DTU

Linear Filtering

Stochastic Processes and Wiener Filter

Stochastic Processes I

- A sequence of random variables
- **Discrete time signal:** u_n, where $n \in \mathbb{Z}$ is time index
- **Continuous time signal:** $u(t)$

Stochastic Processes II

First-and second-order statistics

$$p(u_n, u_m, \ldots, u_r; n, m, \ldots, r) \tag{1}$$

$$\mu_n := \mathbb{E}[u_n] = \int_{-\infty}^{\infty} u_n p(u_n)\, du_n \tag{2}$$

$$\mathrm{cov}(n, m) := \mathbb{E}[(u_n - \mathbb{E}[u_n])(u_m - \mathbb{E}[u_m])] \tag{3}$$

$$r_u(n, m) := \mathbb{E}[u_n u_m] \tag{4}$$

$$r_{uv}(n, m) := \mathbb{E}[u_n v_m] \tag{5}$$

Stochastic Processes III

Stationarity

Strict-sense stationarity:

- $p(u_n, u_m, \ldots, u_r) = p(u_{n-k}, u_{m-k}, \ldots, u_{r-k})$

Wide-sense stationarity:

- Constant mean: $\mu_n = \mu$
- Autocorrelation/autocovariance only depends on the time lag:
  - $r_u(n, n-k) = r_u(k) = \mathbb{E}[u_n u_{n-k}]$

Stochastic Processes IV

Estimators

- Sample Mean estimator: $\hat{\mu}_N = \frac{1}{N}\sum_{n=1}^{N} u_n$
- Sample Auto-correlation Estimator:
  - Asymptotically unbiased:

$$\hat{r}_{uv}(k) = \frac{1}{N}\sum_{n=k}^{N-1} u_n v_{n-k}, \quad k = 0, 1, \ldots, N-1$$

  - Unbiased:

$$\hat{r}'_{uv}(k) = \frac{1}{N-k}\sum_{n=k}^{N-1} u_n v_{n-k}, \quad k = 0, 1, \ldots, N-1$$

Stochastic Processes V

Ergodicity

- Complete statistics can be computed from any realization of the stochastic process
- Second-order ergodic processes are sense stationary

Auto-regressive process $AR(l)$

$$u_n + a_1 u_{n-1} + \cdots + a_l u_{n-l} = \eta_n$$

- η_n is a white noise process with variance σ_η^2

The Wiener Filter I

Filtering as a learning problem

- Input: u_n (Stochastic process)
- Linear system: w_n
- Output: d_n (Desired signal)

The Wiener Filter is a linear filter that minimizes Mean Squared Error (MSE):

$$\hat{d}_n = \sum_{i=0}^{l-1} w_i u_{n-i} = \mathbf{w}^T \mathbf{u}_n$$

$$\hat{w} = \arg\min \mathbb{E}_{\mathbf{w}} \left[\left(d_n - \hat{d}_n \right)^2 \right]$$

The Wiener Filter II

- Linear Estimation: $y(n) := \hat{d}_n = \boldsymbol{\theta}^T \boldsymbol{x}_n$
- Cost function using MSE: $J(\boldsymbol{\theta}) = \mathbb{E}\left[(d_n - \hat{d}_n)^2\right]$
- The **normal equations** can be expressed in matrix form as follows:

$$\Sigma_x \theta_* = \mathbf{p}$$

where $\mathbf{p}$, denoted as $\mathbf{r}_{dx}$ in the book, is the vector of cross-correlations between the desired response and the input, given by:

$$\mathbf{p} = \begin{bmatrix} r_{dx}(0) \\ r_{dx}(1) \\ \vdots \\ r_{dx}(l-1) \end{bmatrix}$$

The Wiener Filter III

The matrix Σ_x, representing the autocorrelation matrix of the input, is structured as:

$$\Sigma_x = \begin{bmatrix} r_x(0) & r_x(1) & \cdots & r_x(l-1) \\ r_x(1) & r_x(0) & \cdots & r_x(l-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_x(l-1) & r_x(l-2) & \cdots & r_x(0) \end{bmatrix}$$

- Optimal Parameters: $\boldsymbol{\theta}_* = \Sigma_x^{-1} \cdot \boldsymbol{r}_{dx}$

Essential parts of Exercise 3.2: The Wiener Filter I

Input signal: $u_n = s_n + \epsilon_n$

- s_n is the source signal of interest modeled as an AR(1) process, $s_n = as_{n-1} + \eta_n$
- ϵ_n is white noise
- Desired signal $d_n = s_n$

Procedure

- Determine the analytical expressions for $r_u(k)$ and $r_{du}(k)$
- Normal equations
- Code example: Solve the equations to find optimal parameters

Essential parts of Exercise 3.2: The Wiener Filter II

Determine the analytical expression for $r_u(k)$

$$
\begin{aligned}
r_u(k) &= \mathbb{E}[u_n u_{n-k}] = \mathbb{E}[(s_n + \epsilon_n)(s_{n-k} + \epsilon_{n-k})] \\
&= \mathbb{E}[s_n s_{n-k}] + \mathbb{E}[s_n \epsilon_{n-k}] + \mathbb{E}[\epsilon_n s_{n-k}] + \mathbb{E}[\epsilon_n \epsilon_{n-k}]. \\
&= \mathbb{E}[s_n s_{n-k}] + \mathbb{E}[\epsilon_n \epsilon_{n-k}] = r_s(k) + r_\epsilon(k) \quad (s_n \text{ and } \epsilon_n \text{ are uncorrelated})
\end{aligned}
$$

$$
r_u(k) = \frac{a^{|k|}}{1 - a^2}\sigma_\eta^2 + \delta(k)\sigma_w^2.
$$

Essential parts of Exercise 3.2: The Wiener Filter III

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

$$
\begin{aligned}
r_{du}(k) &= \mathbb{E}[d_n u_{n-k}] = \mathbb{E}[s_n(s_{n-k} + \epsilon_{n-k})] \\
&= \mathbb{E}[s_n s_{n-k}] + \mathbb{E}[s_n \epsilon_{n-k}] \\
&= r_s(k).
\end{aligned}
$$

Essential parts of Exercise 3.2: The Wiener Filter IV

Normal equations

The Wiener filter solution, with filter length $l = 3$, is given by:
$(\Sigma_s + \Sigma_\epsilon)\mathbf{w} = \mathbf{r}_{du}$

$$\begin{pmatrix} r_s(0) & r_s(1) & r_s(2) \\ r_s(1) & r_s(0) & r_s(1) \\ r_s(2) & r_s(1) & r_s(0) \end{pmatrix} + \begin{pmatrix} \sigma_\epsilon^2 & 0 & 0 \\ 0 & \sigma_\epsilon^2 & 0 \\ 0 & 0 & \sigma_\epsilon^2 \end{pmatrix} \mathbf{w} = \begin{pmatrix} r_s(0) \\ r_s(1) \\ r_s(2) \end{pmatrix}$$

Essential parts of Exercise 3.2: The Wiener Filter V

Find optimal parameters

```
a = 0.6
sigma_eta = 0.8
sigma_epsilon = 1
sigma_s = np.sqrt(sigma_eta**2/(1-a**2))
Gamma_ss = np.array([[1, a, a**2], [a, 1, a], [a**2, a, 1]]) *
sigma_s**2
Gamma_epsilon = np.eye(3) * np.square(sigma_epsilon)
Gamma_l = Gamma_ss+Gamma_epsilon
gamma_ss = np.array([[1], [a], [a**2]]) * sigma_s**2
w = np.linalg.solve(Gamma_l, gamma_ss)
print(w)
[[0.44512195][0.15][0.05487805]]
```