Digital Signal Processing 2/ Advanced Digital Signal Processing Lecture 13, Matched Filter, Prediction

Gerald Schuller, TU Ilmenau

Matched Filters

Remember the goal of a matched filter $h_M(n)$:

$$y(n)*h_M(n)\rightarrow x(n+n_d)*h_M(n)$$

with some signal delay n_d (no signal fidelity, just high SNR for detection), where $y(n)=x(n+n_d)+v(n)$ is our (delayed) signal with additive noise $v(n)$.

Application examples are in communications, where you would like to detect a 0 or 1,for instance in CDMA, where each user gets a unique pseudo-random 1/0 sequence (so-called chip-sequences) to represent 0 or 1, in which different users and signals are separated using matched filters. Another example is for detecting **known** signals or patterns, like object or face recognition in images. In general we would like to detect **deterministic signals** $x(n)$.

This means our goal is to **maximize the SNR** at the moment of detection, with our original signal $x(n)$ and noise $v(n)$,

$$SNR=\frac{|x(n)*h_M(n)|^2}{E(|v(n)*h_M(n)|^2)}$$

We would like to maximize this SNR at the time of detection, using $h_M(n)$. To do that, first we assume $v(n)$ to be independent white noise. Then the denominator of the SNR expression is just a scaled fixed power expression. Using our formulation of a matrix $\boldsymbol{V}$ for the noise signal and the vector $\boldsymbol{h_M}$ for our filter (again, it contains the time reversed impulse response), we obtain

$$E(|v(n)*h_M(n)|^2)=E(|\boldsymbol{V}\cdot\boldsymbol{h_M}|^2)=E(\boldsymbol{h_M^T}\cdot\boldsymbol{V^T}\cdot\boldsymbol{V}\cdot\boldsymbol{h_M})=$$
$$=\boldsymbol{h_M^T}\cdot E(\boldsymbol{V^T}\cdot\boldsymbol{V})\cdot\boldsymbol{h_M}=\boldsymbol{h_M^T}\cdot\sigma_v^2\cdot\boldsymbol{I}\cdot\boldsymbol{h_M}=$$
$$=\sigma_v^2\cdot\boldsymbol{h_M^T}\cdot\boldsymbol{h_M}$$

(Remember: the autocorrelation function of noise is a delta function, since noise samples are uncorrelated to all their neighbour samples, there are only correlated with themselves, and the correlation with itself is simply the noise power σ^2. The autocorrelation matrix $\boldsymbol{V^T}\cdot\boldsymbol{V}$ hence has all zero entries, except on the diagonal, where it is the noise power, hence noise power time the identity matrix $\sigma^2\cdot\boldsymbol{I}$)

The last expression is simply the squared norm (the sum of the squares of its coefficients) of our vector of our filter coefficients $\boldsymbol{h_M}$

multiplied with the noise power σ_v^2 .

Keeping the above **norm** of our filter vector **constant**, we only need to **maximize the numerator** of our SNR fraction to maximize the SNR. We apply the Cauchy-Schwartz inequality (see e.g. http://en.wikipedia.org/wiki/Cauchy%E2%80%93Schwarz_inequality), which says, for 2 (column) vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ and their scalar product we get

$$\boldsymbol{a}^T \cdot \boldsymbol{b} \leq \sqrt{\boldsymbol{a}^t \cdot \boldsymbol{a}} \cdot \sqrt{\boldsymbol{b}^t \cdot \boldsymbol{b}}$$

This is also written with the norm $\|\boldsymbol{a}\|$ and $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$ scalar product as

$$|\langle \boldsymbol{a}, \boldsymbol{b} \rangle| \leq \|\boldsymbol{a}\| \cdot \|\boldsymbol{b}\|$$

We obtain equality if both vectors are co-linear, $\boldsymbol{b} = k \cdot \boldsymbol{a}$, with some scalar k . This tells us how to solve the maximization task.

We get for our **numerator,**

$$|x(n) * h_M(n)|^2 = |\boldsymbol{x}(n) \cdot \boldsymbol{h}_M|^2$$

where we rewrote our convolution, analog to our matrix formulation, as a scalar vector multiplication, with $\boldsymbol{h}_M$ as our **vector** of the **time-reversed** matched filter impulse response, and now with only one row of the signal matrix at a time, for only one convolution sample at a time,

$\boldsymbol{x}(n) = [x(n), x(n+1), \ldots, x(n+L-1)]$

where L is the size of our filter vector.
Here we can now apply Cauchy-Schwartz,

$$|\boldsymbol{x}(n)\cdot\boldsymbol{h}_{\boldsymbol{M}}|^2\leq\|\boldsymbol{x}(n)\|^2\cdot\|\boldsymbol{h}_{\boldsymbol{M}}\|^2$$

and we get the equality (the maximum) if we set

$$\boldsymbol{h}_{\boldsymbol{M}}=k\cdot\boldsymbol{x}(n)$$

where we can choose the factor as $k=1$.
Since we have this inequality for all time steps n of our convolution, we choose the one where the vector $\boldsymbol{x}(n)$ has the maximum energy. This is where we **capture the entire, non-zero, wave form** of our signal $\boldsymbol{x}(n)$ with our filter.
Since our filter vector $\boldsymbol{h}_{\boldsymbol{M}}$ contains the time-reversed impulse response, we obtain the entire **time reversed signal** as our **matched filter**:

$$h_M(n)=x(L-1-n)$$

assuming our signal to detect is located between $0\leq n< L$.
Since we have a convolution of the signal with its time reverse version, we get a convolution length of $2\mathrm{L}-1$ samples, with its maximum at the center, when both waveforms completely overlap, after L samples, which is exactly the signal length. Hence we get the **detection** of our signal after we completely received it, at the **end of our signal**.
Observe: Since we convolve the signal with the time-reversed version of our pattern to be

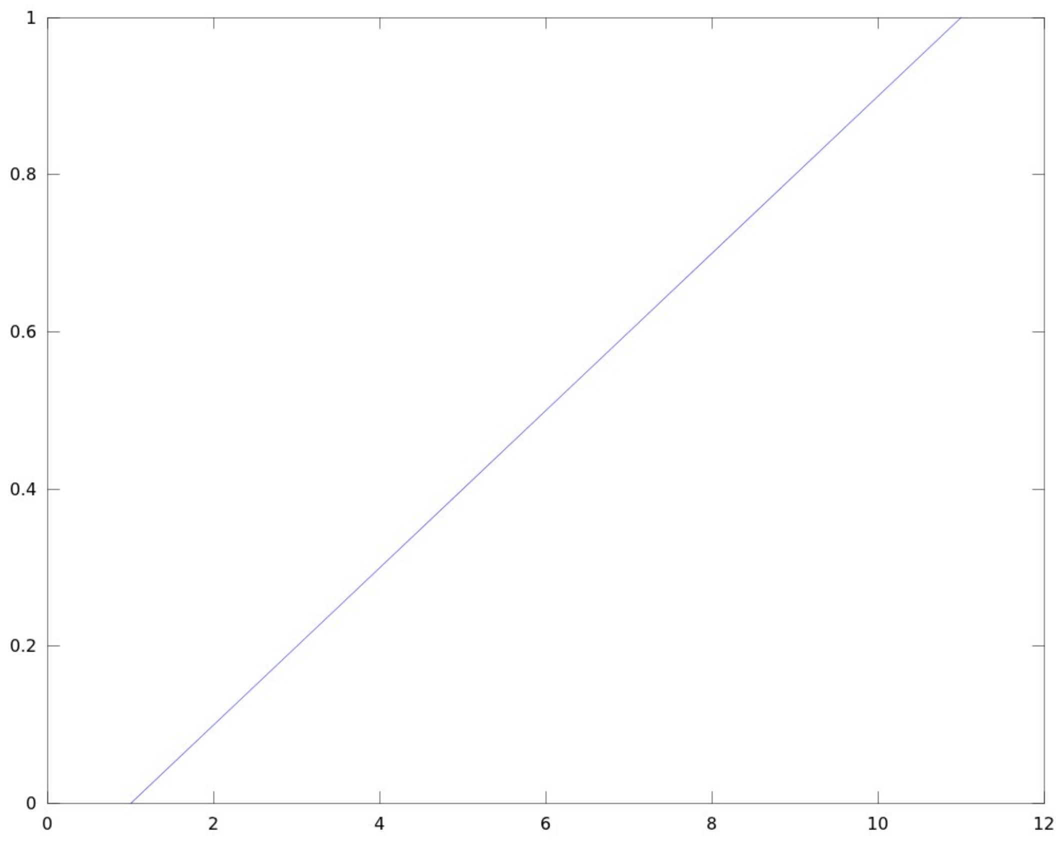
detected, this becomes identical to computing the correlation of the signal with the pattern to be detected.
For the continuous-time version see, for instance:
http://www.ece.gatech.edu/research/labs/sarl/tutorials/ECE4606/14-MatchedFilter.pdf

In Conclusion: The matched filter has the shape of the time reversed signal to be searched for.

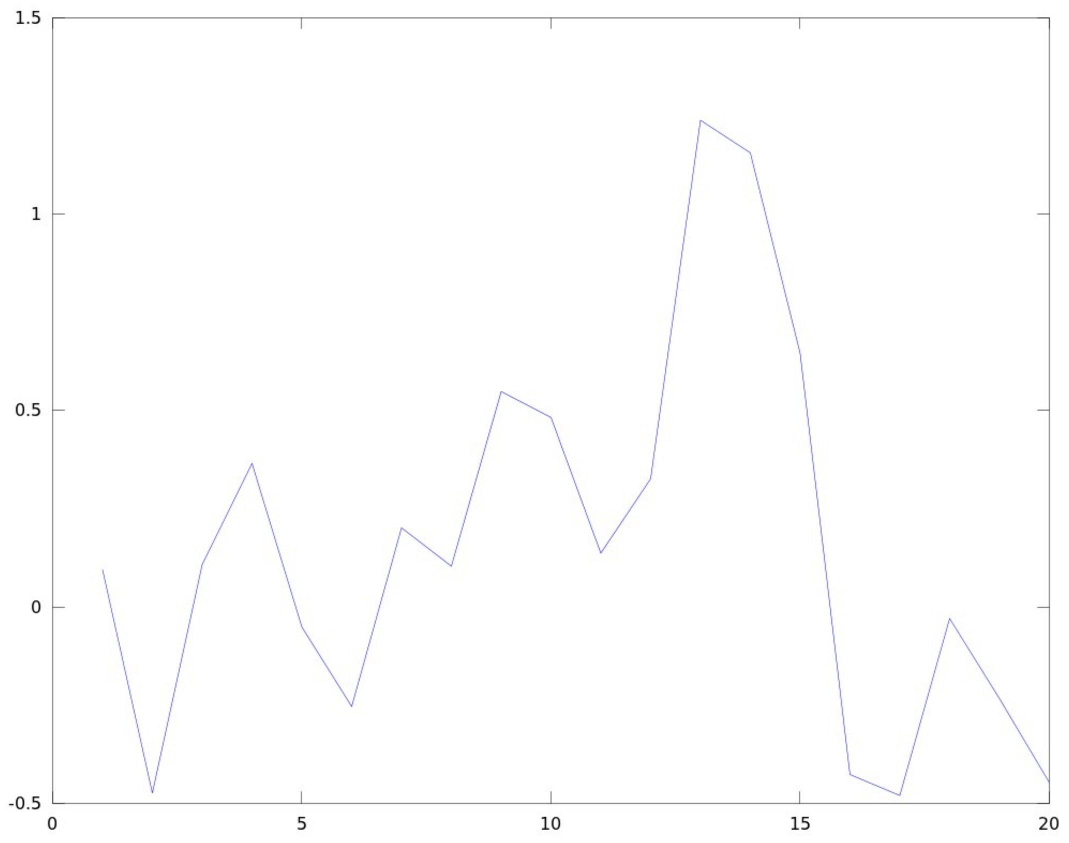
Matlab/Octave Example

Construct a signal sig (length 11):

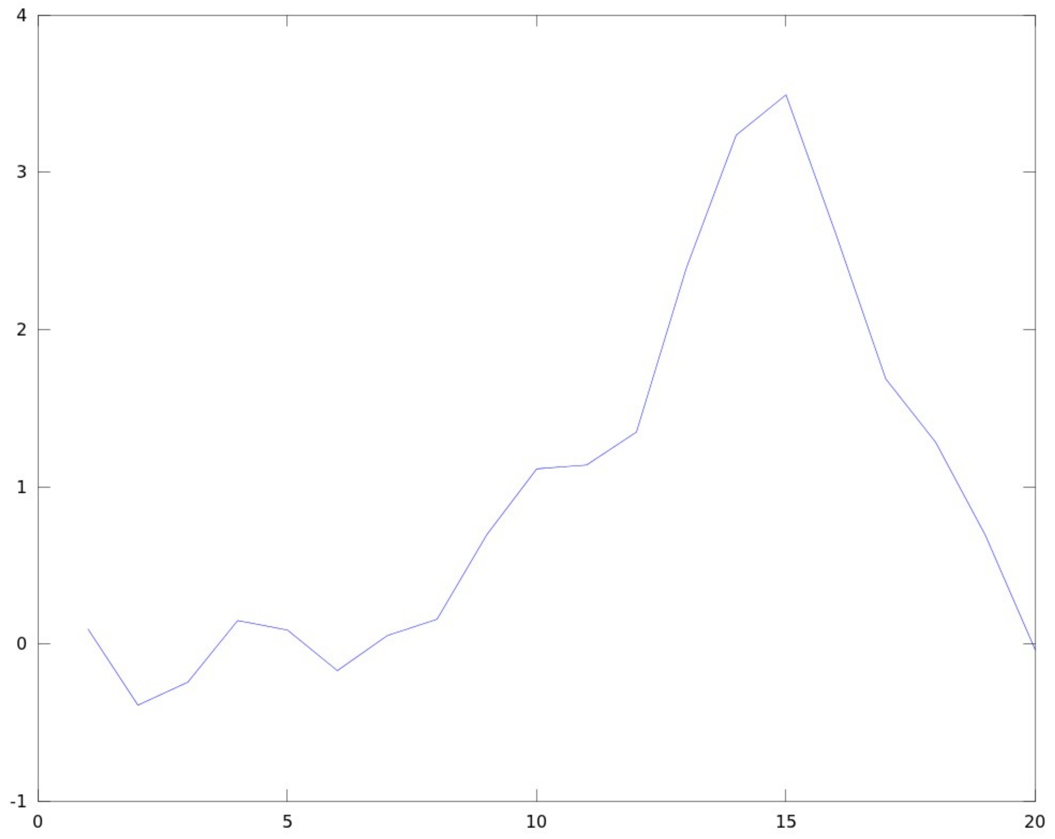
```
sig=0:0.1:1
%sig =
%Columns 1 through 7:
%0.00000 0.10000 0.20000 0.30000 0.40000 0.50000 0.60000
%Columns 8 through 11:
%0.70000 0.80000 0.90000 1.00000
plot(sig)
```



```
signoise=rand(1,20)-0.5+ [zeros(1,4),sig,zeros(1,5)];
plot(signoise)
```



```
h=fliplr(sig);
signoisemf=filter(h,1,signoise);
plot(signoisemf)
```



This is now the output of our matched filter. We can see that we have a maximum at time 15, which signals the end of our detected signal. Hence we know that the signal started at 15-L(length of the filter)=15-11=4, which was indeed the case since we added 4 zeros in the beginning.
The matched filtering process can also be viewed as computing the correlation function of the noisy signal with the original signal.

Prediction

Prediction can be seen as a special case of a Wiener filter, where the noise of our signal corresponds to a shift of our signal into the past. Our goal is to make a “good” estimation of the present sample of our signal, based on **past signal samples.** “Good” here means, again in a mean squared error sense.
Basically we can now take our Wiener Filter formulation, and specialize it to this case.
Looking at our matrix formulation, we get

$$\begin{bmatrix} 0 & x(0) \\ x(0) & x(1) \\ x(1) & x(2) \\ \vdots & \vdots \end{bmatrix} \cdot \begin{bmatrix} h(2) \\ h(1) \end{bmatrix} = \begin{bmatrix} x(1) \\ x(2) \\ x(3) \\ \vdots \end{bmatrix}$$

This means, the input to our filter is always starting at 1 sample in the past, going down further into the past, and its goal is to estimate or “predict” the next coming sample. Basically this means that instead of additive white noise, our **distortion** is now a **delay** operator (fortunately, this is a linear operator).
Now we can again use our approach with pseudo-inverses to obtain our mean-squared error solution,

$$\boldsymbol{h} = \left(\boldsymbol{A}^T \cdot \boldsymbol{A}\right)^{-1} \boldsymbol{A}^T \cdot \boldsymbol{x}$$

with the matrix $\boldsymbol{A}$ now defined as our above matrix. This now also leads to a statistical description, with $\boldsymbol{A}^T \cdot \boldsymbol{A}$ converging to

$$\boldsymbol{A}^T \cdot \boldsymbol{A} \rightarrow \boldsymbol{R}_{xx} = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) & r_{xx}(2) & \dots \\ r_{xx}(1) & r_{xx}(0) & r_{xx}(1) & \dots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

This is plausible, because now y(n) is just the delayed signal, and the auto-correlation function of the delayed signal is identical to the auto-correlation function of the original function.

Next we need the cross-correlation $\boldsymbol{A}^T \cdot \boldsymbol{x}$. Since we now just have this 1 sample delay as our target vector, this converges to the auto-correlation vector, starting at **lag 1**,

$$\boldsymbol{A}^T \cdot \boldsymbol{x} \rightarrow \boldsymbol{r}_{xx} = \begin{bmatrix} r_{xx}(1) \\ r_{xx}(2) \\ \vdots \end{bmatrix}$$

So together we get the solution for our prediction filter as

$$\boldsymbol{h} = (\boldsymbol{R}_{xx})^{-1} \boldsymbol{r}_{xx}$$