02471 Machine Learning for Signal Processing

Week 3: Stochastic Processes and Linear Filtering

This exercise is based on S. Theodoridis: Machine Learning, A Bayesian and Optimization Perspective 2nd edition, section(s) 4.1–4.3, 4.5–4.7.

The objective of the exercise is to get acquainted with stochastic processes, linear filtering and the interplay between linear regression from last week and classical DSP filtering.

Overview

The exercise have the following structure

3.1 will derive the correlation functions for an auto-regressive process (AR). We will use this result to compare a theoretical expression to one that is estimated from data and inspect how they differ.

3.2 will now consider a signal that is modeled by an AR process, contaminated with noise, and use a linear filter to suppress the noise. We will use the Wiener filter solution (which is the linear filter that minimizes the mean squared error), and compare the theoretical optimal filter to a filter estimated from real samples.

3.3 will consider a real-world example of filtering on an audio signal, which can be approximated as an AR process. The audio is contaminated with noise, and a Wiener filter is then used to suppress the noise.

The exercises should be completed in order, but, if you would like to see the Wiener filter in action before deriving the results, consider starting with 3.3.

Notation

- There are two traditions of notation w.r.t. signals. In DSP literature, a signal is often denoted as $x(n)$ with n being the time index. In ML literature, the signal is often denoted x_n if it is an observed entity, and x_n or X_n if it is a random variable. In this exercise, we will use the ML 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 3.x.y is in the file 3_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.

3.1 Correlation functions

The correlation between two signals measures the similarity between the signals and is widely used in signal processing applications. This exercise will introduce you to deriving correlation functions for stochastic signals. The correlation functions are required for the autocorrelation matrix and cross-correlation vector for the linear filter. We will also demonstrate correlation functions experimentally.

Make sure to read section 2.4.2–2.4.4 in the book before carrying out this exercise.

Exercise 3.1.1

Consider the signal $\mathrm{u}_n = c_1\mathrm{x}_n + c_2\mathrm{y}_{n-d}$, where c_1, c_2 are constants and $d > 0$ is a delay.

By using the definition of autocorrelation $r_u(k) = \mathbb{E}[\mathrm{u}_n\mathrm{u}_{n-k}]$ and cross-correlation $r_{uv}(k) = \mathbb{E}[\mathrm{u}_n\mathrm{v}_{n-k}]$, express $r_u(k)$ in terms of $r_x(k)$, $r_y(k)$, $r_{xy}(k)$, and the constants c_1, c_2 and d.

Exercise 3.1.2

Consider now two wide-sense stationary (WSS) random signals, x_n, y_n for which N_x and N_y samples have been recorded.

Complete the code in function `crosscor(x,y,k)` which estimates the cross-correlation function $r_{xy}(k)$ for $k = -K, -K+1, \cdots, 0, \cdots, K-1, K$, where $0 \leq K \leq \min(N_x, N_y) - 1$ using the biased, but asymptotically unbiased (which means the estimate is unbiased as $N \to \infty$) estimate:

$$\hat{r}_{xy}(k) = \frac{1}{N}\sum_{n=k}^{N-1} x_n y_{n-k}, \quad \text{for } k = 0, 1, \cdots, N-1$$

That the estimator is biased can be shown by evaluating $\mathbb{E}[\hat{r}_{xy}(k)] = \big(1 - \frac{k}{N}\big)r_{xy}(k)$ (not part of the curriculum).

Exercise 3.1.3

To verify the found analytical expressions, we generate some signals.

Inspect and run the code for this exercise (`ex3_1_3`). The code generates u_n, x_n and y_n according to the previous exercise, and plots the estimated autocorrelation function $r_u(k)$. Compare with a plot generated by using the expression found with estimates for the correlation functions in exercise 3.1.1.

Exercise 3.1.4 (This is a difficult exercise)

A great deal of signals can be adequately modeled as autoregressive (AR) processes, such as audio signals, and general time-series forecasting. We will consider the simplest case, a 1st order AR process, often denoted as AR(1), where the number refers to the amount of previous outputs remembered. There are two ways to define the AR(1) process, and it simply depends on whether the noise term (η_n) is isolated on the r.h.s, or the signal value u_n is isolated on the l.h.s. The two ways to define the AR process are:

$$\begin{aligned} u_n + a' u_{n-1} = \eta_n \qquad &\Leftrightarrow \\ u_n = a u_{n-1} + \eta_n \end{aligned}$$

where a is a real constant ($a \in \mathbb{R}, |a| < 1, a = -a'$), and η_n is a white noise sequence with variance σ_v^2. In this exercise, we consider the form $u_n = a u_{n-1} + \eta_n$.

According to the ML book, the analytical expression for $r_u(k)$ is

$$r_u(k) = \frac{a^{|k|}}{1 - a^2} r_\eta(0)$$

Prove the relation by inserting in the expression for u_n into $r_u(k) = \mathbb{E}[u_n u_{n-k}]$.

Exercise 3.1.5

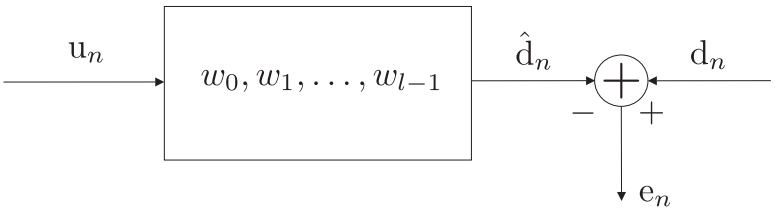
The corresponding code (`ex3_1_5`) generates signals u_n of lengths $N = 30$ and $N = 1000$, $k = N - 1$, with parameters $a = 0.1$ and $a = 0.9$. Compute and plot the estimated autocorrelation function in all 4 cases and compare with the theoretical autocorrelation function.

Exercise 3.1.6

As a final step, we will demonstrate the application of cross correlation functions on real data. We use the daily new confirmed the cases of COVID-19 in Denmark and cross-correlate with the cases on newly hospitalized patients. Use the associated code (`ex3_1_6`) to load the data, and modify the script to determine the most significant lags. What does this lag imply?

3.2 Wiener filter

Before completing this exercise, be sure to read section 4.5 in the book. Here the linear filter is depicted as



We will derive the expressions required to apply a Wiener filter and establish the theoretical performance of the filter. To this end, we will consider the example signal

$$u_n = s_n + \epsilon_n$$

where s_n is the source signal of interest and is modeled as an AR(1) process, $\mathrm{s}_n = a\mathrm{s}_{n-1} + \eta_n$, and ϵ_n is Gaussian white noise. The goal is to retrieve the clean signal s_n from the noisy signal u_n. Thus the desired signal is $\mathrm{d}_n = \mathrm{s}_n$. We will design a filter of length l with impulse response $\boldsymbol{w} = [w_0\, w_1\, \cdots\, w_{l-1}]^T$, that estimates s_n. To apply the linear filter, we require the two expressions: $r_u(k)$ and $r_{du}(k)$.

Exercise 3.2.1

Determine the analytical expression for $r_u(k)$.

Exercise 3.2.2

Determine the analytical expression for $r_{du}(k)$.

Exercise 3.2.3

Write the FIR Wiener filter solution, with filter length $l = 3$, on the form $(\Sigma_s + \Sigma_\epsilon)\boldsymbol{w} = \boldsymbol{r}_{du}$.

Exercise 3.2.4

Consider the case where $\sigma_\epsilon^2 \to 0$. What kind of filtering task are we then solving? (consider the filter figure). Determine the solution for $\boldsymbol{w}$ for $\sigma_\epsilon^2 \to 0$. Why is this a reasonable solution, i.e explain why the filter "chooses" to do what it does from the solution.

Exercise 3.2.5

Consider the case where $\sigma_\epsilon^2 \gg \sigma_\eta^2$. What kind of filtering task are we then solving? (consider the filter figure). Determine the solution for $\boldsymbol{w}$ when $\sigma_\epsilon^2 \gg \sigma_\eta^2$, and explain why this solution is a reasonable filtering solution.

Exercise 3.2.6

Now we consider a practical example with following coefficients: $l = 3$, $a = 0.6$, $\sigma_\eta^2 = 0.64$, and $\sigma_\epsilon^2 = 1$. The code snippets for this exercise (`ex3_2_6`) carries out the calculations for the filter coefficients. Inspect the code and verify the code aligns with your theoretical derived expressions.

Exercise 3.2.7

The error of the wiener filter, when using the optimal coefficients, is denoted as the *minimum mean squared error*, and is, for filter length l denoted MMSE_l. The expression is written as

$$\mathrm{MMSE}_l = \sigma_d^2 - \boldsymbol{r}_{du}^T \boldsymbol{\theta}_*$$

where $\boldsymbol{r}_{dx}$ is the cross-correlation vector between d_n and u_n, and $\boldsymbol{\theta}_*$ is the optimal solution, i.e. $\boldsymbol{\theta}_* = \boldsymbol{w}$. Implement the calculation of MMSE_l in the code (`ex3_2_7`).

Exercise 3.2.8

We will now estimate correlation functions from data which enables us to create the "plug-in" filter solution (when the correlation functions are estimated from data, we call the solution "plug-in"). Inspect and use the code (`ex3_2_8`) to generate data and obtain $r_u(k)$ and $r_{du}(k)$ for $k = 0, \pm 1, \pm 2$, and obtain the plug-in filter solution $\boldsymbol{w} = \hat{\Sigma}_u^{-1}\hat{\boldsymbol{r}}_{du}$ by substituting the true correlations functions with estimates. $\hat{\Sigma}_u$ is the estimated input correlation matrix and $\hat{\boldsymbol{r}}_{du}$ is the estimated cross-correlation vector (ref p. 136 in the book).

Exercise 3.2.9

What is the difference between the optimal (theoretical) filter solution and the plug-in filter. Compare the filters e.g., by evaluating the mean square error.

3.3 Noise removal using the Wiener filter

This exercise will demonstrate the application of a Wiener filter on real data, where there is an audio piece contaminated by noise, and then the Wiener filter is used to suppress the noise.

Exercise 3.3.1

Load the clean speech and noisy speech and listen to the sounds.

Exercise 3.3.2

Use the Wiener filter in the noise reduction setup, where the clean speech signal is used as desired signal, and conduct a rudimentary time-frequency analysis by inspecting the spectrograms of the signals. Try and listen to the result.

Exercise 3.3.3

Experiment with different filter lengths (the filter order). Is the result satisfactory? Why/Why not?

HINTS

Exercise 3.1.1

We have the definition, $r_u(k) = E[\mathrm{u}_n \mathrm{u}_{n-k}]$, solve by substitution, and take advantage of wide-sense stationarity (WSS). That means e.g. when $r_u(k)$ is a WSS process, the correlation function is a function only of the time-differences, so time indices can be shifted ie. $r_u(k) = E[\mathrm{u}_n \mathrm{u}_{n-k}] = E[\mathrm{u}_{n-m} \mathrm{u}_{n-k-m}]$. The final result is $r_u(k) = c_1^2 r_x(k) + c_1 c_2 (r_{xy}(d+k) + r_{xy}(d-k)) + c_2^2 r_y(k)$.

Exercise 3.1.4

Consider only $k > 0$, and insert the signal in the expression: $r_u(k) = \mathbb{E}[\mathrm{u}_n \mathrm{u}_{n-k}] = \mathbb{E}[(a\mathrm{u}_{n-1} + \eta_n)\mathrm{u}_{n-k}]$. Isolate $r_u(k)$ to generate a recursive formula. Once the formula is created, consider $k = 0$ to create the final expression. The final result is $r_u(k) = \frac{a^{|k|}}{1-a^2} r_\eta(0)$.

Exercise 3.2.1

Insert into the expression for $r_u(k) = \mathbb{E}[\mathrm{u}_n \mathrm{u}_{n-k}]$ (note, $\mathrm{u}_n = \mathrm{s}_n + \eta_n$) to obtain the final result, $r_u(k) = r_s(k) + r_\epsilon(k)$.

Exercise 3.2.2

Insert into the expression for $r_{du}(k) = \mathbb{E}[\mathrm{d}_n \mathrm{u}_{n-k}]$ (note, $\mathrm{d}_n = \mathrm{s}_n$) to obtain the final result, $r_{du}(k) = r_s(k)$.

Exercise 3.2.3

First use the fact that $r_\epsilon(0) = \sigma_\epsilon^2$ and otherwise $r_\epsilon(k) = 0$, $k \neq 0$, since it is a white noise signal. Then write out the input covariance matrix fully, and plug in the solutions from exercise 3.2.1 and 3.2.2 to obtain the solution

$$\left(\begin{bmatrix} r_s(0) & r_s(1) & r_s(2) \\ r_s(1) & r_s(0) & r_s(1) \\ r_s(2) & r_s(1) & r_s(0) \end{bmatrix} + \begin{bmatrix} \sigma_\epsilon^2 & 0 & 0 \\ 0 & \sigma_\epsilon^2 & 0 \\ 0 & 0 & \sigma_\epsilon^2 \end{bmatrix} \right) \mathbf{h} = \begin{bmatrix} r_s(0) \\ r_s(1) \\ r_s(2) \end{bmatrix}$$