

02471 Machine Learning for Signal Processing

Introduction

Tommy Sonne Alstrøm

Cognitive Systems section

$$f(x+\Delta x) = \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x)$$

$\Delta \int_a^b \Theta^{\sqrt{17}} + \Omega \delta e^{i\pi} = -1$

$\Sigma!$

∞ χ^2 \gg \approx

$\{2.7182818284\}$ \circ λ

dtucompute.dtu.dk

Outline

- Machine learning and signal processing
- Preliminaries
- Next week

- Is it a signal processing course?
 - What is signal processing?
- Is it a machine learning course?
 - What is machine learning?

What is machine learning and signal processing

Signal Processing

- The study of capturing, processing and manipulating signals
- What is a signal?

Machine Learning

- The study of discovering and extracting information from data
- For data we will use signals

In this course, we will work almost exclusively with time-series data, such as audio and EEG signals, primarily from a data-driven perspective.

- Traditional signal processing does not really care (much) about its input's content
- Traditional machine learning is not very signals-friendly
- MLSP combines both disciplines to perform learning on signals data
- Many examples of MLSP in the real-world

From signal processing to machine learning

- 1948 – Shannon, “A mathematical theory of communication”
- 1950s – Audio engineering drives the field
- 1960s – **Digital** signal processing begins
- 1980s – Silicon empowers DSP
- 1990s – DSP meets Machine Learning

- Sparsity-aware learning
- Information-theoretic learning
- Adaptive filtering
- Change-point detection
- ...

Typical applications

Signal Processing and Machine learning applications

- Sound processing (e.g. hearing aids, spotify etc)
- Images and videos
- Waveforms from telecommunications
- Electrical signals from sensors, EGG, MRI
- Health data that are recorded sequentially
- ...

Examples from the course

- Week 5 – removal of noise from speech
- Week 7 – Use sparse methods to recover signals
- Week 11 – Use Kalman filters to track objects
- Week 13 – Use Kernel method to denoise / identify anomalies

Preliminaries

Digital Signal Processing essentials

- Analog vs digital signals
- Sampling
- Filtering
- Frequency analysis
- Time-frequency analysis (The short-time Fourier transform – STFT)
- Reading material: DSP 1.1–1.7, 1.9, 2.1–2.2, 2.6–2.7, 3.1–3.3, 3.5–3.6, 5.1–5.3.1, 9.1–9.1.6.
- Course DSP primer can be found at
 - <https://github.com/philgzl/dsp-primer>
 - <https://nbviewer.jupyter.org/github/philgzl/dsp-primer/blob/master/notebook.ipynb>

Probability theory essentials

- Be able to manipulate and evaluate expectations.
- Can evaluate joint and condition probabilities.
- Understand variance, Covariance and Correlations
- Reading material: ML 2.1–2.3.

Machine learning essentials

- Understand supervised and unsupervised learning:
 - Cross validation
 - Classification (k -nearest neighbor)
 - Clustering (k -means) – will connect to k -svd.
- Reading material: ML 1.1–1.5, 3.4, 3.12–3.13, 7.1–7.5, 12.6.1.

Matrix derivatives essentials

- Matrix derivatives will be used heavily in the course.
- Material: ML Appendix A.1–A.2.

Lagrange multipliers essentials

- Constrained optimization with Lagrange multipliers.
- Material: ML Appendix C.1

Schedule for the rest of the day

- Review preliminaries as needed.
- Work on Problem set 1.
- TA's will be present until 17.00.

Next week

Week 2 material; 3.1–3.3, 3.5, 3.8–3.11

Parameter estimation:

- Biased and unbiased parameter estimation.
- Bias–Variance dilemma.
- Maximum likelihood estimation.
- Bayesian parameter estimation.

02471 Machine Learning for Signal Processing

Parameter Estimation

Tommy Sonne Alstrøm

Cognitive Systems section

$$f(x+\Delta x) = \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x)$$

Δ $\int_a^b \Theta^{\sqrt{17}} + \Omega \int \delta e^{i\pi} = -1$
Σ $\infty \approx \{2.7182818284\}$ θερτυθιοπσδφγηξκλ
χ² > 000, ≈ λ!

Outline

- Administrative
- Last Week
- Tools needed for parameter estimation
 - Cross validation
 - Matrix derivatives
- Parameter Estimation
- Biased and unbiased estimation
- Maximum likelihood and Bayesian inference
- Next week

Material: ML () 3.1–3.3, 3.5, 3.8–3.11

- **Feedback** from **you** is a critical component for improving both the course and my teaching.
- Most significant feedback
 - Preliminary videos worked great
- Type of feedback
 - Mention one thing that worked?
 - Mention one thing should be improved (both in current lecture and last weeks exercise)?
 - Mention one thing you would change if you gave the lecture.

Last Week

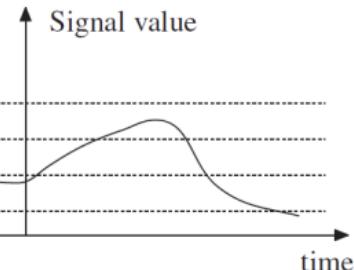
Last Week

From analog to digital signals

Continuous-time signal

Continuous time

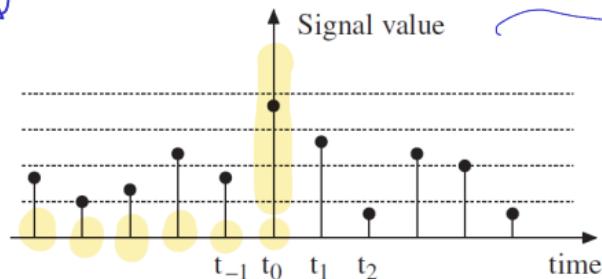
Continuous signal values



Discrete-time signal

Discrete time

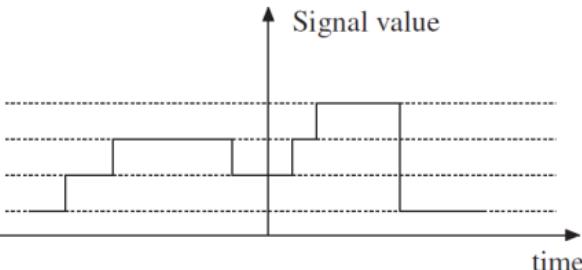
Continuous signal values



Continuous-time signal

Continuous time

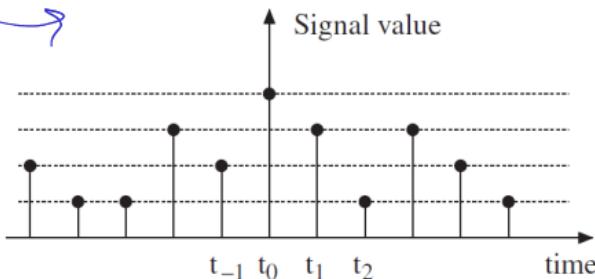
Discrete signal values



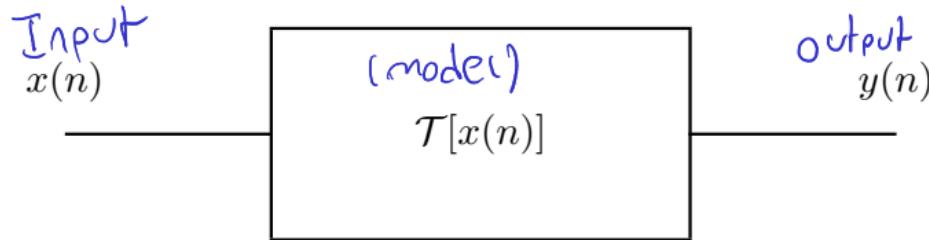
Discrete-time signal

Discrete time

Discrete signal values



Traditional DSP system



This system is restricted to be a linear time-invariant (LTI) system.

General linear system

$$y(n) = \underbrace{- \sum_{k=1}^N a_k y(n-k)}_{\text{Recursive (AR)}} + \underbrace{\sum_{k=0}^M b_k x(n-k)}_{\text{Non-recursive (MA)}}$$

The equation shows the general form of a linear difference equation. It consists of two parts: a recursive part (AR) and a non-recursive part (MA). The recursive part is labeled "IIR" and the non-recursive part is labeled "FIR". The coefficients a_k and b_k are highlighted with yellow circles.

Impulse response and convolution sum of time-invariant linear system (LTI)

If

$$h(n) = \mathcal{T}[\delta(n)]$$

system
v

where

$$\delta(n) = 1 \quad \text{for } n = 0 \tag{1}$$

$$\delta(n) = 0 \quad \text{otherwise} \tag{2}$$

then the output of the system is a convolution sum.

Response of a LTI system

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$$

↑
IR
Input

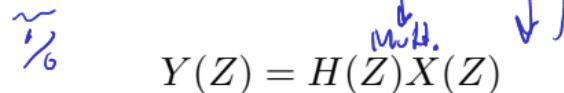
$$= \sum_{k=-\infty}^{\infty} h(k)x(n-k) = h(n) * x(n)$$

The System function

The time-domain expression

$$y(n) = h(n) * x(n)$$

Can be proven to be, in the z -domain


$$\tilde{\tilde{Z}} \quad Y(Z) = H(Z)X(Z)$$

The z-transform and fourier transforms are equal when

$$X(f) = X(z)_{z=e^{j2\pi f}}$$

$$Y(f) = H(f)X(f)$$

Expectation (mean)

$$\mathbb{E}[x] := \int_{-\infty}^{+\infty} x p_x(x) dx$$

A yellow highlighter has been used to emphasize the integral expression. Above the integral, there are three question marks (???) and a small blue arrow pointing to the term $p_x(x)$. Below the integral, the letters "pdf" are written in blue.

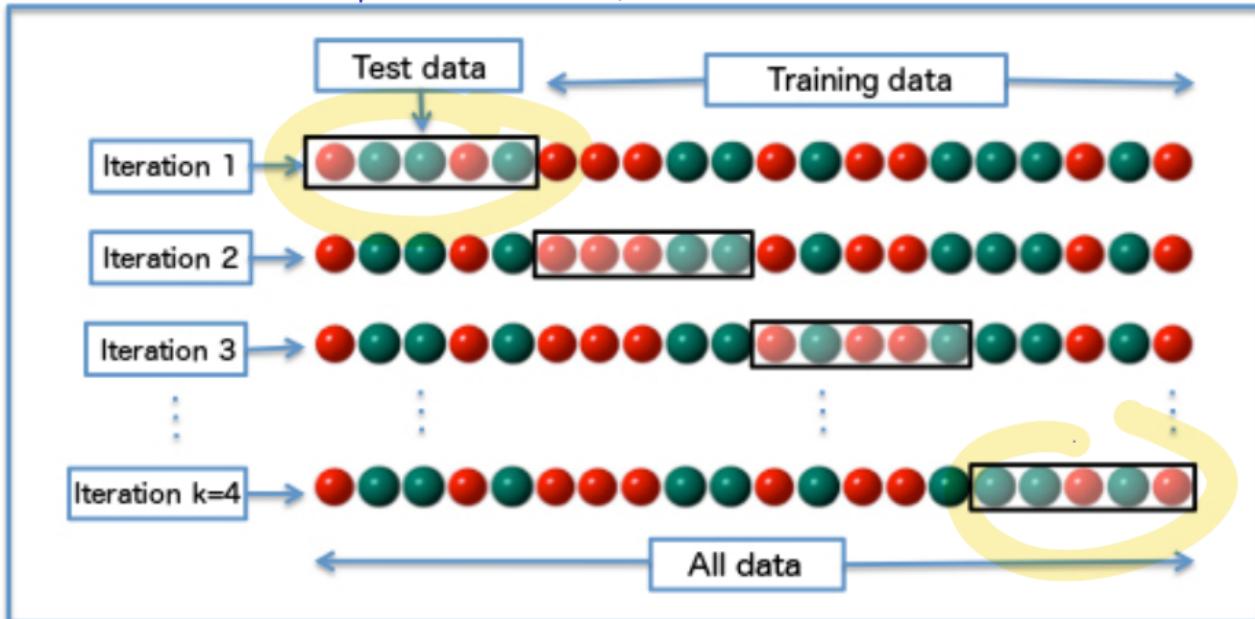
Last weeks summary

- Digital signal processing is processing of quantized discrete-time signals, the two primary tools being:
 - Filtering of signals (changing the content of the signals) using linear time-invariant (LTI) systems.
- The impulse response of a LTI filter fully describes the filter functionality, and the filter is applied by convolution of the input signal with the impulse response.
- We need tools from probability, especially expectations and probability density functions.
- Machine learning is often characterized by data-driven approaches of learning models, as opposed to filter design in DSP.
- We will largely assume you know the basics, such as cross validation, classification (k -nearest neighbor) and k -means.

Tools needed for parameter estimation

Tools needed for parameter estimation

Cross validation



Do you see any problems here if the data is a time-series?

Tools needed for parameter estimation

Cross validation for time-series

74

C. Bergmeir et al. / Computational Statistics and Data Analysis 120 (2018) 70–83

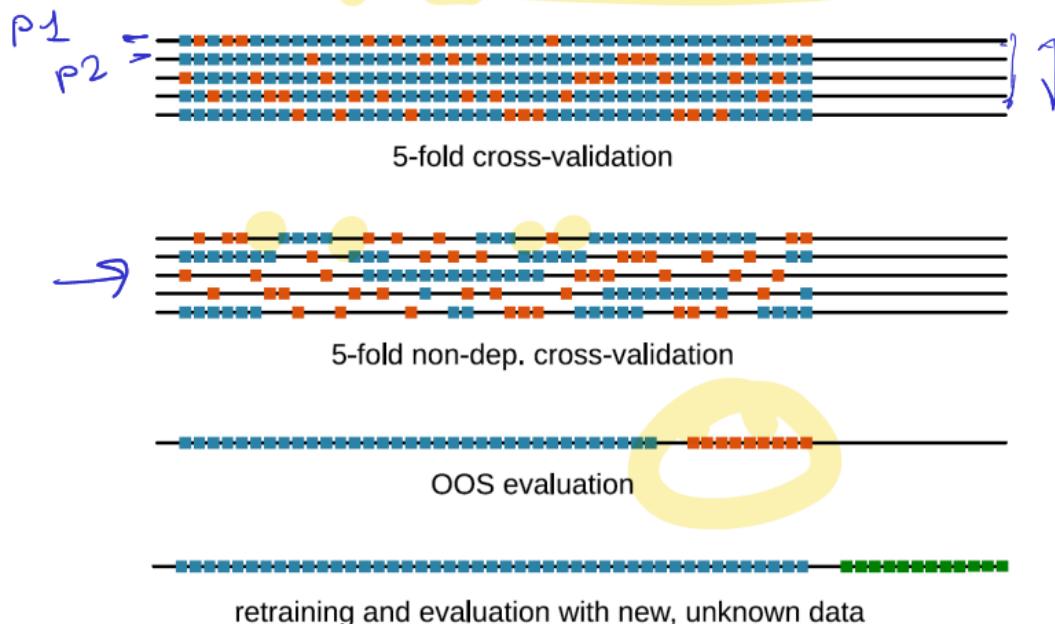


Fig. 2. Training and test sets used for the experiments. The blue and orange dots represent values in the training and test set, respectively. The green dots represent future data not available at the time of model building. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Tools needed for parameter estimation

Vector layout I

The book assumes vectors are column vectors, this is contrary to e.g. DTU Mathematics 1 (01005) where vectors are usually row vectors

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_l \end{bmatrix}^T = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_l \end{bmatrix}$$

In this layout, the gradient of a multivariate function is also a column vector

$$\nabla f(\mathbf{x}) := \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} := \left[\frac{\partial f(\mathbf{x})}{\partial x_1} \quad \frac{\partial f(\mathbf{x})}{\partial x_2} \quad \dots \quad \frac{\partial f(\mathbf{x})}{\partial x_l} \right]^T$$

In other literature, often the gradient is defined as a row vector:

$$\nabla f(\mathbf{x}) := \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} := \left(\frac{\partial f(\mathbf{x})}{\partial x_1} \quad \frac{\partial f(\mathbf{x})}{\partial x_2} \quad \dots \quad \frac{\partial f(\mathbf{x})}{\partial x_l} \right) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1} \quad \frac{\partial f(\mathbf{x})}{\partial x_2} \quad \dots \quad \frac{\partial f(\mathbf{x})}{\partial x_l} \right]$$

Tools needed for parameter estimation

Vector layout II

This layout is called **denominator layout** (also called “Hessian formulation”).

For vector derivatives this layout is important, e.g for **denominator layout**:

$$\frac{\partial \mathbf{b}^T \mathbf{A} \mathbf{x}}{\partial \mathbf{x}} = \mathbf{b}^T \mathbf{A}$$

But for **numerator layout** (also called “Jacobian formulation”) we get

$$\left[\frac{\partial \mathbf{b}^T \mathbf{A} \mathbf{x}}{\partial \mathbf{x}} \right]^T = [\mathbf{A}^T \mathbf{b}]^T$$

Read more about layouts at

https://en.wikipedia.org/wiki/Matrix_calculus

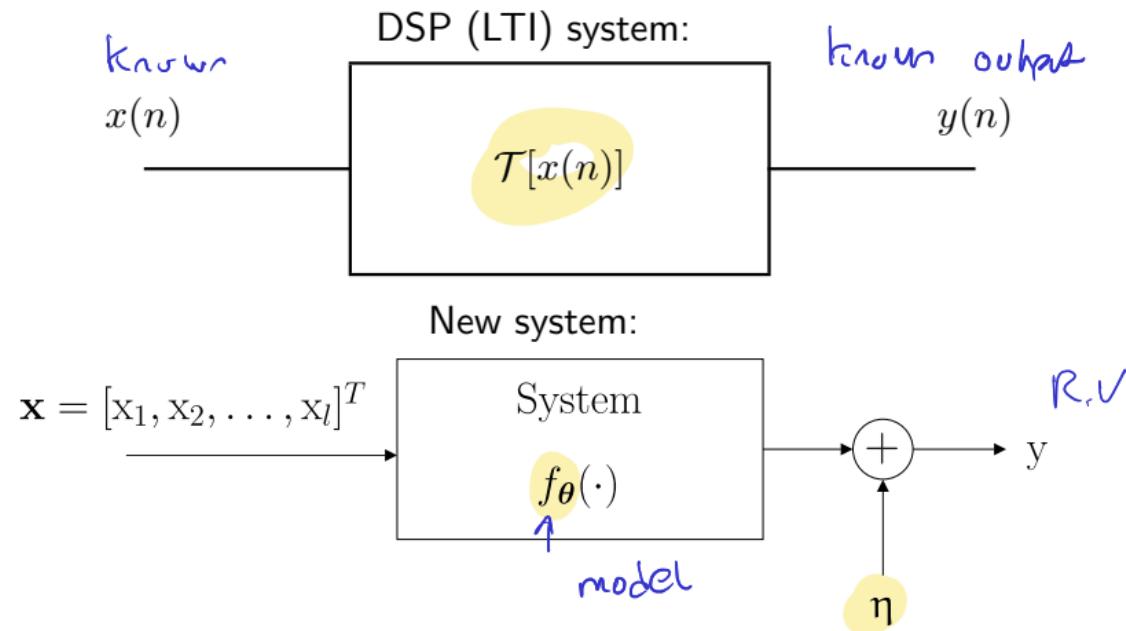
If you are uncomfortable with matrix derivatives, check the video on the course website.

- Cross-validation for time-series requires extra attention to avoid using information from test in train.
- Matrix calculus is equivalent to ordinary calculus and actually makes your life much easier (once learned).
- If you look-up rules outside the book, it is important to be aware of the vector layout convention.
- We will use matrix calculus regularly in the course to derive update rules.
- Check Appendix A for useful rules and identities when doing exercises.

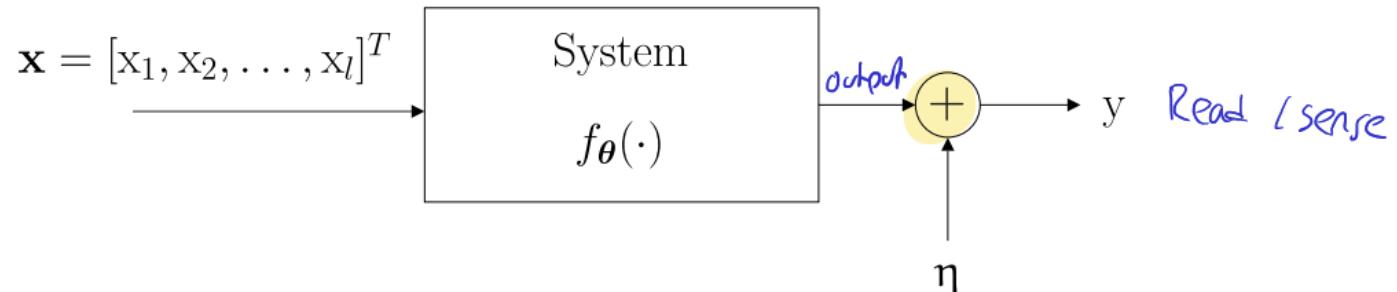
Parameter Estimation

Parameter Estimation

Our new system



What is parameter estimation



This system can be written formally as

$$y = f_{\theta}(\mathbf{x}) + \eta$$

noise

Two tasks must be carried out to use this system:

- Choose the function $f_{\theta}(\cdot) := f(\cdot, \theta)$.
- Once chosen, estimate the parameters θ .

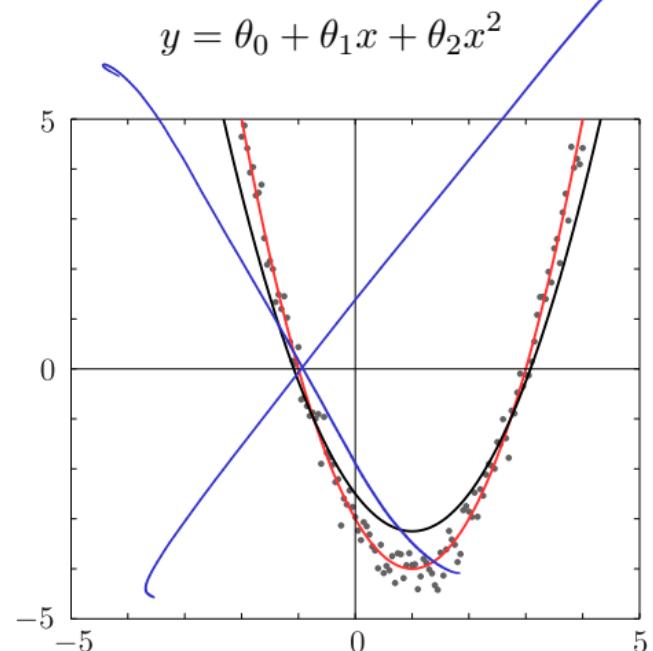
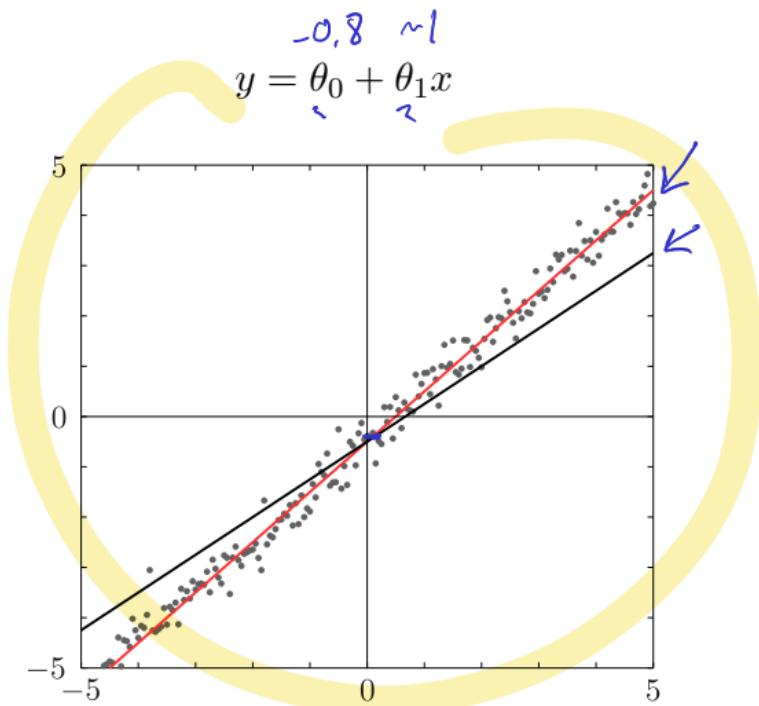
This is the topic for today!!.

Remarks on notation:

- y is a random variable, y is a scalar.
- \mathbf{x} is a random vector, x is a vector
- θ is a parameter vector.
- η is a random variable (noise).

Parameter Estimation Two examples

ML 3.2



Warm-up: What is the value of θ for the two cases (red lines)?

Relation to existing material

Let us rewrite the second order polynomial, $y(x) = \theta_0 + \theta_1 x + \theta_2 x^2$, as follows:

Let

$$\phi(x) := [x^0 \ x^1 \ x^2]^T \in \mathbb{R}^3$$

Then

$$y(x) = \sum_{k=0}^2 \theta_k \phi_k(x) = \theta_0 x^0 + \theta_1 x^1 + \theta_2 x^2$$

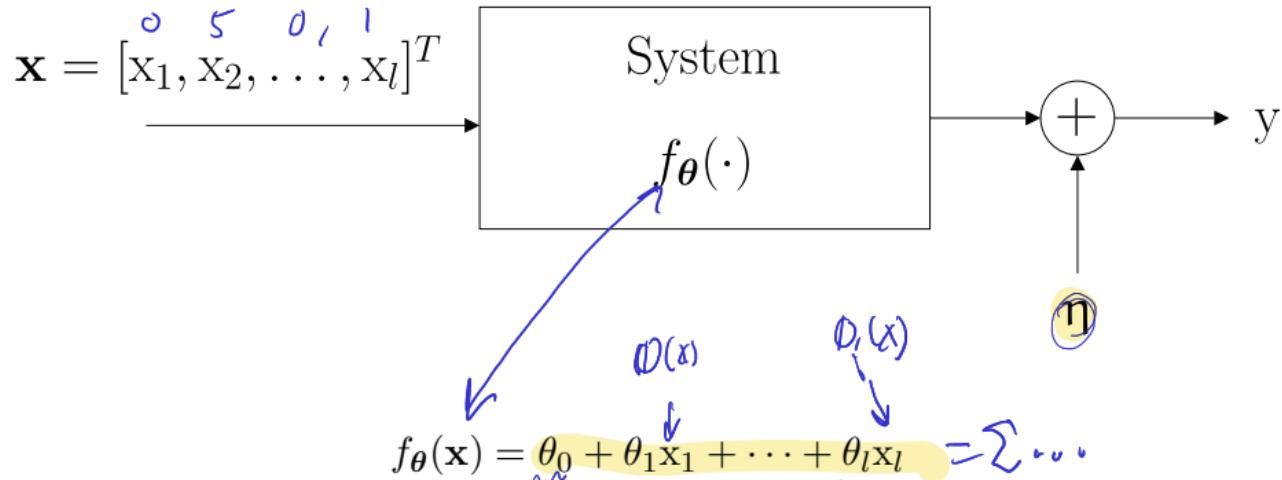
The Fourier series

$$x(t) = \sum_{k=-\infty}^{\infty} c_k e^{j2\pi k F_0 t}$$

Polynomials was one choice of basis, complex exponential was another choice of basis

Parameter Estimation

Linear Regression

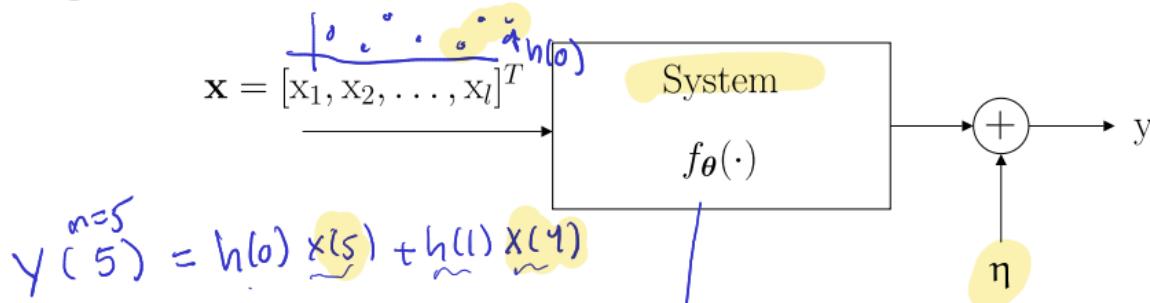


This leads to the linear regression model, written as

$$y = \theta^T \begin{bmatrix} 1 \\ x \end{bmatrix} + \eta$$

Often the "1" is implicit, and the model is written as $y = \theta^T \tilde{x} + \eta$

Linear regression and relation to filters



Linear regression

$$y = \theta^T x + \eta \sim \mathcal{N}(0, \sigma^2)$$

Is the system now a LTI filter? Discuss for a few minutes

Response of a LTI system (reminder from last week)

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

Loss functions – how do we actually estimate θ from data?

Loss function

$$J(\boldsymbol{\theta}) := \sum_{n=1}^N \mathcal{L}(y_n, f_{\boldsymbol{\theta}}(\mathbf{x}_n))$$

DESIGN PHASE

To do parameter estimation, we minimize the loss

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

A popular choice is the Least-Squares loss function

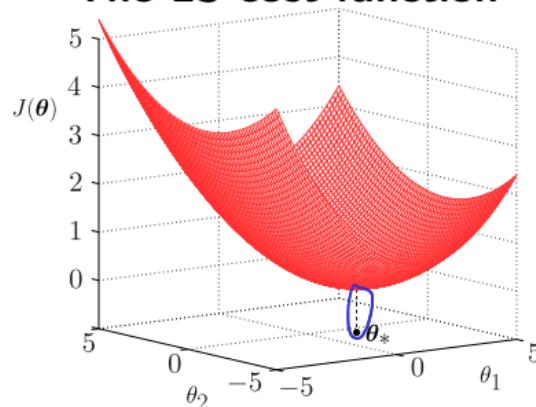
Least-Squares (LS) loss function

$$\mathcal{L}(y_n, f_{\boldsymbol{\theta}}(\mathbf{x}_n)) = (y_n - f_{\boldsymbol{\theta}}(\mathbf{x}_n))^2$$

$$J(\boldsymbol{\theta}) = \sum_{n=1}^N (y_n - f_{\boldsymbol{\theta}}(\mathbf{x}_n))^2$$

Parameter Estimation

The LS cost function



IUL 3.3

From Appendix A.1, we have the following

$$\mathcal{N}(\mu, \sigma^2)$$
$$z = n_1 + n_2 + n_3 + \dots + n_L$$

$$\frac{\partial \mathbf{a}^T \mathbf{x}}{\partial \mathbf{x}} = \frac{\partial \mathbf{x}^T \mathbf{a}}{\partial \mathbf{x}} = \mathbf{a}$$
$$\frac{\partial \mathbf{x}^T \mathbf{A} \mathbf{x}}{\partial \mathbf{x}} = (\mathbf{A} + \mathbf{A}^T) \mathbf{x}$$

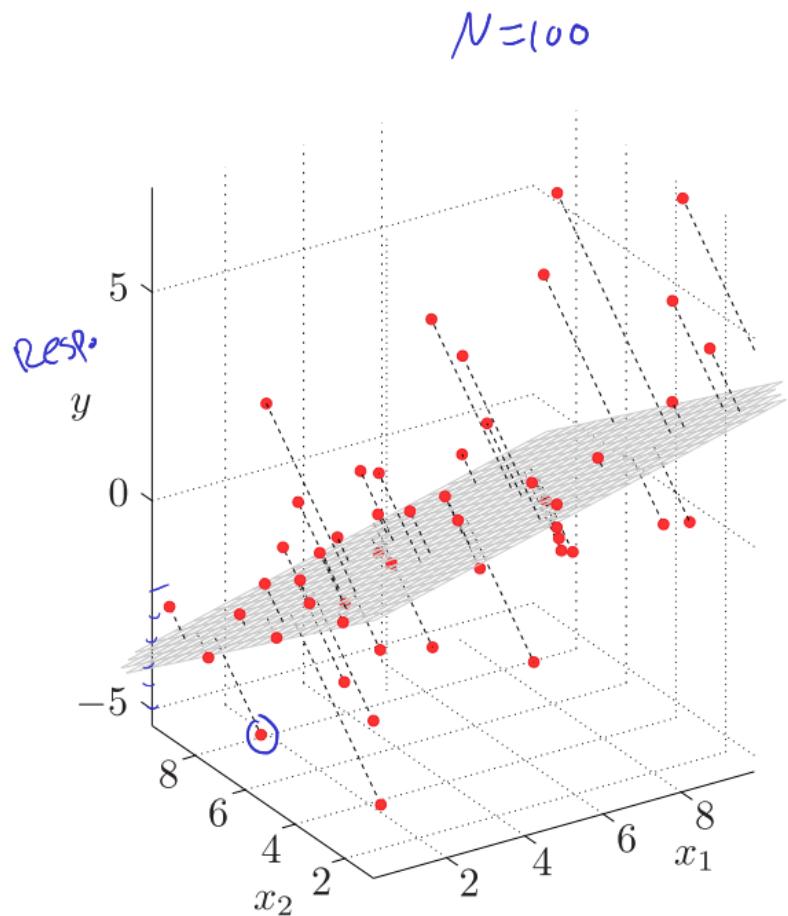
LS Estimate

$$\theta_* = (X^T X)^{-1} X^T \mathbf{y}$$

To obtain the LS estimate, put the derivative, $\frac{d}{d\theta} J(\theta) = 0$ and isolate for θ .

Parameter Estimation Example

$$X = \begin{bmatrix} 2 \\ 4 \\ \vdots \\ \vdots \\ 100 \end{bmatrix}$$



- Parameter estimation is concerned with how to estimate parameters of a model.
- We use loss functions to measure how well we estimate the parameters.
- To keep it simple, we will work with a linear model.
- Linear regression in statistics/ML corresponds to linear filtering in DSP.
- The estimated parameters are uncertain, if we sample a new i.i.d dataset, the parameter estimate change, hence the parameters are themselves stochastic.

Biased and unbiased estimation

Consider; the observations are random; our dataset $\mathcal{D} = (\mathbf{y}, \mathbf{X})$ will change, our observed data is only one realization of the underlying stochastic process (we will get back to this next week). Even if x is held constant, y will still change every time we "measure":

$$\mathbf{y} = \boldsymbol{\theta}^T \mathbf{x} + \eta$$

one vector

Our estimator is random in itself

Estimator of the unknown vector θ

$$\hat{\boldsymbol{\theta}} = g(\mathbf{y}, \mathbf{X})$$

↑
estimate of RV

Biased and unbiased estimation

Definitions

$$\frac{1}{N} \quad \text{or} \quad \frac{1}{N-1}$$

$\hat{\theta}$ is the estimate of the optimal value θ_o

Bias definition

$$\text{Bias}[\hat{\theta}] = \mathbb{E}[\hat{\theta}] - \theta_o = 0 \Rightarrow \mathbb{E}[\hat{\theta}] = \theta_o \quad \rightarrow 1\%$$

That means an unbiased estimator ($\text{Bias}[\hat{\theta}] = 0$) attains the value $\mathbb{E}[\hat{\theta}] = \theta_o$.

Variance definition

$$\text{Var}[\hat{\theta}] = \mathbb{E} \left[(\hat{\theta} - \mathbb{E}[\hat{\theta}])^2 \right] \quad 7\%$$

Mean Squared Error definition

$$\text{MSE}[\hat{\theta}] = \mathbb{E} \left[(\hat{\theta} - \theta_o)^2 \right] \quad 92\%$$

Which of these should we minimize?

Biased and unbiased estimation Bias–Variance decomposition



The mean squared error can be decomposed as follows

$$\begin{aligned}\text{MSE}[\hat{\theta}] &= \mathbb{E}\left[\left(\hat{\theta} - \theta_o\right)^2\right] \\ &= \mathbb{E}\left[\left(\left(\hat{\theta} - \mathbb{E}[\hat{\theta}]\right) + \left(\mathbb{E}[\hat{\theta}] - \theta_o\right)\right)^2\right]\end{aligned}$$

ps 2

The cross-terms vanish, hence we are left with the following result

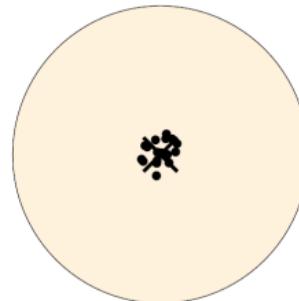
Bias–Variance decomposition

$$\text{MSE}[\hat{\theta}] = \underbrace{\mathbb{E}\left[\left(\hat{\theta} - \mathbb{E}[\hat{\theta}]\right)^2\right]}_{\text{Variance}} + \underbrace{\left(\mathbb{E}[\hat{\theta}] - \theta_o\right)^2}_{\text{Bias}^2}$$

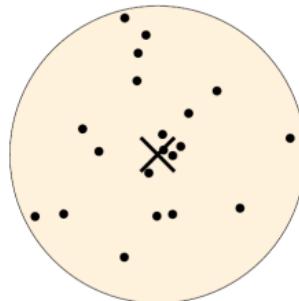
Biased and unbiased estimation

Examples of estimations

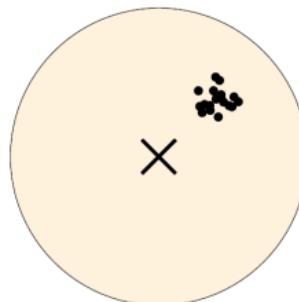
Low bias low variance



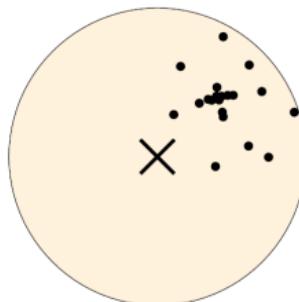
Low bias high variance



high bias low variance



High bias high variance



Biased and unbiased estimation

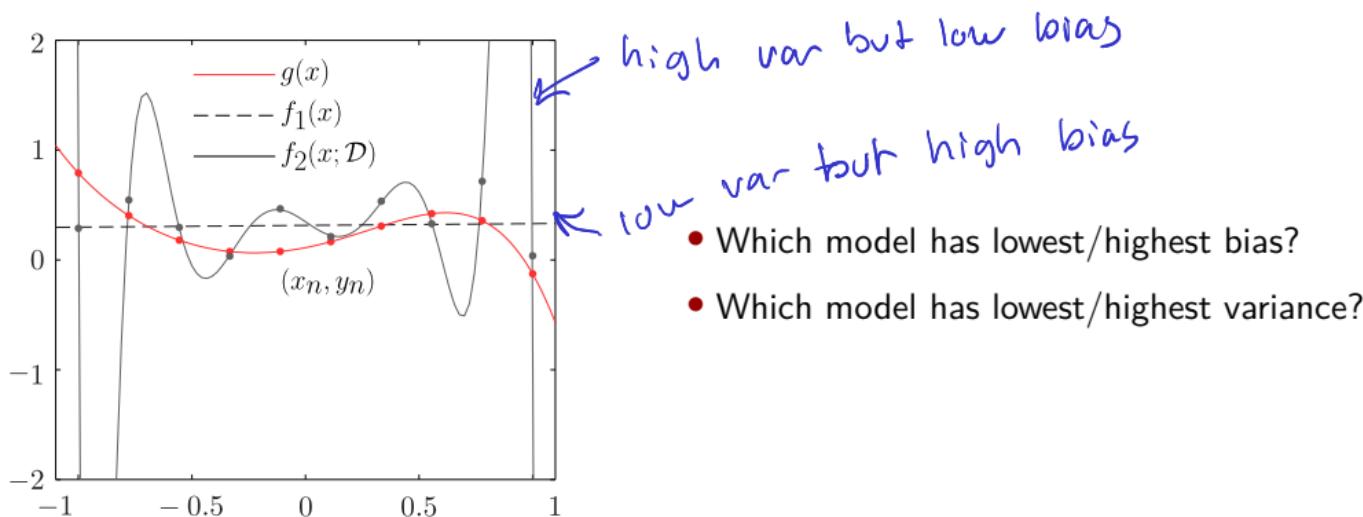
Bias–Variance dilemma

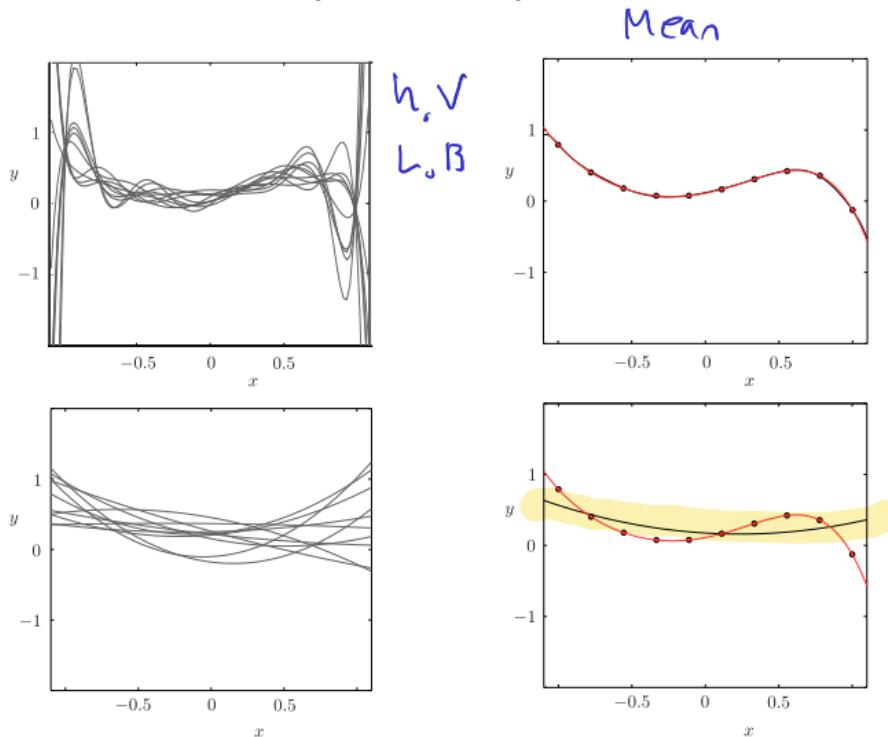
PS 2



Bias–Variance tradeoff

$$\mathbb{E}_{\mathcal{D}} \left[(f(\mathbf{x}; \mathcal{D}) - \mathbb{E}[y|\mathbf{x}])^2 \right] = \underbrace{\mathbb{E}_{\mathcal{D}} \left[(f(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[f(\mathbf{x}; \mathcal{D})])^2 \right]}_{\text{Variance}} + \underbrace{(\mathbb{E}_{\mathcal{D}}[f(\mathbf{x}; \mathcal{D})] - \mathbb{E}[y|\mathbf{x}])^2}_{\text{Bias}^2}$$



Bias Variance tradeoff examples (figure 3.8)

Unbiased estimation

How can we improve our estimation ?

Let us assume we have L datasets and consider an unbiased estimator $\hat{\theta}$, where, for each dataset we have the estimate $\hat{\theta}_i$.

$$\hat{\theta}^{(L)} = \frac{1}{L} \sum_{i=1}^L \hat{\theta}_i$$

Then $\hat{\theta}^{(L)}$ is an unbiased estimator with MSE

$$\text{MSE}[\hat{\theta}^{(L)}] = \frac{1}{L} \text{MSE}[\hat{\theta}_i]$$

You will derive this result in the exercise.

What is the "issue" with this result?

ex 2.2

Biased estimation – an alternative route to reduce the MSE

Define the following biased estimator, where $\hat{\theta}_u$ is an unbiased estimator, $\alpha \in \mathbb{R}$

$$\hat{\theta}_b = (1 + \alpha)\hat{\theta}_u$$

Then, if we require $\text{MSE}(\hat{\theta}_b) < \text{MSE}(\hat{\theta}_u)$, we arrive at

$$-\frac{2\text{MSE}(\hat{\theta}_u)}{\text{MSE}(\hat{\theta}_u) + \boxed{\theta_o^2}} < \boxed{\alpha} < 0$$

ex 2.1.3

And in addition, we can show that the norm of the biased estimator is smaller than the norm of the unbiased estimator

$$|\text{MSE}(\hat{\theta}_b)| < |\text{MSE}(\hat{\theta}_u)|$$

You will derive this result on your own in the exercise.

Ridge regression

The Ridge regression loss function is a norm shrinking loss:

Ridge regression

$$\mathcal{L}(\boldsymbol{\theta}, \lambda) = \sum_{n=1}^N (y_n - \boldsymbol{\theta}^T \mathbf{x}_n)^2 + \lambda \|\boldsymbol{\theta}\|_2^2$$

It can be shown (ML problem 3.12) that $\text{MSE}(\hat{\boldsymbol{\theta}}_b) < \text{MSE}(\hat{\boldsymbol{\theta}}_u)$ if

$$\lambda \in [0, \infty[,$$

$$\theta_o^2 \leq \frac{\sigma_\eta^2}{N}$$

$$\lambda \in \left[0, \frac{2\sigma_\eta^2}{\theta_o^2 - \frac{\sigma_\eta^2}{N}} \right],$$

$$\theta_o^2 > \frac{\sigma_\eta^2}{N}$$

- Bias-variance trade-off is crucial to understand, and offers direct interpretation of the source of errors and behavior of models.
- We have theoretic evidence that bias will lower MSE.
- The more data we have, the less regularization we need (the smaller the λ).

Maximum likelihood and Bayesian inference

Maximum likelihood estimation and Bayesian inference

Bayes theorem

$$p(\boldsymbol{\theta}|\mathcal{X}) = \frac{p(\mathcal{X}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{X})}$$

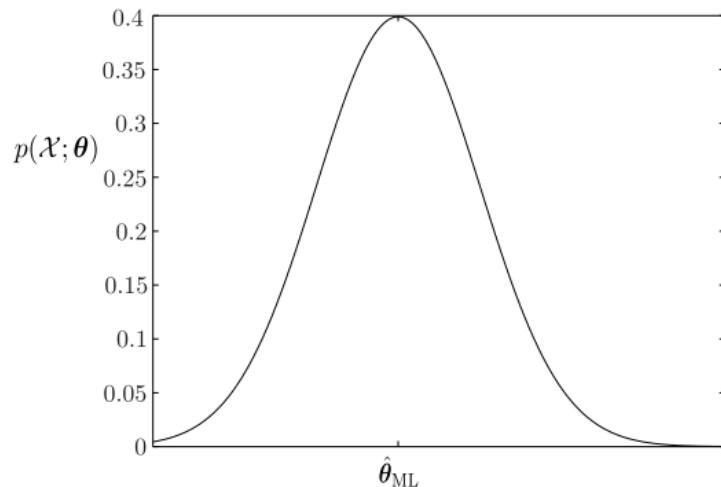
Taking the log we get

$$\begin{aligned}\ln p(\boldsymbol{\theta}|\mathcal{X}) &= \ln p(\mathcal{X}|\boldsymbol{\theta}) + \ln p(\boldsymbol{\theta}) - \ln p(\mathcal{X}) \\ \underbrace{\ln p(\boldsymbol{\theta}|\mathcal{X})}_{\text{log posterior}} &\propto \underbrace{\ln p(\mathcal{X}|\boldsymbol{\theta})}_{\text{log likelihood}} + \underbrace{\ln p(\boldsymbol{\theta})}_{\text{log prior}}\end{aligned}$$

- Using a normal distribution as likelihood leads to LS Estimate (ML example 3.7).
- Using a normal distribution as prior leads to Ridge regression (ML example 3.8).

Maximum likelihood and Bayesian inference

Maximum likelihood illustrated

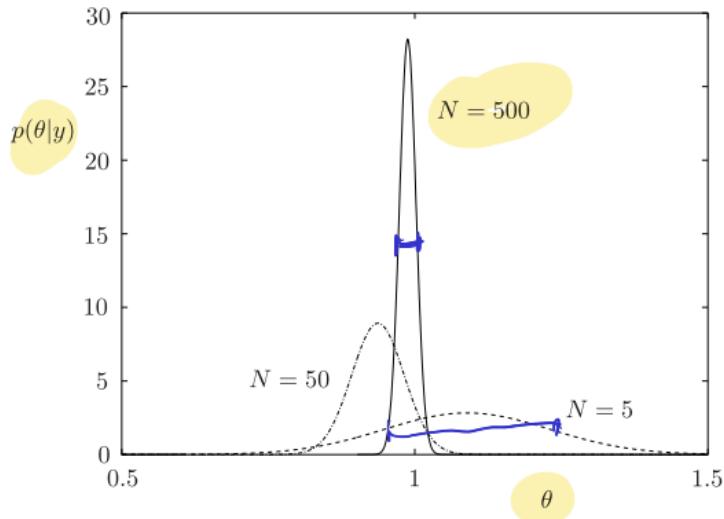


Maximum likelihood estimation

$$\hat{\theta}_{\text{ML}} = \arg \max_{\theta} p(\mathcal{X} | \theta)$$

Maximum likelihood and Bayesian inference

Bayesian inference illustrated

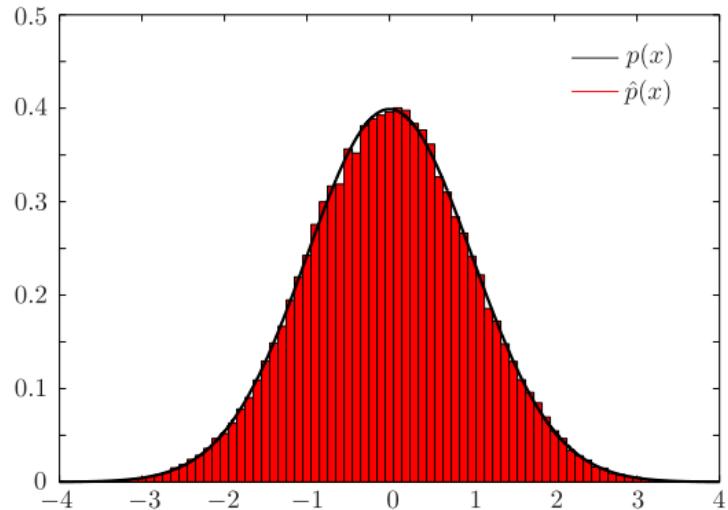
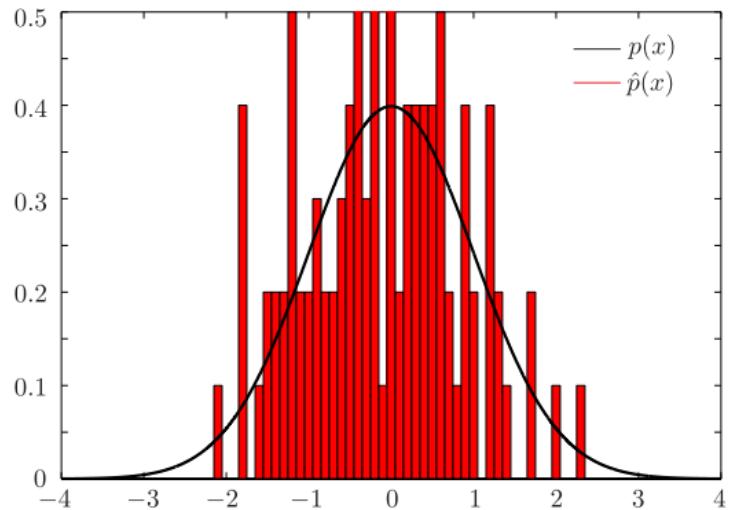


Maximum posterior estimation

$$\hat{\theta}_{\text{MAP}} = \arg \max_{\theta} p(\theta|\mathcal{X})$$

Maximum likelihood and Bayesian inference

Bayesian inference using sampling



Outside the scope of this course – taught in

- 02443 Stochastic Simulation.
- 02477 Bayesian machine learning

Maximum likelihood and Bayesian inference

Lecture Summary



- Matrix calculus is equivalent to ordinary calculus, but can make your life easier.
- Parameter estimations is concerned with estimating parameters of the function, using cost functions. But the estimations are random as well.
- We can do parameter estimation by loss functions (maximum likelihood, or Bayesian inference.) *week 10*
- Maximum likelihood offers an elegant way to understand the noise model.
- Bayesian inference offers an elegant way to learn the uncertainty of the parameter estimates.
- Bias-variance trade-off is crucial to understand, and offers direct interpretation of the source of errors and behavior of models.

Next week

$2_0 1 \rightarrow 2_0 3$

Week 3 material; 2.4, 4.1–4.3, 4.5–4.7

Linear filtering:

- Stochastic Processes.
- The Wiener filter (linear filtering).
- Typical applications of the Wiener filter.

02471 Machine Learning for Signal Processing

Stochastic processes and linear filtering

Tommy Sonne Alstrøm

Cognitive Systems section

$$f(x+\Delta x) = \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x)$$

Stochastic processes and linear filtering

Tommy Sonne Alstrøm

Department of Applied Mathematics and Computer Science

DTU Compute

Mathematical symbols: Θ , \int_a^b , $\sqrt{17}$, δ , $e^{i\pi} = -1$, ∞ , Σ , $!$, λ , χ^2 , \gg , \approx , \circ , $\{2.7182818284\}$.

Outline

- Course admin
- Last Week
- Frequency analysis
- The Wiener filter
- Typical applications of filtering
- Linear filtering using mean-squared error
- Stochastic Processes
- Next Week

Material: ML 2.4, 4.1–4.3, 4.5, 4.7

Administrative notes

- Next week lecture (21/9) will be video lectures instead, and TA's will be present 13.00-17.00
- Problem set 1 deadline postponed 4 days, new date due 26/9: Thursday 21
 - Pass/No pass. no pass attempts can be handed in again.
 - In groups of 2 (or 3 max), everyone needs to hand in individually, but clearly write on DTU Learn who you worked with. In this case, you can hand in identical files.
 - ONLY upload PDFs.

- Feedback: most significant items to improve:
 -
- Type of feedback
 - Mention one thing that worked?
 - Mention one thing should be improved (both in current lecture and last weeks exercise)?
 - Mention one thing you would change if you gave the lecture.
- Some people are dropping the course (as expected). If you do, please let me know the reasons.
Approx 25 people left the course.

Last Week

Two commonly used rules

$$\frac{\partial \mathbf{a}^T \mathbf{x}}{\partial \mathbf{x}} = \frac{\partial \mathbf{x}^T \mathbf{a}}{\partial \mathbf{x}} = \mathbf{a}$$
$$\frac{\partial \mathbf{x}^T \mathbf{A} \mathbf{x}}{\partial \mathbf{x}} = (\mathbf{A} + \mathbf{A}^T) \mathbf{x}$$

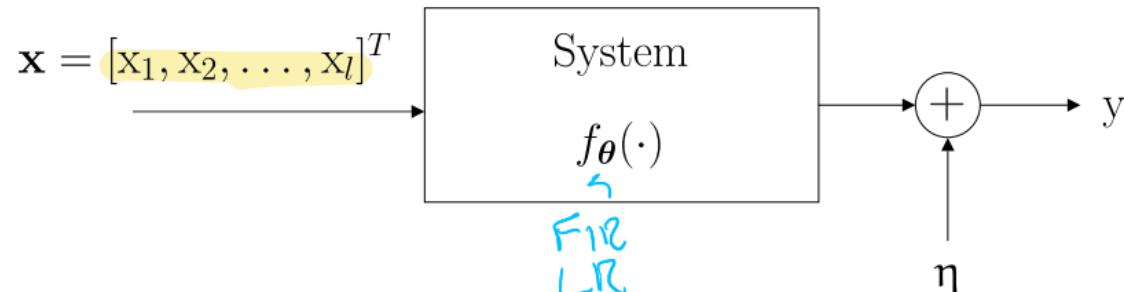
Things to remember

- It is an extension of normal calculus. The derivatives are simply organized in vectors and matrices.
- When you look-up rules, it is important to be aware of the vector layout convention.
- We will use matrix calculus regularly in the course to derive update rules.
- Check Appendix A for useful rules and identities when doing exercises.
- $(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T$, $(\mathbf{B}^T)^T = \mathbf{B}$!

~~z ari~~

Last Week

What is parameter estimation



We considered the general system

$$y = f_{\theta}(\mathbf{x}) + \eta$$

And limited the examples to the linear model that lead to the linear regression model

$$f_{\theta}(x) = \theta_0 + \theta_1 x_1 + \dots + \theta_l x_l$$

$$y = \boldsymbol{\theta}^T \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix} + \eta$$

Remarks on notation:

- y is a random variable, y is a scalar.
- \mathbf{x} is a random vector, x is a vector
- θ is a parameter vector.
- η is a random variable (noise).

Estimate θ from data?

Approaches

- Point estimate of θ , denoted $\hat{\theta}$, by loss functions (such as the squared error).
- Maximum likelihood point estimate of θ , denoted $\hat{\theta}_{\text{ML}}$, where the underlying data generation process is considered by a choice of probability density function to model the data and noise, $p(\mathcal{D}|\theta)$.

week 9 Bayes

Estimating θ from loss functions

Least-Squares (LS) loss function

$$J(\theta) = \sum_{n=1}^N (y_n - f_{\theta}(x_n))^2$$

$$\theta_* = \arg \min_{\theta} J(\theta)$$

The solution then becomes

LS Estimate

$$\theta_* = (X^T X)^{-1} X^T y$$

Biased estimation

Theory that justify biased estimation

Define the following biased estimator:

t_{reg}

$$\hat{\theta}_b = (1 + \alpha)\hat{\theta}_u$$

Choose α in the range

$$-2 < -\frac{2\text{MSE}(\hat{\theta}_u)}{\text{MSE}(\hat{\theta}_u) + \theta_o^2} < \alpha < 0$$

motivate regularization

Then $|\hat{\theta}_b| < |\hat{\theta}_u|$ and

$$\text{MSE}(\hat{\theta}_b) < \text{MSE}(\hat{\theta}_u)$$

The Ridge regression approach is an example of a norm shrinking loss the results in biased estimator:

Ridge regression

$$\mathcal{L}(\boldsymbol{\theta}, \lambda) = \sum_{n=1}^N (y_n - \boldsymbol{\theta}^T \mathbf{x}_n)^2 + \lambda \|\boldsymbol{\theta}\|_2^2$$

Last week summary

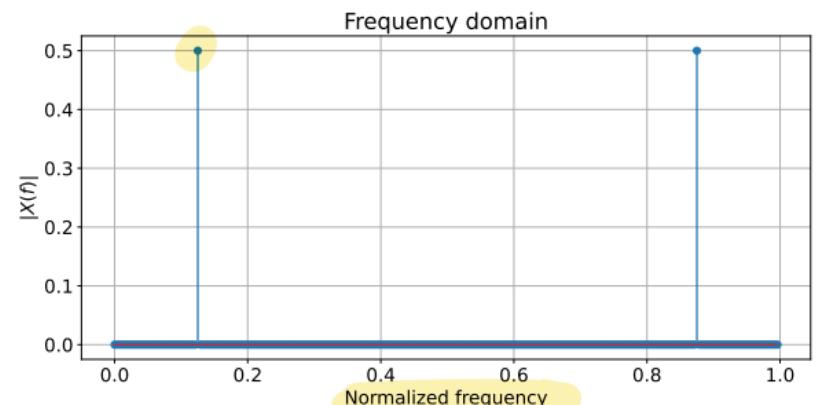
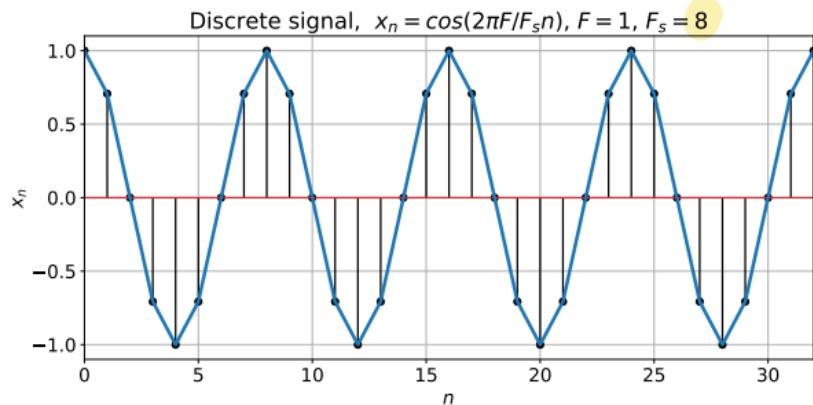
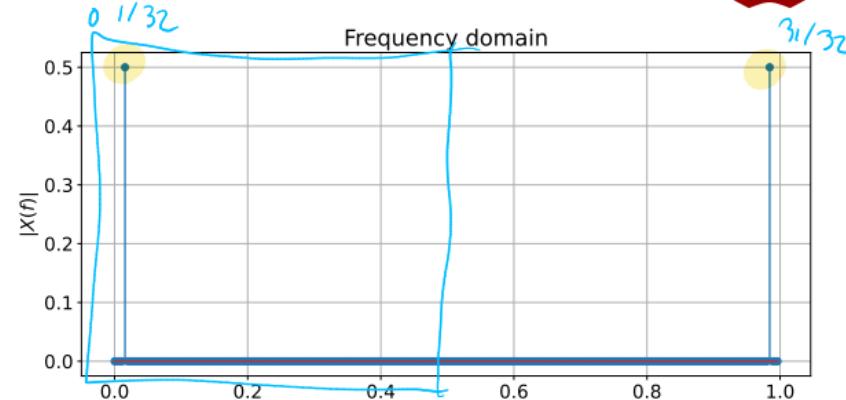
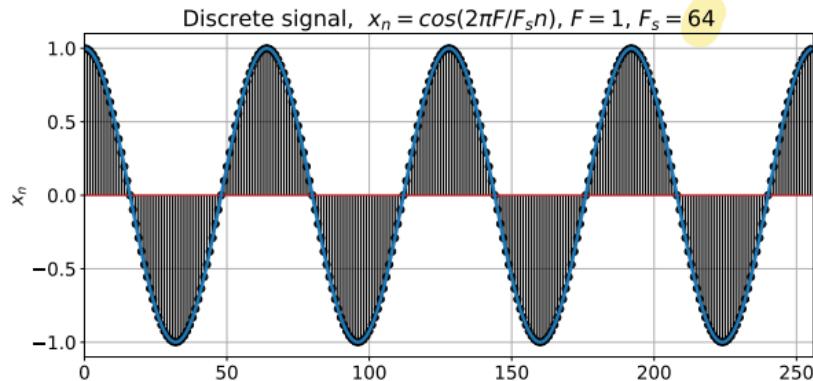
- Matrix calculus is equivalent to ordinary calculus and makes your life easier (promise!).
- We can do parameter estimation by loss functions, maximum likelihood, or Bayes formula.
- If we focus on the MSE, we can decompose the loss into a bias and variance term that offers a direct interpretation of the source of error.
- Biased estimation can, with suitable model choices, result in a lower MSE compared to unbiased estimation.

Frequency analysis

Frequency analysis

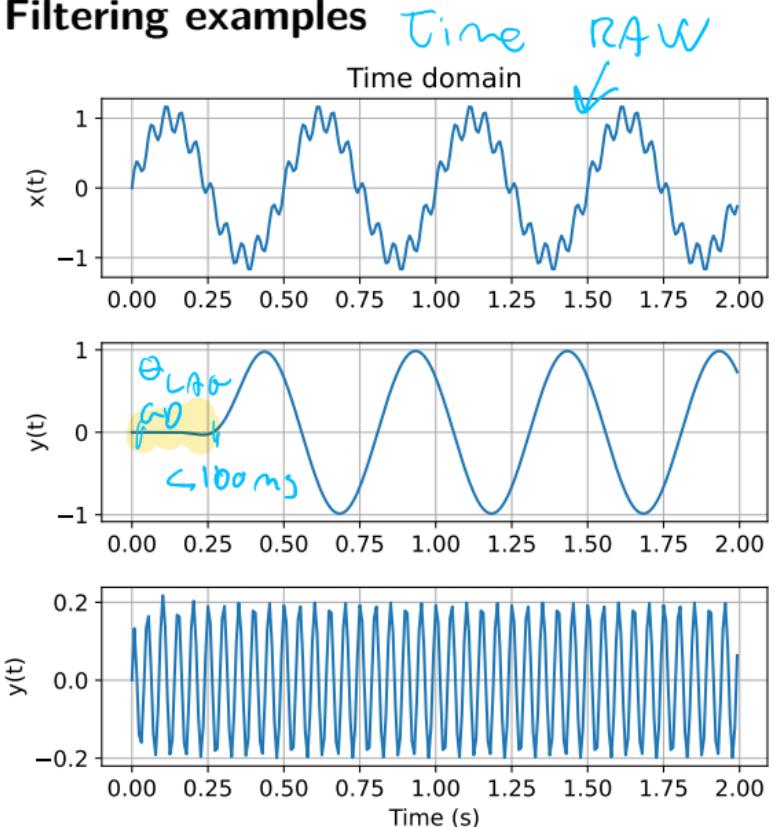
Frequency analysis

Time



Frequency analysis

Filtering examples

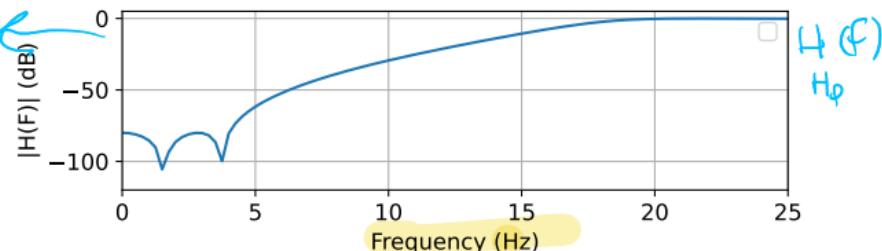
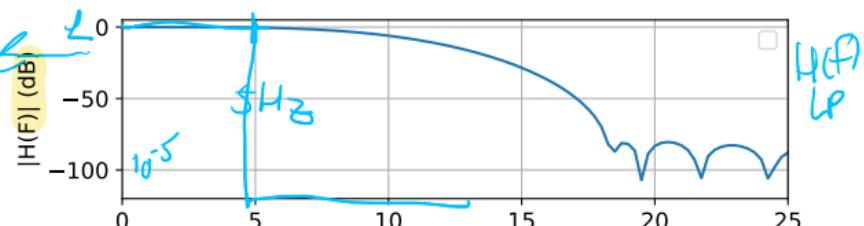
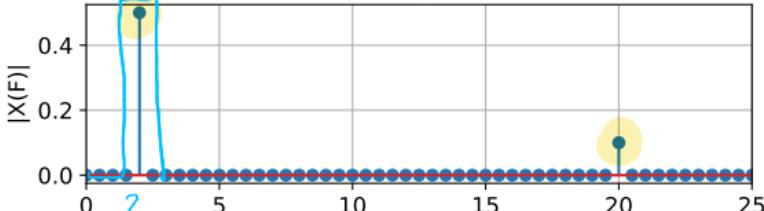


Time
RAW

$$Y(f) = H(f)X(f)$$

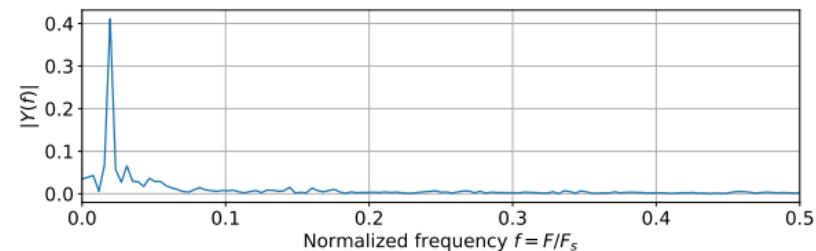
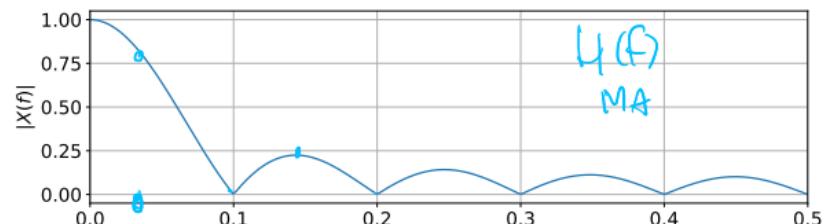
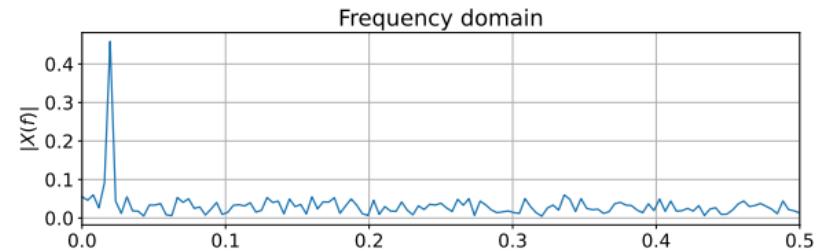
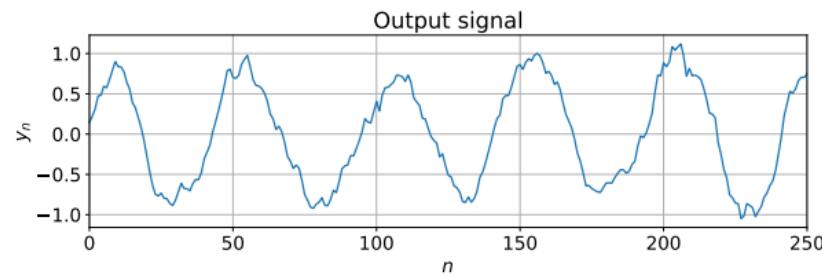
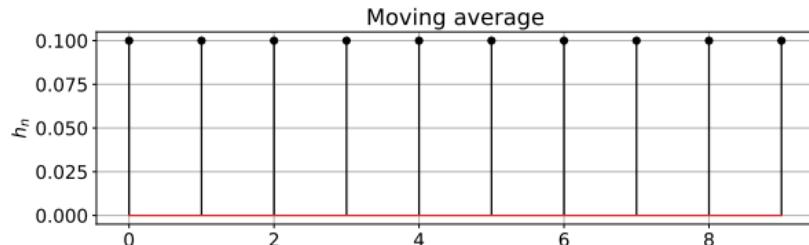
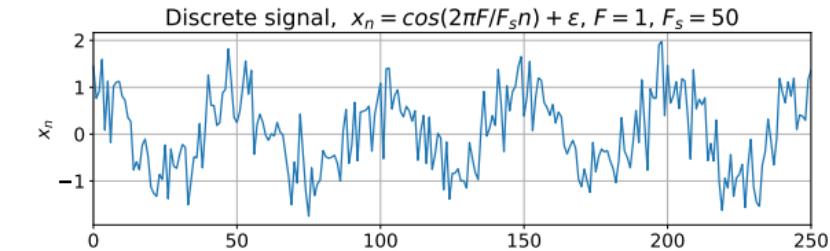
Input
Filter

Frequency domain



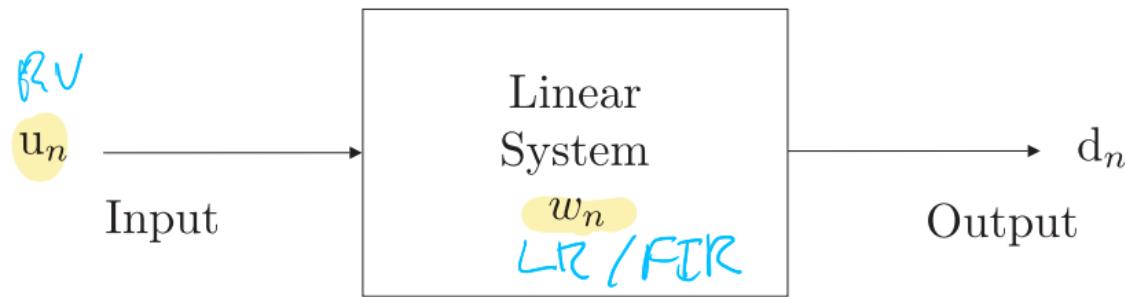
Frequency analysis

Filtering example – moving average



The Wiener filter

This weeks system

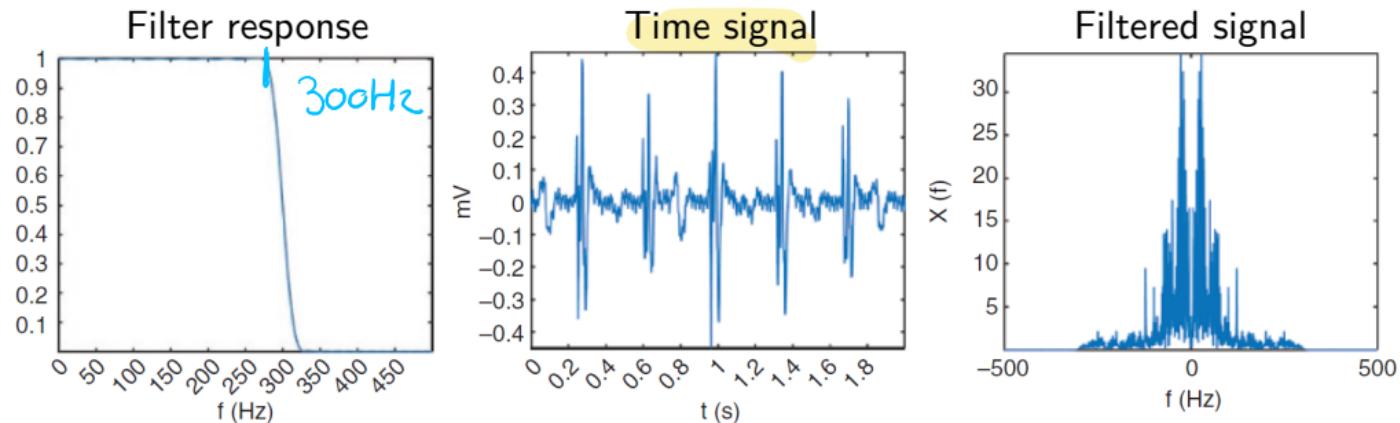


The input signal u_n , also written as $u_n = u(n)$, or often in DSP literature denoted as $x(n)$, is now considered to be a stochastic process.

The Wiener filter

Two approaches to filter design

Design filter with specific frequency response:



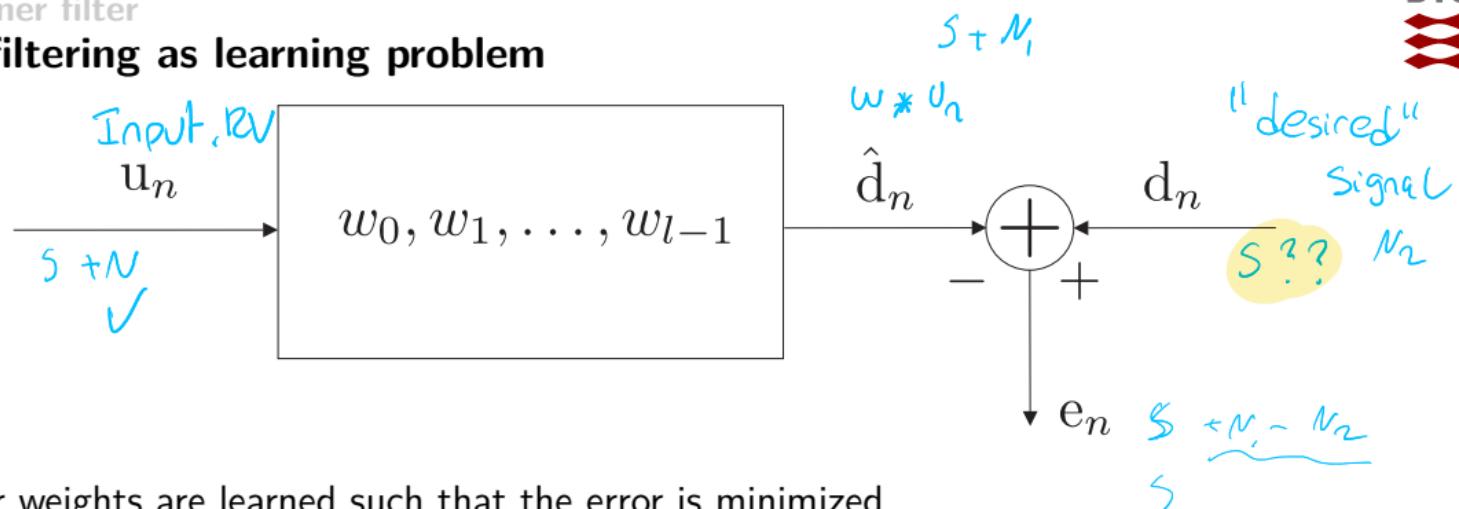
Alternative idea!

Create a filter as a learning problem, ie. one that minimized a cost function.

Discuss pros and cons, where could the two approached be applicable?

The Wiener filter

Linear filtering as learning problem



The filter weights are learned such that the error is minimized

$$e_n = d_n - \hat{d}_n$$

label / desired
output

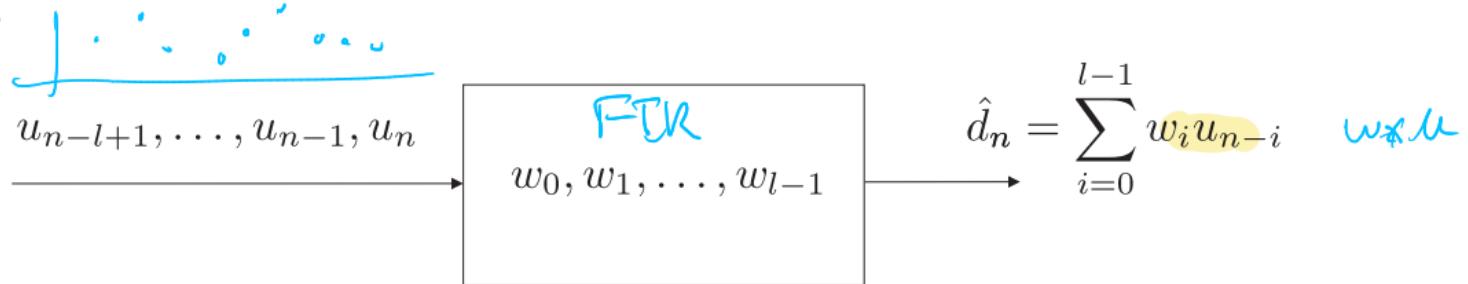
If we use an LTI filter and minimize the mean squared error, the filter is called a **Wiener filter**.

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

To use such filter, we need to specify a **desired** signal, d_n .

Linear filtering as learning problem

Once the filter coefficients are learned, the filter works like an ordinary linear time-invariant filter, i.e.



A word on notation:

- u_n is a stochastic process.
- u_n is a realization (concrete measurements) of the stochastic process, u_n .

More on this topic later.

There are two basic approaches to filter design

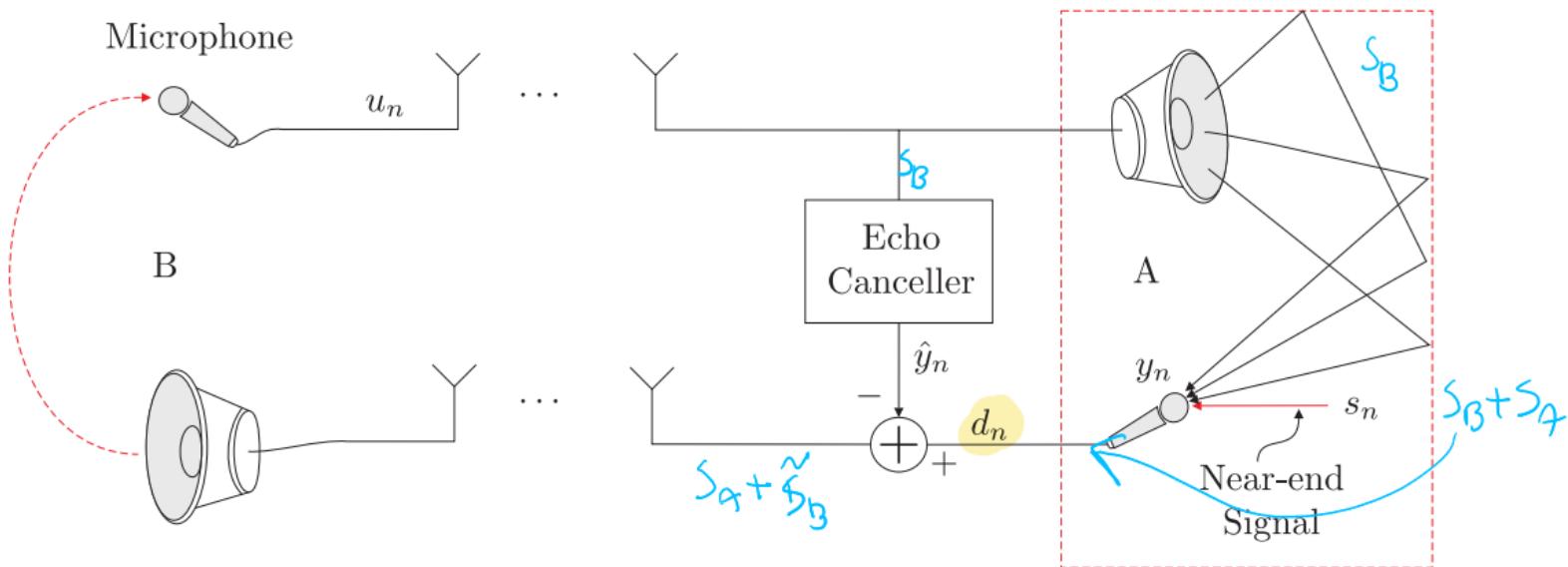
- Design a filter with the correct frequency response.
- Setup a learning problem, and learn the optimal filter from the data.
- The linear filter that minimizes the mean squared error (MSE) is called the **Wiener filter**.
z 4.5

Typical applications of filtering

Typical applications of filtering (ML sec 4.7)

- Echo cancellation
- Noise cancellation
- System identification
- Interference cancellation
- Channel equalization

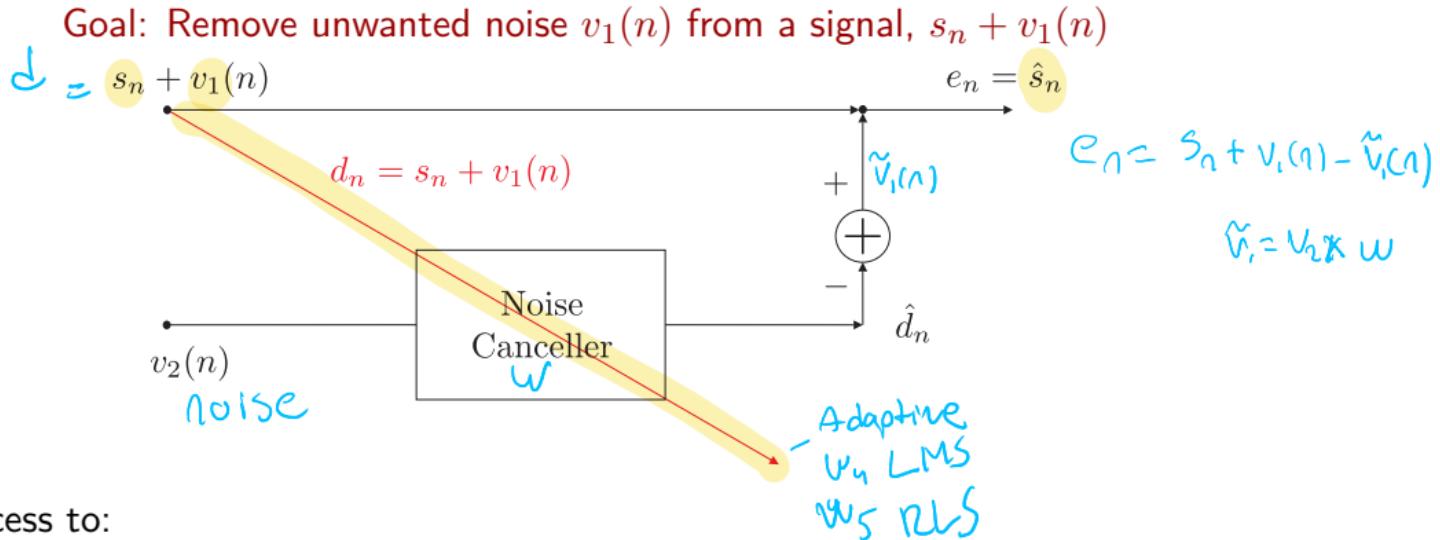
Typical applications of filtering Echo cancellation



Application; e.g. video conference or telephone call with load speaker (location A). Without echo canceling, the audio would loop back to location B.

Typical applications of filtering Noise cancellation

ex 3.3



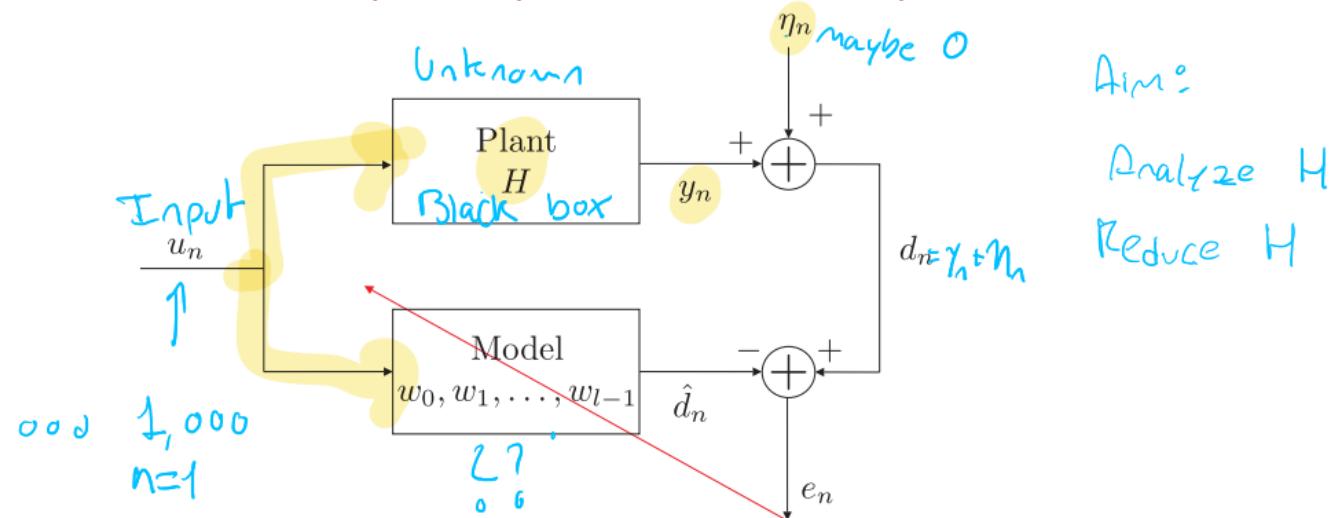
We have access to:

- The signal of interest, $s_n + v_1(n)$, where we want to suppress $v_1(n)$.
- A noise source $v_2(n)$, that is statistically related to $v_1(n)$ but statistically unrelated to s_n . Note: the signals $s_n + v_1(n)$ and $v_2(n)$ can change place.

We will work with this setup in the exercise.

Typical applications of filtering System identification

Goal: Model the impulse response of the unknown plant H



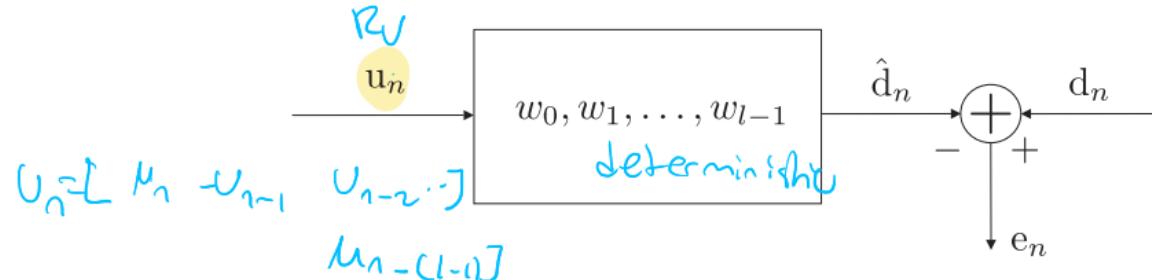
We have access to:

- The input signal u_n
- The noisy output d_n

- The linear filter setup has been successfully applied in many areas.
- We need to either know the statistical properties of our signal, or have training data.
- We can analyze the performance of the setups from a theoretical viewpoint if we model the signals as stochastic processes.

Linear filtering using mean-squared error

Linear filtering using mean-squared error

Linear filtering

$$u_n = s_n + \eta_n = \text{signal} + \text{noise}$$

- Filtering $\hat{d}_n = s_n$
- Smoothing $\hat{d}_n = s_n$ (with a circled D)
- Prediction $\hat{d}_n = s_n$ (with a circled D)

The wiener filter

The filter which minimizes the mean-squared error is the Wiener filter

Linear filtering using mean-squared error

The normal equations

We want to obtain an estimate of θ to the linear model (remember, x_n is random)

$$y(n) := \hat{d}_n = \theta^T x_n \quad \text{Fin } \theta_* x_n \quad w x_n$$

that minimize the mean squared error $J(\theta) = \mathbb{E} \left[(d_n - \hat{d}_n)^2 \right]$, i.e

$$\theta_* := \arg \min_{\theta} J(\theta)$$

Note, θ_* is used and not $\hat{\theta}$ (as last week), because when we know the statistical properties of the signals, we can find the optimal solution, and not just an estimate based on one specific dataset.

Linear filtering using mean-squared error

almost ex 2.1

Minimizing the cost function

To avoid notation clutter, we drop the n subscript, and obtain:

$$\hat{d} = \theta^T x \quad (\Theta * x_n)$$

$$J(\theta) = \mathbb{E}[(d - \hat{d})^2]$$

$$\nabla J(\theta) = \nabla \mathbb{E}[(d - \theta^T x)(d - \theta^T x)]$$

$$= \nabla \mathbb{E}[d^2 - \cancel{\theta^T x} - \theta^T x d - \cancel{(\theta^T x)^T \theta}]$$

$$a^T b = b^T a$$

$$= \nabla \mathbb{E}[d^2 - 2\cancel{\theta^T x} d + \theta^T x x^T \theta]$$

$$= \nabla_{\theta} (\mathbb{E}[d^2] - 2\theta^T \mathbb{E}[xd] + \theta^T \mathbb{E}[xx^T] \theta)$$

$$= -2\mathbb{E}[xd] + 2\mathbb{E}[xx^T]\theta = \mathbf{0} \quad \text{if possible} \Rightarrow \text{closed form}$$

$$\Rightarrow \mathbb{E}[xx^T]\theta = \mathbb{E}[xd] \quad \checkmark \quad \Rightarrow \sum_{x \in \mathcal{X}} \theta = r_{xd} \Rightarrow \theta = \sum_{x \in \mathcal{X}} r_x \quad \approx (x^T x) x y \quad \text{LR}$$

We recognize these terms as auto-correlation and cross-correlation functions.

Normal Equations

$\Sigma_x \theta_* = \mathbf{r}_{dx}$ (\mathbf{r}_{dx} is denoted \mathbf{p} in the book)

$$\mathbf{p} = [r_{dx}(0) \ r_{dx}(1) \ \cdots \ r_{dx}(l-1)]^T$$

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

To obtain the solution, θ_* we get

$$\theta_* = \Sigma_x^{-1} \mathbf{r}_{dx}$$

Relate this result to our Wiener filter: x is our input signal, d is our target signal, i.e our desired signal. So to apply a Wiener filter, we need to quantities: the auto-correlation of our input signal, and the cross correlation between our input signal and desired signal.

Theory needed

What can we do now?

- We know how to optimally learn the filter coefficients if we know the statistical properties of the signals.

Theory needed - how do we get to know the statistical properties?

- **Stochastic processes**, since the signals we just described are random, we can use tools from probability theory.

Linear filtering using mean-squared error

Summary



- We minimized the mean squared error for a linear filter, to obtain the Wiener filter solution.
- The solution is $\theta_* = \Sigma_x^{-1} r_{dx}$. !!!
- We need to carry out the following actions:
 - Specify the desired signal.
 - Compute the autocorrelation function for the input signal. $\Sigma_d = E[d d^T]$
 - Compute the cross-correlation function between the input signal and desired signal. $r_{dx} = E[x d]$

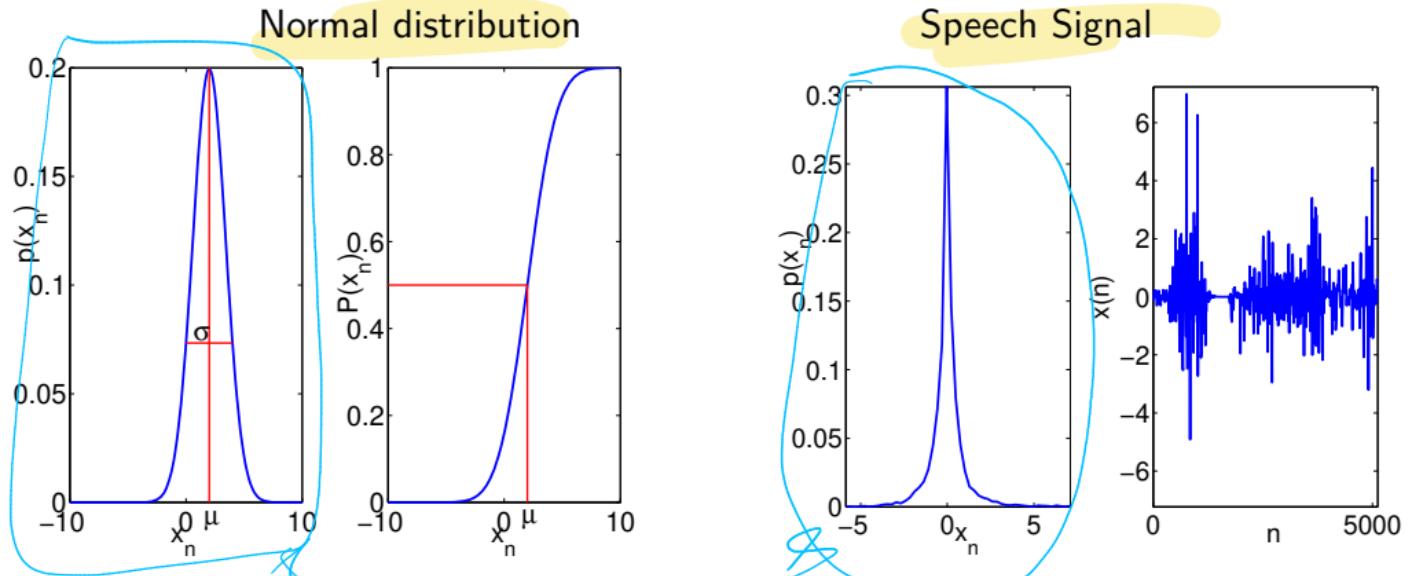
Stochastic Processes

Stochastic Processes

Motivation

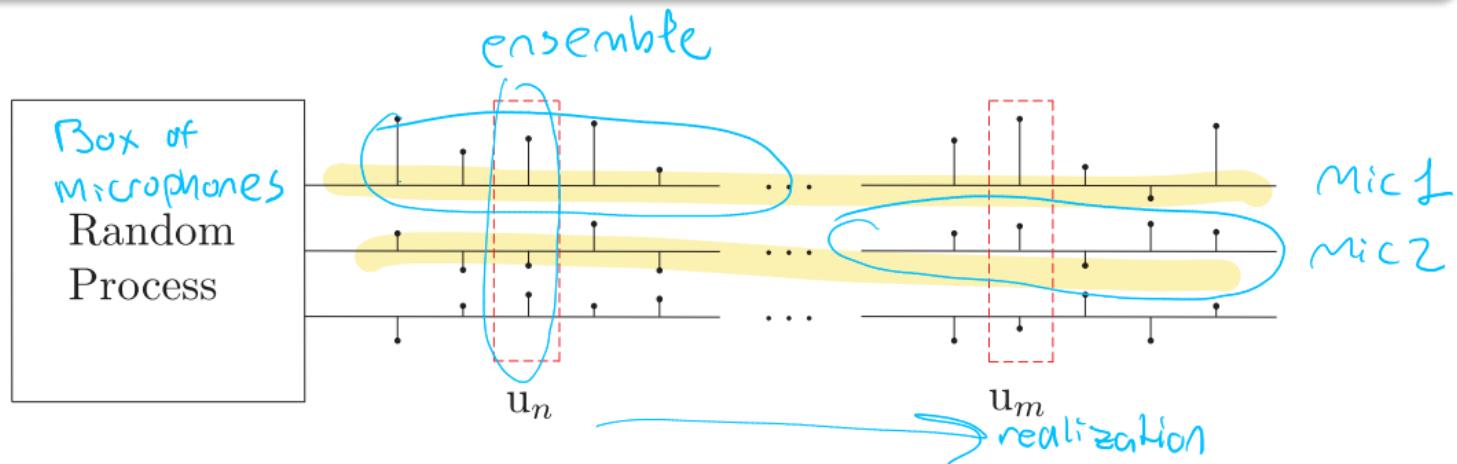
- Many real-world signals can be adequately be described as a stochastic process.

Example:



Stochastic process

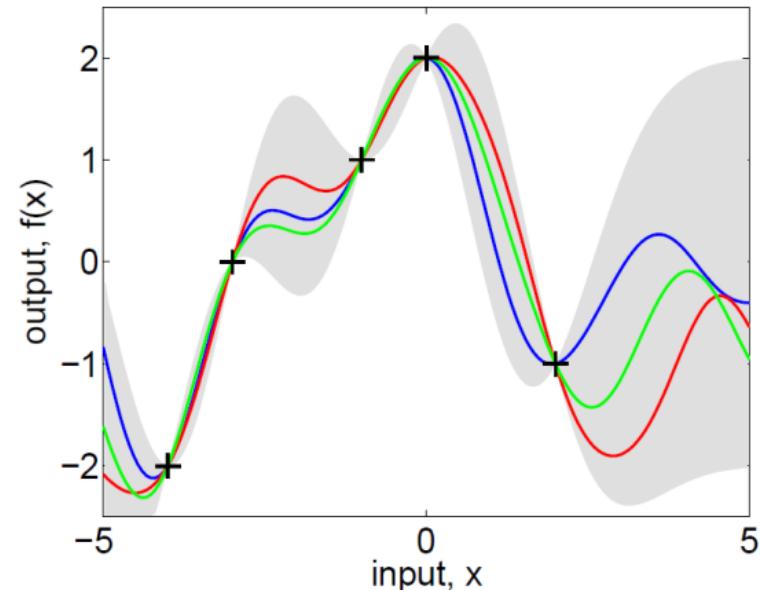
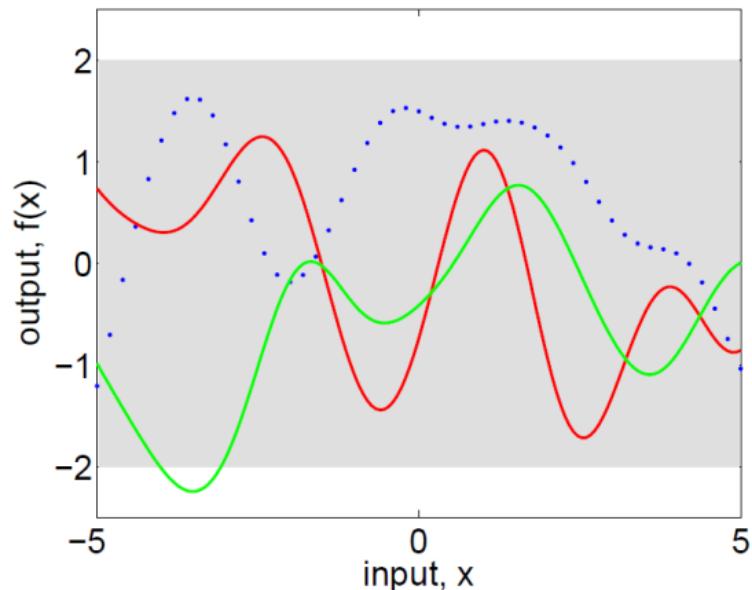
A stochastic process (or random process) can be considered as an ensemble of sequences, and individual examples are known as **realizations**.



u_n is the random variable at time instant n , and u_m is the random variable at time instant m . If the underlying probability density function does not change over time, the process is **stationary**.

Example of a stochastic process

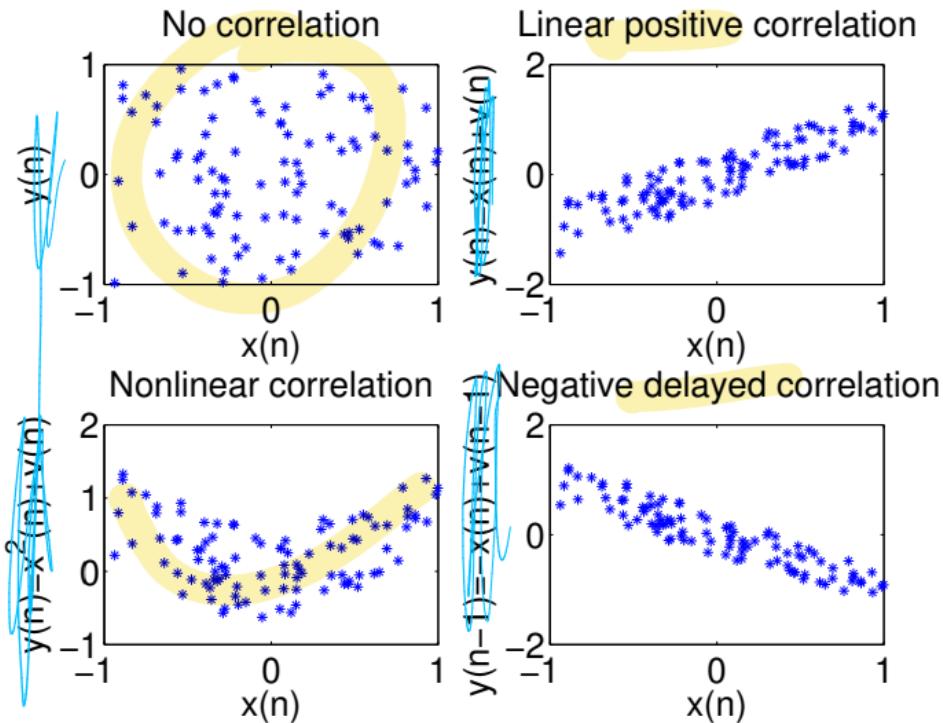
GP



Ref: "Gaussian Processes for Machine Learning", Carl Edward Rasmussen and Christopher K. I. Williams MIT Press, 2006.

Concept of correlation

3.1.6



Mean at Time n

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

Autocovariance at time instants n, m

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

Autocorrelation at time instants n, m

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

Cross-correlation at time instants n, m

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

These quantities are difficult to obtain in practice, so we impose certain assumptions about our signal of interest.

Weaker requirements – wide-sense stationarity

Wide-Sense Stationarity (WSS)

A stochastic process is said to be wide-sense stationary if the mean value is constant over time, and the autocovariance / autocorrelation sequences only depend on the differences of involved time instances

That means we can simply the following

For WSS processes

$$\mu_n = \mu$$

$$r_u(n, n - k) = r_u(k) = \mathbb{E}[u_n u_{n-k}]$$

$$r_{uv}(n, n - k) = r_{uv}(k) = \mathbb{E}[u_n v_{n-k}]$$

Definitions of estimators

$$\mu_u = \frac{1}{N} \sum_{n=1}^N u_n$$

$$r_{uv}(k) = \frac{1}{N} \sum_{n=k}^{N-1} u_n v_{n-k}, \quad k = 0, 1, \dots, N-1 \quad (\text{asymptotically unbiased})$$

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

Note: cross-correlation functions can both be defined as normalized and un-normalized.

E.g. in time-series analysis, often the normalized cross-correlation is used.

See more:

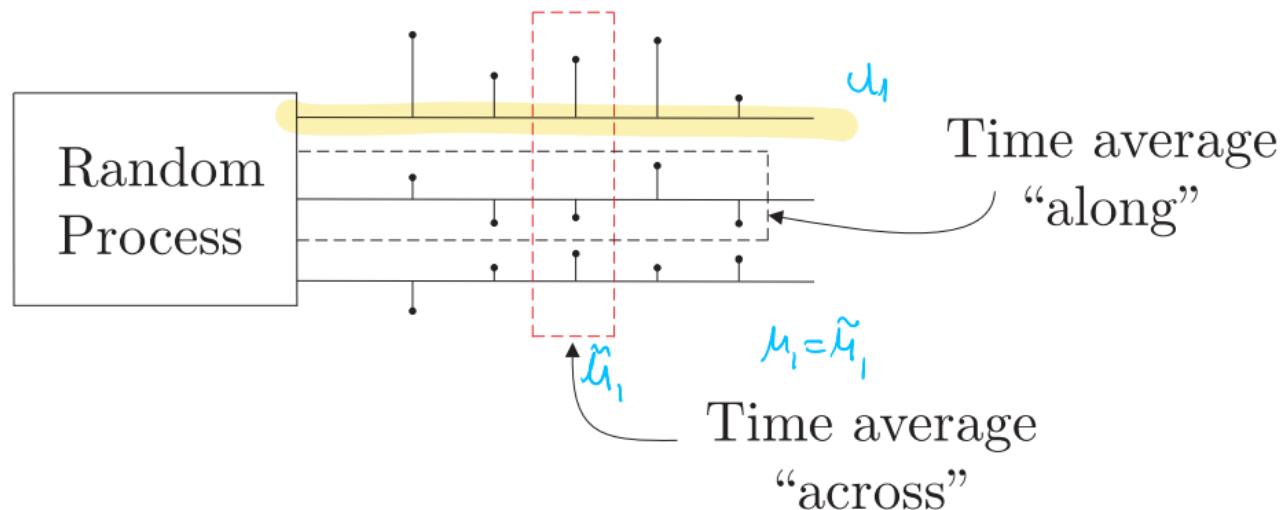
<https://en.wikipedia.org/wiki/Cross-correlation#Normalization>

Weaker requirements – ergodic processes

Mean-ergodic and covariance-ergodic

A stochastic process is said to be mean-ergodic and covariance-ergodic if the mean and covariance can be determined by any one of the realizations. Ergodic processes are also WSS.

Ensemble averages "across the process" can be obtained as time averages "along the process"



Example WSS process

White noise sequence

The white-noise sequence is both mean ergodic and covariance ergodic, and defined as

$$\mathbb{E}[\eta] = 0 \quad \text{and} \quad r(k) = \begin{cases} \sigma_\eta^2, & \text{if } k = 0, \\ 0, & \text{if } k \neq 0. \end{cases}$$

Example WSS process, that is **not covariance-ergodic**

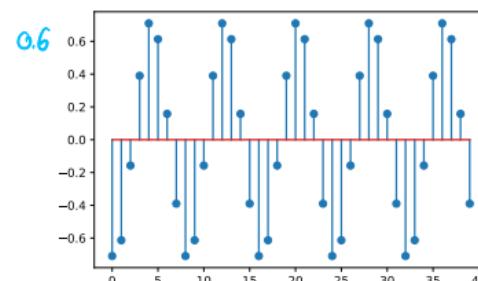
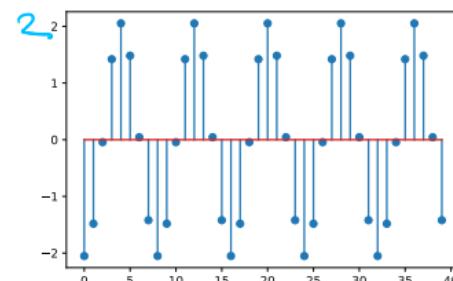
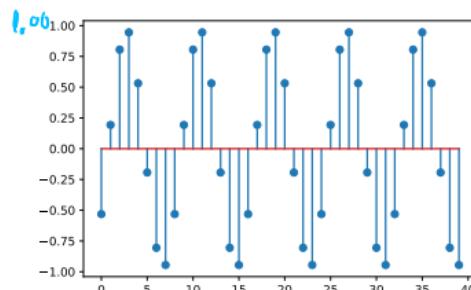
Consider the process

$$x_n = a \cos(n\lambda) + b \sin(n\lambda)$$

$$a \sim \mathcal{N}(0, \sigma_a^2), b \sim \mathcal{N}(0, \sigma_b^2), \lambda \in [0, \pi]$$

This process is WSS ($r_x(k) = \cos(\lambda k)$), mean-ergodic, but **not** covariance-ergodic. %

Example realizations



Autoregressive models

2.4 ex 3.1.4 , ps 2

An autoregressive process of order l , denoted as AR(l), follows the following

Autoregressive process (asymptotically WSS)

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

or identically written as (neglecting the sign of a 's)

$$u_n = \sum_{k=1}^l a_k u_{n-k} + \eta_n \quad (\text{signal like})$$

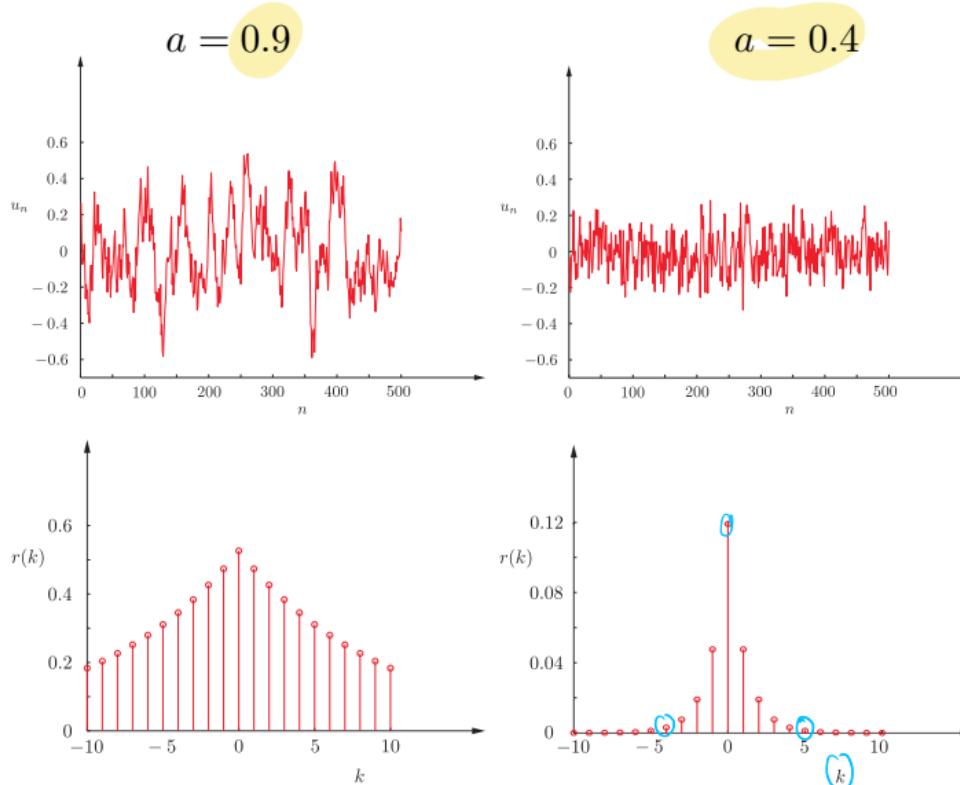
where η_n is a white noise process with variance σ_η^2 , i.e. the samples at η_n and η_m are uncorrelated for any $n \neq m$, and $\mathbb{E}[\eta] = 0$.

This is a very powerful model, that can model many real-life signals well, as the model simply specify that the value at instant n , is a linear combination of the l previous samples, added with (measurement) noise.

We will work heavily with an AR(1) process in the exercises

Stochastic Processes

Example of AR(1) process and the autocorrelation function



asym.
WSS

Stochastic processes summary

- We work with wide-sense stationary processes, which means their mean is constant over time, and their correlation functions can be computed with only the time lag taken into account. ✓
- For real signals, we assume they are mean-ergodic and covariance-ergodic (a locally sensible assumption), so we can compute statistics from only one realization. ✓
- As signal model of particular interest, we use an AR(l) process, which models many real-world phenomena adequately.
- Note: for a rigorous treatment of stochastic processes, DTU offers the course, "02407 Stochastic Processes - Probability 2".

- Instead of designing filters by hand, we can train the filters from data. If we use a LTI filter and the Mean Squared Error as our cost function, the filter is called Wiener filter (or linear filtering).
- We apply theory from stochastic processes to analyze this setup and benchmark the systems.
- The stochastic processes we work with, will be wide-sense stationary and ergodic. That means we can estimate the mean and correlation functions using only “one” realization (example). This makes the system applicable to real data.
- Many of the applications require adaptive filtering to be useful in practice. This will be the topic for the next two weeks.

Next week

Material: ML 2.6, 5.1–5.5.1 (skip 5.3.1), 5.9.

- Adaptive filtering
- Stochastic Gradient descent
- Least-Mean-Squares (LMS) adaptive algorithm, and normalized LMS (NLMS)

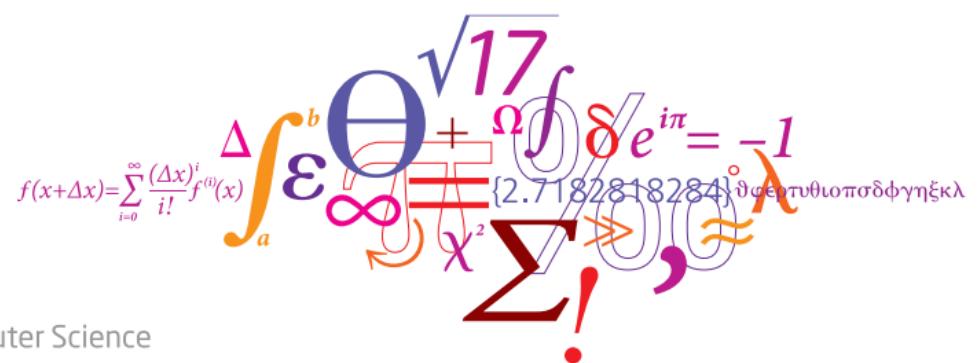
ONLY video lecture, I will be travelling, TA's are present from 13.00-17.00.

02471 Machine Learning for Signal Processing

Adaptive Linear Filtering with LMS

Tommy Sonne Alstrøm

Cognitive Systems section



Outline

- Course admin
- Last week review
- LMS and NLMS
- Justification of LMS
- Affine Projection Algorithm and normalized LMS
- Steady-state convergence analysis of LMS
- Next Week

Material: ML 2.6, 5.1–5.5.1, (skip 5.3.1), 5.6, 5.9.

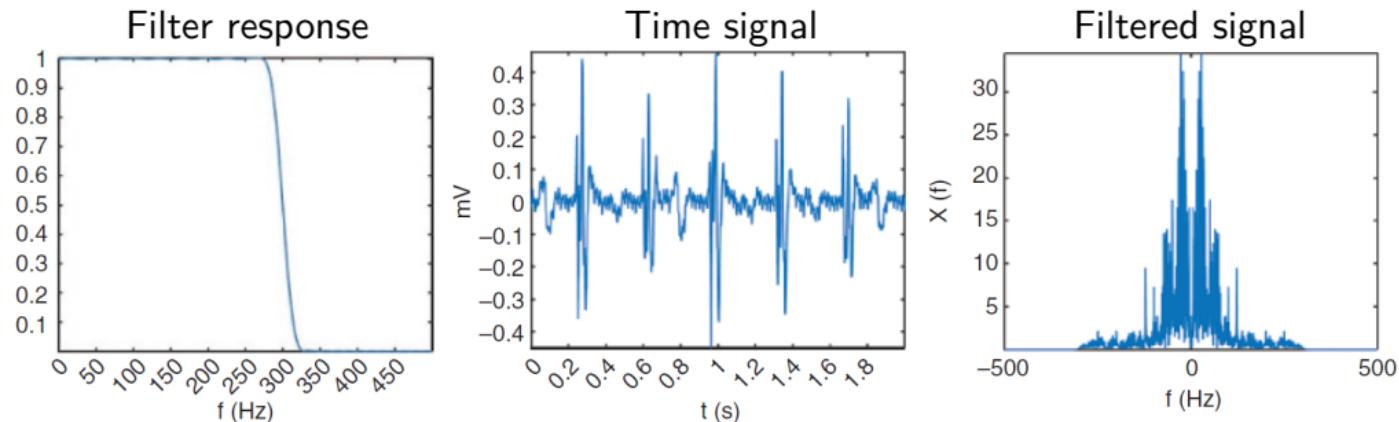
Administrative notes

- Problem set 1 due Monday 26/9, pass/no pass.
 - No pass attempts can be handed in again.
 - Everyone needs to hand in individually, but clearly write on DTU Learn who you worked with
 - ONLY upload PDFs.
- Feedback from you is a critical component for improving both the course and my teaching.
- Type of feedback
 - Mention one thing that worked?
 - Mention one thing should be improved (both in current lecture and last weeks exercise)?
 - Mention one thing you would change if you gave the lecture.

Last week review

Two approaches to filter design

Design filter with specific frequency response:



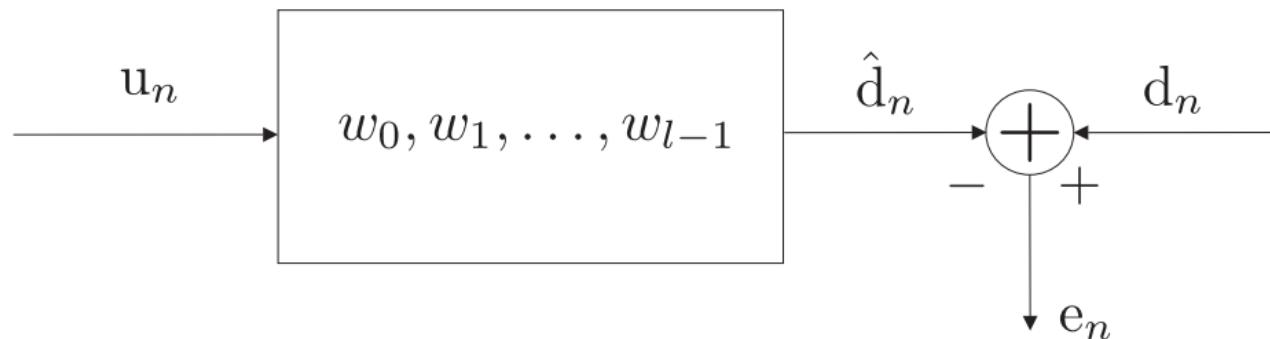
Idea:

Create a filter as a learning problem, ie. one that minimized a cost function.

Last week review

Linear filtering as learning problem

Filtering as a learning problem



The filter weights θ (denoted as $\mathbf{w} = [w_0, w_1, \dots, w_{l-1}]^T$ to indicate a linear filter) are learned such that the error is minimized

$$e_n = d_n - \hat{d}_n$$

If we choose to minimize the mean squared error, the filter is called a **Wiener filter**.

To use such filter, we need to specify a **desired** signal, d_n .

The normal equations

Minimizing the mean squared error $J(\theta) = \mathbb{E}[(d_n - \hat{d}_n)^2]$ results in the normal equations

Normal Equations

$$\Sigma_u \theta_* = \mathbf{r}_{du} \quad (\mathbf{r}_{du} \text{ is denoted } \mathbf{p} \text{ in the book})$$

$$\mathbf{p} = [r_{du}(0) \ r_{du}(1) \ \cdots \ r_{du}(l-1)]^T$$

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

Wiener filter solution

$$\theta_* = \Sigma_u^{-1} \mathbf{r}_{du}$$

Last week review

Stochastic processes

Auto-correlation and cross-correlation expressions for WSS processes (unnormalized)

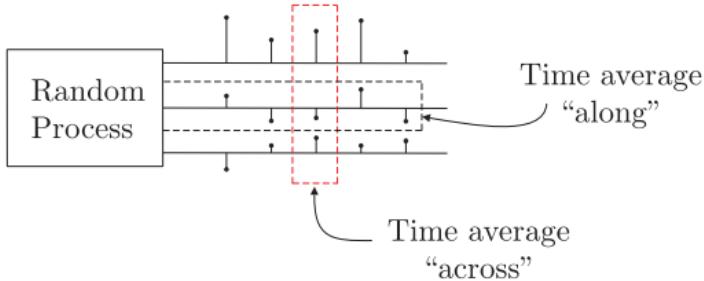
$$\mu_n = \mu \quad (\text{constant mean})$$

$$r_u(k) = \mathbb{E}[u_n u_{n-k}]$$

$$r_{uv}(k) = \mathbb{E}[u_n v_{n-k}]$$

These are unnormalized, see <https://en.wikipedia.org/wiki/Autocorrelation>

Additionally, for mean-ergodic and covariance-ergodic processes



$$\mu_u = \frac{1}{N} \sum_{n=1}^N u_n$$

$$r_{uv}(k) = \frac{1}{N} \sum_{n=0}^{N-1} u(n)v(n-k), \quad k = 0, 1, \dots, N-1$$

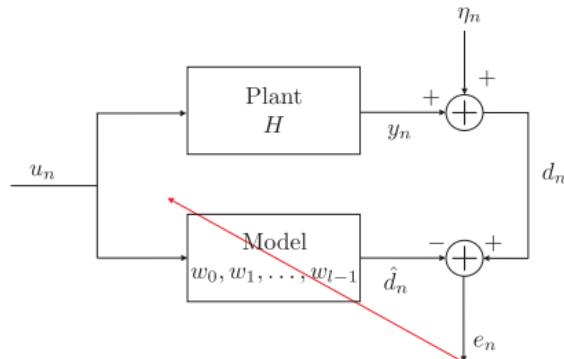
$$r'_{uv}(k) = \frac{1}{N-k} \sum_{n=k}^{N-1} u(n)v(n-k), \quad k = 0, 1, \dots, N-1$$

Last week review

Typical applications

System identification

Goal: Model the impulse response of H



We have access to:

- The input signal u_n , and the noisy output d_n

Common for all: usually the signals are not wide-sense stationary but may be locally wide-sense stationary.

Other applications

- Interference cancellation
- Echo cancellation
- Channel Equalization
- EEG signal analysis
- ...

Last week summary

- Instead of designing filters by hand, we can train the filters from data. If we use a LTI filter and minimize the MSE the filter is called Wiener filter (or Linear filtering).
- We apply theory from stochastic processes to analyze and design wiener filters.
- We will in general be working with wide-sense stationary and ergodic processes. That means we can estimate the mean and correlation functions using only “one” realization (example). This makes the system applicable to real data.
- Many of the applications require adaptive filtering / online learning to be useful in practice.

LMS and NLMS

Adaptive filtering problem statement

- The signal statistics are slowly changing, so we need to change our filter weights as well.
- We need to have access to second-order statistics.
- In practice we rarely have access to second-order statistics, so we will focus on algorithm that can learn the statistics iteratively using data.

The Least-Mean-Squares Algorithm

The LMS algorithm – algorithm 5.1 in the book

- **Initialize**
 - $\theta_{-1} = \mathbf{0} \in \mathbb{R}^l$
 - Select the value of μ
- **For** $n = 0, 1, \dots$, **Do**
 - $e_n = y_n - \theta_{n-1}^T \mathbf{x}_n$
 - $\theta_n = \theta_{n-1} + \mu e_n \mathbf{x}_n$
- **End For**

Parameters:

μ is the step size.

l is a filter length.

NLMS

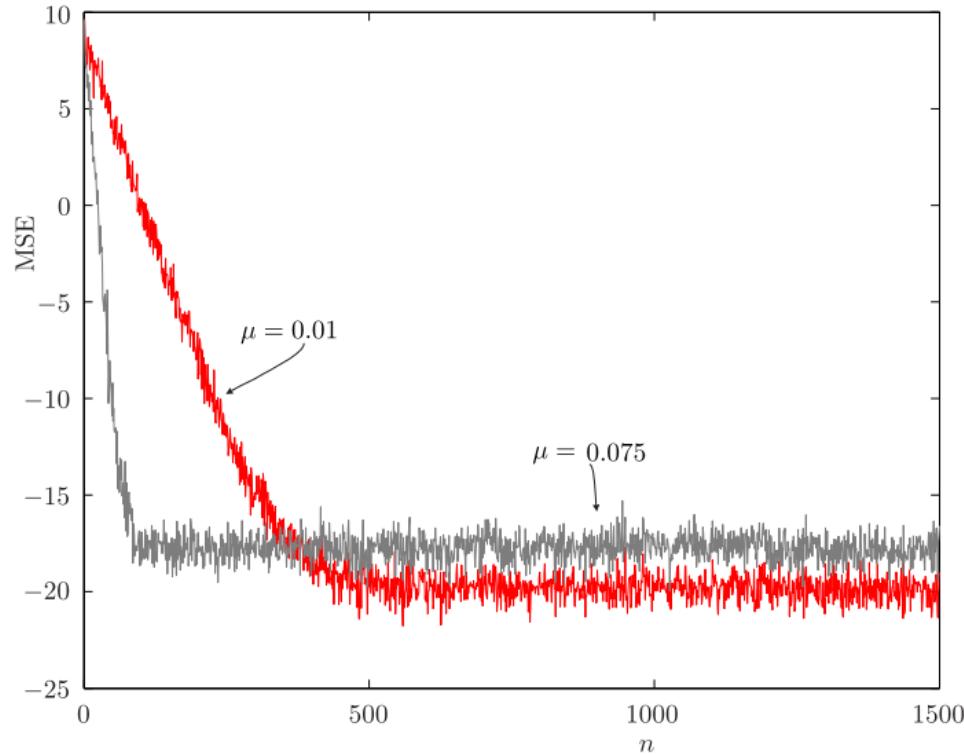
- Initialize
 - $\theta_{-1} = \mathbf{0} \in \mathbb{R}^l$
 - Select the value of $0 < \mu < 2$, and a small δ value
- For $n = 0, 1, \dots$, Do
 - $e_n = y_n - \theta_{n-1}^T \mathbf{x}_n$
 - $\theta_n = \theta_{n-1} + \frac{\mu}{\mathbf{x}_n^T \mathbf{x}_n + \delta} e_n \mathbf{x}_n$
- End For

Parameters:

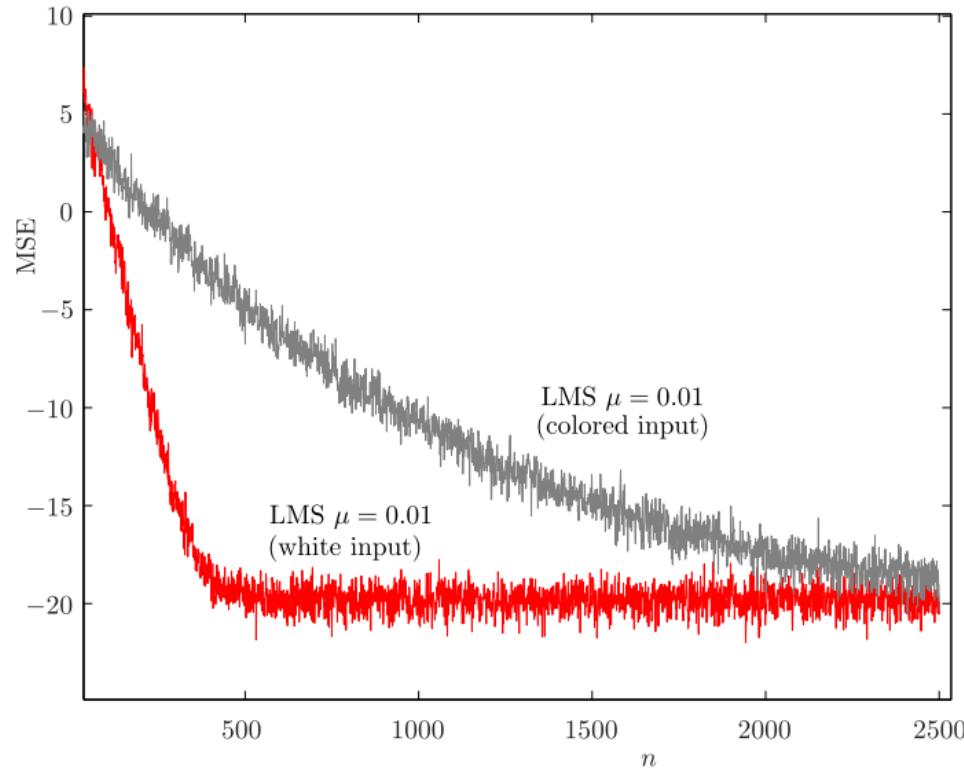
μ is the step size.

l is a filter length.

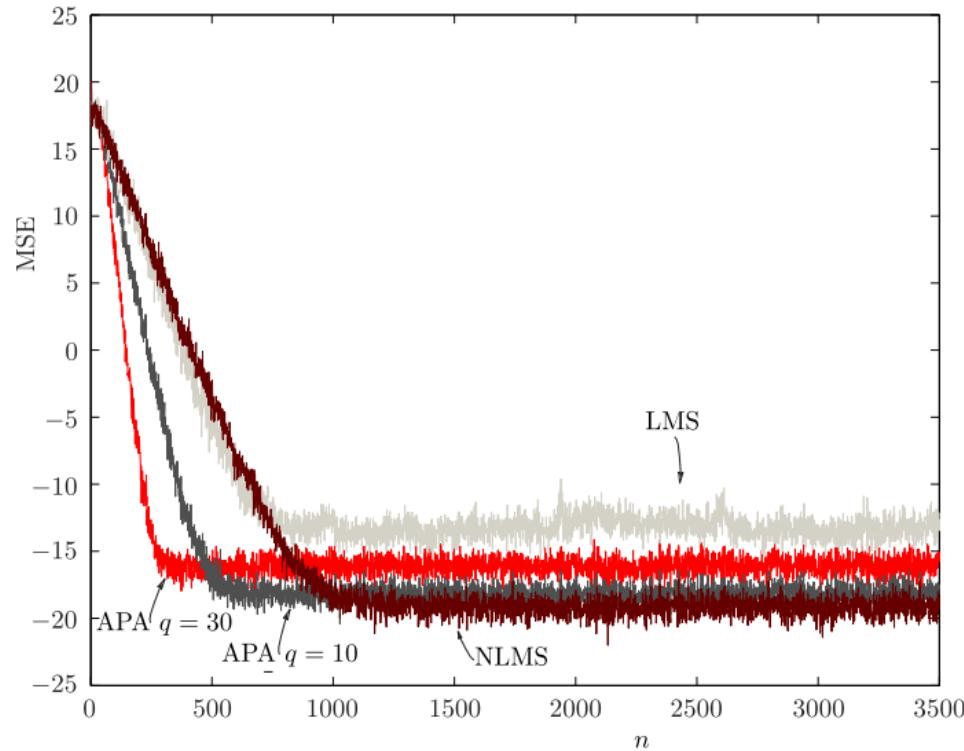
Convergence rate LMS



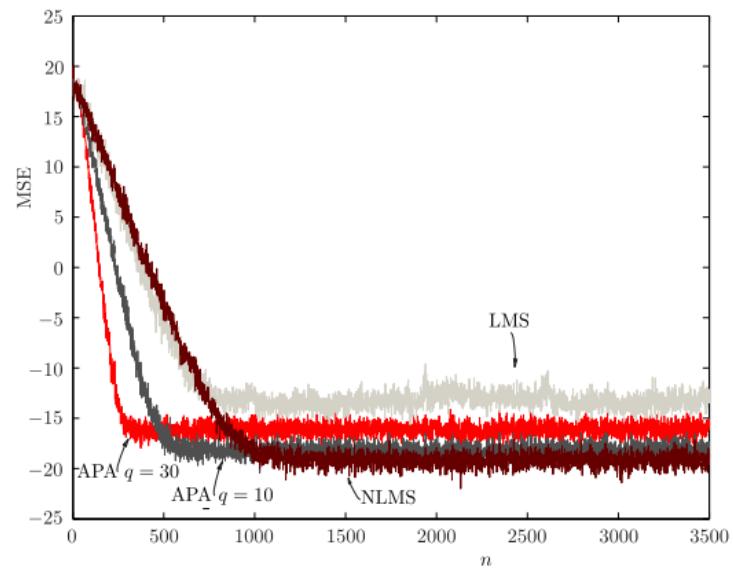
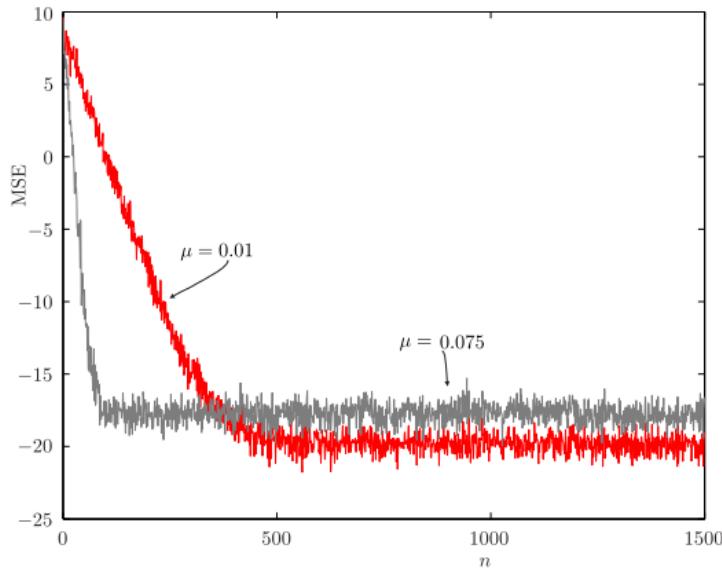
Convergence rate LMS



Convergence rate NLMS



LMS and NLMS Convergences



Discuss pros and cons of learning curves - can you think if applications appropriate of the difference curves?

- Many applications require adaptive algorithms (also called sequential learning, or online learning in machine learning).
- The value of μ affects the performance greatly (need theory to guide us).
- NLMS is generally preferred over LMS.

Justification of LMS

Steps involved in developing LMS

- ① Develop iterative scheme for when statistics are known ($r_x(k)$ and $r_{dx}(k)$ are known).
- ② Convert iterative scheme to data-driven approach ($r_x(k)$ and $r_{dx}(k)$ are estimated).
- ③ Ensure algorithm has agility (can adapt to changing environment).
- ④ Perform convergence analysis for steady state.

How did we arrive at LMS?

We want to develop an iterative scheme

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} + \mu_i \Delta \boldsymbol{\theta}^{(i)}, \quad \mu_i > 0, \quad i > 0$$

such that

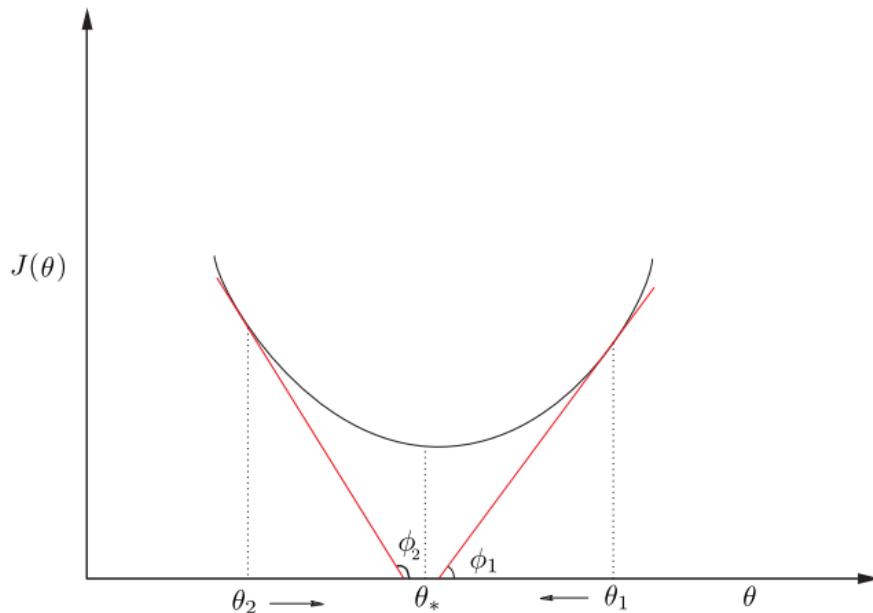
$$J(\boldsymbol{\theta}^{(i)}) < J(\boldsymbol{\theta}^{(i-1)})$$

where $J(\cdot)$ is the differentiable cost function.

1-dimensional example

Iterative scheme:

$$\theta^{(i)} = \theta^{(i-1)} + \mu_i \Delta\theta^{(i)}$$

If $\Delta\theta^{(i)}$ is the negative gradient direction we will end up at the minima.

Gradient Descent

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} - \mu_i \nabla J(\boldsymbol{\theta}^{(i-1)})$$

From last week, we found the gradient, $\nabla J(\boldsymbol{\theta}) = \Sigma_x \boldsymbol{\theta} - \mathbf{p}$.

At the optimum, $\nabla J(\boldsymbol{\theta}) = \Sigma_x \boldsymbol{\theta} - \mathbf{p} = \mathbf{0}$.

Issue: this only works if we have access to second order statistics.

Use Robbins–Monro scheme – stochastic approximation

If we have a cost function defined w.r.t. an expectation, we can apply the Robbins-Monro algorithm

Robbins-Monro algorithm

$$J(\boldsymbol{\theta}) = \mathbb{E}[\mathcal{L}(\boldsymbol{\theta}, \mathbf{y}, \mathbf{x})]$$

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}[\nabla \mathcal{L}(\boldsymbol{\theta}, \mathbf{y}, \mathbf{x})]$$

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} - \mu_n \nabla \mathcal{L}(\boldsymbol{\theta}_{n-1}, \mathbf{y}_n, \mathbf{x}_n)$$

$$\sum_n \mu_n^2 < \infty, \quad \sum_n \mu_n \rightarrow \infty : \quad \text{convergence conditions.}$$

Derivation of the update

$$\begin{aligned} J(\boldsymbol{\theta}) &= \mathbb{E}[e^2] \quad e = y - \boldsymbol{\theta}^T \mathbf{x} \\ \nabla J(\boldsymbol{\theta}) &= \mathbb{E}[\nabla e^2] \\ &= \mathbb{E}[2e\nabla e] \\ &= -2\mathbb{E}[(y - \boldsymbol{\theta}^T \mathbf{x})\mathbf{x}] \end{aligned}$$

The Robbins–Monro scheme was

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \mu_n \mathcal{L}(\boldsymbol{\theta}_{n-1}, \mathbf{y}_n, \mathbf{x}_n)$$

So we get

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \mu_n(y_n - \boldsymbol{\theta}_{(n-1)}^T \mathbf{x}_n)\mathbf{x}_n$$

The Robbins-Monro update

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} + \mu_n(y_n - \mathbf{x}_n^T \boldsymbol{\theta}^{(i-1)}) \mathbf{x}_n$$

To gain agility to track non-stationary environments, we use a constant step size.

LMS update

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} + \mu(y_n - \mathbf{x}_n^T \boldsymbol{\theta}^{(i-1)}) \mathbf{x}_n$$

- The LMS algorithm is a stochastic gradient algorithm that estimates second-order statistics from data.
- To obtain agility, we use a fixed step-size. This means convergence guarantees provided by Robbins-Monro are invalid.
 - Care must be taken for choosing the step-size (sec 5.3).
 - We can use the eigenvalues of Σ_x to guide our choice of step-size.
 - Large spread in eigenvalues means slow convergence (sec 5.3).
- LMS is a flexible algorithm that requires tuning of the step-size.

Affine Projection Algorithm and normalized LMS

Affine Projection Algorithm and normalized LMS

The Affine Projection Algorithm (APA)

The APA algorithm minimizes the following problem

$$\begin{aligned}\theta_n &= \arg \min_{\theta} \|\theta - \theta_{n-1}\|^2 \\ \text{s.t. } & \boldsymbol{x}_{n-i}^T \boldsymbol{\theta} = y_{n-i}, \quad i = 0, 1, \dots, q-1\end{aligned}$$

Lagrange multiplier

$$\begin{aligned}& \text{minimize} && J(\boldsymbol{\theta}) \\ & \text{subject to} && f_i(\boldsymbol{\theta}) = 0, \quad i = 1, 2, \dots, m\end{aligned}$$

This problem can be solved by optimizing a modified function, usually called **the Lagrangian**

$$L(\boldsymbol{\theta}, \boldsymbol{\lambda}) = J(\boldsymbol{\theta}) - \sum_{i=1}^m \lambda_i f_i(\boldsymbol{\theta})$$

The saddle point of this function is then found by solving $\nabla L(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \mathbf{0}$

Affine Projection Algorithm and normalized LMS

Deriving updates for APA I



$$\begin{aligned}\boldsymbol{\theta}_n &= \arg \min_{\boldsymbol{\theta}} \|\boldsymbol{\theta} - \boldsymbol{\theta}_{n-1}\|^2 \\ \text{s.t. } \quad \boldsymbol{x}_{n-i}^T \boldsymbol{\theta} &= y_{n-i}, \quad i = 0, 1, \dots, q-1\end{aligned}$$

$$L(\boldsymbol{\theta}, \boldsymbol{\lambda}) = (\boldsymbol{\theta} - \boldsymbol{\theta}_{n-1})^T (\boldsymbol{\theta} - \boldsymbol{\theta}_{n-1}) + \sum_{i=0}^{q-1} \lambda_i (y_{n-i} - \boldsymbol{\theta}^T \boldsymbol{x}_{n-i})$$

$$\frac{d}{d\boldsymbol{\theta}} L(\boldsymbol{\theta}, \boldsymbol{\lambda}) = 2\boldsymbol{\theta} - 2\boldsymbol{\theta}_{n-1} - \sum_{i=0}^{q-1} \lambda_i \boldsymbol{x}_{n-i}$$

$$\frac{d}{d\boldsymbol{\theta}} L(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \mathbf{0} \Rightarrow \boldsymbol{\theta} = \boldsymbol{\theta}_{n-1} + \frac{1}{2} \sum_{i=0}^{q-1} \lambda_i \boldsymbol{x}_{n-i}$$

$$X_n := [\boldsymbol{x}_n, \dots, \boldsymbol{x}_{n-q+1}]^T, \quad \boldsymbol{\lambda}^T := [\lambda_0, \dots, \lambda_{q-1}], \quad \boldsymbol{y}_n^T := [y_n, \dots, y_{n-q+1}]$$

Affine Projection Algorithm and normalized LMS

Deriving updates for APA II



$$\boldsymbol{\theta} = \boldsymbol{\theta}_{n-1} + \frac{1}{2} X_n^T \boldsymbol{\lambda}$$

$$X_n \boldsymbol{\theta} = \mathbf{y}_n$$

Solve the two equations for the two unknowns:

$$X_n \left(\boldsymbol{\theta}_{n-1} + \frac{1}{2} X_n^T \boldsymbol{\lambda} \right) = \mathbf{y}_n \quad \Leftrightarrow$$

$$\frac{1}{2} X_n X_n^T \boldsymbol{\lambda} = \mathbf{y}_n - X_n \boldsymbol{\theta}_{n-1} \quad \Leftrightarrow$$

$$\frac{1}{2} \boldsymbol{\lambda} = \left(X_n X_n^T \right)^{-1} (\mathbf{y}_n - X_n \boldsymbol{\theta}_{n-1}) \Rightarrow$$

$$\boldsymbol{\theta} = \boldsymbol{\theta}_{n-1} + X_n^T \left(X_n X_n^T \right)^{-1} (\mathbf{y}_n - X_n \boldsymbol{\theta}_{n-1})$$

Affine Projection Algorithm and normalized LMS

Deriving updates for APA III



$$\boldsymbol{\theta} = \boldsymbol{\theta}_{n-1} + X_n^T \left(X_n X_n^T \right)^{-1} (\mathbf{y}_n - X_n \boldsymbol{\theta}_{n-1})$$

$$\mathbf{e}_n := \mathbf{y}_n - X_n \boldsymbol{\theta}_{n-1}$$

$$\boldsymbol{\theta} = \boldsymbol{\theta}_{n-1} + X_n^T \left(X_n X_n^T \right)^{-1} \mathbf{e}_n$$

The update algorithm then becomes

$$\mathbf{e}_n = \mathbf{y}_n - X_n \boldsymbol{\theta}_{n-1}$$

$$\boldsymbol{\theta} = \boldsymbol{\theta}_{n-1} + \mu X_n^T \left(\delta I + X_n X_n^T \right)^{-1} \mathbf{e}_n$$

For $q = 1$

$$e_n = y_n - \mathbf{x}_n \boldsymbol{\theta}_{n-1}$$

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \frac{\mu}{\mathbf{x}_n^T \mathbf{x}_n + \delta} e_n \mathbf{x}_n$$

APA and NLMS

The APA algorithm minimizes the following problem

$$\begin{aligned}\boldsymbol{\theta}_n &= \arg \min_{\boldsymbol{\theta}} \|\boldsymbol{\theta} - \boldsymbol{\theta}_{n-1}\|^2 \\ \text{s.t. } \boldsymbol{x}_{n-i}^T \boldsymbol{\theta} &= y_{n-i}, \quad i = 0, 1, \dots, q-1\end{aligned}$$

Define the following matrix

$$X_n^T := [\boldsymbol{x}_n, \dots, \boldsymbol{x}_{n-q+1}]$$

Then, by using Lagrange multipliers, we get the following update, ($0 \leq \mu \leq 2$)

$$\begin{aligned}\boldsymbol{e}_n &= \boldsymbol{y}_n - X_n \boldsymbol{\theta}_{n-1} \\ \boldsymbol{\theta} &= \boldsymbol{\theta}_{n-1} + \mu X_n^T (\delta I + X_n X_n^T)^{-1} \boldsymbol{e}_n\end{aligned}$$

For $q = 1$, we call the algorithm nlms.

Steady-state convergence analysis of LMS

Steady-state convergence analysis of LMS

Convergence analysis of coefficients

Define the coefficient error vector as

$$\mathbf{c}_n := \boldsymbol{\theta}_n - \boldsymbol{\theta}_*$$

Analyzing convergence in the mean,

$$\mathbb{E} [\mathbf{c}_n]$$

as $n \rightarrow \infty$ we get the following results

$$0 < \mu < \frac{2}{\lambda_{\max}}$$

However analyzing the covariance matrix, $n \rightarrow \infty$

$$\Sigma_{c,n} := \mathbb{E} [\mathbf{c}_n \mathbf{c}_n^T]$$

$$0 < \mu < \frac{2}{\text{trace}\{\Sigma_x\}} = \frac{2}{l \cdot r_x(0)}$$

Steady-state convergence analysis of LMS

Convergence analysis of error

Define the misalignment as

Misalignment

$$\mathcal{M} := \frac{J_{\text{exc}}}{J_{\min}}$$

Analyzing the mean squared error behavior, we get the error at step n :

$$\begin{aligned} J_n &:= \mathbb{E} [e_n^2] = \dots (\text{eq 5.46} - 5.47) \\ &= J_{\min} + \text{trace} \{ \Sigma_x \Sigma_{c,n-1} \} \end{aligned}$$

Excess MSE at time instant n

$$J_{\text{exc},n} = \text{trace} \{ \Sigma_x \Sigma_{c,n-1} \} : \quad \text{excess MSE at time instant } n$$

For small values of μ , we get

Expected misalignment

$$\mathcal{M} \simeq \frac{1}{2}\mu \text{trace}\{\Sigma_x\} : \quad \text{misadjustment}$$

- The convergence analysis involve a fair amount of assumptions and is lengthy to derive.
- You are expected to be familiar with the material but can skip any derivation that the book defers to “Problem X derives this”.
- The convergence analysis provides theoretical foundation for choosing the step-size μ . We need agility in the algorithm, but also low misalignment. This is a trade-off.

- Adaptive algorithms, such as LMS and NLMS can be used to estimate second-order statistics from data.
- The algorithms needs to be carefully tuned per application basis, as there is trade-offs between agility and total error.
- In general, NLMS is considered more stable and better than LMS.
- These algorithms are deployed in many real-world problems to enable filtering in a changing environment.

Material: ML 5.12, 6.1–6.3 (until "The LS estimator is BLUE"), 6.6–6.8, 6.12.

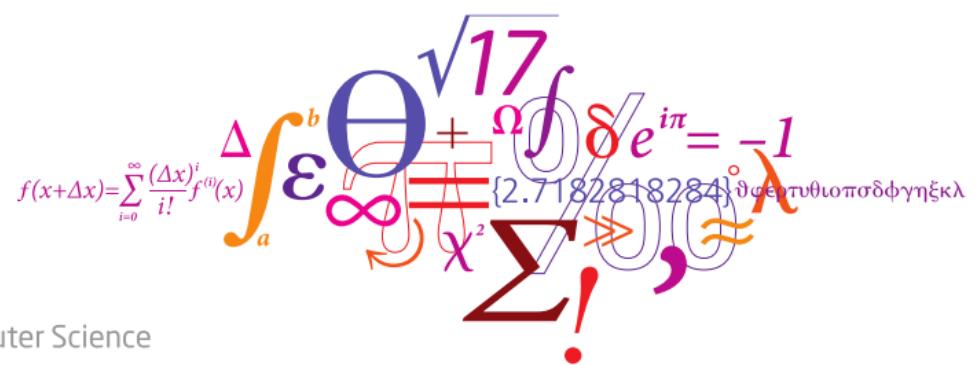
- Adaptive filtering continued
- Tracking performance in non-stationary environment
- Recursive Least-Squares (RLS) adaptive algorithm

02471 Machine Learning for Signal Processing

Adaptive Linear Filtering with RLS

Tommy Sonne Alstrøm

Cognitive Systems section



Outline

- Course admin
- Last week review
- Recursive Least Squares (RLS) 
- Stationary simulation model and steady-state convergence analysis of LMS
- Non-stationary simulation model and convergence analysis
- Derivation of the RLS algorithm  S_{ol}
- Relation between RLS and Newton's method
- Next Week

Material: ML 5.12, 6.1–6.3 (until "The LS estimator is BLUE"), 6.5 (only page 265), 6.6–6.8, 6.12. Most important sections are 6.6, 6.8, 6.12 and Algorithm 6.1.

Tentative course outline by lecture module

Week	Topic	Material (ML)
35	Digital signal processing, probability theory, machine learning	1.1–2.3
36	Matrix derivatives, constrained optimization, parameter estimation	3.1–3.3, 3.5, 3.8–3.11, A.1–A.2, C.1–C.2
37	Linear filtering	2.4, 4.1–4.3, 4.5–4.7
38	Adaptive filtering, LMS	2.6, 5.1–5.5.1, 5.9, 5.12
39	Adaptive filtering, RLS	6.1–6.3, 6.5–6.8, 6.12
40	Sparsity aware learning	8.2, 8.10.1–8.10.2, 9.1–9.5, 9.9
41	Shrinkage algorithms, Time-frequency analysis	10.1–10.2, 10.5–10.6
43	Dictionary learning, ICA, k -svd	2.5, 19.1–19.3, 19.5–19.7
44	Bayesian Modeling and EM	11.2, 12.1–12.2, 12.4–12.5, 12.10
45	State-space models, Hidden Markov models	15.1–15.3.1, 15.7, 16.4–16.5
46	State-space models, Kalman filter	4.9–4.9.1, 4.10, 17.3
47	Kernel methods, Kernel ridge regression	11.1–11.5, 11.7
48	Kernel methods, Support vector regression	11.8

Problem set 1 and 2 hand-ins

All problem sets are compulsory hand-ins. They will not be graded, but must be approved in order to attend the exam. The problem sets works as **formative** feedback and is your opportunity to get written feedback on your work.

Problem set 1, late submissions due 9/10

Material from week 1 as well as crucial prerequisites.

Problem set 2, due 22/10 (postponed one week)

Material from week 2–7: Parameter estimation, Linear filtering (Wiener), Time frequency analysis, Adaptive filtering, Compressed sensing. **The problem set is meant as a half way test.**

Last week review

The Least-Mean-Squares Algorithm

The LMS algorithm – algorithm 5.1 in the book

- Initialize
 - $\theta_{-1} = \mathbf{0} \in \mathbb{R}^l$
 - Select the value of μ
- For $n = 0, 1, \dots$, Do
 - $e_n = y_n - \theta_{n-1}^T \mathbf{x}_n$ ↪
 - $\theta_n = \theta_{n-1} + \mu e_n \mathbf{x}_n$
- End For

Parameters:

μ is the step size.

l is a filter length.

The Affine Projection Algorithm (APA)

The APA algorithm minimizes the following problem

$$\begin{aligned}\theta_n &= \arg \min_{\theta} \|\theta - \theta_{n-1}\|^2 \\ \text{s.t. } & \quad \mathbf{x}_{n-i}^T \theta = y_{n-i}, \quad i = 0, 1, \dots, q-1\end{aligned}$$

} design

Define the following matrix

$$X_n = [\mathbf{x}_n, \dots, \mathbf{x}_{n-q+1}]^T$$

Then, by using Lagrange multipliers, we got the following update

$$\begin{aligned}\theta_n &= \theta_{n-1} + X_n^T (X_n X_n^T)^{-1} e_n \\ e_n &= \mathbf{y}_n - X_n \theta_{n-1}\end{aligned}$$

For $q = 1$, we call the algorithm nlms.

The Normalized Least-Mean-Square Algorithm 5.2

NLMS

- Initialize

- $\theta_{-1} = \mathbf{0} \in \mathbb{R}^l$
- Select the value of $0 < \mu < 2$, and a small δ value

- For $n = 0, 1, \dots$, Do

- $e_n = y_n - \theta_{n-1}^T x_n$
- $\theta_n = \theta_{n-1} + \frac{\mu}{x_n^T x_n + \delta} e_n x_n$

- End For

what if $\mu=1$

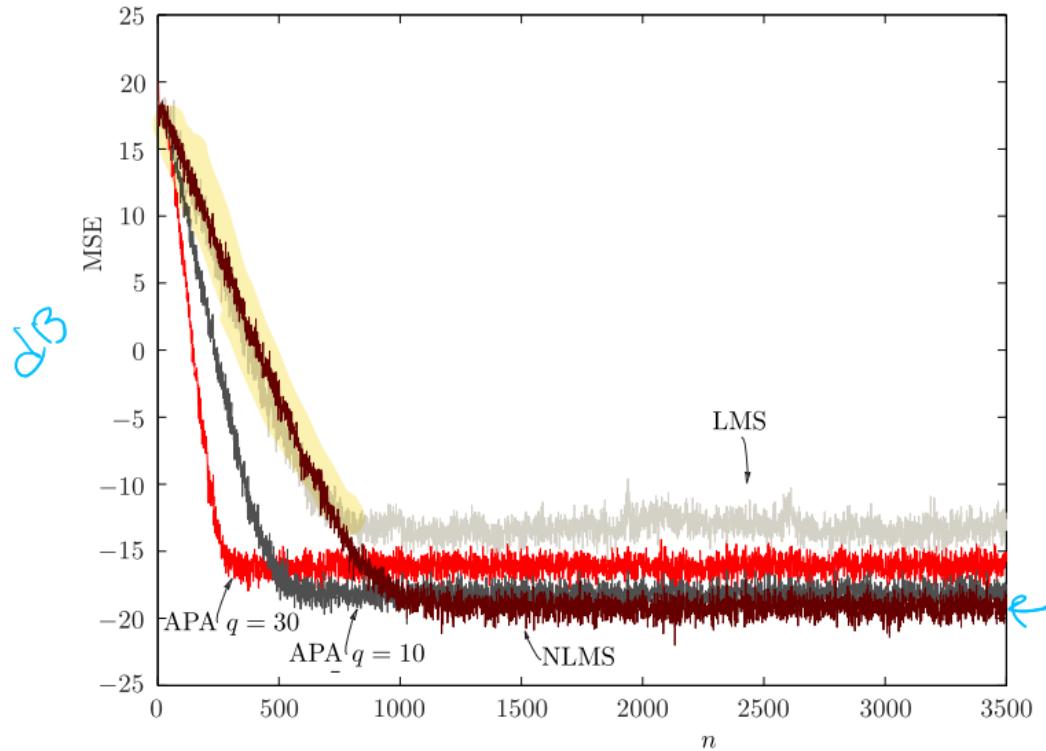
↑
energy of input

Parameters:

μ is the step size.

l is a filter length.

Convergence rates



Recursive Least Squares (RLS)

Recursive Least Squares (RLS)

RLS teaser – stationary environment convergence rate

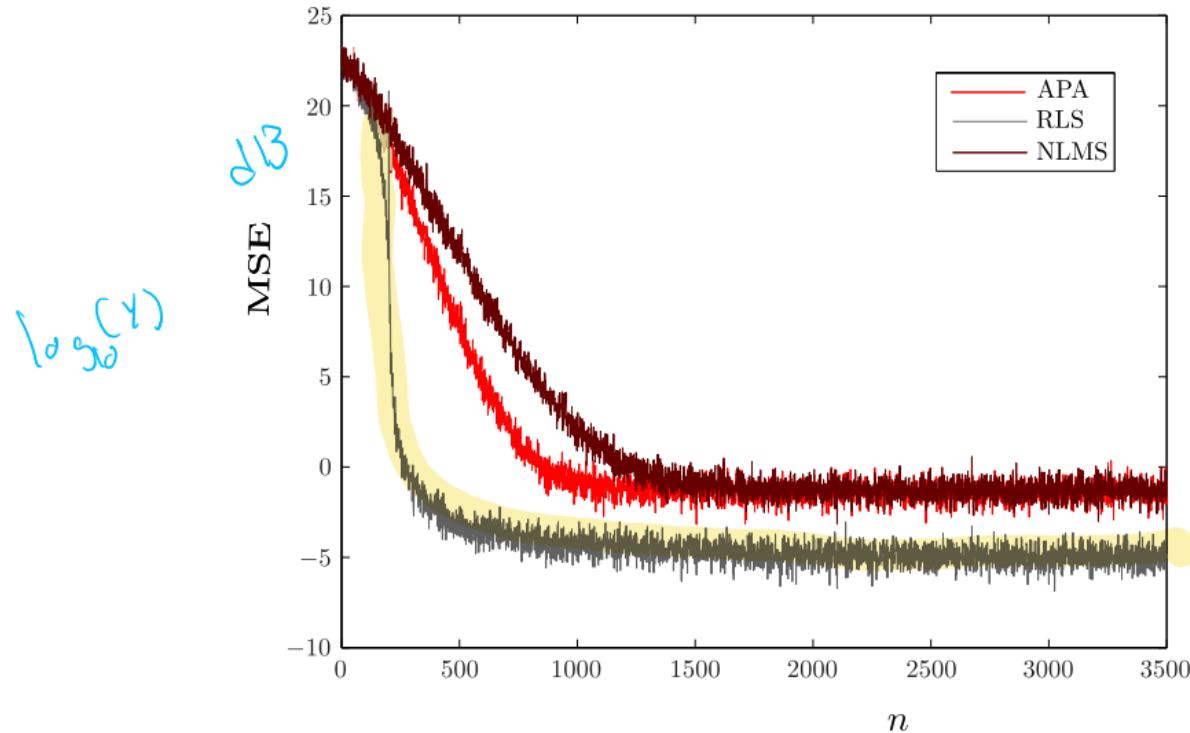


Fig 6.8
ex 5.2

The Recursive Least-Squares – algorithm 6.1

RLS

- Initialize

- $\theta_{-1} = \mathbf{0} \in \mathbb{R}^l$; any other value is also possible.
- $P_{-1} = \lambda^{-1}I$; $\lambda > 0$ a user-defined variable
- Select β close to 1. *Matrix*

- For $n = 0, 1, \dots$, Do

- $e_n = y_n - \theta_{n-1}^T \mathbf{x}_n$ *"lms"*
- $\mathbf{z}_n = P_{n-1} \mathbf{x}_n$
- $\mathbf{k}_n = \frac{\mathbf{z}_n}{\beta + \mathbf{x}_n^T \mathbf{z}_n}$
- $\theta_n = \theta_{n-1} + \mathbf{k}_n e_n$ *"lms"*
- $P_n = \beta^{-1} P_{n-1} - \beta^{-1} \mathbf{k}_n \mathbf{z}_n^T$

- End For

Parameters:

β forget factor.

λ is regularization.

- RLS obtains superior convergence rate compared to ~~RLS~~ (or so it seems, details to follow).
- The algorithm is far more complex and computationally demanding so it is not all good.

Stationary simulation model and steady-state convergence analysis of LMS

Stationary simulation model and steady-state convergence analysis of LMS

How do we simulate stationary environment?

ML 6.12

Steady state environment

- $\theta_o \in \mathbb{R}^l$ Optimal filter weights.
- η_n is zero mean Gaussian i.i.d (white noise).

$$\text{output } y_n = \theta_o^T x_n + \eta_n \text{ noise}$$

↑
optimal

$$E_{\theta} = \| \Theta \cdot \theta_o \|_2 \rightarrow 0$$

Stationary simulation model and steady-state convergence analysis of LMS

Analysis of coefficients

Define the coefficient error vector as

$$c_n := \theta_n - \theta_*$$

current estimate
optimal

Analyzing convergence in the mean,

$$\mathbb{E}[c_n] \rightarrow 0$$

as $n \rightarrow \infty$ we get the following results

$$0 < \mu < \frac{2}{\lambda_{\max}}$$

eigenvalue of Σ_x

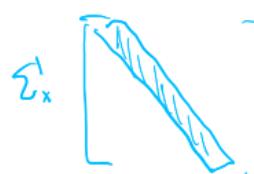
$$\lambda_{\max, \text{small } \Sigma} < \lambda_{\max, \text{big } \Sigma}$$

However analyzing the covariance matrix, $n \rightarrow \infty$

$$\Sigma_{c,n} := \mathbb{E}[c_n c_n^T]$$

$$0 < \mu < \frac{2}{\text{trace}\{\Sigma_x\}} = \frac{2}{l r_x(0)}$$

filter length



Stationary simulation model and steady-state convergence analysis of LMS

Convergence analysis of error

Define the misalignment as

$$J_{\text{tot}} = J_{\min} + J_{\text{exc}}$$

Misalignment

$$\mathcal{M} := \frac{J_{\text{exc}}}{J_{\min}} = 0$$

Analyzing the mean squared error behavior, we get the error at step n :

$$J_n := \mathbb{E} [e_n^2] = \dots \quad (\text{eq 5.46 - 5.47})$$

$$= J_{\min} + \text{trace} \{ \Sigma_x \Sigma_{c,n-1} \}$$

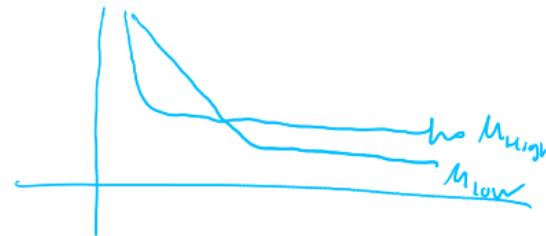
always $J_{\text{exc}} \rightarrow$ would like to go to 0

Excess MSE at time instant n

$$J_{\text{exc},n} = \text{trace} \{ \Sigma_x \Sigma_{c,n-1} \} : \quad \text{excess MSE at time instant } n$$

Stationary simulation model and steady-state convergence analysis of LMS

Practical misalignment for LMS



For small values of μ , we get

Expected misalignment

$$\mathcal{M} \simeq \frac{1}{2}\mu \text{trace}\{\Sigma_x\} : \quad \text{misadjustment}$$

Stationary simulation model and steady-state convergence analysis of LMS LMS summary

Defining the coefficient error vector as $c_n := \theta_n - \theta_*$ and require convergence on the diagonal of the covariance matrix $\Sigma_{c,n} := \mathbb{E} [\mathbf{c}_n \mathbf{c}_n^T]$ we got

LMS Step size bounds

$$0 < \mu < \frac{2}{\text{trace}\{\Sigma_x\}} = \frac{2}{l \cdot r_x(0)}$$

Misalignment and excess MSE

$$J_n = J_{\min} + J_{\text{exc}}$$

$$\mathcal{M} := \frac{J_{\text{exc}}}{J_{\min}}$$

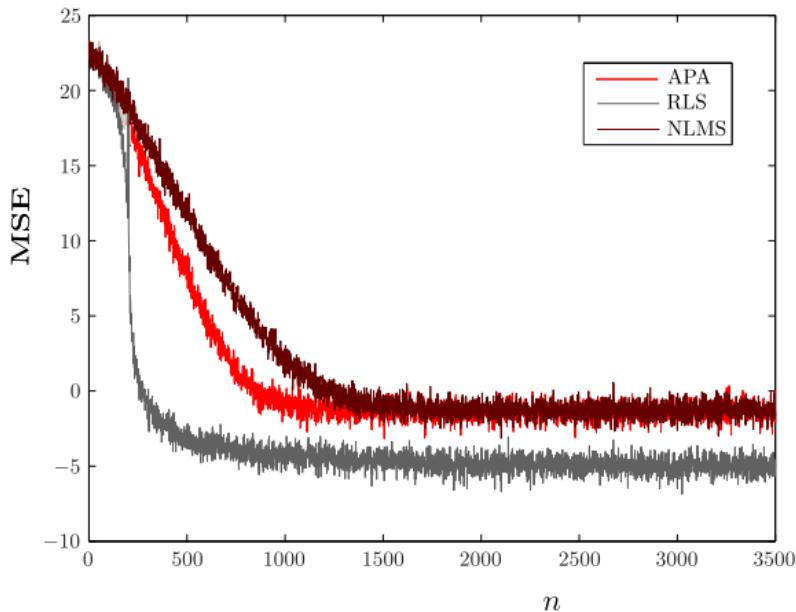
Excess MSE and misadjustment for LMS at time instant n

$$J_{\text{exc},n} = \text{trace}\{\Sigma_x \Sigma_{c,n-1}\}$$

$$\mathcal{M} \simeq \frac{1}{2} \mu \text{trace}\{\Sigma_x\} : \quad \text{misadjustment}$$

Stationary simulation model and steady-state convergence analysis of LMS

Convergence rates of RLS – stationary environment (example 6.1)



Data generation model:

$$y_n = \boldsymbol{\theta}_o^T \mathbf{x}_n + \eta_n, \quad \boldsymbol{\theta} \in \mathbb{R}^{200}, \eta_n \sim \mathcal{N}(0, 0.01I)$$

For APA: $\mu = 0.2, \delta = 0.001, q = 30$.

For RLS: $\beta = 1, \lambda = 0.1$.

For NLMS: $\mu = 1.2, \delta = 0.001$.

APA and NLMS parameters chosen so algorithms converge to same noise floor.

RLS parameters chosen for optimal noise floor.

You will create these simulations in the exercise today.

Non-stationary simulation model and convergence analysis

Non-stationary simulation model and convergence analysis

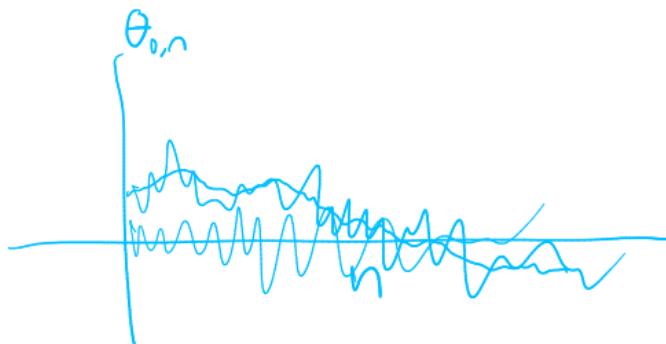
How do we simulate non-stationary environment?

Non-stationary environment

$$y_n = \theta_{o,n}^T \mathbf{x}_n + \eta_n$$

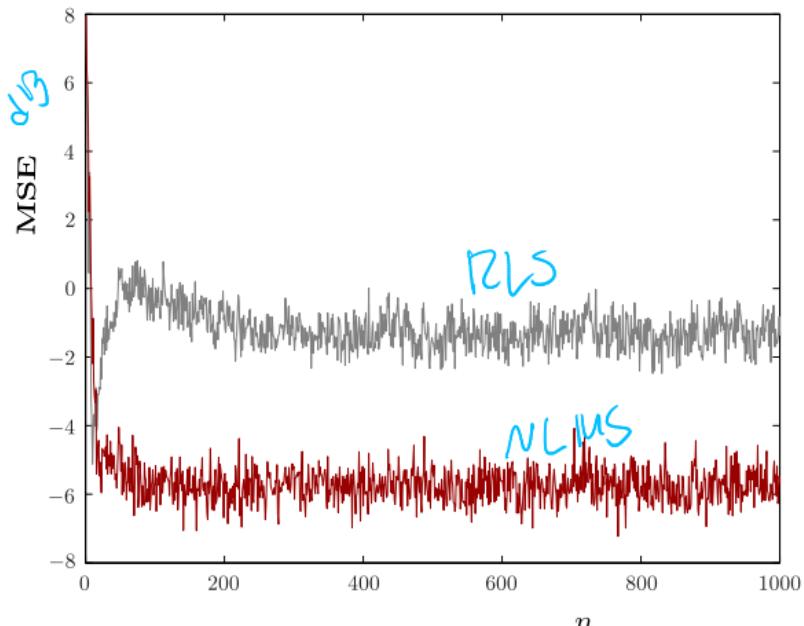
$$\theta_{o,n} = \alpha \theta_{o,n-1} + \omega_n \quad \text{AR(1)}$$

- $\alpha < 1$ is the autoregressive coefficient (memory).
- ω_n is zero mean Gaussian i.i.d with diagonal covariance (white noise).



Non-stationary simulation model and convergence analysis

Convergence rates of RLS, time-tracking (example 6.2)



Gray: RLS, Red: NLMS

ex 6.3

Data generation model:

$$y_n = \boldsymbol{\theta}_o^T \mathbf{x}_n + \eta_n, \quad \boldsymbol{\theta} \in \mathbb{R}^5, \quad \eta_n \sim \mathcal{N}(0, 0.01I)$$

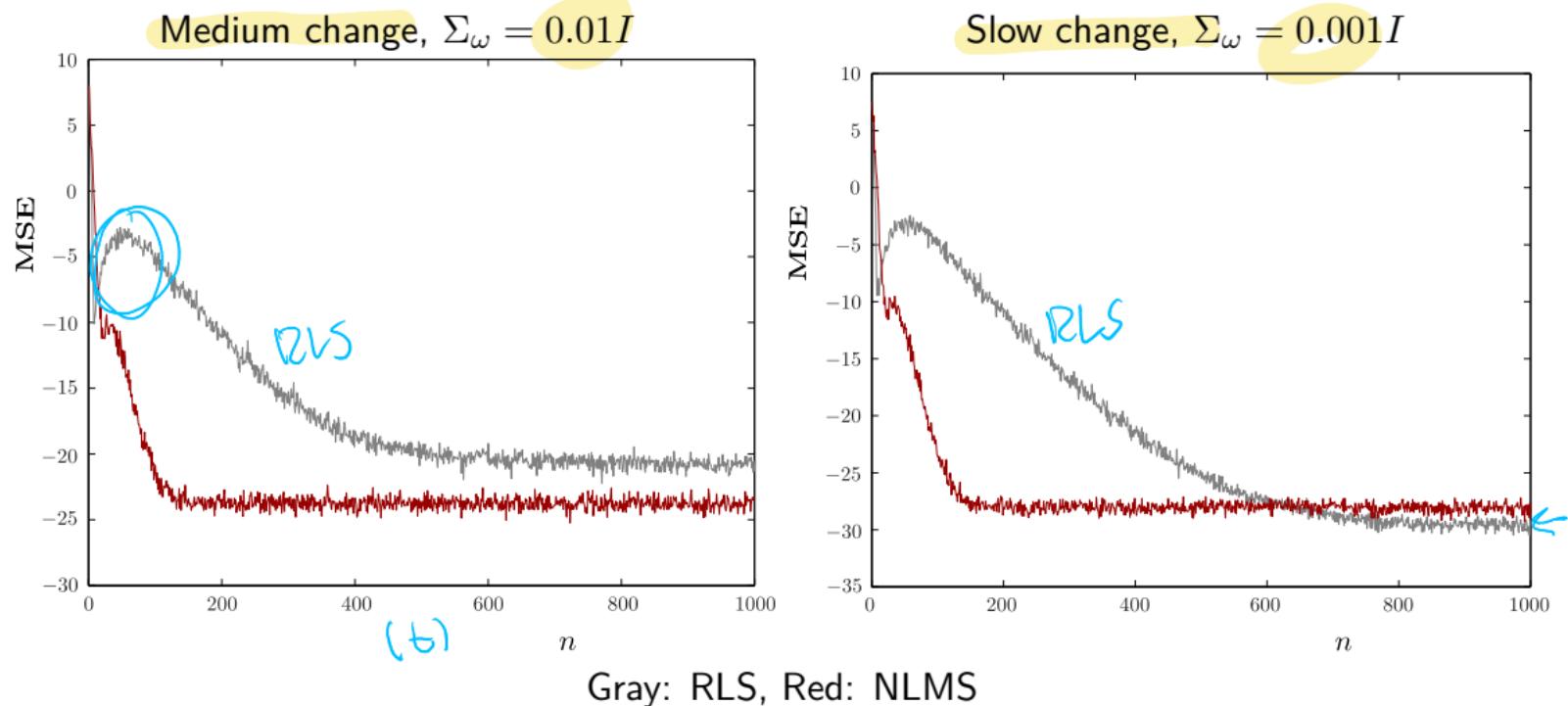
$$\boldsymbol{\theta}_{o,n} = \alpha \boldsymbol{\theta}_{o,n-1} + \omega_n, \quad \alpha = 0.97, \quad \omega_n \sim \mathcal{N}(0, 0.1I)$$

For RLS: $\beta = 0.995, \lambda = 0.001$.

For NLMS: $\mu = 0.5, \delta = 0.001$.

Parameters were selected for best performance.

Non-stationary simulation model and convergence analysis
Convergence rates of RLS, time-tracking



Convergence analysis of LMS under non-stationary environment

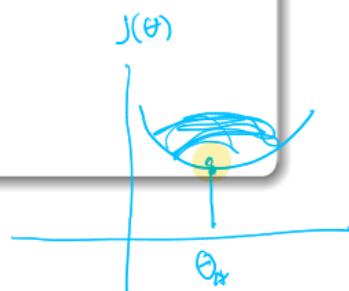
Ch 5.12

Stationary environment excess MSE

$$J_{\text{exc}} \simeq \frac{1}{2} \mu \sigma_\eta^2 \text{trace} \{ \Sigma_x \}$$

Non-stationary environment excess MSE

$$J_{\text{exc}} \simeq \frac{1}{2} \left(\underbrace{\mu \sigma_\eta^2 \text{trace} \{ \Sigma_x \}}_{\text{excess}} + \underbrace{\frac{1}{\mu} \text{trace} \{ \Sigma_\omega \}}_{\text{lag}} \right)$$



Is there anything in particular we should consider?

$$J = J_{\min} + J_{\text{exc}} + J_{\text{lag}}$$

Non-stationary simulation model and convergence analysis

Excess MSE for the adaptive algorithms (table 6.1)

If we analyze the error rates under the presented non-stationary model, we obtain the following at steady-state (assumptions and limitations apply!!)

Algorithm	Excess MSE (l denotes filter length)
LMS	$J(\mu) \frac{1}{2} \mu \sigma_\eta^2 \text{trace}\{\Sigma_x\} + \frac{1}{2} \mu^{-1} \text{trace}\{\Sigma_\omega\}$ $\Rightarrow MK_1 + \mu^1 K_2$
APA	$\frac{1}{2} \mu \sigma_\eta^2 \text{trace}\{\Sigma_x\} \mathbb{E}\left[\frac{q}{\ x\ ^2}\right] + \frac{1}{2} \mu^{-1} \text{trace}\{\Sigma_x\} \text{trace}\{\Sigma_\omega\}$
NLMS	$\frac{1}{2} \mu \sigma_\eta^2 \text{trace}\{\Sigma_x\} \frac{1}{\sigma_x^2(l-2)} + \frac{1}{2} \mu^{-1} \text{trace}\{\Sigma_x\} \text{trace}\{\Sigma_\omega\}$
RLS	$\frac{1}{2}(1-\beta)\sigma_\eta^2 l + \frac{1}{2}(1-\beta)^{-1} \text{trace}\{\Sigma_\omega \Sigma_x\}$

With these results, we can find the optimal μ or β (how??), and e.g. obtain the following:

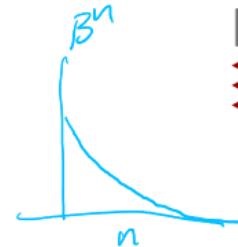
LMS/RLS expected error ratio

$$\frac{J_{\min}^{\text{LMS}}}{J_{\min}^{\text{RLS}}} = \sqrt{\frac{\text{trace}\{\Sigma_x\} \text{trace}\{\Sigma_\omega\}}{l \text{trace}\{\Sigma_\omega \Sigma_x\}}}$$

- Many applications have time-changing environment that calls for adaptive algorithms
- RLS can have superior convergence speed compared to NLMS
- RLS is not always better though, it has higher computational complexity and can also diverge due to finite floating point precision.
- NLMS is very stable and the “safe” choice.

Derivation of the RLS algorithm

Where does RLS come from

 $B=1 \rightarrow$ Ridge Reg

Exponentially weighted least-squares

$$J(\theta, \beta, \lambda) = \sum_{i=0}^n \beta^{n-i} (y_i - \theta^T x_i)^2 + \lambda \beta^{n+1} \|\theta\|^2$$

design

RLS solves the minimization problem at observation n :

$$\theta_n = \min_{\theta} J(\theta, \beta, \lambda)$$

$\propto \beta^{-1}$ num of data points
 Have you seen such cost function before? Discuss the roles of β , λ , and n
 Regularizer

Derivation of the RLS algorithm

Minimizing the cost function

Exponentially weighted least-squares

$$J(\boldsymbol{\theta}, \beta, \lambda) = \sum_{i=0}^n \beta^{n-i} (y_i - \boldsymbol{\theta}^T \mathbf{x}_i)^2 + \lambda \beta^{n+1} \|\boldsymbol{\theta}\|^2$$

- ① Take the derivative wrt $\boldsymbol{\theta}$ and set to zero.
- ② Create time-iterative updates, i.e. on the form $\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \Delta$

Derivation of the RLS algorithm

Minimizing the cost function



Exponentially weighted least-squares

$$J(\boldsymbol{\theta}, \beta, \lambda) = \sum_{i=0}^n \beta^{n-i} (y_i - \boldsymbol{\theta}^T \mathbf{x}_i)^2 + \lambda \beta^{n+1} \|\boldsymbol{\theta}\|^2$$

The derivative is

$$\frac{d}{d\boldsymbol{\theta}} J(\boldsymbol{\theta}, \beta, \lambda) = -2 \sum_{i=0}^n \beta^{n-i} (y_i - \boldsymbol{\theta}^T \mathbf{x}_i) \mathbf{x}_i + 2\lambda \beta^{n+1} I \boldsymbol{\theta}$$

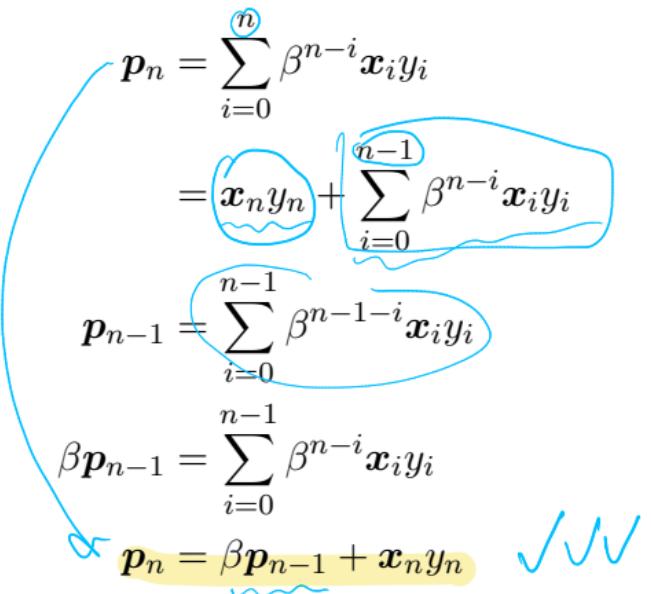
Which leads to

$$\Phi_n := \sum_{i=0}^n \beta^{n-i} \mathbf{x}_i \mathbf{x}_i^T + \lambda \beta^{n+1} I$$

$$\mathbf{p}_n := \sum_{i=0}^n \beta^{n-i} \mathbf{x}_i y_i$$

$$\boldsymbol{\theta}_n = \Phi_n^{-1} \mathbf{p}_n$$

Deriving the time-iterative updates

$$\begin{aligned} \mathbf{p}_n &= \sum_{i=0}^n \beta^{n-i} \mathbf{x}_i y_i \\ &= \mathbf{x}_n y_n + \sum_{i=0}^{n-1} \beta^{n-i} \mathbf{x}_i y_i \\ \mathbf{p}_{n-1} &= \sum_{i=0}^{n-1} \beta^{n-1-i} \mathbf{x}_i y_i \\ \beta \mathbf{p}_{n-1} &= \sum_{i=0}^{n-1} \beta^{n-i} \mathbf{x}_i y_i \\ \mathbf{p}_n &= \beta \mathbf{p}_{n-1} + \mathbf{x}_n y_n \end{aligned}$$


Deriving the time-iterative updates

Using a similar approach for Φ_n , we arrive at

$$\begin{aligned}\Phi_n &= \beta\Phi_{n-1} + \mathbf{x}_n\mathbf{x}_n^T \\ p_n &= \beta p_{n-1} + \mathbf{x}_n y_n\end{aligned}$$

The goal was to find $\theta_n = \Phi_n^{-1} p_n$, thus we need to compute

$$\Phi_n^{-1} = (\beta\Phi_{n-1} + \mathbf{x}_n\mathbf{x}_n^T)^{-1}$$

We use Woodbury's inversion formula (in the exercise).

$$(A + BD^{-1}C)^{-1} = A^{-1} - A^{-1}B(D + CA^{-1}B)^{-1}CA^{-1}$$

What is A , B , C and D ?

Derivation of the RLS algorithm

Primary result



Inversion recursion, and Kalman gain k

$$\Phi_n^{-1} = \beta^{-1} \Phi_{n-1}^{-1} - \beta^{-1} \mathbf{k}_n \mathbf{x}_n^T \Phi_{n-1}^{-1}$$
$$\mathbf{k}_n = \frac{\beta^{-1} \Phi_{n-1}^{-1} \mathbf{x}_n}{1 + \beta^{-1} \mathbf{x}_n^T \Phi_{n-1}^{-1} \mathbf{x}_n}$$

For notational purposes we define $P_n = \Phi_n^{-1}$, and, for later, we rewrite \mathbf{k}_n , to get

$$P_n = \beta^{-1} P_{n-1} - \beta^{-1} \mathbf{k}_n \mathbf{x}_n^T P_{n-1}$$
$$\mathbf{k}_n = \left(\beta^{-1} P_{n-1} - \beta^{-1} \mathbf{k}_n \mathbf{x}_n^T P_{n-1} \right) \mathbf{x}_n$$
$$= P_n \mathbf{x}_n$$

Derivation of the RLS algorithm

RLS Summary

RLS minimized the exponentially weighted least-squares cost function

Exponentially weighted least-squares

$$J(\boldsymbol{\theta}, \beta, \lambda) = \sum_{i=0}^n \beta^{n-i} (y_i - \boldsymbol{\theta}^T \mathbf{x}_i)^2 + \lambda \beta^{n+1} \|\boldsymbol{\theta}\|^2$$

Choose forget factor β and regularization λ

We have an iterative update formula for $\boldsymbol{\theta}_n$, meaning we can update $\boldsymbol{\theta}_n$ at each new observation (y_n, \mathbf{x}_n) , instead of solving the cost function anew.

Update formula

$$\begin{aligned}\boldsymbol{\theta}_n &= \boldsymbol{\theta}_{n-1} + \mathbf{k}_n e_n \\ \mathbf{k}_n &= (\beta^{-1} P_{n-1} - \beta^{-1} \mathbf{k}_n \mathbf{x}_n^T P_{n-1}) \mathbf{x}_n \\ P_n &= \beta^{-1} P_{n-1} - \beta^{-1} \mathbf{k}_n \mathbf{x}_n^T P_{n-1} \\ e_n &= y_n - \mathbf{x}_n^T \boldsymbol{\theta}_{n-1}\end{aligned}$$

- The RLS algorithm is obtained by a modified Ridge regression cost function.
- The RLS can efficiently be implemented in a iterative update manner instead of solving the cost function completely at every step.
5.1
- It can diverge with finite precision arithmetic, and more robust implementations exists.
- You will derive this algorithm completely in the first exercise, and they replicate the convergence plots.

Relation between RLS and Newton's method

Newton's iterative minimization method**Newton's iterative scheme**

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} - \underbrace{\mu_i \left(\nabla^2 J(\boldsymbol{\theta}^{(i-1)}) \right)^{-1}}_{\text{Stepsize}} \nabla J(\boldsymbol{\theta}^{(i-1)})$$

Assuming $\boldsymbol{\theta}_*$ to be the minimum, quadratic convergence means that at each iteration i the deviation from the optimum value follows the following pattern

$$\ln \ln \frac{1}{\|\boldsymbol{\theta}^{(i)} - \boldsymbol{\theta}_*\|^2} \propto i : \quad \text{quadratic convergence rate}$$

$$\ln \frac{1}{\|\boldsymbol{\theta}^{(i)} - \boldsymbol{\theta}_*\|^2} \propto i : \quad \text{linear convergence rate}$$

Applying Newton's method to the mean squared error loss

$$\begin{aligned}
 J(\theta) &= \frac{1}{2} \mathbb{E} \left[(y - \theta^T \mathbf{x})^2 \right] = \frac{1}{2} \sigma_y^2 + \frac{1}{2} \theta^T \mathbb{E}[\mathbf{x} \mathbf{x}^T] \theta - \theta^T \mathbb{E}[\mathbf{x} y] \\
 -\nabla J(\theta) &= \mathbb{E}[\mathbf{x} y] - \mathbb{E}[\mathbf{x} \mathbf{x}^T] \theta = \mathbb{E} \left[\mathbf{x} \left(y - \mathbf{x}^T \theta \right) \right] = \mathbb{E}[\mathbf{x} e] \\
 \nabla^2 J(\theta) &= \mathbb{E}[\mathbf{x} \mathbf{x}^T] = \Sigma_x
 \end{aligned}$$

Now apply newtons iteration scheme on observations

$$\theta_n = \theta_{n-1} + \mu_n \Sigma_x^{-1} \mathbf{x}_n e_n$$

Newton's iterative scheme

$$\theta^{(i)} = \theta^{(i-1)} - \underbrace{\mu_i \left(\nabla^2 J(\theta^{(i-1)}) \right)^{-1}}_{\text{Stepsize}} \nabla J(\theta^{(i-1)})$$

Relation between RLS and Newton's method

Approximating the covariance matrix

From last slide

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \mu_n \boldsymbol{\Sigma}_x^{-1} \mathbf{x}_n e_n$$

If we apply the following approximation (why does this make sense?)

$$\boldsymbol{\Sigma}_x \simeq \frac{1}{n+1} \Phi_n = \left(\underbrace{\frac{1}{n+1} \lambda \beta^{n+1} I}_{\text{as } n \rightarrow \infty} + \frac{1}{n+1} \sum_{i=0}^n \beta^{n-i} \mathbf{x}_i \mathbf{x}_i^T \right)$$

data

Then we can obtain the RLS update formula for $\boldsymbol{\theta}_n$ by reading off the following coefficients:

$$\mu_n = \frac{1}{n+1}$$

$$\mathbf{k}_n = P_n \mathbf{x}_n$$

$$P_n = \left(\lambda \beta^{n+1} I + \sum_{i=0}^n \beta^{n-i} \mathbf{x}_i \mathbf{x}_i^T \right)^{-1}$$

Relation between RLS and Newton's method

Summary



- Newton's method is a second order method that obtains quadratic convergence speed.
- RLS can be derived using Newton's method, and thus reveals RLS is a method that uses the second order derivatives.
- This explains the superior convergence rate of RLS.

Relation between RLS and Newton's method

Lecture summary



- The recursive least squares (RLS) achieves quadratic convergence speed.
- RLS is computationally more expensive compared to LMS and APA. *NLMS*
- RLS will achieve superior steady state performance.
- For tracking performance, there are still tradeoffs to consider, and sometimes NLMS performance is better (and more robust).

Material: ML 8.2, 8.10.1–8.10.2, 9.1–9.5, 9.9

- Sparsity-aware learning.
- Alternatives to the ℓ_2 norm.
- Compressed sensing.
- Leads to dictionary learning (later).

02471 Machine Learning for Signal Processing

Sparsity-aware learning

Tommy Sonne Alstrøm

Cognitive Systems section

$$f(x+\Delta x) = \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x)$$

Complex mathematical symbols and equations are overlaid on the slide, including:
- A large blue integral symbol with a purple Greek letter Θ inside.
- A purple square root symbol with the number 17 inside.
- A red summation symbol with an exclamation mark at the bottom.
- A purple infinity symbol.
- A red equals sign with a blue fraction above it.
- A purple delta symbol with a red e^{iπ} term.
- A purple minus sign with a red -1 term.
- A purple plus sign with a red Ω symbol.
- A purple integral symbol with a red δ symbol.
- A purple infinity symbol with a red λ symbol.
- A purple summation symbol with a red λ symbol.
- A purple infinity symbol with a red χ symbol.
- A purple infinity symbol with a red ≈ symbol.
- A purple infinity symbol with a red > symbol.
- A purple infinity symbol with a red ≈ symbol.
- A purple infinity symbol with a red ≈ symbol.
- A purple infinity symbol with a red ≈ symbol.
- A purple infinity symbol with a red ≈ symbol.

Outline

- Course admin
- Last week review
- Sparsity-aware learning
- Norms
- LASSO
- Next Week

Material: ML 8.2, 8.10.1–8.10.2, 9.1–9.5, 9.9.

- Feedback from last week:
 - Would like to have more graphical illustrations of the procedures-
- **Feedback** from **you** is a critical component for improving both the course and my teaching.
- Type of feedback
 - Mention one thing that worked?
 - Mention one thing should be improved (both in current lecture and last weeks exercise)?
 - Mention one thing you would change if you gave the lecture.

The story so far and what the future holds

- Problem set 1 assignments are corrected (2nd hand-in due next week – see DTU Learn).
- Problem set 2 can be solved, except problem 2.6, which uses material for week 7.

What you have learned so far:

- Parameter estimation [L2 regularization, biased estimation, mean squared error minimization]. **Todo:** L1 regularization (this week), Bayesian parameter estimation (in 4 weeks)
- Filtering signals [Stochastic processes, correlation functions, Wiener filter, linear prediction, adaptive filtering using stochastic gradient decent (LMS, APA/NLMS), adaptive filtering using regularization (RLS)]

The topic the next three weeks will be sparse signal representations and dictionary learning

- Signal representations [Time frequency analysis, sparsity aware learning, factor models]
- Sparsity aware sensing (lasso, sparse priors), compressed sensing, dictionary learning [Independent component analysis, Non-negative matrix factorization, k -SVD]

Learning objectives

Learning objectives

A student who has met the objectives of the course will be able to:

- Explain, apply and analyze properties of discrete time signal processing systems
- Apply the short time Fourier transform to compute the spectrogram of a signal and analyze the signal content
- Explain compressed sensing and determine the relevant parameters in specific applications
- Deduce and determine how to apply factor models such as non-negative matrix factorization (NMF), independent component analysis (ICA) and sparse coding
- Deduce and apply correlation functions for various signal classes, in particular for stochastic signals
- Analyze filtering problems and demonstrate the application of least squares filter components such as the Wiener filter
- Describe, apply and derive non-linear signal processing methods based such as kernel methods and reproducing kernel Hilbert space for applications such as denoising
- Derive maximum likelihood estimates and apply the EM algorithm to learn model parameters
- Describe, apply and derive state-space models such as Kalman filters and Hidden Markov models
- Solve and interpret the result of signal processing systems by use of a programming language
- Design simple signal processing systems based on an analysis of involved signal characteristics, the objective of the processing system, and utility of methods presented in the course
- Describe a number of signal processing applications and interpret the results

Last week review

Last week review

The RLS algorithm

RLS minimized the exponentially weighted least-squares cost function

Exponentially weighted least-squares

$$J(\boldsymbol{\theta}, \beta, \lambda) = \sum_{i=0}^n \beta^{n-i} (y_i - \boldsymbol{\theta}^T \mathbf{x}_i)^2 + \lambda \beta^{n+1} \|\boldsymbol{\theta}\|^2$$

MSE Reg

Choose forget factor β and regularization λ .

We have an iterative update formula for $\boldsymbol{\theta}_n$, meaning we can update $\boldsymbol{\theta}_n$ at each new observation (y_n, \mathbf{x}_n) , instead of solving the cost function anew.

Update formula

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \mathbf{k}_n e_n$$

$$\mathbf{k}_n = (\beta^{-1} P_{n-1} - \beta^{-1} \mathbf{k}_n \mathbf{x}_n^T P_{n-1}) \mathbf{x}_n$$

$$P_n = \beta^{-1} P_{n-1} - \beta^{-1} \mathbf{k}_n \mathbf{x}_n^T P_{n-1}$$

$$e_n = y_n - \mathbf{x}_n^T \boldsymbol{\theta}_{n-1}$$

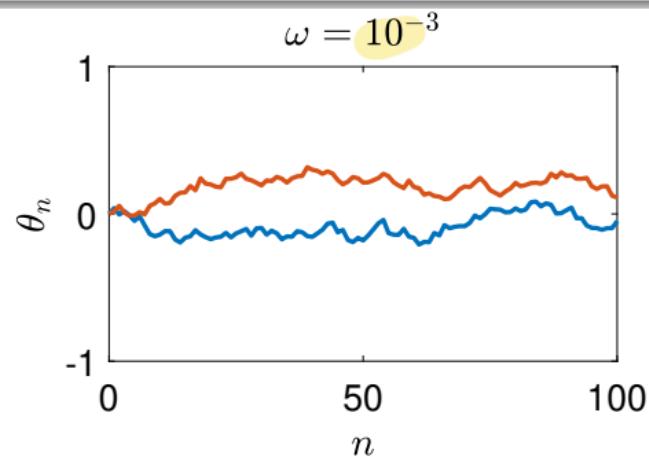
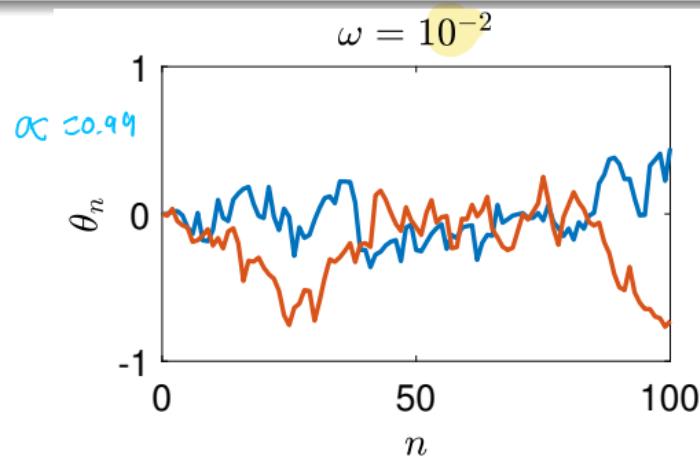
Last week review

How do we simulate non-stationary environment?

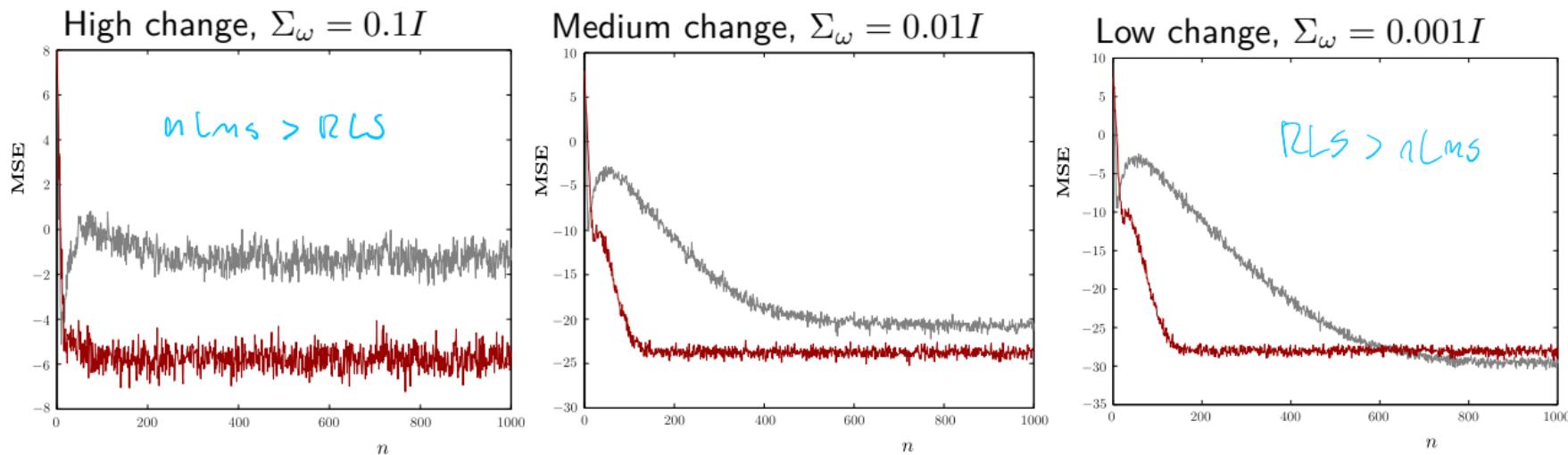
Non-stationary environment

$$y_n = \theta_o^T x_n + \eta_n \quad \text{obs}$$
$$\theta_{o,n} = \alpha \theta_{o,n-1} + \omega_n \quad \text{change}$$

- $\alpha < 1$ is the autoregressive coefficient (memory).
- ω_n is zero mean Gaussian i.i.d with diagonal covariance (white noise).



Convergence rates of RLS, time-tracking



Gray: RLS, Red: NLMS. y -axis is in dB.

Convergence results explained by $J_{\text{exc}} = J_{\text{excess}} + J_{\text{lag}}$.

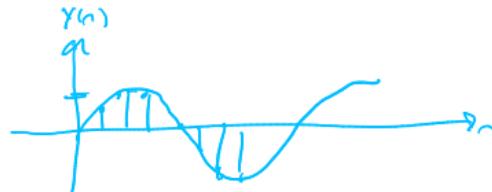
For NLMS: $J_{\text{exc}} = \frac{1}{2}\mu\sigma_\eta^2 \text{trace}\{\Sigma_x\} \frac{1}{\sigma_x^2(l-2)} + \frac{1}{2}\mu^{-1} \text{trace}\{\Sigma_x\} \text{trace}\{\Sigma_\omega\}$

For RLS: $J_{\text{exc}} = \frac{1}{2}(1-\beta)\sigma_\eta^2 l + \frac{1}{2}(1-\beta)^{-1} \text{trace}\{\Sigma_\omega \Sigma_x\}$.

- RLS can under certain circumstances converge faster than NLMS
 - There exist more theory on convergence analysis of RLS vs LMS, on when to choose RLS over LMS, it depends on the signal properties
- We need to consider both time-tracking performance and stationary performance.
- We have models to simulate both scenarios.
- Even though the algorithms are "old", they are still very relevant.

Sparsity-aware learning

What is sparsity-aware learning



In a number of practical problems, it is known that either the underlying model is sparse or it is sparse in a “transform” domain, e.g., in the Fourier transform domain.

By **sparse models** is meant that most of the unknown coordinates in the model are zero.

Recall from Chapter 3, the use of a regularizer can shrink the norm of the obtained solution. In this vein, we will see that by adopting appropriate **regularizers**, one can help the optimization process to identify the coordinates of the zeros.

Leads to **compressed sensing**, where the goal is to directly acquire **as few samples as possible** that encode the minimum information, which is needed to obtain a **compressed signal representation**.

Algorithms covered so far will struggle with either non-white noise, or presence of outliers.

A linear system with multiple solutions (under-determined system):

$$y = \mathbf{a} \cdot \mathbf{x} \Rightarrow 2 = \begin{bmatrix} 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ $\begin{bmatrix} 0 \\ 2 \end{bmatrix}$ $\begin{bmatrix} 2 \\ 0 \end{bmatrix}$

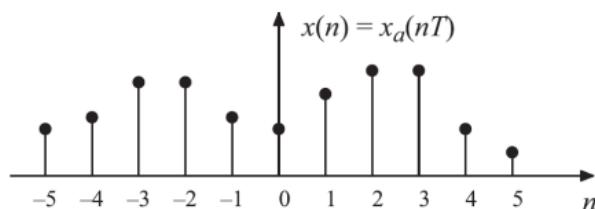
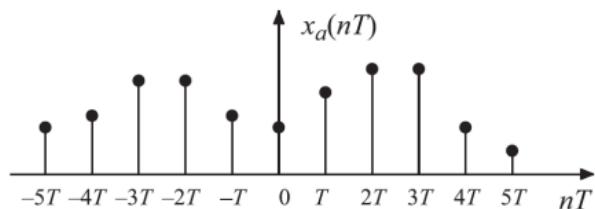
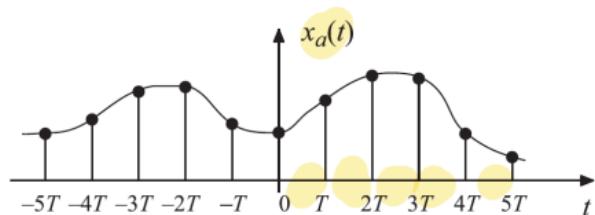
Which solution would you pick?

Sparsity-aware learning

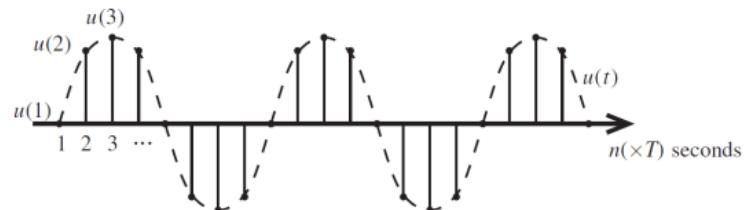
Traditional sampling rate

$$F_{\max} : F_s \geq 2 \cdot F_{\max}$$

Traditional sampling



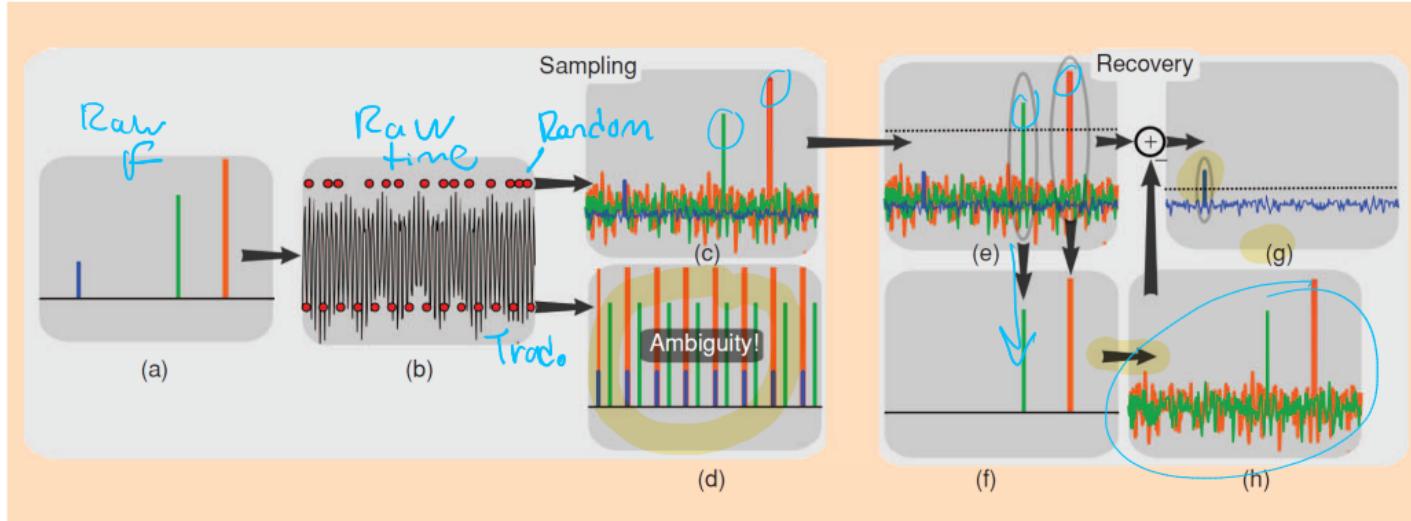
Suppose you were given this signal:



Can you sample more clever?

Sparsity-aware learning

Traditional under-sampling

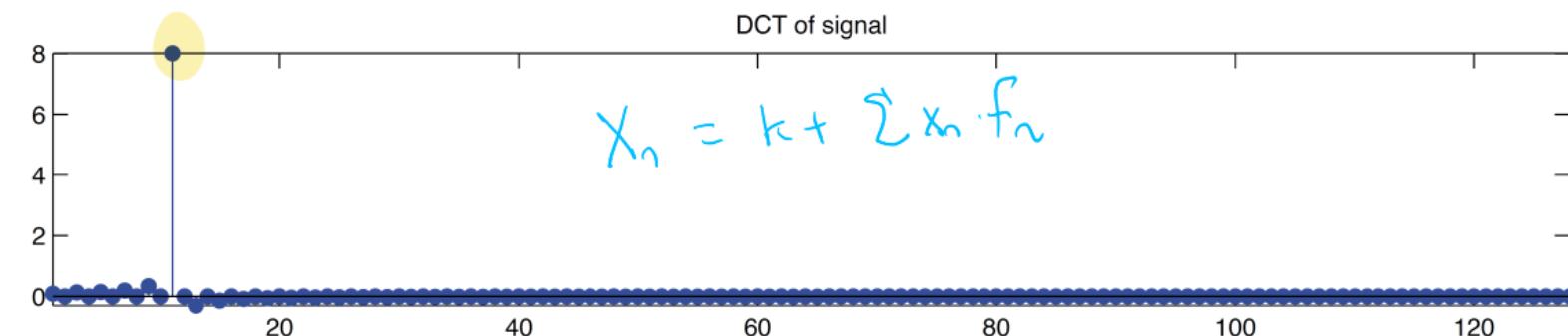
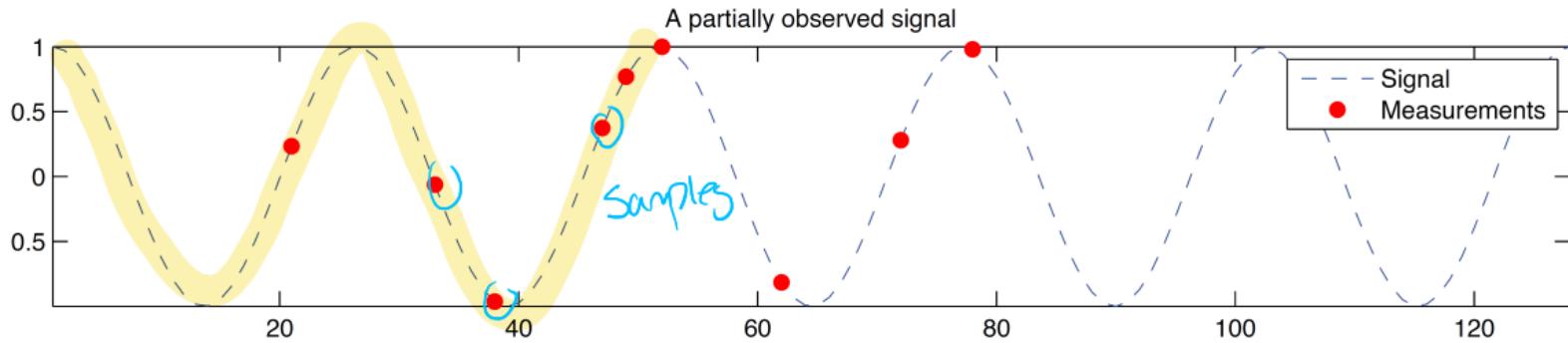


[FIG5] Heuristic procedure for reconstruction from undersampled data. A sparse signal (a) is 8-fold undersampled in its 1-D k -space domain (b). Equispaced undersampling results in signal aliasing (d) preventing recovery. Pseudo-random undersampling results in incoherent interference (c). Some strong signal components stick above the interference level, are detected and recovered by thresholding (e) and (f). The interference of these components is computed (g) and subtracted (h), thus lowering the total interference level and enabling recovery of weaker components.

M. Lustig, D. L. Donoho, J. M. Santos and J. M. Pauly, "Compressed Sensing MRI," in IEEE Signal Processing Magazine, vol. 25, no. 2, pp. 72-82, March 2008.

Sparsity-aware learning

Random sampling

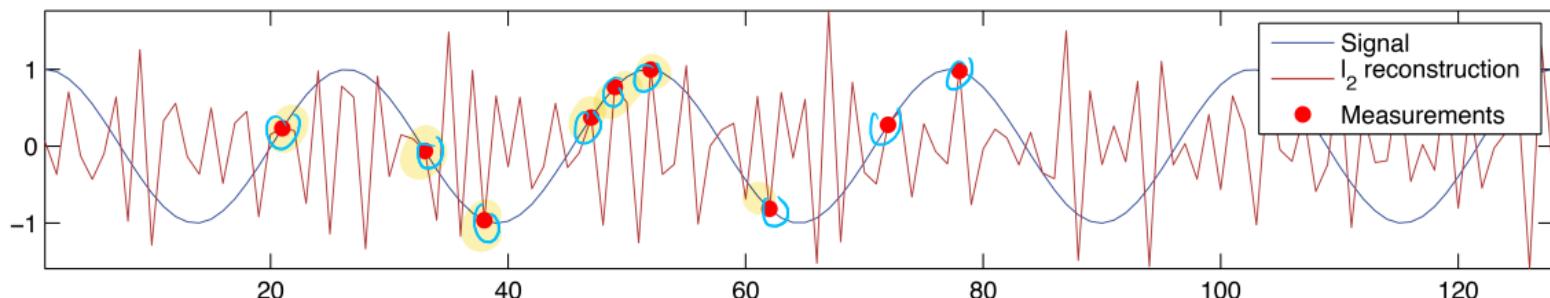


Example courtesy Prof. Paris Smaragdis

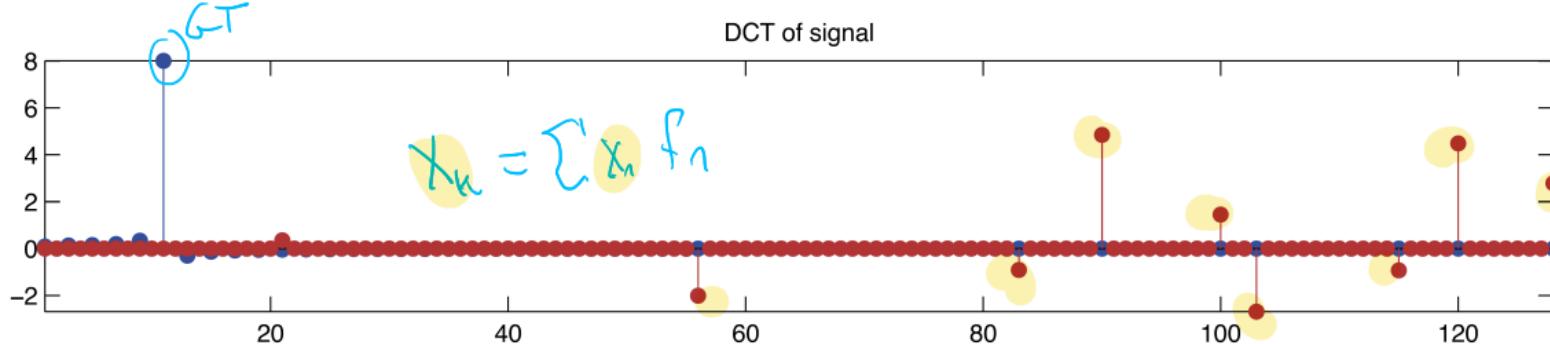
Sparsity-aware learning

Reconstruction using ℓ_2

A partially observed signal



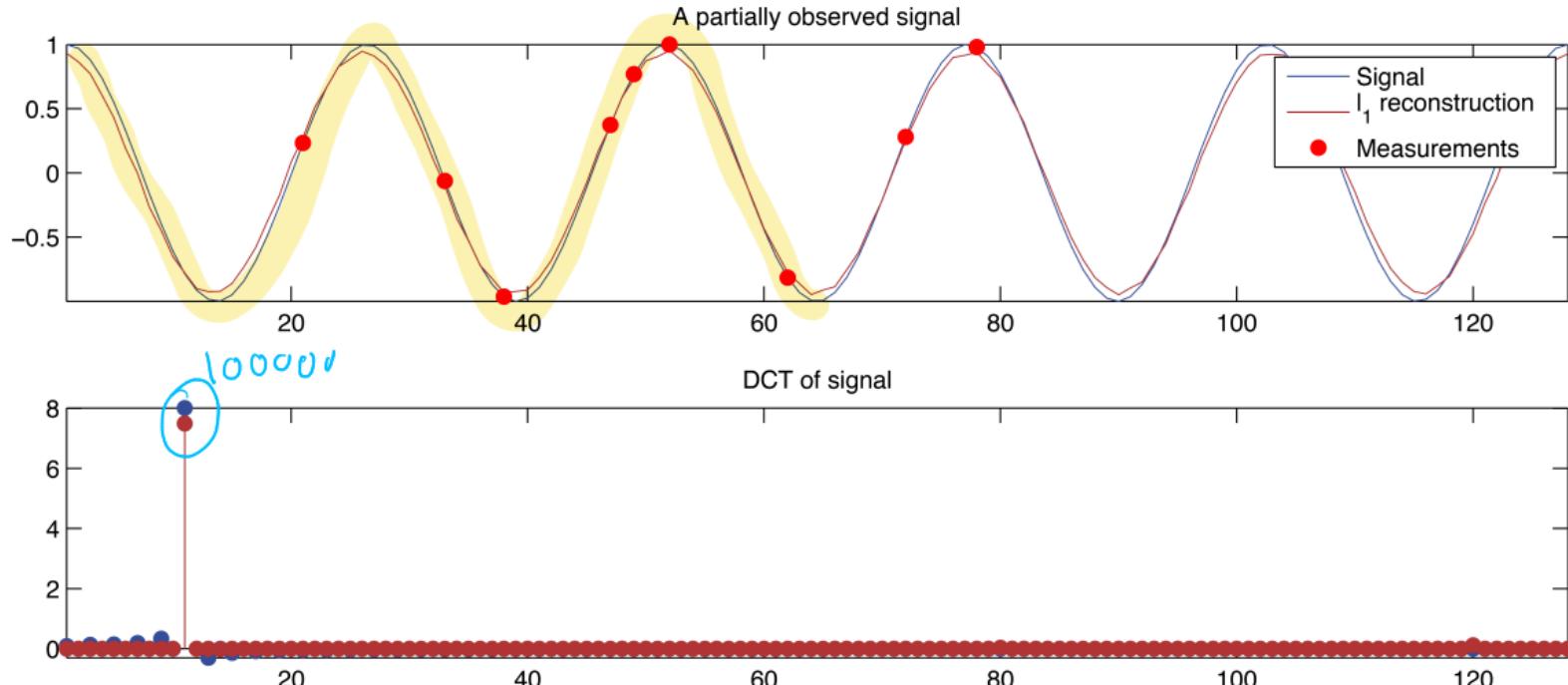
DCT of signal



Example courtesy Prof. Paris Smaragdis

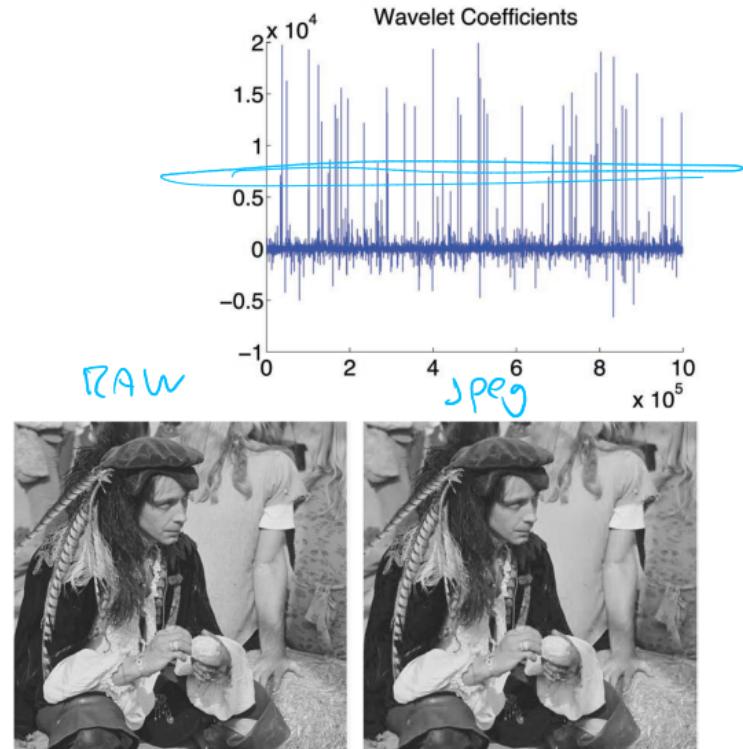
Sparsity-aware learning

Reconstruction using ℓ_1



Example courtesy Prof. Paris Smaragdis

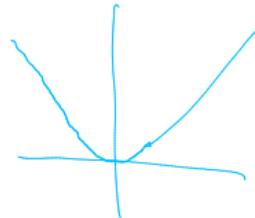
Sparsity-aware learning Wavelet compression



Sparsity-aware learning have many cool applications, in statistics, signal processing, and machine learning.

- Signal recovery in under-determined system.
- Leads to dictionary learning.
- ... and to compressed sensing, where we sample and compress at the same time.


Norms



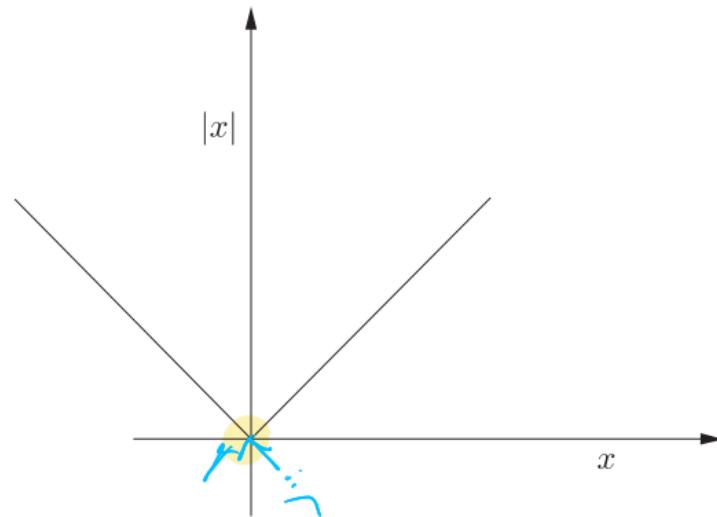
Ridge regression

$$J(\boldsymbol{\theta}, \lambda) = \sum_{i=1}^N (y_i - \boldsymbol{\theta}^T \mathbf{x}_i)^2 + \lambda \|\boldsymbol{\theta}\|_2^2 \quad eq.(3.39)$$

Can we think of another way to construct our regularization to promote sparsity

How about the absolute value?

$$f(x) = |x|$$



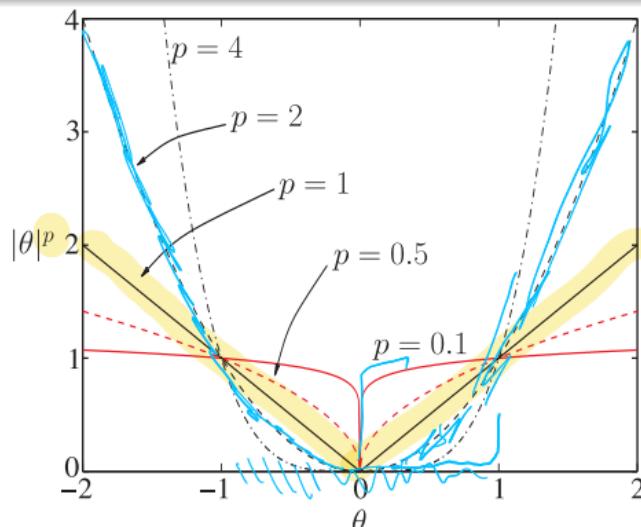
Is there any problems in using this function?

The ℓ_p normThe ℓ_p norm

$$\|\theta\|_p := \left(\sum_{i=1}^l |\theta_i|^p \right)^{1/p}$$

 $p \geq 1$

l is the size of vector θ .



Norms

Norms

The ℓ_p norm

$$\|\theta\|_p := \left(\sum_{i=1}^l |\theta_i|^p \right)^{1/p}, \quad p \geq 1$$

l is the size of vector θ .

Definition of norm

Let V be a vector space. A norm on V is a function $\|\cdot\| : V \rightarrow \mathbb{R}$ that satisfies the following three conditions, $\forall \theta \in V$:

- 1 ① $\|\theta\|_p \geq 0$, $\|\theta\|_p = 0 \Leftrightarrow \theta = 0$ ✓
 - 2 ② $\|\alpha\theta\|_p = |\alpha| \|\theta\|_p$, $\forall \alpha \in \mathbb{R}$
 - 3 ③ $\|\theta_1 + \theta_2\|_p \leq \|\theta_1\|_p + \|\theta_2\|_p$ (triangle inequality)
- $p = \frac{1}{2}$ $\theta = 1$ $\{\sqrt{1}, \sqrt{1}\}^2 = \sqrt{2}$

Why is ℓ_p not a norm for $p < 1$? can you create an example that breaks the triangle inequality?

How does the ℓ_p norm behave

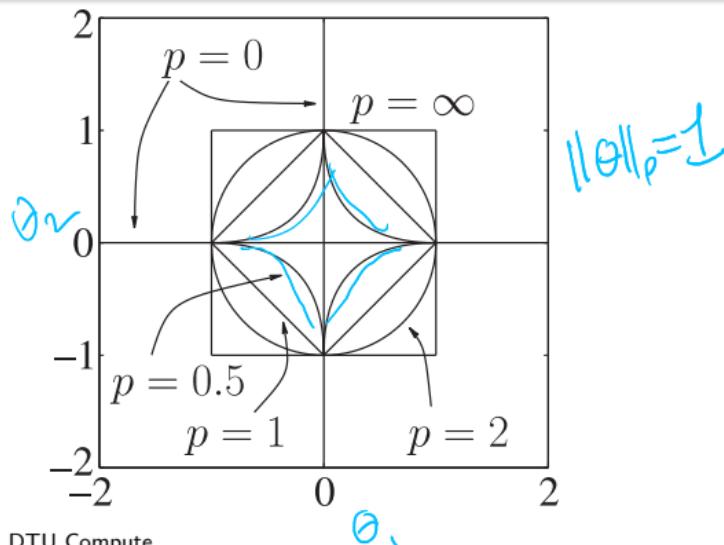
Special cases

$$\|\theta\|_\infty := \arg \max_i |\theta_i| \quad (\text{max element})$$

$$\ell_p \quad \lim_{p \rightarrow \infty}$$

$$\|\theta\|_0 := |\{i \mid \theta_i \neq 0, i = 1, \dots, l\}| \quad (\text{number of nonzeros})$$

$$\lim_{p \rightarrow 0}$$



Notation remarks:

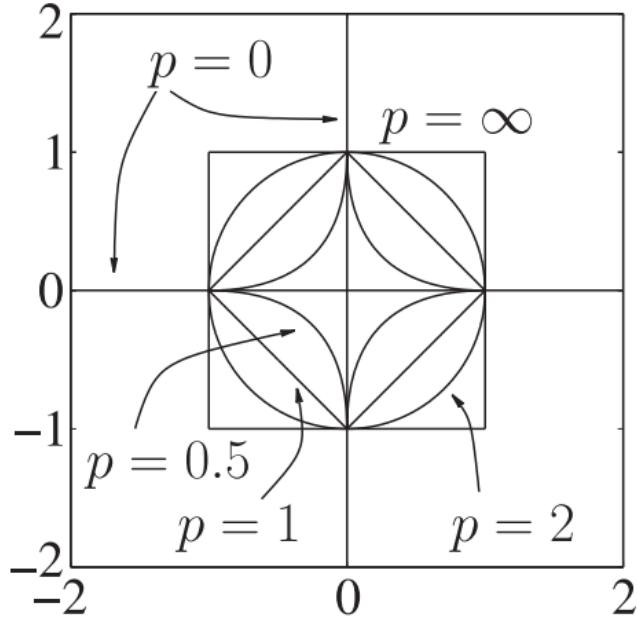
$|x|$ is the numerical value when $x \in \mathbb{R}$.

$|\mathcal{X}|$ is the cardinality (size) of the set \mathcal{X} .

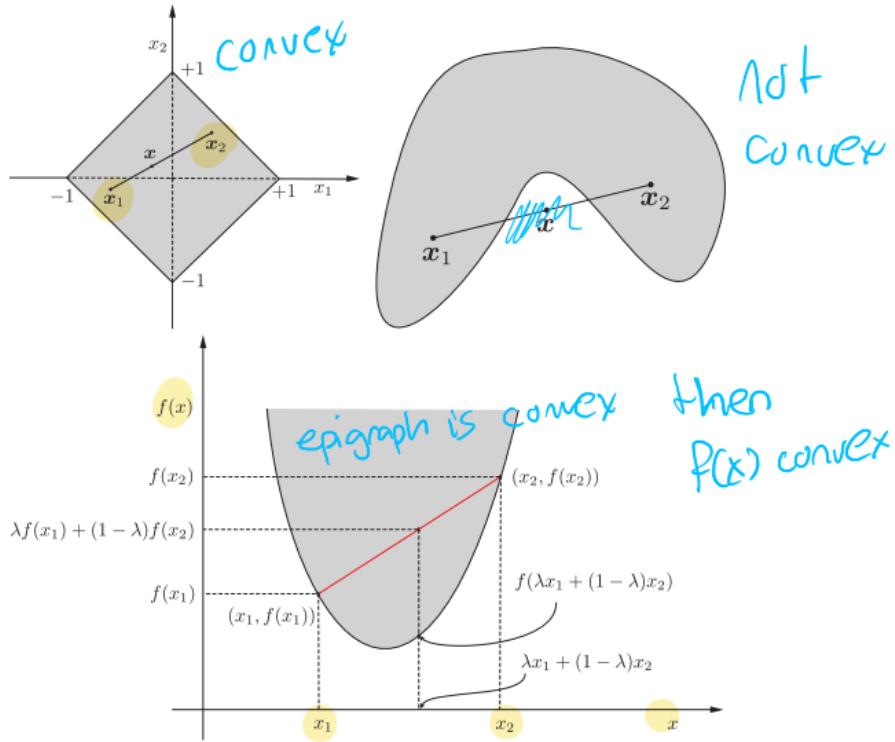
How can these different norms promote sparsity?

Norms

The ℓ_p norm is convex ($p \geq 1$)



Ch 8



- The ℓ_p norm can in different configurations regularize our learning problem.
- ℓ_p is only a true norm for $p \geq 1$. ex 6.1
- ℓ_0 is the most sparse "norm", and counts the number of nonzero elements.
- The ℓ_1 norm is the most sparse true norm, and is convex, hence we can optimize it.
- The ℓ_1 norm is not differentiable though.

$f_{\ell_1}(\cdot)$

For a rigorous treatment of norms: 01325 – Function spaces and mathematical analysis.

For a rigorous treatment of optimization: 02612 – Constrained Optimization.

LASSO

Exponentially weighted least-squares

$$J(\boldsymbol{\theta}, \beta, \lambda) = \sum_{i=0}^n \beta^{n-i} (y_i - \boldsymbol{\theta}^T \mathbf{x}_i)^2 + \lambda \beta^{n+1} \|\boldsymbol{\theta}\|_2^2$$

Recursive LASSO cost function

$$J(\boldsymbol{\theta}, \beta, \lambda) = \sum_{i=0}^n \beta^{n-i} (y_i - \boldsymbol{\theta}^T \mathbf{x}_i)^2 + \lambda \beta^{n+1} \|\boldsymbol{\theta}\|_1$$

Ref: D. Angelosante and G. B. Giannakis, "RLS-weighted Lasso for adaptive estimation of sparse signals," 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, Taipei, 2009, pp. 3245-3248.

RLS-weighted Lasso convergence

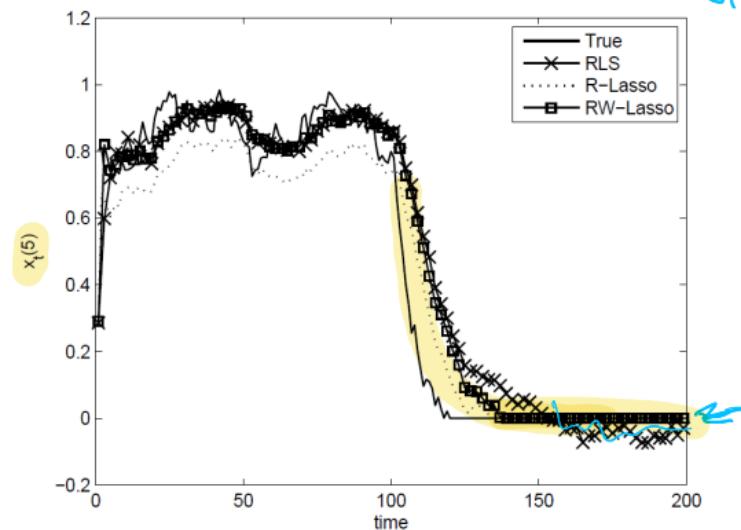


Fig. 3. Time evolution of the x_5 signal entry.

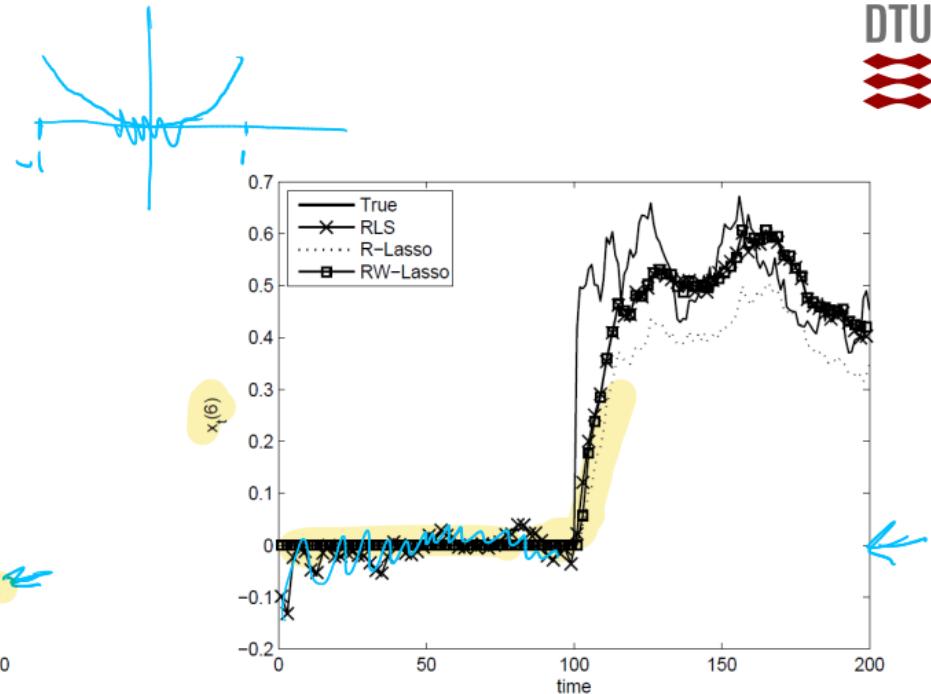


Fig. 4. Time evolution of the x_6 signal entry.

Ref: D. Angelosante and G. B. Giannakis, "RLS-weighted Lasso for adaptive estimation of sparse signals," 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, Taipei, 2009, pp. 3245-3248.

The Least Absolute Shrinkage and Selection Operator (LASSO)

We have the familiar regression task (on matrix form)

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\eta}, \quad \mathbf{y} \in \mathbb{R}^N, \mathbf{X} \in \mathbb{R}^{N \times l}, \boldsymbol{\theta} \in \mathbb{R}^l, \boldsymbol{\eta} \in \mathbb{R}^N,$$

LASSO cost function

$$J(\boldsymbol{\theta}, \lambda) = \sum_{i=1}^N (y_i - \boldsymbol{\theta}^T \mathbf{x}_i)^2 + \lambda \|\boldsymbol{\theta}\|_1$$

Note: $\sum_{i=1}^N a_i^2 = \mathbf{a} \cdot \mathbf{a} = \mathbf{a}^T \mathbf{a} = \|\mathbf{a}\|_2^2$

The LASSO cost function can equivalently be written as

$$J(\boldsymbol{\theta}, \lambda) = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_1$$

LASSO minimization, $p = 1$

$$\hat{\theta}_1 = \arg \min_{\theta} J(\theta, \lambda) = \sum_{i=1}^N (y_i - \theta^T x_i)^2 + \lambda \|\theta\|_p^p$$

$\begin{matrix} p=2 & \text{Ridge} \\ p=1 & \text{Lasso} \end{matrix}$

Nomenclature

 $\lambda \approx 0 \quad LS$

- $\hat{\theta}_{LS}$ denotes the least squares solution
- $\hat{\theta}_R$ denotes the least squares solution with ℓ_2 regularization (Ridge regression)
- $\hat{\theta}_1$ denotes the least squares solution with ℓ_1 regularization (LASSO)

Input–output of the weights when $X^T X = I$

The weight estimates when $X^T X = I$

$$\hat{\theta}_{LS} = X^T y$$

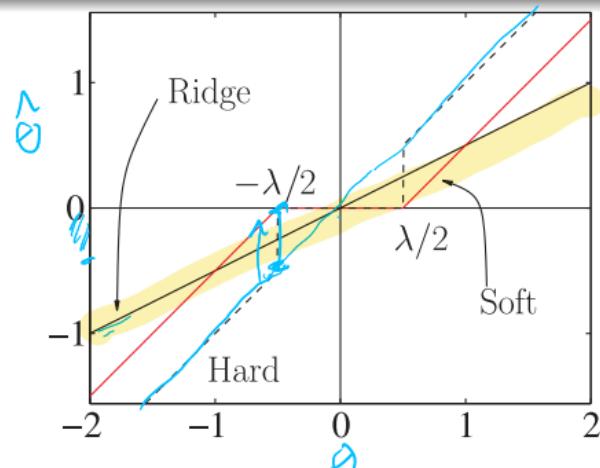
$$\hat{\theta}_R = \frac{1}{1 + \lambda} \hat{\theta}_{LS}$$

$$\hat{\theta}_{1,i} = \text{sgn}(\hat{\theta}_{LS,i}) \left(|\hat{\theta}_{LS,i}| - \frac{\lambda}{2} \right)_+ \quad i = 1, 2, \dots, l$$

$$\Theta = (X^T X)^{-1} X^T y$$

$$\Theta = (X^T X + \lambda I)^{-1} X^T y$$

ex 6.2.2



An example

Assume $\lambda = 1$ and $X^T X = I$, then given a solution for $\hat{\theta}_{LS}$, compute the weights for the Ridge, Soft, and Hard thresholds. We have

$$\hat{\theta}_{LS} = [0.4, 0.6]^T$$

$$\hat{\theta}_R = \frac{1}{1 + \lambda} \hat{\theta}_{LS} = \frac{1}{2} [0.4 \ 0.6]^T = [0.2 \ 0.3]^T$$

$$\hat{\theta}_{1,i} = \text{sgn}(\hat{\theta}_{LS,i}) \left(|\hat{\theta}_{LS,i}| - \frac{\lambda}{2} \right)_+ \quad i = 1, 2, \dots, l$$

We get ...

Compute on your own

$$\begin{aligned}\hat{\theta}_{1,1} &= \text{sgn}(0.4) \cdot (|0.4| - \frac{1}{2})_+ \\ &= 1 \cdot (0.4 - 0.5)_+ = (0 - 0.1)_+ = 0\end{aligned}$$

$$\hat{\theta}_{1,2} = 0.1$$

LASSO minimization

$$\hat{\theta}_1 = \arg \min_{\theta} J(\theta, \lambda) = (\mathbf{y} - X\theta)^T (\mathbf{y} - X\theta) + \lambda \|\theta\|_1$$

The minimization problem can equivalently be written as

$$\hat{\theta}_1 = \arg \min_{\theta} (\mathbf{y} - X\theta)^T (\mathbf{y} - X\theta)$$

Subject to : $\|\theta\|_1 \leq \rho$

For a specific choice of λ and ρ .

To see this, use Lagrangian multipliers (appendix C):

minimize : $J(\theta)$

subject to : $f(\theta) \geq 0$

Lagrangian : $L(\theta, \lambda) = J(\theta) - \lambda f(\theta)$

The LASSO solution

Take the derivative of the cost function, and put to zero, solve for θ : $\frac{d}{d\theta} J(\theta, \lambda) = 0$

Problem: the ℓ_1 norm is not differentiable.

Solution: use the subgradient method (sec 8.10).

Subgradient Algorithm

$$\theta^{(i)} = \theta^{(i-1)} - \mu_i J'(\theta^{(i-1)}) \quad \text{GD}$$

$J'(\cdot)$ denotes any subgradient of $J(\cdot)$.

Converges if the stepsize μ_i diminishes over time. Additionally

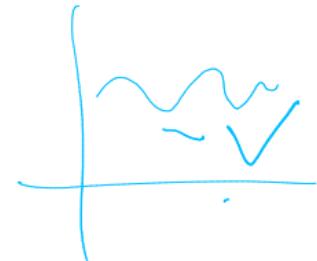
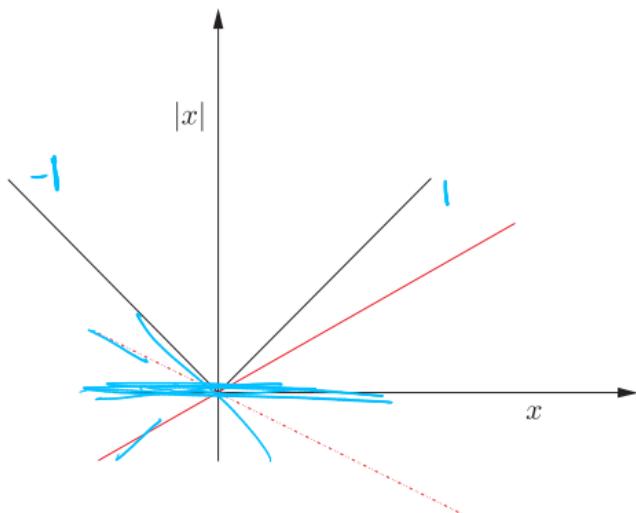
Condition for a minimizer

Given a convex function f , the zero vector must belong to the subgradient at x_* , then x_* is a minimizer, i.e.

$$0 \in \partial f(x_*)$$

$\partial f(x_*)$ is the subdifferential of $f(x)$ at x_* (the set of all subgradients).

$$\partial f(x) = \begin{cases} \text{sgn}(x), & \text{if } x \neq 0 \\ [-1, 1], & \text{if } x = 0 \end{cases}$$



Can formally be derived using definition of subdifferentials (example 8.4).

LASSO

Simplified analysis of weights

Assume $X^T X = I$, then

$$\hat{\theta}_{LS} = (X^T X)^{-1} X^T \mathbf{y} = X^T \mathbf{y}$$

$$\hat{\theta}_R = (X^T X + \lambda I)^{-1} X^T \mathbf{y} = \frac{1}{1 + \lambda} \hat{\theta}_{LS}$$

6.2

For ℓ_1 , we get

$$J(\boldsymbol{\theta}, \lambda) = (\mathbf{y} - X\boldsymbol{\theta})^T (\mathbf{y} - X\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_1$$
$$\partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, \lambda) = -2X^T \mathbf{y} + 2X^T X\boldsymbol{\theta} + \lambda \partial \|\boldsymbol{\theta}\|_1$$

Which lead to

$$0 \in -\theta_{LS,i} + \hat{\theta}_{i,1} + \frac{\lambda}{2} \partial |\hat{\theta}_{i,1}|$$

Which again leads to (which you will derive in the exercise)

$$\hat{\theta}_{1,i} = \text{sgn}(\hat{\theta}_{LS,i}) \left(|\hat{\theta}_{LS,i}| - \frac{\lambda}{2} \right)_+ \quad i = 1, 2, \dots, l$$

When is ℓ_1 unique? Example of recovery I

- Consider an unknown system.
- The system only has two parameters, and you would like to recover those parameters.
- You know the system is on the form $f(x_1, x_2) = x_1\theta_1 + x_2\theta_2$.
- You can only measure once, that is, you can only select one pair of (x_1, x_2) .
- To simulate this environment, we create the θ 's for the unknown system, and select those as $\hookrightarrow (\theta_1, \theta_2) = (0, 1)$ (arbitrary choice).
- We will now see what happens if we select 3 possible measurements, $x_a = (1/2, 1)$, $x_b = (1, 1)$, $x_c = (2, 1)$.
- These three measurements will all give a response of $f(x_1, x_2) = 1$ for the unknown system.

When is ℓ_1 unique? Example of recovery II

True $\theta = (0, 1)$.

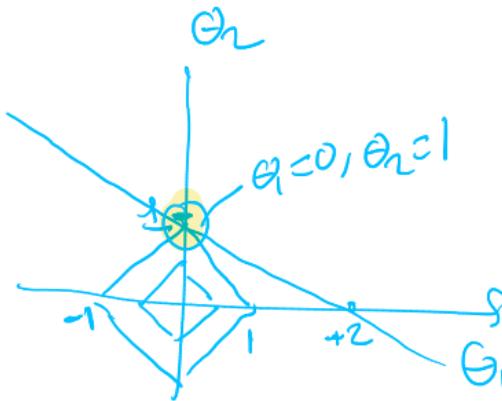
Test measurement: $x_a = (0.5, 1)$

System: $f(x) = x_1\theta_1 + x_2\theta_2$

$$= 0.5 \cdot 0 + 1 \cdot 1 = 1$$

$$y = x_1\theta_1 + x_2\theta_2$$

$$1 = 0.5 \cdot \theta_1 + 1 \cdot \theta_2$$

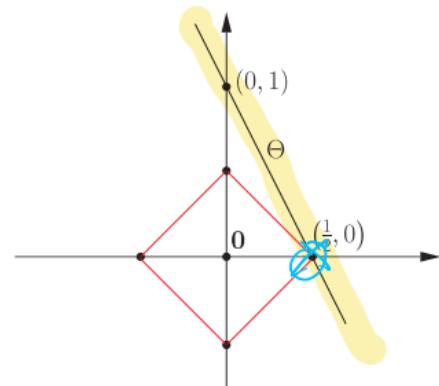
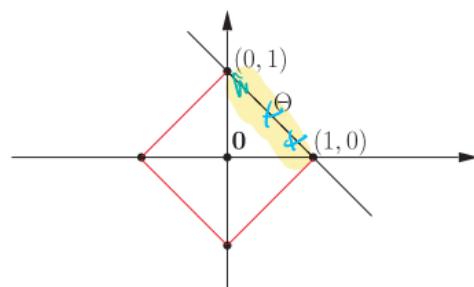
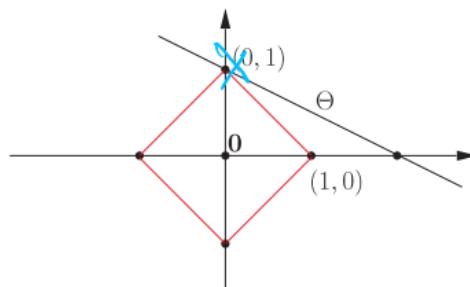


When is ℓ_1 unique? Example of recovery II

True $\theta = (0, 1)$.

Test measurements: $x_a = (1/2, 1)$, $x_b = (1, 1)$, $x_c = (2, 1)$ (all result in $f(x) = 1$).

Let us draw possible solutions to the linear system, $f(x) = x_1\theta_1 + x_2\theta_2 = 1$, and select the solution with the lowest ℓ_1 norm.



Compressed sensing explores the construction of X , and it turns out that a good X is a random X (sec 9.6–9.8).

- In the pursuit of sparse solution, we arrive at using the ℓ_1 norm as regularizer (LASSO) as the computationally most efficient norm.
 - The norm is convex.
 - No closed form solution, but solved with subgradients.
 - Has a soft thresholding operation, in the special case $X^T X = I$, sets weights to zero once they are numerically smaller than $\lambda/2$.
 - Almost always have a unique solution.
- The ℓ_0 "norm" leads to the sparsest solution, but is not convex.
- Under special circumstances, the ℓ_1 regularization will find the sparsest solution.
- We only have ONE solution (so far) for the LASSO when $X^T X = I$. Algorithms will be looked at next week.

Material: ML 10.1, 10.2–10.2.1 (until p.476), 10.2.2 (until p.482), 10.5–10.6.

- Estimating the LASSO solution:
 - LARS and Matching Pursuit algorithms
- Towards dictionary learning
 - Sparse analysis models
 - Time-frequency analysis

02471 Machine Learning for Signal Processing

Sparse analysis models and time-frequency analysis

Tommy Sonne Alstrøm

Cognitive Systems section

$$f(x+\Delta x) = \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x)$$

$\Delta \int_a^b \Theta^{\sqrt{17}} + \Omega \delta e^{i\pi} = -1$

$\Sigma!$

∞ χ^2 \gg \approx

$\{2.7182818284\}$ \circ λ

dtucompute.dtu.dk

Outline

- Course admin
- Last week review
- Sparsity promoting algorithms
- Signal representation
- Time-frequency analysis
 - The short-time Fourier transform (STFT)
 - Wavelets
- Next Week

Material: ML 10.1, 10.2–10.2.1 (until p.476), 10.2.2 (until p.482), 10.5–10.6.

Feedback

- Feedback from you is a critical component for improving both the course and my teaching.
- Feedback: most significant items to improve (I got a lot more than written here):
 -
 -
 -
- Feedback group (talk after the lecture, or send me a brief text on teams/mail with your comments):
 - Yu, Yemu
 - Carstensen, Aske Schultz
 - Ehlers, August Christian Nyeland
 - Piekarska, Barbara Nina
- Type of feedback
 - Mention one thing that worked?
 - Mention one thing should be improved (both in current lecture and last weeks exercise)?
 - Mention one thing you would change if you gave the lecture.

Last week review

What is sparsity-aware learning

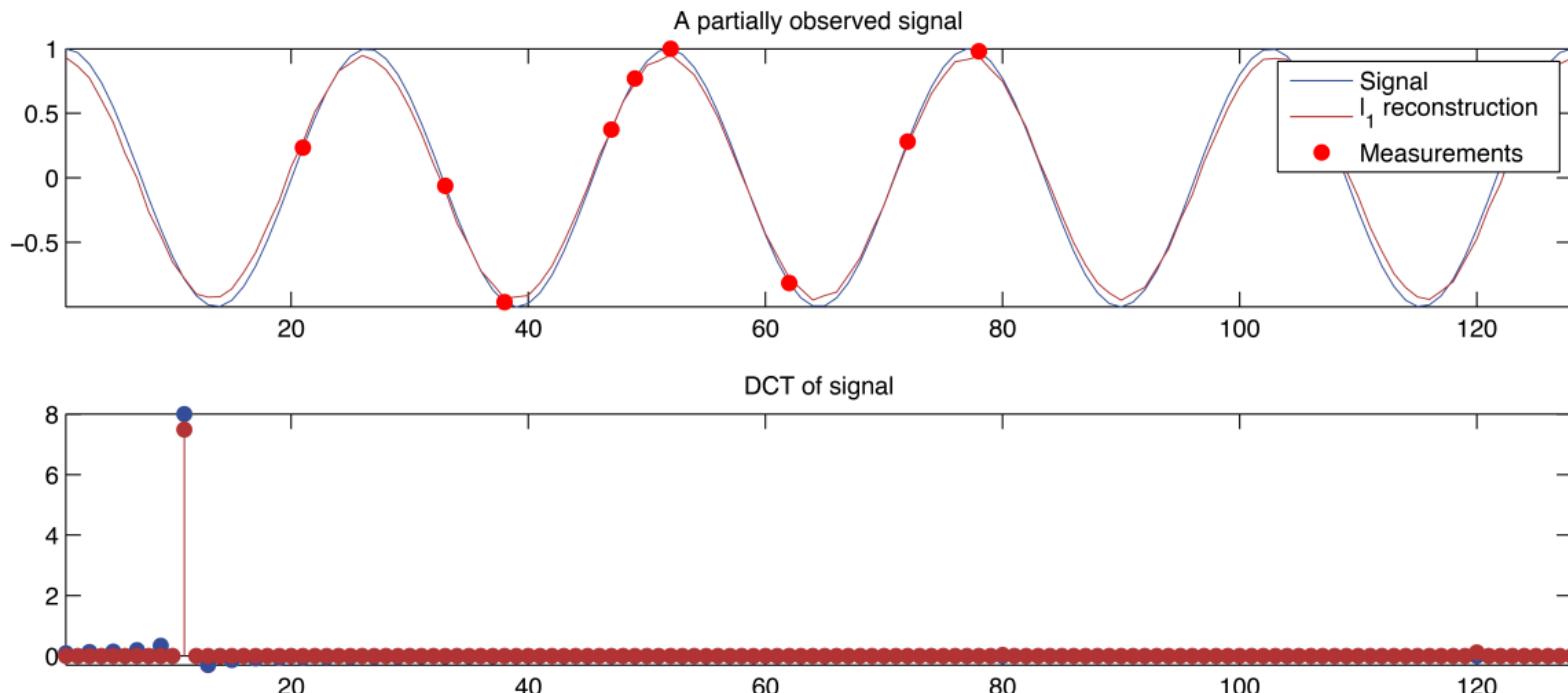
In a number of practical problems, it is known that either the underlying model is sparse or it is sparse in a transform domain, e.g., in the Fourier transform domain.

By **sparse models** is meant that most of the unknown coordinates in the model are zero.

Leads to **compressed sensing**, where the goal is to directly acquire **as few samples as possible** that encode the minimum information, which is needed to obtain a **compressed signal representation**.

The solution is often found using ℓ_1 norm regularization, and the corresponding solution is called LASSO (least absolute shrinkage and selection operator).

Signal reconstruction using ℓ_1



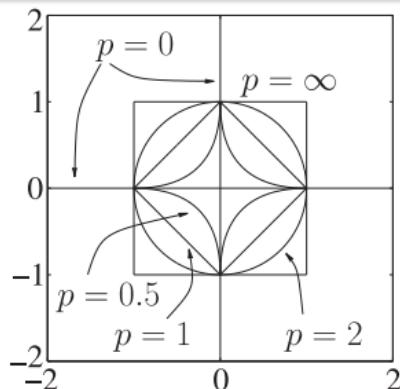
The ℓ_p norm

$$\|\boldsymbol{\theta}\|_p := \left(\sum_{i=1}^l |\theta_i|^p \right)^{1/p}, \quad 0 < p < \infty$$

$$\|\boldsymbol{\theta}\|_\infty := \arg \max_i |\theta_i| \quad (\text{max element})$$

$$\|\boldsymbol{\theta}\|_0 := |\{i \mid \theta_i \neq 0, i = 1, \dots, l\}| \quad (\text{number of nonzeros})$$

l is the size of vector $\boldsymbol{\theta}$.



Notation remarks:

$|x|$ is the numerical value when $x \in \mathbb{R}$.

$|\mathcal{X}|$ is the cardinality (size) of the set \mathcal{X} .

LASSO minimization problem

LASSO minimization

$$\hat{\boldsymbol{\theta}}_1 = \arg \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, \lambda) = \sum_{i=0}^n (y_i - \boldsymbol{\theta}^T \mathbf{x}_i)^2 + \lambda \|\boldsymbol{\theta}\|_1$$

Nomenclature

- $\hat{\boldsymbol{\theta}}_{LS}$ denotes the least squares solution
- $\hat{\boldsymbol{\theta}}_R$ denotes the least squares solution with ℓ_2 regularization (Ridge regression)
- $\hat{\boldsymbol{\theta}}_1$ denotes the least squares solution with ℓ_1 regularization (LASSO)

When is ℓ_1 unique? Example of recovery I

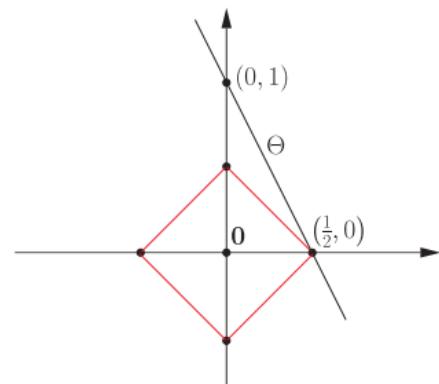
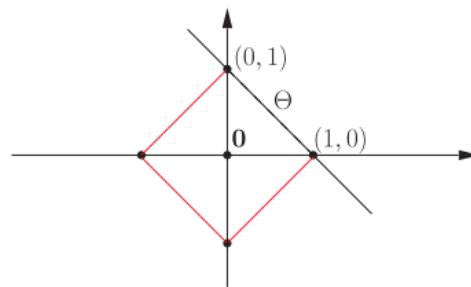
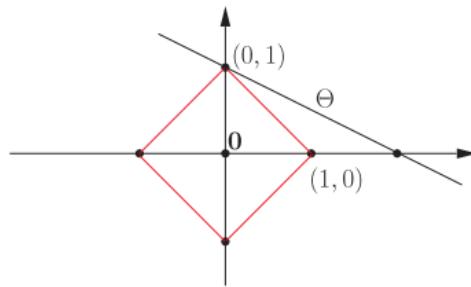
- Consider an unknown system.
- The system only has two parameters, and you would like to recover those parameters.
- You know the system is on the form $f(x_1, x_2) = \theta_1 x_1 + \theta_2 x_2$.
- You can only **measure once**, that is, you can only select one pair of (x_1, x_2) .
- To simulate this environment, we create the θ 's for the unknown system, and select those as $(\theta_1, \theta_2) = (0, 1)$ (arbitrary choice).
- We will now see what happens if we select 3 possible measurements, $x_a = (1/2, 1)$, $x_b = (1, 1)$, $x_c = (2, 1)$.
- These three measurements will all give a response of $f(x_1, x_2) = 1$ for the unknown system.

When is ℓ_1 unique? Example of recovery II

True $\theta = (0, 1)$.

Test measurements: $x_a = (1/2, 1)$, $x_b = (1, 1)T$, $x_c = (2, 1)$ (all result in $f(x) = 1$).

Let us draw possible solutions to the linear system, $f(x) = \theta_1 x_1 + \theta_2 x_2 = 1$, and select the solution with the lowest ℓ_1 norm.



Sparsity-aware learning have many cool applications, in statistics, signal processing, machine learning.

- In the pursuit of sparse solution, we arrived at using the ℓ_1 norm as the computationally most efficient norm.
 - The norm is convex.
 - The ℓ_0 "norm" leads to the sparsest solution, but is not convex.
- Under special circumstances, the ℓ_1 regularization will find the sparsest solution.
- LASSO solves the ℓ_1 norm regularization problem.

Sparsity promoting algorithms

The OMP algorithm – algorithm 10.1 in the book

- **Initialize**

- $\theta^{(0)} = \mathbf{0} \in \mathbb{R}^l$.
- $S^{(0)} = \emptyset$.
- $e^{(0)} = y$.

- **For** $i = 1, \dots, k$ **Do**

- Select the column in X that forms the smallest angle with the error.
- Update the indices of active vectors, $S^{(i)}$.
- Update the parameter vector $\theta^{(i)}$ using least squares using the columns in X indexed by $S^{(i)}$.
- Update the error vector.

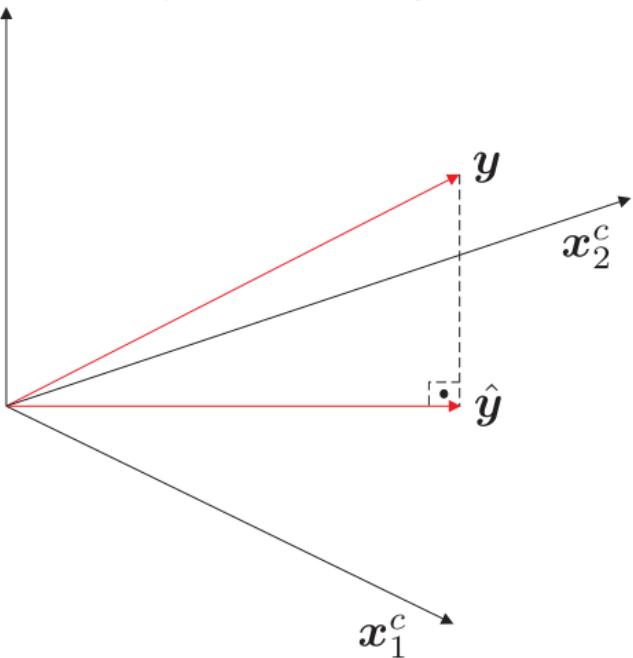
- **End For**

Parameters:

k is the number of non-zero components (must be smaller than the number of observations)

Sparsity promoting algorithms
Why does the OMP work?

Remember, the inner product actually carries out a projection



The naive IST formula (10.3)–(10.7) in the book (estimates the LASSO solution)

- Initialize
 - $\theta^{(0)} = \mathbf{0} \in \mathbb{R}^l$.
 - Select the value of μ
 - Select the value of λ
- For $i = 1, \dots$ Do
 - $e^{(i-1)} = \mathbf{y} - X\theta^{(i-1)}$
 - $\tilde{\theta} = \theta^{(i-1)} + \mu X^T e^{(i-1)}$
 - $\theta^{(i)} = \text{sign}(\tilde{\theta}) \max(|\tilde{\theta}| - \lambda\mu, 0)$
- End For

Parameters:

μ is still the step size, but also affects the shrinkage.

λ is the regularization parameter.

If we have the cost function,

$$J(\boldsymbol{\theta}) = \frac{1}{2} \|\mathbf{y} - X\boldsymbol{\theta}\|_2^2$$

we know from earlier weeks that the gradient descent update is

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} + \mu X^T \mathbf{e}^{(i-1)}$$

However, it turns out that this is also the solution to the following optimization problem

$$\boldsymbol{\theta}^{(i)} = \arg \min_{\boldsymbol{\theta} \in \mathbb{R}^l} \left\{ J\left(\boldsymbol{\theta}^{(i-1)}\right) + \left(\boldsymbol{\theta} - \boldsymbol{\theta}^{(i-1)}\right)^T \frac{\partial J\left(\boldsymbol{\theta}^{(i-1)}\right)}{\partial \boldsymbol{\theta}} + \frac{1}{2\mu} \|\boldsymbol{\theta} - \boldsymbol{\theta}^{(i-1)}\|_2^2 \right\}$$

This is shown by taking the derivative w.r.t $\boldsymbol{\theta}$ and set to zero.

Sparsity promoting algorithms Route to IST step 2



From previous slides:

$$\boldsymbol{\theta}^{(i)} = \arg \min_{\boldsymbol{\theta} \in \mathbb{R}^l} \left\{ J(\boldsymbol{\theta}^{(i-1)}) + (\boldsymbol{\theta} - \boldsymbol{\theta}^{(i-1)})^T \frac{\partial J(\boldsymbol{\theta}^{(i-1)})}{\partial \boldsymbol{\theta}} + \frac{1}{2\mu} \|\boldsymbol{\theta} - \boldsymbol{\theta}^{(i-1)}\|_2^2 \right\}$$

LASSO minimizes

$$\boldsymbol{\theta}^{(i)} = \arg \min_{\boldsymbol{\theta} \in \mathbb{R}^l} \left\{ \frac{1}{2} \|\mathbf{y} - X\boldsymbol{\theta}\|_2^2 + \lambda \|\boldsymbol{\theta}\|_1 \right\}$$

Combining these gives

$$\boldsymbol{\theta}^{(i)} = \arg \min_{\boldsymbol{\theta} \in \mathbb{R}^l} \left\{ J(\boldsymbol{\theta}^{(i-1)}) + (\boldsymbol{\theta} - \boldsymbol{\theta}^{(i-1)})^T \frac{\partial J(\boldsymbol{\theta}^{(i-1)})}{\partial \boldsymbol{\theta}} + \frac{1}{2\mu} \|\boldsymbol{\theta} - \boldsymbol{\theta}^{(i-1)}\|_2^2 + \lambda \|\boldsymbol{\theta}\|_1 \right\}$$

Solving this minimization problem (taking the derivative, set to 0, and solve for $\boldsymbol{\theta}$), will yield the IST update (and avoids the problem we had last week where we assumed $X^T X = I$).

- We presented two algorithms,
- OMP is a heuristic greedy approach, that simply builds up a k -sparse solution vector in k steps. Hence, it solves the ℓ_0 solution.
 - There is no guarantee that OMP finds the optimal ℓ_0 solution.
 - LARS and LARS-LASSO are extensions of OMP.
- IST is an iterative shrinkage/thresholding (IST) type algorithm, and estimates the ℓ_1 solution.
 - It is a naive implementation we derived, and in practice one should use e.g. FISTA.
- Under certain circumstances (theory not part of curriculum, but listed in 9.6–9.7), the ℓ_0 and ℓ_1 minimizer has the same solution.
- You will implement both in the exercise today.

Signal representation

Analysis

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j\frac{2\pi}{N}kn}$$

Synthesis

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)e^{j\frac{2\pi}{N}kn}$$

k corresponds to frequency kF_s/N

Signal representation

Linear signal representations

A linear signal representation model can be thought of as

Linear signal representations

$$\begin{aligned}\tilde{s} &= \Phi^H s && \text{analysis} \\ s &= \Phi \tilde{s} && \text{synthetic}\end{aligned}$$

- s is the vector of raw samples.
- \tilde{s} is the transformed vector.
- Φ is the unitary transformation matrix, $\Phi\Phi^H = I$.

There are many choices of matrices, we can choose Φ^H as a matrix of Fourier coefficients (complex matrix).

Other choices could be the DCT matrix (which is real), or wavelet matrix.

$$\text{DCT-II (dct)} : X_k = \sum_{n=0}^{N-1} x_n \cos \left[\frac{\pi}{N} \left(n + \frac{1}{2} \right) k \right] \quad k = 0, \dots, N-1$$

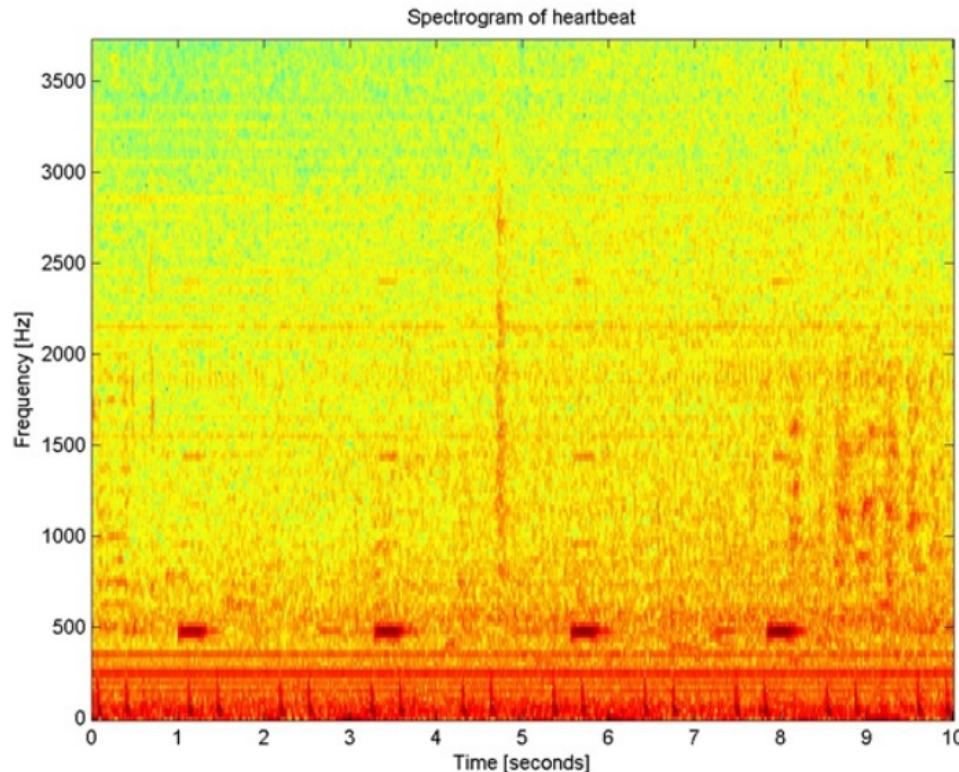
$$\text{DCT-III (idct)} : X_k = \frac{1}{2} x_0 + \sum_{n=1}^{N-1} x_n \cos \left[\frac{\pi}{N} n \left(k + \frac{1}{2} \right) \right] \quad k = 0, \dots, N-1$$

These transforms can be organized as real matrices, such that Φ^H corresponds to DCT-II and Φ corresponds to DCT-III.

- We operate with linear signal representations.
- The Fourier transform, the DCT transform (and more) can be recast as linear projections.
- The model is closely related to dictionary learning, where Φ is estimated from data instead of predefined.

Time-frequency analysis

The spectrogram of a heartbeat



DFT

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j \frac{2\pi}{N} kn}$$

STFT

$$X(n, k) = \sum_{m=-\infty}^{\infty} \underbrace{x(m)w(m-n)}_{\text{new signal: } \hat{x}(n)} e^{-j \frac{2\pi}{N} kn}$$

Without loss of generality, let us define

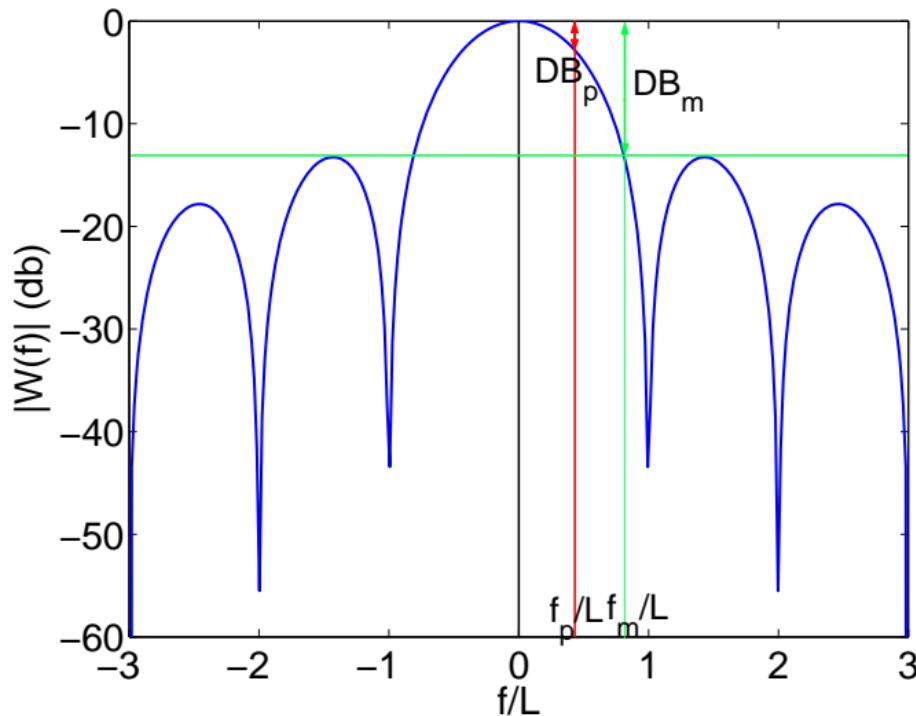
$$\hat{x}(n) = x(n) \cdot w(n)$$

From properties of Fourier transform, we know multiplication in the time domain is convolution in the frequency domain

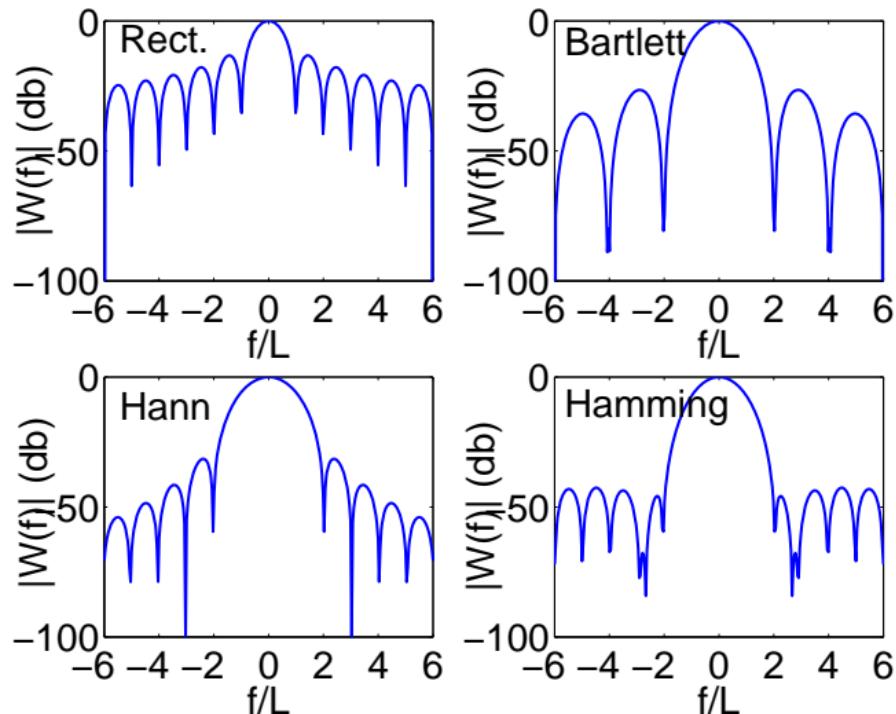
$$\hat{X}(f) = X(f) * W(f) = \int_{-1/2}^{1/2} X(s)W(f - s)ds$$

How would the ideal $W(f)$ look like?

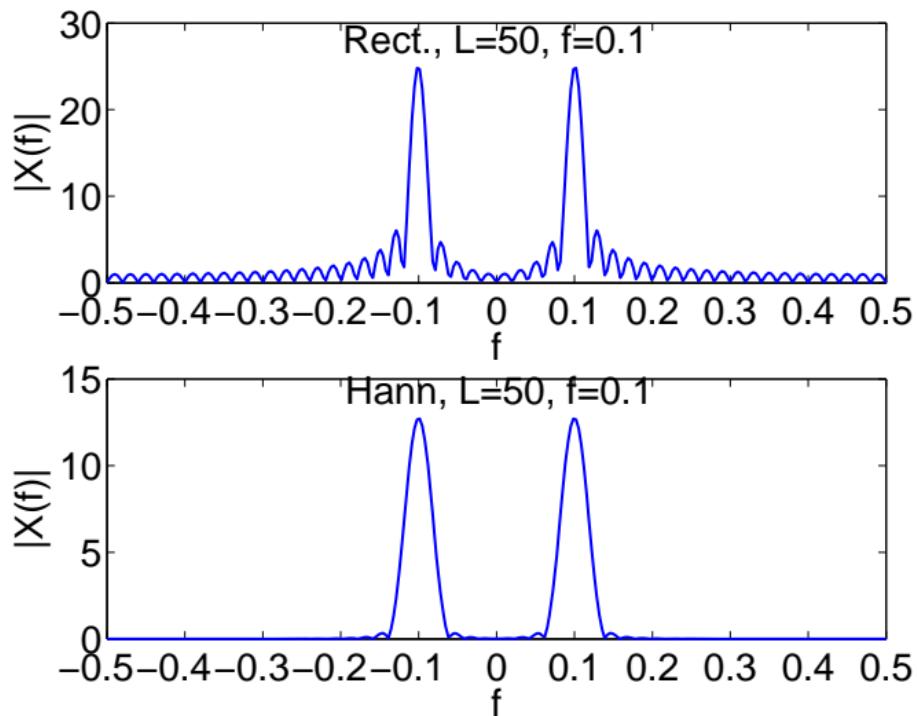
On window functions: resolution and leakage

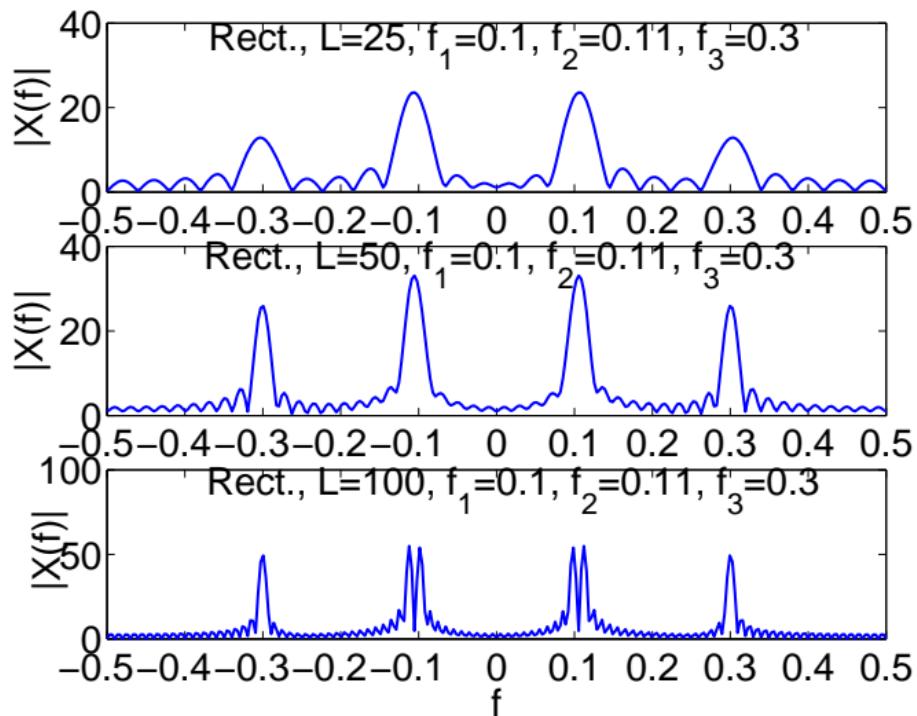


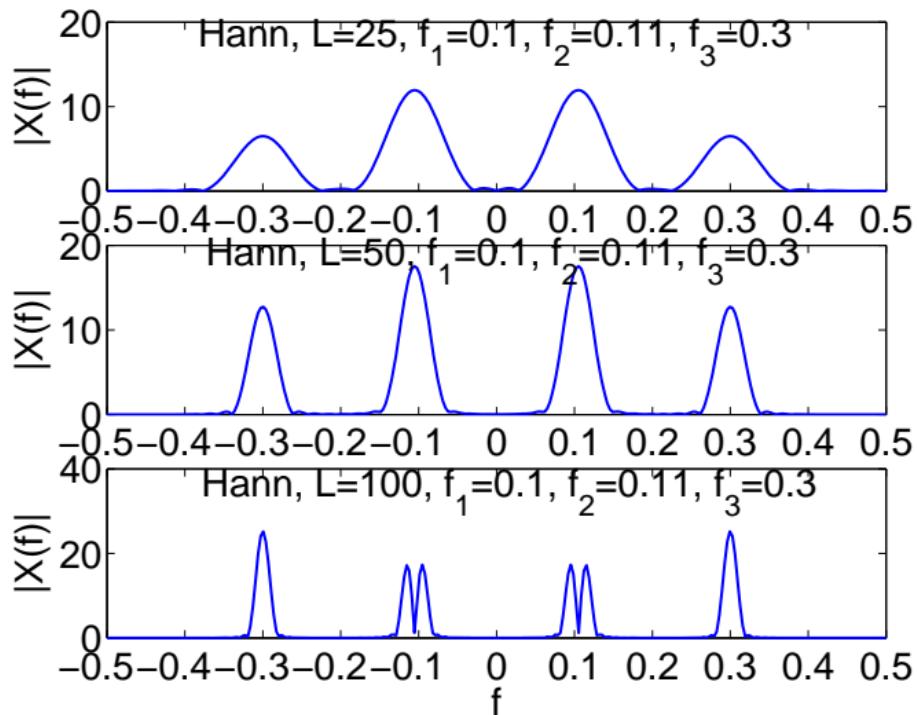
Spectral shape of windows



Example: sinusoid signal



Example: rectangular on 3 sinusoid

Example: Hanning on 3 sinusoids

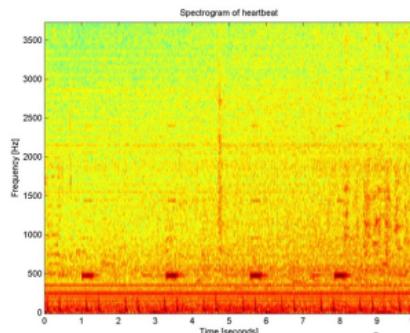
STFT

$$X(n, k) = \sum_{m=-\infty}^{\infty} x(m)w(m-n)e^{-j\frac{2\pi}{N}kn}$$

The spectrogram

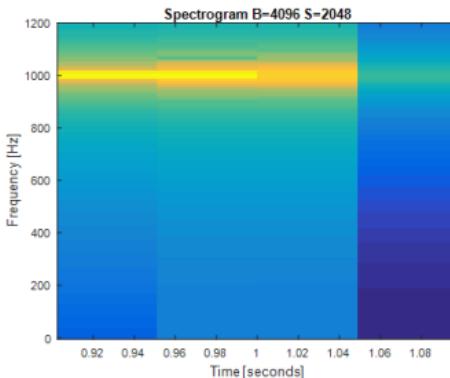
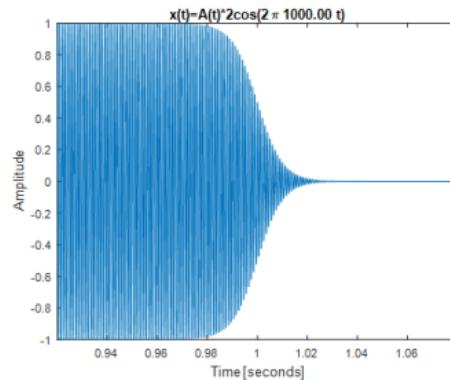
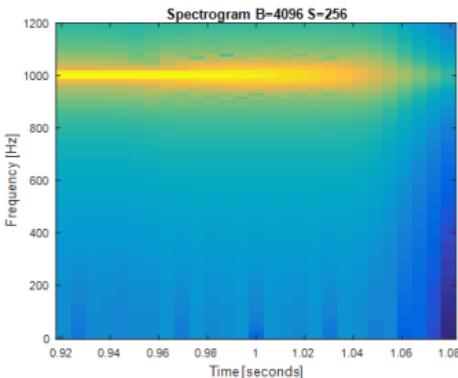
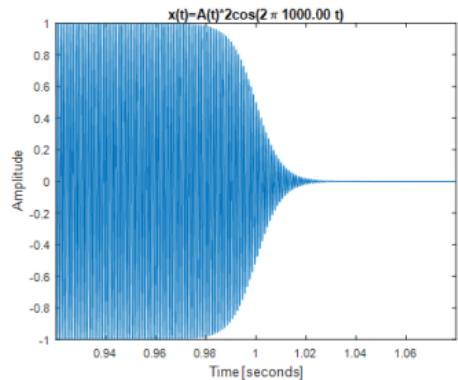
The magnitude spectrum computed using STFT, ie $|X(n, k)|$.

Two important parameters; the **block size** B (window size), and the **hop size** S (stride, or window overlap).



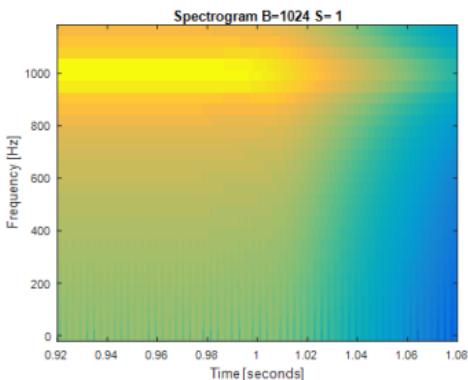
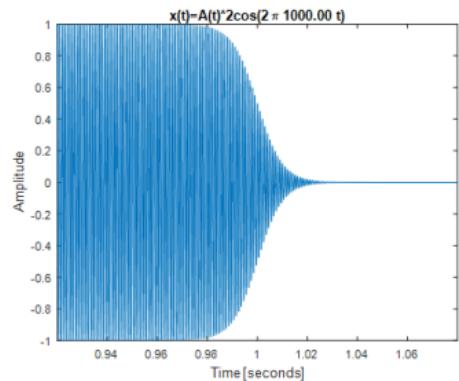
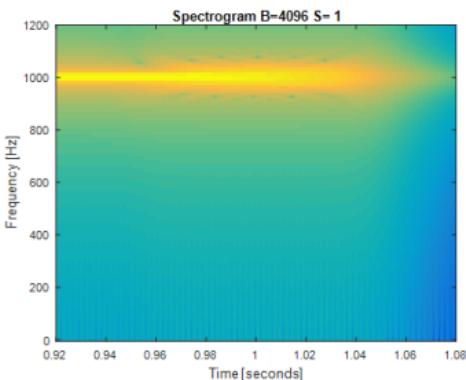
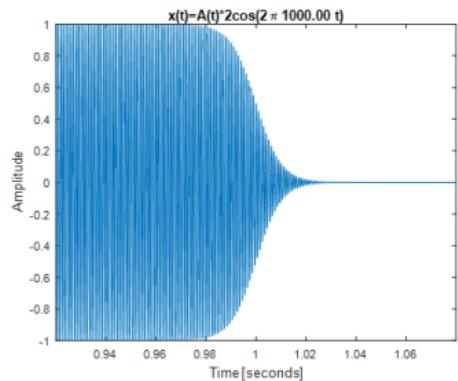
Time-frequency analysis

STFT of a cosine



Time-frequency analysis

STFT of a cosine



Considerations for the STFT

- Time overlap of windows: smooth transitions little perceptual artifacts, however, should be done so "perfect" that synthesis is possible
- The shape and length of the window: slow variation of amplitude within a window and little leakage
- The number of points N (frequency bins) in the DFT: only one partial in each band

Continuous-time Wavelet transform

$$X(s, \tau) = \int_t x(t) \phi_{s,\tau}^*(t) dt$$
$$x(t) = \int_{s,\tau} X(s, \tau) \phi_{s,\tau}(t) ds d\tau$$

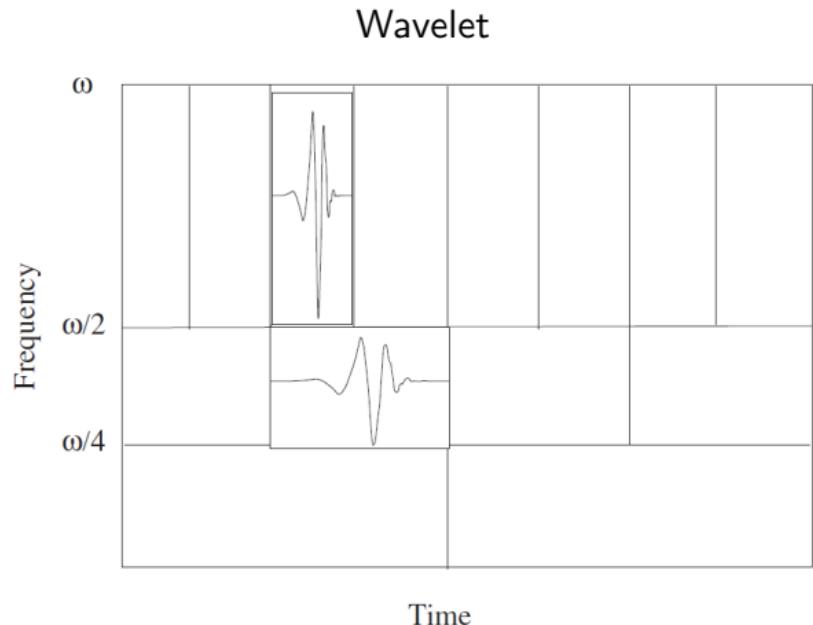
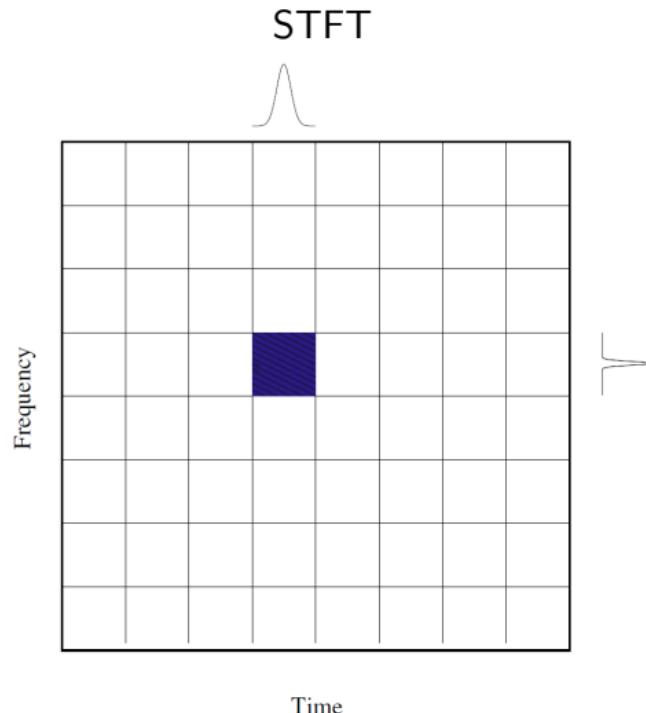
Let us compare these equations to the Continuous time Fourier transform

Continuous-time Fourier transform

$$X(F) = \int_t x(t) e^{-j2\pi F t} dt$$
$$x(t) = \int_F X(F) e^{j2\pi F t} dF$$

Time-frequency analysis

Wavelets vs STFT



- Two algorithms were presented, OMP and IST. And they solve ℓ_0 and ℓ_1 respectively.
- This leads to linear signal representation models.
- If the signal is not stationary, the representation models can be applied on smaller chunks of the signal. This approach is called “time-frequency analysis”
 - This can be e.g. the Fourier transform, which lead to the short-time Fourier transform (STFT).
 - Other window choices leads to the Gabor transform.
 - Other base function choices leads to wavelets.
 - The TF approach can be used to extract features from signals (the time series gets a vector-space representation), that can then be applied to a machine learning classifier.

Next week

Material: ML 2.5, 19.1–19.3, 19.5–19.7.

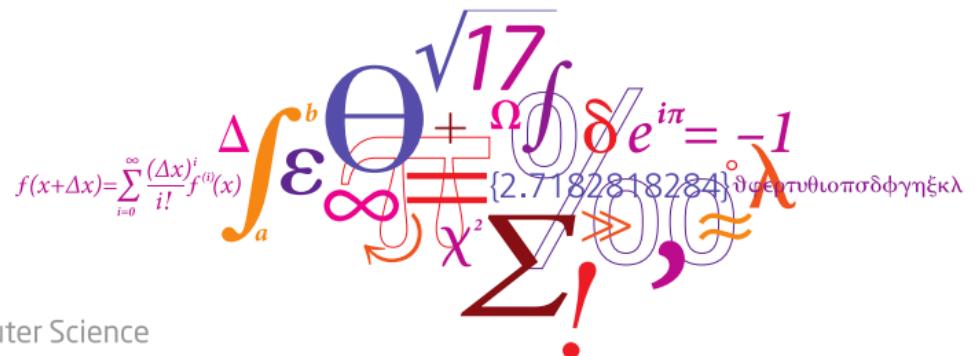
- PCA brief primer (very brief, leads to ICA).
- Independent component analysis (ICA).
- Dictionary learning (NMF, k -SVD).

02471 Machine Learning for Signal Processing

Dictionary learning and source separation

Tommy Sonne Alstrøm

Cognitive Systems section



Outline

- Last week review
- Dictionary learning – applications
- Source separation – applications
- Relation to earlier topics
- Dictionary learning
 - k -SVD
 - Non-negative Matrix Factorization (NMF)
- Independent Component Analysis (ICA)
- Next week

Material: 2.5, 19.1–19.3, 19.5–19.7

The story so far and what the future holds

- Problem set 2 re-submissions due Friday 03/11 at 23.59
- Problem set 3 is available and is due 15/12 23.59, and counts 20% towards the final grade

What you have learned so far:

- Parameter estimation [L2 regularization, biased estimation, mean squared error minimization]. L1 regularization **Todo: Bayesian parameter estimation** (next week)
- Filtering signals [Stochastic processes, correlation functions, Wiener filter, linear prediction, adaptive filtering using stochastic gradient decent (LMS, APA/NLMS), adaptive filtering using regularization (RLS)]

Sparse signal representations and dictionary learning

- Signal representations [Time frequency analysis, sparsity aware learning, **Todo: factor models**]
- Sparsity aware sensing (lasso, sparse priors), compressed sensing, **Todo: dictionary learning** [**Independent component analysis**, **Non-negative matrix factorization**, **k-SVD**]

Last week review

Linear signal representations

A linear signal representation model can be thought of as

Linear signal representations

$$\begin{array}{ll} \tilde{s} = \Phi^H s & \text{analysis} \\ s = \Phi \tilde{s} & \text{synthesis} \end{array}$$

$\tilde{s} = \Phi^H s$ $s = \Phi \tilde{s}$

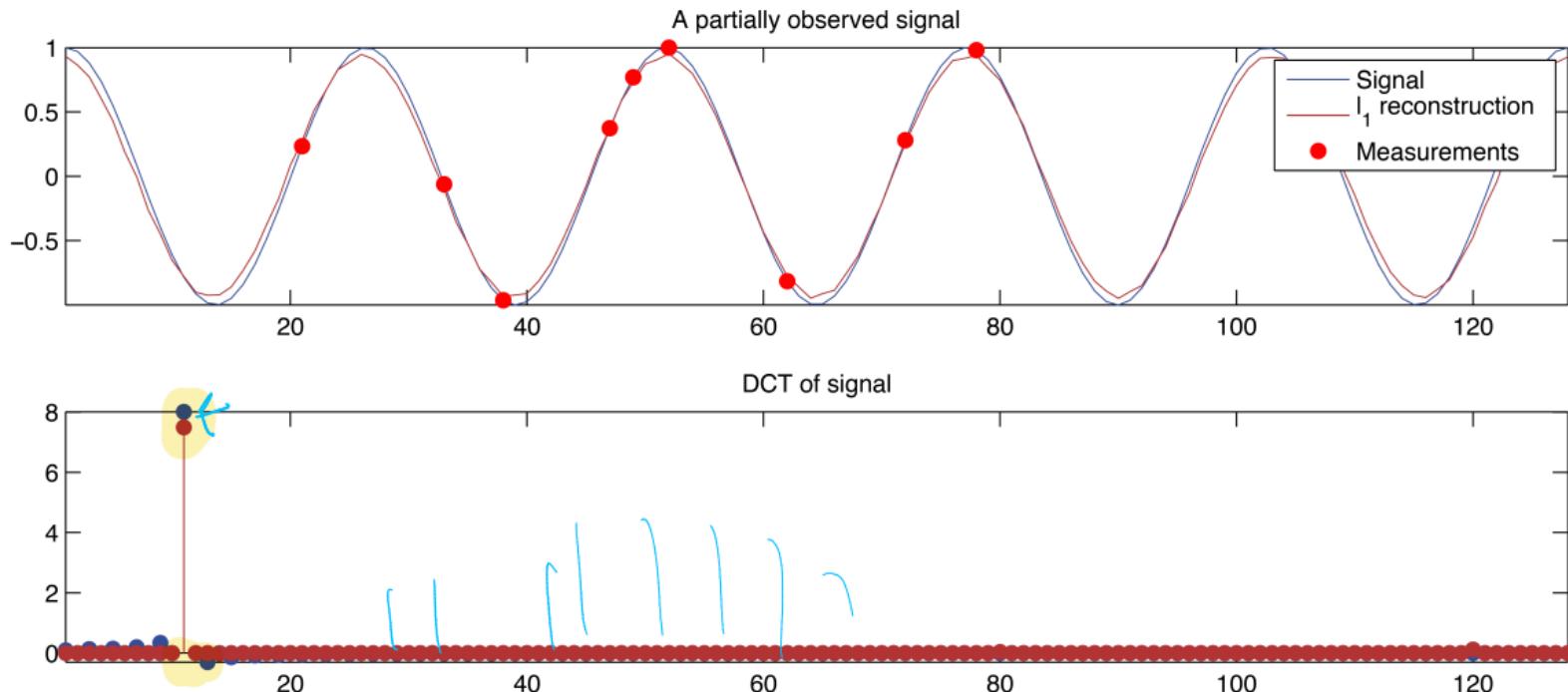
$$\begin{array}{l} \tilde{s} = F_q(s) \\ s = G_{\theta_2}(\tilde{s}) \end{array}$$

- s is the vector of raw samples.
- \tilde{s} is the transformed vector.
- Φ is the unitary transformation matrix, $\Phi \Phi^H = I$.

There are many choices of matrices, we can choose Φ^H as a matrix of fourier coefficients (complex matrix), the DCT matrix (which is real), or a wavelet matrix.

Last week review

Signal reconstruction using ℓ_1



The OMP algorithm – algorithm 10.1 in the book

• Initialize

- $\theta^{(0)} = \mathbf{0} \in \mathbb{R}^l$.
- $S^{(0)} = \emptyset$.
- $e^{(0)} = y$.

• For $i = 1, \dots, k$ Do

- Select the column in X that forms the smallest angle with the error.
- Update the indices of active vectors, $S^{(i)}$.
- Update the parameter vector $\theta^{(i)}$ using least squares using the columns in X indexed by $S^{(i)}$.
- Update the error vector.

• End For

Parameters:

k is the number of non-zero components (must be smaller than the number of observations)

Iterative Shrinkage/thresholding (IST)

The naive IST formula (10.3)–(10.7) in the book (estimates the LASSO solution)

- Initialize

- $\theta^{(0)} = \mathbf{0} \in \mathbb{R}^l$.
- Select the value of μ
- Select the value of $\lambda \sim$ Controls sparsity

- For $i = 1, \dots$ Do

- $e^{(i-1)} = \mathbf{y} - X\theta^{(i-1)}$
- $\tilde{\theta} = \theta^{(i-1)} + \mu X^T e^{(i-1)}$
- $\theta^{(i)} = \text{sign}(\tilde{\theta}) \max(|\tilde{\theta}| - \lambda\mu, 0)$

- End For

Parameters:

μ is still the step size, but also affects the shrinkage.

λ is the regularization parameter.

The spectrogram

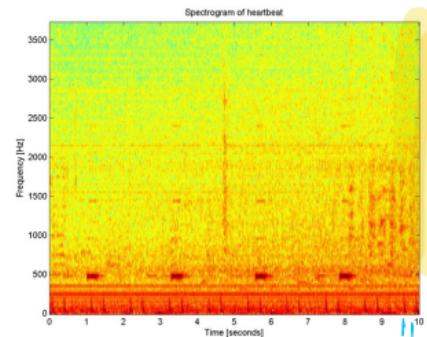
STFT

$$X(n, k) = \sum_{m=-\infty}^{\infty} x(m)w(m-n)e^{-j\frac{2\pi}{N}kn}$$

The spectrogram

The magnitude spectrum computed using STFT, ie $|X(n, k)|$.

Two important parameters; the **block size B** , and the **hop size S** .



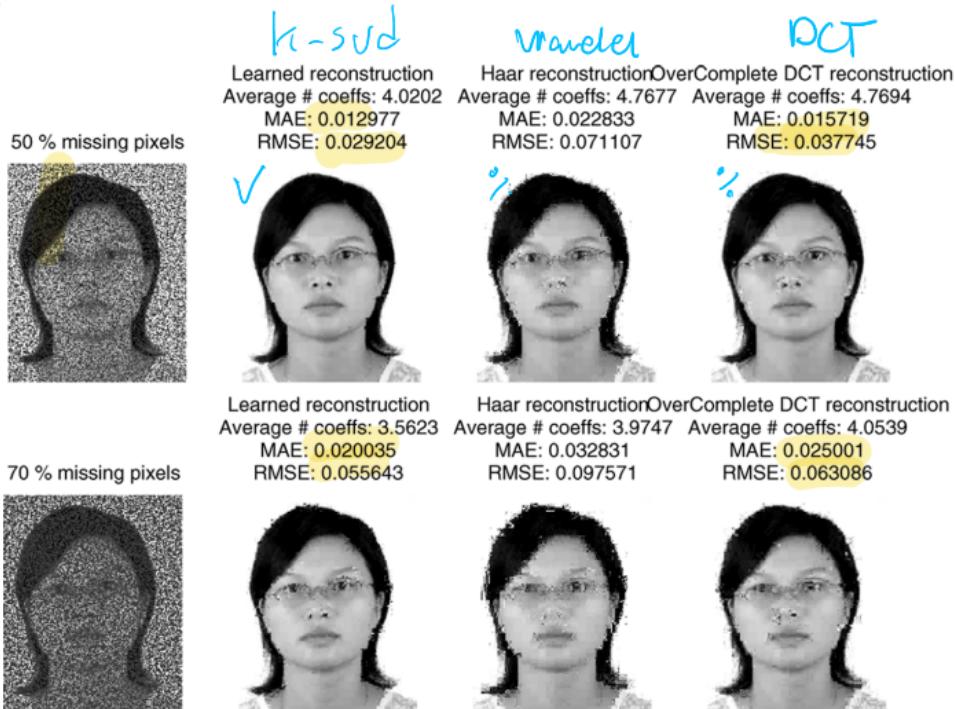
Lecture summary

- Two algorithms were presented, OMP and IST. And they solve ℓ_0 and ℓ_1 respectively.
 - OMP is used e.g. in k -SVD (today).
- This lead to linear signal representation models.
- If the signal is not stationary, the representation models can be applied on smaller chunks of the signal. This approach is called “time-frequency analysis”.
- The models are designed using domain knowledge. If we want to learn the models from data, in machine learning, we call it dictionary learning, or source separation.

Dictionary learning – applications

Dictionary learning – applications

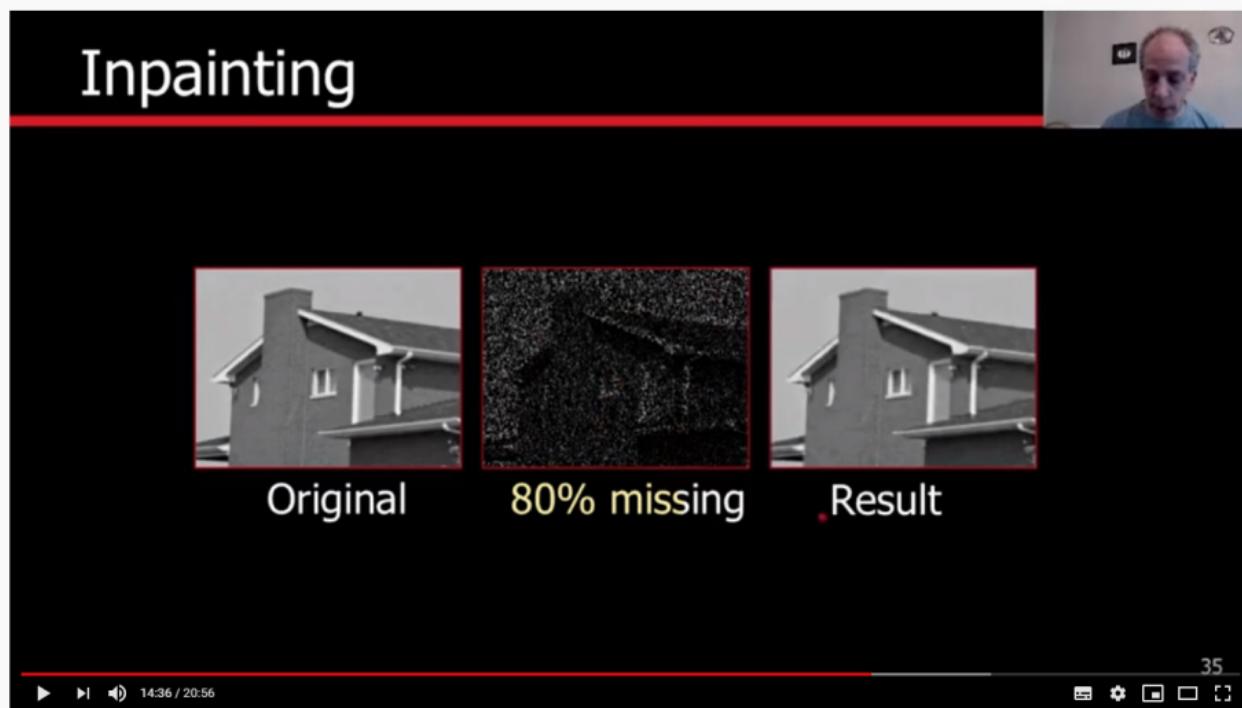
Image denoising



Michal Aharon, Michael Elad, and Alfred Bruckstein, "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation", 2006.

Dictionary learning – applications

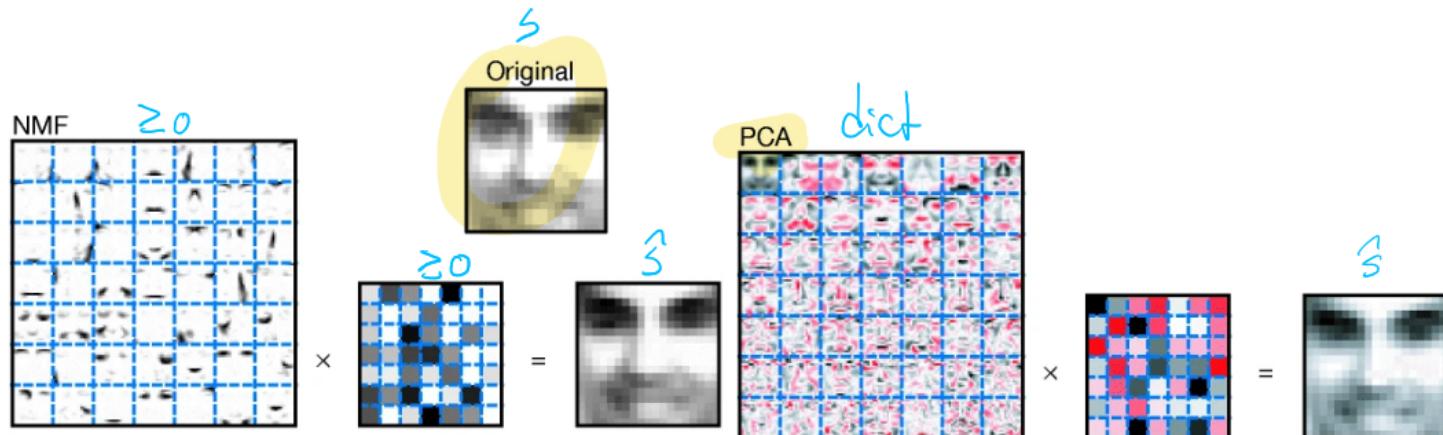
Dictionary learning – applications



<https://youtu.be/SIP1Fb4ZZ80?t=572>

Dictionary learning – applications

Image decomposition



Learning the parts of objects by non-negative matrix factorization, 1999 Daniel D. Lee and H. Sebastian Seung

Dictionary learning – applications

Spectral denoising

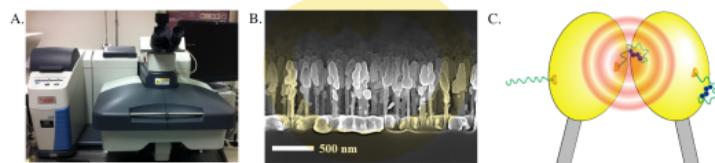


Fig. 1. A. The DXR™ Raman Microscope used to collect data. B. A side-view up of a Raman substrate depicting the nanopillars, courtesy of Kaiyu Wu, DTU. C. Illustration of the principle behind the SERS substrates. The two left pillars have molecules on them but in order to get the improved SNR the molecule needs to be captured in the hot spot as shown on the right. This is achieved by leaning the pillars through solvent evaporation.

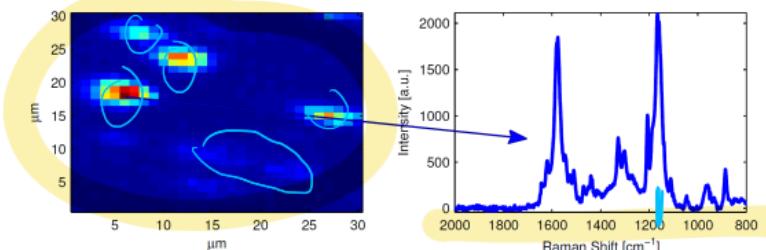
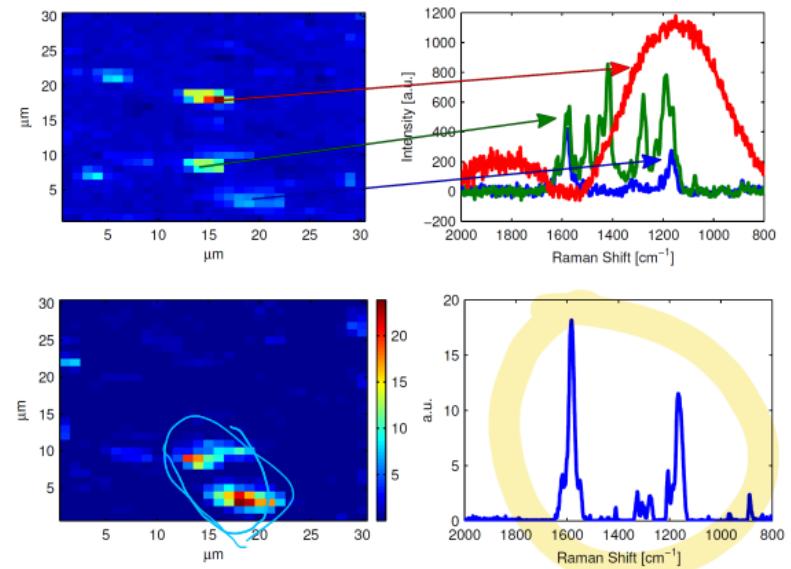


Fig. 2. Illustration of an uncontaminated SERS measurement using a Raman map at 1166 cm⁻¹ (left) and an example spectrum for Estradiol Glow (right). Hot spots containing molecules are readily identified on the Raman map as the red areas. On the spectrum the peaks at 1166 cm⁻¹ and 1580 cm⁻¹ are considered the major discriminative.

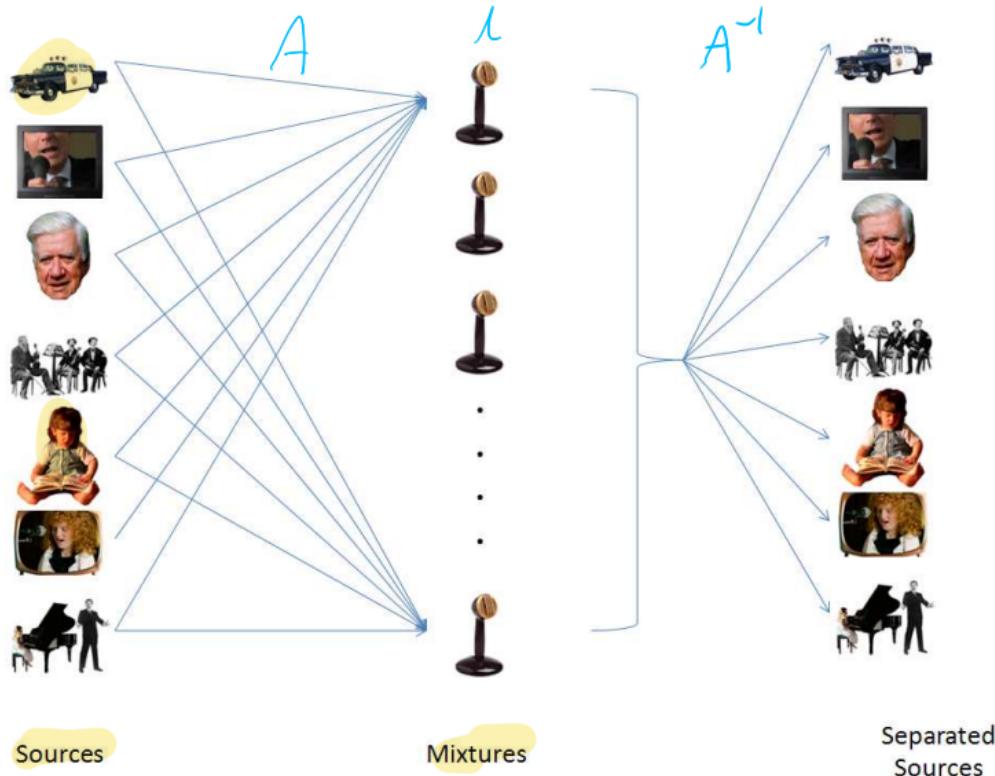


Alstrøm et al. Improving the robustness of surface enhanced Raman spectroscopy based sensors by Bayesian non-negative matrix factorization, 2014.

Source separation – applications Source separation in audio

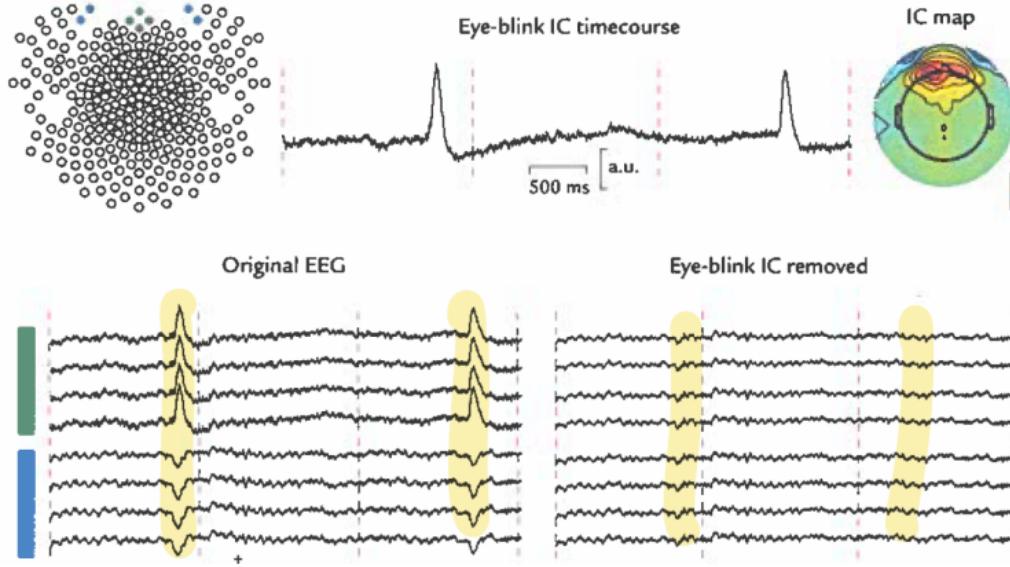
ICA

8.4
ex 8.5



Source separation – applications

Source separation in EEG



"Combining EEG and eye tracking: identification, characterization, and correction of eye movement artifacts in electroencephalographic data", 2012, Michael Plöchl, José P. Ossandón, and Peter König.

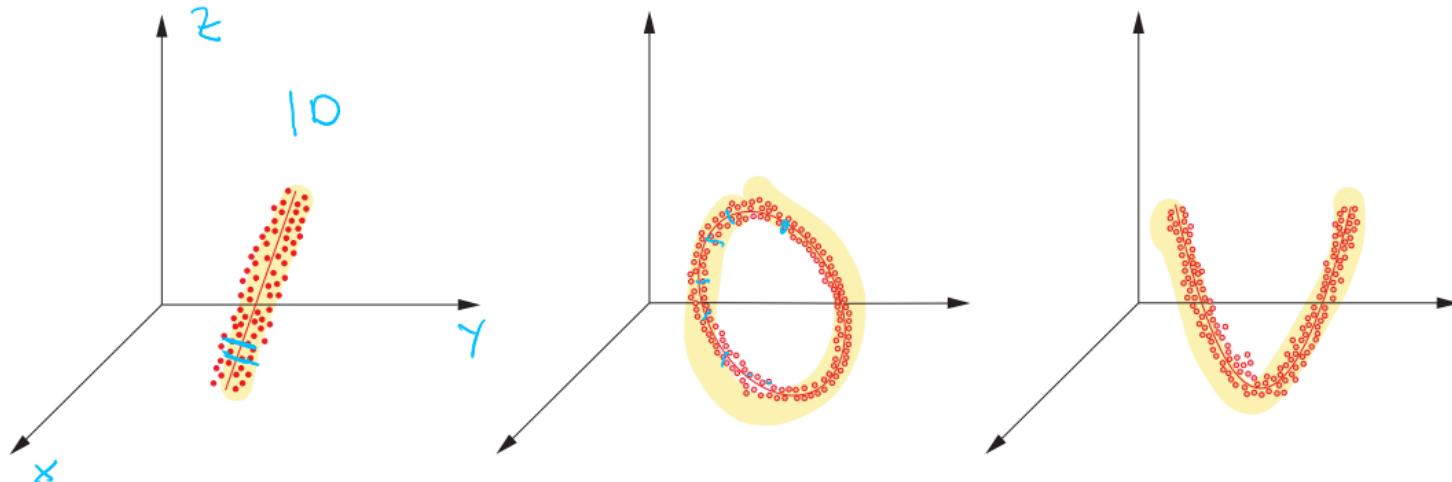
- Blind source separation and dictionary learning are closely related, it only depends on the constraints enforced on the model.
- Dictionary learning is used to learn a dictionary of the data, that can later be used to encode the data.
- Blind source separation seeks to unmix the data, because the source recovery is the task.
- Has many applications in both machine learning and signal processing.
- An active research topic.

Relation to earlier topics

Relation to earlier topics

Relation to sparsity aware–learning

We want to identify good representations for objects



A good representation for a circle

$$\mathbf{x} = [r \cos \theta, r \sin \theta]^T$$

The data is one-dimensional under this representation (intrinsic dimensionality).

The linear factor model

Factor model

$$\mathbf{x} = A\mathbf{z}$$

if we observe N measurements, we get

$X := [\mathbf{x}_1, \dots, \mathbf{x}_N]$ ($l \times n$), A ($l \times m$), $Z := [\mathbf{z}_1, \dots, \mathbf{z}_n]$ ($m \times n$):

$$X = AZ \quad \text{— needs to be estimated}$$

We have only observed X , so to estimate A and Z , we need additional constraints.

ICA, NMF, PCA and k -SVD are methods that use the above model, but impose different constraints in order to enforce the desired structure on A and Z .

Traditionally, digital signal processing seeks to design A (e.g. DCT), machine learning seeks to learn A .

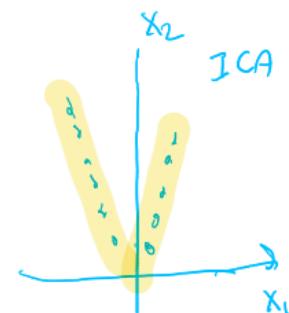
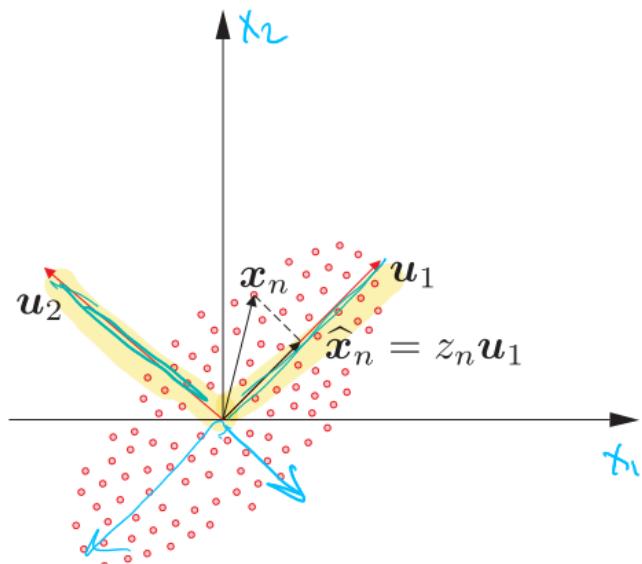
Relation to earlier topics

PCA

PCA will scale and rotate the coordinate system. We have the model

$$\mathbf{z} = U^T \mathbf{x}$$

Where U^T is the projection matrix, i.e \mathbf{x} is projected unto the space spanned by U , and \mathbf{z} is the coordinate in that new subspace.



Dictionary learning

Towards dictionary learning

The k -SVD model

$$X \approx AZ + E$$

$$X \in \mathbb{R}^{l \times n}$$

$$A \in \mathbb{R}^{l \times m}$$

$$Z \in \mathbb{R}^{m \times n}$$

$m \gg l$ overcomplete dictionary

$$X [] = [\cdot \cdot \cdot] e^T []$$

The k -SVD optimization problem

$$\hat{A}, \hat{Z} = \arg \min_{A, Z} \|X - AZ\|_F^2$$

$$\text{s.t. } \|\mathbf{z}_n\|_0 \leq T_o$$

OMP
 $T_o = 2$

$$\|A\|_F = \sqrt{\sum_{i=1}^l \sum_{j=1}^n |a_{ij}|^2}$$

Frobenius norm

$$[\quad \quad]$$

Step 1, sparse coding (membership update)

$$\begin{aligned}\hat{z}_n &= \arg \min_{\mathbf{z}_n} \|\mathbf{x}_n - A\mathbf{z}_n\|^2, \quad \forall n \\ \text{s.t. } \|\mathbf{z}_n\|_0 &\leq T_o \quad T_o = 4\end{aligned}$$

Step 2, cookbook/dictionary update (update vectors)

For the k 'th column in A , $\forall k$,

- ① Identify non-zeros in row $Z_{k,:}$ and denote that set K .
- ② Perform PCA/SVD on $X_{K,:}$:
- ③ Set the k 'th column in A to the principal component 1.

Is this similar to something you know?

https://youtu.be/-Y_Y4e-EKT0?t=642

The non-negative matrix factorization

Non-negative matrix factorization model

$$\tilde{X} \approx AZ + E \sim \mathcal{N}(0, \sigma^2)$$

$$X \in \mathbb{R}^{l \times n}$$

$$A \in \mathbb{R}_+^{l \times m}$$

$$Z \in \mathbb{R}_+^{m \times n}$$

$$[] = []^T$$

Model implications: models the superposition principle well.

X can be allowed to be “a little” negative, since we usually allow white noise residuals.

Extension:

- Put sparsity constraints on A and/or D .
- Put smoothness constraints on A , e.g. if A contains a smooth spectrum.

Good overview: The Why and How of Non-negative Matrix Factorization, 2014, Nicolas Gillis.

The ICA model

$$\mathbf{x} = A\mathbf{z}$$

$$X \in \mathbb{R}^{l \times n}$$

Mixtures & datapoints

$$A \in \mathbb{R}^{l \times l}$$

$$Z \in \mathbb{R}^{l \times n}$$

$$z = A^{-1} \cdot x$$

The ICA optimization problem

Identify A , but with respect to maximize statistical independence on the random variables in \mathbf{z}

But is this the task of PCA?

PCA identifies components such that z_i and z_j are uncorrelated ($i \neq j$), i.e (if zero-mean):

$$\mathbb{E}[z_i z_j] = 0 \quad = E[z_i] \cdot E[z_j]$$

ICA identifies component such that z_i and z_j are statistical independent ($i \neq j$), i.e

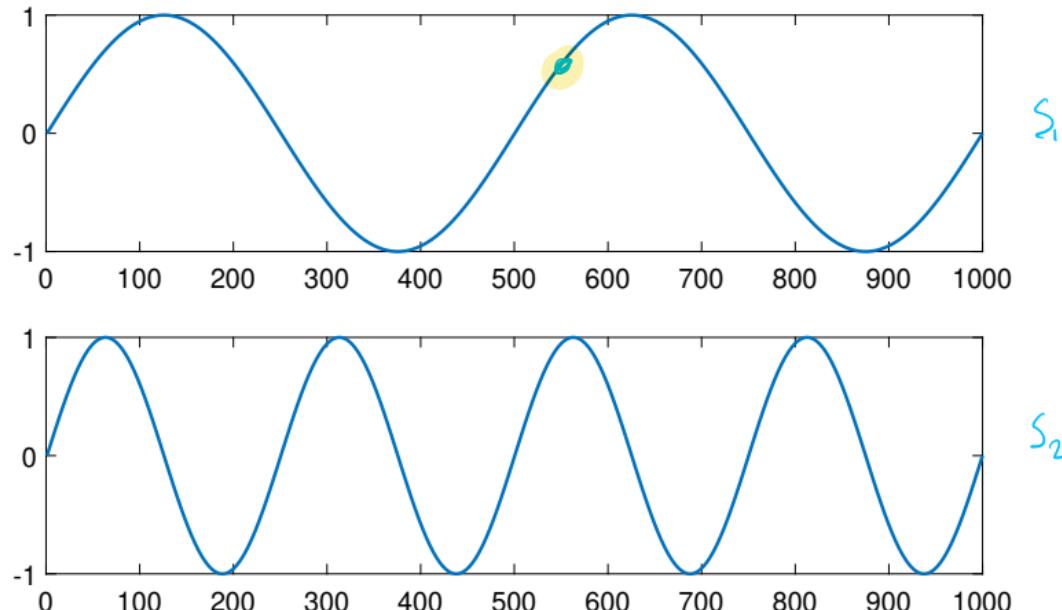
$$\begin{aligned} p(z_i | z_j) p(z_j) &= p(z_i, z_j) = p(z_i)p(z_j) \\ &= p(z_i) \end{aligned}$$

For Gaussian variables (if z is Gaussian), these statements are identical, see eq. (2.79)–(2.80).

Independent Component Analysis (ICA)

Example

Consider the following problem:



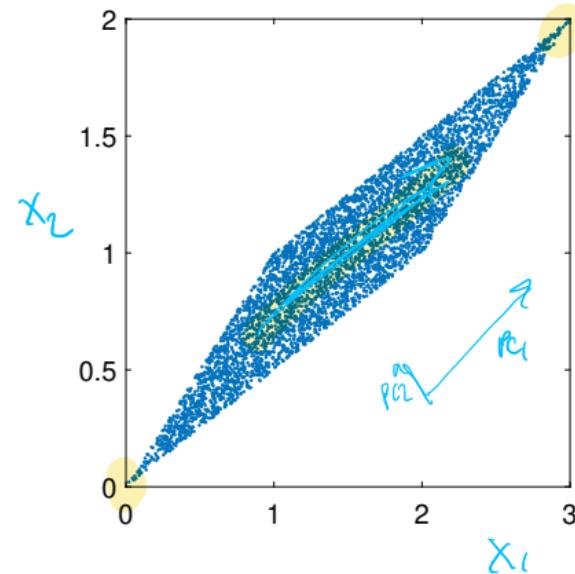
Signals are uncorrelated (covariance is zero), but not independent.

Independent Component Analysis (ICA)

Example 2

Consider the model

$$\begin{aligned}s_1 &\sim U(0, 1) \\ s_2 &\sim U(0, 1) \\ \mathbf{x} &= \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix} \cdot [s_1 \ s_2]^T \\ A &\end{aligned}$$



What would PCA produce?

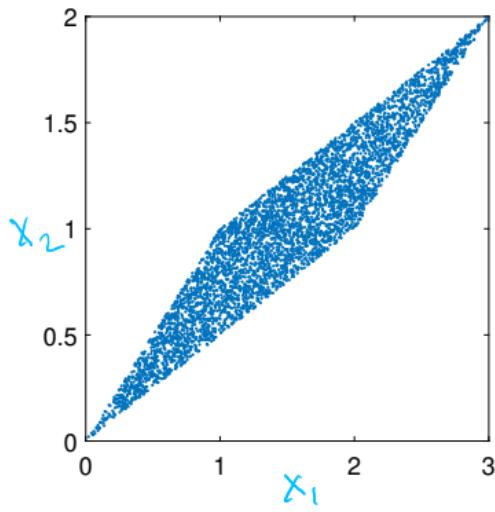
Independent Component Analysis (ICA)

Example 2

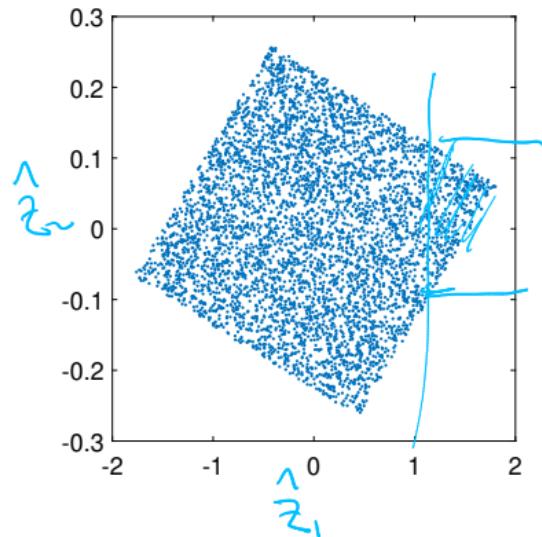
Ex 8.1



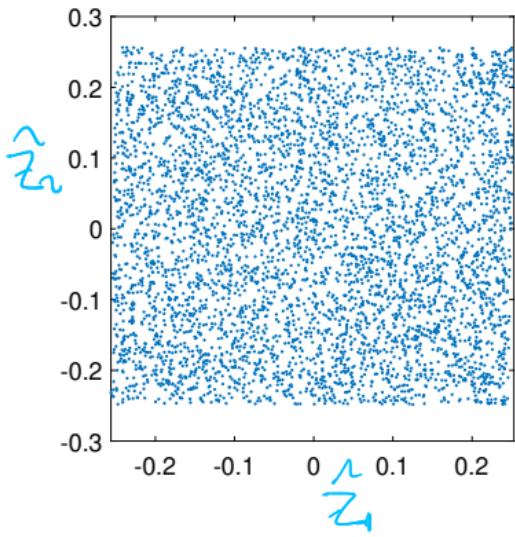
Observations



PCA



ICA



Independent Component Analysis (ICA)

ICA disclaimer



ICA is a huge topic, there are books dedicated to only this. E.g “Independent Component Analysis by Hyvärinen, Aapo; Karhunen, Juha; Oja, Erkki”, free at
<https://findit.dtu.dk/en/catalog/2304962533>

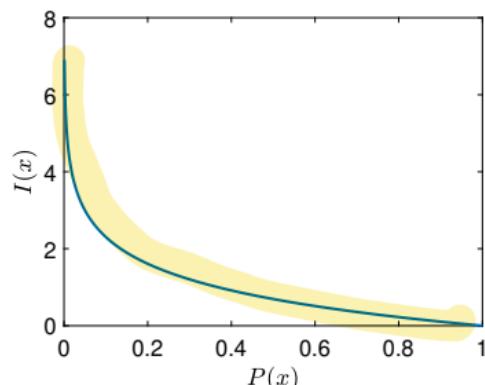
For real applications, use FastICA library to perform ICA.

Information of event (discrete variable)

The information for a particular value x for a discrete random variable x is

$$I(x) := -\log P(x) \quad \text{0} \leq P(x) \leq 1$$

The information measures the amount of “surprise”, e.g. sun in Sahara is expected thus this message has very low information, but water in Sahara is unexpected, thus this message has high information.



The information provided by the occurrence of event y about event x is called **mutual information**

Mutual information of events (discrete variable)

$$I(x; y) := \log \frac{P(x|y)}{P(x)}$$

if $P(x|y) = P(x) \Rightarrow I(x; y) = 0$
 $P(x|y) > P(x) \Rightarrow I(x; y) > 0$

Work through example 2.6 on your own.

Average mutual information of random variables (discrete case)

$$\mathbb{E}[I(x; y)] \quad I(x; y) := \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} P(x, y) \log \frac{P(x, y)}{P(x)P(y)}$$

The mutual information is a rewrite of $I(x; y) := \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} P(x, y)I(x; y)$.

Average mutual information of random variables (continuous case)

$$I(x; y) := \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} dx dy$$

One approach is to minimize the mutual information

$$p(x|y)p(y) = p(x)p(y)$$

$$I(x; y) := \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p(x, y) \log \frac{\hat{p}(x, y)}{p(x)p(y)} dx dy$$

if $\min I(x, y)$

\Rightarrow where x and y are ind.

$$\log 1$$



It can be shown that $I(x, y) \geq 0$ and $I(x, y) = 0$ implies that x and y are statistically independent.

$$= A^T$$

If $s = [x \ y]^T$ is our unobserved sources, estimate W such that we get sources that has minimum mutual information.

$$X = A \cdot Z$$

Independent Component Analysis (ICA)

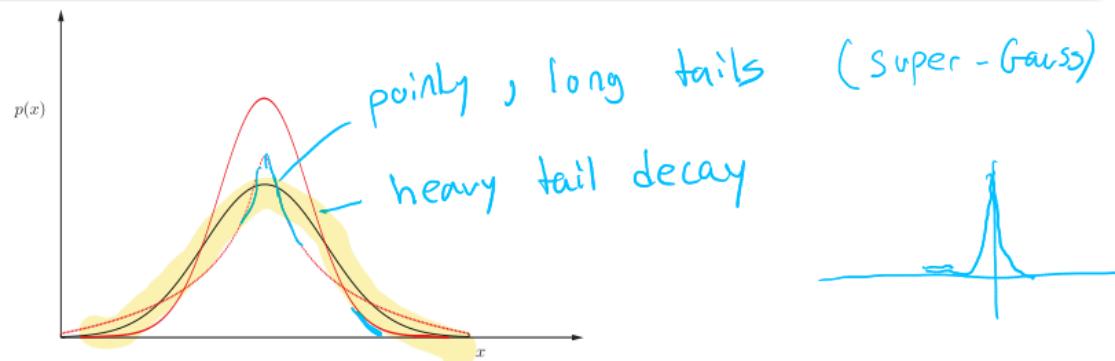
ICA update

ICA based on mutual information and natural gradient descent

- ① Randomly initialize W (W is the unmixing matrix and estimates A^{-1} ; model is $X = AZ$).
- ② Update rule: $W^{(i)} = W^{(i-1)} + \mu_i (I - \mathbb{E}[\phi(\mathbf{z})\mathbf{z}^T]) W^{(i-1)}$

Choose $\phi(\mathbf{z})$ as

- $\phi(\mathbf{z}) = 2 \tanh(\mathbf{z})$ if \mathbf{z} is super-Gaussian.
- $\phi(\mathbf{z}) = \mathbf{z} - \tanh(\mathbf{z})$ if \mathbf{z} is sub-Gaussian.



Independent Component Analysis (ICA)

Lecture summary



PCA, ICA, k -SVD, NMF - which should I choose ??? Depends on what you want to do:

- PCA obtains uncorrelated features, and great for feature extraction and dimensionality reduction.
- ICA obtains maximal independence but does not reduce dimensionality. Provides a sparser output and is perceptually more relevant.
 - There are more than one way to arrive at the ICA solution. We choose mutual information.
- NMF is best for analysis of non-negative data, e.g. pixels, energies, count data, etc, and often provides interpretable results (for non-negative data).
- k -SVD is best for sparse coding if a compact sparse dictionary is wanted with direct control on sparsity, but often is not very interpretable.

Next week

Week 46 material; ML 11.2, 12.1–12.2, 12.4–12.5, 12.10 (online).

- Bayesian linear regression.
- Expectation Maximization.

02471 Machine Learning for Signal Processing

Bayesian Inference and the EM algorithm

Tommy Sonne Alstrøm

Cognitive Systems section

$$f(x+\Delta x) = \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x)$$

$\Delta \int_a^b \Theta^{\sqrt{17}} \delta e^{i\pi} = -1$

Ω {2.7182818284} λ

$\Sigma!$

χ^2

\approx

\gg

\approx

\circ

\approx

\approx

Outline

- Course admin
- Assessment
- Last week review
- General linear models
- Parameter estimation
- Maximum likelihood and Bayesian inference
- Bayesian modeling
- Bayesian estimation in general linear models
- EM algorithm
- Next week

Material: 3.10–3.11 (recap), 11.2, 12.1–12.2, 12.4–12.5 (12.10 useful relations)

The story so far and what the future holds

What you have learned so far:

- Parameter estimation [L2 regularization, biased estimation, mean squared error minimization]. L1 regularization, **today: Bayesian parameter estimation**.
- Filtering signals [Stochastic processes, correlation functions, Wiener filter, linear prediction, adaptive filtering using stochastic gradient decent (LMS, APA/NLMS), adaptive filtering using regularization (RLS)]
- Signal representations [Time frequency analysis with STFT], Sparsity aware sensing (lasso, sparse priors), factor models [Independent component analysis, Non-negative matrix factorization, k -SVD],

Next three weeks: Bayesian parameter estimation and probabilistic graphical models

- **Inference and EM [Related to parameter estimation] (today)**. We will need EM for HMM.
- Sequential models [hidden Markov models, linear dynamical systems, Kalman filter].

Learning objectives

Learning objectives

A student who has met the objectives of the course will be able to:

- Explain, apply and analyze properties of discrete time signal processing systems
- Apply the short time Fourier transform to compute the spectrogram of a signal and analyze the signal content
- Explain compressed sensing and determine the relevant parameters in specific applications
- Deduce and determine how to apply factor models such as non-negative matrix factorization (NMF), independent component analysis (ICA) and sparse coding
- Deduce and apply correlation functions for various signal classes, in particular for stochastic signals
- Analyze filtering problems and demonstrate the application of least squares filter components such as the Wiener filter
- Describe, apply and derive non-linear signal processing methods based such as kernel methods and reproducing kernel Hilbert space for applications such as denoising
- Derive maximum likelihood estimates and apply the EM algorithm to learn model parameters
- Describe, apply and derive state-space models such as Kalman filters and Hidden Markov models
- Solve and interpret the result of signal processing systems by use of a programming language
- Design simple signal processing systems based on an analysis of involved signal characteristics, the objective of the processing system, and utility of methods presented in the course
- Describe a number of signal processing applications and interpret the results

Last week review

The linear factor model

Traditionally, digital signal processing seeks to **design** A (e.g. DCT), machine learning seeks to **learn** A .

Factor model with n measurements

$$\begin{aligned} X &= AZ \\ X &:= [\mathbf{x}_1, \dots, \mathbf{x}_N], \quad (l \times n) \end{aligned}$$

ICA, NMF, PCA and k -SVD are methods that use the above model, but **impose different constraints** in order to enforce the desired structure on A and Z .

- k -SVD, $A \in \mathbb{R}^{l \times m}$, $Z \in \mathbb{R}^{m \times n}$, T_0 -sparse loadings.
- NMF, $A \in \mathbb{R}_+^{l \times m}$, $Z \in \mathbb{R}_+^{l \times n}$.
- ICA, $A \in \mathbb{R}^{l \times l}$, $Z \in \mathbb{R}^{l \times n}$, estimate A^{-1} , minimize probabilistic dependence in z vectors.

Information of event (discrete variable)

The information measures the amount of “surprise”.

The information for a particular value x for a discrete random variable x is

$$I(x) := -\log P(x), \quad P(x) \leq 1 \Rightarrow I(x) \geq 0$$

Mutual information of events (discrete variable)

The information content provided by the occurrence of event y about event x is called **mutual information**

$$I(x; y) := \log \frac{P(x|y)}{P(x)}, \quad P(x, y) = P(x|y)P(y), \Rightarrow I(x; y) = \log \frac{P(x, y)}{P(x)P(y)}$$

Average mutual information

Mutual information of random variables (discrete case)

$$I(x; y) := \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} P(x, y) \log \frac{P(x, y)}{P(x)P(y)}$$

Average mutual information (continuous case)

$$I(x; y) := \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} dx dy$$

It can be shown that $I(x, y) \geq 0$ and $I(x, y) = 0$ implies that x and y are statistically independent.

The ICA model

$$\mathbf{x} = A\mathbf{z}$$

$$X \in \mathbb{R}^{l \times n}$$

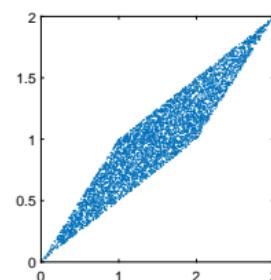
$$A \in \mathbb{R}^{l \times l}$$

$$Z \in \mathbb{R}^{l \times n}$$

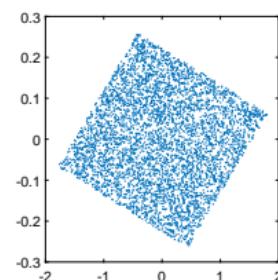
The ICA optimization problem:

Identify A^{-1} , but with respect to maximize statistical independence on the random variables in \mathbf{z}

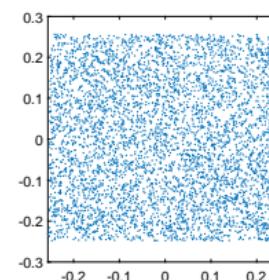
Observations



PCA



ICA

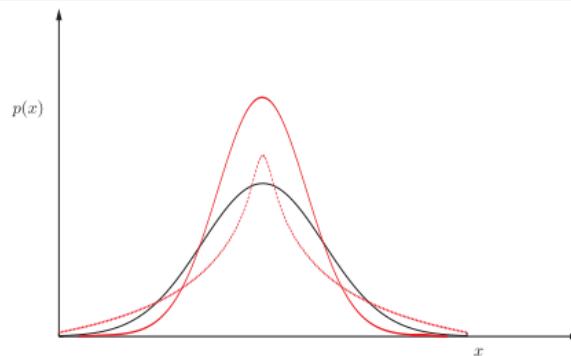


ICA based on mutual information and natural gradient descent

- ① Randomly initialize W (W is the unmixing matrix and estimates A^{-1} ; model is $X = AZ$).
- ② Update rule: $W^{(i)} = W^{(i-1)} + \mu_i (I - \mathbb{E}[\phi(\mathbf{z})\mathbf{z}^T]) W^{(i-1)}$

Choose $\phi(\mathbf{z})$ as

- $\phi(\mathbf{z}) = 2 \tanh(\mathbf{z})$ if \mathbf{z} is super-Gaussian (more spiky, longer tails).
- $\phi(\mathbf{z}) = \mathbf{z} - \tanh(\mathbf{z})$ if \mathbf{z} is sub-Gaussian (faster tail decay).



- ICA obtains maximal independence but does not reduce dimensionality.
- PCA obtains uncorrelated features, and great for feature extraction and dimensionality reduction.
- NMF is best for analysis of non-negative data, e.g. pixels, energies, count data, etc.
- k -SVD is best for sparse coding if a compact sparse dictionary is wanted with direct control on sparsity.

General linear models

The general linear model

General linear models (NB book calls this generalized linear models)

$$y = f(\mathbf{x}, \boldsymbol{\theta}) := \theta_0 + \sum_{k=1}^K \theta_k \phi_k(\mathbf{x}) + \eta$$

$\phi_k(\mathbf{x})$ is any function that maps $\mathbf{x} \in \mathbb{R}^l$, $\phi_k : \mathbb{R}^l \rightarrow \mathbb{R}$

Example:

A popular choice for the two-dimensional case is:

$$\phi_1(\mathbf{x}) = x_1, \phi_2(\mathbf{x}) = x_2, \phi_3(\mathbf{x}) = x_1^2, \phi_4(\mathbf{x}) = x_2^2, \phi_5(\mathbf{x}) = x_1 x_2$$

What is this model?

The general linear model

General linear models (NB book calls this generalized linear models)

$$y = f(\mathbf{x}, \boldsymbol{\theta}) := \theta_0 + \sum_{k=1}^K \theta_k \phi_k(\mathbf{x}) + \eta$$

$\phi_k(\mathbf{x})$ is any function that maps $\mathbf{x} \in \mathbb{R}^l$, $\phi_k : \mathbb{R}^l \rightarrow \mathbb{R}$

Example:

A popular choice for the two-dimensional case is:

$$\phi_1(\mathbf{x}) = x_1, \phi_2(\mathbf{x}) = x_2, \phi_3(\mathbf{x}) = x_1^2, \phi_4(\mathbf{x}) = x_2^2, \phi_5(\mathbf{x}) = x_1 x_2$$

A linear model with both quadratic effects and interaction effect. The model is used e.g. in ANOVA analysis and statistical analysis of experiments

The model is linear in parameters $\boldsymbol{\theta}$, hence we can use all tools developed so far.

The general linear model matrix form

Suppose we observe N measurements, $(y, \mathbf{x}) \in \mathbb{R} \times \mathbb{R}^l$, then

General linear model in matrix form

$$\mathbf{y} = \Phi\boldsymbol{\theta} + \boldsymbol{\eta}$$

$$\mathbf{y} = [y_1 \ y_2 \ \cdots \ y_N]^T \in \mathbb{R}^{N \times 1}$$

$$\phi(\mathbf{x}) = [\phi_1(\mathbf{x}) \ \phi_2(\mathbf{x}) \ \cdots \ \phi_{K-1}(\mathbf{x}) \ 1]^T \in \mathbb{R}^{K \times 1}$$

$$\Phi = [\phi(\mathbf{x}_1) \ \phi(\mathbf{x}_2) \ \cdots \ \phi(\mathbf{x}_N)]^T \in \mathbb{R}^{N \times K}$$

Why? we had something simple, why make it more complicated ???

We will work with this model for the rest of the lecture.

Note: this model is elaborated substantively in the course 02424 Advanced Data Analysis and Statistical Modeling

- The general linear model approach is a powerful idea to capture non-linear trends in data, and still keep the parameter estimation task linear.
- Having this in mind, empowers all results we have derived so far for linear models.

Parameter estimation

Parameter estimation so far

Loss function approach

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$J(\boldsymbol{\theta}) := \sum_{n=1}^N \mathcal{L}(y_n, f_{\boldsymbol{\theta}}(\mathbf{x}_n))$$

Least-Squares (LS) loss function

$$\mathcal{L}(y_n, f_{\boldsymbol{\theta}}(\mathbf{x}_n)) = (y_n - f_{\boldsymbol{\theta}}(\mathbf{x}_n))^2$$

$$\begin{aligned} J(\boldsymbol{\theta}) &= \sum_{n=1}^N (y_n - f_{\boldsymbol{\theta}}(\mathbf{x}_n))^2 \\ &= \|\mathbf{y} - f_{\boldsymbol{\theta}}(\mathbf{X})\|_2^2 \end{aligned}$$

Bayes theorem

$$p(\boldsymbol{\theta}|\mathcal{X}) = \frac{p(\mathcal{X}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{X})} \quad \mathcal{X} \text{ is our observed data } (\mathbf{y}, \mathbf{X})$$

Taking the log we get

$$\begin{aligned}\ln p(\boldsymbol{\theta}|\mathcal{X}) &= \ln p(\mathcal{X}|\boldsymbol{\theta}) + \ln p(\boldsymbol{\theta}) - \ln p(\mathcal{X}) \\ \underbrace{\ln p(\boldsymbol{\theta}|\mathcal{X})}_{\text{log posterior}} &\propto \underbrace{\ln p(\mathcal{X}|\boldsymbol{\theta})}_{\text{log likelihood}} + \underbrace{\ln p(\boldsymbol{\theta})}_{\text{log prior}}\end{aligned}$$

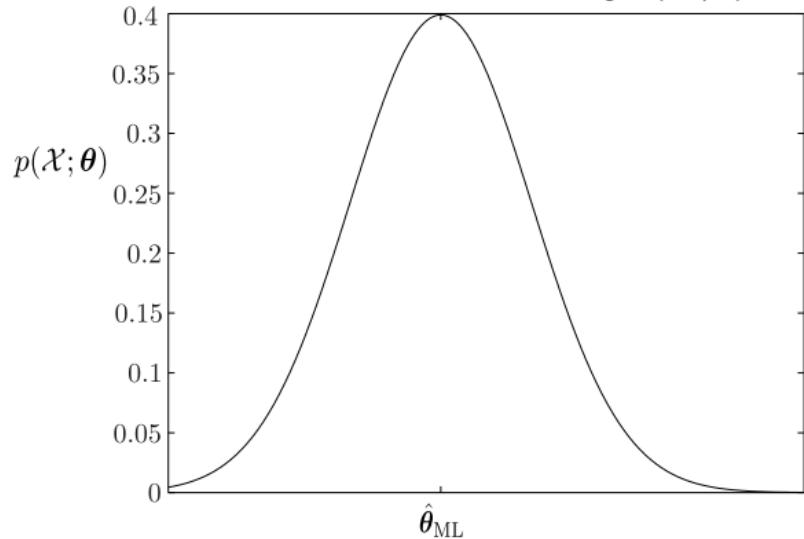
There is a direct correspondence between Bayes theorem and loss functions used so far:

- MSE: Normal likelihood with variance $\sigma^2 I$ + constant prior.
- Ridge: Normal likelihood with variance $\sigma^2 I$ + Normal prior with variance $\sigma_p^2 I$.
- LASSO: Normal likelihood with variance $\sigma^2 I$ + Laplace prior.

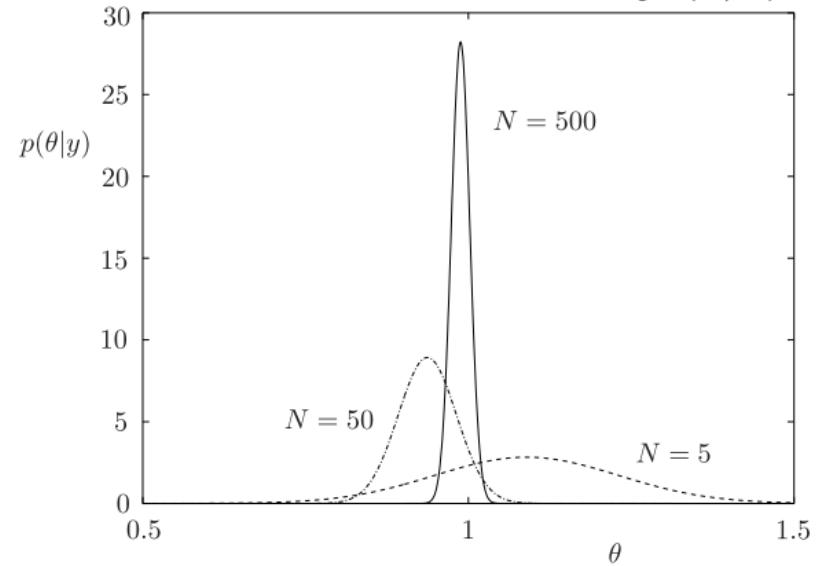
Maximum likelihood and Bayesian inference illustrated

$$\ln p(\boldsymbol{\theta}|\mathcal{X}) = \ln p(\mathcal{X}|\boldsymbol{\theta}) + \ln p(\boldsymbol{\theta}) - \ln p(\mathcal{X})$$

ML estimation: $\hat{\boldsymbol{\theta}}_{\text{ML}} = \arg \max_{\boldsymbol{\theta}} p(\mathcal{X}|\boldsymbol{\theta})$



MAP estimation: $\hat{\boldsymbol{\theta}}_{\text{MAP}} = \arg \max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{X})$



Maximum likelihood and Bayesian inference

A principled framework

Let us consider the marginal distribution (from eq. 2.23):

Marginal distribution

$$p(y) = \int_{-\infty}^{\infty} p(y, \theta) d\theta = \int_{-\infty}^{\infty} p(y|\theta)p(\theta)d\theta$$

A more informative name is the **prior predictive distribution**. Let us modify this with our observed data \mathcal{X} , and our model parameters θ .

For a specific point x where we want to predict y , we get

$$\begin{aligned} p(y|x, \mathcal{X}) &= \int_{-\infty}^{\infty} p(y, \theta|x, \mathcal{X}) d\theta \\ &= \int_{-\infty}^{\infty} p(y|x, \theta)p(\theta|\mathcal{X})d\theta \end{aligned}$$

This distribution is called the **posterior predictive distribution**, or sometimes just the **predictive distribution**.

Maximum likelihood and Bayesian inference

Three approaches to prediction

Maximum likelihood

$$\hat{\boldsymbol{\theta}}_{\text{ML}} = \arg \max_{\boldsymbol{\theta}} p(\mathcal{X}|\boldsymbol{\theta})$$

Maximum a posteriori

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \arg \max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{X})$$

Use the estimated weights, $\hat{\boldsymbol{\theta}}$ to perform prediction, $\hat{y} = f(\mathbf{x}, \hat{\boldsymbol{\theta}})$.

Posterior predictive distribution

$$p(y|\mathbf{x}, \mathcal{X}) = \int_{-\infty}^{\infty} p(y|\mathbf{x}, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{X})d\boldsymbol{\theta}$$

Use the mean of the posterior predictive distribution to perform prediction $\hat{y} = \mathbb{E}[p(y|\mathbf{x}, \mathcal{X})]$.

- There are a couple of ways to address the parameter estimation problem:
 - Define a cost function and optimize to get point estimates.
 - Define a likelihood function (noise model) and optimize to get point estimates.
 - Define a likelihood function (noise model) and prior probability that encodes prior beliefs, and optimize to get point estimates.
 - Define a likelihood function (noise model) and prior probability that encodes prior beliefs, and learn the complete probability distribution for our parameters.
- We can directly relate the cost function approach to a statistical approach using Bayes formula. In particular
 - LS regression corresponds to Normal likelihood.
 - Ridge regression corresponds to Normal likelihood+Normal prior.
 - LASSO regression corresponds to Normal likelihood+Laplace prior.

Bayesian modeling

Three steps in Bayesian data analysis

- ① Setting up a **full probability model** – a joint probability distribution for all observable and un-observable quantities in a problem. The model should be consistent with knowledge about the underlying scientific problem and the data collection process.
- ② Conditioning on observed data: calculating and interpreting the appropriate **posterior distribution** – the conditional probability distribution of the unobserved quantities of ultimate interest, given the observed data.
- ③ Evaluating the fit of the model and the implications of the resulting posterior distribution: does the model fit the data, are the substantive conclusions reasonable, and how sensitive are the results to the modeling assumptions in step 1?

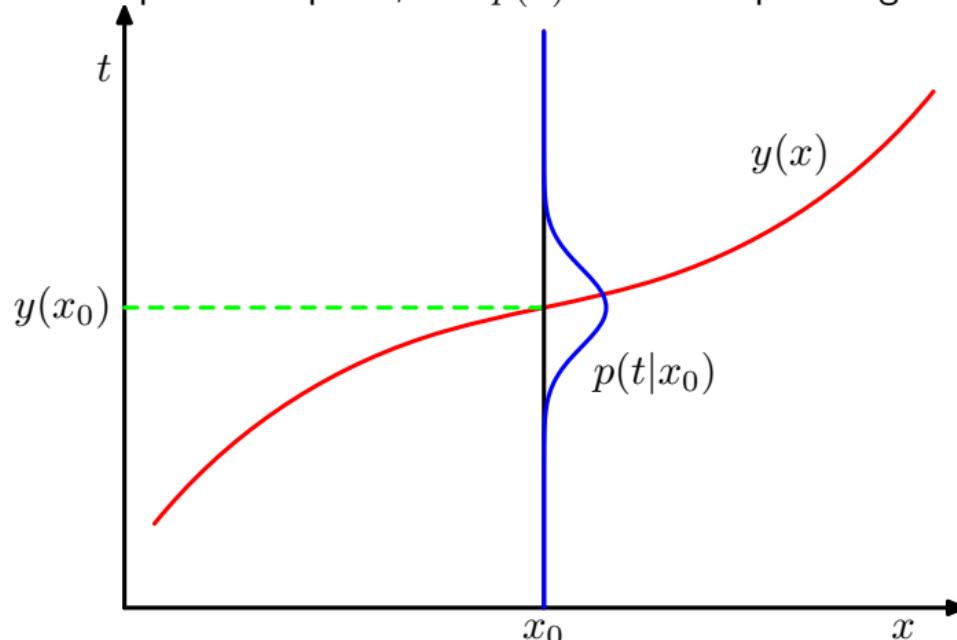
If necessary, one can alter or expand the model and repeat the three steps.

Source: Bayesian data analysis, 2014, Gelman, Carlin, Stern, Dunson, Vehtari, Rubin

Bayesian modeling

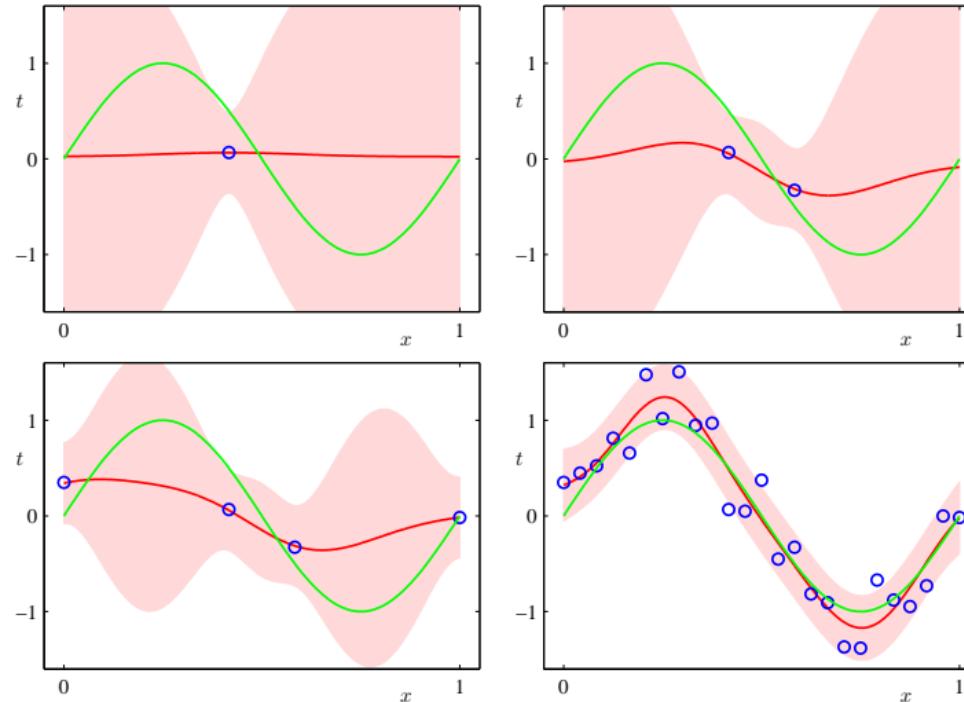
Why Bayesian? example for posterior predictive distribution

Example: $p(t|x_0)$ is the posterior predictive distribution conditioned on a specific point x_0 , $y(x_0)$ is the mean at that particular point, and $p(x)$ is the complete regression.



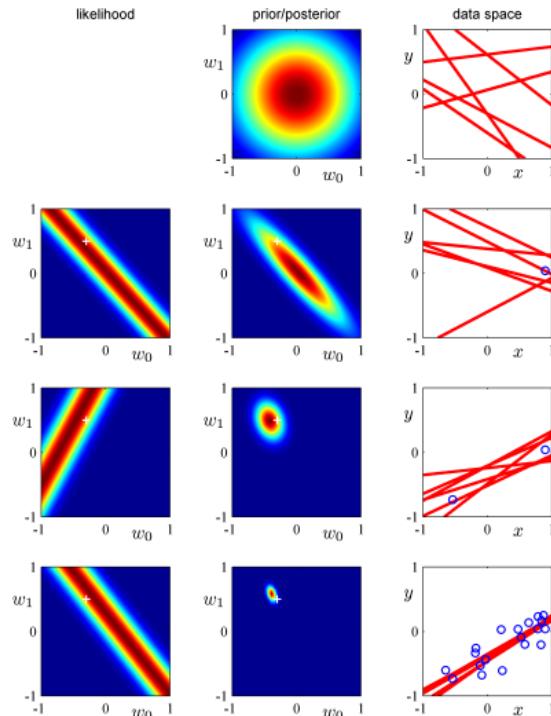
Source: Pattern Recognition and Machine Learning, 2006, C. Bishop

Why Bayesian? inherent uncertainty quantification



Source: Pattern Recognition and Machine Learning, 2006, C. Bishop

Why Bayesian? sequential learning



Source: Pattern Recognition and Machine Learning, 2006, C. Bishop

Why Bayesian? can lead to better predictions

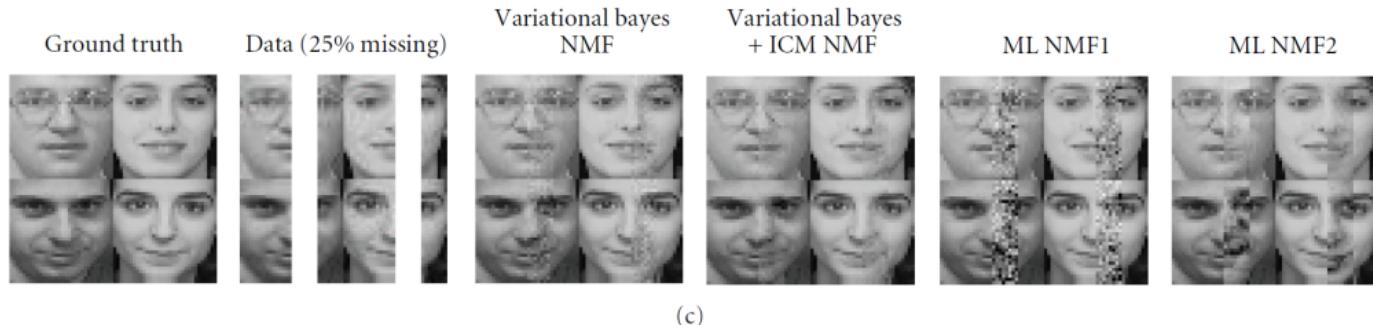


FIGURE 6: Results of a typical run. (a) Example images from the dataset. (b) Comparison of the reconstruction accuracy of different methods in terms of SNR (in dB), organised according to the sparseness of the solution. (c) (from left to right). The ground truth, data with missing pixels. The reconstructions of VB, VB + ICM, and ML-NMF with two initialisation strategies (1 = random, 2 = to image).

Bayesian Inference for Nonnegative Matrix Factorisation Models, 2009, Ali Taylan Cemgil

Note: there are numerous ways to carry out Bayesian inference, the book enumerates some in sec 12.2.3.

Most of those are taught in 02477 Bayesian machine learning.

Bayesian estimation in general linear models

Bayesian modeling: a model with Gaussian likelihood and Gaussian prior

To avoid notational clutter, we will now ignore x and \mathcal{X} , and ie. write $p(\mathbf{y}|\boldsymbol{\theta}) = p(\mathbf{y}|x, \mathcal{X}, \boldsymbol{\theta})$.

Assume, from now on: both the likelihood and the prior is Gaussian:

$$\begin{aligned} p(\boldsymbol{\theta}) &= \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\theta}_0, \Sigma_{\boldsymbol{\theta}}) \\ p(\mathbf{y}|\boldsymbol{\theta}) &= \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}_y, \Sigma_y) \end{aligned}$$

For now, we assume $\boldsymbol{\theta}_0$ and $\Sigma_{\boldsymbol{\theta}}$ are given, and we need to determine $\boldsymbol{\mu}_y$ and Σ_y .

Bayesian estimation in general linear models

Derivation of μ_y and Σ_y

We assume the model

$$\mathbf{y} = \Phi\boldsymbol{\theta} + \boldsymbol{\eta}$$

Let us evaluate the mean:

$$\begin{aligned}\boldsymbol{\mu}_y &= \mathbb{E}[\mathbf{y}] \\ &= \mathbb{E}[\Phi\boldsymbol{\theta} + \boldsymbol{\eta}] \\ &= \Phi\boldsymbol{\theta} + \mathbb{E}[\boldsymbol{\eta}]\end{aligned}$$

For zero-mean noise ($\mathbb{E}[\boldsymbol{\eta}] = 0$) we then have $\boldsymbol{\mu}_y = \Phi\boldsymbol{\theta}$. For the covariance we get

$$\begin{aligned}\boldsymbol{\Sigma}_y &:= \mathbb{E}\left[(\mathbf{y} - \mathbb{E}[\mathbf{y}]) (\mathbf{y} - \mathbb{E}[\mathbf{y}])^T\right] \\ &= \mathbb{E}\left[(\Phi\boldsymbol{\theta} + \boldsymbol{\eta} - \Phi\boldsymbol{\theta}) (\Phi\boldsymbol{\theta} + \boldsymbol{\eta} - \Phi\boldsymbol{\theta})^T\right] \\ &= \mathbb{E}\left[\boldsymbol{\eta}\boldsymbol{\eta}^T\right] \\ &= \boldsymbol{\Sigma}_{\boldsymbol{\eta}}\end{aligned}$$

Bayesian estimation in general linear models

Our updated model I

By substitution our model now becomes

$$p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\theta}_0, \Sigma_{\theta})$$
$$p(\mathbf{y}|\boldsymbol{\theta}) = \mathcal{N}(\mathbf{y}; \Phi\boldsymbol{\theta}, \Sigma_{\eta})$$

Our goal: determine $p(\boldsymbol{\theta}|\mathbf{y})$.

We can use relations from sec 12.10. (Skim them to familiarize yourself with the of relations!)

The section is not part of the regular download, you can find it on DTU learn.

The notation choice of x and y is arbitrary. In sec. 12.10, we find:

- Conditional $p(x|y)$ when $p(x, y)$ is Gaussian.
- The marginal $p(x)$ when $p(x, y)$ is Gaussian.
- The posterior $p(y|x)$ when $p(x|y)$ and $p(y)$ are Gaussian distributions (will be used in todays exercise).

Bayesian estimation in general linear models

Our updated model II

By substitution our model now becomes

$$\begin{aligned} p(\boldsymbol{\theta}) &= \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\theta}_0, \Sigma_{\theta}) \\ p(\mathbf{y}|\boldsymbol{\theta}) &= \mathcal{N}(\mathbf{y}; \Phi\boldsymbol{\theta}, \Sigma_{\eta}) \end{aligned}$$

From sec 12.10. IF:

$$\begin{aligned} p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_z, \Sigma_z) \\ p(\mathbf{t}|\mathbf{z}) &= \mathcal{N}(\mathbf{t}|\mathbf{z}; A\mathbf{z}, \Sigma_{t|z}) \end{aligned}$$

Then the posterior is:

$$\begin{aligned} p(\mathbf{z}|\mathbf{t}) &= \mathcal{N}(\mathbf{z}|\mathbf{t}; \boldsymbol{\mu}_{z|t}, \Sigma_{z|t}) \\ \boldsymbol{\mu}_{z|t} &= \boldsymbol{\mu}_z + \Sigma_{z|t} A^T \Sigma_{t|z}^{-1} (\mathbf{t} - A\boldsymbol{\mu}_z) \\ \Sigma_{z|t} &= (\Sigma_z^{-1} + A^T \Sigma_{t|z}^{-1} A)^{-1} \end{aligned}$$

What is \mathbf{z} , \mathbf{t} , $\boldsymbol{\mu}_z$, Σ_z , A , and $\Sigma_{t|z}$?

Bayesian estimation in general linear models

Our updated model II

By substitution our model now becomes

$$\begin{aligned} p(\boldsymbol{\theta}) &= \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\theta}_0, \Sigma_{\theta}) \\ p(\mathbf{y}|\boldsymbol{\theta}) &= \mathcal{N}(\mathbf{y}; \Phi\boldsymbol{\theta}, \Sigma_{\eta}) \end{aligned}$$

From sec 12.10. IF:

$$\begin{aligned} p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_z, \Sigma_z) \\ p(\mathbf{t}|\mathbf{z}) &= \mathcal{N}(\mathbf{t}|\mathbf{z}; A\mathbf{z}, \Sigma_{t|z}) \end{aligned}$$

Then the posterior is:

$$\begin{aligned} p(\mathbf{z}|\mathbf{t}) &= \mathcal{N}(\mathbf{z}|\mathbf{t}; \boldsymbol{\mu}_{z|t}, \Sigma_{z|t}) \\ \boldsymbol{\mu}_{z|t} &= \boldsymbol{\mu}_z + \Sigma_{z|t} A^T \Sigma_{t|z}^{-1} (\mathbf{t} - A\boldsymbol{\mu}_z) \\ \Sigma_{z|t} &= (\Sigma_z^{-1} + A^T \Sigma_{t|z}^{-1} A)^{-1} \end{aligned}$$

Answer: $\mathbf{z} = \boldsymbol{\theta}$, $\mathbf{t} = \mathbf{y}$, $\boldsymbol{\mu}_z = \boldsymbol{\theta}_0$, $\Sigma_z = \Sigma_{\theta}$, $A = \Phi$, and $\Sigma_{t|z} = \Sigma_{\eta}$.

The complete Bayesian general linear regression model, with Gaussian likelihood and Gaussian prior

$$\begin{aligned} \mathbf{y} &= \Phi\boldsymbol{\theta} + \boldsymbol{\eta} \\ p(\boldsymbol{\theta}) &= \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\theta}_0, \Sigma_{\theta}) \\ p(\mathbf{y}|\boldsymbol{\theta}) &= \mathcal{N}(\mathbf{y}; \Phi\boldsymbol{\theta}, \Sigma_{\eta}) \\ p(\boldsymbol{\theta}|\mathbf{y}) &= \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}_{\theta|y}, \Sigma_{\theta|y}) \\ \boldsymbol{\mu}_{\theta|y} &= \boldsymbol{\theta}_0 + \Sigma_{\theta|y}\Phi^T\Sigma_{\eta}^{-1}(\mathbf{y} - \Phi\boldsymbol{\theta}_0) \\ \Sigma_{\theta|y} &= (\Sigma_{\theta}^{-1} + \Phi^T\Sigma_{\eta}^{-1}\Phi)^{-1} \end{aligned}$$

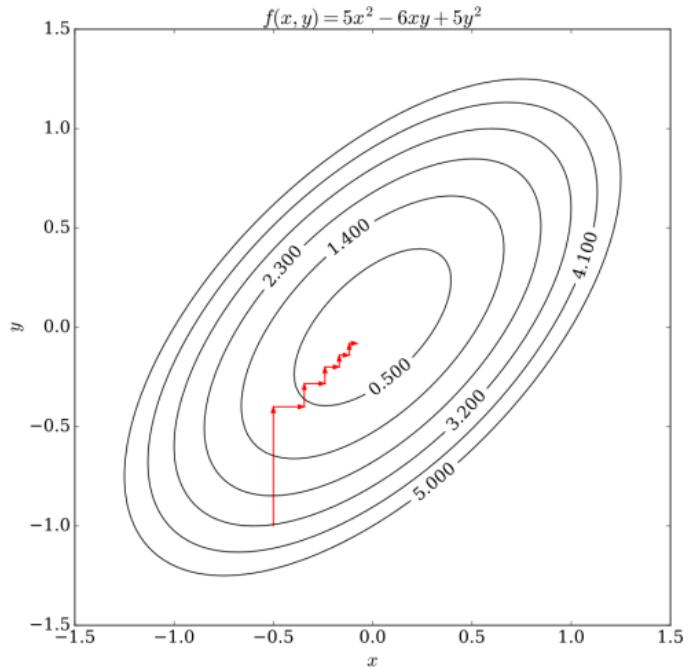
What is required to be estimated to use this model?

EM algorithm

EM algorithm

Coordinate descent

The algorithm have a similar feel to coordinate descent (Source: wikipedia):



The EM algorithm is an iterative algorithm, similar to coordinate descent

The EM algorithm

Steps to carry out before optimization:

- Specification of the complete log-likelihood, $\ln p(\mathbf{y}, \boldsymbol{\theta})$ (choice of model).
- Derive $Q(\boldsymbol{\xi}, \boldsymbol{\xi}^{(j)}) = \mathbb{E}[\ln p(\mathbf{y}, \boldsymbol{\theta}; \boldsymbol{\xi}^{(j)})]$ to create update formulas.

Randomly initialize $\boldsymbol{\xi}^{(0)}$ and run until convergence (e.g until $\|\boldsymbol{\xi}^{(j+1)} - \boldsymbol{\xi}^{(j)}\| < \epsilon$)

① Compute $Q(\boldsymbol{\xi}, \boldsymbol{\xi}^{(j)})$

② Maximize $Q(\boldsymbol{\xi}, \boldsymbol{\xi}^{(j)})$ in order to get $\boldsymbol{\xi}^{(j+1)}$

Algorithm analysis and derivation outside the scope of this course.

EM algorithm

An example EM update

Let us simplify our linear model and let $\boldsymbol{\theta}_0 = \mathbf{0}$, $\Sigma_\theta = \alpha^{-1}I$ and $\Sigma_\eta = \beta^{-1}I$.

The parameters we need to learn using EM is then $\xi = [\alpha, \beta]^T$.

The original model:

$$\begin{aligned} p(\boldsymbol{\theta}) &= \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\theta}_0, \Sigma_\theta) \\ p(\mathbf{y}|\boldsymbol{\theta}) &= \mathcal{N}(\mathbf{y}; \Phi\boldsymbol{\theta}, \Sigma_\eta) \\ p(\boldsymbol{\theta}|\mathbf{y}) &= \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}_{\theta|y}, \Sigma_{\theta|y}) \\ \boldsymbol{\mu}_{\theta|y} &= \boldsymbol{\theta}_0 + \Sigma_{\theta|y}\Phi^T\Sigma_\eta^{-1}(\mathbf{y} - \Phi\boldsymbol{\theta}_0) \\ \Sigma_{\theta|y} &= (\Sigma_\theta^{-1} + \Phi^T\Sigma_\eta^{-1}\Phi)^{-1} \end{aligned}$$

Changes to:

$$\begin{aligned} p(\boldsymbol{\theta}) &= \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \alpha^{-1}I) \\ p(\mathbf{y}|\boldsymbol{\theta}) &= \mathcal{N}(\mathbf{y}; \Phi\boldsymbol{\theta}, \beta^{-1}I) \\ p(\boldsymbol{\theta}|\mathbf{y}) &= \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}_{\theta|y}, \Sigma_{\theta|y}) \\ \boldsymbol{\mu}_{\theta|y} &= \beta\Sigma_{\theta|y}\Phi^T\mathbf{y} \\ \Sigma_{\theta|y} &= (\alpha I + \beta\Phi^T\Phi)^{-1} \end{aligned}$$

An example EM update

EM for Bayesian linear regression

Expectation step:

$$\boldsymbol{\mu}_{\theta|y}^{(j)} = \beta^{(j)} \Sigma_{\theta|y}^{(j)} \Phi^T \mathbf{y}$$

$$\Sigma_{\theta|y}^{(j)} = (\alpha^{(j)} I + \beta^{(j)} \Phi^T \Phi)^{-1}$$

$$A^{(j)} = \text{trace}(\Sigma_{\theta|y}^{(j)}) + \|\boldsymbol{\mu}_{\theta|y}^{(j)}\|^2$$

$$B^{(j)} = \|\mathbf{y} - \Phi \boldsymbol{\mu}_{\theta|y}^{(j)}\|^2 + \text{trace}(\Phi \Sigma_{\theta|y}^{(j)} \Phi^T)$$

$$\mathcal{Q}(\alpha, \beta; \alpha^{(j)}, \beta^{(j)}) = \frac{N}{2} \ln \beta + \frac{K}{2} \ln \alpha - \frac{\beta}{2} B^{(j)} - \frac{\alpha}{2} A^{(j)} - \left(\frac{N}{2} + \frac{K}{2} \right) \ln 2\pi$$

Maximization step:

$$\arg \max_{\alpha, \beta} \mathcal{Q}(\alpha, \beta; \alpha^{(j)}, \beta^{(j)}) \Rightarrow$$

$$\alpha^{(j+1)} = K/A^{(j)}$$

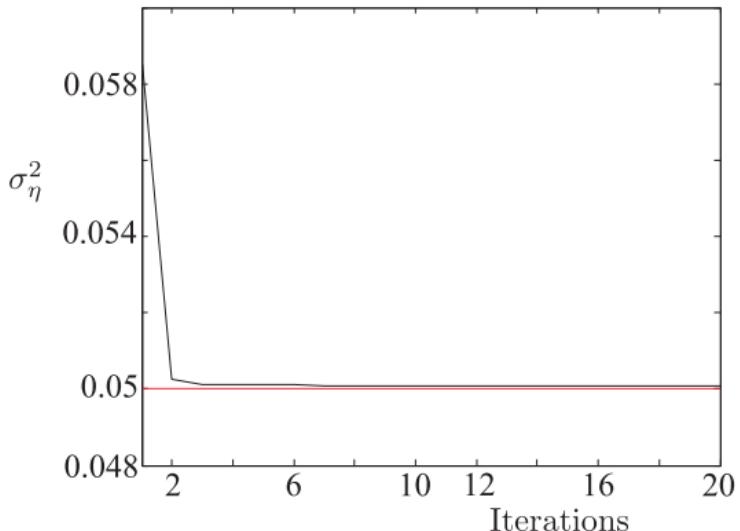
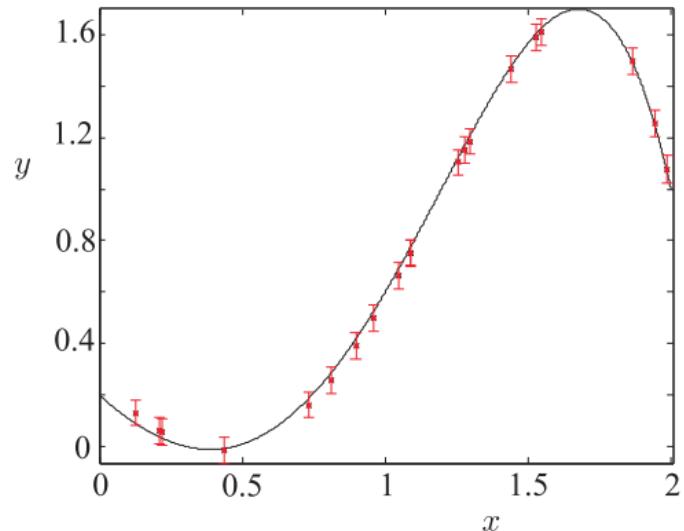
$$\beta^{(j+1)} = N/B^{(j)}$$

EM algorithm

EM results

Example 12.1 and 12.2: generate data from $y_n = \sum_{i=0}^5 \theta_i x_n^i + \eta_n$

Fit and Convergence curve (noise and error bars estimated directly):



You will derive and reproduce this in the exercise.

Lecture summary

- General linear models allows for modeling non-linear trends but still has linear parameter estimation.
- In the Bayesian approach, we learn a distribution over the parameters θ , and not only a parameter estimate.
- Bayesian data modeling provide inherent uncertainty quantification since we learn the distributions, and not point estimates.
- Regularization is inherent using the prior. Example for Ridge regression: the regularization parameter λ is “learned” without performing cross-validation.
- The EM algorithm is a scheme that can be applied, similar to how we have used gradient descent so far.
- Instead of deriving gradients as input to gradient descent, we derive $\mathcal{Q}(\xi, \xi^{(j)}) = \mathbb{E}[\ln p(\mathbf{y}, \theta; \xi^{(j)})]$ to create update formulas.
- EM can be used to learn distribution parameters. We will need EM again for Hidden Markov Model (next week).

Next week

Week 46 material; 15.1–15.3.1, 15.7, 16.4–16.5.

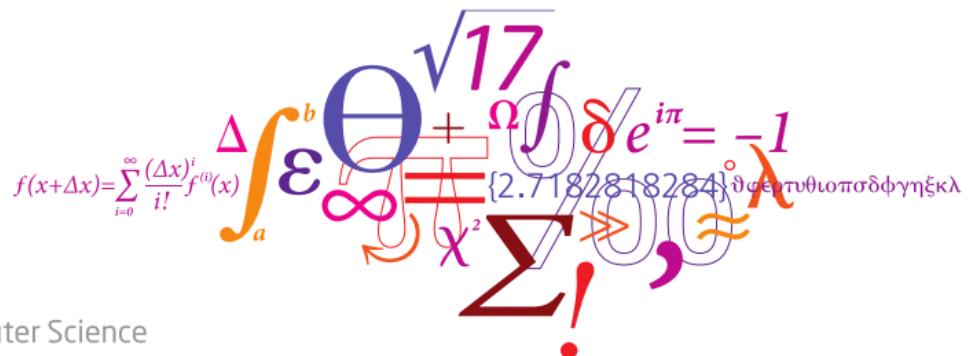
- Probabilistic graphical models.
- Hidden Markov models.

02471 Machine Learning for Signal Processing

State-space models – Hidden Markov Models

Tommy Sonne Alstrøm

Cognitive Systems section



Outline

- Last week review
- This week
- Probabilistic graphical models
- Examples
- The Hidden Markov Model
- Mathematical aspects of HMM
- Next week

Material: 15.1–15.3.1, 15.7, 16.4–16.5 (only 16.4–16.5 will be part of curriculum)

Course outline

So far:

- Parameter estimation [Regularization, biased estimation, mean squared error minimization].
- Signal representations [Time frequency analysis, sparsity aware learning, factor models].
- Filtering signals [Linear regression, adaptive filtering using stochastic gradient decent (LMS, NLMS, RLS), adaptive filtering using regularization].

Next weeks

- Inference and EM [Related to parameter estimation] (today).
- Sequential models [hidden markov models, linear dynamical systems, kalman filter].
- Kernel methods [non-linear models]. $\sim 15\text{ p.}$

Last week review

Last week review

Three approaches to prediction

Maximum likelihood

Point est $\hat{\theta}_{\text{ML}} = \arg \max_{\theta} p(\mathcal{X}|\theta)$

Maximum a posteriori

Point est $\hat{\theta}_{\text{MAP}} = \arg \max_{\theta} p(\theta|\mathcal{X})$ MLE + prior

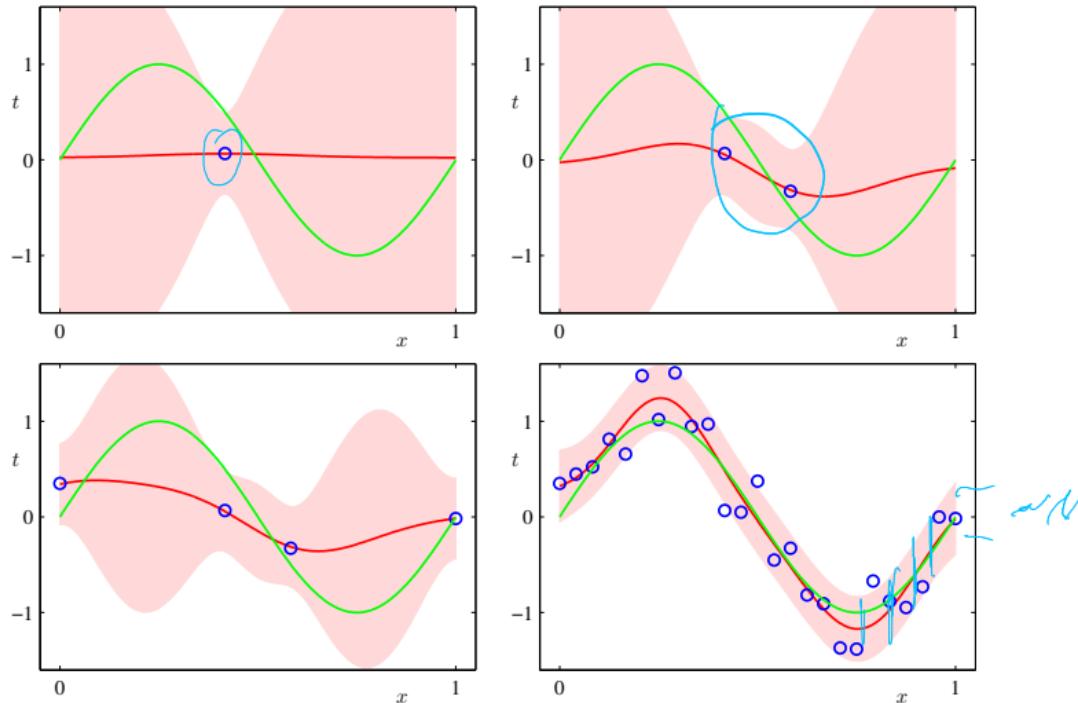
Use the estimated weights, $\hat{\theta}$ to perform prediction, $\hat{y} = f(\mathbf{x}, \hat{\theta})$.

Posterior predictive distribution

$$p(y|\mathbf{x}, \mathcal{X}) = \int_{-\infty}^{\infty} p(y|\mathbf{x}, \theta) p(\theta|\mathcal{X}) d\theta$$

Use the mean of the posterior predictive distribution to perform prediction $\hat{y} = \mathbb{E}[p(y|\mathbf{x}, \mathcal{X})]$.

Why Bayesian? inherent uncertainty quantification



Source: Pattern Recognition and Machine Learning, 2006, C. Bishop

Is an iterative algorithm, similar to coordinate descent

The EM algorithm

Steps to carry out before optimization:

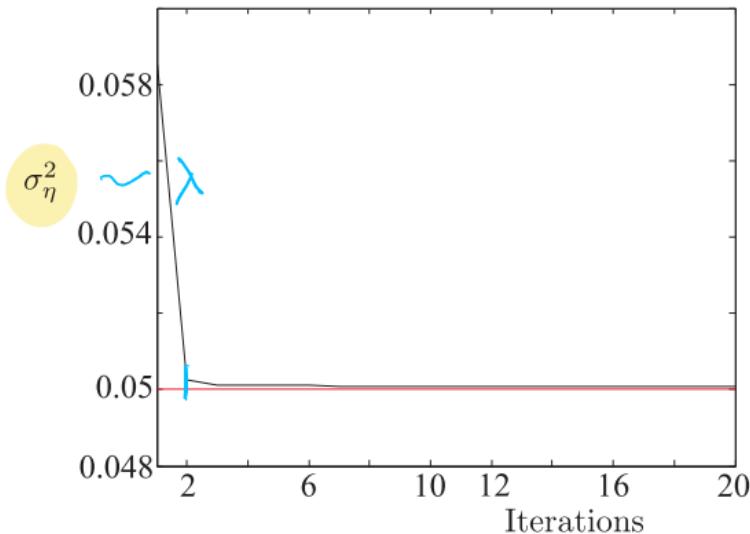
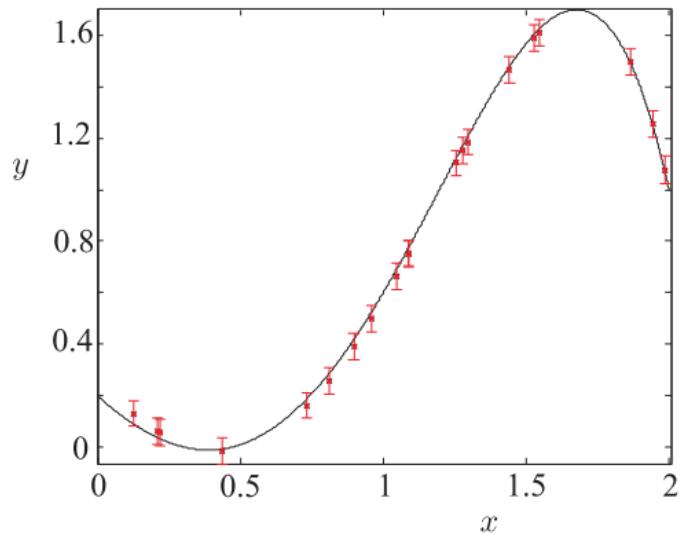
- Specification of the complete log-likelihood, $\ln p(\mathbf{y}, \boldsymbol{\theta})$ (choice of model)
- Derive $Q(\boldsymbol{\xi}, \boldsymbol{\xi}^{(j)}) = \mathbb{E}[\ln p(\mathbf{y}, \boldsymbol{\theta}; \boldsymbol{\xi}^{(j)})]$ to create update formulas. wrt p(hatenH Pato)

Randomly initialize $\boldsymbol{\xi}^{(0)}$ and run until convergence (e.g until $\|\boldsymbol{\xi}^{(j+1)} - \boldsymbol{\xi}^{(j)}\| < \epsilon$)

- ① Compute $Q(\boldsymbol{\xi}, \boldsymbol{\xi}^{(j)})$
- ② Maximize $Q(\boldsymbol{\xi}, \boldsymbol{\xi}^{(j)})$ in order to get $\boldsymbol{\xi}^{(j+1)}$

$\boldsymbol{\xi}^{(0)}$ needs to be spes.

Last week review EM results



- General linear models allows for modeling non-linearities but still keeps linear parameter estimation.
- Bayesian data modeling provide inherent uncertainty quantification since we learn the distributions, and not point estimates.
- Learning the models is more tedious.
- Often not tractable, but fully tractable if all distributions are Gaussian.
- EM can be used to learn distribution parameters.

A note on the material

This topic can be taught in a number of ways, because the material has been developed independently in different fields.

- In machine learning literature it is typically called Bayesian networks, which belongs to the area of "probabilistic graphical models".
- Statistics call them Markov models.
- Signal processing and time series analysis typically call it state-space models.

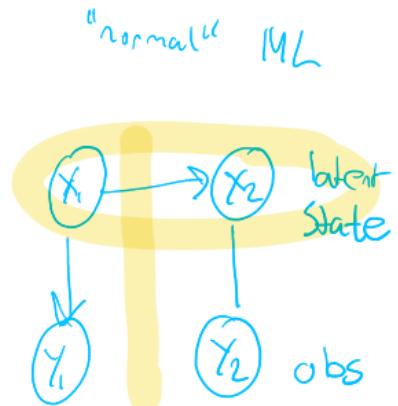
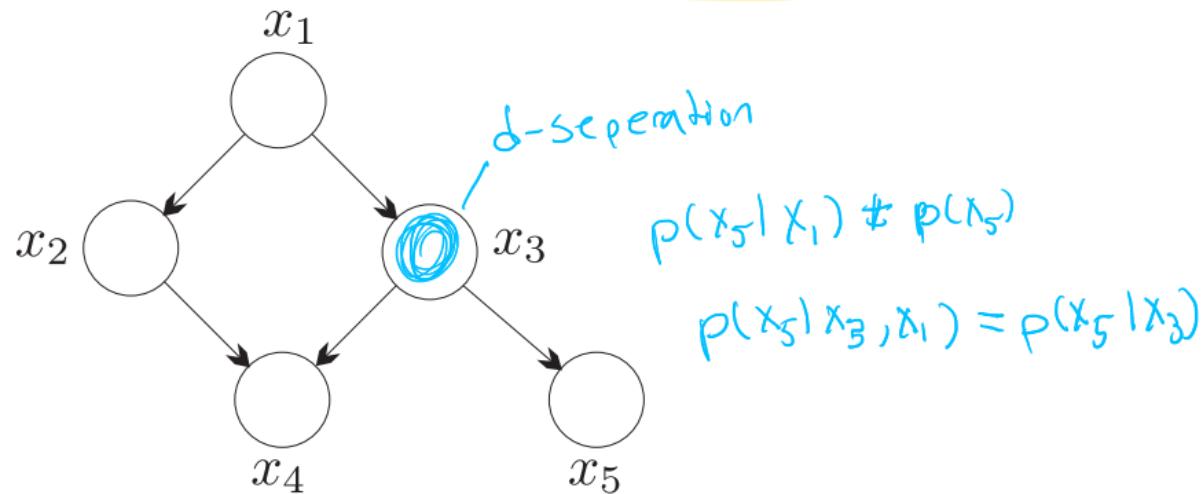
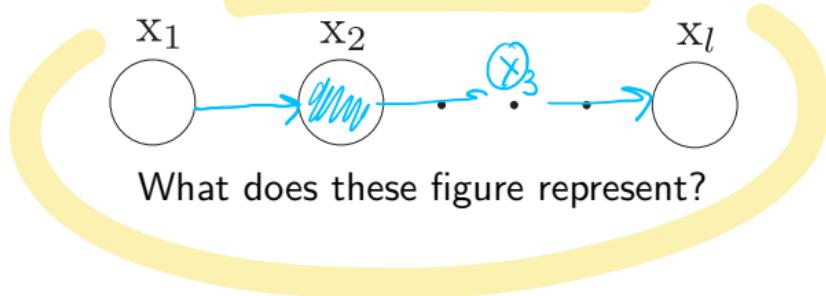
(They are not 100% exactly the same)

A note on the material: chapter 13 in "Pattern Recognition and Machine Learning by Christopher Bishop" <https://www.microsoft.com/en-us/research/people/cmbishop/> presents this material more coherently. In the ML book, this is unfortunately scattered across four chapters (4, 15, 16, and 17).

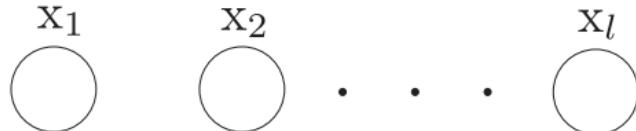
Probabilistic graphical models

Our starting point, the graphical model

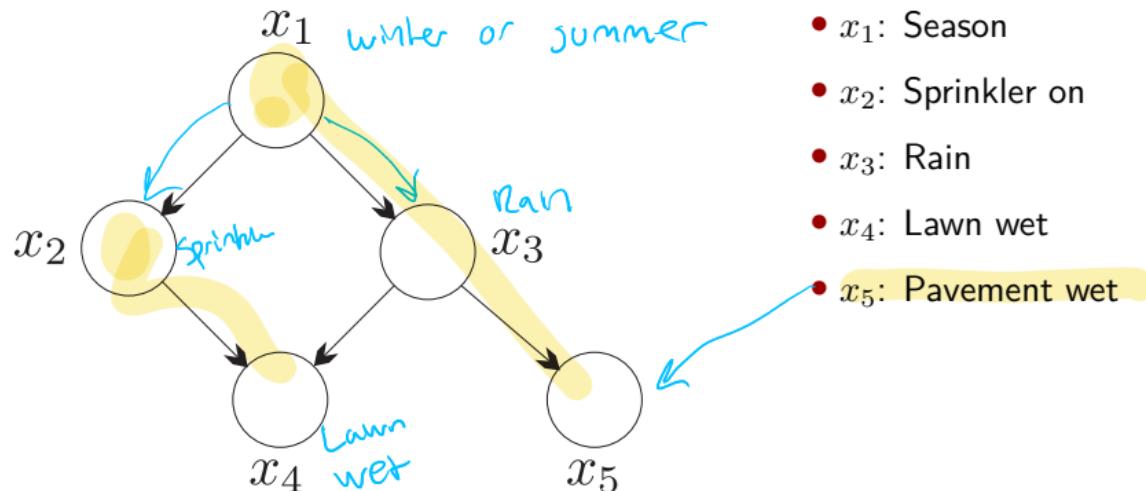
ind



Our starting point, the graphical model



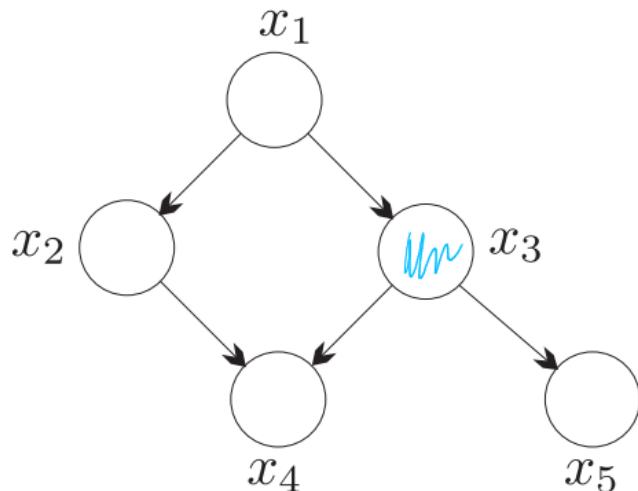
What does this figure represent?



- x_1 : Season
- x_2 : Sprinkler on
- x_3 : Rain
- x_4 : Lawn wet
- x_5 : Pavement wet

Definition of a Bayesian network and the Markov condition

A Bayesian network structure is a directed acyclic graph (DAG) whose nodes represent random variables, and every variable (node), is conditionally independent of the set of all its non-descendants, given the set of all its parents. This is known as the Markov condition.



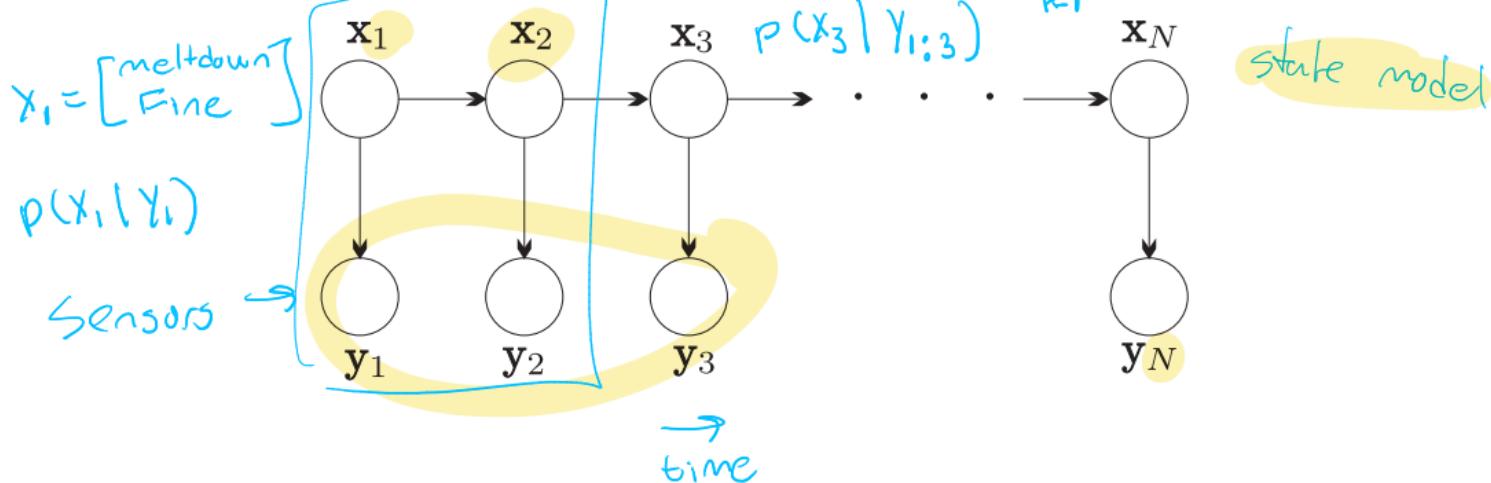
Example If we have observed x_3 , rain or no rain, then x_5 the probability of the pavement being wet, is no longer dependent on x_1 .

Formally we write

$$p(x_5|x_3, x_1) = p(x_5|x_3)$$

Probabilistic graphical models

State-space model as a Bayesian network



State-space formulation

$$\mathbf{x}_n = F_n \mathbf{x}_{n-1} + \mathbf{\eta}_n \sim \mathcal{N}(0, Q_n)$$

$$\mathbf{y}_n = H_n \mathbf{x}_n + \mathbf{v}_n \sim \mathcal{N}(0, R_n)$$

Probabilistic formulation

M,L

$$p(\mathbf{x}_n | \mathbf{x}_{n-1}) = \mathcal{N}(\mathbf{x}_n; \underline{F_n \mathbf{x}_{n-1}}, Q_n)$$

$$p(\mathbf{y}_n | \mathbf{x}_n) = \mathcal{N}(\mathbf{y}_n; \underline{H_n \mathbf{x}_n}, R_n)$$

How are Q_n and R_n related to the state-space formulation?

Linear dynamical system

encode physics

$$\mathbf{x}_n = F_n \mathbf{x}_{n-1} + \boldsymbol{\eta}_n, \quad \text{State equation}$$

$$\mathbf{y}_n = H_n \mathbf{x}_n + \mathbf{v}_n, \quad \text{Observation equation}$$

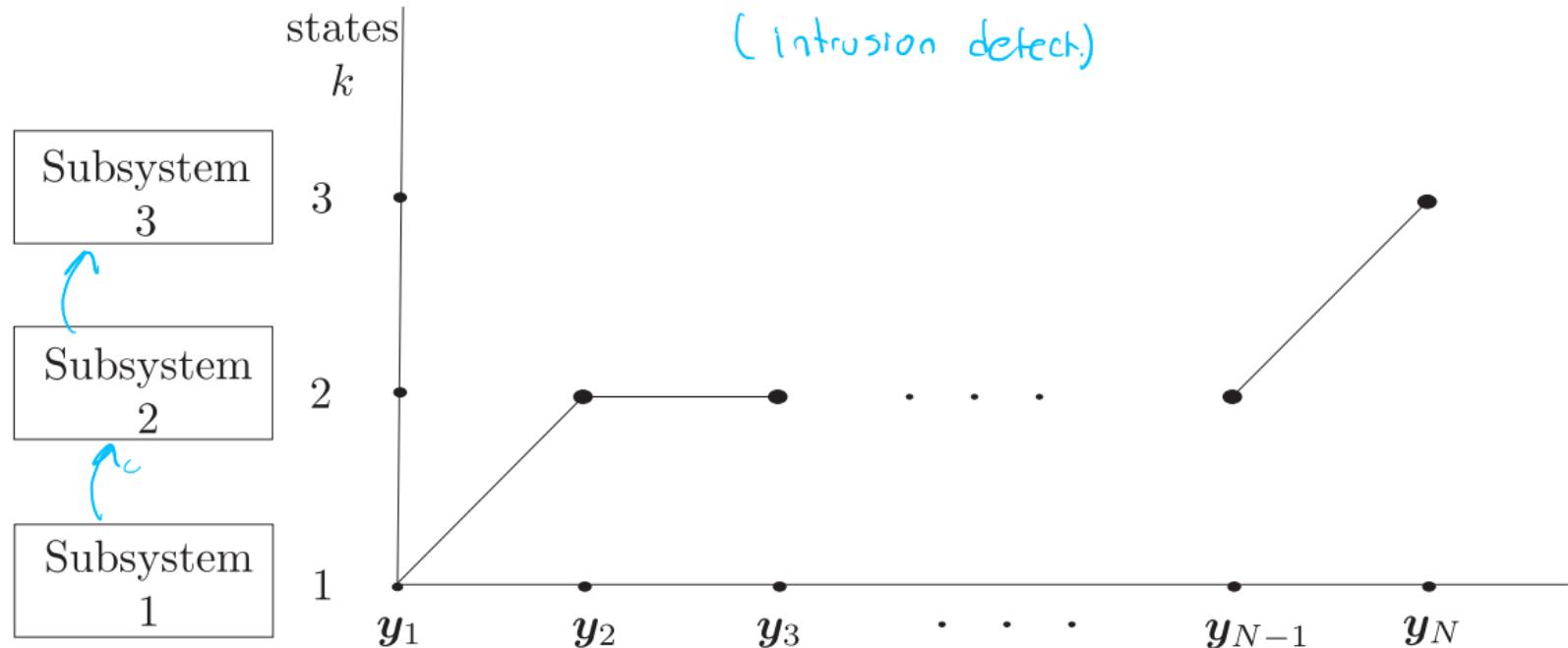
- If \mathbf{x}_n is discrete, we call the model a **Hidden Markov Model** (HMM).
- If \mathbf{x}_n is continuous and Gaussian, we call the model a **Linear dynamical system** (LDS).
- Additionally, if F_n , H_n , $\boldsymbol{\eta}_n$, and \mathbf{v}_n are known, we call it **Kalman filtering**. Or more precisely, **inference** in a linear dynamical system is called **Kalman filtering**.

Examples

Example of a left-to-right HMM

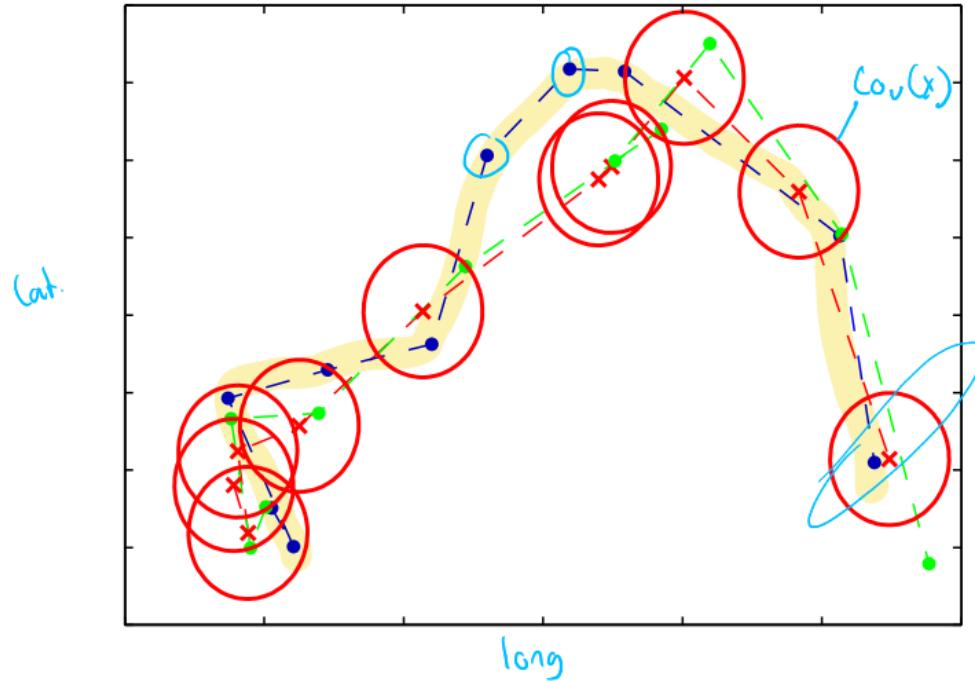
(meltdown case)

(intrusion detect.)



Example: event detection.

Example of Kalman filter



Example: moving object tracking. Green: noisy measurements, blue: true location, red: predicted.

More inspiration

Introduction to Hidden Markov Models with Python Networkx and Sklearn:

<http://www.blackarbs.com/blog/introduction-hidden-markov-models-python-networkx-sklearn/2/9/2017>

An example for implementing the Kalman filter for navigation where the vehicle state, position, and velocity are estimated by using sensor output from an inertial measurement unit (IMU) and a global navigation satellite system (GNSS) receiver:

<https://www.intechopen.com/books/introduction-and-implementations-of-the-kalman-filter/introduction-to-kalman-filter-and-its-applications>

Location tracking <https://jonathan-hui.medium.com/self-driving-object-tracking-intuition-and-the-math-behind-kalman-filter-657d11dd0a90>

Linked material is not part of the curriculum, they are suggested to aid your learning process.

For PGMs, consider the course “42186 – Model-based machine learning”

The Hidden Markov Model

HMM model parameters

A HMM model is fully described by the following set of parameters:

① Number of states K .

$$X_n = \{0, 1\}^K$$

② Initial state probability, P_k .

$$P_k = [P_1, \dots, P_K]$$

$$\sum P_k = 1$$

③ Transition probabilities, P_{ij} .

$$P = [P_{ij}]_{K \times K} \quad \sum_{j=1}^K P_{ij} = 1 \quad \forall i$$

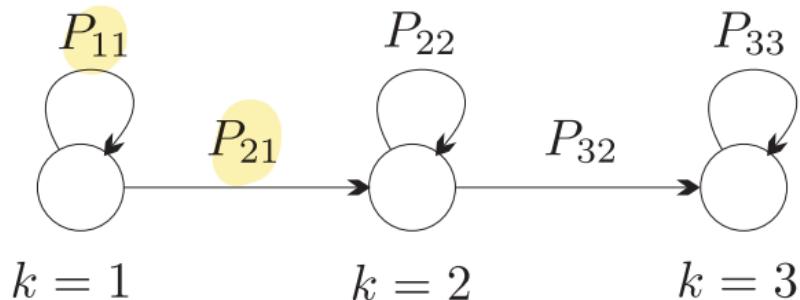
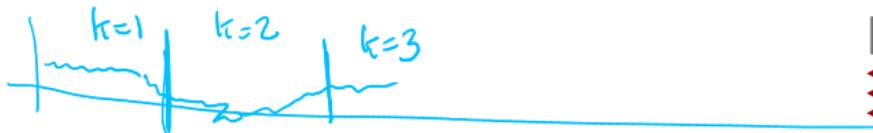
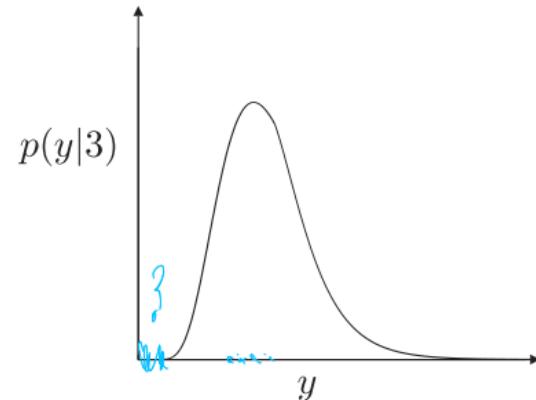
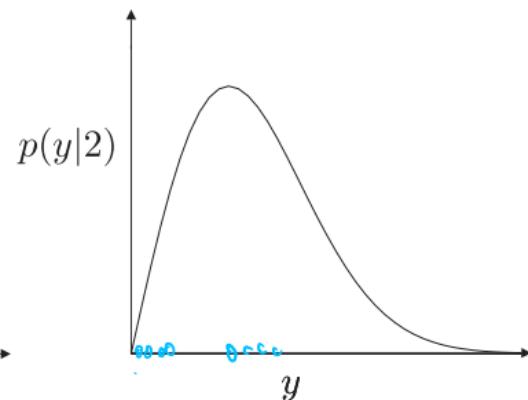
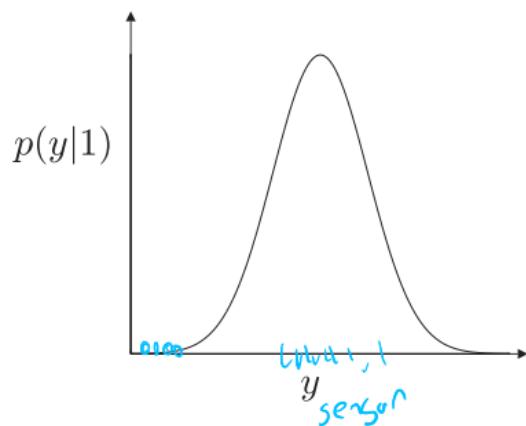
④ State emission distributions $p(y|k)$.

We can ask different questions: $p(\text{sensor}|\text{state})$.

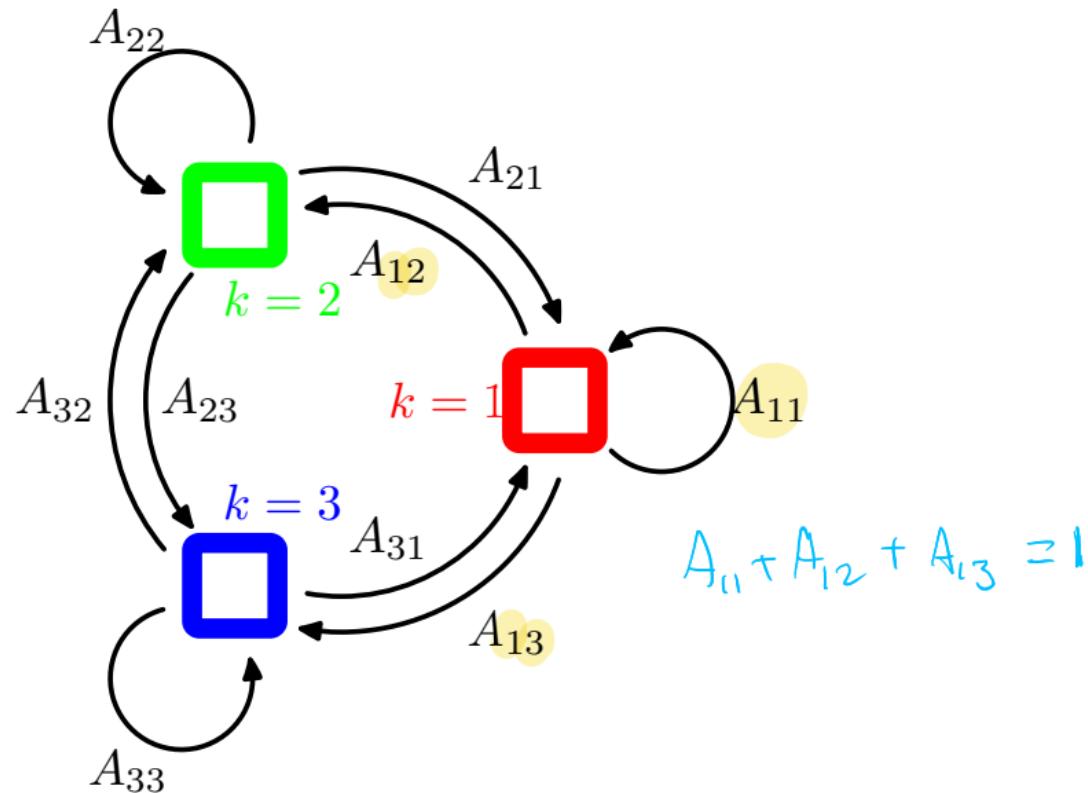
- Given an observed sequence y_1, \dots, y_n , which HMM, out of a database of HMMs most likely generated the sequence? Example?
- Given an observed sequence y_1, \dots, y_n , which state k are we most likely in, or, what is the predicted value y_{n+1} ? (why don't we just use regression??)

$$p(x_k=k|y_{1:n})$$

$$p(y_{n+1}|y_{1:n})$$

A three-state left-to-right HMM $k = 1$ $k = 2$ $k = 3$ 

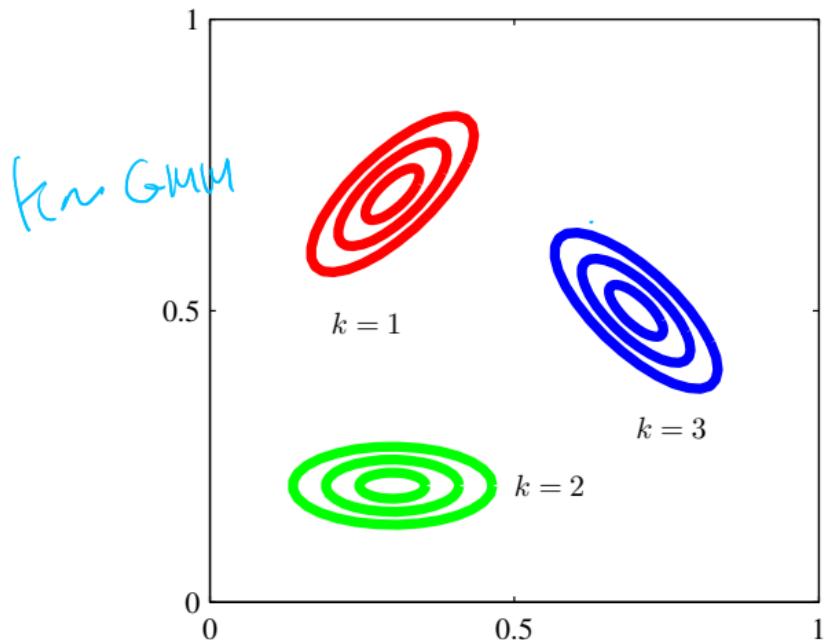
The Hidden Markov Model A full three-state HMM



The Hidden Markov Model

A simulation from a HMM

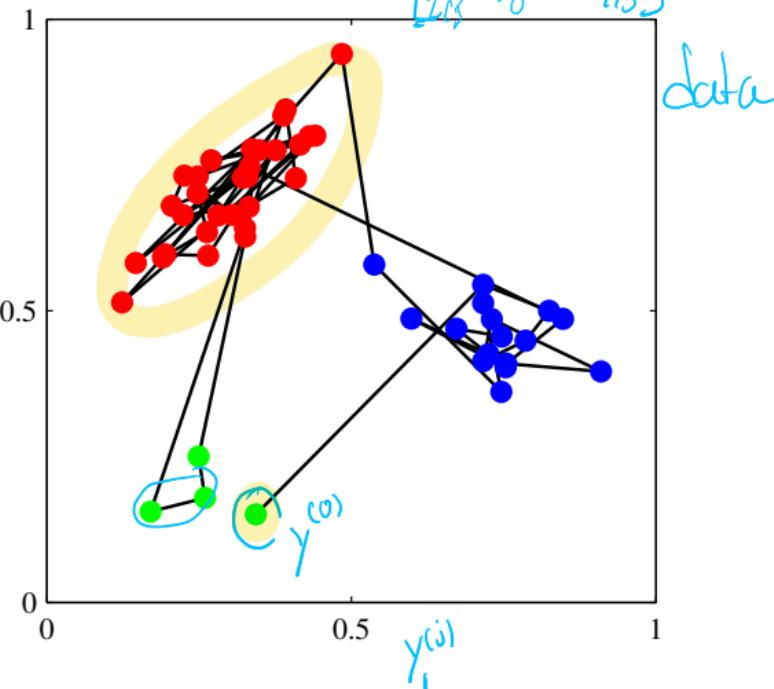
$$p(\gamma|k)$$



$$P_k = [0.2 \ 0.6 \ 0.2]$$

$$P_{ij} = \begin{bmatrix} 0.2 & 1.0 & 0.2 \\ 0.25 & 0.5 & 0.25 \\ 0.15 & 0.0 & 0.15 \end{bmatrix}$$

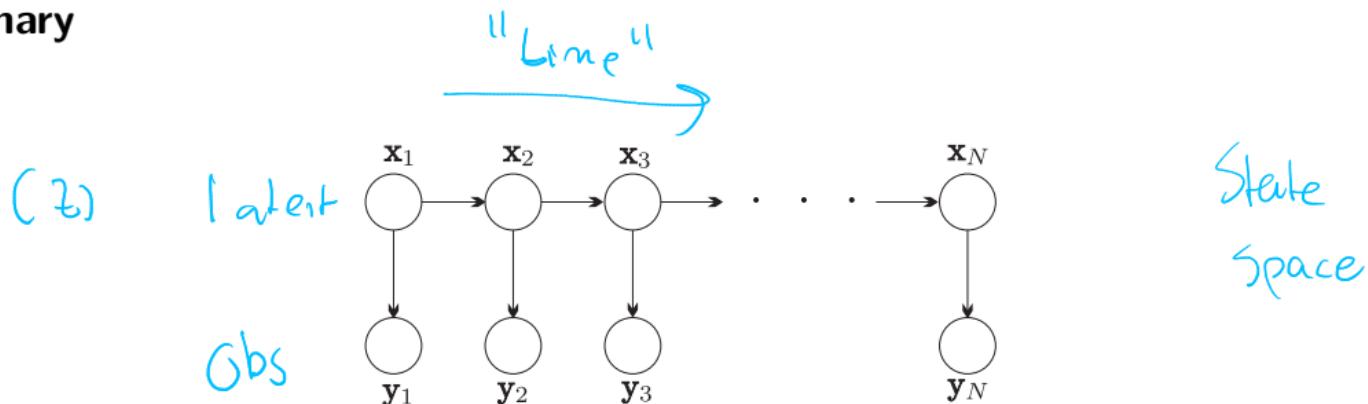
DTU



Put in the terms I have just talked about, what are we looking at?

The Hidden Markov Model

Summary



- A hidden Markov model (HMM) models the situation where you have K distinct states of your system.
- The observations can be discrete or continuous.
- Is well suited for a number of applications, e.g. sound classification, classification of bytes in communication etc.

Mathematical aspects of HMM

Mathematical aspects of HMM How to perform prediction

Sum rule: $P(x) = \sum_{y \in \mathcal{Y}} P(x, y)$

Product rule: $P(x, y) = P(x|y)P(y)$

Bayes Theorem: $P(y|x) = \frac{P(x,y)}{P(x)}$

$$p(y_1, y_2) = \sum_{i=1}^k \sum_{j=1}^k p(y_1, y_2, x_i^{(j)}, x_2^{(j)})$$

k^2

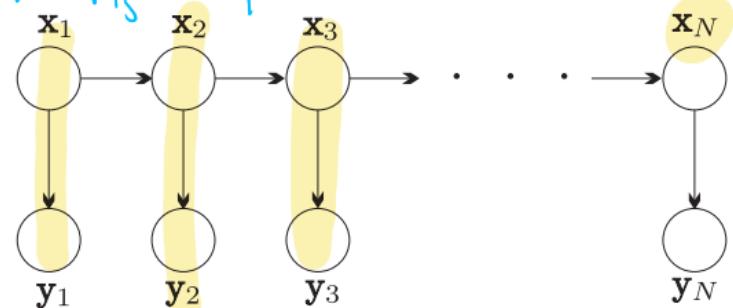
$$k^N$$

$$N \cdot k \quad (\text{sum-product})$$

Explain

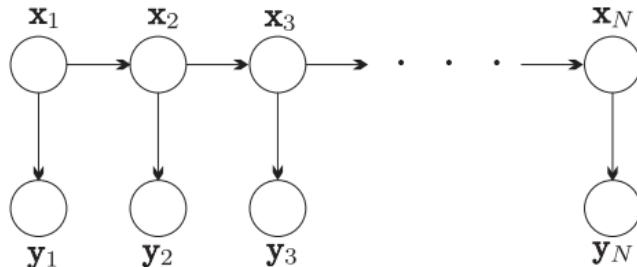
$$p(y|k)$$

$$p(x_j|x_{j-1}) = P_{ij}$$



Mathematical aspects of HMM

How to learn the parameters



Ex 10.2

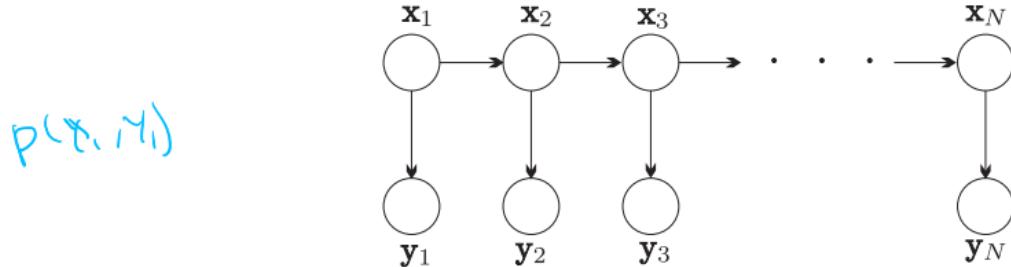
As a reminder, the EM algorithm consist of the following steps:

- ① Specification of the complete log likelihood, $\ln p(\mathcal{X}, \mathcal{X}^l)$ (the model).
- ② Derive $Q(\xi, \xi^{(j)}) = \mathbb{E}[\ln p(\mathcal{X}, \mathcal{X}^l; \xi^{(j)})]$. $p(x, y | \xi)$
- ③ Maximize $Q(\xi, \xi^{(j)})$ in order to get $\xi^{(j+1)}$.

where \mathcal{X} denotes the set of observations, \mathcal{X}^l denotes the set of latent random variables, and ξ is a vector of distribution parameters.

Mathematical aspects of HMM

How to learn the parameters



$p(x, y)$

Net $P(Y, X) = P(x_1)p(y_1|x_1) \prod_{n=2}^N p(x_n|x_{n-1})p(y_n|x_n)$ p(x_n, y_n | all prev data) Sum / product

$$P(x_1) = \prod_{k=1}^K P_k^{x_{1,k}}$$

$$P_1^{x_{1,1}} P_2^{x_{1,2}}$$

$$x_{1,k} \in O_c \quad \text{if } k=2$$

$$P(x_n|x_{n-1}) = \prod_{i=1}^K \prod_{j=1}^K P_{ij}^{x_{n-1,j} x_{n,i}}$$

$$P_1^0 P_2^1 = P_2 = [1, 0]$$

Con $p(y_n|x_n) = \prod_{k=1}^K (p(y_n|k; \theta_k))^{x_{n,k}}$

Mathematical aspects of HMM **Supplementary video material**



Prof. Patterson describes the Hidden Markov Model, starting with the Markov Model

https://www.youtube.com/watch?v=J_y5hx_ySCg&list=PLix7MmR3doRo3NGNzrq48FItR3TDyuLCo

Alternative (mathematicalmonk) <https://www.youtube.com/watch?v=7zDARfKVm7>

Videos are not part of the curriculum, they are suggested to aid your learning process.

- How do I perform (complete) inference (prediction) in HMM ? (ML, sec 16.5.1)
- How do I fully train the parameters in HMM? (require full EM updates, sec 16.5.2)
- How do I compute the most likely state, given a sequence (Viterbi ML, sec 15.7+16.5.2)

- Presented the state-space model, which is a very popular ML model for sequential data.
- If x_n is discrete, we call the model a Hidden Markov Model (HMM).
- Is well suited for a number of applications, e.g. sound classification, classification of bytes in communication , gene sequencing, diagnosis etc, anywhere you have a finite number of states.
- HMM is trained using the EM algorithm.
- If x_n is continuous and Gaussian, we call the model a Linear dynamical system (LDS).
- For Linear dynamical systems, Kalman filtering is the prediction/update formulas (next week).
- We did not cover how to fully train/perform inference in HMM.

Next week

Week 47 material; 4.9–4.9.1, 4.10, 17.3

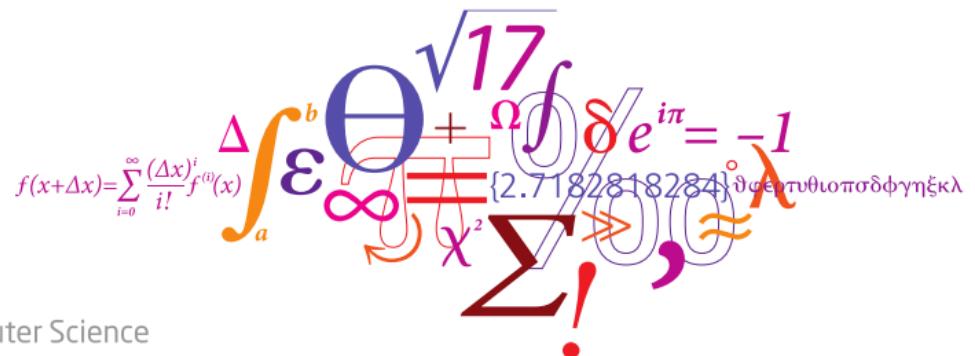
- State-space models (LDS)
- Kalman filter

02471 Machine Learning for Signal Processing

State-space models – Linear Dynamical Systems

Tommy Sonne Alstrøm

Cognitive Systems section



Outline

- Course admin
- Last week review
- Kalman filtering
- (Filtering from a Bayesian viewpoint)
- Next week

Material: 4.9–4.9.1, 4.10, (17.3)

4,7 → 4,9

- Official evaluation is up, please answer the survey <https://evaluering.dtu.dk/>

What you have learned so far:

- Parameter estimation [L2 regularization, biased estimation, mean squared error minimization]. L1 regularization, Bayesian parameter estimation.
- Filtering signals [Stochastic processes, correlation functions, Wiener filter, linear prediction, adaptive filtering using stochastic gradient decent (LMS, APA/NLMS), adaptive filtering using regularization (RLS)]
- Signal representations [Time frequency analysis with STFT], Sparsity aware sensing (lasso, sparse priors), factor models [Independent component analysis, Non-negative matrix factorization, k -SVD],
- Bayesian parameter estimation and probabilistic graphical models. Inference and EM.

Next weeks:

- Today: Kalman filtering
- Kernel methods [non-linear models, kernels, kernel Ridge regression, support vector regression].

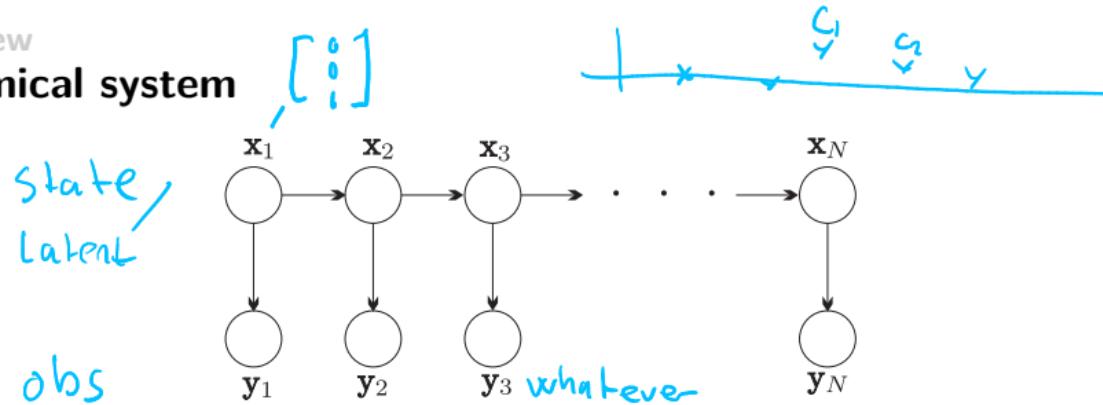
Learning objectives

Learning objectives

A student who has met the objectives of the course will be able to:

- Explain, apply and analyze properties of discrete time signal processing systems
- Apply the short time Fourier transform to compute the spectrogram of a signal and analyze the signal content
- Explain compressed sensing and determine the relevant parameters in specific applications
- Deduce and determine how to apply factor models such as non-negative matrix factorization (NMF), independent component analysis (ICA) and sparse coding
- Deduce and apply correlation functions for various signal classes, in particular for stochastic signals
- Analyze filtering problems and demonstrate the application of least squares filter components such as the Wiener filter
- **Y** Describe, apply and derive non-linear signal processing methods based such as kernel methods and reproducing kernel Hilbert space for applications such as denoising
- Derive maximum likelihood estimates and apply the EM algorithm to learn model parameters
- Describe, apply and derive state-space models such as Kalman filters and Hidden Markov models
- Solve and interpret the result of signal processing systems by use of a programming language
- Design simple signal processing systems based on an analysis of involved signal characteristics, the objective of the processing system, and utility of methods presented in the course
- Describe a number of signal processing applications and interpret the results

Last week review

Linear dynamical system**Linear dynamical system**

$$\mathbf{x}_n = F_n \mathbf{x}_{n-1} + \boldsymbol{\eta}_n, \quad \text{State equation}$$

$$\mathbf{y}_n = H_n \mathbf{x}_n + \mathbf{v}_n, \quad \text{Observation equation}$$

- If \mathbf{x}_n is discrete, we call the model a **Hidden Markov model** (HMM).
- If \mathbf{x}_n is continuous and Gaussian, we call the model a **linear dynamical system** (LDS).
- Additionally, if F_n , H_n , $\boldsymbol{\eta}_n$, and \mathbf{v}_n are known, we call it **Kalman filtering**. Or more precisely, **inference** in a linear dynamical system is called **Kalman filtering**.

HMM model parameters

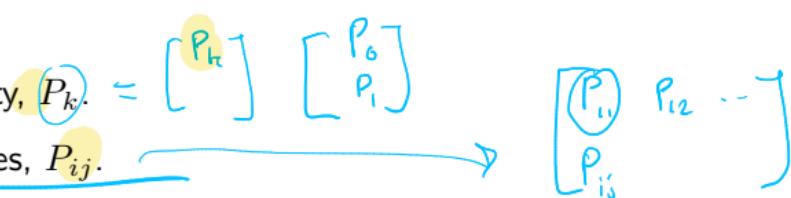
A HMM model is fully described by the following set of parameters:

① Number of states K .

② Initial state probability, $P_k = [P_k]$

③ Transition probabilities, P_{ij} .

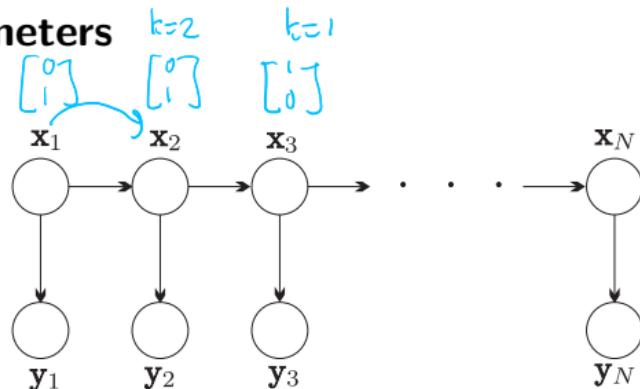
④ State emission distributions $p(y|k)$. whatever



We can ask two different questions:

- Given an observed sequence y_1, \dots, y_n , which HMM, out of a database of HMMs most likely generated the sequence? Example?
- Given an observed sequence y_1, \dots, y_n , which state k are we most likely in, or, what is the predicted value y_{n+1} ?

How to learn the parameters



$$P_k = []$$

$$P_{ij} = []$$



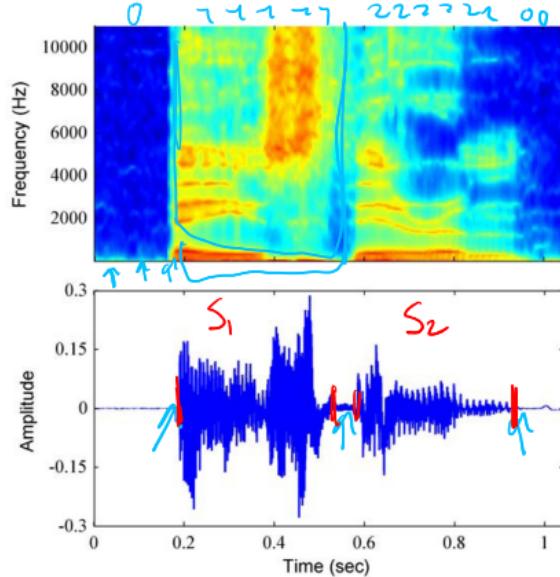
As a reminder, the EM algorithm consist of the following steps:

- ① Specification of the complete log likelihood, $\ln p(\mathcal{X}, \mathcal{X}^l)$ (the model).
- ② Derive $Q(\xi, \xi^{(j)}) = \mathbb{E}[\ln p(\mathcal{X}, \mathcal{X}^l; \xi^{(j)})]$.
- ③ Maximize $Q(\xi, \xi^{(j)})$ in order to get $\xi^{(j+1)}$.

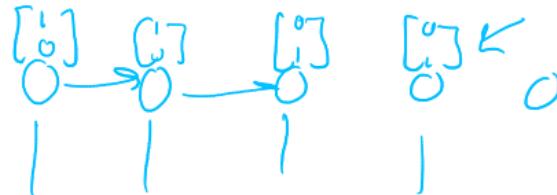
where \mathcal{X} denotes the set of observations, \mathcal{X}^l denotes the set of latent random variables, and ξ is a vector of distribution parameters.

Last week review

HMM on audio classification



b	ey	z	th	ih	er	em
	Bayes'	Theorem				



$$P(HMM_1 | S_i)$$
$$P(HMM_2 | S_i)$$

plug n play ↪



From: Bishop (2006). Pattern Recognition and Machine Learning

A hidden Markov model (HMM) models the situation where you have K distinct states of your system.

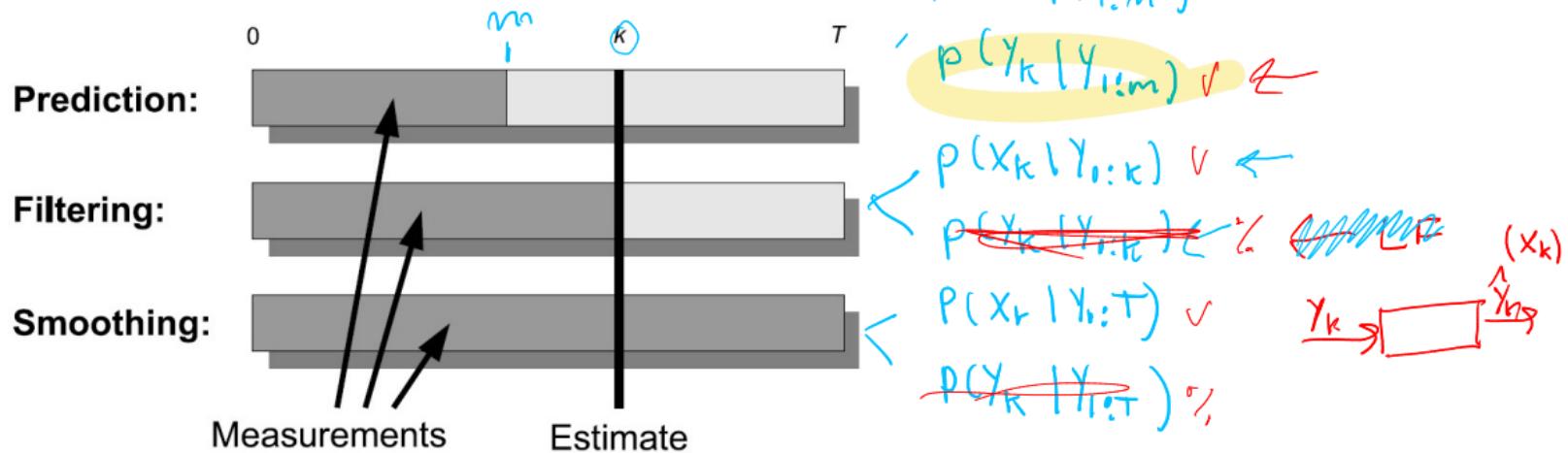
The observations can be discrete or continuous.

Is well suited for a number of applications, e.g. sound classification, classification of bytes in communication etc.

Kalman filtering

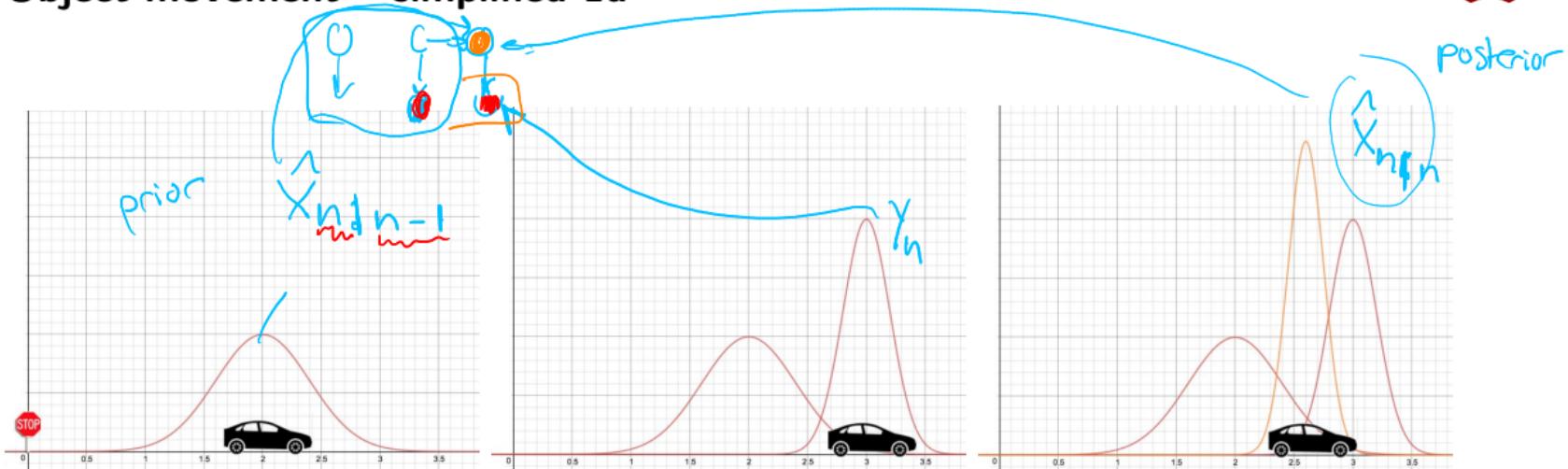
Kalman filtering

Different types of operations



From: Simo Särkkä (2013). Bayesian Filtering and Smoothing. Cambridge University Press

Object movement – simplified 1d



<https://jonathan-hui.medium.com/>

self-driving-object-tracking-intuition-and-the-math-behind-kalman-filter-657d11dd0a90

Kalman filtering

Object movement – simplified 1d



$$\text{Model} \quad X_n = F_{\text{at}} X_{n-1} + u_n \sim \mathcal{N}(0, Q)$$
$$Y_n = H_{\text{at}} X_n + v_n$$

$$Q = \begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix}$$

q Δt 35

State vector encoding

$$x_n = \begin{bmatrix} \text{position}_n \\ \text{velocity}_n \\ \text{acc}_n \end{bmatrix} = \begin{bmatrix} p_n \\ v_n \end{bmatrix}$$

state $X_n \in \mathbb{R}^2$

As difference equations:

$$p_n = p_{n-1} + v_{n-1} \Delta t$$
$$v_n = v_{n-1}$$

In matrix form

$$x_n = \underbrace{\begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}}_F \begin{bmatrix} p_{n-1} \\ v_{n-1} \end{bmatrix}$$

obs $Y_n \in \mathbb{R}^1$

$$\begin{aligned} p_{n-1} &= 2 \\ v_{n-1} &= 1 \\ \Delta t &= 1 \end{aligned} \quad \left| \begin{array}{l} p_n = 3 \end{array} \right.$$

$$F = \begin{bmatrix} 1 & \Delta t \end{bmatrix} \in \mathbb{R}^{2 \times 2}$$

$$H = \begin{bmatrix} 1 & 0 \end{bmatrix} \in \mathbb{R}^{1 \times 2}$$

Kalman filtering

Kalman filtering

Linear dynamical system

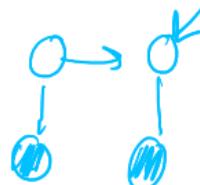
$$\mathbf{x}_n = F_n \mathbf{x}_{n-1} + \boldsymbol{\eta}_n, \quad \text{State equation}$$

$$\mathbf{y}_n = H_n \mathbf{x}_n + \mathbf{v}_n, \quad \text{Observation equation}$$

Kalman filter has two stages; prediction, and update (or correction). For prediction, we seek estimation formulas for:

- $\hat{\mathbf{x}}_{n|n-1}$ (called prior estimator)
- $P_{n|n-1}$ (called prior covariance matrix)

prediction



For update (correction), we seek estimation formulas for

- $\hat{\mathbf{x}}_{n|n}$ (called posterior estimator)
- $P_{n|n}$ (called posterior covariance matrix)

correction

11.1.4

Additionally, we define the following recursion

$$\hat{\mathbf{x}}_{n|n} := \hat{\mathbf{x}}_{n|n-1} + K_n \mathbf{e}_n$$

??

LMS: $\theta_n = \theta_{n-1} + [Lx] e_n$
RLS: $\theta_n = \theta_{n-1} + K_n e_n$

You will derive these expressions in the exercise.

Linear dynamical system

$$\mathbf{x}_n = F_n \mathbf{x}_{n-1} + \boldsymbol{\eta}_n, \quad \text{State equation}$$

$$\mathbf{y}_n = H_n \mathbf{x}_n + \mathbf{v}_n, \quad \text{Observation equation}$$

In Kalman filtering, we make the following assumptions:

- The distribution of the noise terms are known, and have the following properties

- (V) • $\mathbb{E}[\boldsymbol{\eta}_n \boldsymbol{\eta}_n^T] := Q_n$ ✓
• $\mathbb{E}[\boldsymbol{\eta}_n \boldsymbol{\eta}_m^T] = 0, n \neq m$ process noise is i.i.d
(V) • $\mathbb{E}[\boldsymbol{\eta}_n] = \mathbf{0}$ "no drift"
✓ • $\mathbb{E}[\mathbf{v}_n \mathbf{v}_n^T] := R_n$ ✓
✓ • $\mathbb{E}[\mathbf{v}_n \mathbf{v}_m^T] = 0, n \neq m$ meas. noise is i.i.d
✓ • $\mathbb{E}[\mathbf{v}_n] = \mathbf{0}$ $\mathcal{N}(0, \sigma^2)$
→ • $\mathbb{E}[\boldsymbol{\eta}_n \mathbf{v}_m^T] = 0, \forall n, \forall m$ (V) process noise \rightarrow meas. noise

- The matrices F_n , H_n , Q_n , and R_n are known.

What does these assumptions mean? Are they realistic?

Linear dynamical system

$$\mathbf{x}_n = F_n \mathbf{x}_{n-1} + \boldsymbol{\eta}_n, \quad \text{State equation}$$

$$\mathbf{y}_n = H_n \mathbf{x}_n + \mathbf{v}_n, \quad \text{Observation equation}$$

We make the following definitions

$$\hat{\mathbf{x}}_n = \mathbf{x}_{n-1} + K_n e_n$$

MSE

error

$$\left\{ \begin{array}{l} \hat{\mathbf{y}}_n := H_n \hat{\mathbf{x}}_{n|n-1} \\ \mathbf{e}_n := \mathbf{y}_n - \hat{\mathbf{y}}_n \leftarrow \text{known} \\ \mathbf{e}_{n|n} := \mathbf{x}_n - \hat{\mathbf{x}}_{n|n} ? \checkmark \\ \mathbf{e}_{n|n-1} := \mathbf{x}_n - \hat{\mathbf{x}}_{n|n-1} ? \checkmark \\ P_{n|n} := \mathbb{E}[\mathbf{e}_{n|n} \mathbf{e}_{n|n}^T] \\ P_{n|n-1} := \mathbb{E}[\mathbf{e}_{n|n-1} \mathbf{e}_{n|n-1}^T] \end{array} \right.$$

min $J = \mathbb{E}[\mathbf{e}_{n|n}^T \mathbf{e}_{n|n}]$

K_n

II.1.4

IDEA!!

on the States

Kalman filters two stages

Kalman filter has two stages; prediction, and update (or correction). For prediction, we seek estimation formulas for:

- $\hat{x}_{n|n-1}$
- $P_{n|n-1}$

For update (correction), we seek estimation formulas for

- $\hat{x}_{n|n}$
- $P_{n|n}$

You will derive these expressions in the exercise.

Kalman filter equations

Linear dynamical system

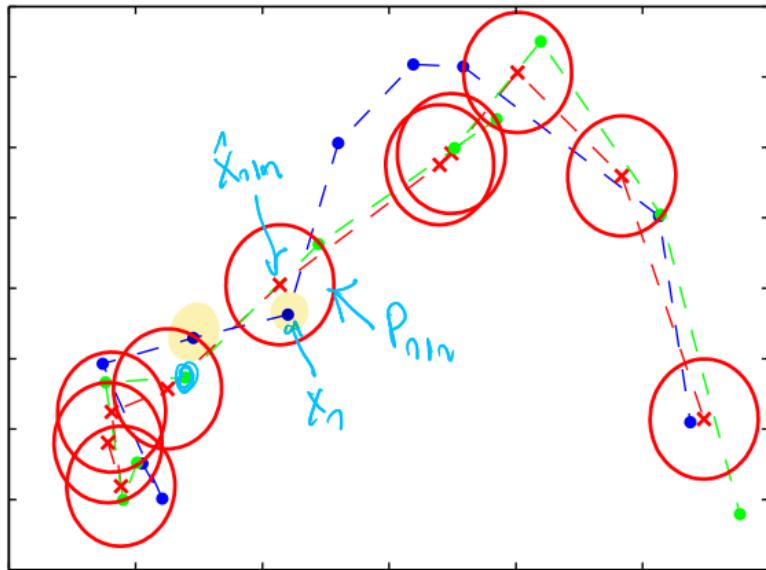
$$\mathbf{x}_n = F_n \mathbf{x}_{n-1} + \boldsymbol{\eta}_n, \quad \text{State equation}$$

$$\mathbf{y}_n = H_n \mathbf{x}_n + \mathbf{v}_n, \quad \text{Observation equation}$$

Kalman filtering

- Initialize
 - $\hat{\mathbf{x}}_{1|0} = \mathbb{E}[\mathbf{x}_1]$
 - $P_{1|0} = \Pi_0$ I
- For $n = 1, 2, \dots$, Do
 - $\mathbf{e}_n = \mathbf{y}_n - H_n \hat{\mathbf{x}}_{n|n-1}$
 - $K_n = P_{n|n-1} H_n^T (R_n + H_n P_{n|n-1} H_n^T)^{-1}$
 - $\hat{\mathbf{x}}_{n|n} = \hat{\mathbf{x}}_{n|n-1} + K_n \mathbf{e}_n$ ||RLS||
 - $P_{n|n} = P_{n|n-1} - K_n H_n P_{n|n-1}$
 - $\hat{\mathbf{x}}_{n+1|n} = F_{n+1} \hat{\mathbf{x}}_{n|n}$
 - $P_{n+1|n} = F_{n+1} P_{n|n} F_{n+1}^T + Q_{n+1}$
- End For

Example of Kalman filter

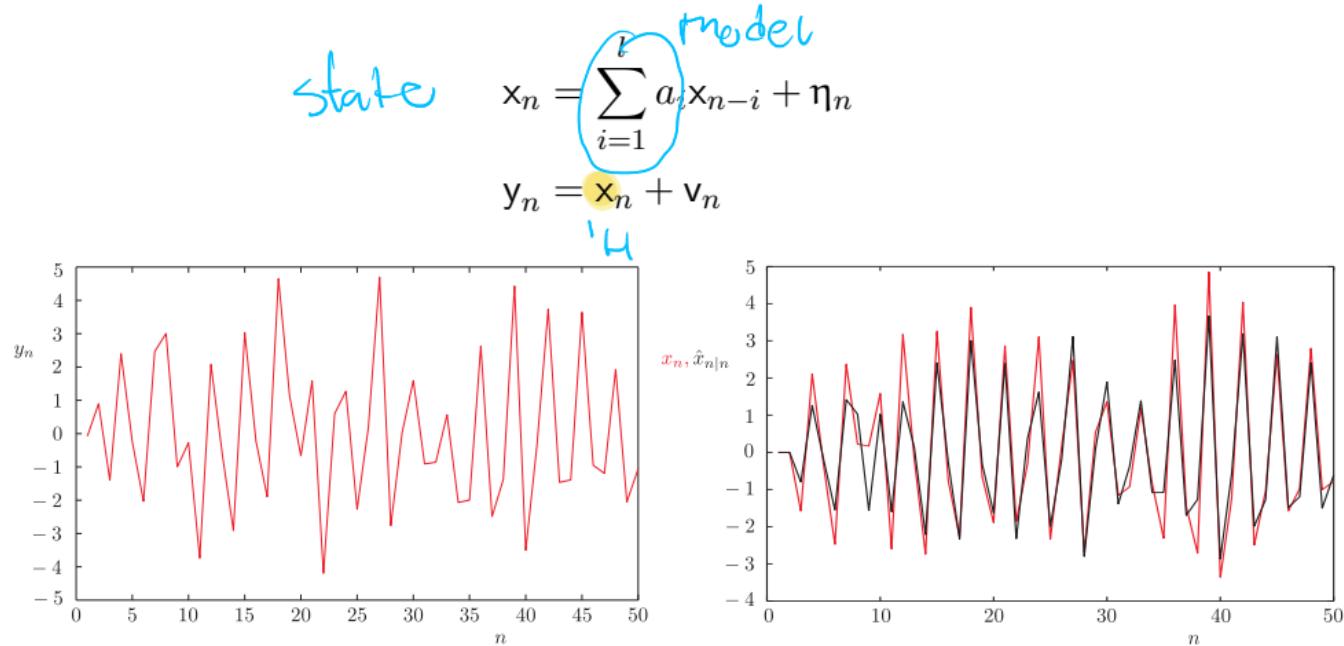


Example: moving object tracking. Green: noisy measurements, blue: true location, red: predicted.

Kalman filtering

Example of AR-process and Kalman filtering

Let us consider the following model for data generation



You will try this system in the exercise.

process

- For \mathbf{x}_n is continuous and Gaussian, we call the model a Linear dynamical system (LDS).
- For Linear dynamical systems, Kalman filtering is the prediction/update formulas.
- Kalman filtering is used heavily in e.g. object tracking, where the "location" is sensed using noisy sensor readouts.

Filtering from a Bayesian viewpoint

The general state-space model

$\mathbf{x}_n = \mathbf{f}_n(\mathbf{x}_{n-1}, \boldsymbol{\eta}_n)$: state equation

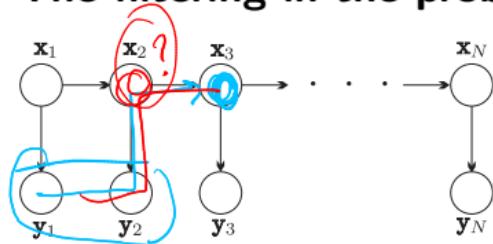
$\mathbf{y}_n = \mathbf{h}_n(\mathbf{x}_n, \mathbf{v}_n)$: observations equation

Filtering: $p(\mathbf{x}_n | \mathbf{y}_{1:n})$

Smoothing: $p(\mathbf{x}_n | \mathbf{y}_{1:N})$, $1 \leq n \leq N$

Filtering from a Bayesian viewpoint

The filtering in the probabilistic setting



model

$$\mathbf{x}_n = F_n \mathbf{x}_{n-1} + \eta_n$$
$$\mathbf{y}_n = H_n \mathbf{x}_n + v_n$$

Kalman filter

noise

$$p(\eta_n) = \mathcal{N}(\eta_n; \mathbf{0}, Q_n)$$
$$p(v_n) = \mathcal{N}(v_n; \mathbf{0}, R_n)$$

$$p(\mathbf{x}_n | \mathbf{y}_{1:n}) = \frac{p(\mathbf{y}_n | \mathbf{x}_n) p(\mathbf{x}_n | \mathbf{y}_{1:n-1})}{p(\mathbf{y}_n | \mathbf{y}_{1:n-1})}$$

Bayes

$$p(\mathbf{x}_n | \mathbf{y}_{1:n-1}) = \int p(\mathbf{x}_n | \mathbf{x}_{n-1}) p(\mathbf{x}_{n-1} | \mathbf{y}_{1:n-1}) d\mathbf{x}_{n-1}$$

- For linear dynamical systems, Kalman filtering is the prediction/update formulas.
- Kalman filtering requires specification of model parameters.
- Is used heavily in e.g. object tracking, where the "location" is sensed using noisy sensor readouts.
- What I didn't tell you? (only if you are curious)
 - How do I train the parameters in LDS - use EM (Bishop 13.2).
 - How do I perform smoothing? (Bishop 13.2).
 - What if I have a non-linear model? Use e.g. particle filtering. (ML book 17.2+17.4).

Next week

Next week



Week 47 material; 11.5–11.7

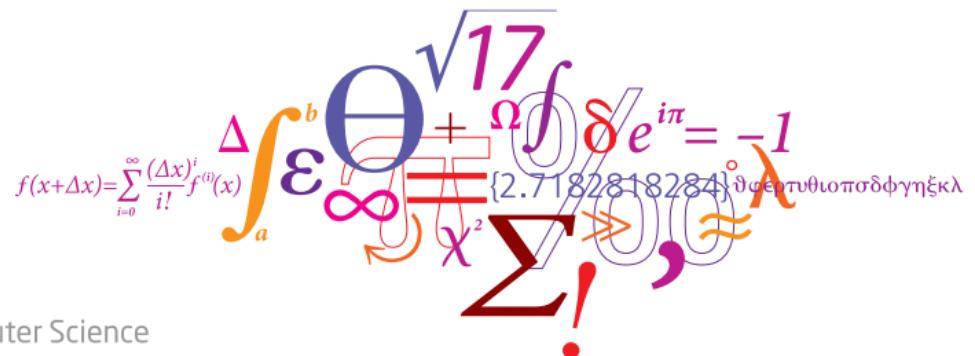
- Kernel methods.
- Kernel ridge regression.

02471 Machine Learning for Signal Processing

Kernel methods and kernel ridge regression

Tommy Sonne Alstrøm

Cognitive Systems section



Outline

- Last week review
- Non-linear modeling
 - Anomaly detection and de-noising
 - Using general linear models
- Kernel machines
- Examples of kernels
- Kernel algorithms
 - Kernel ridge regression
- Next week

Material: 11.5–11.7 (skip the proof for Theorem 11.2, 11.6.1–11.6.2).

Feedback

- Please remember to fill in the end-of-course evaluation: <https://evaluering.dtu.dk>
- There is a lot of great feedback, thank you very much!
- Most predominant feedback (things to improve) from evaluation:
 - More examples during the lecture, e.g. include examples similar to the problem sets.
 - It should be more clear from the course database, that this is a very mathematical-oriented course and not so much a "machine learning" course (misleading to call this ML course).
 - too much focus on theory and proofs. It would be nicer with some hands-on work.
 - the book becomes harder to understand (later) ... the content needs to be condensed in written form (which is what the corresponding exercise does)

Course outline

What you have learned so far:

- Parameter estimation [L2 regularization, biased estimation, mean squared error minimization]. L1 regularization, Bayesian parameter estimation.
- Filtering signals [Stochastic processes, correlation functions, Wiener filter, linear prediction, adaptive filtering using stochastic gradient decent (LMS, APA/NLMS), adaptive filtering using regularization (RLS)]
- Signal representations [Time frequency analysis with STFT], Sparsity aware sensing (lasso, sparse priors), factor models [Independent component analysis, Non-negative matrix factorization, k -SVD],
- Bayesian parameter estimation and probabilistic graphical models, Kalman filtering. Inference and EM.

Next two weeks:

- Kernel methods: Today: non-linear models, kernels, kernel Ridge regression, support vector regression.

Learning objectives

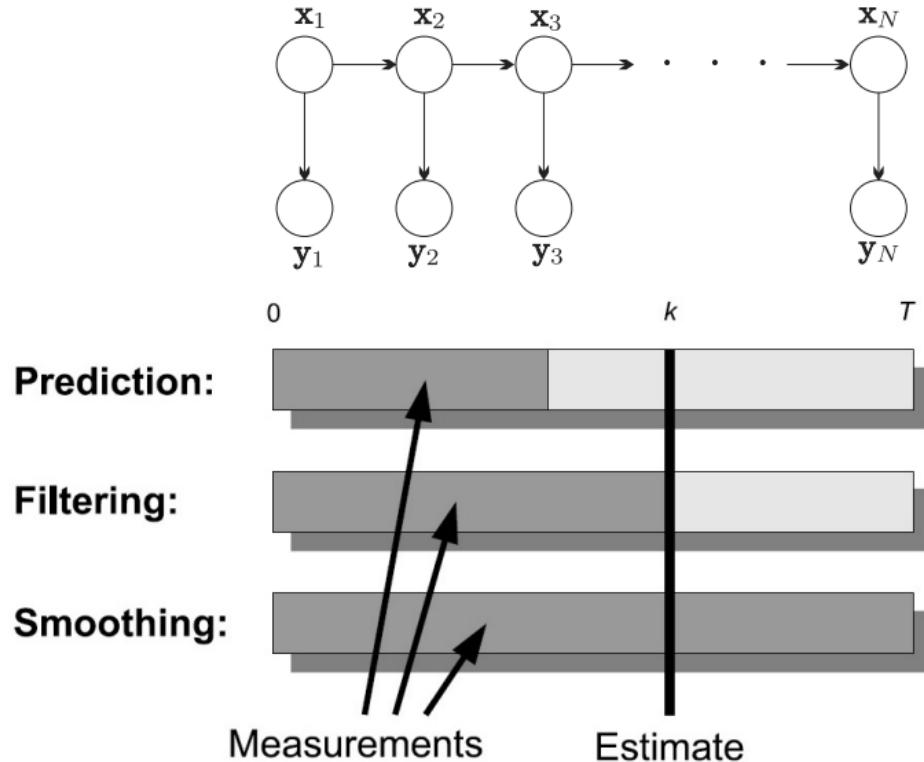
Learning objectives

A student who has met the objectives of the course will be able to:

- Explain, apply and analyze properties of discrete time signal processing systems
- Apply the short time Fourier transform to compute the spectrogram of a signal and analyze the signal content
- Explain compressed sensing and determine the relevant parameters in specific applications
- Deduce and determine how to apply factor models such as non-negative matrix factorization (NMF), independent component analysis (ICA) and sparse coding
- Deduce and apply correlation functions for various signal classes, in particular for stochastic signals
- Analyze filtering problems and demonstrate the application of least squares filter components such as the Wiener filter
- Describe, apply and derive non-linear signal processing methods based such as kernel methods and reproducing kernel Hilbert space for applications such as denoising
- Derive maximum likelihood estimates and apply the EM algorithm to learn model parameters
- Describe, apply and derive state-space models such as Kalman filters and Hidden Markov models
- Solve and interpret the result of signal processing systems by use of a programming language
- Design simple signal processing systems based on an analysis of involved signal characteristics, the objective of the processing system, and utility of methods presented in the course
- Describe a number of signal processing applications and interpret the results

Last week review

State-space model and types of operations



Linear dynamical system

$$\mathbf{x}_n = F_n \mathbf{x}_{n-1} + \boldsymbol{\eta}_n, \quad \text{State equation}$$

$$\mathbf{y}_n = H_n \mathbf{x}_n + \mathbf{v}_n, \quad \text{Observation equation}$$

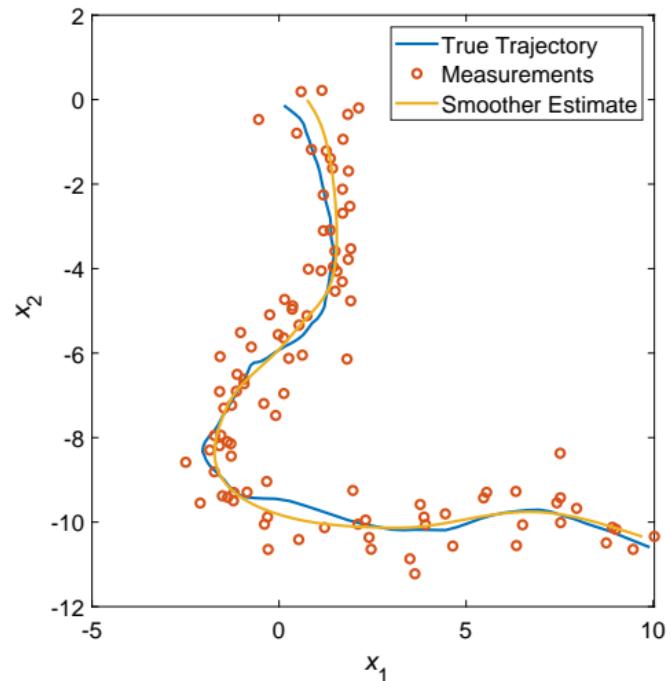
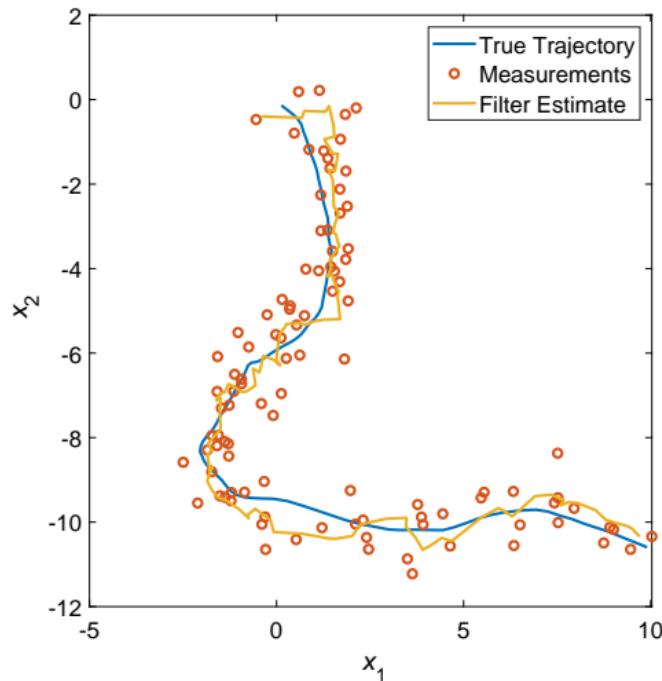
Kalman filter has two stages; prediction, and update (or correction). For prediction, we seek estimation formulas for:

- $\hat{\mathbf{x}}_{n|n-1}$ (called prior estimator)
- $P_{n|n-1}$ (called prior covariance matrix)

For update (correction), we seek estimation formulas for

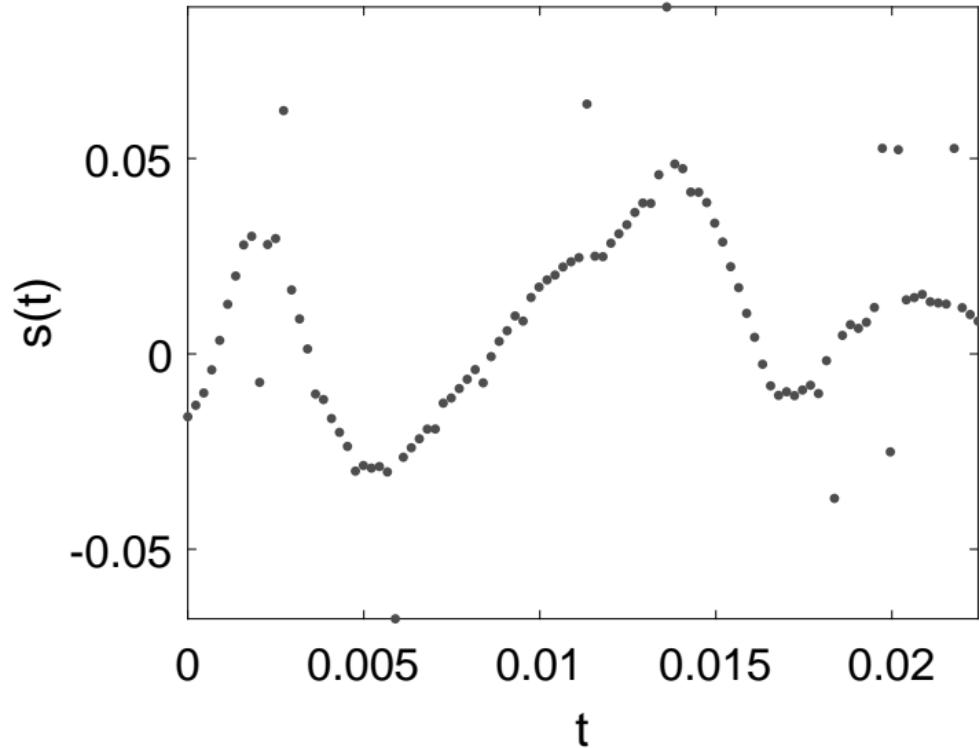
- $\hat{\mathbf{x}}_{n|n}$ (called posterior estimator)
- $P_{n|n}$ (called posterior covariance matrix)

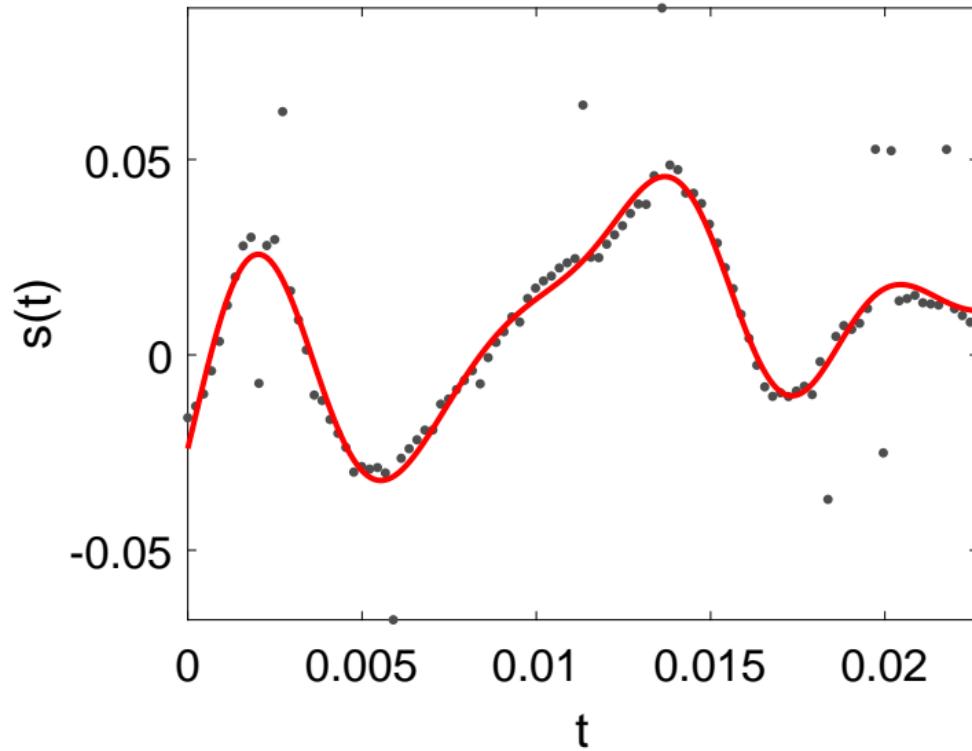
Filtering and smoothing



- For linear dynamical systems, Kalman filtering is the prediction/update formulas.
- Kalman filtering requires specification of model parameters.
- Is used heavily in e.g. object tracking, where the "location" is sensed using noisy sensor readouts.

Non-linear modeling

Example: outliers present

Example: one solution

General linear models

$$y = f(\mathbf{x}, \boldsymbol{\theta}) := \theta_0 + \sum_{k=1}^K \theta_k \phi_k(\mathbf{x})$$

$\phi_k(\mathbf{x})$ is any function that maps $\mathbf{x} \in \mathbb{R}^l$, $\phi_k : \mathbb{R}^l \rightarrow \mathbb{R}$

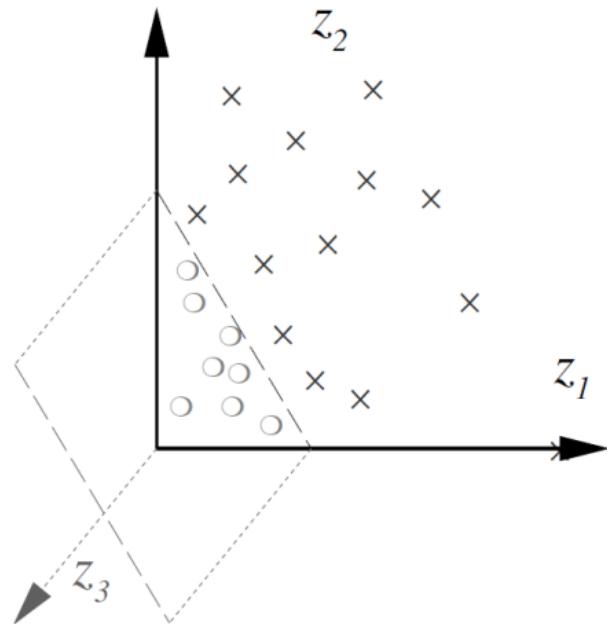
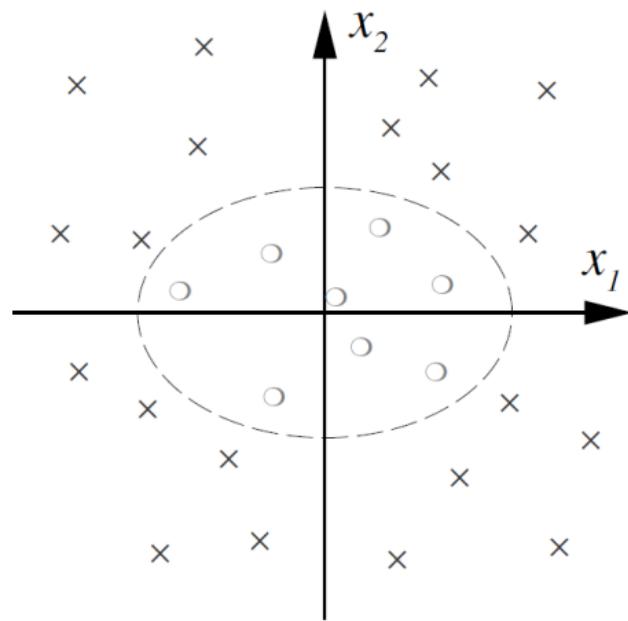
Example

$$\phi(\mathbf{x}) : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$\phi(\mathbf{x}) = \begin{bmatrix} x_1^2 & \sqrt{2}x_1x_2 & x_2^2 \end{bmatrix}^T$$

What kind of data is this useful for?

This particular mapping leads to linear estimation



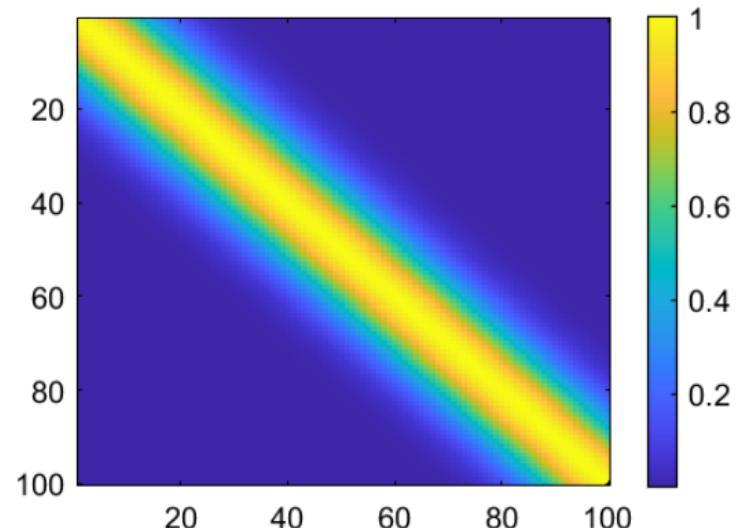
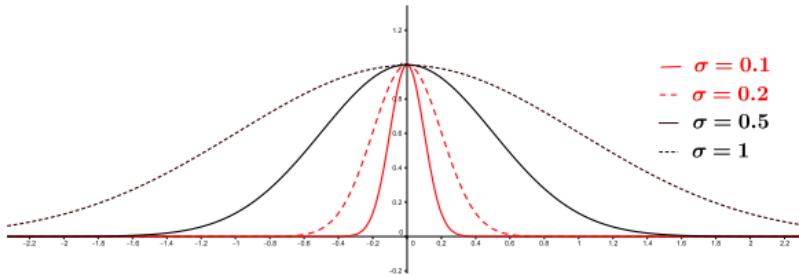
Source: Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond — 2002,
by Schölkopf, Bernhard; Smola, Alexander J.

Kernel machines

Kernel machines

Example: Gaussian kernel

$$\kappa(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{y}\|^2\right)$$



Reproducing kernel Hilbert space (RKHS)

A space with the following properties

- ① A well defined norm, $\|x\|$, that satisfies the usual norm properties (sec 9.2) (normed vector space).
- ② Is complete – which loosely speaking, behaves nicely and all elements exist, that is, every convergent series has smaller and smaller elements (Banach space).
- ③ An inner product, denoted as $\langle \cdot, \cdot \rangle$, e.g. in \mathbb{R}^N , we have $\langle x, y \rangle = x^T y = x \cdot y$. Additionally, the norm satisfies $\|x\| = \sqrt{\langle x, x \rangle}$, e.g. in \mathbb{R}^N we have $\|x\| = \sqrt{x^T x}$ (Hilbert space, denoted \mathbb{H}).
- ④ Has a special function, called the kernel, $\kappa(\cdot, x) \in \mathbb{H}$, with the reproducing property, which essentially means $\kappa(\cdot, x)$ is bounded for bounded input (RKHS).

Criteria 1–3 are treated extensively in:

- 01125 Fundamental topological concepts and metric spaces.
- 01325 Function spaces and mathematical analysis.

Which is not part of the listed prerequisites.

Some nice properties of functions in RKHS

Kernel properties, assuming $\kappa(\cdot, \mathbf{x}) \in \mathbb{H}$, and has the reproducing property

$$\langle \kappa(\cdot, \mathbf{x}), \kappa(\cdot, \mathbf{y}) \rangle = \kappa(\mathbf{x}, \mathbf{y}) = \kappa(\mathbf{y}, \mathbf{x})$$

Or written differently, if $\phi(\mathbf{x}) := \kappa(\cdot, \mathbf{x})$, then

$$\langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle = \kappa(\mathbf{x}, \mathbf{y}), \quad \text{Kernel Trick}$$

$$\mathcal{K} = \begin{bmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1) & \cdots & \kappa(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \vdots & \vdots \\ \kappa(\mathbf{x}_N, \mathbf{x}_1) & \cdots & \kappa(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}$$

$$\mathcal{K} = \mathcal{K}^T$$

$$\mathbf{a} \mathcal{K} \mathbf{a} \geq 0, \quad \mathbf{a} \in \mathbb{R}^l, \quad \mathcal{K} \text{ is positive semi-definite}$$

Expectations from you: can use the properties, but not show the properties!

Representer theorem

Let

$$\Omega : [0, \infty) \rightarrow \mathbb{R}$$

be an arbitrary strictly monotonic increasing function. Let also

$$\mathcal{L} : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R} \cup \{\infty\}$$

be an arbitrary loss function. Then each minimizer, $f \in \mathbb{H}$, of the regularized minimization task

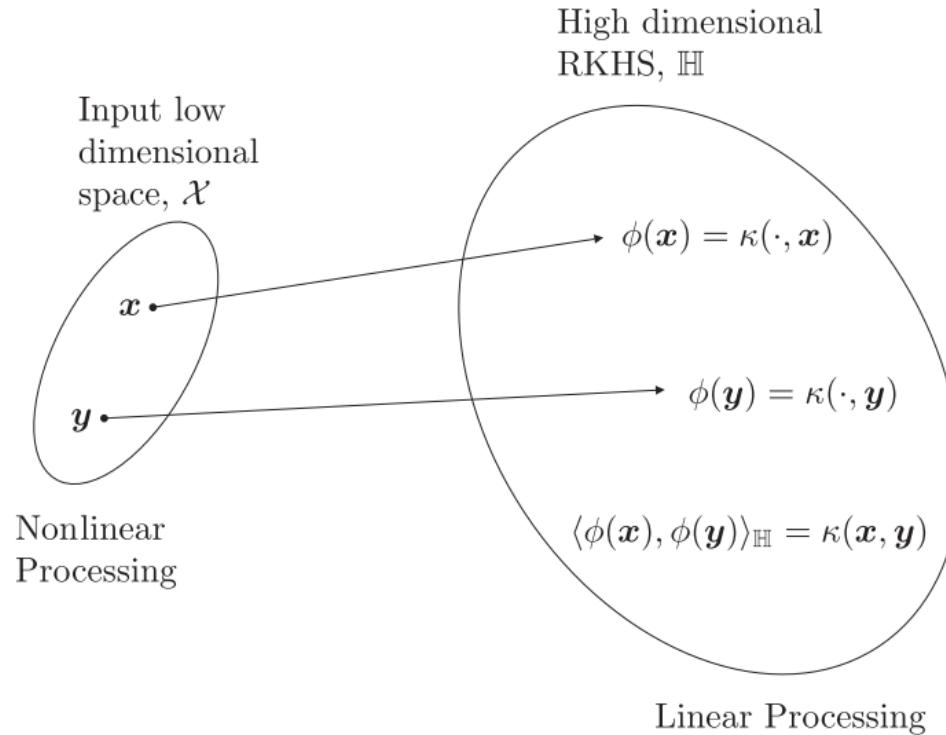
$$\hat{f}(\cdot) = \arg \min_{f \in \mathbb{H}} J(f) := \sum_{n=1}^N \mathcal{L}(y_n, f(\mathbf{x}_n)) + \lambda \Omega(\|f\|^2)$$

admits a representation of the form,

$$\hat{f}(\cdot) = \sum_{n=1}^N \theta_n \kappa(\cdot, \mathbf{x}_n)$$

where $\theta_n \in \mathbb{R}, n = 1, 2, \dots, N$

Consequences – linear processing



Consequences – algorithms can generally be kernelized

- ① Map (implicitly) the input training data to an RKHS.
- ② Solve the linear estimation task in \mathbb{H} .
- ③ Cast the algorithm in terms of inner products $\langle \mathbf{x}, \mathbf{y} \rangle$ (\mathbb{R}^l is a Hilbert space).
- ④ Replace each inner product by a kernel evaluation, that is, $\langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle = \kappa(\mathbf{x}, \mathbf{y})$.

Simply example:

$$\phi(\mathbf{x}) = [x_1^2 \quad \sqrt{2}x_1x_2 \quad x_2^2]^T$$

The corresponding kernel would be

$$\kappa(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y})^2, \quad \text{Polynomial kernel}$$

Can you show this? what trick do we need?

- Reproducing kernel Hilbert space enables linear processing while obtaining nonlinear decision boundaries.
- If you limit yourself to already proven reproducing kernels, you do **not** as such need to understand the theory behind RKHS but can readily apply it.
- Choice of kernel and kernel parameters is critical for performance.

Examples of kernels

Examples of kernels

Example of kernels



The Gaussian (or exponential, or rbf) kernel (be aware that sometimes $\gamma = \frac{1}{2\sigma^2}$):

$$\kappa(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{y}\|^2\right)$$

The inhomogeneous polynomial kernel:

$$\kappa(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + c)^r$$

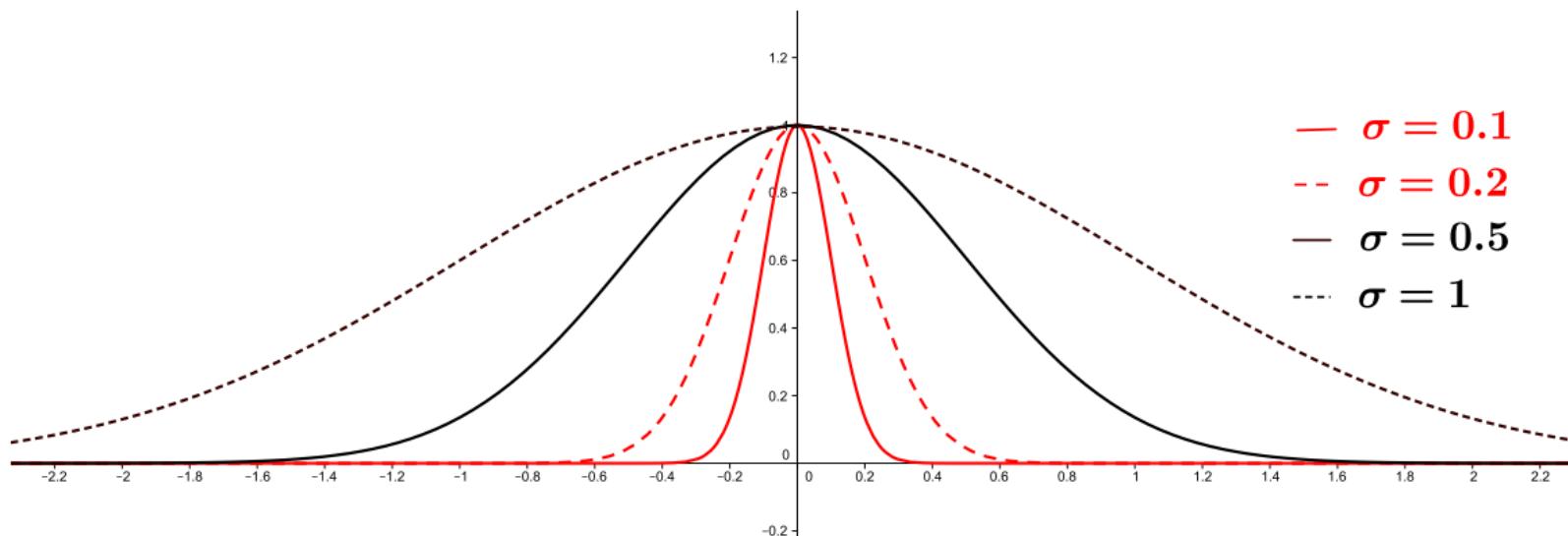
The Laplacian kernel:

$$\kappa(\mathbf{x}, \mathbf{y}) = \exp(-t\|\mathbf{x} - \mathbf{y}\|)$$

Examples of kernels

Example: Gaussian kernel

$$\kappa(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{y}\|^2\right)$$



Examples of kernels

Creating new kernels

Techniques for Constructing New Kernels.

Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following new kernels will also be valid:

$$k(\mathbf{x}, \mathbf{x}') = ck_1(\mathbf{x}, \mathbf{x}') \quad (6.13)$$

$$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}') \quad (6.14)$$

$$k(\mathbf{x}, \mathbf{x}') = q(k_1(\mathbf{x}, \mathbf{x}')) \quad (6.15)$$

$$k(\mathbf{x}, \mathbf{x}') = \exp(k_1(\mathbf{x}, \mathbf{x}')) \quad (6.16)$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \quad (6.17)$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}') \quad (6.18)$$

$$k(\mathbf{x}, \mathbf{x}') = k_3(\phi(\mathbf{x}), \phi(\mathbf{x}')) \quad (6.19)$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{A} \mathbf{x}' \quad (6.20)$$

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b) \quad (6.21)$$

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a)k_b(\mathbf{x}_b, \mathbf{x}'_b) \quad (6.22)$$

where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\phi(\mathbf{x})$ is a function from \mathbf{x} to \mathbb{R}^M , $k_3(\cdot, \cdot)$ is a valid kernel in \mathbb{R}^M , \mathbf{A} is a symmetric positive semidefinite matrix, \mathbf{x}_a and \mathbf{x}_b are variables (not necessarily disjoint) with $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, and k_a and k_b are valid kernel functions over their respective spaces.

Source: Pattern Recognition and Machine Learning, 2006, C. Bishop

Kernel algorithms

Example – Kernel ridge regression

Kernel Ridge regression (without bias)

Assume the regression task (η_n is white noise)

$$y_n = g(\mathbf{x}_n) + \eta_n, \quad n = 1, 2, \dots, N$$

Assume the solution (according to representer theorem)

$$f(\cdot) = \sum_{m=1}^N \theta_m \kappa(\cdot, \mathbf{x}_m)$$

The model, where $C \in \mathbb{R}$ is the regularization parameter, is then:

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$J(\boldsymbol{\theta}) := \sum_{n=1}^N \left(y_n - \sum_{m=1}^N \theta_m \kappa(\mathbf{x}_n, \mathbf{x}_m) \right)^2 + C \langle f, f \rangle$$

$$\hat{\boldsymbol{\theta}} = (\mathcal{K} + CI)^{-1} \mathbf{y}$$

Ridge regression vs kernel ridge regression

Recall from section 3.8 that the ridge regression (without bias) minimizes the following function:

$$J_{RR}(\boldsymbol{\theta}) := \sum_{n=1}^N \left(y_n - \sum_{i=1}^l \theta_i x_{ni} \right)^2 + \lambda \sum_{i=1}^l |\theta_i|^2$$

The kernel ridge regression (without bias) instead minimizes

$$J_{KRR}(\boldsymbol{\theta}) := \sum_{n=1}^N \left(y_n - \sum_{m=1}^N \theta_m \kappa(\mathbf{x}_n, \mathbf{x}_m) \right)^2 + C \langle f, f \rangle$$

where $C \in \mathbb{R}$ is a regularization parameter.

Derivation of kernel ridge regression

From representer theorem

$$\begin{aligned}\hat{f}(\cdot) &= \arg \min_{f \in \mathbb{H}} J(f) := \sum_{n=1}^N \mathcal{L}(y_n, f(\mathbf{x}_n)) + \lambda \Omega(\|f\|^2) \\ \hat{f}(\cdot) &= \sum_{n=1}^N \theta_n \kappa(\cdot, \mathbf{x}_n)\end{aligned}$$

We can with $f(\mathbf{x}) = \sum_{m=1}^N \theta_m \kappa(\mathbf{x}, \mathbf{x}_m)$, and a squared loss, arrive at the kernel ridge regression cost function:

$$J(\boldsymbol{\theta}) := \sum_{n=1}^N \left(y_n - \sum_{m=1}^N \theta_m \kappa(\mathbf{x}_n, \mathbf{x}_m) \right)^2 + C \langle \mathbf{f}, \mathbf{f} \rangle$$

- Reproducing kernel Hilbert space enables linear processing while obtaining nonlinear signal processing.
- If you limit yourself to apply already proven reproducing kernels, you do **not** need to understand the theory behind RKHS to apply kernel methods. Use list of kernels.
- Choice of kernel and kernel parameters is critical for performance.
- No restrictions on x , only on the kernels.
- Requires tuning of parameters, but is not “that” sensitive.
- The kernel trick is widely used, also for e.g. kernel LMS, kernel-k-means etc.

Next week

Material: 11.8.

- Exam preparation and general Q/A (maximum 45 minutes)
- Lecture (will start latest at 14.00, but if talk about exam is shorter we may start earlier):
 - Anomaly detection using Huber loss
 - Support vector regression.

02471 Machine Learning for Signal Processing

Kernel methods – Support vector regression

Tommy Sonne Alstrøm

Cognitive Systems section

$$f(x+\Delta x) = \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x)$$

Δ ∫^b_a Θ^{√17} + Ω ∫ δ e^{iπ} = -1

Σ! ≈ λ {2.7182818284} θ φ ε π μ σ δ φ γ η κ λ

∞ χ² > 000 ,

Outline

- Course admin
- Last week review
- Anomaly detection and de-noising
- Support vector regression

Material: 11.8.

So far:

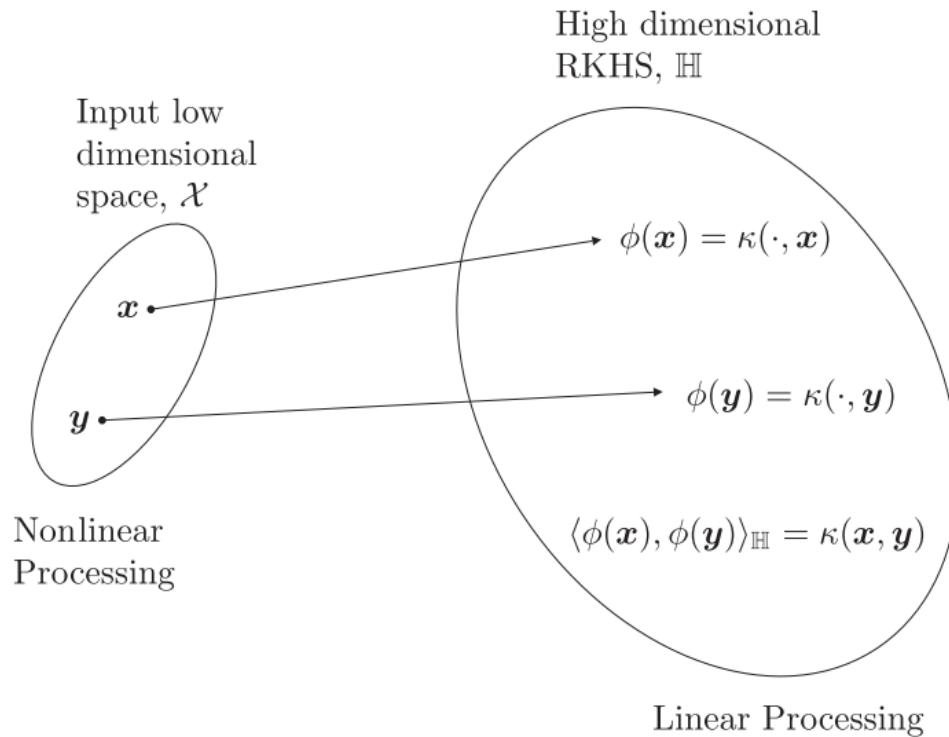
- Parameter estimation [Regularization, biased estimation, mean squared error minimization].
- Signal representations [Time frequency analysis, sparsity aware learning, factor models].
- Filtering signals [Linear regression, adaptive filtering using stochastic gradient decent (LMS, NLMS, RLS), adaptive filtering using regularization].
- Sequential models [hidden markov models, linear dynamical systems, kalman filter].

Next weeks

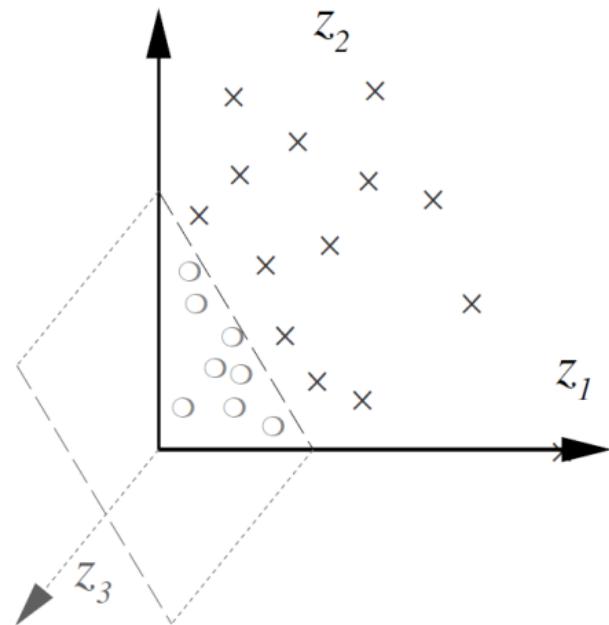
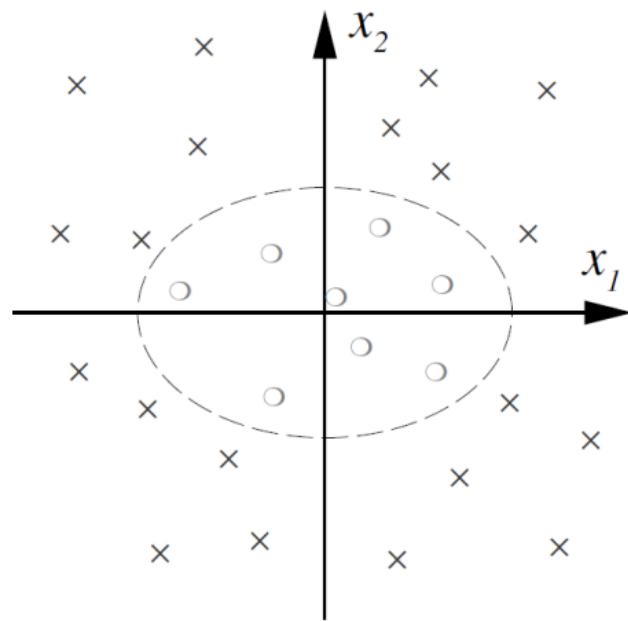
- Kernel methods [kernel ridge regression, **support vector regression**].

Last week review

Consequences – linear processing



This particular mapping leads to linear estimation



Source: Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond — 2002,
by Schölkopf, Bernhard; Smola, Alexander J.

Algorithms can generally be kernelized, if they can be expressed as inner products

- ① Map (implicitly) the input training data to an RKHS.
- ② Solve the linear estimation task in \mathbb{H} .
- ③ Cast the algorithm in terms of inner products $\langle \mathbf{x}_n, \mathbf{x}_m \rangle$ (\mathbb{R}^l is a Hilbert space).
- ④ Replace each inner product by a kernel evaluation, that is, $\langle \phi(\mathbf{x}_n), \phi(\mathbf{x}_m) \rangle = \kappa(\mathbf{x}_n, \mathbf{x}_m)$.

Simply example:

$$\phi(\mathbf{x}) = \begin{bmatrix} x_1^2 & \sqrt{2}x_1x_2 & x_2^2 \end{bmatrix}^T$$

The corresponding kernel would be

$$\kappa(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y})^2, \quad \text{Polynomial kernel}$$

Some kernel properties

Representer theorem in a nutshell

Linearity comes from representer theorem, stating a solution can (under certain circumstances) be written as

$$\hat{f}(\cdot) = \sum_{n=1}^N \theta_n \kappa(\cdot, \mathbf{x}_n)$$

Kernel properties, assuming $\kappa(\cdot, x) \in \mathbb{H}$, and has the reproducing property

$$\langle \kappa(\cdot, \mathbf{x}), \kappa(\cdot, \mathbf{y}) \rangle = \kappa(\mathbf{x}, \mathbf{y}) = \kappa(\mathbf{y}, \mathbf{x})$$

$$\langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle = \kappa(\mathbf{x}, \mathbf{y}), \quad \text{Kernel Trick}$$

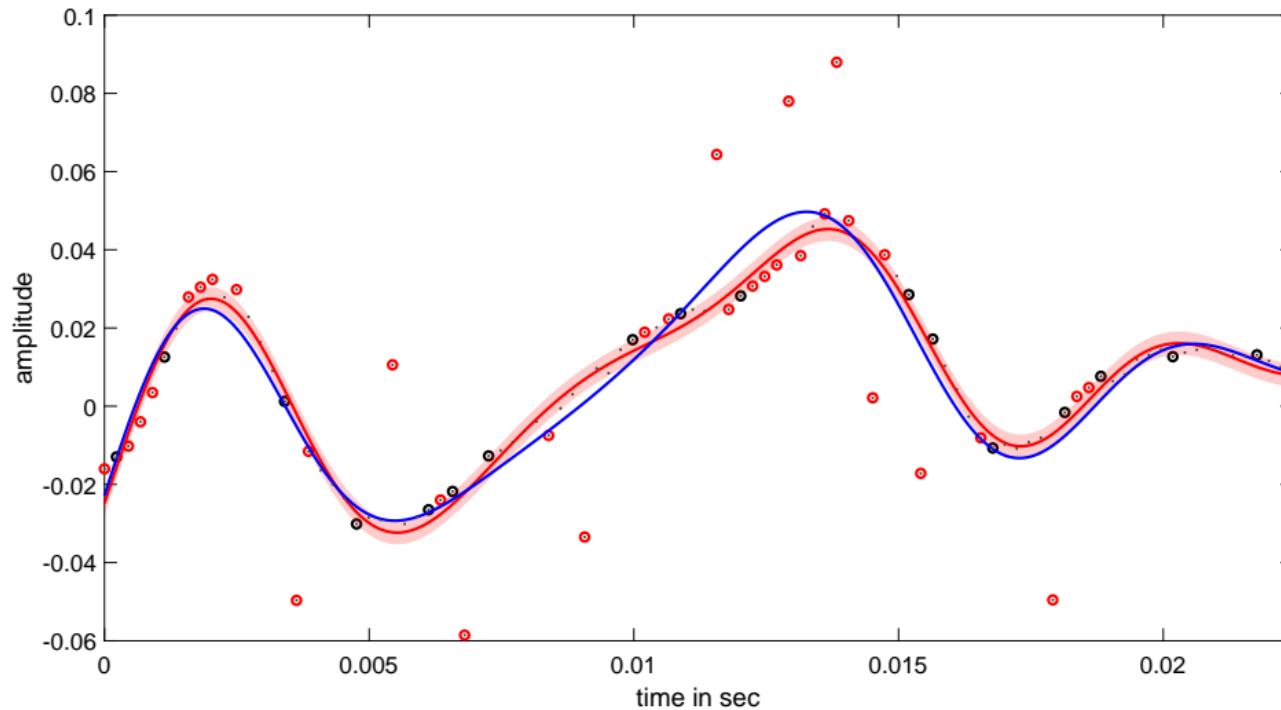
$$\mathcal{K} = \begin{bmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1) & \cdots & \kappa(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \vdots & \vdots \\ \kappa(\mathbf{x}_N, \mathbf{x}_1) & \cdots & \kappa(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}$$

$$\mathcal{K} = \mathcal{K}^T$$

$$\mathbf{a} \mathcal{K} \mathbf{a} \geq 0, \quad \mathbf{a} \in \mathbb{R}^l, \quad \mathcal{K} \text{ is positive semi-definite}$$

- Reproducing kernel Hilbert space enables linear processing while obtaining nonlinear signal processing.
- If you limit yourself to apply already proven reproducing kernels, you do **not** need to understand the theory behind RKHS to apply kernel methods. Use list of kernels.
- Choice of kernel and kernel parameters is critical for performance.
- No restrictions on x , only on the kernels.
- Requires tuning of parameters, but is not “that” sensitive.

Results - support vector regression



Anomaly detection and de-noising

One solution – the Huber loss

Assume the regression task (η_n is i.i.d. noise)

$$y_n = g(\mathbf{x}_n) + \eta_n, \quad n = 1, 2, \dots, N$$

Huber showed that, assuming the noise follows a symmetric pdf, the optimal loss is

Huber loss

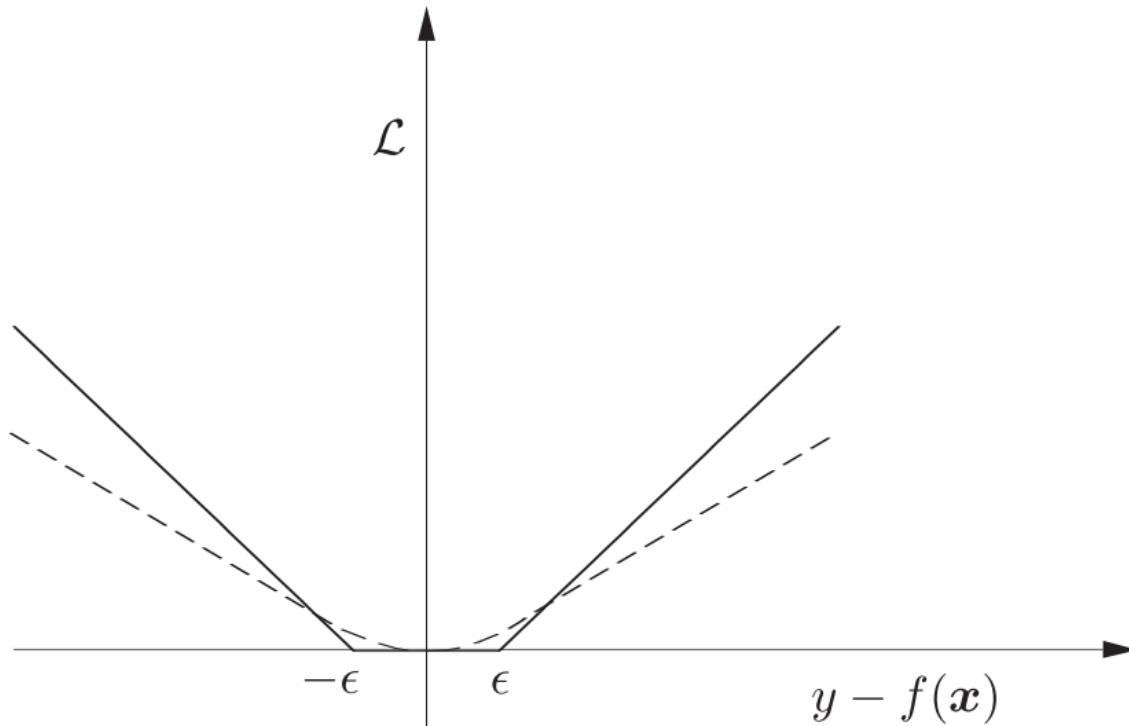
$$\mathcal{L}(y, f(\mathbf{x})) = \begin{cases} \epsilon|y - f(\mathbf{x})| - \frac{\epsilon^2}{2}, & \text{if } |y - f(\mathbf{x})| > \epsilon \\ \frac{1}{2}|y - f(\mathbf{x})|^2, & \text{if } |y - f(\mathbf{x})| \leq \epsilon \end{cases}$$

An alternative, and more computationally efficient loss is the ϵ -insensitive loss

ϵ -insensitive loss

$$\mathcal{L}(y, f(\mathbf{x})) = \begin{cases} |y - f(\mathbf{x})| - \epsilon, & \text{if } |y - f(\mathbf{x})| > \epsilon \\ 0, & \text{if } |y - f(\mathbf{x})| \leq \epsilon \end{cases}$$

Anomaly detection and de-noising
The loss functions



Support vector regression

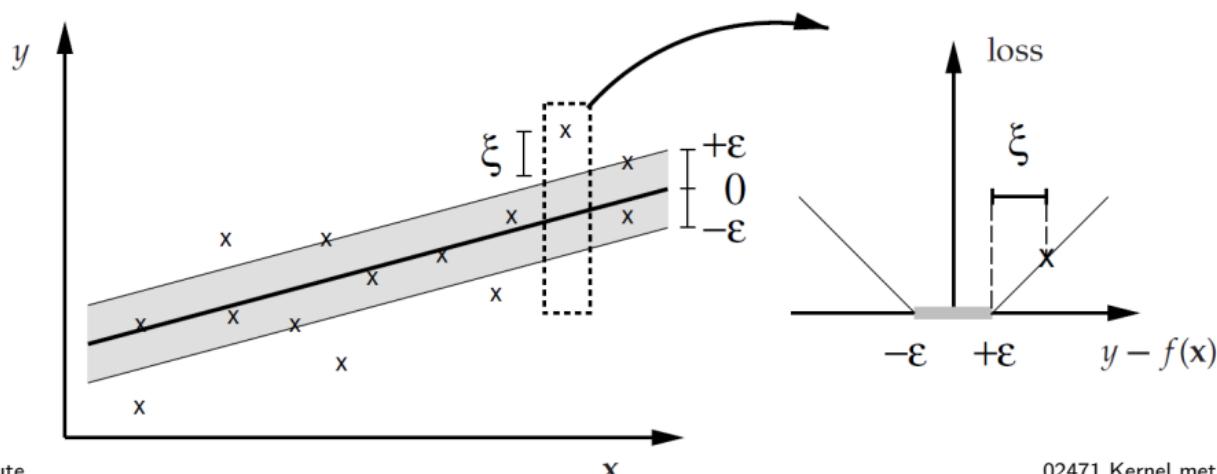
Support vector regression

Introduce slack variables

This loss leads to support vector regression through a couple of steps.

First, rewrite the loss by introducing slack variables ξ_n and $\tilde{\xi}_n$:

$$\begin{aligned}y_n - f(\mathbf{x}_n) &\leq \epsilon + \xi_n \\-(y_n - f(\mathbf{x}_n)) &\leq \epsilon + \tilde{\xi}_n\end{aligned}$$



Regularized minimization of slack variables

$$\arg \min_{\theta, \xi, \tilde{\xi}} J(\theta, \xi, \tilde{\xi}) := \frac{1}{2} \|\theta\|^2 + C \left(\sum_{n=1}^N \xi_n + \sum_{n=1}^N \tilde{\xi}_n \right)$$

$$\text{s.t.} \quad y_n - f(\mathbf{x}_n) \leq \epsilon + \xi_n$$

$$-(y_n - f(\mathbf{x}_n)) \leq \epsilon + \tilde{\xi}_n$$

$$\xi_n \geq 0$$

$$\tilde{\xi}_n \geq 0$$

Next step ??

Support vector regression Lagrarian multipliers

From appendix C.2. If we have the problem:

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & J(\boldsymbol{\theta}) \\ \text{s.t.} \quad & f_i(\boldsymbol{\theta}) \geq 0, \quad i = 1, 2, \dots, m \end{aligned}$$

then the Lagrangian is

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) = J(\boldsymbol{\theta}) - \sum_{i=1}^m \lambda_i f_i(\boldsymbol{\theta})$$

And is solved by:

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_*} &= \mathbf{0} \\ \lambda_i \geq 0 & \quad i = 1, 2, \dots, m \\ \lambda_i f_i(\boldsymbol{\theta}_*) &= 0 \quad i = 1, 2, \dots, m \end{aligned}$$

Support vector regression The solution



Assume $f(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x} + \theta_0$, this leads to the solution

$$\hat{\boldsymbol{\theta}} = \sum_{n=1}^N (\tilde{\lambda}_n - \lambda_n) \mathbf{x}_n$$

which leads to the solution

$$\hat{y} = f(\mathbf{x}) = \sum_{n=1}^N (\tilde{\lambda}_n - \lambda_n) \mathbf{x}_n^T \mathbf{x} + \hat{\theta}_0$$

This is an unusual solution! Can we leverage on this?

Do you see any weakness in this solution?

Support vector regression

Remember our kernelization recipe

Step 1–4 basically:

“Replace each inner product by a kernel evaluation, that is, $\langle \mathbf{x}_n, \mathbf{x}_m \rangle = \kappa(\mathbf{x}_n, \mathbf{x}_m)$ ”

The prediction equation was

$$\hat{y} = f(\mathbf{x}) = \sum_{n=1}^N (\tilde{\lambda}_n - \lambda_n) \mathbf{x}_n^T \mathbf{x} + \hat{\theta}_0$$

The kernel trick results in

Support vector regression prediction

$$\hat{y} = \sum_{n=1}^N (\tilde{\lambda}_n - \lambda_n) \kappa(\mathbf{x}_n, \mathbf{x}) + \hat{\theta}_0$$

The points for which either λ_n or $\tilde{\lambda}_n$ are non-zero are called the **support vectors**.

Additionally, it can be shown that; $\lambda_n \tilde{\lambda}_n = 0$, and $0 \leq \tilde{\lambda}_n \leq C$, $0 \leq \lambda_n \leq C$.

ϵ -support vector regression – the complete problem**Support vector regression prediction**

$$\hat{y} = \sum_{n=1}^N (\tilde{\lambda}_n - \lambda_n) \kappa(\mathbf{x}_n, \mathbf{x}) + \hat{\theta}_0$$

Support vector regression minimization

$$\begin{aligned} \arg \max_{\boldsymbol{\lambda}, \tilde{\boldsymbol{\lambda}}, \epsilon} \quad & \sum_{n=1}^N (\tilde{\lambda}_n - \lambda_n) y_n - \epsilon (\tilde{\lambda}_n - \lambda_n) \\ & - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N (\tilde{\lambda}_n - \lambda_n)(\tilde{\lambda}_m - \lambda_m) \kappa(\mathbf{x}_n, \mathbf{x}_m) \\ \text{s.t.} \quad & 0 \leq \tilde{\lambda}_n \leq C, \quad 0 \leq \lambda_n \leq C, \quad n = 1, 2, \dots, N \\ & \sum_{n=1}^N \tilde{\lambda}_n = \sum_{n=1}^N \lambda_n \end{aligned}$$

- Huber loss is the optimal robust loss if the noise follows a symmetric pdf.
- A tractable loss that leads to support vector regression is the ϵ -insensitive loss.
- Support vector regression:
 - seems to not have "training", but store points instead.
 - is solved by Lagrangian multipliers.
 - is very useful for de-noising and for anomaly detection.

Next week

Next week



Winter is ~~coming~~ here... exam...

Once more, thanks for all who evaluated. We use the feedback!

The course usually lacks TAs, please consider being TA next year.

Thank you for attending.