Maths for AI

Lecture 3: Convolution

Matthew Roughan
<matthew.roughan@adelaide.edu.au>

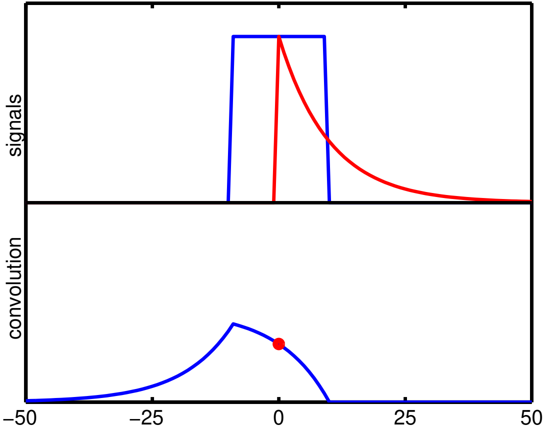
School of Mathematical Sciences and AIML,
University of Adelaide

March 15, 2022

Convolutions

What is a convolution?

$$f(t) * g(t) = [f * g](t) = \int_{-\infty}^{\infty} f(u)\, g(t-u)\, du$$



Fourier Transform of a Convolution

$$\boxed{\mathcal{F}\left\{f_1(t) * f_2(t)\right\} \rightarrow F_1(s)F_2(s)}$$

$$\begin{aligned}
\mathcal{F}\left\{f(t) * g(t)\right\} &= \mathcal{F}\left\{\int_{-\infty}^{\infty} f(u)\, g(t-u)\, du\right\} \\
&= \mathcal{F}\left\{\left[\int_{-\infty}^{\infty} f(u)\, g(t-u)\, du\right]\right\} \\
&= \int_{-\infty}^{\infty} f(u) \int_{-\infty}^{\infty} g(t-u) e^{-i2\pi st}\, dt\, du \\
&= \int_{-\infty}^{\infty} f(u) G(s) e^{-i2\pi su}\, du \\
&= G(s) \int_{-\infty}^{\infty} f(u) e^{-i2\pi su}\, du \\
&= F(s)G(s)
\end{aligned}$$

Convolution example

Convolution of two rectangular pulses $r(t)$ where

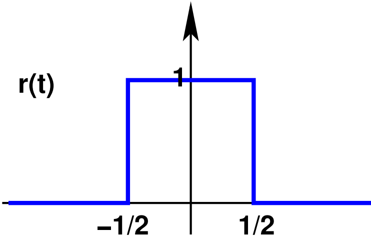
$r(t) = u(t + 1/2) - u(t - 1/2)$, and $u(t) = \begin{cases} 0, & t < 0 \\ 1, & t \geq 0 \end{cases}$

$$
\begin{aligned}
r(t) * r(t) &= \int_{-\infty}^{\infty} r(s)\, r(t - s)\, ds \\
&= \begin{cases} 0, & \text{if } t < -1 \\ \displaystyle\int_{-1/2}^{1/2+t} r(s)\, r(t - s)\, ds, & \text{if } -1 \leq t \leq 0 \\ \displaystyle\int_{t-1/2}^{1/2} r(s)\, r(t - s)\, ds, & \text{if } 0 \leq t \leq 1 \\ 0, & \text{if } t > -1 \end{cases}
\end{aligned}
$$

Convolution example

For $-1 \leq t \leq 0$, the convolution $r(t) * r(t)$ is

$$\begin{aligned} r(t) * r(t) &= \int_{-1/2}^{1/2+t} r(s)\, r(t-s)\, ds, \\ &= \int_{-1/2}^{1/2+t} 1\, ds, \\ &= [t]_{-1/2}^{1/2+t} \\ &= 1/2 + t - -1/2 \\ &= 1 + t, \end{aligned}$$



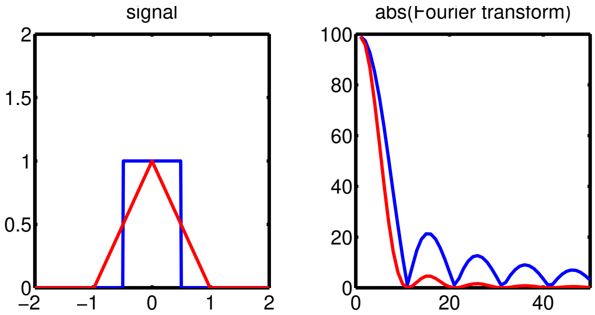
Similarly for $0 \leq t \leq 1$, we get $r(t) * r(t) = 1 - t$

Convolution example

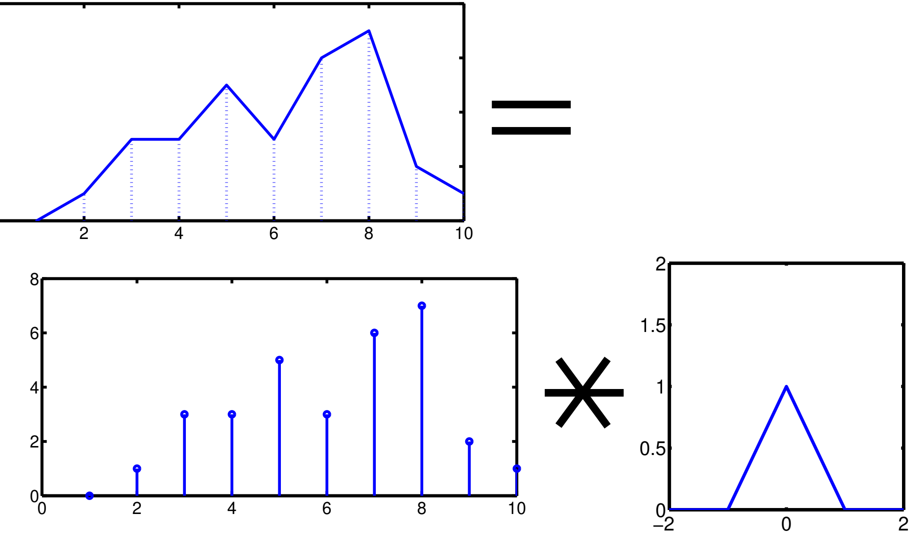
Result is a Triangular pulse

$$r(t) * r(t) \quad = \quad \begin{cases} 0, & \text{if } t < -1 \\ 1 + t, & \text{if } -1 \leq t \leq 0 \\ 1 - t, & \text{if } 0 \leq t \leq 1 \\ 0, & \text{if } t > -1 \end{cases}$$

$\mathcal{F}\{r(t)\} = \mathrm{sinc}(s)$ hence from the convolution theorem
$\mathcal{F}\{r(t) * r(t)\} = \mathrm{sinc}^2(s)$



Convolution example: interpolation



Convolution example: interpolation

Fourier transformation of a piecewise linear function

$$f(t) = \left[\sum_{i=1}^{n} f_i \delta(t - t_i) \right] * r(t) * r(t)$$

is

$$F(s) = \left[\sum_{i=1}^{n} f_i e^{-i2\pi s t_i} \right] \operatorname{sinc}^2(s)$$

Discrete Convolution

$$[x_1 * x_2](n) = \sum_{i=-\infty}^{\infty} x_1(i)x_2(n-i) = \sum_{i=-\infty}^{\infty} x_1(n-i)x_2(i)$$

Now remember the impact of convolutions in DFTs, e.g.

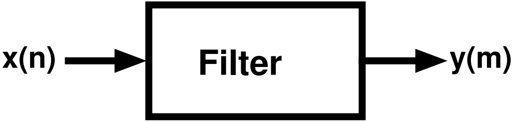
$$\mathcal{F}\{x_1 * x_2\} = X_1(k)X_2(k)$$

where $\mathcal{F}\{x_1(n)\} = X_1(k)$ and $\mathcal{F}\{x_2(n)\} = X_2(k)$.

Discrete convolutions have a special role in signal processing as *Linear Time-Invariant Filters*

Filters

A filter takes some input $x(n)$, and produces an output $y(n)$, which has been filtered to extract certain features (e.g. trend, seasonality, ...)



References:

- Brockwell and Davis, 1996
- Box and Jenkins, 1976
- Anderson and Moore, 1979

Possible filter properties

- **invertibility:** The mapping $x(t) \to y(t)$ must be 1:1, so that each input signal has a unique output signal (don't need to invert all possible outputs).
- **memory:** $y(t_0)$ depends on $x(t)$ for $t \neq t_0$.
- **causality:** $y(t_0)$ only depends on $x(t)$ for $t \leq t_0$.
- **stability:** Bounded Input Bounded Output (BIBO). If $|x(t)| \leq M$ for all t and some M, then $|y(t)| \leq R$ for all t and some R.
- **time invariance:** time shift doesn't matter, i.e. $x(t) \to y(t)$ implies $x(t - t_0) \to y(t - t_0)$.
- **linearity:** principle of superposition: $x_i \to y_i$, $i = 1, 2$ implies that for all $a_1, a_2 \in \mathbb{R}$, $a_1 x_1 + a_2 x_2 \to a_1 y_1 + a_2 y_2$.

Linear Filters

Response is linear in the input, e.g. given the filter,

$$\begin{aligned} \mathcal{L}x_1 &\rightarrow y_1 \\ \mathcal{L}x_2 &\rightarrow y_2 \end{aligned}$$

Then

$$\mathcal{L}ax_1 + bx_2 \rightarrow ay_1 + by_2$$

The output of linear filters can be written as a linear combination of the inputs.

$$y(m) = \sum_{i=-\infty}^{\infty} w(m,i)x(m-i)$$

Linear Time Invariant Filters

- time invariant filters don't change over time, so $w(m, i) = w(i)$

The output of linear filters can be written as a linear combination of the inputs.

$$y(m) = \sum_{i=-\infty}^{\infty} w(i)x(m-i)$$

Note that this is a discrete convolution!

Linear Time Invariant Causal Filters

- time invariant filters don't change over time, so $w(m, i) = w(i)$
- causal filters only depend on the past, so $w(-i) = 0$, for $i > 0$.

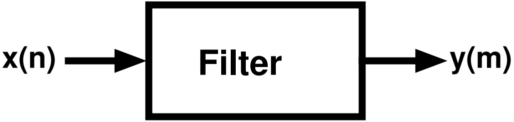
The output of linear filters can be written as a linear combination of the inputs.

$$y(m) = \sum_{i=0}^{\infty} w(i)x(m-i)$$

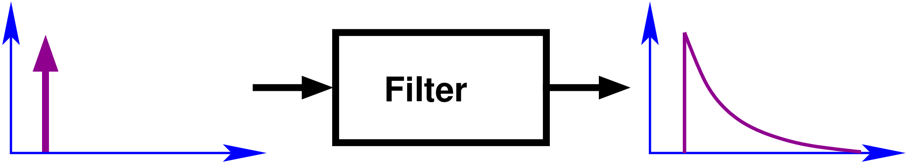
Note that this is also a discrete convolution!

Impulse response

Given a filter:



The impulse response is the output of the filter given an impulse as the input.



Impulse response

For a linear, time-invariant filter F, the impulse response is

$$I_F(m) = \sum_{i=-\infty}^{\infty} w(i)\delta_{mi} = w(m)$$

where δ_{nk} is the Kronecker delta, defined by

$$\delta_{nk} = \begin{cases} 1 & \text{if } n = k \\ 0 & \text{otherwise} \end{cases}$$

So a **linear time-invariant filter** can be completely characterized by its **impulse response**.

Impulse response

Note that any signal $x(n)$ can be written as a linear combination of impulses, e.g.

$$x(n) = \sum_{k=-\infty}^{\infty} \delta_{nk} x(k)$$

Given linearity of the filter, the output can be written as the same linear combination of the impulse responses, e.g.

$$\begin{aligned} y(m) &= \sum_{i=-\infty}^{\infty} w(i) \left[\sum_{k=-\infty}^{\infty} \delta_{m-i,k} x(k) \right] \\ &= \sum_{i=-\infty}^{\infty} w(i) x(m-i) \end{aligned}$$

Memory

Filters can have finite, or infinite memory

- **FIR:** Finite Impulse Response filters have an impulse response which have a finite number of terms, i.e. $\exists N$ such that

$$w(n) = 0, \;\; \forall |n| > N$$

- **IIR:** Infinite Impulse Response filters have an impulse response with an infinite number of terms.

  though for BIBO we require a finite sum, e.g.

$$\sum_{i=-\infty}^{\infty} |w(i)| < \infty$$

FIR example: Moving Average

(finite) **Moving Average (MA)**

$$y(n) = \sum_{i=-N}^{N} b(i) x(n-i)$$

typical example, symmetric rectangular windowed MA

$$y(n) = \frac{1}{2N+1} \sum_{i=-N}^{N} x(n-i)$$

NB: this is a non-causal filter

FIR example: difference

A **difference operator** (or filter) looks like

$$y(n) = x(n) - x(n-1)$$

Note this is a special case of the MA above
b(0) = 1, b(1) = -1
but this terminology is used differently in different fields

- signal processing and stats: MA as defined above
- financial time series: MA $\Rightarrow$ low pass

NB: this is a causal filter

Example of IIR filter: EWMA

Exponentially Weighted Moving Average (EWMA)

$$y(n) = a y(n-1) + (1-a) x(n)$$

alternative IIR representation

$$y(n) = (1-a) \sum_{i=0}^{\infty} a^i x(n-i)$$

gives exponentially decreasing weight to historical data

More general case **Autoregressive (AR)** filters

$$y(n) = \sum_{i=1}^{p} a(i) y(n-i) + b(0) x(n)$$

Transfer function

- we can represent LTI filter as convolution
- in Fourier domain, convolution becomes a simple product
- LTI filter is completely characterized by FT of its impulse response
- we call the FT of the impulse response the **Transfer function**, e.g.

$$W(k) = DFT(w(n))$$

- The transfer function tells us the impact of the filter on different components of the spectrum of a signal

Types of filters

- **low pass:** pass low frequencies, stop high frequencies.

  these filters act as smoothers of the data.

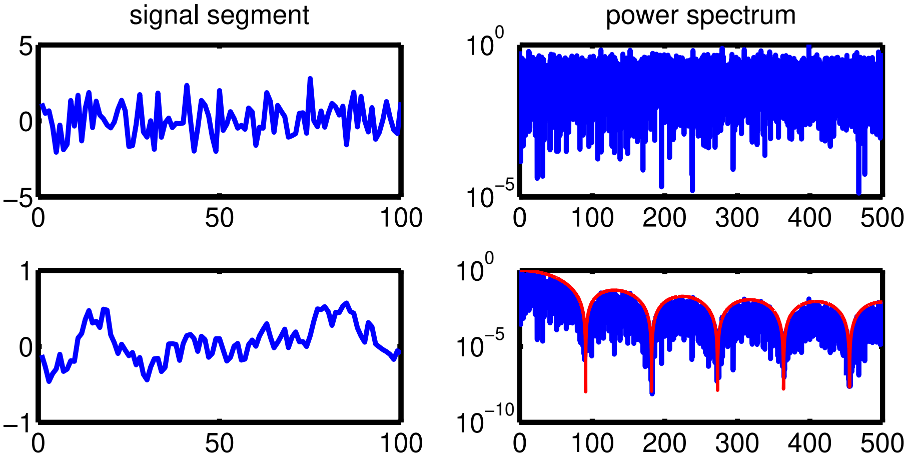
  e.g. EWMA, MA
- **high pass:** pass high frequencies, stop low frequencies.

  e.g. differencer – highlights edges
- **band pass:** pass a band of frequencies
- **notch:** exclude a band (sometimes called bandstop)

  e.g. remove signal at a particular frequency to prevent feedback ("ringing out")

Example: MA $y(n) = \frac{1}{2N+1} \sum_{i=-N}^{N} x(n-i)$

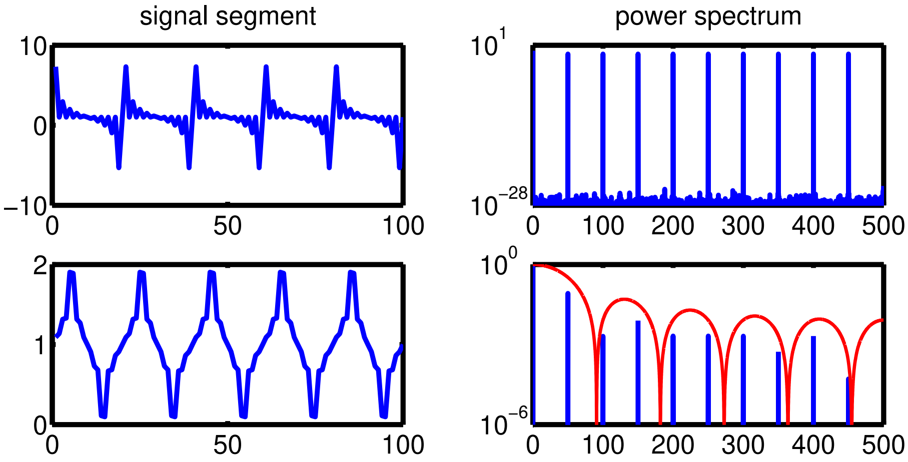
$f_s = 1000$, $N = 10,000$, input white noise, $N = 5$



low pass

Example: MA $y(n) = \frac{1}{2N+1}\sum_{i=-N}^{N} x(n-i)$

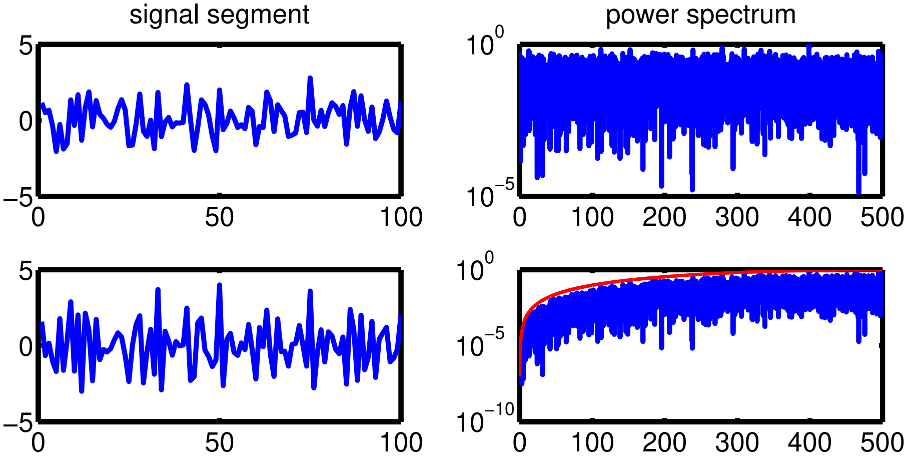
$f_s = 1000$, $N = 10,000$, input 10 sines evenly spaced freq.



low pass

Example: difference $y(n) = x(n) - x(n-1)$

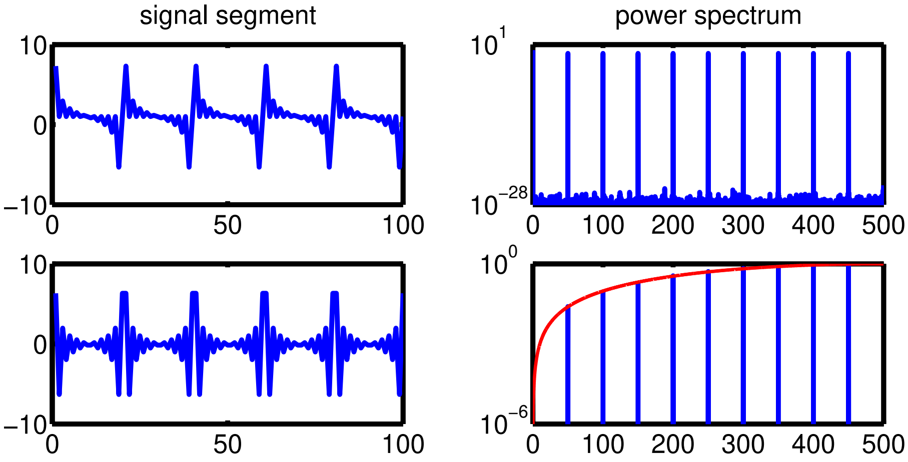
$f_s = 1000$, $N = 10,000$, input white noise



high pass

Example: difference $y(n) = x(n) - x(n-1)$

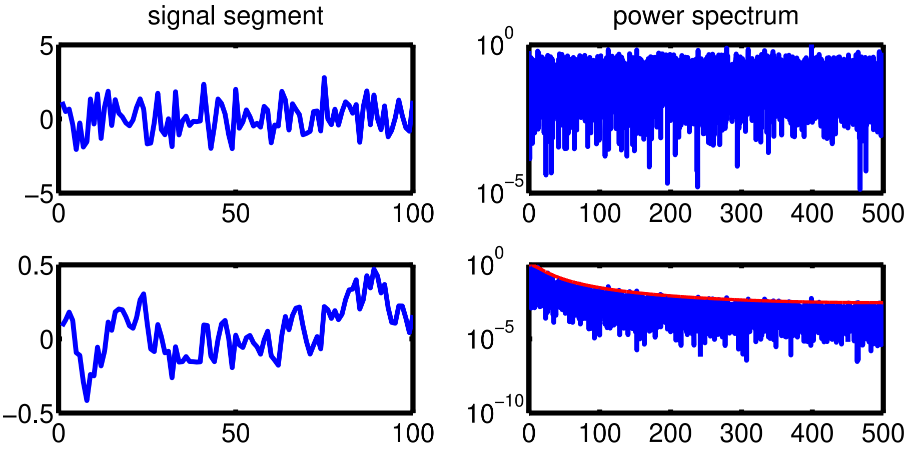
$f_s = 1000$, $N = 10,000$, input 10 sines evenly spaced freq.



high pass

Example: EWMA $y(n) = ay(n-1) + (1-a)x(n)$

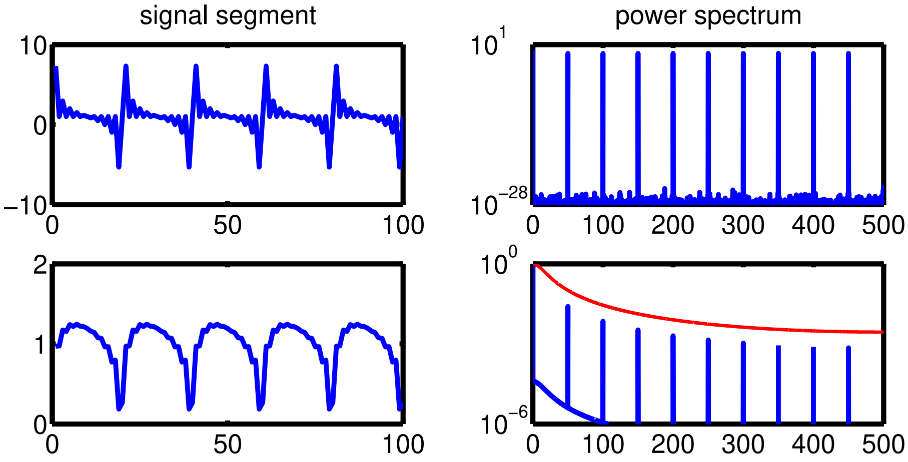
$f_s = 1000$, $N = 10,000$, input white noise, $a = 0.9$



low pass

Example: EWMA $y(n) = ay(n-1) + (1-a)x(n)$

$f_s = 1000$, $N = 10,000$, input 10 sines evenly spaced freq.



low pass

Do It: What do they sound like?

- Use Colab and Numpy to generate a *white noise* sequence
  - White means frequency spectrum is flat.
  - Implied Gaussianity

  So this is IID Gaussian.
- Try out some filters and listen to the results
  - White noise (or static)
  - Low pass, MA filter, to smooth it
  - High-pass, difference

  We're doing this in Numpy because PyTorch is a little too helpful

We should spend a lot more time talking about noise...

We could do an entire course on filter design

Filtering in the frequency domain

We could filter thus:

- `fft`
- `filter`
- `ifft`,

but this requires $O(N \log N)$ operations, which grows non-linearly in N. For many applications, we can't afford to have filtering operations grow faster than $O(N)$, e.g. real-time applications,

- The number of data points will be $f_s T$
- the time available for computation is T
- time available per data point is $1/t_s$, which is constant with respect to N.

Convolution in 2D

Convolution generalizes to 2D, e.g., the two-dimensional convolution of continuous functions $f(x, y) = \delta(x)r(y)$ with $g(x, y) = r(x)\delta(y)$ is

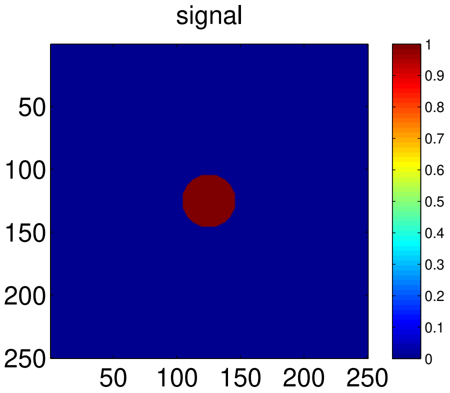
$$[f * g](x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x', y')g(x - x', y - y')\, dx'\, dy'$$

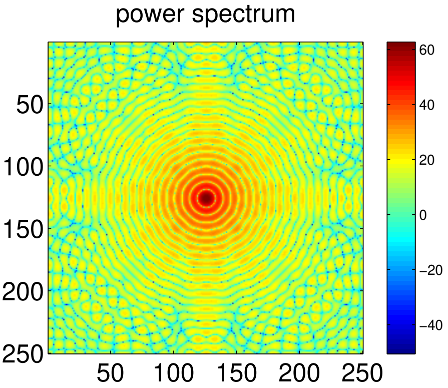
Likewise, for discrete signals we can extend the idea of a LTI filter (a convolution) to 2D

$$[x * y](n, m) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} x(k, l)y(n - k, m - l)$$

Filters in 2D

2D data





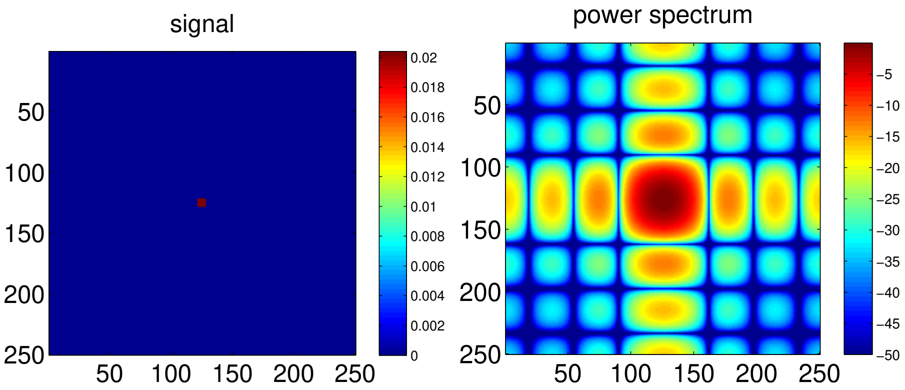
Filters in 2D

Spatial rectangular low-pass

$$\frac{1}{49}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

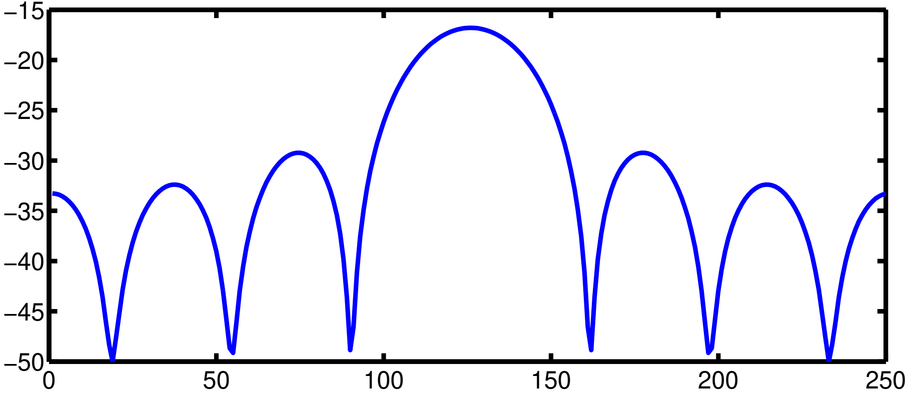
Filters in 2D

Spatial rectangular low pass, frequency response



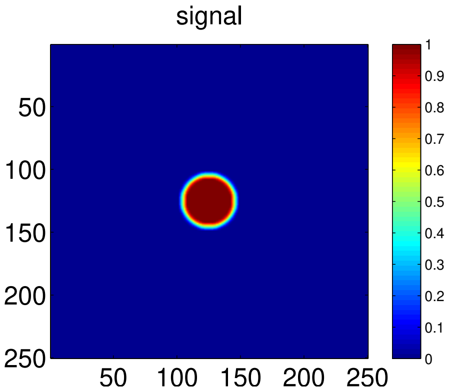
Filters in 2D

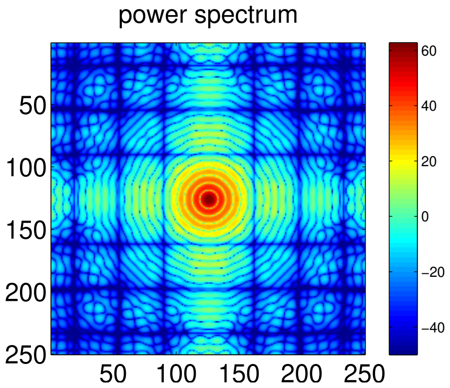
Spatial rectangular low pass, frequency response 1D



Filters in 2D

Spatial rectangular low pass





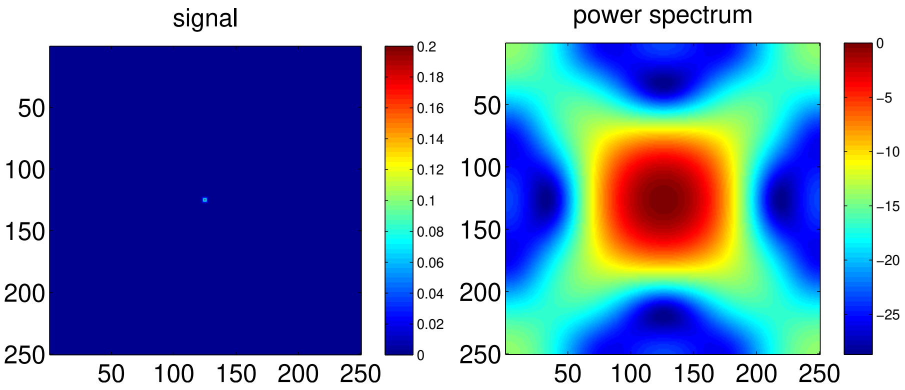
Filters in 2D

Approximately Gaussian low-pass

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 4 & 4 & 4 & 1 \\ 1 & 4 & 12 & 4 & 1 \\ 1 & 4 & 4 & 4 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

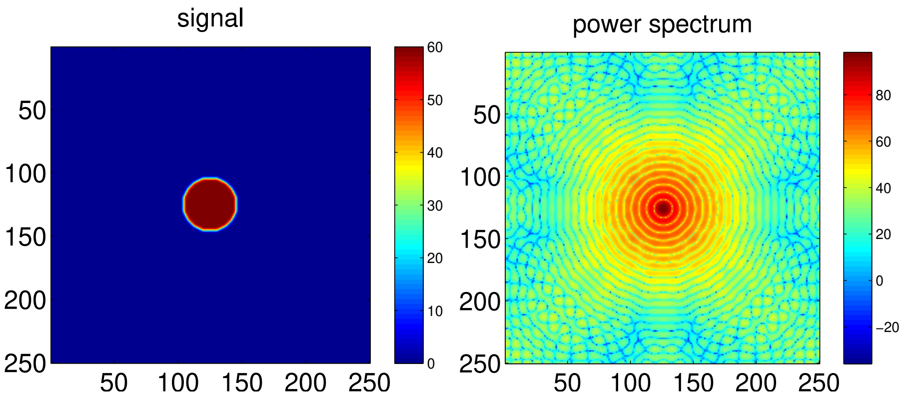
Filters in 2D

Approximately Gaussian low-pass, frequency response



Filters in 2D

Approximately Gaussian low-pass



Edge detection

Edge detection is a high pass operation, but there are a number of ways to do this in 2D.

- gradient: look for max and min in gradients in image
  - Sobel
  - Prewitt
  - Roberts
- Laplacian: look for zeros of second derivative
  - Marrs-Hildreth

Sobel Edge detection

Look for vertical and horizontal edge separately, using filters

$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

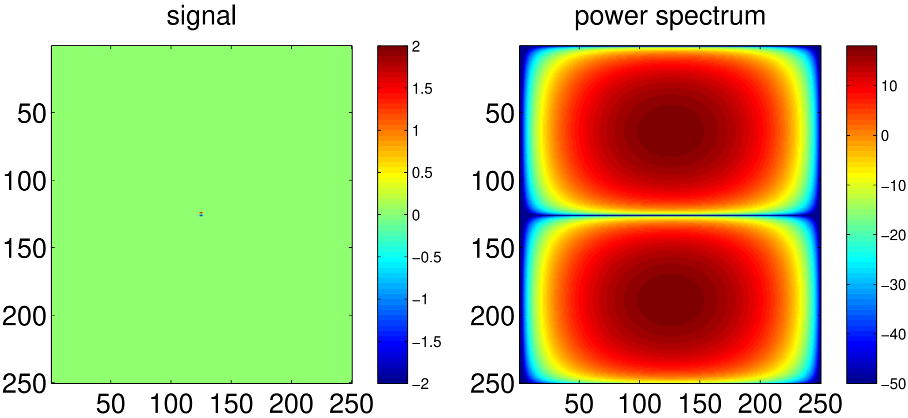
The threshold for values above or below T.

Alternatively, can combine to get edge magnitude and direction by

$$\begin{aligned} M_{\text{sobel}} &= \sqrt{M_{\text{vertical}}^2 + M_{\text{horizontal}}^2} \\ \phi_{\text{sobel}} &= \tan^{-1}\left(M_{\text{vertical}}/M_{\text{horizontal}}\right) \end{aligned}$$

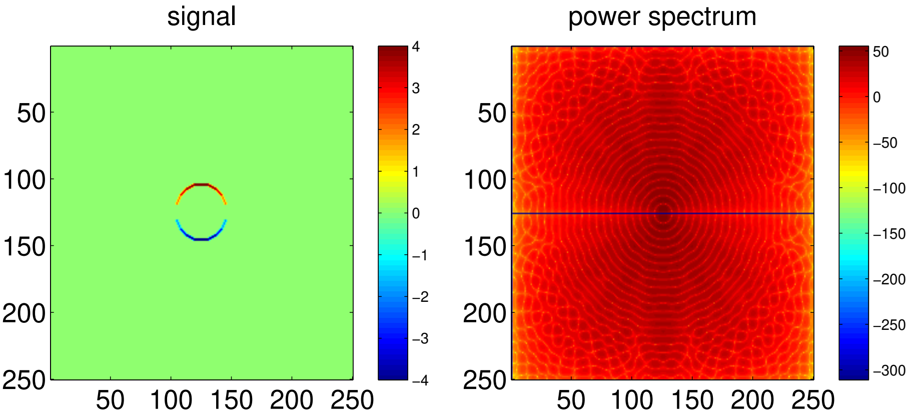
Filters in 2D

Sobel edge detection, freq. response



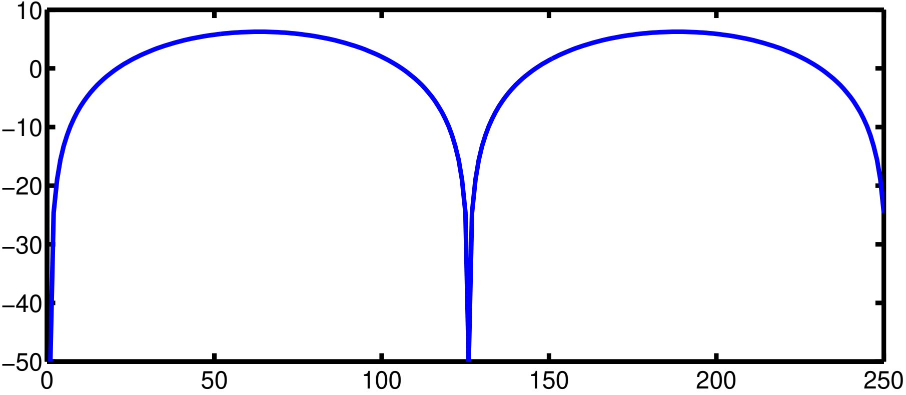
Filters in 2D

Sobel edge detection applied to image.



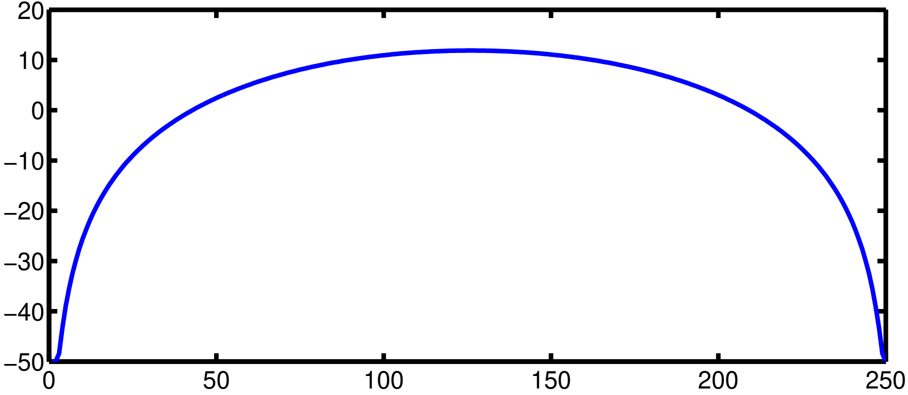
Filters in 2D

Vertical high pass (Sobel edge detection), freq. response, horizontal



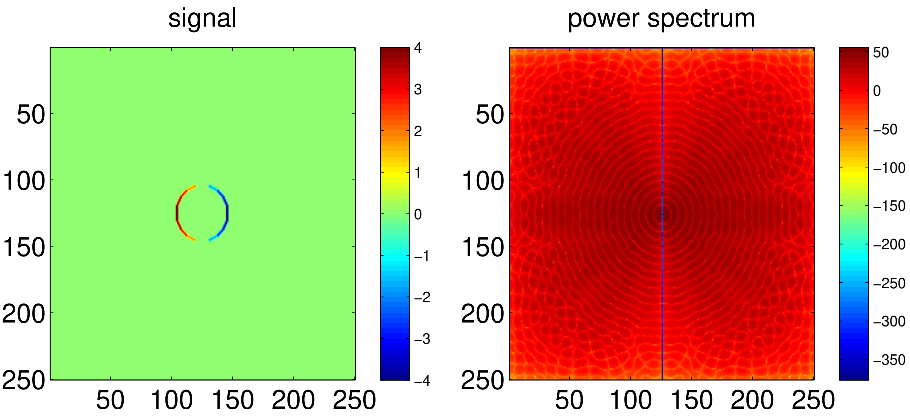
Filters in 2D

Vertical high pass (Sobel edge detection), freq. response, vertical



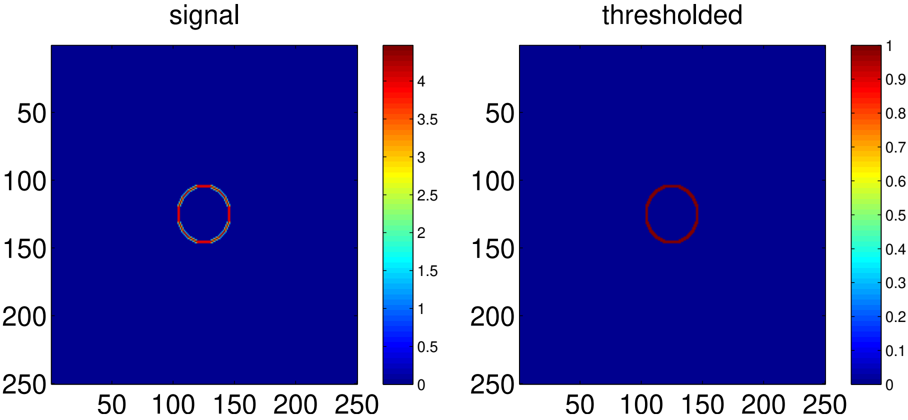
Filters in 2D

Horizontal high pass (Sobel edge detection)



Filters in 2D

Combined Sobel filters, and thresholded version



Prewitt filters

Look for vertical and horizontal edge separately, using filters

$$\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

Roberts filters

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Laplacian

Approximation of $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$

3×3

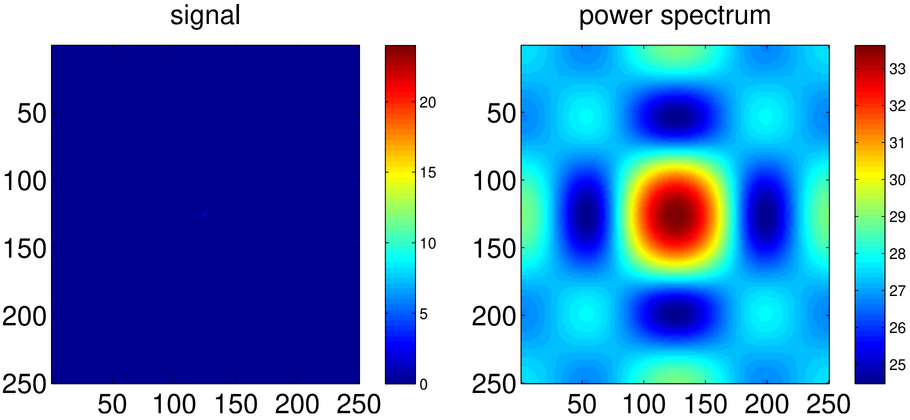
$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

5×5

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 24 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

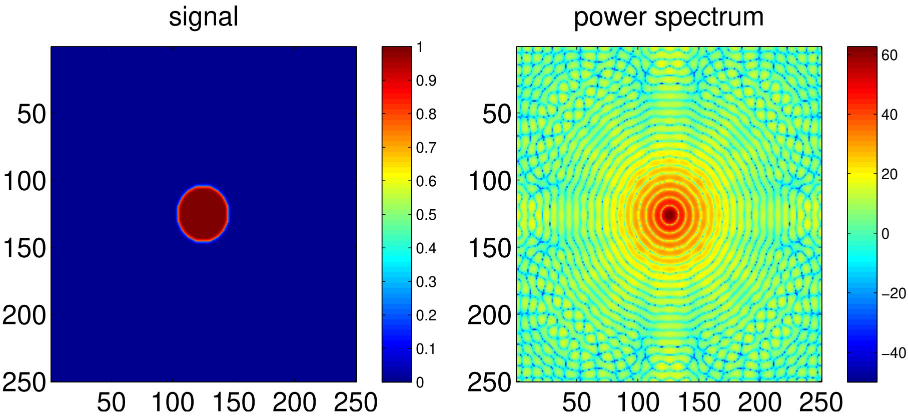
Filters in 2D

5×5 Laplacian, and its frequency response



Filters in 2D

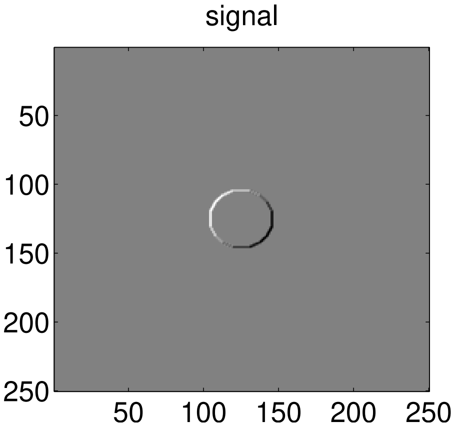
5 $\times$ 5 Laplacian applied to image



Embossing an image

Fancy effects from simple filters, e.g. take $\theta = \pi/6$ and

$$\text{image} = M_{\text{sobel}}^{\text{horizontal}} \cos(\theta) + M_{\text{sobel}}^{\text{vertical}} \sin(\theta)$$



Embossing an image