Stoch Project 5 - Wiener Filters
May 8, 2020
Sophie Jaw and Rebecca Bartenberg

$$\begin{tikzcd} & & d[n] \arrow[d] & & \\ s[n] \arrow[r] & \boxed{c[n]} \arrow[r] & \oplus \arrow[r, "r[n]"] & \boxed{h[n]} \arrow[r] & \hat{s}[n] \end{tikzcd}$$

$$R_{rr}[n] = R_{ss}[n] * R_{cc}[n] + R_{dd}[n]$$

w/ $R_{ss}[n] = \delta[n]$

$$R_{cc}[n] = \sum_{k=0}^{2} c[k+n]\,c[k]$$
$$= c[0+n]c[0] + c[1+n]c[1] + c[2+n]c[2]$$
$$= c[n] + 0.2c[n+1] + 0.4c[n+2]$$

$$R_{dd}[n] = \sigma^2 \delta[n]$$

So

$$R_{rr}[n] = \delta[n] * (c[n] + 0.2c[n+1] + 0.4c[n+2]) + \sigma^2\delta[n]$$

(anything convolved w/ delta is itself)

$$= c[n] + 0.2c[n+1] + 0.4c[n+2] + \sigma^2\delta[n]$$

$R_{rr}[-3] = 0 = R_{rr}[3]$
$R_{rr}[-2] = 0.4 = R_{rr}[2]$
$R_{rr}[-1] = 0.28 = R_{rr}[1]$
$R_{rr}[0] = 1.2 + \sigma^2$

$$[R_{rr}]_{4\times4} = \begin{bmatrix} 1.2+\sigma^2 & 0.28 & 0.4 & 0 \\ 0.28 & 1.2+\sigma^2 & 0.28 & 0.4 \\ 0.4 & 0.28 & 1.2+\sigma^2 & 0.28 \\ 0 & 0.4 & 0.28 & 1.2+\sigma^2 \end{bmatrix}$$

$$R_{sr}[n] = R_{ss}[n] * c[n]$$
$$= \sum_{k=0}^{2} \delta[k+n]\,c[k]$$
$$= \delta[0+n]c[0] + \delta[1+n]c[1] + \delta[2+n]c[2]$$
$$= \delta[n] + 0.2\delta[n+1] + 0.4\delta[n+2]$$

$R_{sr}[0] = 1$
$R_{sr}[1] = 0$
$R_{sr}[2] = 0$
$R_{sr}[3] = 0$

So, for $\sigma^2 = 1$,

$$\begin{bmatrix} 2.2 & 0.28 & 0.4 & 0 \\ 0.28 & 2.2 & 0.28 & 0.4 \\ 0.4 & 0.28 & 2.2 & 0.28 \\ 0 & 0.4 & 0.28 & 2.2 \end{bmatrix} \begin{bmatrix} h[0] \\ h[1] \\ h[2] \\ h[3] \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Solved with MATLAB:

$h[0] = 0.48$, $h[1] = -0.05$, $h[2] = -0.08$, $h[3] = 0.02$

$h = [0.48 \;\; -0.05 \;\; -0.08 \;\; 0.02]$

Stoch Project 5 --- Wiener Filters

Rebecca Gartenberg & Sophie Jaro

Contents

Explanation of code

```
% A signal s[n] was filtered with FIR filter c[n] and Gaussian white noise
% d[n] was added to the filtered signal. The purpose of this project was to
% obtain an estimate, s_hat[n], of the original signal by filtering the
% observed signal r[n] with a Wiener filter h[n].

% In this project we derived Wiener filters of lengths 4, 6 and 10.
% To derive the Wiener filter, we used the autocorrelation Rrr and cross
% correlation Rsr to solve the normal equations for LMMSE estimation found in eqn 11.11.

% We implemented the Wiener filters to find the estimated signal. Then we
% computed mean squared error (MSE) for each filter length. For each filter
% length, the estimation process was iterated 1000 times. The MSE for each
% filter length was the average of the 1000 iterations. This was repeated
% for 4 different variances.

% The calculated MSE for each of the 12 filters can be seen in the table.
```

```
clc
clear all

noise_var_vec = [0.1, 0.5, 1, 4]; % variance of the noise
N_vec = [4, 6, 10]; % Wiener filter lengths

% loop through variance
for n = 1:length(noise_var_vec)
    noise_var = noise_var_vec(n);
    %loop through filter length
    for j = 1:length(N_vec)
        N = N_vec(j);
        % repeated iterations of estimate
        for k = 1:1000
            % Calculate Rrr and Rsr for the normal equations
            for i = 1:N
                m = i-1;
                Rrr_vec(i,1) = c(m) + 0.2*c(m+1) + 0.4*c(m+2) + noise_var*delta(m);
                Rsr(i,1) = delta(m)*c(0) + delta(m+1)*c(1) + delta(m+2)*c(2);
            end

            Rrr = toeplitz(Rrr_vec); % create normal equation matrix (11.11)
            h = Rrr\Rsr; % solve for Wiener filter

            chan = [1 0.2 0.4]; % channel filter c[n]
            d = ((noise_var)^0.5)*randn(1,1); % Gaussian white noise d[n]

            % original signal s[n]
            s = randi([0,1],1,10000); % 10000 i.i.d processes which takes value +/-1 with equal probability
            s(s==0)=-1;

            % Filter signal with channel
            y = filter(chan, 1, s);

            % Add Noise to filtered signal
            r = y + d;

            % Estimate signal s[n] by filtering with Wiener filter
            s_hat = filter(h, 1, r);

            % MSE
            MSE_vec(j,k) = mean((s-s_hat).^2);
        end
        MSE(j, n) = mean(MSE_vec(j,:));
    end
end
```

Table of MSE for several filter lengths and noise variance

```
MSE = MSE.'; % transposed for table formatting
T = array2table(round(MSE, 3), 'RowNames', {'sigma^2 = 0.1', 'sigma^2 = 0.5', 'sigma^2 = 1', 'sigma^2 = 4'}, 'VariableNames', {'filt_length4','filt_length6','filt_l
```

```
T =
```

4×3 table

	filt_length4	filt_length6	filt_length10
sigma^2 = 0.1	0.057	0.047	0.047
sigma^2 = 0.5	0.243	0.252	0.246
sigma^2 = 1	0.405	0.42	0.422
sigma^2 = 4	0.771	0.772	0.77

Functions

```
% delta function that takes value of 1 at 0 and 0 everywehere else
function x = delta(m)
if m == 0
    x = 1;
else
    x = 0;
end
end

% channel filter
function x = c(m)
chan = [1, 0.2, 0.4];
if m == 0 || m == 1 || m == 2
    x = chan(m+1);
else
    x = 0;
end
end
```

Published with MATLAB® R2019a