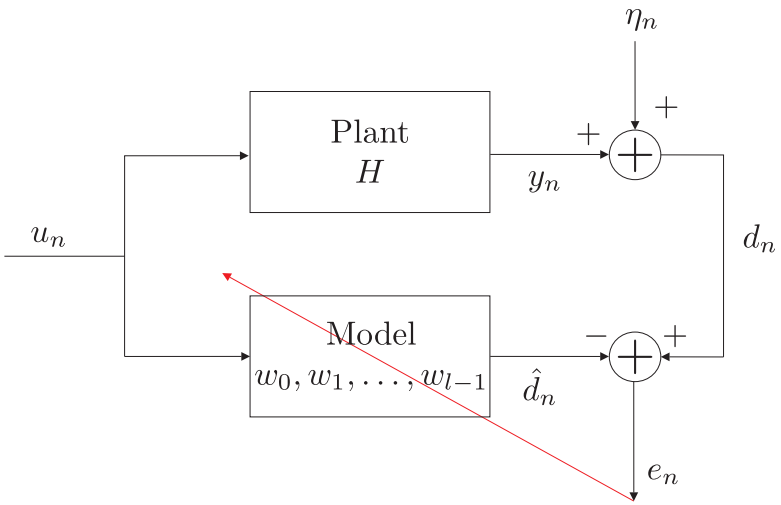
Exercise 4.1.4

We are now given the information that the variance (i.e. the power) of the desired signal is $\sigma_d^2 = 0.9$. Compute the minimum mean squared error for the filter.

4.2 System identification using a Wiener filter

In this exercise the system identification setup is considered. Make sure to read sec 4.7.2 and 5.5–5.5.1 in the book before completing this exercise.

The system identification setup is depicted on the following figure (note that the variables are written as realizations of signals);



The figure contains the following signals: the input signal to the systems, impulse response for the unknown system, impulse response for the filter to be trained, measurement noise (assumed to be white noise), measured output from the unknown system, output from the filter, residual noise from the filter, and a signal that is used as target for the filter.

The goal of this exercise is to derive a solution for the Wiener filter setup and an expression for the expected error for the Wiener filter setup. In other words, we want to apply the Wiener filter setup from last week to a system identification setup.

Exercise 4.2.1

Identify all the signals on the figure, i.e. determine what the different symbols signify (use the terminology used to describe Wiener Filter signals). What is the input sequence, desired sequence and error sequence? What is the filter length?

Exercise 4.2.2

Now we let our input signal (also sometimes called the excitation signal), be a white noise sequence with variance σ_u^2. Additionally, we assume that the η_n is uncorrelated with u_n. If we organize the input sequence in a vector, $\mathbf{u}_n = [\mathrm{u}_n, \cdots, \mathrm{u}_{n-1}, \mathrm{u}_{n-l+1}]^T$, we have $\mathrm{y}_n = \mathbf{u}_n * H = \mathbf{u}^T H$.

Show that $\mathbb{E}[(\mathbf{u}_n * H)\mathrm{u}_{n-k}] = h_k \sigma_{\mathrm{u}}^2$, where $H = [h_0, h_1, \cdots, h_{p-1}]^T$, assuming $p = l$. This result will be used in the next exercise.

Exercise 4.2.3

By solving the normal equations, as was done in exercise 4.1, we obtain the solution for the filter weights $\boldsymbol{w} = [w_0, w_1, \cdots, w_{l-1}]^T$.

Derive expressions for the input correlation matrix Σ_u and cross-correlation vector $\boldsymbol{p}$ by using the expressions for correlation functions, and show that $\boldsymbol{w}_* = H$ (assuming $p = l$).

Exercise 4.2.4 (This is a difficult exercise)

Derive a result for the mean squared error, $\mathbb{E}[\mathrm{e}_n^2]$, and show that the mean squared error can be expressed as $\mathbb{E}[\mathrm{e}_n^2] = \sigma_u^2 \|H - \boldsymbol{w}\|_2^2 + \sigma_\eta^2$ (assuming $p = l$). What is your interpretation of this result?

Exercise 4.2.5

Now we investigate the results obtains so far with data (realizations of the stochastic processes). We consider the setup where $H = \{1, 0.8, 0.2\}$, and let the excitation signal, u_n, be a realization of a white noise sequence of length 200. Furthermore, we assume $\eta_n = 0$ (no measurement noise).

Inspect and modify the code for this exercise (`ex4_2_5`) and use it to compute the plug-in Wiener filter solution. Consider the estimated weights compared to the true weights, e.g. by creating a curve with the size of the signal on the x-axis and error on y-axis.

Exercise 4.2.6

Now replicate the previous exercise, but compute the weights using the LMS and NLMS algorithm instead, by using the code for `ex4_2_6` as template. Consider the residuals and the convergence of the adaptive filter, e.g by plotting the error and the weights over time.

Experiment with different learning rates. Try to add noise to the output sequence, e.g. consider $\eta_n \neq 0$. Comment on the results.

Exercise 4.2.7

Now increase the excitation signal to a length of 500 and use the NLMS algorithm in a system identification setup. Load the impulse responses lpir.mat, hpir.mat, and bpir.mat and use these impulse responses as H and identify the impulse responses. Use the code `ex4_2_7` as template. Consider the residuals and the convergence of the adaptive filter, both in time domain and frequency domain. Comment on the results.

4.3 Adaptive filtering on audio

This exercise will demonstrate the application of an adaptive Wiener filter on real data, where there is piece of audio contaminated by noise, and then the filter is used to suppress the noise. It is a repetition from the audio exercise last week, but now with an adaptive filter.

Exercise 4.3.1

Inspect and use the code for `ex4_3_1`. Use the LMS algorithm in the noise reduction setup and experiment with different filter lengths (the filter order) and step sizes. Is the result satisfactory (Listen to the sounds)? Why/why not?

HINTS

Exercise 4.1.1

The formulas (4.5)–(4.6) and (4.43) from the ML book can be of help.

Exercise 4.1.3

The formula (4.9) is the formula to use.

Exercise 4.2.2

First replace the convolution operation with a sum. Secondly, the h_i can be taken out of expectation term, since the linear filter weights are considered as deterministic. Also remember u_n is a white noise sequence.

Exercise 4.2.3

The input sequence is u_n, and the desired signal is $\mathrm{d}_n = \mathrm{y}_n + \eta_n$, where y_n is the output of the unknown system (the plant), which is $\mathrm{y}_n = H * \mathbf{u}_n$.

To compute $\boldsymbol{p}$, we write out $r_{du}(k)$ completely, and solve using the normal equations.

Exercise 4.2.4

Solve by substitution and squaring e_n. Use the fact that convolution is distributive, i.e.

$$f * (g + h) = f * g + f * h$$