COM-202: Signal Processing

Chapter 8.a: Stochastic and adaptive signal processing

adaptive signal processing (aka “machine learning”)

Adaptive Signal Processing

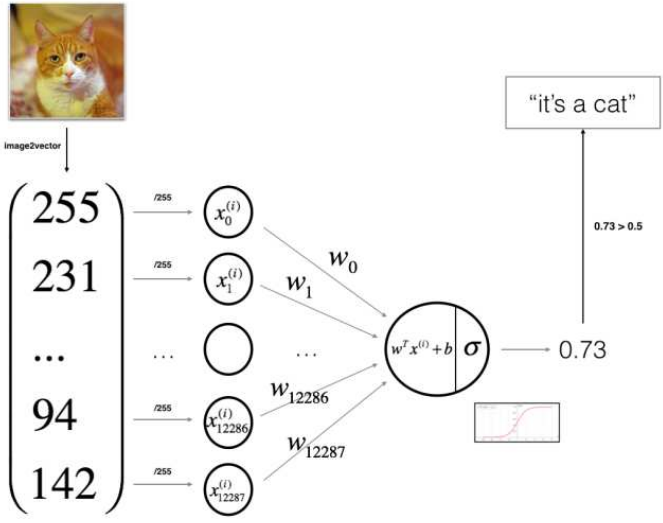
- deterministic signals are completely known; e.g. $x[n] = \sin((\pi/5)\, n)$
- deterministic signals are not interesting!!
- interesting signals are *not* known in advance; e.g. $s[n] =$ what I'm going to say next
- how can we design processing systems for "unknown" signals?

Adaptation and learning

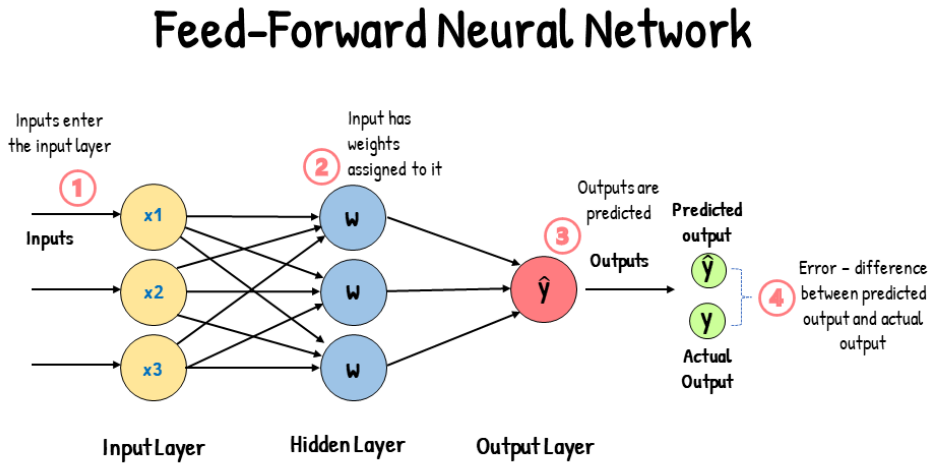
Goals of machine learning:

- design a system that can learn a specific task
- learning should be data-driven (using training data)
- system should be robust to data variability (generalization property)

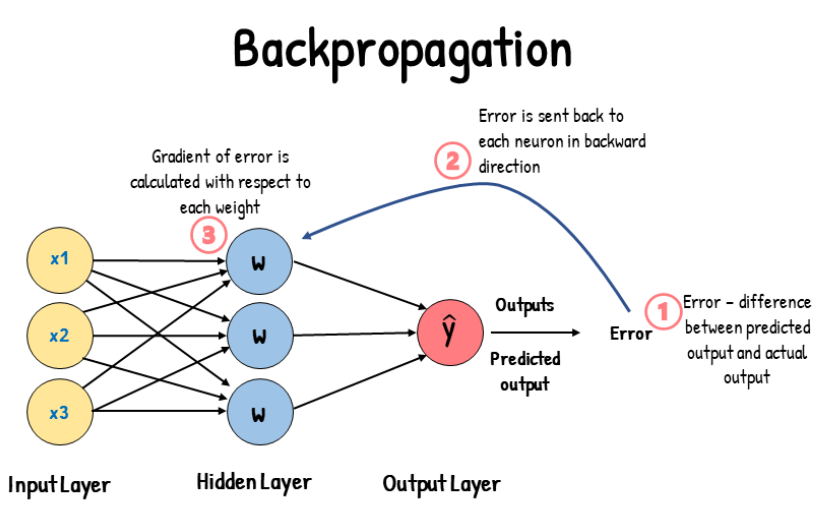
Example: recognizing cats



Inference


Feed-Forward Neural Network
Inputs enter the input layer
1
Inputs
x1
x2
x3
Input has weights assigned to it
2
w
w
w
Outputs are predicted
3
Outputs
ŷ
Predicted output
ŷ
y
Actual Output
4
Error – difference between predicted output and actual output
Input Layer
Hidden Layer
Output Layer

Backpropagation

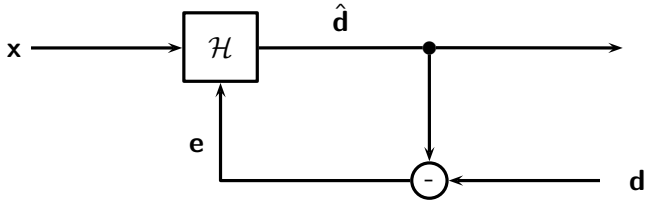


In signal processing terms

Goals of machine learning:

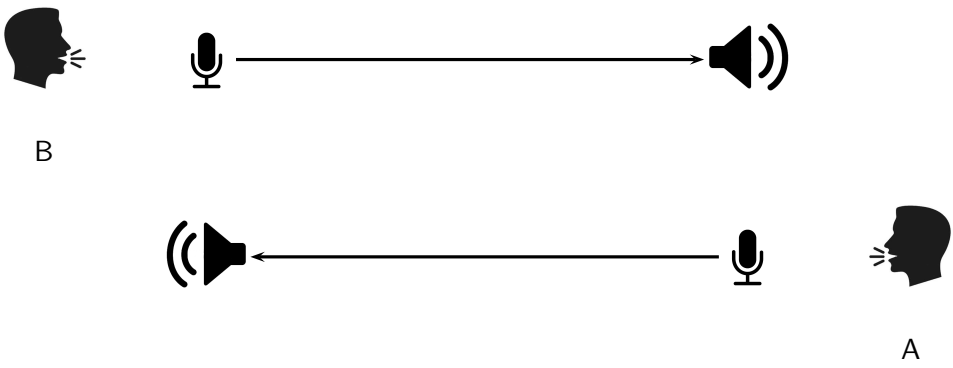
- design a filter that can implement a specific response
- filter design should be data-driven (using training signals)
- the filter should be robust to input variability

Adaptive signal processing

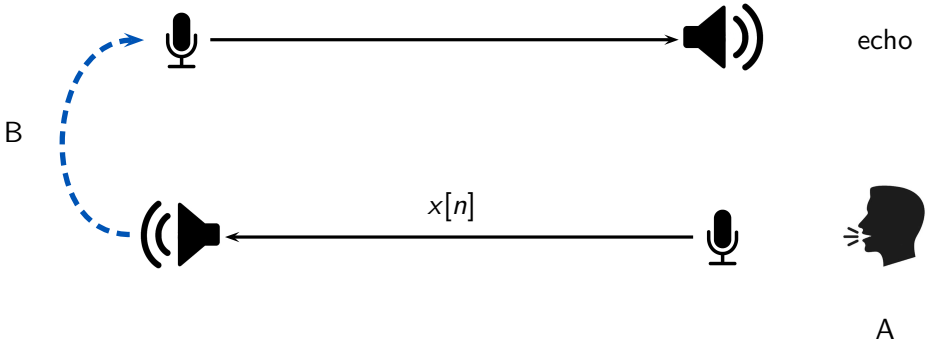


- $\mathbf{x}$: non-deterministic (unknown) input
- $\mathcal{H}$ adaptive filter with *learned* impulse response $\mathbf{h}$
- $\hat{\mathbf{d}} = \mathbf{x} * \mathbf{h}$: filter's output
- $\mathbf{d}$: desired (target) output
- $\mathbf{e} = \mathbf{d} - \hat{\mathbf{d}}$: error signal driving the filter's adaptation

Example: handsfree telephony

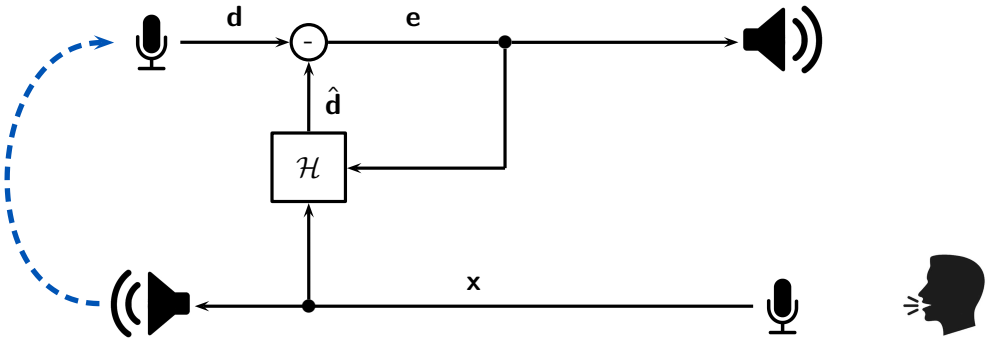


Feedback problem



- A speaks, voice is played by B's loudspeaker
- B's microphone captures A's voice from loudspeaker
- signal is amplified and fed back to A and cycle repeats
- result: high-pitched noise (Larsen's effect)

Adaptive echo cancellation



goal: make $\mathcal{H}$ learn how **x** becomes **d** by making **e** as small as possible

Challenges of adaptive echo cancellation

$\mathcal{H}$ must simulate the combined effects of loudspeaker, microphone and room

- transfer functions of mike, speaker, room are not known
- room's transfer function may change over time
- input signal is unknown

Learning and generalization

- adaptive systems must be able to learn and generalize
- input signals are not known exactly...
- ... but we must be able to “categorize” them!

- categorization requires comparison
- comparison results should be robust to variations in sample values
- comparisons should work also for somewhat “random” inputs

The key ingredient

- the inner product is the fundamental similarity metric in signal processing
- we will use it to build a robust descriptor for random signals
- we will use it to drive the learning process of adaptive systems

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{n=-\infty}^{\infty} x^*[n] y[n]$$

Brief recap of vector notation and operator algebra

signal	sample at time n
$\mathbf{x}$	$x[n]$
$\mathcal{R}\mathbf{x}$	$x[-n]$
$\mathcal{S}^{-d}\mathbf{x}$	$x[n-d]$
$\mathcal{S}^{-d}\mathcal{R}\mathbf{x}$ $\mathcal{R}\mathcal{S}^{d}\mathbf{x}$	$x[-n+d]$
$\mathcal{S}^{d}\mathcal{R}\mathbf{x}$ $\mathcal{R}\mathcal{S}^{-d}\mathbf{x}$	$x[-n-d]$

time reversal and shift:

$$x[-(n-d)] = x[-m]_{m=n-d} = (\mathcal{R}\mathbf{x})[n-d] = (\mathcal{S}^{-d}\mathcal{R}\mathbf{x})[n]$$

$$x[-n+d] = x[m+d]_{m=-n} = (\mathcal{S}^{d}\mathbf{x})[-n] = (\mathcal{R}\mathcal{S}^{d}\mathbf{x})[n]$$

Convolution and time operations

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

$$\begin{aligned}(\mathcal{R}\mathbf{x} * \mathbf{y})[k] &= \sum_{n=-\infty}^{\infty} x[-n]y[k-n] \\ &= \sum_{m=-\infty}^{\infty} x[m]y[-(-k-m)] \\ &= (\mathbf{x} * \mathcal{R}\mathbf{y})[-k]\end{aligned}$$

Convolution and time operations

$$\mathcal{R}\mathbf{x} * \mathbf{y} = \mathcal{R}(\mathbf{x} * \mathcal{R}\mathbf{y})$$
$$\mathbf{x} * \mathcal{R}\mathbf{y} = \mathcal{R}(\mathcal{R}\mathbf{x} * \mathbf{y})$$
$$\mathcal{R}\mathbf{x} * \mathcal{R}\mathbf{y} = \mathcal{R}(\mathbf{x} * \mathbf{y})$$

$$\mathcal{S}^d\mathbf{x} * \mathbf{y} = \mathbf{x} * \mathcal{S}^d\mathbf{y} = \mathcal{S}^d(\mathbf{x} * \mathbf{y})$$
$$\mathcal{S}^c\mathbf{x} * \mathcal{S}^d\mathbf{y} = \mathcal{S}^{c+d}(\mathbf{x} * \mathbf{y})$$

correlation, autocorrelation & spectral density

Correlation

The cross-correlation (or just correlation) between two finite-energy signals is defined as

$$r_{xy}[k] = \langle \mathbf{x}, \mathcal{S}^k \mathbf{y} \rangle = \sum_{n=-\infty}^{\infty} x^*[n] y[n+k]$$

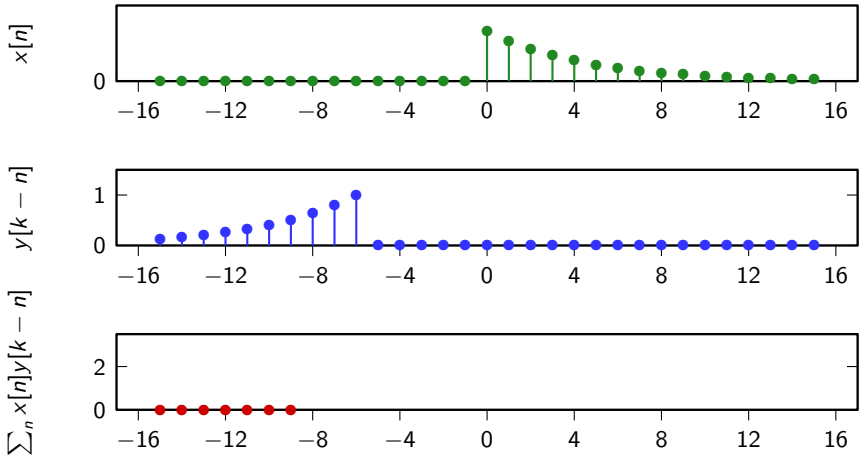
- $r_{xy}[k]$ measures the similarity between $\mathbf{x}$ and $\mathbf{y}$ at a relative shift of k samples
- k is usually called the *lag*

Correlation and convolution

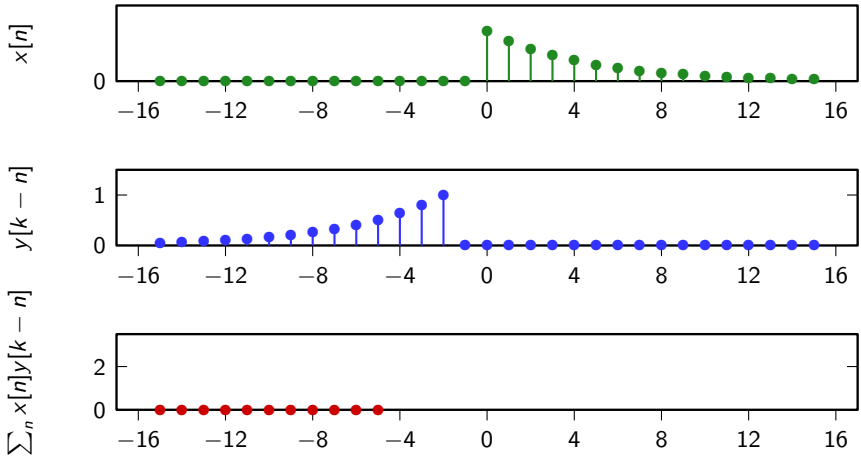
$$
\begin{aligned}
r_{xy}[k] &= \sum_{n=-\infty}^{\infty} x^*[n]y[n+k] \\
&= \sum_{m=-\infty}^{\infty} x^*[-m]y[-m+k] \\
&= \sum_{m=-\infty}^{\infty} (\mathcal{R}\mathbf{x}^*)[m]y[k-m] \\
&= (\mathcal{R}\mathbf{x}^* * \mathbf{y})[k]
\end{aligned}
$$

$$
\mathbf{r}_{xy} = \mathcal{R}\mathbf{x}^* * \mathbf{y}
$$

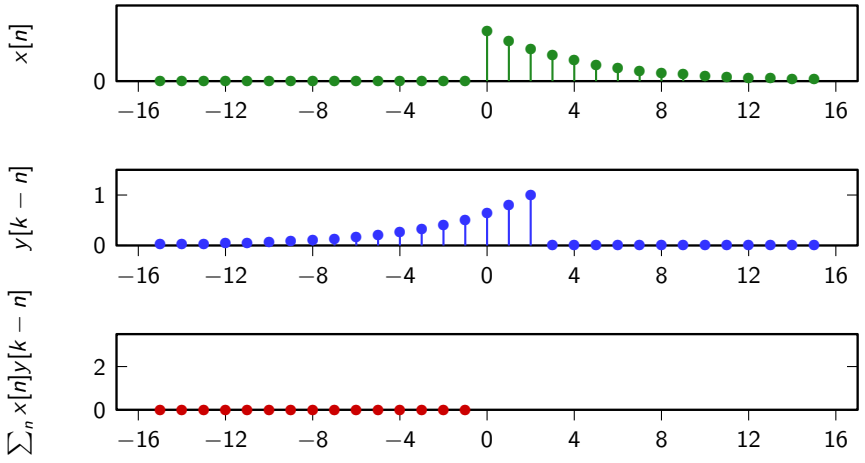
Convolution



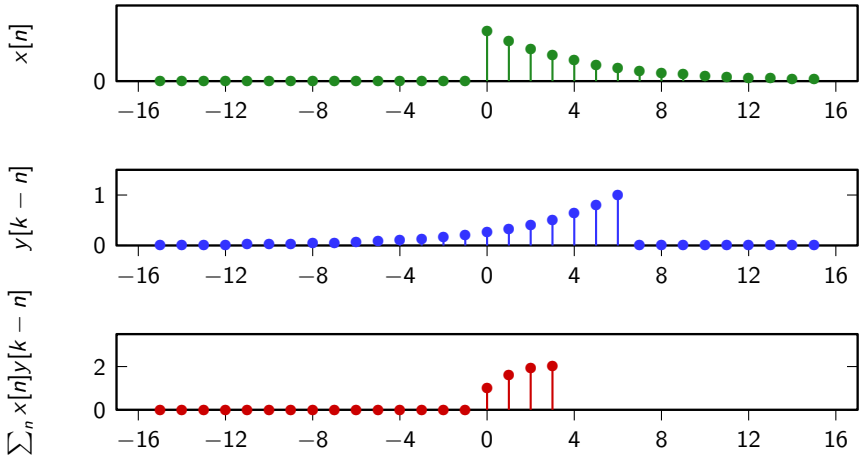
Convolution



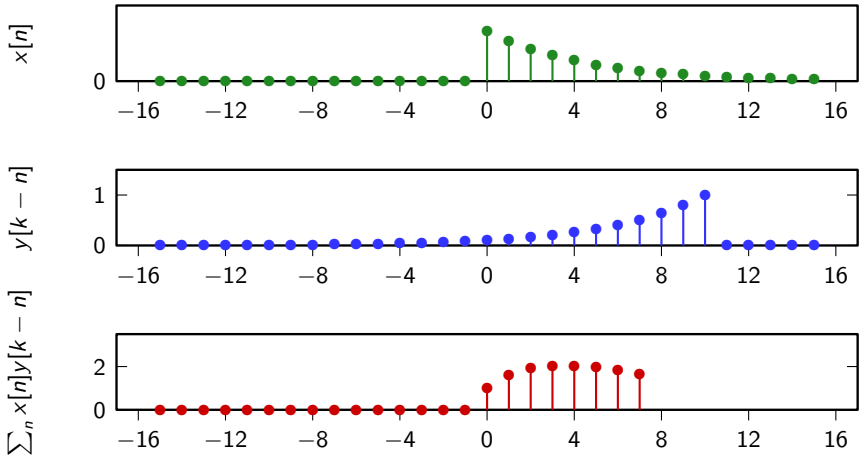
Convolution



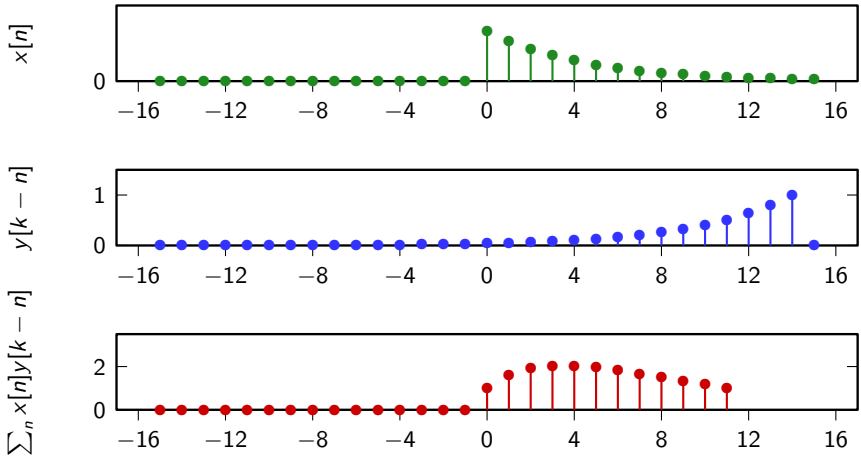
Convolution



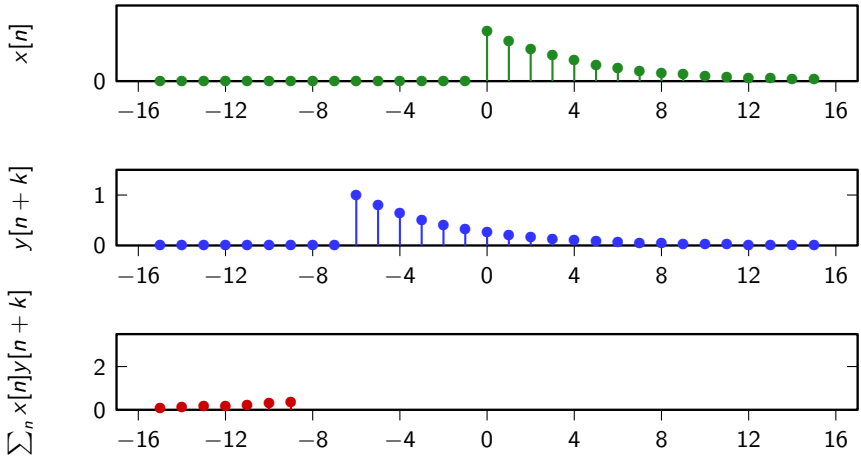
Convolution



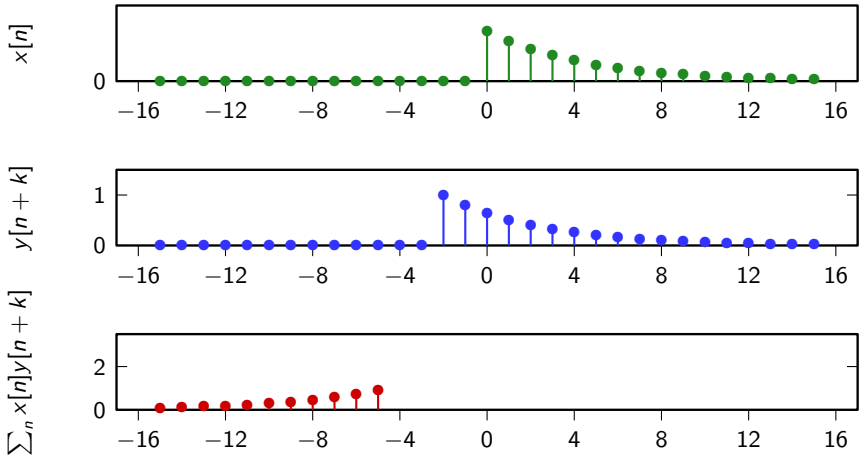
Convolution



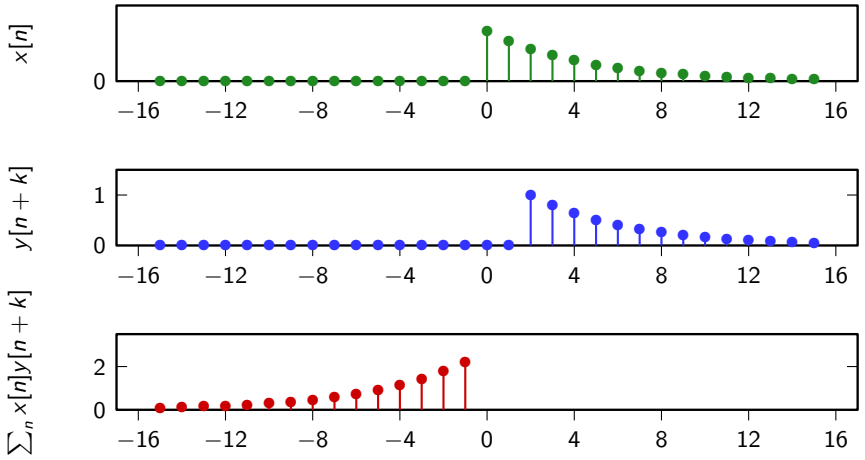
Correlation



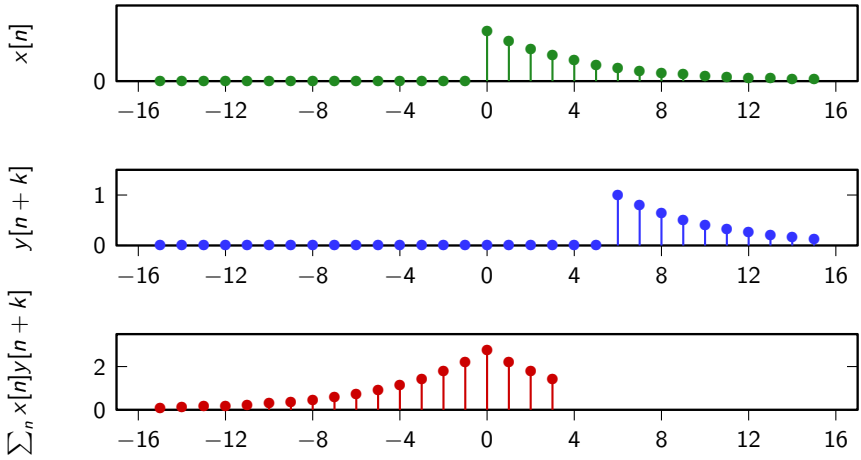
Correlation



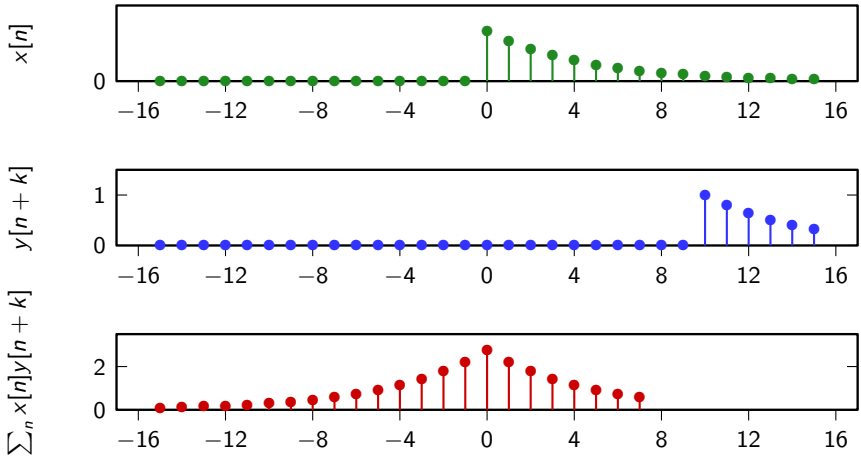
Correlation



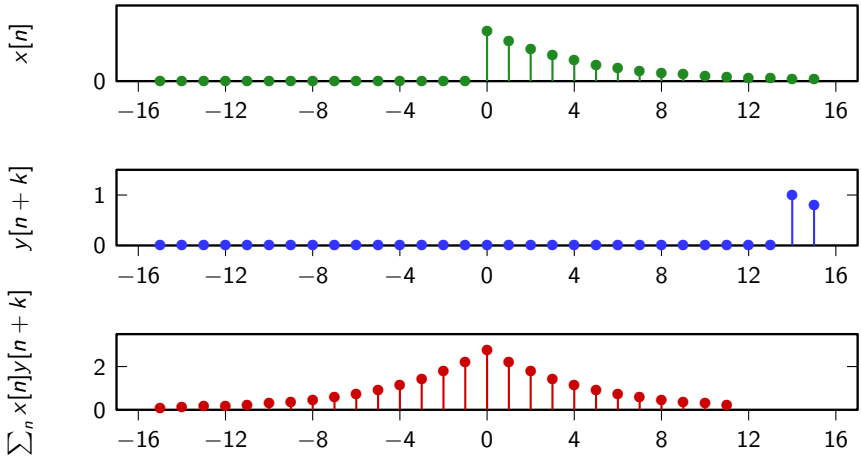
Correlation



Correlation



Correlation



Cross-correlation: the order does matter

$$
\begin{aligned}
\mathbf{r}_{xy} &= \mathcal{R}\mathbf{x}^* * \mathbf{y} \\
\mathbf{r}_{yx} &= \mathcal{R}\mathbf{y}^* * \mathbf{x} \\
&= \mathcal{R}(\mathbf{y}^* * \mathcal{R}\mathbf{x}) \\
&= \mathcal{R}(\mathcal{R}\mathbf{x}^* * \mathbf{y})^* \\
&= \mathcal{R}\mathbf{r}_{xy}^*
\end{aligned}
$$

$$
r_{yx}[k] = r_{xy}^*[-k]
$$

Autocorrelation

$$r_x[k] = \langle \mathbf{x}, \mathcal{S}^k \mathbf{x} \rangle = \sum_{n=-\infty}^{\infty} x^*[n] x[n+k]$$

$$\mathbf{r}_x = \mathcal{R}\mathbf{x}^* * \mathbf{x}$$

- compare signal with a shifted copy of itself
- measures signal's *self-similarity* over time
- well-defined for square-summable (energy) signals

Properties of the autocorrelation

$$\mathbf{r}_x = \mathcal{R}\mathbf{x}^* * \mathbf{x}$$

- $\mathbf{y} = \mathcal{S}^d\mathbf{x} \Rightarrow \mathbf{r}_y = \mathbf{r}_x$ (shift-invariance)
- $\mathbf{r}_x = \mathcal{R}\mathbf{r}_x^*$ (Hermitian symmetry)
- $r_x[0] = \|\mathbf{x}\|^2$ (value in zero is total energy)
- $|r_x[0]| \geq |r_x[k]|$ (peak magnitude in zero)

Proof of the last point

- intuition: a signal is maximally similar to itself!
- proof (assume $x[n] \in \mathbb{R}$ to keep things simpler): for any two reals we have

$$(a-b)^2 = a^2 + b^2 - 2ab \qquad \Rightarrow \qquad ab = [a^2 + b^2 - (a-b)^2]/2$$

and so

$$\begin{aligned} r_x[k] &= \sum_{n=-\infty}^{\infty} x[n]x[n+k] \\ &= \frac{1}{2}\sum_{n=-\infty}^{\infty} x^2[n] + \frac{1}{2}\sum_{n=-\infty}^{\infty} x^2[n+k] - \frac{1}{2}\sum_{n=-\infty}^{\infty} (x[n]-x[n+k])^2 \\ &= r_x[0] - \frac{1}{2}\sum_{n=-\infty}^{\infty} (x[n]-x[n+k])^2 \leq r_x[0] \end{aligned}$$

Application: delay estimation via correlation

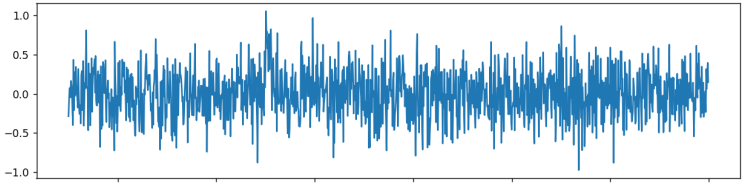
- assume $\mathbf{y} = \mathcal{S}^d \mathbf{x}$, with $\mathbf{x}$ known and d unknown
- we want to find d
- the cross-correlation is $\mathbf{r}_{xy} = \mathcal{R}\mathbf{x}^* * \mathcal{S}^d \mathbf{x} = \mathcal{S}^d(\mathcal{R}\mathbf{x}^* * \mathbf{x}) = \mathcal{S}^d \mathbf{r}_x$
- we know $|r_x[0]| \geq |r_x[m]|$ for all $m \neq 0$ therefore $\mathbf{r}_{xy}$ will have a peak in $-d$
- we can find d by looking for the peak of $\mathbf{r}_{xy}$

$$d = -\arg\max_n \{r_{xy}[n]\}$$

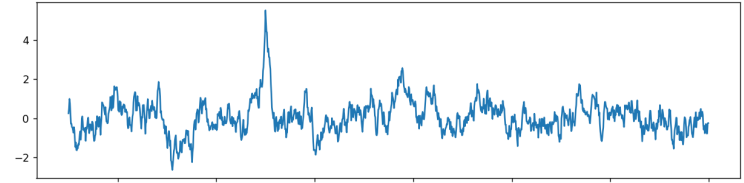
- this works also if the signal is buried in noise

Detection in noise via cross-correlation

$$x[n] = a^n\, u[n], \quad y[n] = x[n-100] + \eta[n], \quad \eta[n] = \text{random noise}$$



$\mathbf{r}_{xy}$



Autocorelation example: delta sequence

$$x[n] = a\delta[n]$$

$$r_x[k] = a^2 \sum_{n=-\infty}^{\infty} \delta[n]\delta[n+k] = a^2\delta[k]$$

$$= a^2\delta[k]$$

Autocorelation example: rect

$$x[n] = \text{rect}\left(\frac{n}{2N}\right)$$

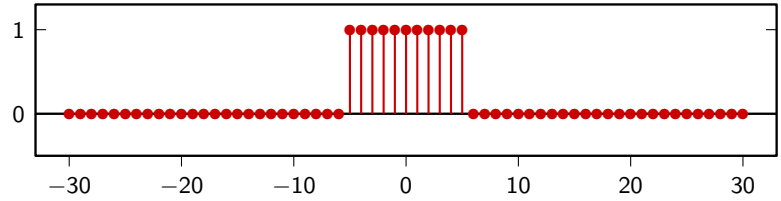
$$r_x[k] = \sum_{n=-\infty}^{\infty} x[n]x[n+k] = \sum_{n=-N}^{N} \text{rect}\left(\frac{n+k}{2N}\right) = \sum_{n=\max\{-N,-N+k\}}^{\min\{N,N+k\}} 1$$

$$= \begin{cases} 2N+1-|k| & |k| \leq 2N \\ 0 & |k| > 2N \end{cases}$$

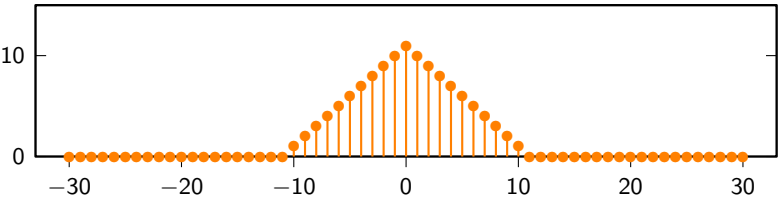
$$= (2N+1-|k|)\,\text{rect}(n/(4N))$$

Autocorelation example: rect

$$x[n] = \text{rect}(n/(2N)), \quad N = 5$$



$$r_x[k] = (2N + 1 - |k|)\,\text{rect}(n/(4N))$$



Autocorelation example: exponential decay

$$x[n] = a^n u[n]$$

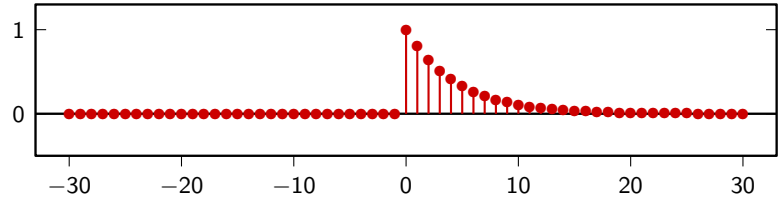
$$r_x[k] = \sum_{n=-\infty}^{\infty} x[n]x[n+k] = \sum_{n=0}^{\infty} a^n a^{n+k} u[n+k] = \sum_{n=\max\{0,-k\}}^{\infty} a^{2n+k}$$

$$= \begin{cases} a^k \sum_{n=0}^{\infty} a^{2n} & k \geq 0 \\ a^{-k} \sum_{n=-k}^{\infty} a^{2n} = a^{|k|} \left(\sum_{n=0}^{\infty} a^{2n} - \sum_{n=0}^{-k-1} a^{2n} \right) & k < 0 \end{cases}$$

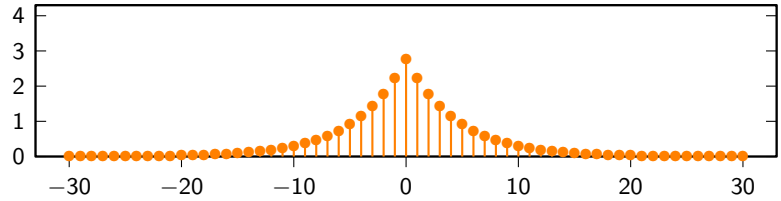
$$= \frac{a^{|k|}}{1 - a^2}$$

Autocorelation example: exponential decay

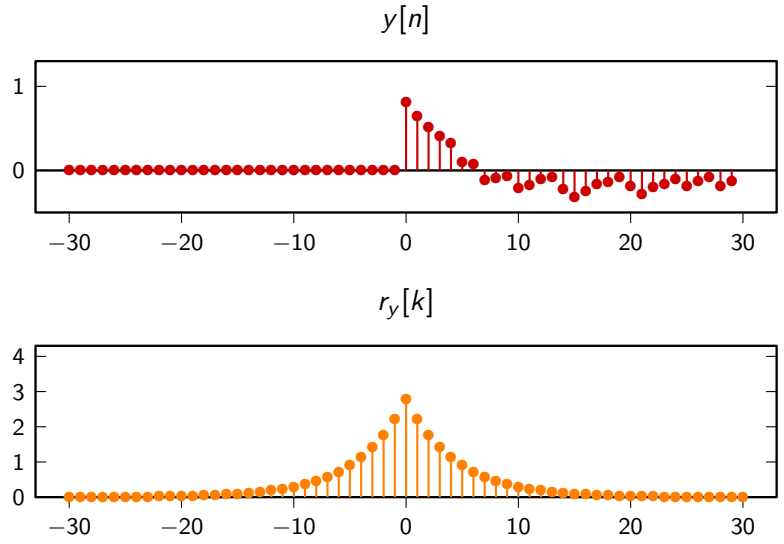
$$x[n] = a^n u[n], \quad a = 0.8$$



$$r_x[k] = a^{|k|}/(1 - a^2)$$



The autocorrelation is a robust descriptor



To understand why, let's move to the frequency domain

$$
\begin{aligned}
\text{DTFT}\left\{\mathbf{r}_x\right\} &= \text{DTFT}\left\{\mathcal{R}\mathbf{x}^* * \mathbf{x})\right\} \\
&= \text{DTFT}\left\{\mathcal{R}\mathbf{x}^*\right\} \cdot \text{DTFT}\left\{\mathbf{x}\right\} \\
&= \mathbf{X}^* \cdot \mathbf{X} \\
&= |\mathbf{X}|^2
\end{aligned}
$$

Energy spectral density

$$P_x(\omega) = \text{DTFT}\left\{\mathbf{r}_x\right\}(\omega) = |X(\omega)|^2$$

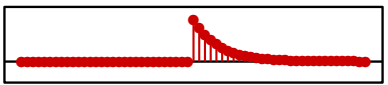
- square magnitude of DTFT is the signal's spectral distribution energy
- DTFT of autocorrelation retains where the energy of the signal is
- phase information is discarded: the shape of the signal in time does not matter
- autocorrelation is invariant to shifts and shape changes

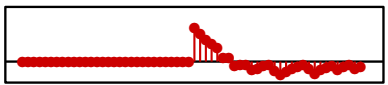
Example revisited: same magnitude, different phase

$x[n] = a^n u[n]$

$\mathbf{y} = \mathbf{h} * \mathbf{x}$

$|H(\omega)| = 1$ *(allpass filter)*



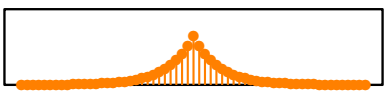


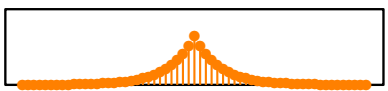
$P_x(\omega) = |X(\omega)|^2 = \left|\frac{1}{1-ae^{-j\omega}}\right|^2$

$P_y(\omega) = |Y(\omega)|^2 = |H(\omega)|^2 \, |X(\omega)|^2 = P_x(\omega)$

$r_x[k] = a^{|k|}/(1-a^2)$

$\mathbf{r}_y = \mathbf{r}_x$





Autocorrelation of a filtered signal

$$\mathbf{y} = \mathbf{h} * \mathbf{x}$$

$$\begin{aligned}
\mathbf{r}_y &= \mathcal{R}\mathbf{y}^* * \mathbf{y} \\
&= (\mathcal{R}\mathbf{h}^* * \mathcal{R}\mathbf{x}^*) * (\mathbf{h} * \mathbf{x}) \\
&= \mathcal{R}\mathbf{h}^*\mathbf{h} * \mathcal{R}\mathbf{x}^* * \mathbf{x} \\
&= \mathbf{r}_h * \mathbf{r}_x
\end{aligned}$$

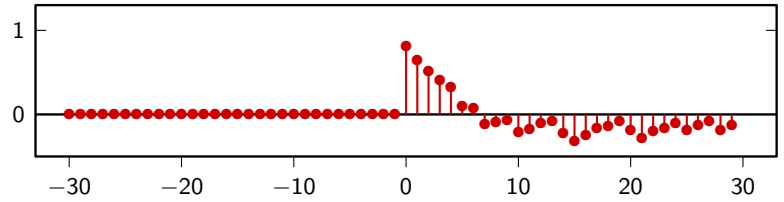
$$P_y(\omega) = |H(\omega)|^2 P_x(\omega)$$

Autocorrelation of a filtered signal

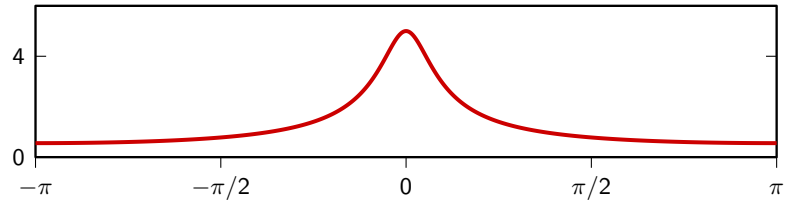
- autocorrelation of the output is the autocorrelation of the input filtered by $|H(\omega)|^2$
- filters act on a signal's PSD "as intended" (lowpass, highpass, etc)
- phase information is discarded since $|H(\omega)|^2$ is real-valued

Example revisited: same magnitude, different phase

$$x[n] = a^n\, u[n], \quad y[n] = (h * x)[n], \quad h[n] \text{ allpass}$$



$$P_y(\omega) = |1 - ae^{j\omega}|^{-2}$$



Intuition

- signals same energy distribution in frequency can look very different in time
- spectral energy distribution is a more robust characterization of a signal
- autocorrelation captures this robust feature

What about power signals?

power signals have infinite energy but their energy per unit of time (i.e. their power) is finite: if $\mathbf{x}$ is a power signal, define

$$x_N[n] = \begin{cases} x[n] & |n| \leq N \\ 0 & |n| > N \end{cases}$$

$$\lim_{N\to\infty} \|\mathbf{x}_N\|^2 = \infty$$

$$\lim_{N\to\infty} \frac{\|\mathbf{x}_N\|^2}{2N+1} < \infty$$

Autocorrelation of power signals

for power signals the correlation is the limit of the normalized partial correlation:

$$\begin{aligned}\mathbf{r}_x &= \lim_{N\to\infty} \frac{\mathbf{r}_{x_N}}{2N+1} \\ &= \lim_{N\to\infty} \frac{\mathcal{R}\mathbf{x}_N^* * \mathbf{x}_N}{2N+1}\end{aligned}$$

similarly, for a cross-correlation,

$$\mathbf{r}_{xy} = \lim_{N\to\infty} \frac{\mathcal{R}\mathbf{x}_N^* * \mathbf{y}_N}{2N+1}$$

Spectral density for power signals

- for an energy signal the squared DTFT shows the spectral energy distribution
- a truncated power signal $\mathbf{x}_N$ is an energy signal
- if $\|\mathbf{x}_N\|^2/(2N+1)$ tends to the average power...
- ...then $|X_N(\omega)|^2/(2N+1)$ should tends to the *power* spectral distribution

Spectral density for power signals

$$
\begin{aligned}
|X_N(\omega)|^2 &= X_N^*(\omega)\, X_N(\omega) \\
&= \text{DTFT}\left\{\mathcal{R}\mathbf{x}_N^*\right\} \text{DTFT}\left\{\mathbf{x}_N\right\} \\
&= \text{DTFT}\left\{\mathcal{R}\mathbf{x}_N^* * \mathbf{x}_N\right\}
\end{aligned}
$$

$$
\begin{aligned}
\lim_{N\to\infty} \frac{|X_N(\omega)|^2}{2N+1} &= \text{DTFT}\left\{\lim_{N\to\infty} \frac{\mathcal{R}\mathbf{x}_N^* * \mathbf{x}_N}{2N+1}\right\} \\
&= \text{DTFT}\left\{\mathbf{r}_x\right\}
\end{aligned}
$$

Power Spectral Density

the Power Spectral Density (PSD) of a power signal $\mathbf{x}$ is *defined* as

$$P_x(\omega) = \text{DTFT}\{\mathbf{r}_x\}(\omega)$$

- shows the power distribution in frequency for the signal
- for a filtered power signal $\mathbf{y} = \mathbf{h} * \mathbf{x}$, the previous result holds:

$$P_y(\omega) = |H(\omega)|^2 P_x(\omega)$$

- again, phase information is discarded

[**Important**: the PSD of a power signal is *not* the square magnitude of its DTFT. The DTFT of a power signal does not exist and their spectral representation is a *generalized* DTFT that contains Dirac deltas. Mathematically it makes no sense to square a Dirac delta.]

Example: constant signal

$$x[n] = a$$

$$\begin{aligned} r_x[k] &= \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} |a|^2 \\ &= |a|^2 \end{aligned}$$

$$P_x(\omega) = |a|^2 \tilde{\delta}(\omega)$$

Example: unit step

$$x[n] = a\,u[n]$$

$$\begin{aligned} r_x[k] &= \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=0}^{N} |a|^2 u[n+k] \\ &= |a|^2 \lim_{N\to\infty} \frac{N+1-\max\{0,k\}}{2N+1} \\ &= \frac{|a|^2}{2} \end{aligned}$$

$$P_x(\omega) = (|a|^2/2)\tilde{\delta}(\omega)$$

Example: complex exponential

$$x[n] = a\, e^{j\omega_0 n}$$

$$\begin{aligned} r_x[k] &= \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} |a|^2 (e^{j\omega_0 n})^* e^{j\omega_0 (n+k)} \\ &= |a|^2 e^{j\omega_0 k} \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} 1 \\ &= |a|^2 e^{j\omega_0 k} \end{aligned}$$

$$P_x(\omega) = |a|^2 \tilde{\delta}(\omega - \omega_0)$$

Example: trigonometric functions

$$x[n] = \cos(\omega_0 n) = (1/2)(e^{j\omega_0 n} + e^{-j\omega_0 n})$$

$$\begin{aligned} r_x[k] &= \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} \frac{1}{4}(e^{-j\omega_0 n} + e^{j\omega_0 n})(e^{j\omega_0(n+k)} + e^{-j\omega_0(n+k)}) \\ &= \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} [(1/2)\cos(\omega_0 k) + (1/2)\cos(2\omega_0 n + \omega_0 k)] \\ &= (1/2)\cos(\omega_0 k) \end{aligned}$$

$$P_x(\omega) = (1/2)\tilde{\delta}(\omega \pm \omega_0)$$

So which is it? Energy or Power?

In practice, it doesn't matter:

- real-world signals have a finite amount of samples
- we can only compute an estimate of the autocorrelation
- estimates are always normalized

The autocorrelation in practice

- assume we know $x[n]$ only for $n = 0, 1, \ldots, N-1$
- the empirical *sample autocorrelation* is defined as:

$$\hat{r}_x[k] = \frac{1}{N} \sum_{n=0}^{N-1-|k|} x^*[n+|k|]x[n], \qquad -N < k < N$$

- number of terms in the sum for $\hat{r}_x[k]$ is $N - |k|$
- as $|k| \to N$, the sum of fewer terms is divided by N: *biased* estimate
- the bias compensates for the smaller amount of data
- rule of thumb: $N > 4k_{\text{max}}$, with k_{max} the maximum needed lag

random signals and white noise

Discrete-time random signals

$\eta[n] =$ a new random value at each n

Example: binary random signal

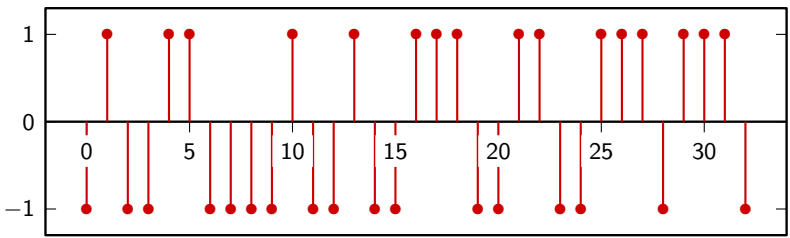
For each new sample, toss a fair coin:

$$\eta[n] = \begin{cases} +1 & \text{if the outcome of the } n\text{-th toss is head} \\ -1 & \text{if the outcome of the } n\text{-th toss is tail} \end{cases}$$

- each sample is either $+1$ or -1 with 50-50 probability
- each sample is statistically independent from all others

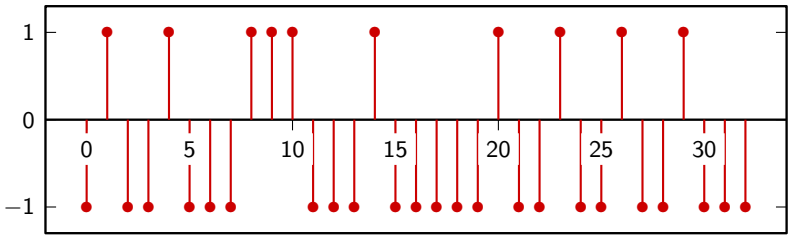
Binary random signal

every time we generate a signal we obtain a different *realization*



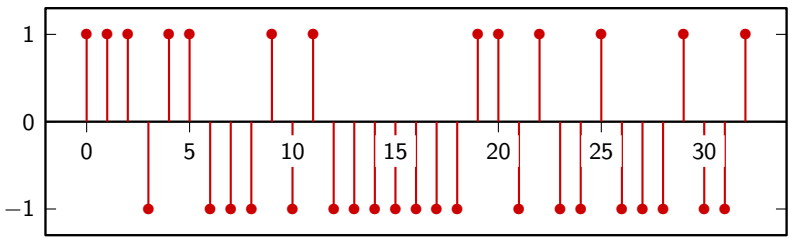
Binary random signal

every time we generate a signal we obtain a different *realization*



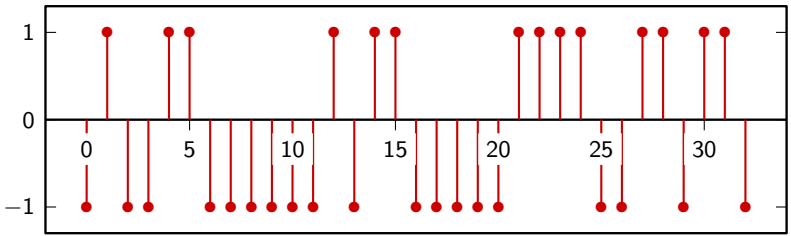
Binary random signal

every time we generate a signal we obtain a different *realization*



Binary random signal

every time we generate a signal we obtain a different *realization*



Properties of the binary random signal

let's look at $2N+1$ samples around $n=0$ for large N:

- the average will go to zero (every $+1$ cancels a -1, and both values equally likely):

$$\frac{1}{2N+1}\sum_{n=-N}^{N}\eta[n]\approx 0$$

- the energy grows linearly with N:

$$\sum_{n=-N}^{N}|\eta[n]|^2=\sum_{n=-N}^{N}1=2N+1$$

- the whole sequence is a power signal since

$$\lim_{N\to\infty}\frac{1}{2N+1}\sum_{n=-N}^{N}|\eta[n]|^2=1$$

Looking for an invariant description

- every time we generate a new binary random signal it looks different
- however, the underlying generation mechanism is always the same (coin toss)
- can we obtain a description of the random signal that does not depend on the actual sequence of sample values?

let's try with the autocorrelation

Autocorrelation of the binary random signal

$$r_\eta[k] = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} \eta[n]\eta[n+k]$$

- if $k = 0$ each term in the sum is $\eta^2[n] = 1$ and thus $r_\eta[0] = 1$
- if $k \neq 0$, because of statistical independence, each term in the sum is

$$\eta[n]\eta[n-k] = \begin{cases} (+1)(+1) = +1 & 25\% \text{ prob.} \\ (-1)(+1) = -1 & 25\% \text{ prob.} \\ (+1)(-1) = -1 & 25\% \text{ prob.} \\ (-1)(-1) = +1 & 25\% \text{ prob.} \end{cases} = \begin{cases} +1 & 50\% \text{ prob.} \\ -1 & 50\% \text{ prob.} \end{cases}$$

as N grows, $r_\eta[k] \to 0$

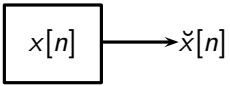
Autocorrelation and PSD of the binary random signal

$$r_\eta[k] = \delta[k]$$

$$P_\eta(\omega) = 1$$

- the binary random signal is self-similar only at lag zero
- the power spectral density is the same at all frequencies
- the binary random signal is an example of *white noise*

Discrete-Time Random Processes



- a discrete-time random process generates an infinite-length sequence of random sample values
- what is the distribution of each sample?
- what are the statistical relations between samples?

Characterization of Discrete-Time Random Processes

- infinite-length sequence of *interdependent* random variables
- a full characterization requires knowing the joint probability density functions

$$f_{x[n_0]x[n_1]\cdots x[n_{k-1}]}(x_0, x_1, \cdots, x_{k-1})$$

  for *all* possible sets of k indices $\{n_0, n_1, \cdots, n_{k-1}\}$ and for *all* $k \in \mathbb{Z}$
- clearly impossible to handle

Manageable Random Processes: Wide-Sense Stationarity

In WSS random processes:

- mean of each sample does not change with time: $\mathrm{E}[x[n]] = m_x$
- the statistical interdependence betweem two samples depends only on their time separation:

$$\mathrm{E}[x[n]x[m]] = c_x[m-n]$$

[WSS is the statistical equivalent to time invariance for systems: the properties of the process do not depend on the absolute observation time, only on the time difference between observations]

Computing expectations, the theory

if x is a random variable with probability density function $f_x(\tau)$

$$\mathrm{E}[x] = \int_{-\infty}^{\infty} \tau \, f_x(\tau) d\tau$$

Computing expectations, the practice

if x is a random variable and we observe M of its realizations $\breve{x}_n$ we can approximate the expected value with the empirical average

$$\mathrm{E}[x] \approx \frac{1}{M} \sum_{n=0}^{M-1} \breve{x}_n$$

as the number of observation grows,

$$\lim_{M \to \infty} \frac{1}{M} \sum_{n=0}^{M-1} \breve{x}_n = \mathrm{E}[x]$$

Computing expectations, the practice

suppose $\mathbf{x}$ is a real-valued, WSS random process and we observe $2N+1$ samples of a realization. The autocorrelation of the observation is

$$r_{x_N}[k] = \frac{1}{2N+1} \sum_{n=-N}^{N} x[n]x[n+k] \approx \mathrm{E}\left[x[n]x[n+k]\right] = c_x[k]$$

as the number of observation grows, the empirical autocorrelation and the probabilistic correlation align

$$r_x[k] = \mathrm{E}\left[x[n]x[n+k]\right]$$

Stochastic signal processing in one slide (WSS processes)

- WSS random processes are equivalent to power signals
- they are characterized by their autocorrelation:

$$r_x[k] = \mathrm{E}\left[x[n]x[n+k]\right] = (\mathcal{R}\mathbf{x} * \mathbf{x})[k]$$

- in the frequency domain, they are described by their spectral density

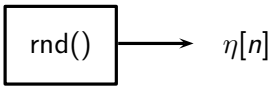
$$P_x(\omega) = \mathrm{DTFT}\left\{\mathbf{r}_x\right\}$$

- filters designed for deterministic signals still work (in magnitude) in the stochastic case

$$\mathbf{y} = \mathbf{h} * \mathbf{x} \quad \Rightarrow P_y(\omega) = |H(\omega)|^2 P_x(\omega)$$

- we lose the concept of phase since we don't know the shape of a realization in advance

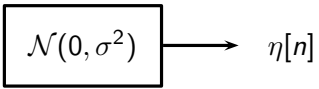
White noise



a discrete-time random sequence $\eta[n]$ is called *white noise* if

- the random samples have zero mean: $\mathrm{E}[x[n]] = 0$ for $n \in \mathbb{Z}$
- the random values have finite variance: $\mathrm{E}\left[|x[n]|^2\right] = r_x[0] = \sigma_\eta^2$
- each sample is independent of all others: $\mathrm{E}[x[n]x[n+k]] = r_x[k] = 0$ for $k \neq 0$

Example: White Gaussian Noise (WGN)



$$f_\eta(\tau) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{\tau^2}{\sigma^2}}$$

Properties of white noise

- statistical independence and finite variance σ_η^2 implies

$$\mathbf{r}_\eta = \sigma_\eta^2 \boldsymbol{\delta}$$

- zero mean implies that, for any statistically independent signal $\mathbf{x}$,

$$\mathbf{r}_{\eta x} = 0$$

PSD of white noise

$$P_\eta(\omega) = \sigma_\eta^2$$

- white noise has equal power at all frequencies
- origin of the name: white light contains energy over the entire visible spectrum
- the PSD does not depend on the distribution of random values, only on the variance

Filtered white noise

- $\eta[n]$ white noise sequence
- $h[n]$ impulse response of stable filter
- $y[n] = (\eta * h)[n]$

$$P_y(\omega) = |H(\omega)|^2 \, \sigma_\eta^2$$

Wold's theorem

- white noise: each random sample is statistically independent, $r_\eta[k] = \delta[k]$
- random signals: random samples are correlated, $r_x[k] \neq \delta[k]$
- any random signal can be obtained by filtering white noise

the autocorrelation as a robust descriptor

Autocorrelation of noisy signal

consider a signal corrupted by independent, additive white noise

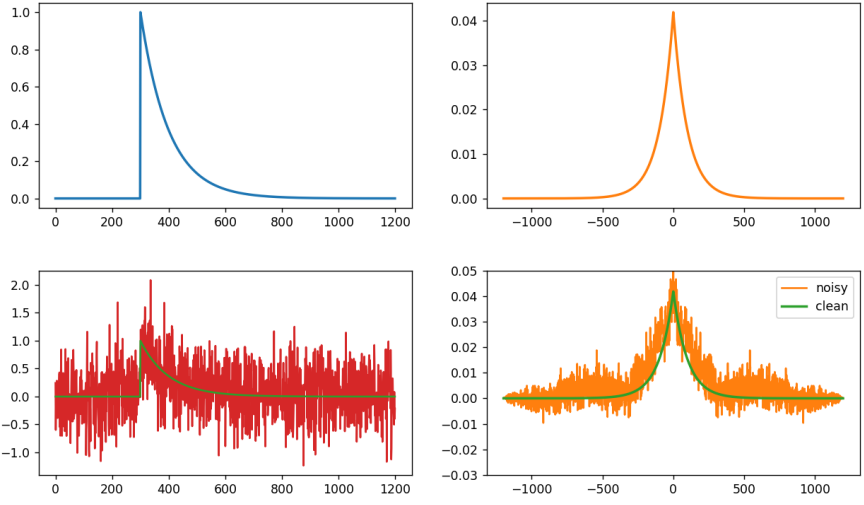
$$\begin{aligned}\mathbf{y} &= \mathbf{x} + \boldsymbol{\eta} \\ \mathbf{r}_y &= \mathcal{R}(\mathbf{x} + \boldsymbol{\eta}) * (\mathbf{x} + \boldsymbol{\eta}) \\ &= \mathbf{r}_x + \mathbf{r}_\eta + \mathbf{r}_{x\eta} + \mathbf{r}_{\eta x}\end{aligned}$$

since signal and noise are independent

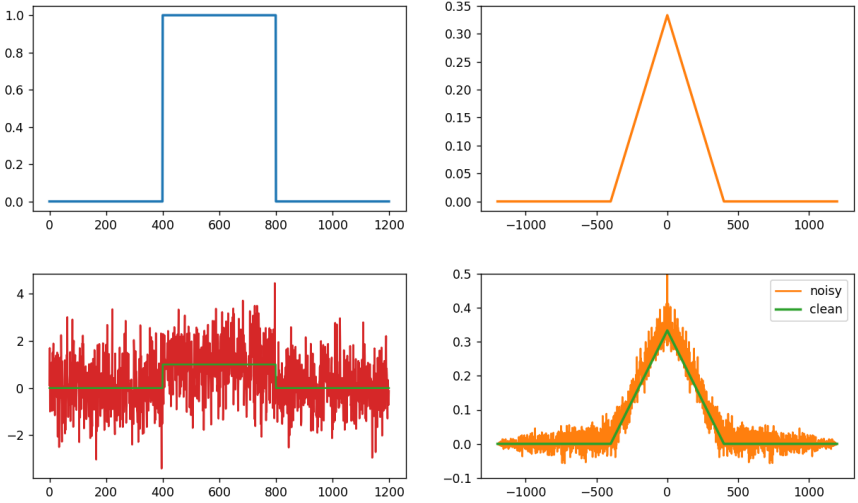
$$= \mathbf{r}_x + \sigma_\eta^2 \boldsymbol{\delta}$$

$$P_y(\omega) = P_x(\omega) + \sigma_\eta^2$$

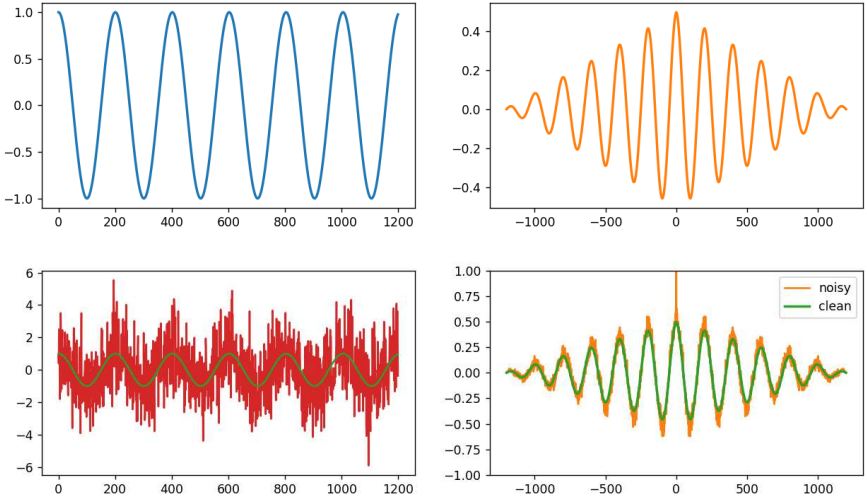
Example: autocorrelation of a decaying exponential



Example: autocorrelation of rectangular pulse



Example: autocorrelation of a sinusoid

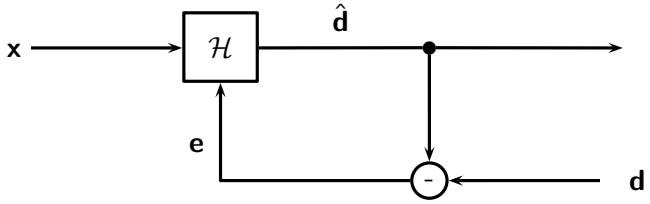


adaptive filters

Goals of adaptive signal processing

- use standard processing tools (filters)
- automatically adapt the filter coefficients as a function of the input data
- implement a robust (stable) adaptation
- be able to "follow" changes in the input

Typical problem setup



- **x**: non-deterministic (unknown) input
- $\mathcal{H}$ adaptive filter with *learned* impulse response **h**
- $\hat{\mathbf{d}} = \mathbf{x} * \mathbf{h}$: filter's output
- **d**: desired (target) output
- $\mathbf{e} = \mathbf{d} - \hat{\mathbf{d}}$: error signal driving the filter's adaptation

Adaptive filters

how can we learn the filter's coefficients so that $\hat{\mathbf{d}} \approx \mathbf{d}$?

- it's not realistic to expect the error signal to be zero ($\mathbf{e} = 0$)...
- ... but we can try to minimize its power (or energy)
- we define a *cost function* expressing $\mathbf{e}$ in terms of $\mathbf{h}$
- finding the optimal filter coefficient becomes a minimization problem

Mean Squared Error (MSE)

$$J(\mathbf{h}) = \|\mathbf{e}\|^2 = \|\mathbf{d} - \mathbf{h} * \mathbf{x}\|^2$$

$$\mathbf{h}_{\text{opt}} = \arg\min_{\mathbf{h}} \{J(\mathbf{h})\}$$

the optimal filter minimizes the squared *norm* of the error

Why a quadratic cost function?

- a quadratic cost function means convex optimization: a global minimum always exists
- expression for the error easily differentiable
- output will be orthogonal to error
- the minimization problem will only involve correlations: robust to noise and randomness

Mean Squared Error minimization

$$J(\mathbf{h}) = \|\mathbf{e}\|^2$$

since the cost function is quadratic and positive, it has a global minimum
to find it, we set to zero the partial derivatives wrt each value of the impulse response:

$$\frac{\partial}{\partial h_i}\|\mathbf{e}\|^2 = 0$$

(to lighten the notation, we'll write h_i instead of $h[i]$)

Different cases, same notation

the MSE may take different forms:

- for energy signals: $\|\mathbf{e}\|^2 = \sum_{n=-\infty}^{\infty} e^2[n]$
- for finite-length signals: $\|\mathbf{e}\|^2 = \frac{1}{N} \sum_{n=-0}^{N-1} e^2[n]$
- for power signals: $\|\mathbf{e}\|^2 = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} e^2[n]$
- for random signals: $\|\mathbf{e}\|^2 = \mathrm{E}\left[e^2[n]\right]$

all these cases are covered by the same notation if we use the autocorrelation: $\|\mathbf{e}\|^2 = r_e[0]$

Different cases, same notation

- in all cases the differentiation can be moved inside the autocorrelation sum
- let's use $\|\mathbf{e}\|^2 = r_e[0] = \mathrm{E}\left[e^2[n]\right]$ for instance

$$\frac{\partial \|\mathbf{e}\|^2}{\partial h_i} = \mathrm{E}\left[\frac{\partial e^2[n]}{\partial h_i}\right]$$

Partial derivatives of the instantaneous squared error

$$\frac{\partial\, e^2[n]}{\partial h_i} = 2e[n]\frac{\partial\, e[n]}{\partial h_i}$$

$$e[n] = d[n] - \sum_k h_k x[n-k]$$

$$\frac{\partial\, e[n]}{\partial h_i} = -x[n-i]$$

$$\frac{\partial\, e^2[n]}{\partial h_i} = 2\left(\sum_k h_k\, x[n-k]x[n-i] - d[n]x[n-i]\right)$$

Averaged partial derivatives

$$
\begin{aligned}
\frac{1}{2}\frac{\partial \|\mathbf{e}\|^2}{\partial h_i} &= \frac{1}{2}\mathrm{E}\left[\frac{\partial e^2[n]}{\partial h_i}\right] \\
&= \mathrm{E}\left[\sum_k h_k\, x[n-k]x[n-i] - d[n]x[n-i]\right] \\
&= \sum_k h_k\, \mathrm{E}\left[x[n-k]x[n-i]\right] - \mathrm{E}\left[d[n]x[n-i]\right] \\
&= (\mathbf{h} * \mathbf{r}_x)[-i] - r_{dx}[-i] \\
&= (\mathbf{h} * \mathbf{r}_x)[i] - r_{xd}[i]
\end{aligned}
$$

Optimal Least Squares solution

- the optimal M-taps filter is found by setting all partial derivatives to zero

$$(\mathbf{h} * \mathbf{r}_x)[i] = r_{xd}[i] \qquad i = 0, 1, 2, \ldots, M-1$$

- that is, we need to solve a linear system of M equations:

$$\sum_{m=0}^{M-1} h[m] r_x[i-m] = r_{xd}[i], \qquad i = 0, 1, \ldots, M-1$$

- this requires the computation of :
  - M values of the input's autocorrelation
  - M values of the cross-correlation between input and desired signal

Optimal Least Squares solution in matrix form

$$\mathbf{h}_{\text{opt}} = \mathbf{R}^{-1}\mathbf{g}$$

$$\mathbf{h} = \begin{bmatrix} h[0] & h[1] & h[2] & \dots & h[M-1] \end{bmatrix}^T$$

$$\mathbf{R} = \begin{bmatrix} r_x[0] & r_x[1] & r_x[2] & \dots & r_x[M-1] \\ r_x[1] & r_x[0] & r_x[1] & \dots & r_x[M-2] \\ r_x[2] & r_x[1] & r_x[0] & \dots & r_x[M-3] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_x[M-1] & r_x[M-2] & \dots & \dots & r_x[0] \end{bmatrix}$$

$$\mathbf{g} = \begin{bmatrix} r_{xd}[0] & r_{xd}[1] & r_{xd}[2] & \dots & r_{xd}[M-1] \end{bmatrix}^T$$

Intuition

- the optimal MSE filter depends only on correlations
- correlations are robust wrt additive noise and changes in signal shape
- implicitly, MSE minimization only relies on spectral *distributions*
- the algorithm works identically for energy, power, and random signals
  (under the hood, we use the appropriate method to compute the correlation values)

Let's take a look at the cost function

$$J(\mathbf{h}) = \|\mathbf{e}\|^2 = r_e[0]$$

$$\begin{aligned}\mathbf{r}_e &= \mathcal{R}\mathbf{e} * \mathbf{e} = \mathcal{R}(\mathbf{d} - \mathbf{h} * \mathbf{x}) * (\mathbf{d} - \mathbf{h} * \mathbf{x}) \\ &= \mathbf{r}_d + (\mathbf{r}_h * \mathbf{r}_x) - (\mathbf{h} * \mathbf{r}_{dx}) - \mathcal{R}(\mathbf{h} * \mathbf{r}_{dx})\end{aligned}$$

$$J(\mathbf{h}) = r_d[0] + (\mathbf{r}_h * \mathbf{r}_x)[0] - 2(\mathbf{h} * \mathbf{r}_{dx})[0]$$

Error surface for M-tap adaptive FIR

FIR convolutions can be expressed as row-column multiplications:

$$(\mathbf{h} * \mathbf{r}_{dx})[0] = \sum_{i=0}^{M-1} h[i] r_{dx}[0-i] = \mathbf{h}^T \mathbf{g}$$

$$\begin{aligned}
\mathbf{h} &= \begin{bmatrix} h_0 & h_1 & \dots & h_{M-1} \end{bmatrix} \\
\mathbf{g} &= \begin{bmatrix} r_{dx}[0] & r_{dx}[-1] & \dots & r_{dx}[-M+1] \end{bmatrix} \\
&= \begin{bmatrix} r_{xd}[0] & r_{xd}[1] & \dots & r_{xd}[M-1] \end{bmatrix}
\end{aligned}$$

Error surface for M-tap adaptive FIR

similarly:

$$
\begin{aligned}
(\mathbf{r}_h * \mathbf{r}_x)[0] &= (\mathcal{R}\mathbf{h} * \mathbf{r}_x * \mathbf{h})[0] \\
&= \sum_{k=0}^{M-1} h[-(0-k)] \sum_{i=0}^{M-1} h[i] r_x[|k-i|] \\
&= \mathbf{h}^T \mathbf{R} \mathbf{h}
\end{aligned}
$$

$$
\mathbf{R} = \begin{bmatrix}
r_x[0] & r_x[1] & r_x[2] & \dots & r_x[M-1] \\
r_x[1] & r_x[0] & r_x[1] & \dots & r_x[M-2] \\
r_x[2] & r_x[1] & r_x[0] & \dots & r_x[M-3] \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
r_x[M-1] & r_x[M-2] & \dots & \dots & r_x[0]
\end{bmatrix}
$$

Error surface

$$J(\mathbf{h}) = r_d[0] + \mathbf{h}^T\mathbf{R}\mathbf{h} - 2\mathbf{h}^T\mathbf{g}$$

- error surface is an elliptic paraboloid with axes proportional to $\sqrt{1/\lambda_i}$, where λ_i are $\mathbf{R}$'s eigenvalues
- the autocorrelation of the input determines the shape of the error surface
- the minimum achievable MSE is for $\mathbf{h}_{\text{opt}} = \mathbf{R}^{-1}\mathbf{g}$:

$$P_e(\mathbf{h}_{\text{opt}}) = r_d[0] - \mathbf{g}^T\mathbf{R}^{-1}\mathbf{g}$$

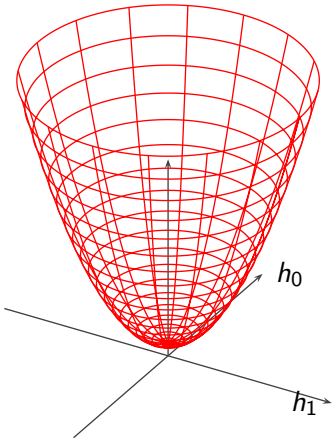
Error surface for $M = 2$

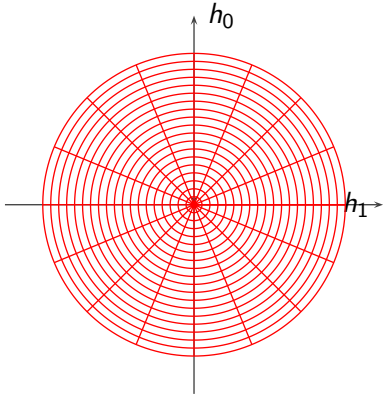
$$J(\mathbf{h}) = r_d[0] + \begin{bmatrix} h_0 & h_1 \end{bmatrix} \begin{bmatrix} r_x[0] & r_x[1] \\ r_x[1] & r_x[0] \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \end{bmatrix} - 2 \begin{bmatrix} h_0 & h_1 \end{bmatrix} \begin{bmatrix} r_{xd}[0] \\ r_{xd}[1] \end{bmatrix}$$

- for $M = 2$ we can plot the error surface
- let's assume $\mathbf{r}_{xd} = 0$ (e.g. input is uncorrelated to desired output)

Error surface for white noise input

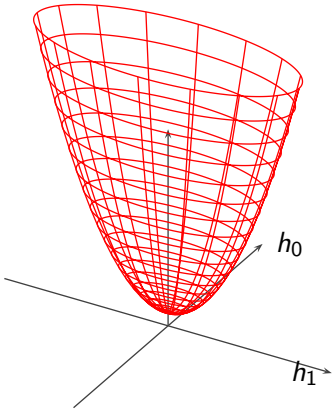
$$\mathbf{r}_x = \delta, \qquad \mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

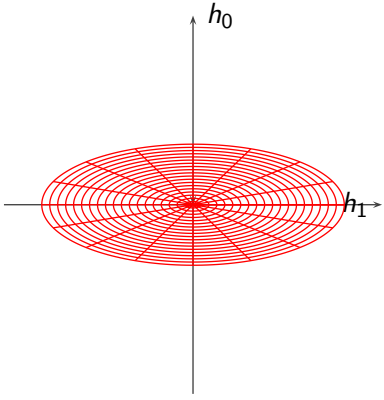




Error surface for correlated input

$$\mathbf{R} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$





So, how is it done in practice?

- the optimal solution requires M values of auto- and cross-correlations
- in practice we compute empirical correlations using the data we have:

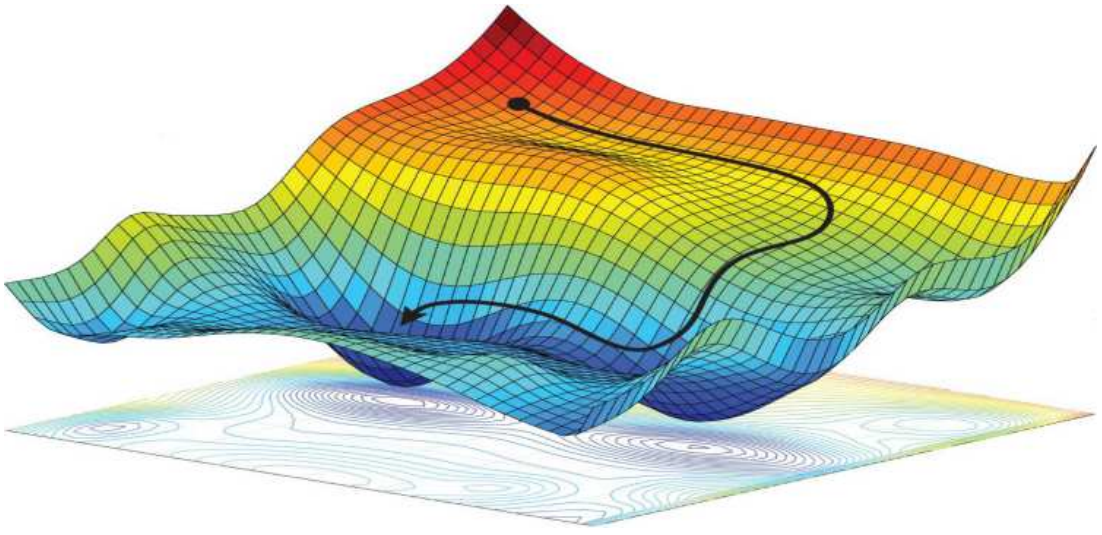
$$r_x[k] \leftarrow \hat{r}_x[k] = \frac{1}{N} \sum_{n=0}^{N-1-|k|} x[n]x[n+|k|]$$

- this requires collecting input data first (post-processing)
- what if we want to adapt in real time?

Error minimization by gradient descent

- most machine learning problems require finding the minimum of a cost function
- a closed form solution exists only in very simple cases (convex optimization)
- in general, the minimum is found *iteratively* via gradient descent
- think of a ball rolling down a bumpy surface until it hits the bottom
- caveat: complicated error surfaces may have local minima!

Gradient descent for nonconvex cost function



Gradient descent

problem setup:

- assume $J(\mathbf{h})$ is a differentiable multivariate function ($\mathbf{h} = \begin{bmatrix} h_0 & h_1 & \dots & h_{M-1} \end{bmatrix}$)
- the gradient for J is the vector:

$$\nabla J(\mathbf{h}) = \begin{bmatrix} \frac{\partial J(\mathbf{h})}{\partial h_0} & \frac{\partial J(\mathbf{h})}{\partial h_1} & \dots & \frac{\partial J(\mathbf{h})}{\partial h_{M-1}} \end{bmatrix}^T$$

to find a (local) minimum with the gradient descent algorithm:

- start with a estimate $\mathbf{h}_0$ for the location of the minimum
- iteratively update the estimate by moving in the direction of steepest descent

$$\mathbf{h}_{n+1} = \mathbf{h}_n - \alpha_n \nabla J(\mathbf{h}_n)$$

- the *learning factor* $\alpha_n < 1$ is a "brake" to prevent overshoots

Gradient descent for convex (MSE) minimization

- for $J(\mathbf{h}) = \|\mathbf{d} - \mathbf{h} * \mathbf{x}\|^2$ the i-th partial derivative is $(\mathbf{h} * \mathbf{x})[i] - r_{xd}[i]$ and so

$$\nabla J(\mathbf{h}) = 2(\mathbf{R}\mathbf{h} - \mathbf{g})$$

- iteration:

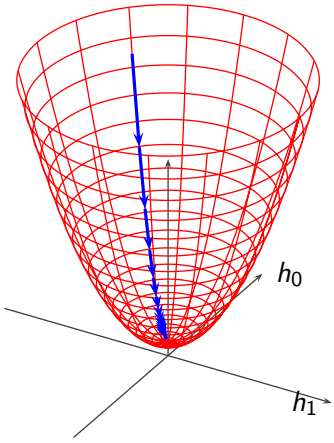
$$\mathbf{h}_{n+1} = (\mathbf{I} - \alpha\mathbf{R})\mathbf{h}_n + \alpha\mathbf{g}$$

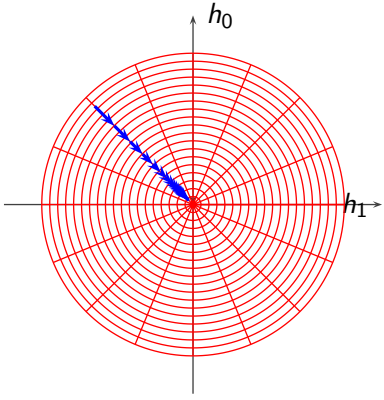
$$\mathbf{h}_n = (\mathbf{I} - \alpha\mathbf{R})^n\mathbf{h}_0 + \alpha\mathbf{g}\sum_{k=0}^{n-1}(\mathbf{I} - \alpha\mathbf{R})^k$$

$$= \mathbf{R}^{-1}\mathbf{g} = \mathbf{h}_{\text{opt}} \qquad \text{if } |\mathbf{I} - \alpha\mathbf{R}| < 1$$

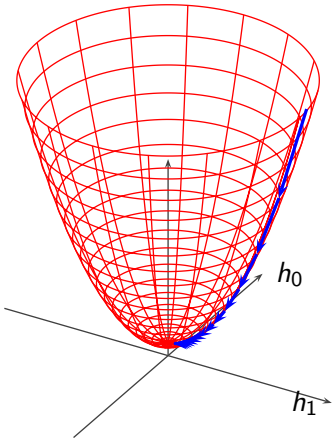
- gradient descent leads to the closed-form solution we found before, but it illustrates the effects of the shape of the error surface and of the learning factor

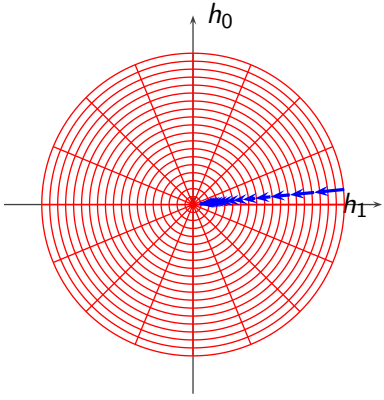
Gradient descent for white noise input



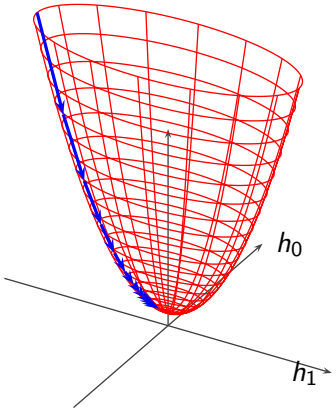


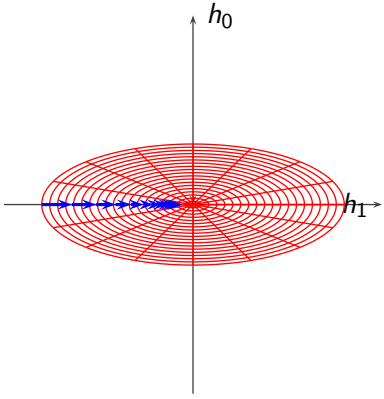
Gradient descent for white noise input



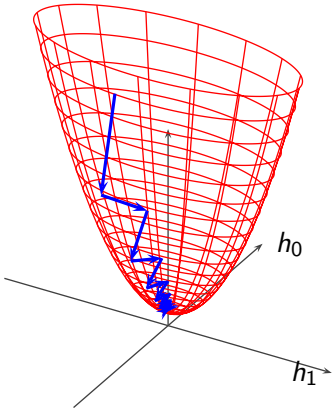


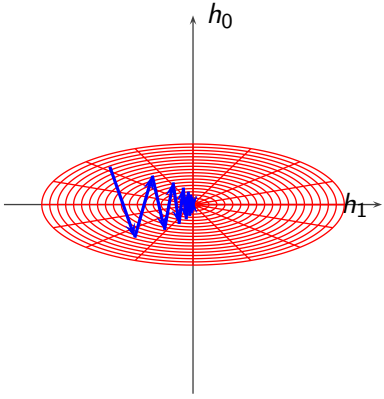
Gradient descent for correlated input: good initial guess



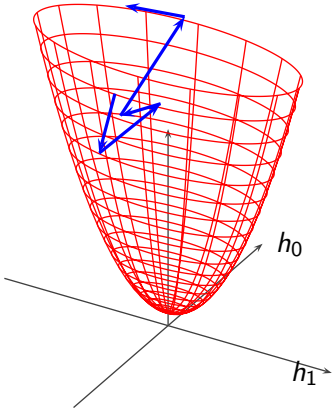


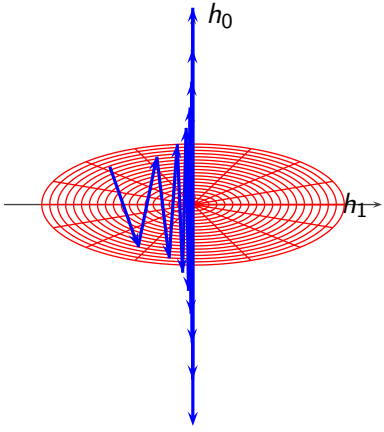
Gradient descent for correlated input: so-so initial guess





Gradient descent for correlated input: learning factor too large





Real-time operation

- so far we still need to gather data first to compute $\mathbf{R}$ and $\mathbf{g}$
- crazy idea: replace the mean squared error by the *instantaneous* squared error
- this does not require averaging and can work in real time
- the result is one of the most successful adaptive DSP algorithms!

Gradient of the instantaneous squared error

instead of using $J(\mathbf{h}) = \|\mathbf{e}\|^2$, use the instantaneous error

$$J(\mathbf{h}) = e^2[n]$$

we computed the partial derivatives of the instantaneous error gradient earlier:

$$\frac{\partial, e^2[n]}{\partial h_i} = 2e[n]\frac{\partial\, e[n]}{\partial h_i} = -2e[n]\, x[n-i]$$

and so the instantaneous gradient is

$$\nabla J(\mathbf{h}) = -2e[n]\mathbf{x}_n$$

with $\mathbf{x}_n = \begin{bmatrix} x[n] & x[n-1] & x[n-2] & \dots & x[n-M+1] \end{bmatrix}^T$

The LMS adaptive filter

- start with an initial guess for the filter coefficients:

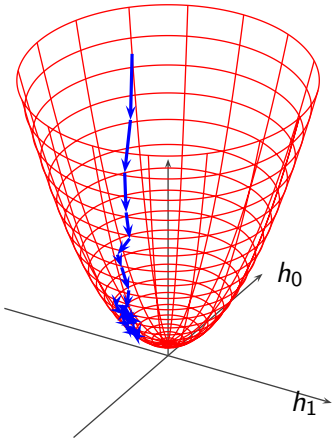
$$\mathbf{h}_0 = \begin{bmatrix} h_0[0] & h_0[1] & \dots & h_0[M-1] \end{bmatrix}^T$$

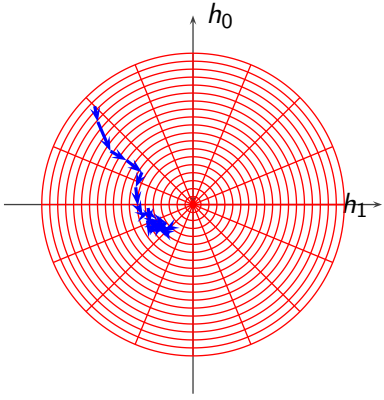
- for each new input sample $x[n]$:
  - compute the filter's output $\mathbf{h}_n^T \mathbf{x}_n = \sum_{k=0}^{M-1} h_n[k] x[n-k]$
  - compute the instantaneous error $e[n] = d[n] - \mathbf{h}_n^T \mathbf{x}_n$
  - update the filter coefficients the gradient $\nabla J(\mathbf{h}_n) = -2e[n]\mathbf{x}_n$

$$e[n] = d[n] - \mathbf{h}_n^T \mathbf{x}_n$$

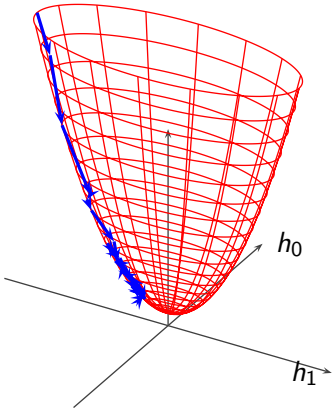
$$\mathbf{h}_{n+1} = \mathbf{h}_n + \alpha_n \, e[n] \, \mathbf{x}_n$$

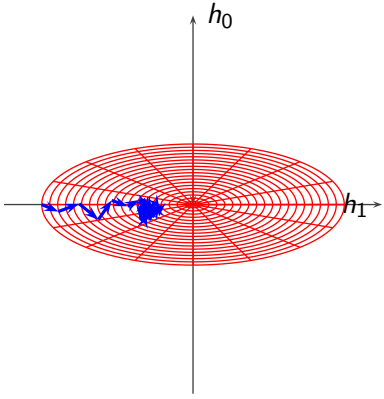
LMS convergence for white noise input





LMS convergence for correlated input

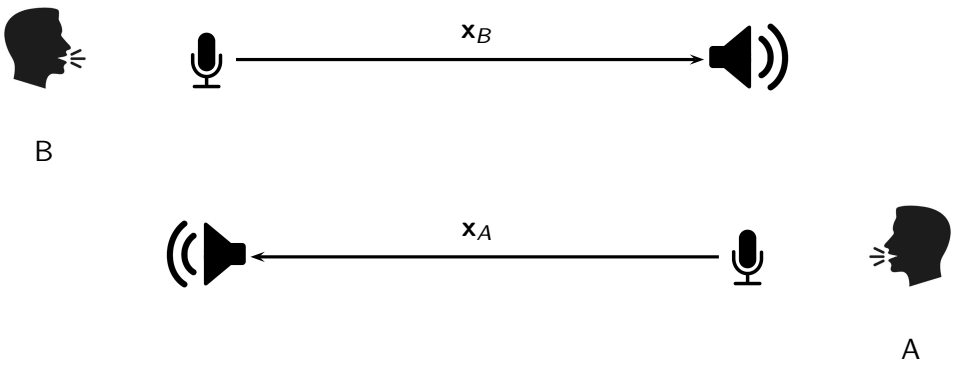




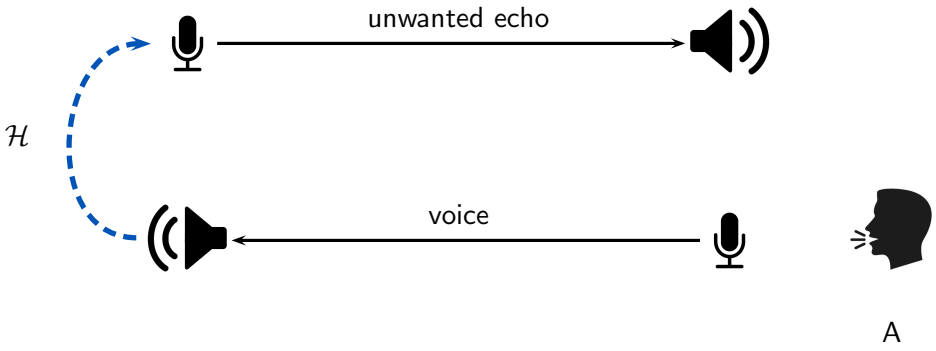
Analysis of the LMS filter

- algorithm is extremely simple and low-cost
- it works very very well
- it keeps adapting all the time: can handle changing conditions
- used in almost all telecommunication devices
- theoretical analysis extremely difficult, however (like AI ;)

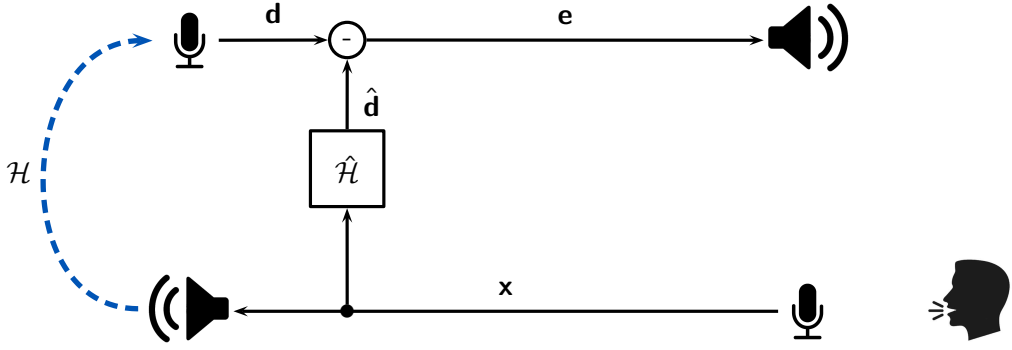
Example: adaptive echo cancellation



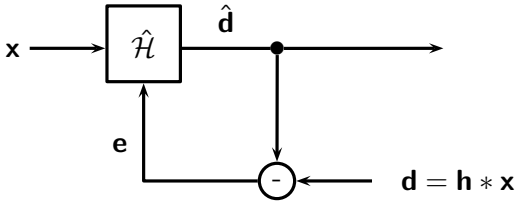
Example: adaptive echo cancellation



Example: adaptive echo cancellation



Echo cancellation as adaptive filtering

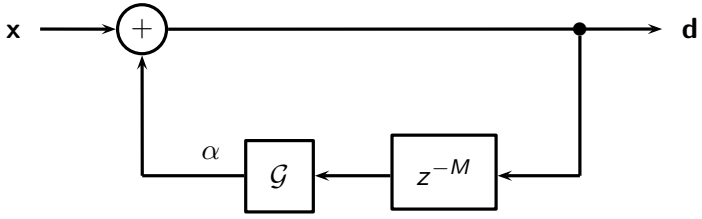


- **x**: speaker's voice (random signal)
- **d**: echo picked up by microphone
- **h**: unknown impulse response of room + loudspeaker + mike
- **e**: residual echo (error signal driving the adaptation)

Training the filter at each end

- most of the time only one person talks at a time
- "desired" signal is local echo, so we can subtract it
- people move, volume changes: $\mathcal{H}$ is time varying!
- use the LMS filter

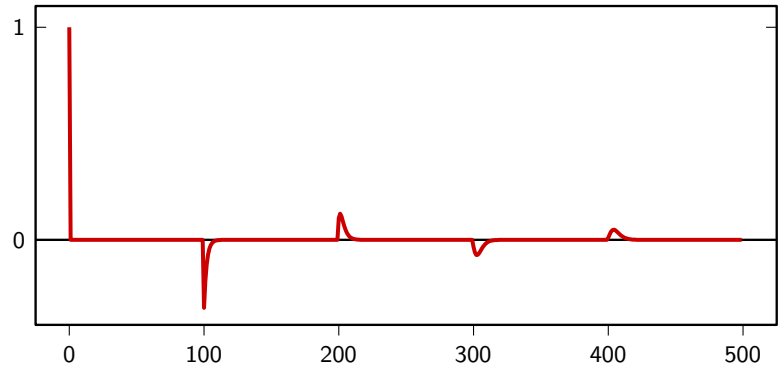
Example: simple echo model



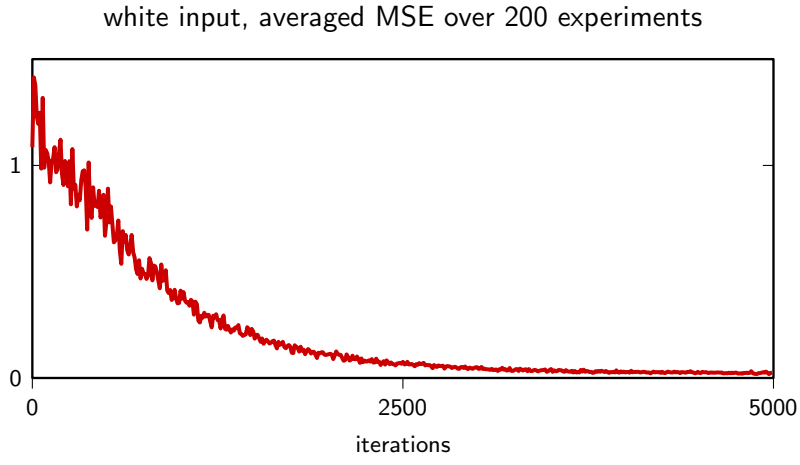
$$G(z) = (1 - \lambda)/(1 - \lambda z^{-1})$$

Echo impulse response

$M = 100, \alpha = -0.8, \lambda = 0.6$



Running the LMS adaptation



LMS can catch up with changes