

TTT4185 Machine Learning for Signal Processing

Introduction

Giampiero Salvi and Tor Andre Myrvoll

Department of Electronic Systems
NTNU

HT2021

Who we are

Lecturers

- Giampiero Salvi: course responsible and lecturer
- Tor Andre Myrvoll: lecturer

Teaching assistants:

- Bettina D'Avila Barros
- Mohammad Adiban
- Ziaoyu Zhu
- Hossein Darvishi
- Zijian Fan

Who I am

Giampiero Salvi

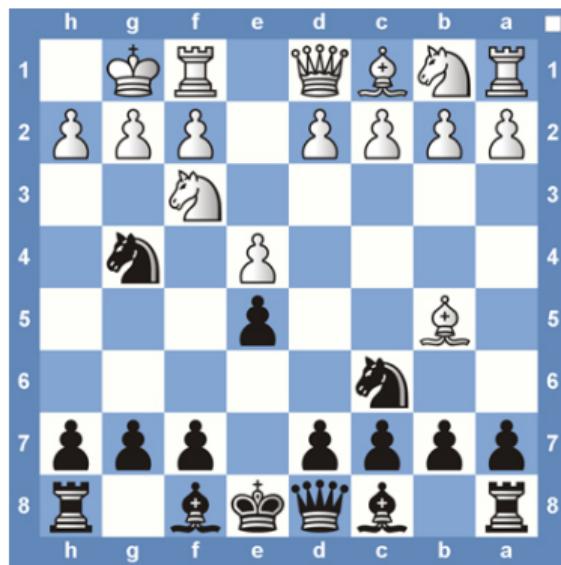
- MSc Electronic Engineering, Sapienza U. Rome, Italy
- PhD Computer Science, KTH, Stockholm, Sweden
 - Speech Technology
- PostDoc IST, Lisbon Portugal
 - Computer Vision and Robotics
- Associate Prof. at KTH since 2013
- Prof. at NTNU since 2019
- responsible 2-year MSc program in ML at KTH (2015-2019)
- teaching ML, Speech Recognition

Who are you?



Why Machine Learning?

AI in the 1970s



AI today



Examples of applications

Google self driving



IBM congestion fees



autonomous ships



Voice assistants



DeepMind AlphaGo



smart buildings



Moravec's Paradox

High cognitive processes

- conscious processes (chess, problem solving, . . .)
- difficult for humans
- easy for computers

Low cognitive processes

- perception, action, (social) interactions
- easy for humans
- difficult for computers

Explanation

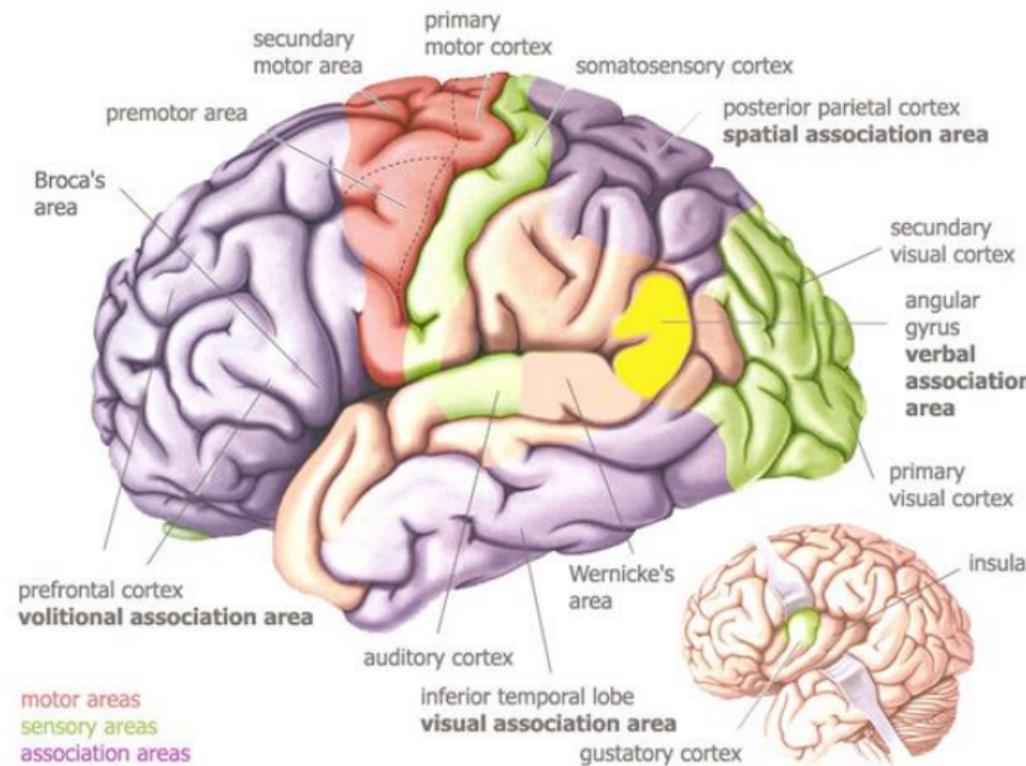
Interactions with the world have evolved over billions of years

- essential for survival and reproduction
- mainly **unconscious processes**
- we are **not aware** of the difficulty

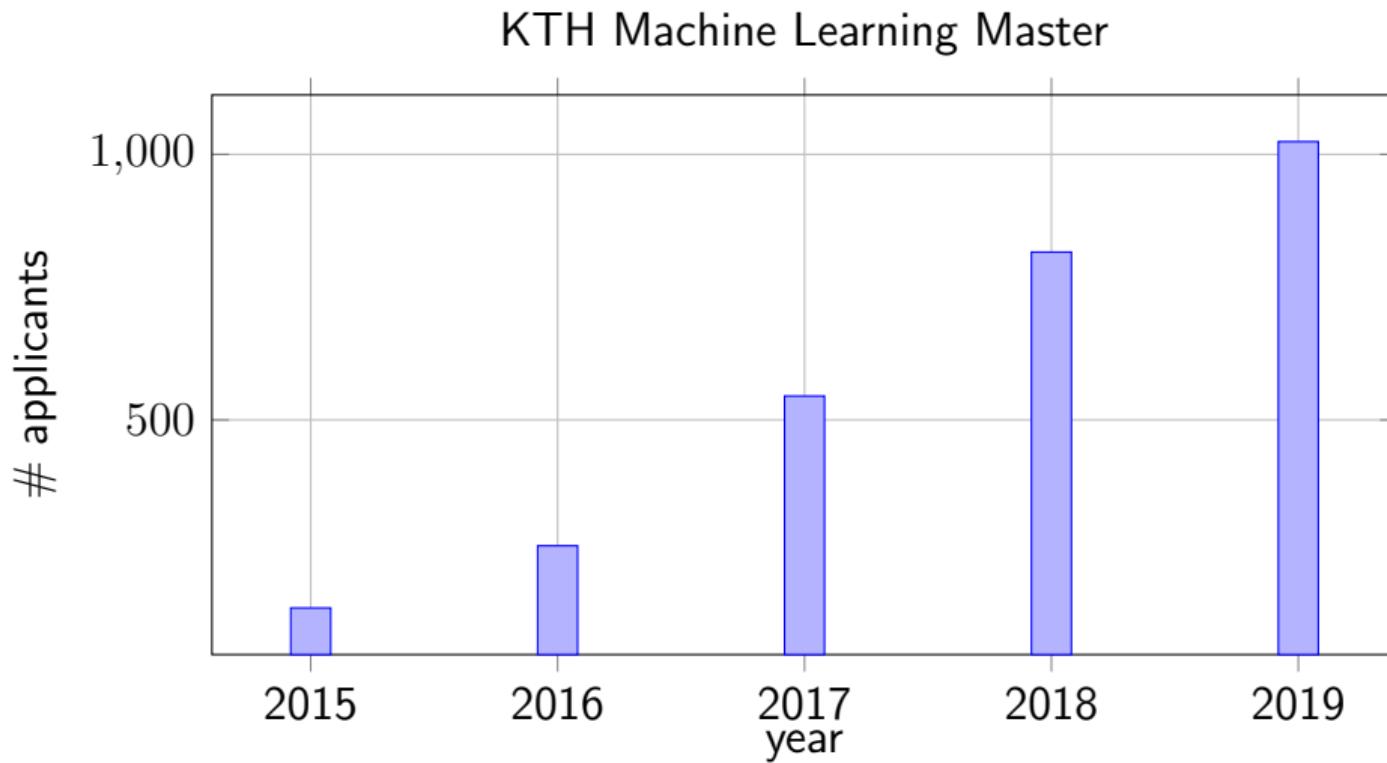
Abstract thinking is much more recent

- often conscious
- we **are aware** of the difficulty

Why does chess feel harder?

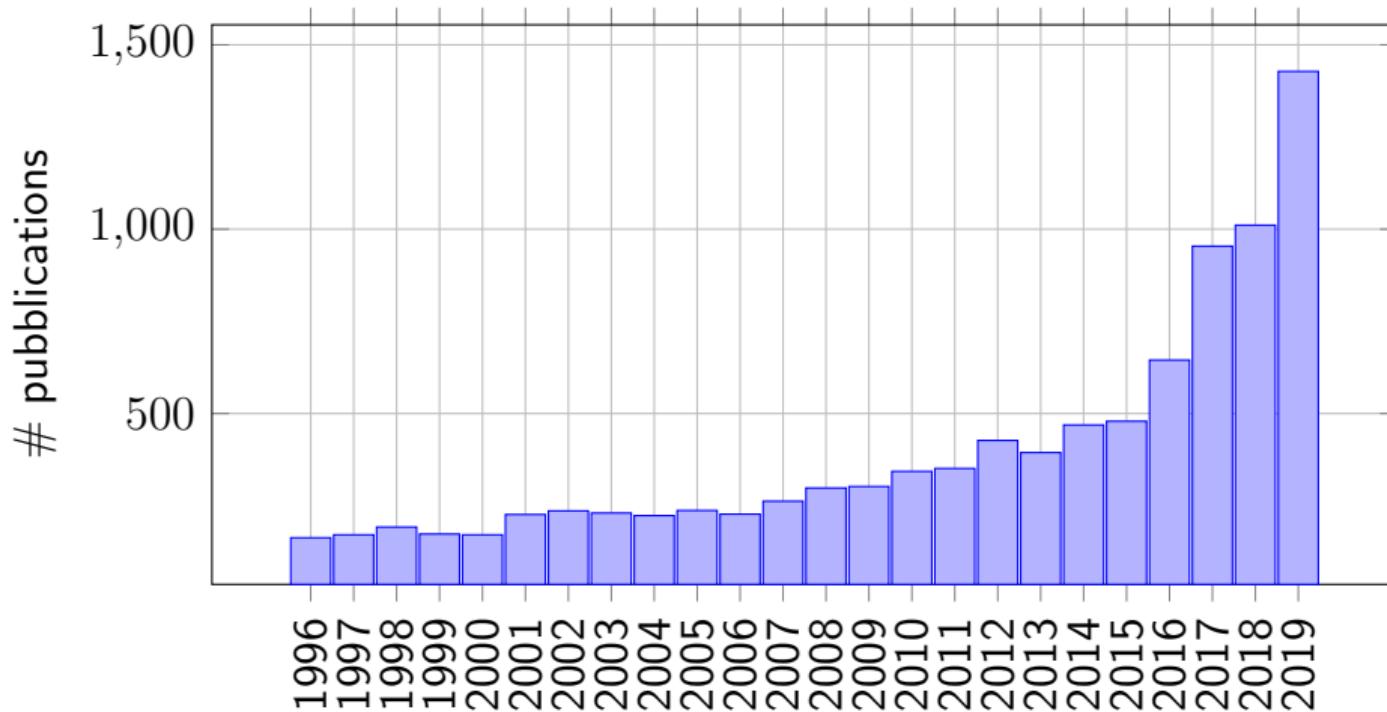


Number of applicants to ML master (KTH)

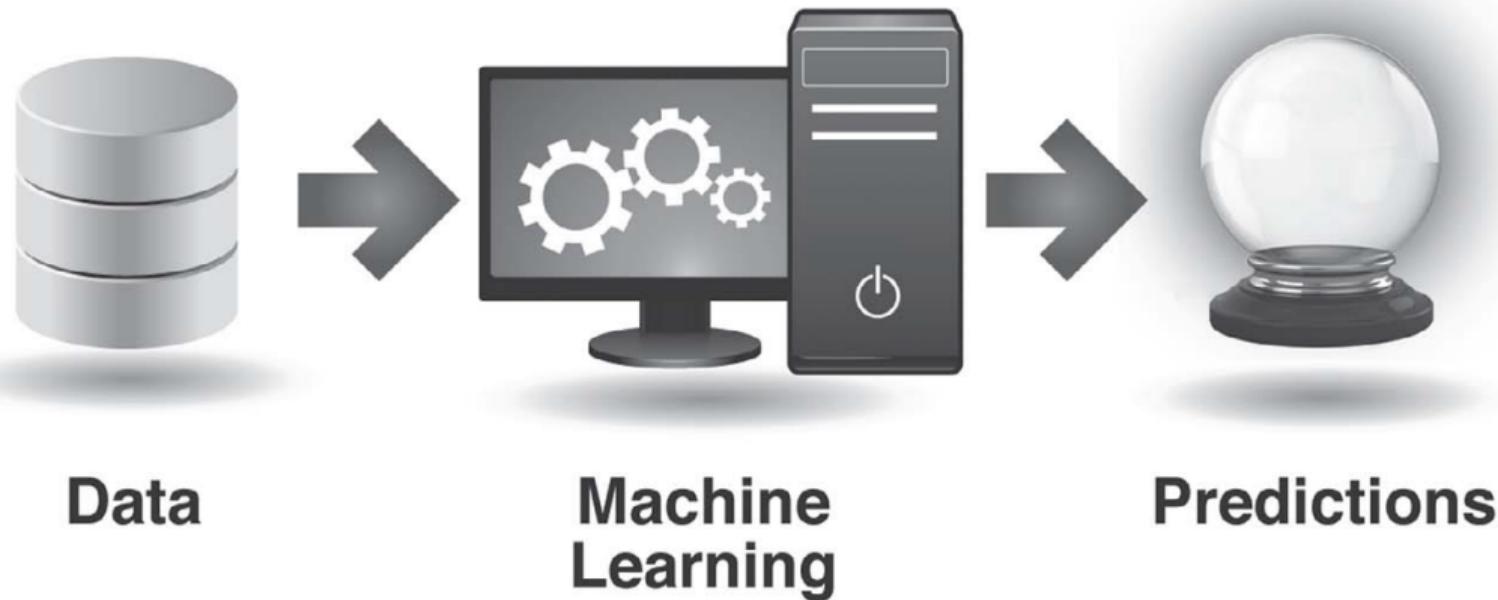


NeurIPS Publications

Number of Publications per Year

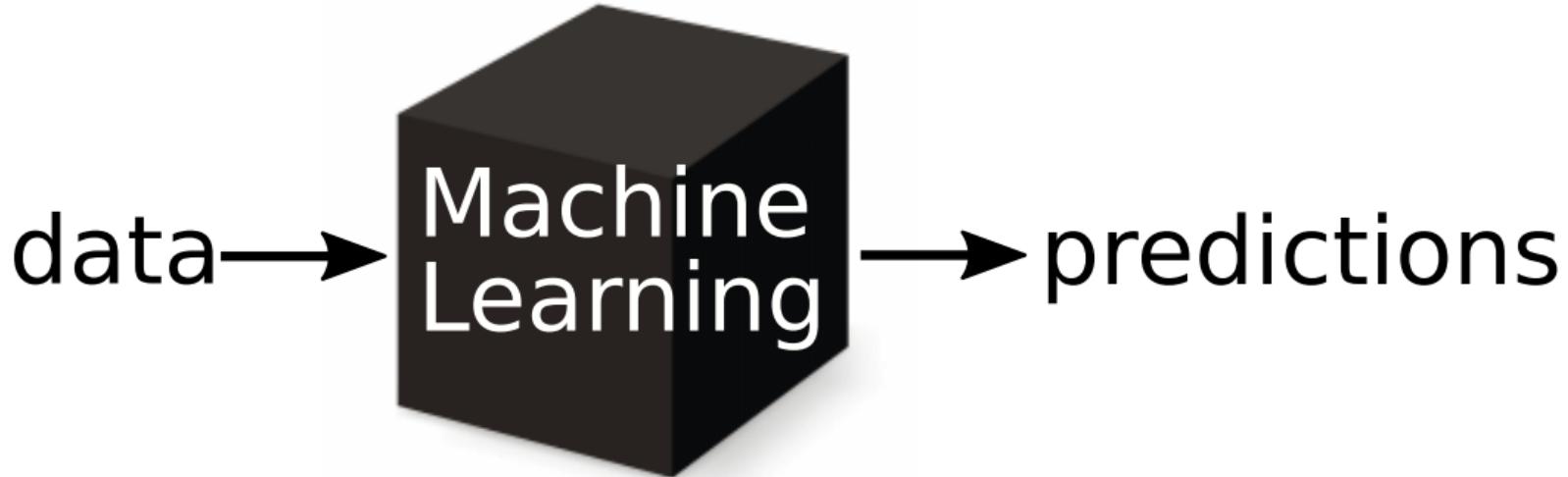


What is Machine Learning?

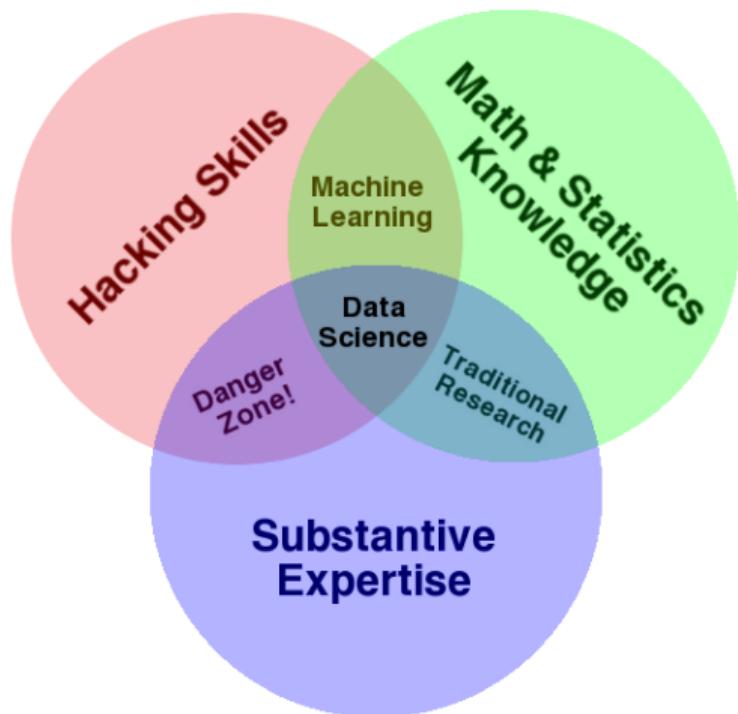


Source: *Predictive Analytics* by Eric Siegel

Challenges: Interpretability



Required skills



Machine Learning ≡
Theoretical ML

Data Science ≡
Applied ML

Source: <http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram>

Substantive Expertise

- dependent on the field
- extraction of relevant features
- necessary to double check predictions from ML (especially in high risk applications such as: medical, cyber security, vehicles)
- necessary to interpret ML results

In this course all examples are on **speech technology**

Why Speech Technology

- Focus of research at the signal processing group since the 70s
- renewed activities at IES:
 - G. Salvi, TB, Svendsen, TA, Myrvoll (20%), M. Siniscalchi (20%)
 - 6 PhD students, 1 postdoc
- complex problem (70 years of research, only recently good results)
- many modalities: acoustic, visual, textual
- many aspects (classification, regression, generation)
- inherently sequential
- all aspects can be approached with machine learning and signal processing

Practical Information

Lecture times:

- Mondays 10:15–12:00 S8
- Wednesdays 10:15–12:00 S6

Written Examination: 17th of December

Three plus one computer assignments:

- introduction to Python is voluntary
- the others are compulsory
- assessed with a short oral presentation
- likely relevant for the written examination

Prerequisites:

- Linear algebra, probability theory and statistics, signal processing

Course objectives

After the course, you should be able to

- **discuss** the theoretical foundations for ML
- **describe** the principles behind specific ML methods
- **apply** specific ML methods to a specific domain (speech technology)
- with the help of domain knowledge, **interpret** the results obtained
- **discuss** how the methods may be applied to other domains

Coarse course plan

Part 1: Speech analysis

- Speech production and perception
- Speech modelling
- Feature extraction

Part 3: Neural Networks and deep learning

- Deep neural networks: Definition, training
- Convolutional and Recursive DNNs
- Special networks: Generative Adversarial Networks, auto-encoders...

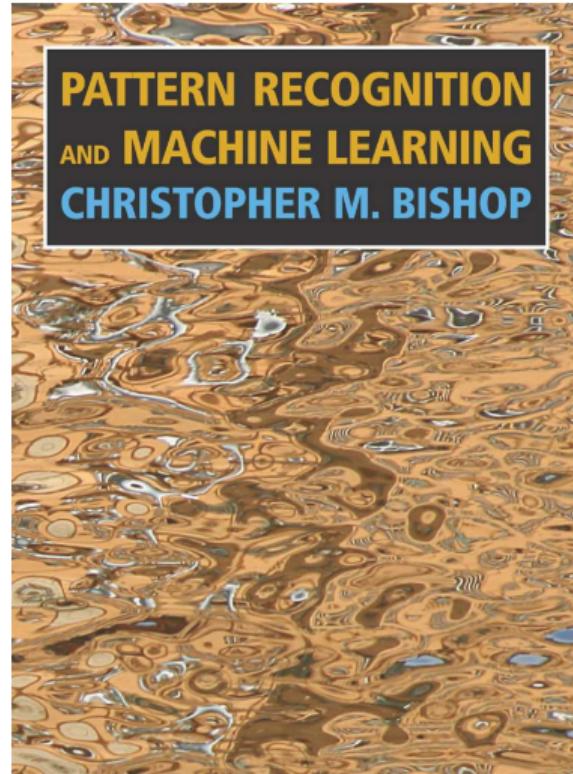
Part 4: Advanced topics

- Probability reminder
- Linear models for regression
- Linear models for classification

- Kernel methods, graphical models
- Unsupervised learning, EM algorithm, Hidden Markov Models
- Model combination, boosting, trees, forests

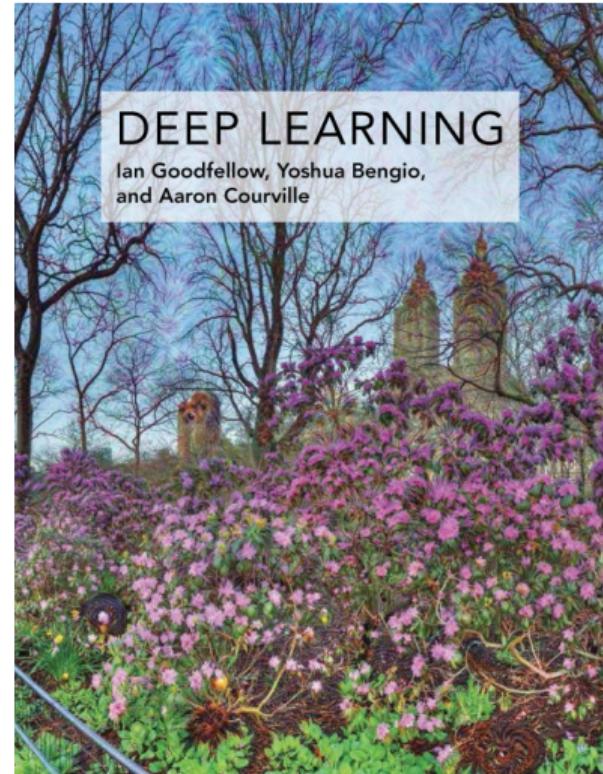
Literature

C. M. Bishop.
Pattern Recognition and Machine
Learning.
Springer, 2006



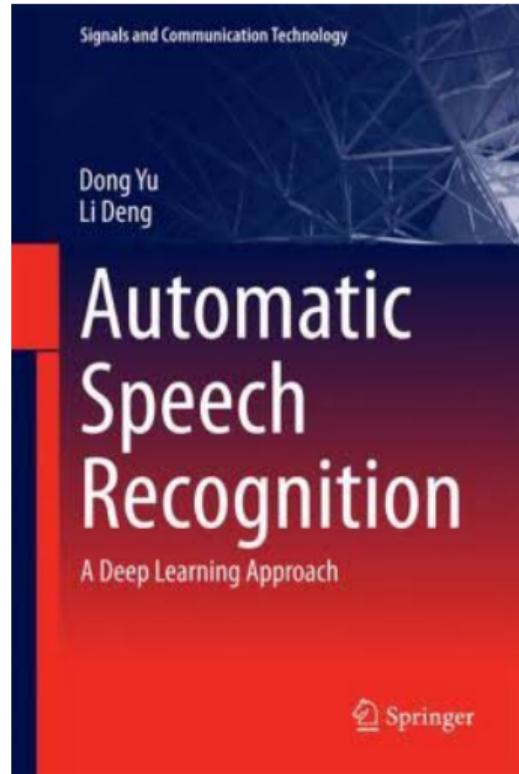
Literature

Goodfellow, Bengio, Courville
Deep Learning
ISBN-10: 0262035618
Freely available at <https://www.deeplearningbook.org>



Literature

D. Yu and L. Deng.
Automatic Speech Recognition, a Deep Learning Approach.
Springer, 2015
Available in PDF through NTNU Library

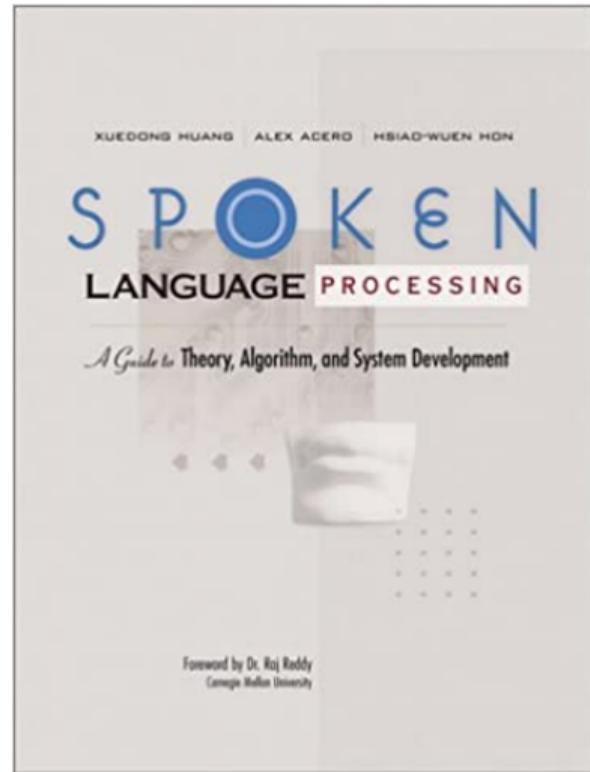


Literature

X. Huang, A. Acero, H-W. Hon

Spoken Language Processing — A guide to theory, algorithm, and system development

ISBN-13: 978-0130226167



Computer assignments

- Based on Python:
 - de facto standard language for machine learning
 - large number of very high quality ML libraries (tensorflow, keras, scikit-learn, pytorch, pandas, ...)
 - free software!!
 - no extra work from research to deployment
- no need to be expert Python programmer
- compulsory to be admitted to the exam
- assessed with short oral presentation (arranged by TAs)
- the introduction to python is voluntary

Course Reference Group

- Need ~ 3 students to take part in 2 short meetings this semester
- you need to be available to other students (collect inputs to the course)

Benefits

- impact on the course this and later semesters
- valuable collaborations and connections

Sign up at the end or through giampiero.salvi@ntnu.no

- we will try our best to keep activities in presence
- if not possible, lectures, meetings with TAs, and assignment presentations will be carried out through Zoom (or equivalent)
- always follow NTNU guidelines: <https://www.ntnu.edu/corona>
- all information relevant to the course will be posted in BlackBoard
- the exam form will also be determined by the current rules
- all questions (if not personal) should be posted in the **Course Forum** under **Discussion Board**

BlackBoard Discussion Board/Course Forum

- ① we will check the forum regularly (whereas email is overloaded)
- ② the answers will benefit all the students
- ③ students are encouraged to answer each other's questions in case they are confident they know the answer. We will, if needed, correct/complete the answers
- ④ questions and answers will not affect your final grade: ask anything freely

Questions



Speech Production and Perception

TTT4185 Machine Learning for Signal Processing

Giampiero Salvi

Department of Electronic Systems
NTNU

HT2021

Why use speech?

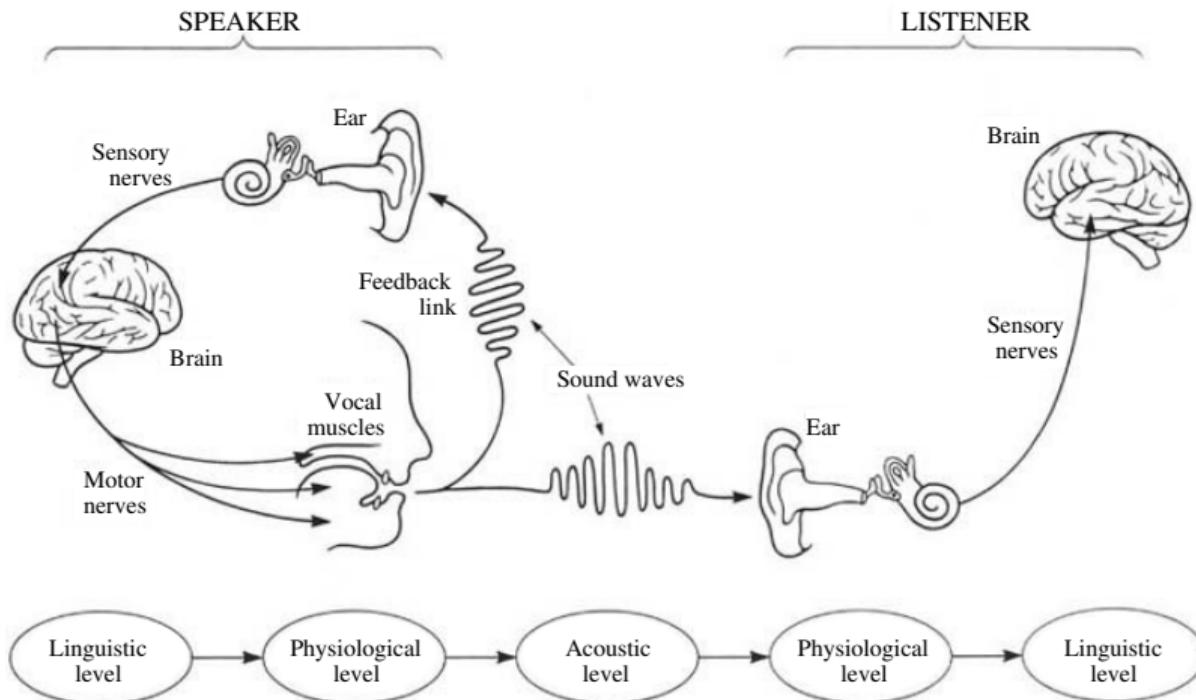
Human-Computer (or -Robot) Interaction

- Natural way of communication (No training needed)
- Leaves hands and eyes free (Good for functionally disabled)
- Effective (Higher data rate than typing)
- Can be transmitted/received inexpensively (phones)

Surveillance/Search

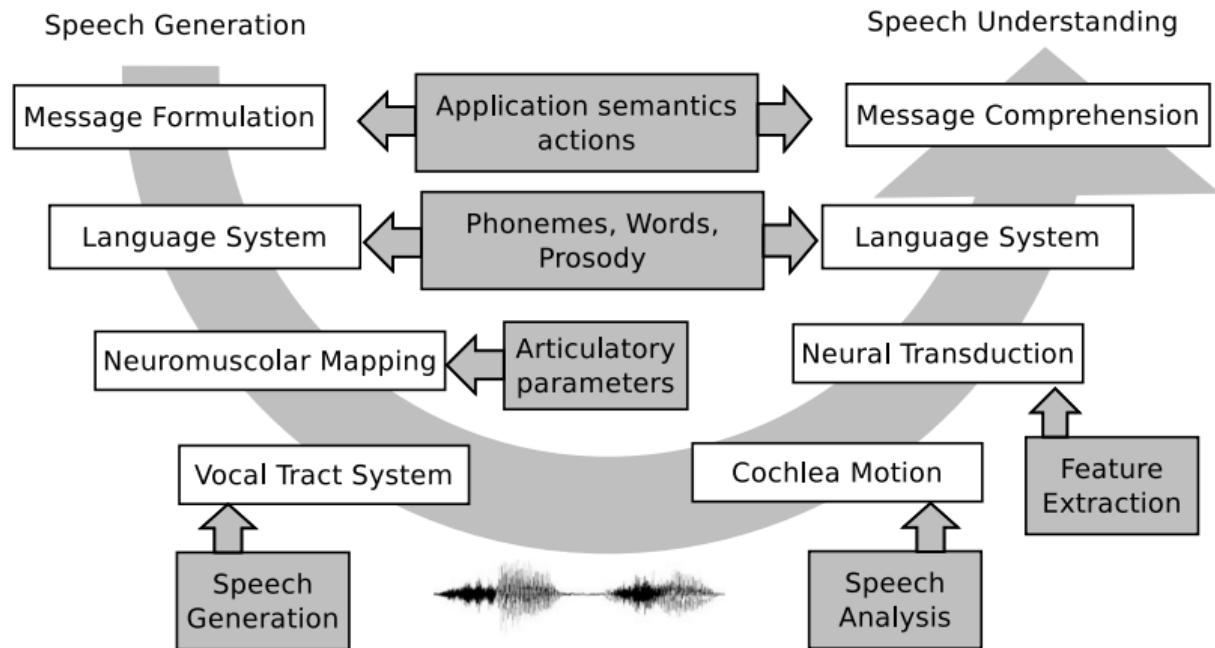
- Transcribe human-human conversations
- produce indexing for broadcast material
- produce subtitles for movies/news

The Speech Chain



Peter Denes, Elliot Pinson, 1963

The Speech Chain revisited



from Huang, Acero and Hon's book

Outline

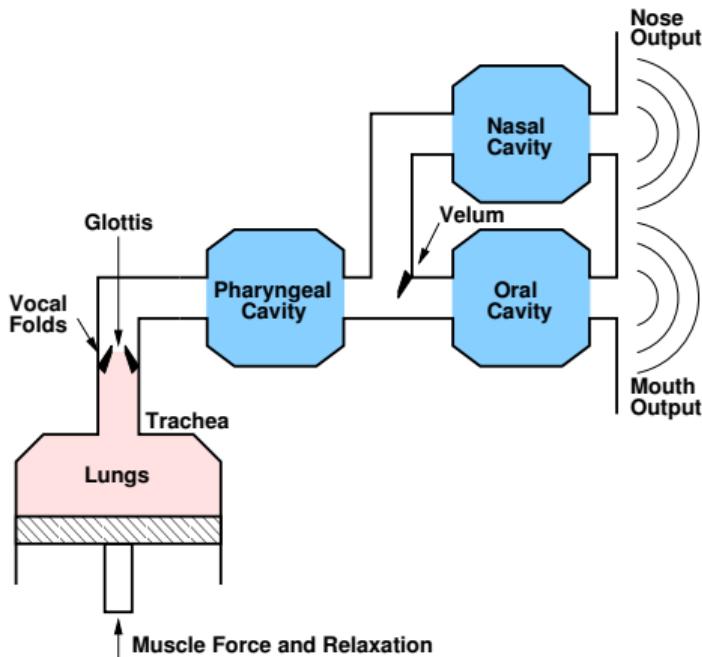
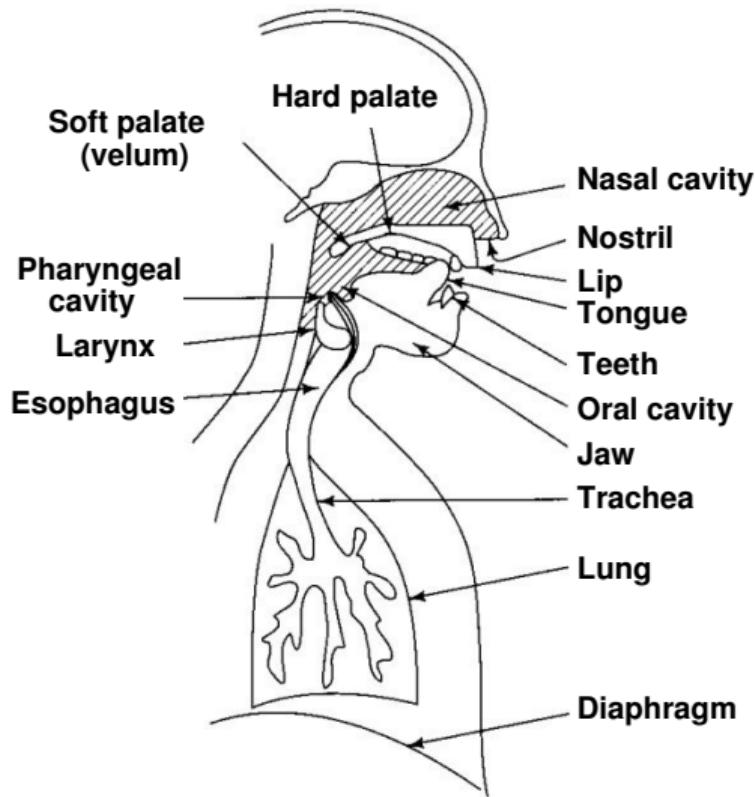
1 Speech Production

- Source/Filter Model

2 Speech Perception

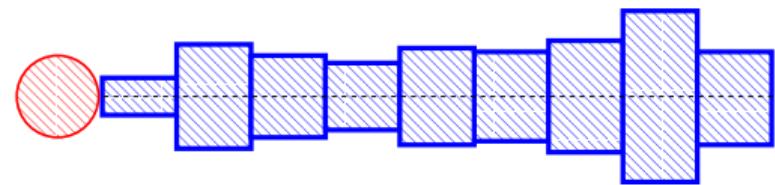
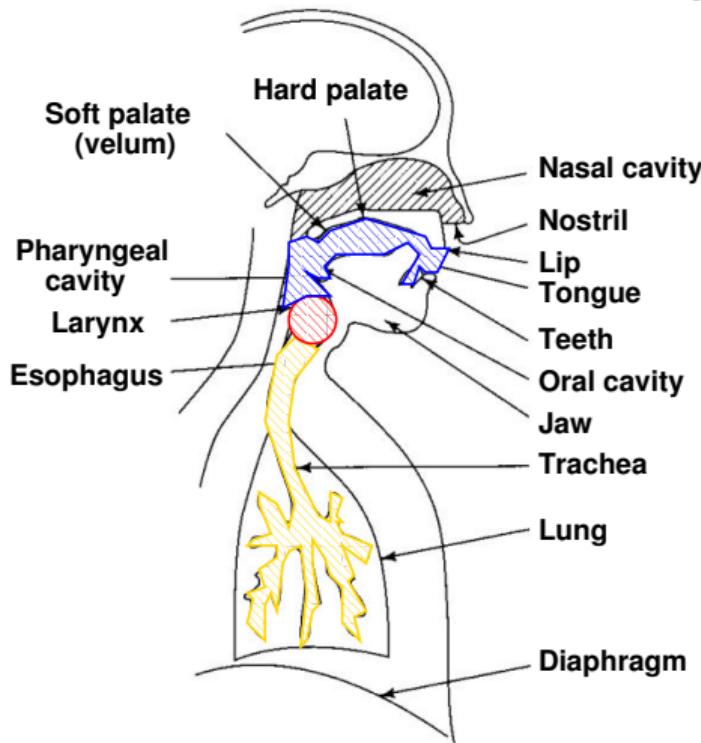
3 Challenges

Physiology



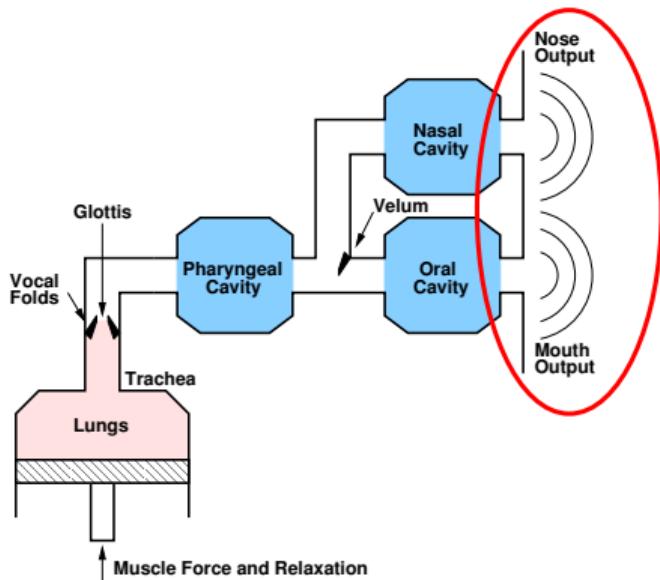
Source/Filter Model, Vowel-like sounds

Vowels



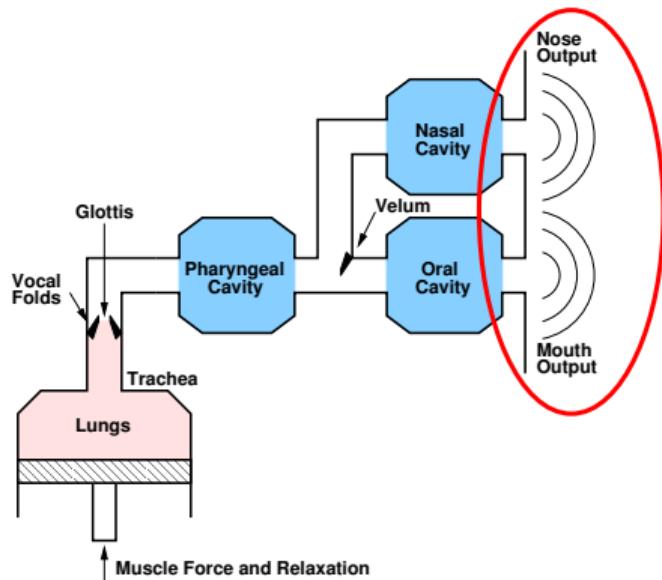
- Source (periodic)
- Front Cavity
- Back Cavity
- Back Cavity (2nd approx.)

Radiation form the Lips/Nose

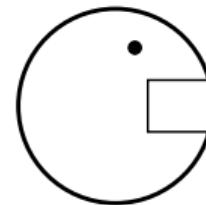


Problem of radiation at the lips plus diffraction about the head too complicated.

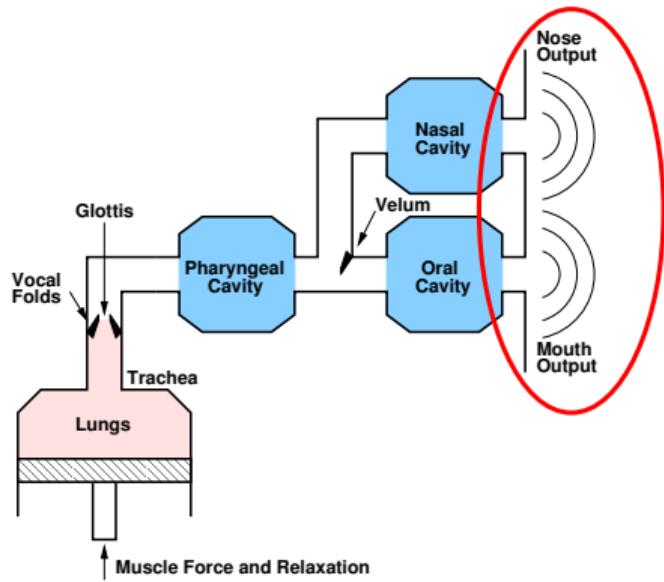
Radiation form the Lips/Nose



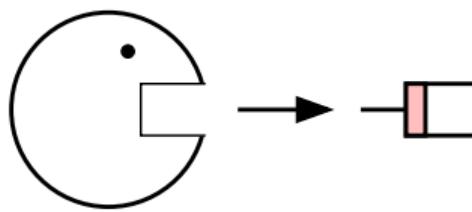
Approx. with a piston in a rigid sphere: solved
but not in closed form



Radiation form the Lips/Nose



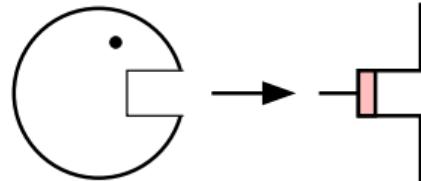
2nd approx: piston in an infinite wall



$$R(z) \approx 1 - \alpha z^{-1}$$

Radiation form the Lips/Nose

2nd approx: piston in an infinite wall



$$R(z) \approx 1 - \alpha z^{-1}$$

Question:

Given $R(z) = 1 - \alpha z^{-1}$ is the transfer function of a linear system, what is the relationship between the system input $x[n]$ and its output $y[n]$?

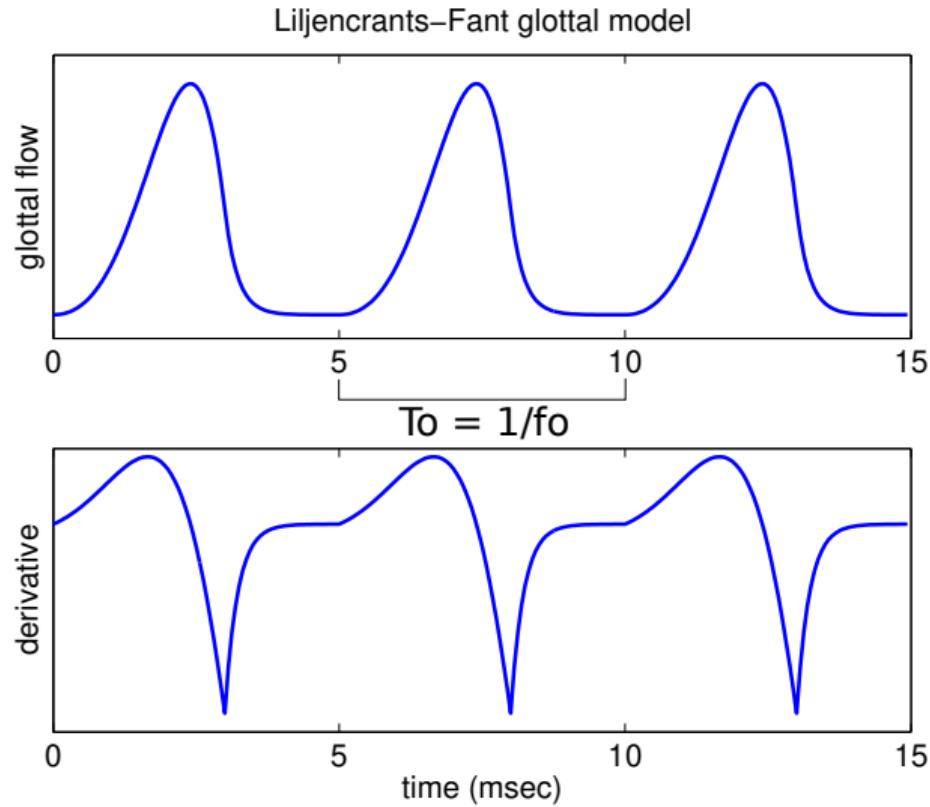
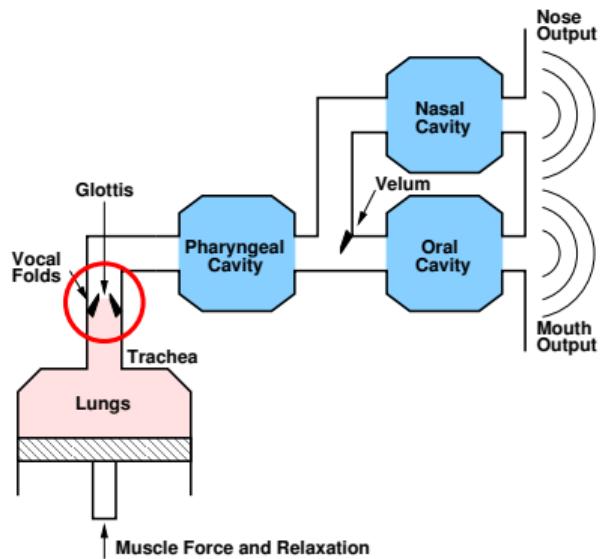
- a) $y[n] = \text{constant}$
- b) $y[n] = 1 - \alpha x[n]$
- c) $y[n] = x[n] - \alpha x[n - 1]$
- d) $y[n] = 1 - \alpha x[n - 1]$

Glottal Flow: Laryngoscopy

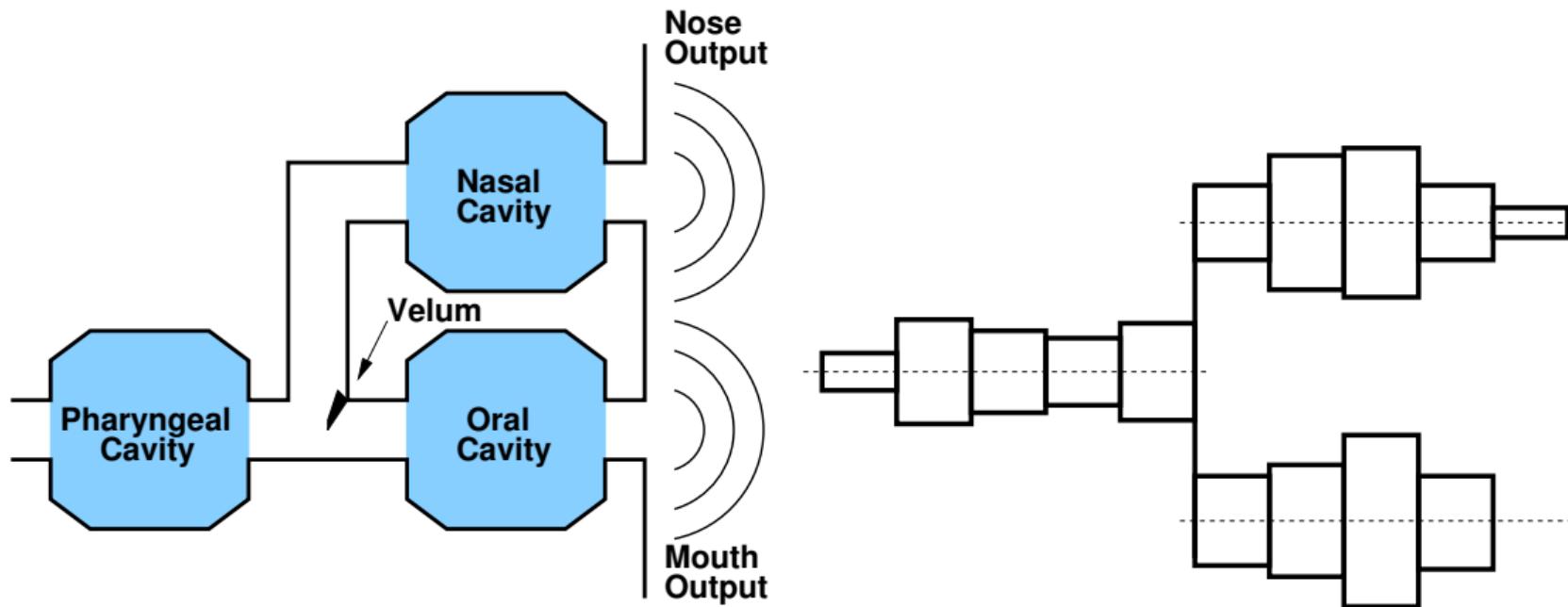


<https://youtu.be/iYpDwhpILkQ>

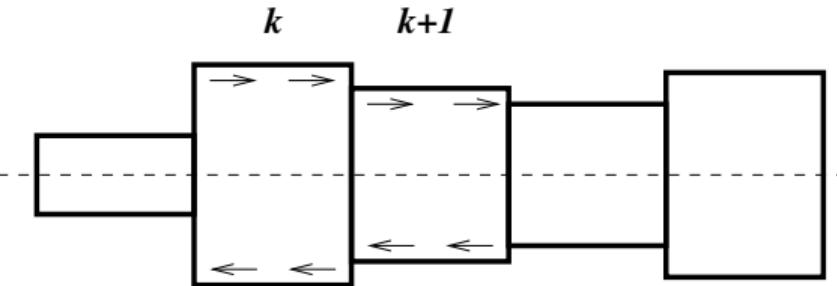
Glottal Flow



Tube Model of the Vocal Tract



Tube Model (cntd.)

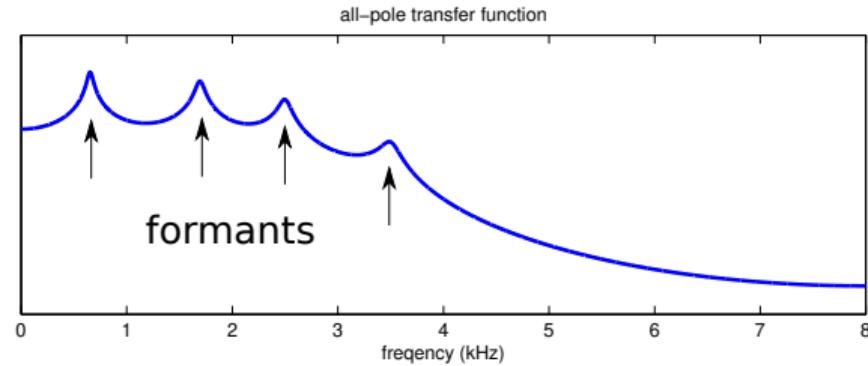
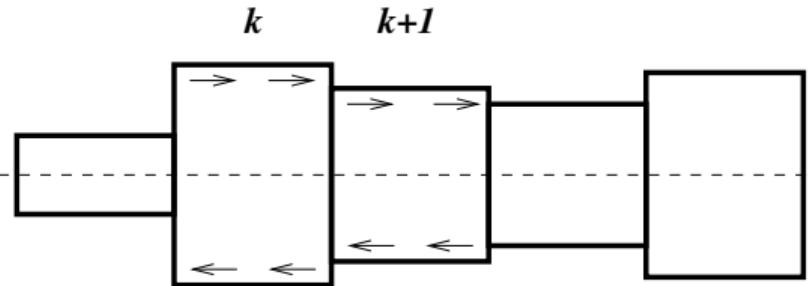


- assume planar wave propagation and lossless tubes
- solve pressure $p(x, t)$ and velocity $u(x, t)$ in each tube according to wave equation
- impose continuity of pressure and velocity at the junctions

⇒ all-pole transfer function ($N = \text{number of tubes}$)

$$V(z) = \frac{Az^{-N/2}}{1 - \sum_{k=1}^N a_k z^{-k}}$$

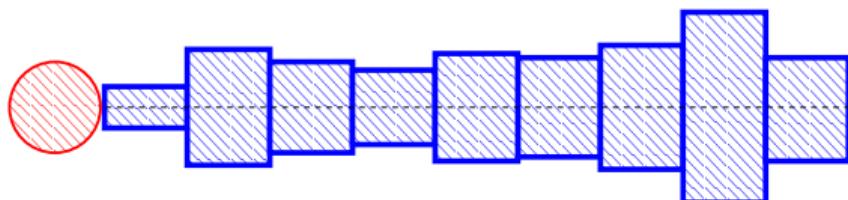
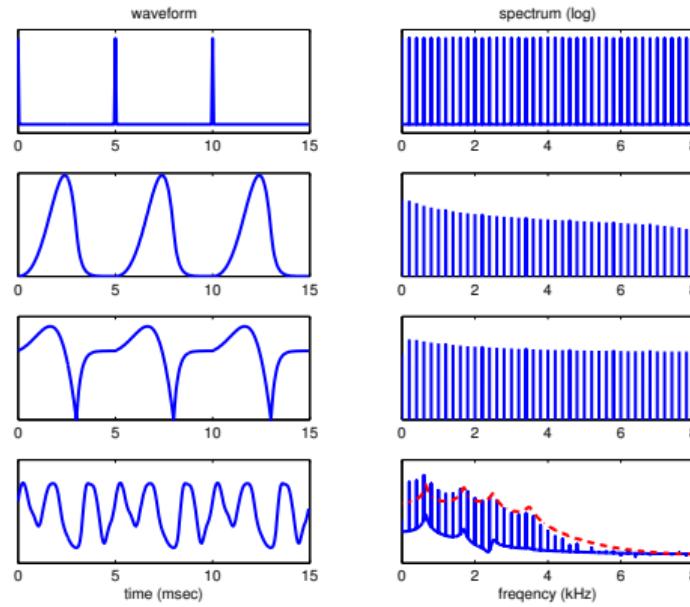
Tube Model (cntd.)



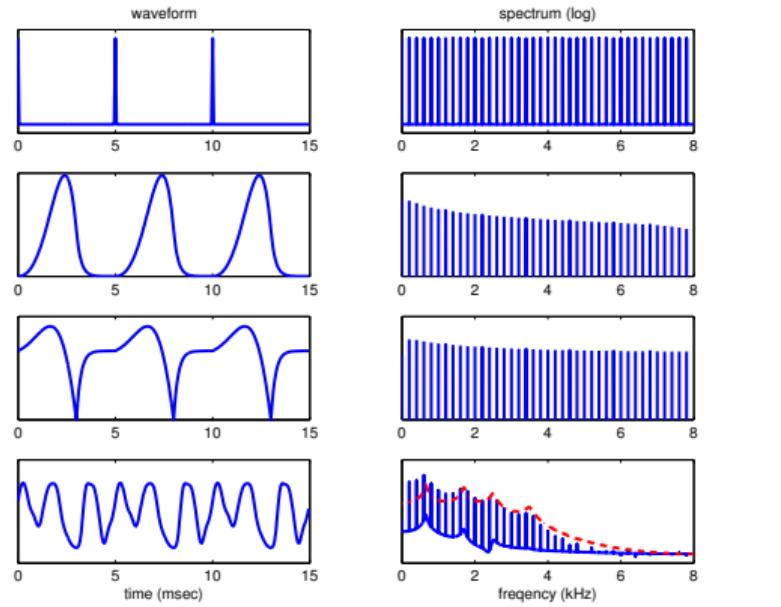
- assume planar wave propagation and lossless tubes
 - solve pressure $p(x, t)$ and velocity $u(x, t)$ in each tube according to wave equation
 - impose continuity of pressure and velocity at the junctions
- ⇒ all-pole transfer function ($N = \text{number of tubes}$)

$$V(z) = \frac{Az^{-N/2}}{1 - \sum_{k=1}^N a_k z^{-k}}$$

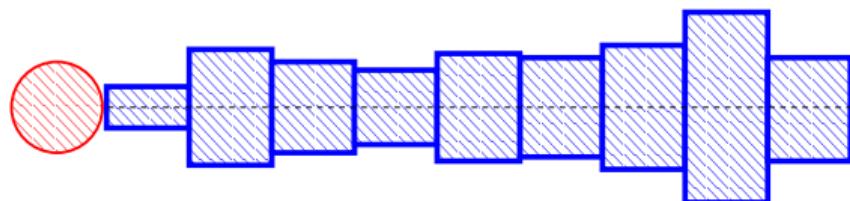
Source/Filter Model: vowel-like sounds



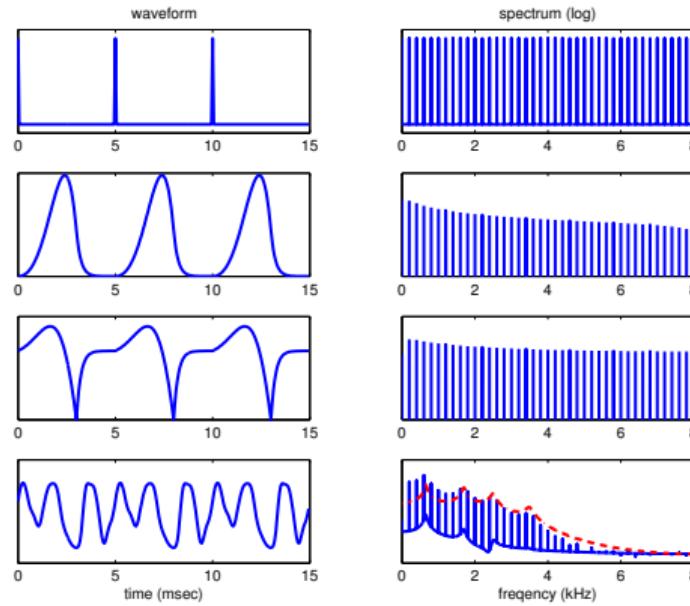
Source/Filter Model: vowel-like sounds



$\leftarrow p[n]$

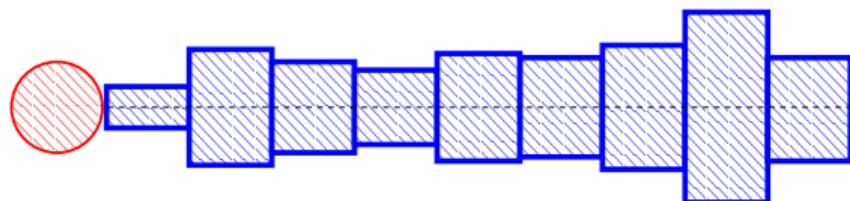


Source/Filter Model: vowel-like sounds

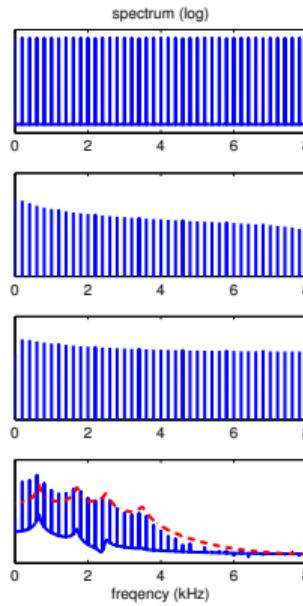
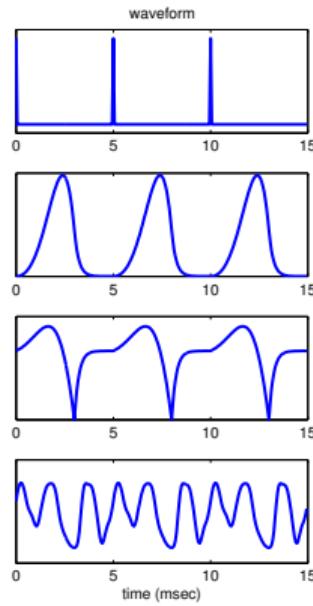


$$\leftarrow p[n]$$

$$\leftarrow p[n] * g[n]$$



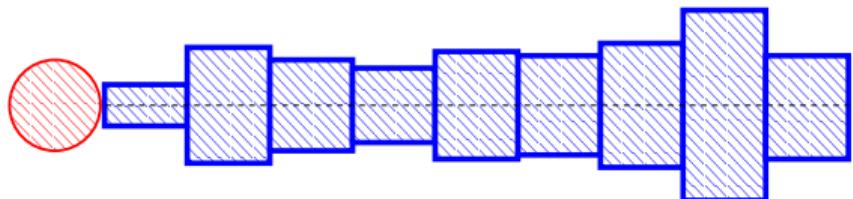
Source/Filter Model: vowel-like sounds



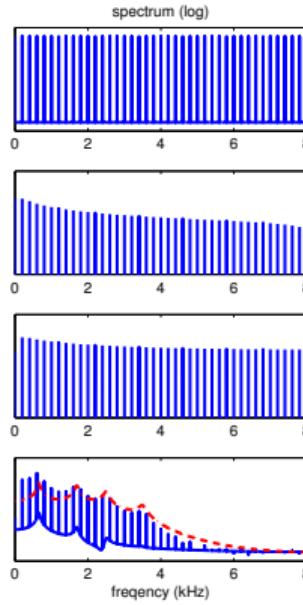
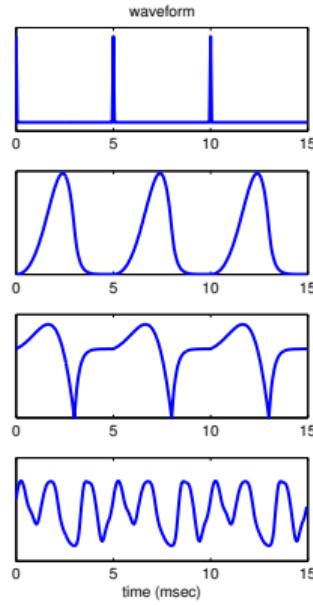
$$\leftarrow p[n]$$

$$\leftarrow p[n] * g[n]$$

$$\leftarrow p[n] * g[n] * r[n]$$



Source/Filter Model: vowel-like sounds

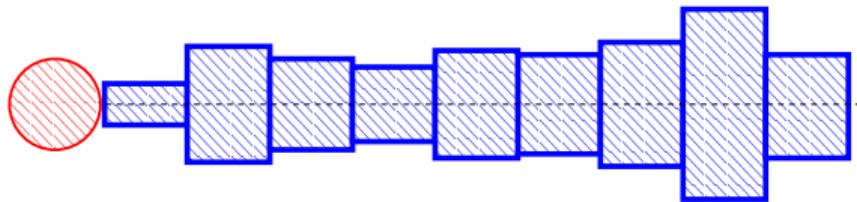


$$\leftarrow p[n]$$

$$\leftarrow p[n] * g[n]$$

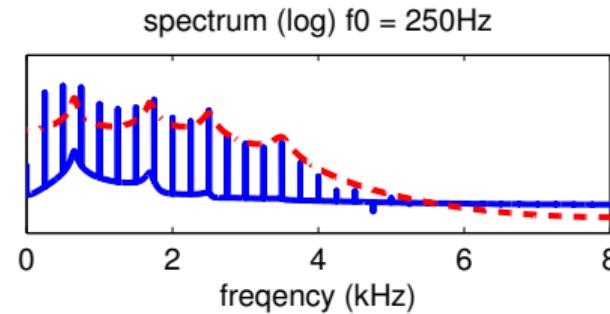
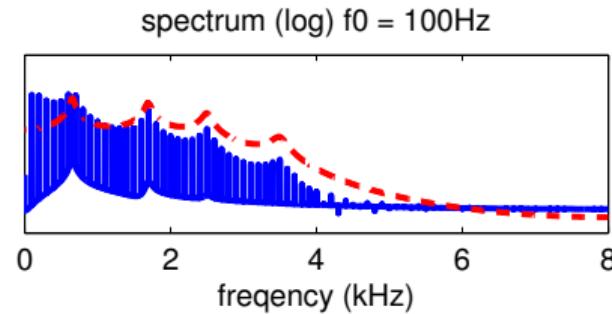
$$\leftarrow p[n] * g[n] * r[n]$$

$$\leftarrow p[n] * g[n] * r[n] * v[n]$$



f_0 and Formants

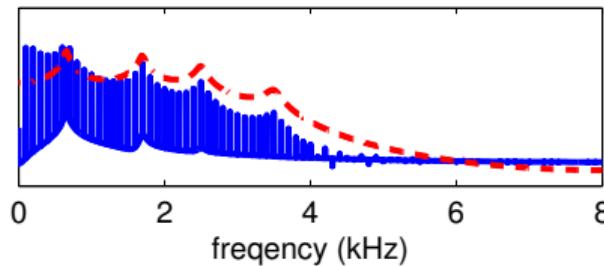
- Varying f_0 (vocal fold oscillation rate)



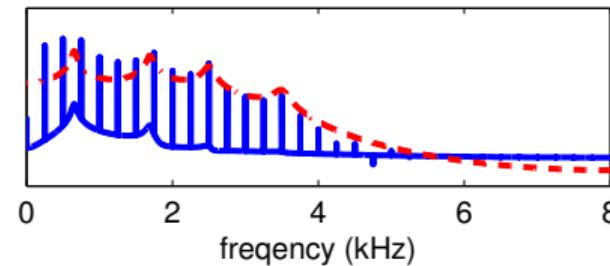
f_0 and Formants

- Varying f_0 (vocal fold oscillation rate)

spectrum (log) $f_0 = 100\text{Hz}$

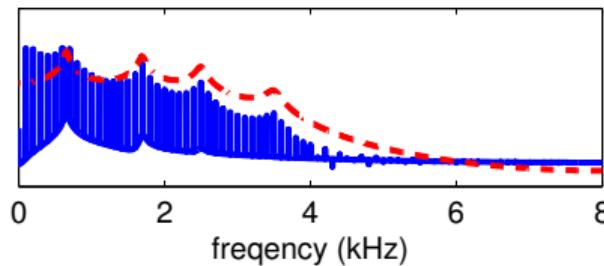


spectrum (log) $f_0 = 250\text{Hz}$

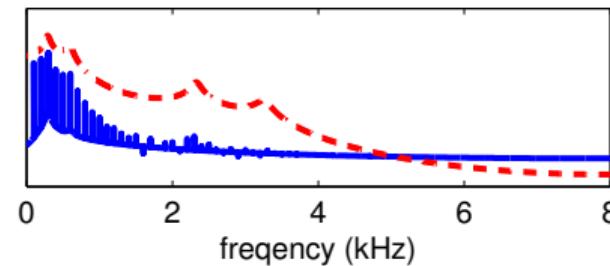


- Varying Formants (vocal tract shape)

spectrum (log) vowel [ɛ]



spectrum (log) vowel [u]



Demonstration of speech sounds

Wavesurfer

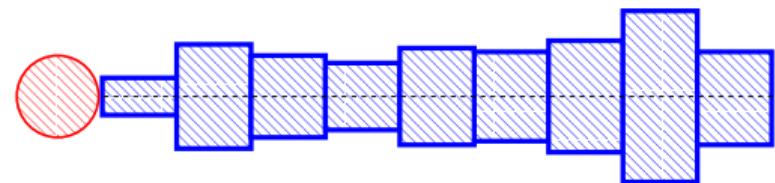
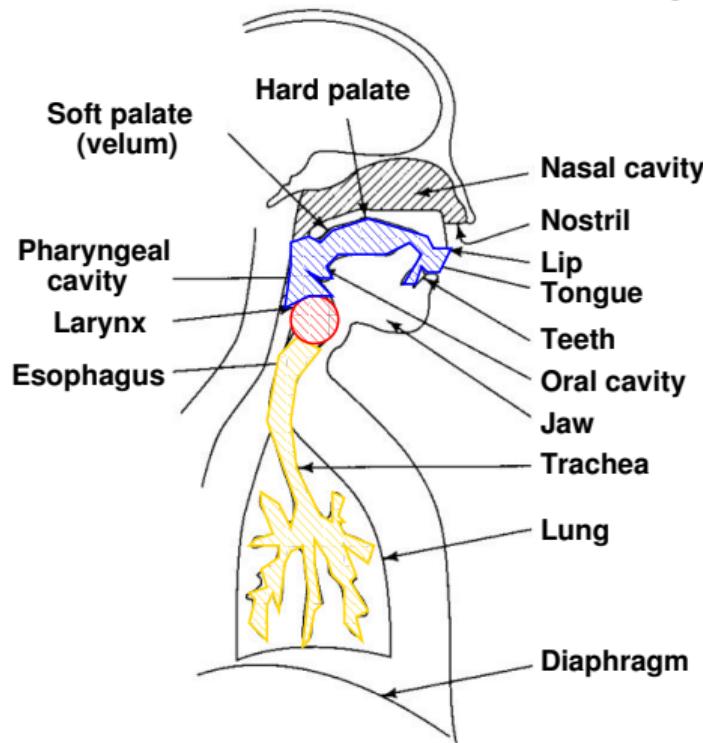
<https://sourceforge.net/projects/wavesurfer/>

Praat

<https://www.fon.hum.uva.nl/praat/>

Source/Filter Model, General Case

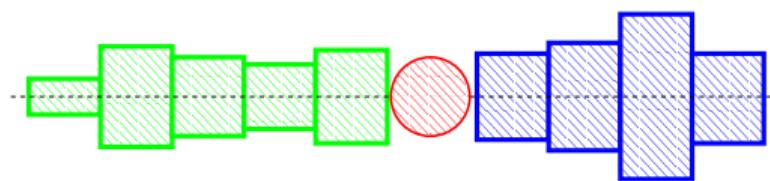
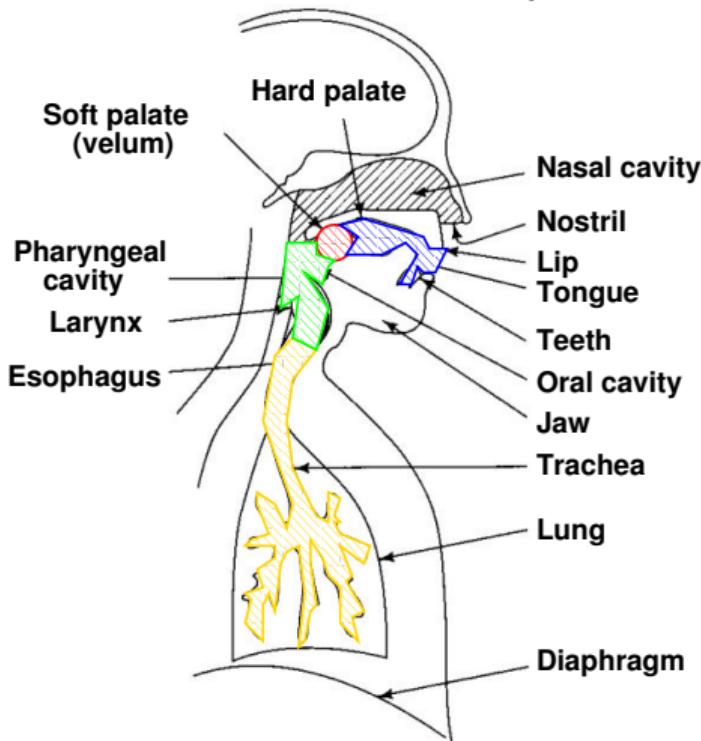
Vowels



- Source (periodic)
- Front Cavity
- Back Cavity
- Back Cavity (2nd approx.)

Source/Filter Model, General Case

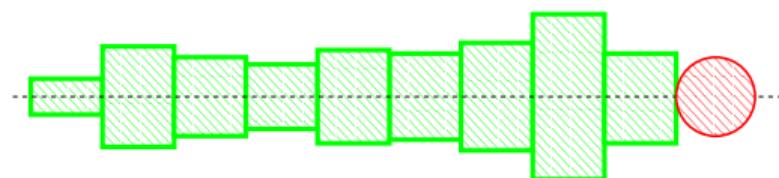
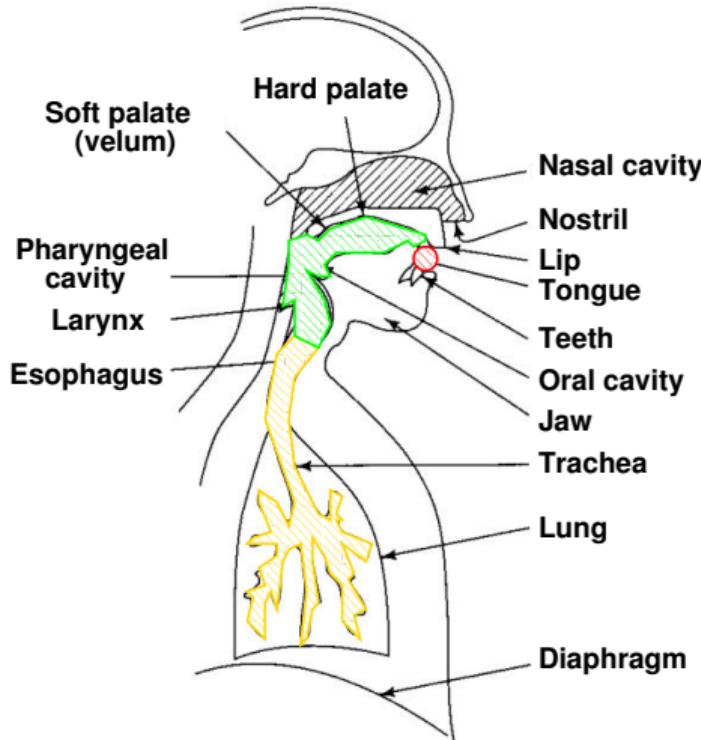
Fricatives (e.g. sh) or Plosive (e.g. k)



- Source (noise or impulsive)
- Front Cavity
- Back Cavity
- Back Cavity (2nd approx.)

Source/Filter Model, General Case

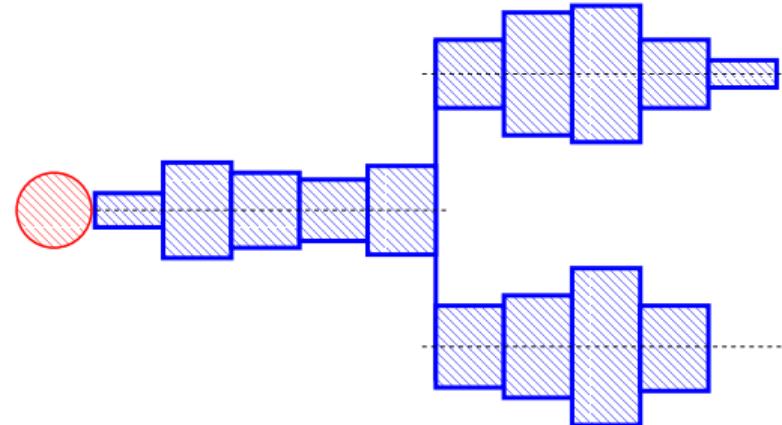
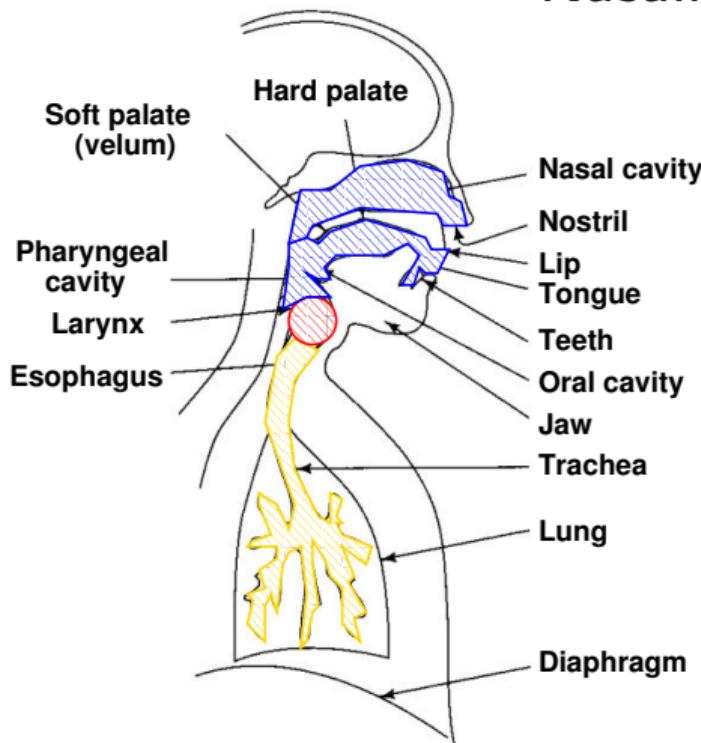
Fricatives (e.g. s) or Plosive (e.g. t)



- Source (noise or impulsive)
- Front Cavity
- Back Cavity
- Back Cavity (2nd approx.)

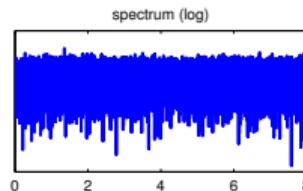
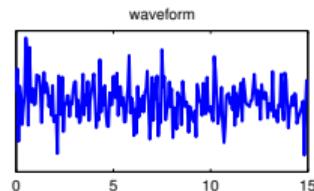
Source/Filter Model, General Case

Nasalised Vowels

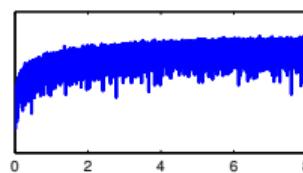
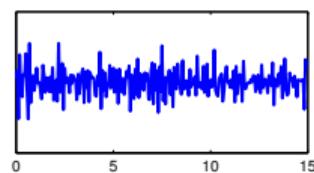


- Source (periodic)
- Front Cavity
- Back Cavity
- Back Cavity (2nd approx.)

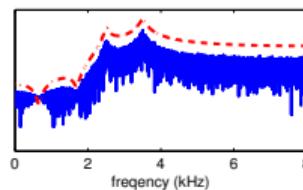
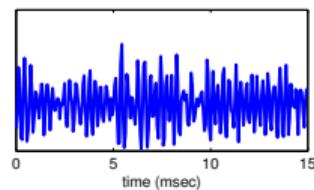
Source/Filter Model: fricative sounds



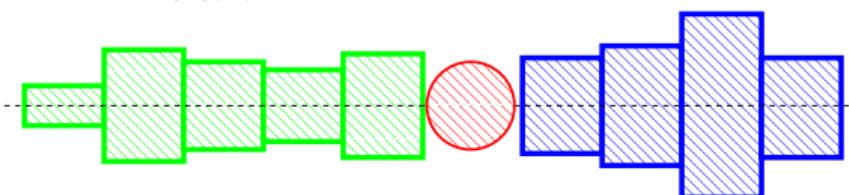
$\leftarrow p[n]$



$\leftarrow p[n] * r[n]$

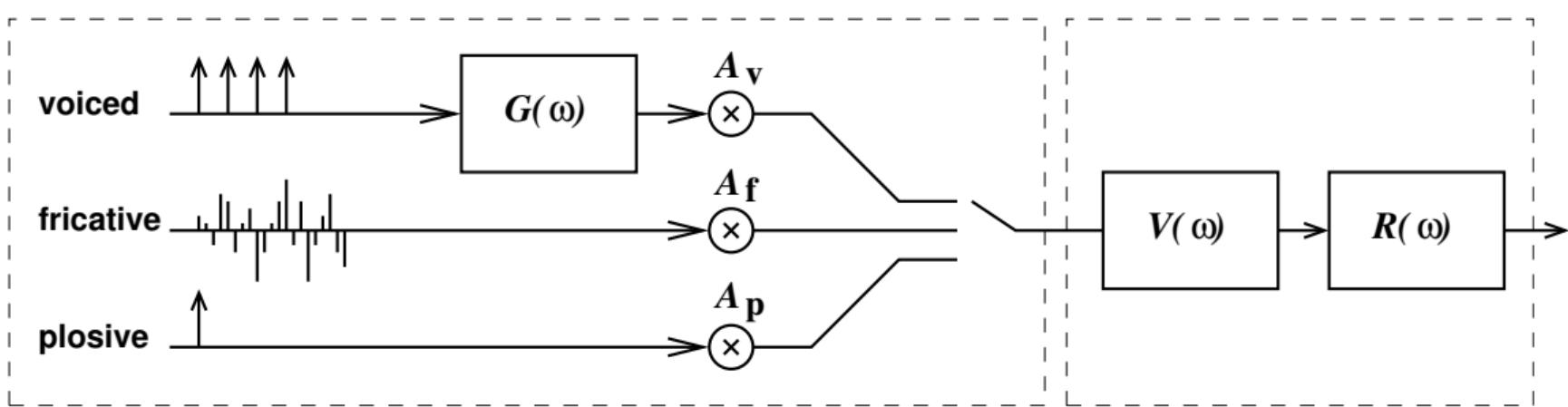


$\leftarrow p[n] * r[n] * v[n]$



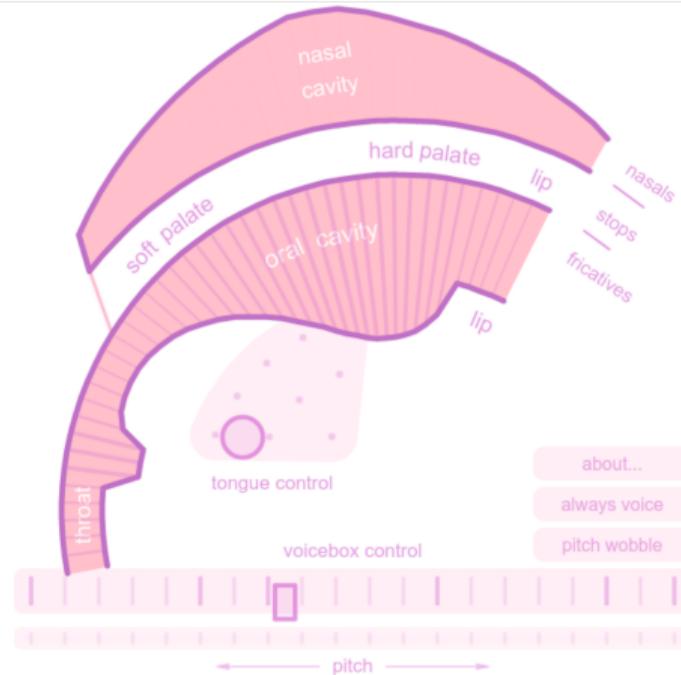
Complete Source/Filter Model

Source



Pink Trombone!

<http://dood.al/pinktrombone/>



IPA Chart: Consonants

THE INTERNATIONAL PHONETIC ALPHABET (2005)

CONSONANTS (PULMONIC)

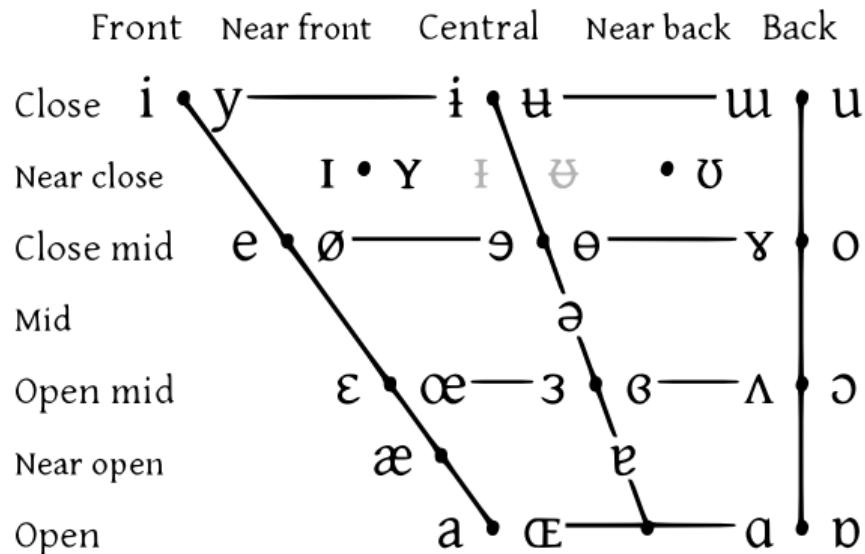
	LABIAL		CORONAL				DORSAL			RADICAL		LARYNGEAL
	Bilabial	Labio-dental	Dental	Alveolar	Palato-alveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Epi-glottal	Glottal
Nasal	m	n̪		n		ɳ	ɲ	ŋ	ɳ			
Plosive	p b	ɸ β		t d		t̪ d̪	c ɟ	k ɡ	q ɢ		ʔ	ʔ
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ɟ	x ɣ	χ ʁ	h	h	h
Approximant		v		ɹ		ɬ	j	w		l̪	ɫ	h̪
Trill	b			r						r̪		ɾ̪
Tap, Flap		v		t̪		t̪						
Lateral fricative			ɬ	ɺ	ɭ	ɭ	ʎ	ɻ				
Lateral approximant			ɬ	ɺ	ɭ	ɭ	ʎ	ɻ				
Lateral flap			ɬ	ɺ	ɭ	ɭ						

Where symbols appear in pairs, the one to the right represents a modally voiced consonant, except for murmured h̪.
 Shaded areas denote articulations judged to be impossible. Light grey letters are unofficial extensions of the IPA.

IPA Chart: Vowels

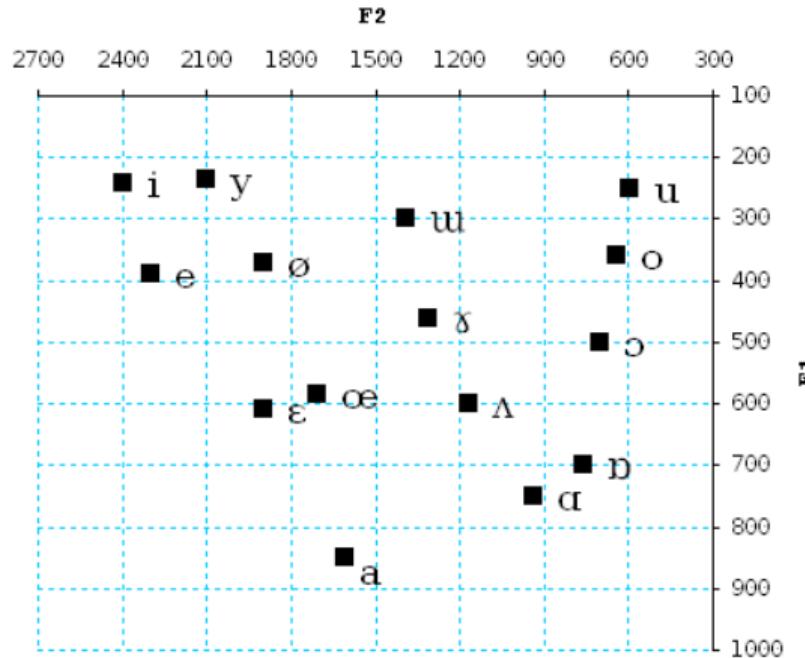
THE INTERNATIONAL PHONETIC ALPHABET (2005)

VOWELS



Vowels at right & left of bullets are rounded & unrounded.

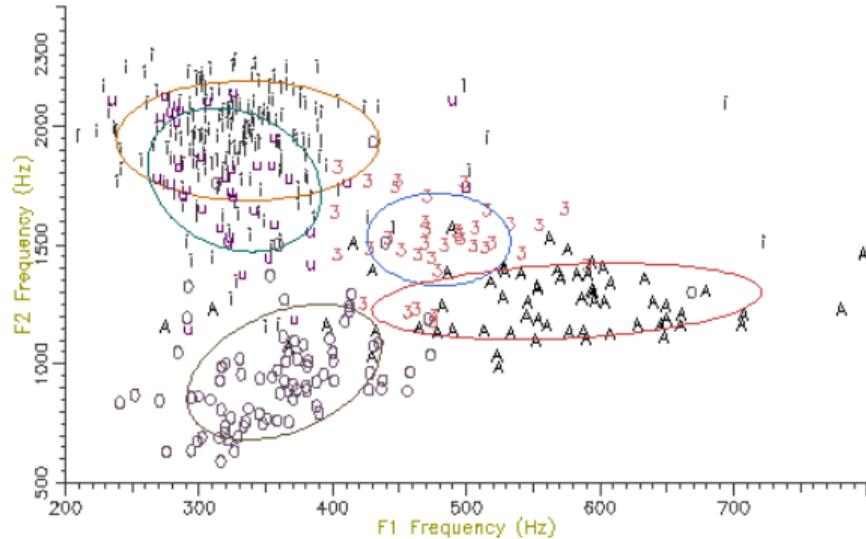
Average formant frequencies (vowels)



Source: <https://en.wikipedia.org/wiki/Formant>

Formant frequencies, single speaker

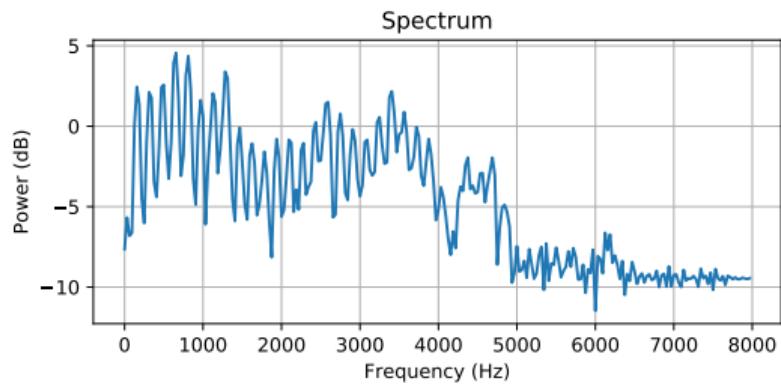
Vowel variability
single speaker, all contexts



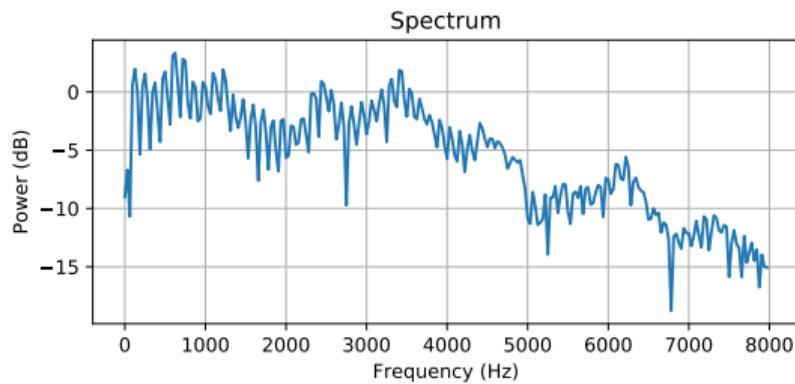
Source: <https://www.phon.ucl.ac.uk/resource/sfs/howto/formant.php>

Question: which is the speech sound with higher f_0 ?

a)

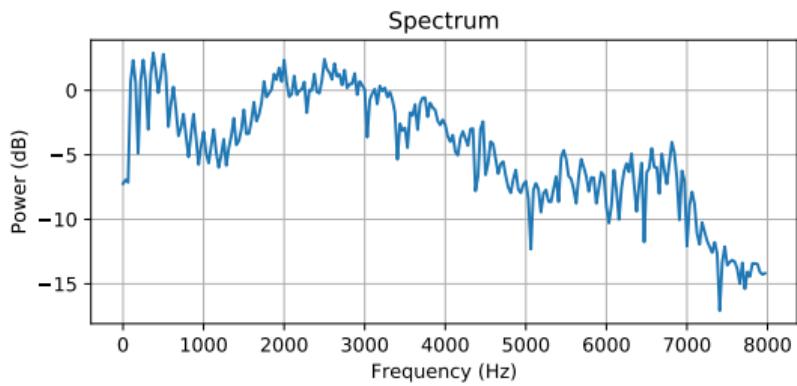


b)

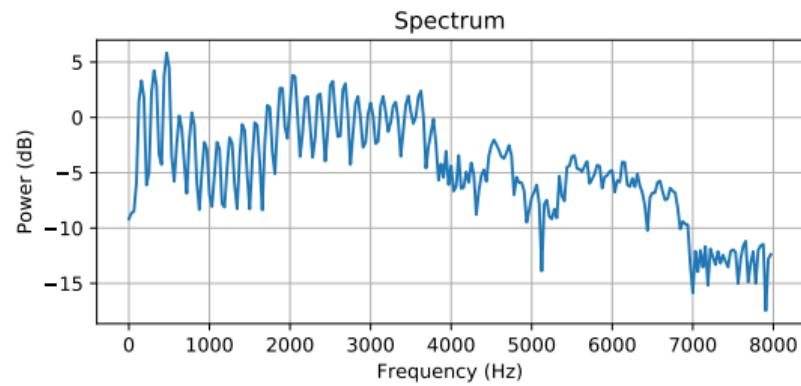


Question: which is the speech sound with higher f_0 ?

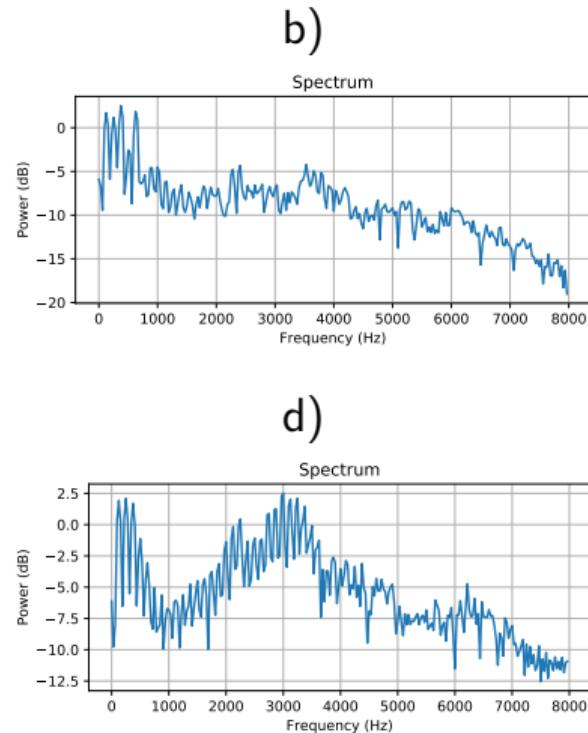
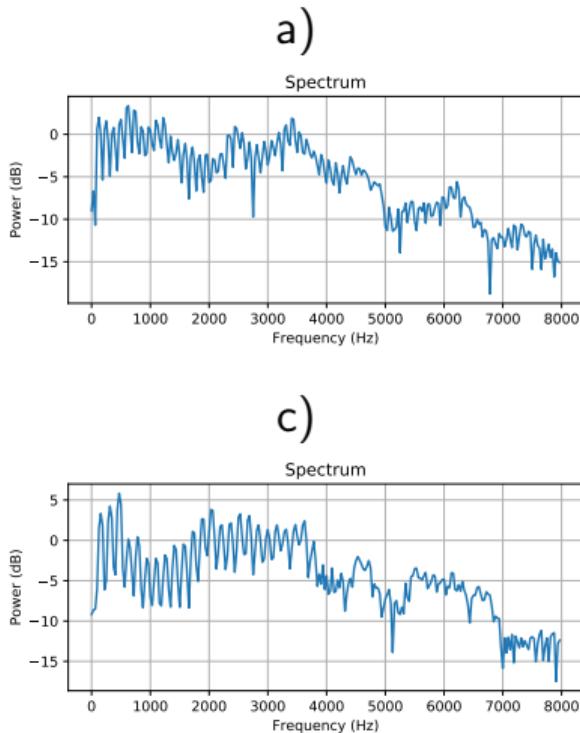
a)



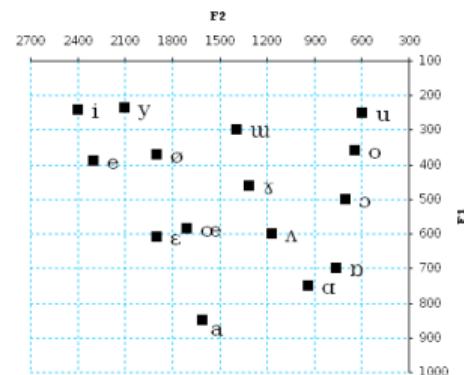
b)



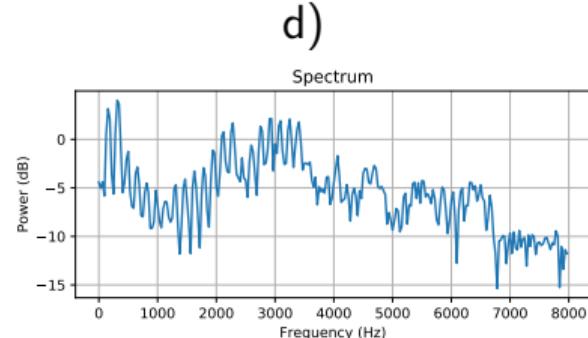
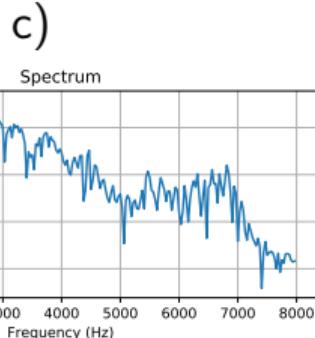
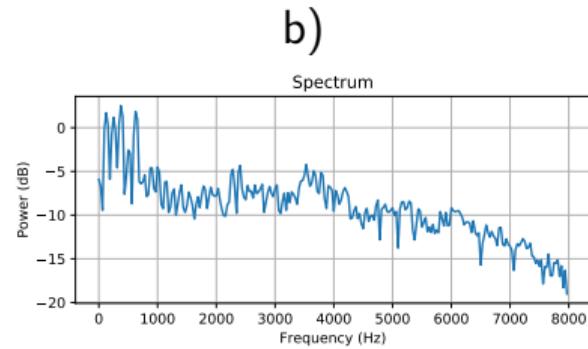
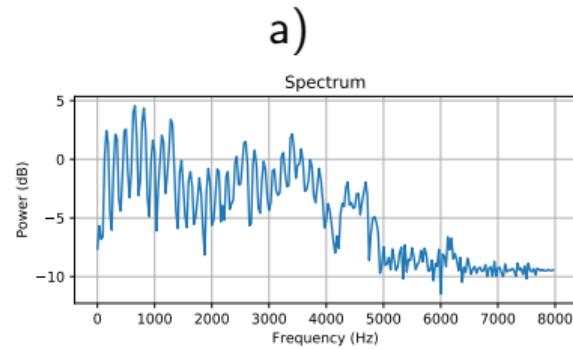
Question: which spectrogram depicts an /a/?



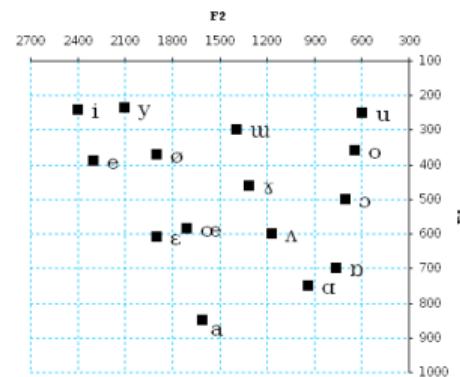
Average formant frequencies



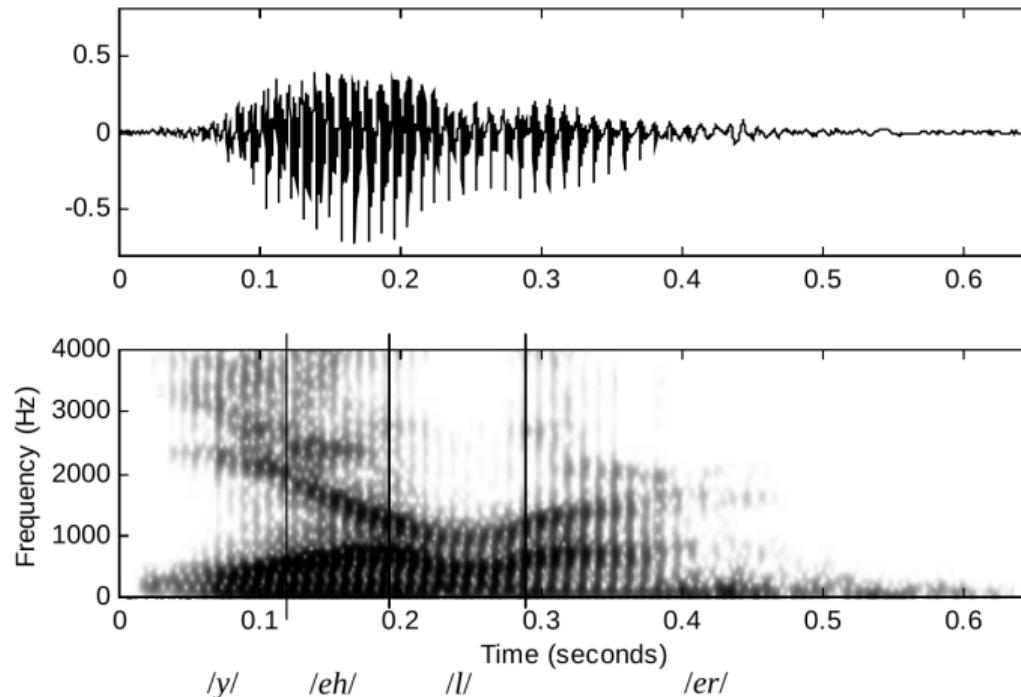
Question: which spectrogram depicts an /i/?



Average formant frequencies

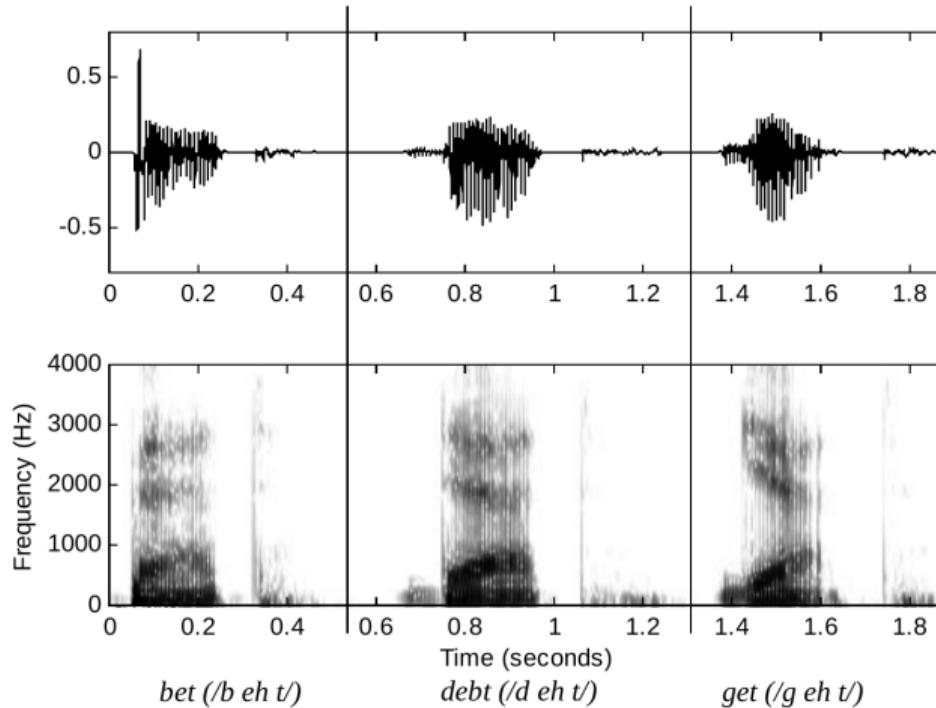


Coarticulation: vowels-semivowels



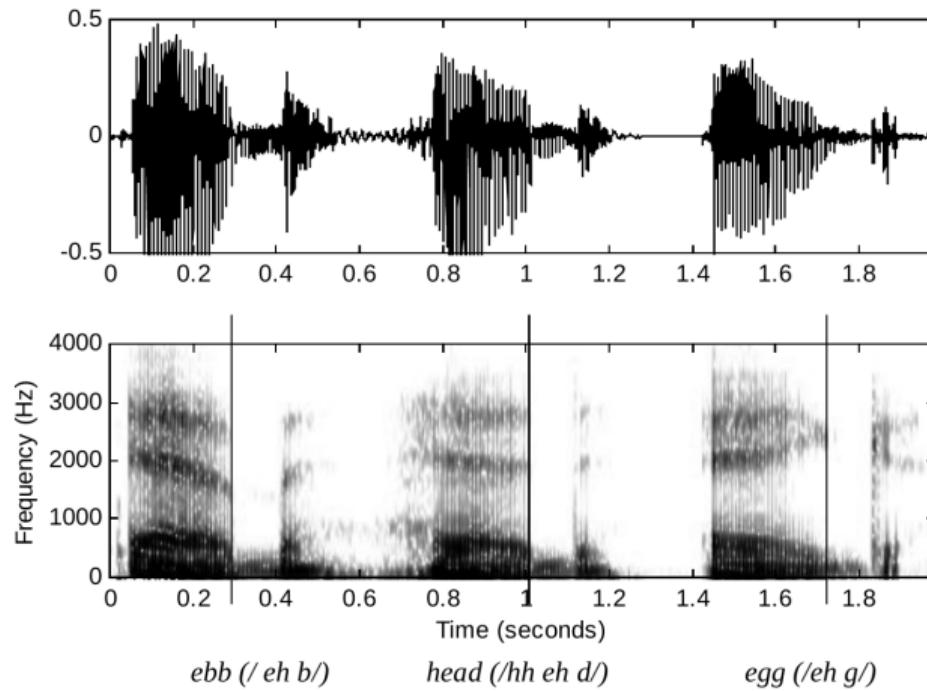
from Huang, Acero and Hon's book

Coarticulation: plosive-vowel



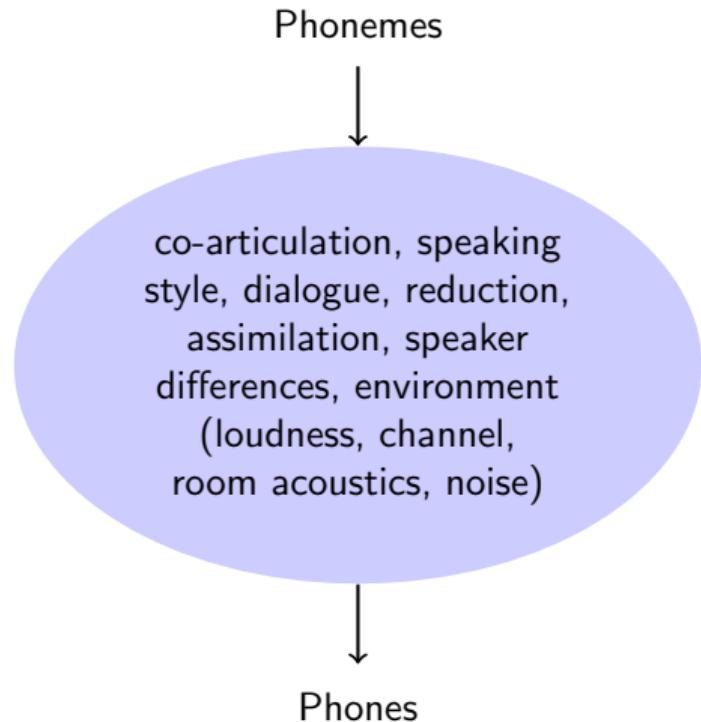
from Huang, Acero and Hon's book

Coarticulation: vowel-plosive

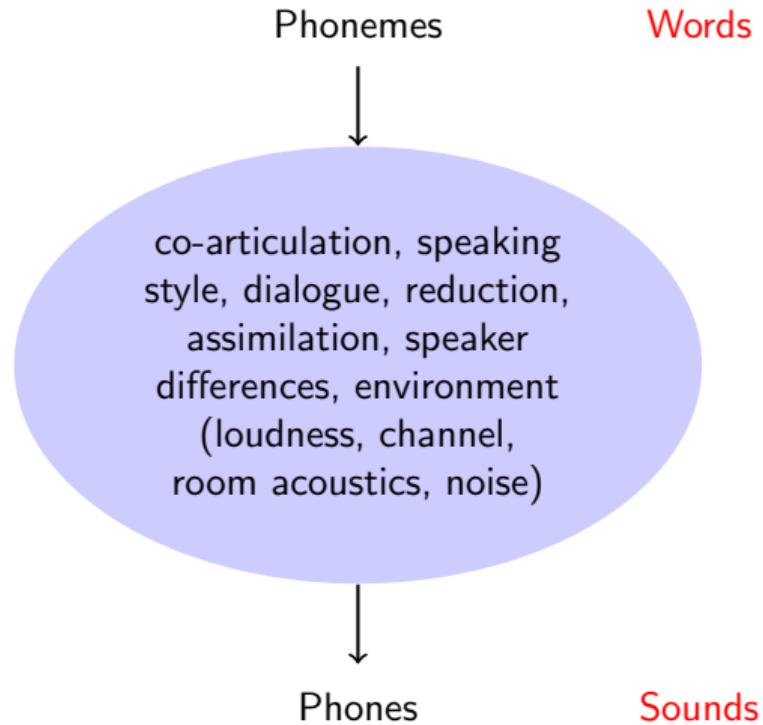


from Huang, Acero and Hon's book

Phonology vs Phonetics



Phonology vs Phonetics



Outline

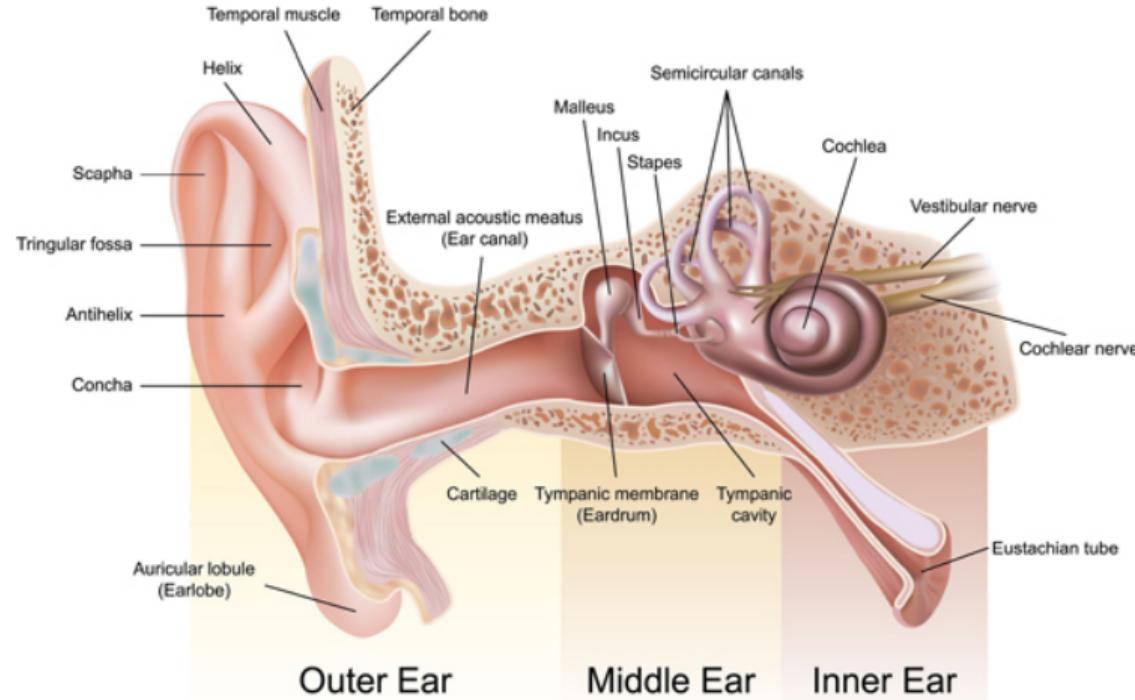
1 Speech Production

- Source/Filter Model

2 Speech Perception

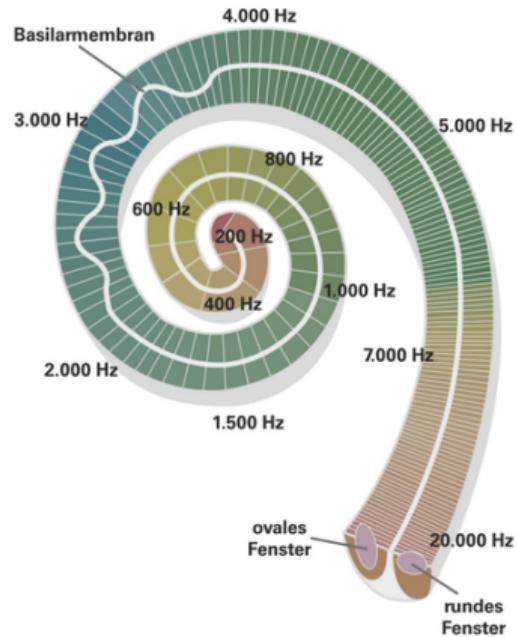
3 Challenges

Anatomy of the ear



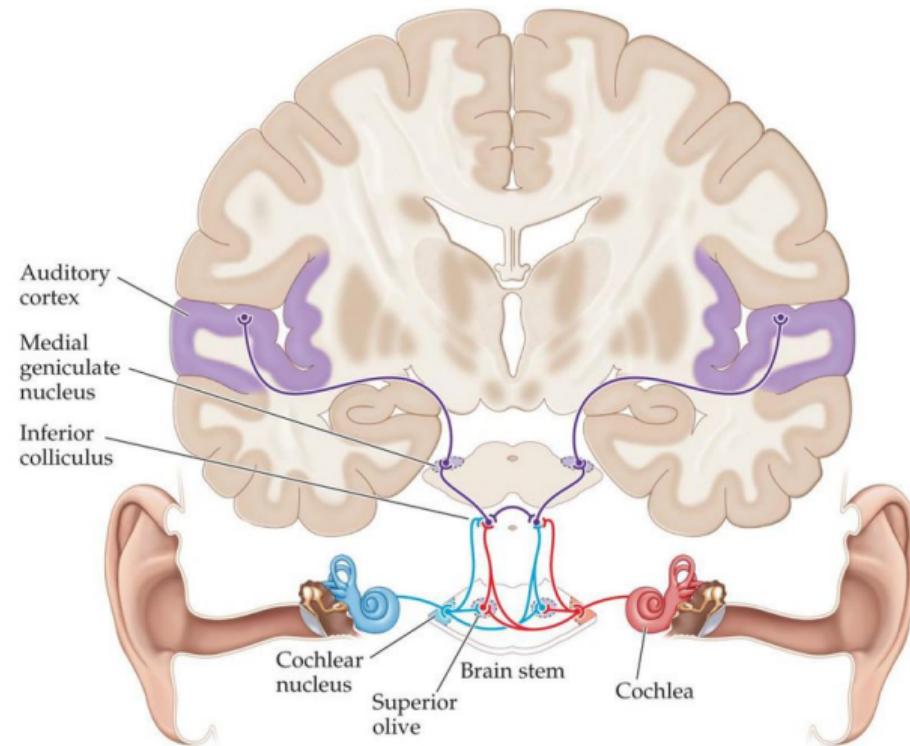
from <https://drjillgordon.com/how-hearing-works>

Cochlea



from https://tu-dresden.de/ing/elektrotechnik/ias/aha/forschung/akustik/psychoakustik?set_language=en

Auditory Pathways

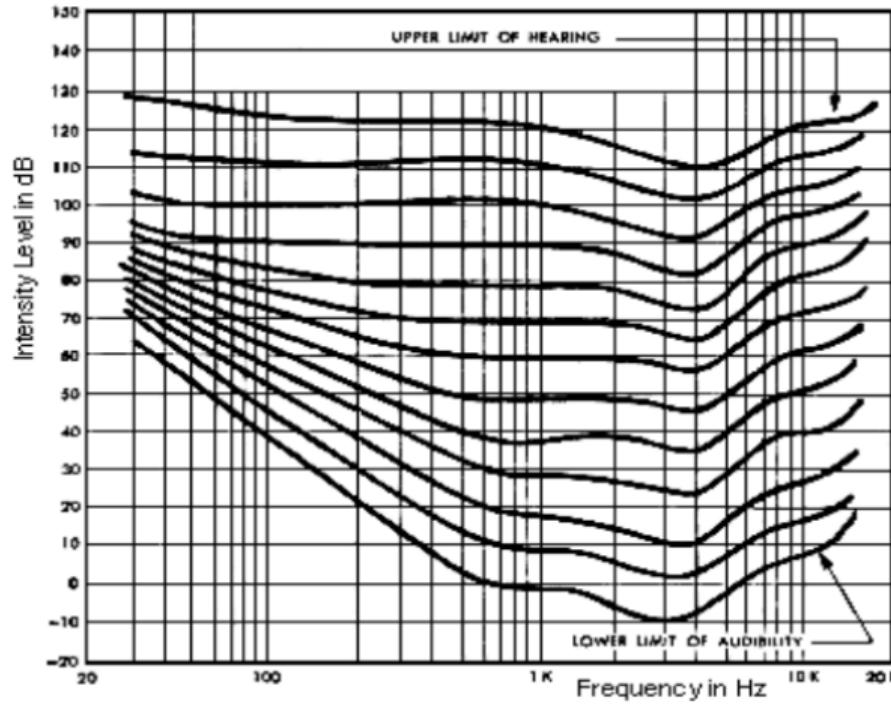


SENSATION & PERCEPTION 5e, Figure 9.20
© 2018 Oxford University Press

Physical vs perceptual attributes

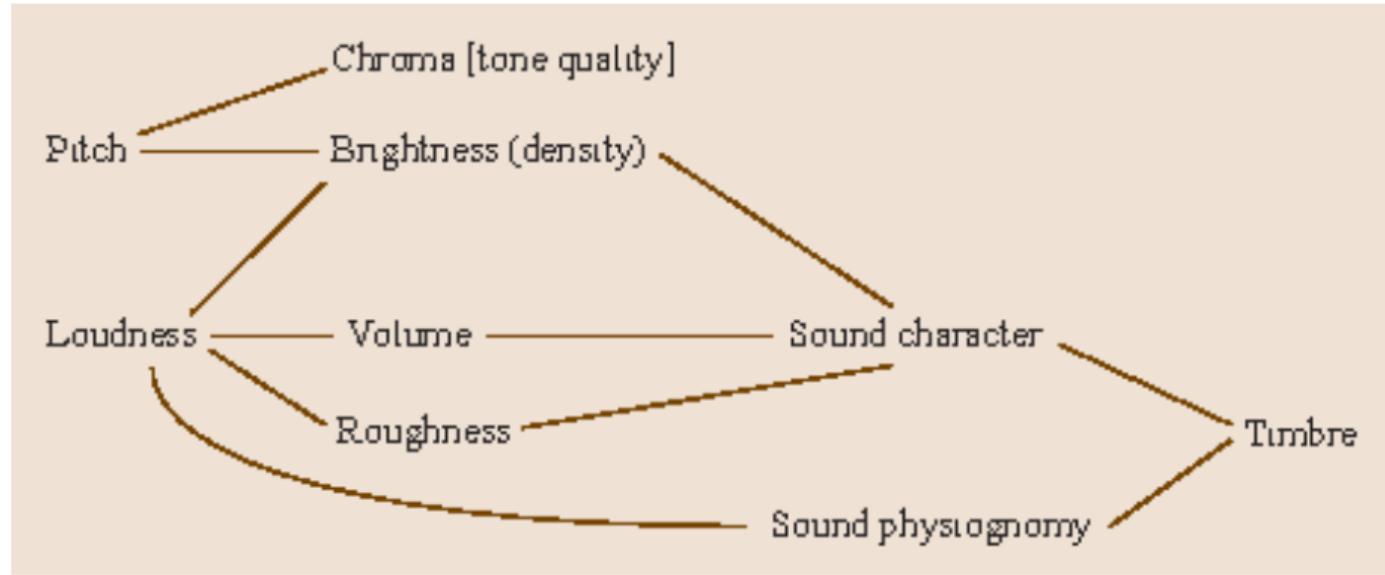
Physical Quantity	Perceptual Quantity
Intensity (SPL)	Loudness
Fundamental frequency (f_0)	Pitch
Spectral shape	Timbre
Onset/offset time	Timing
Phase difference in binaural hearing	Location

Equal-loudness curves (pure tones)



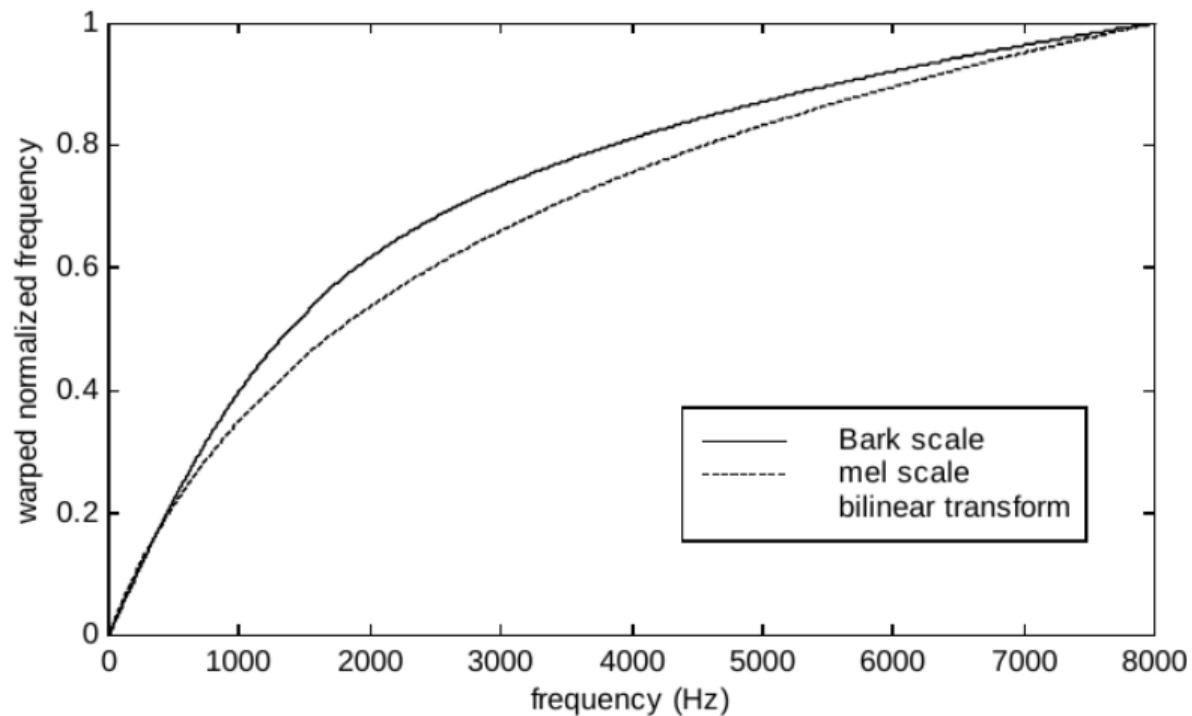
from Huang, Acero and Hon's book

Loudness perception (complex sounds)



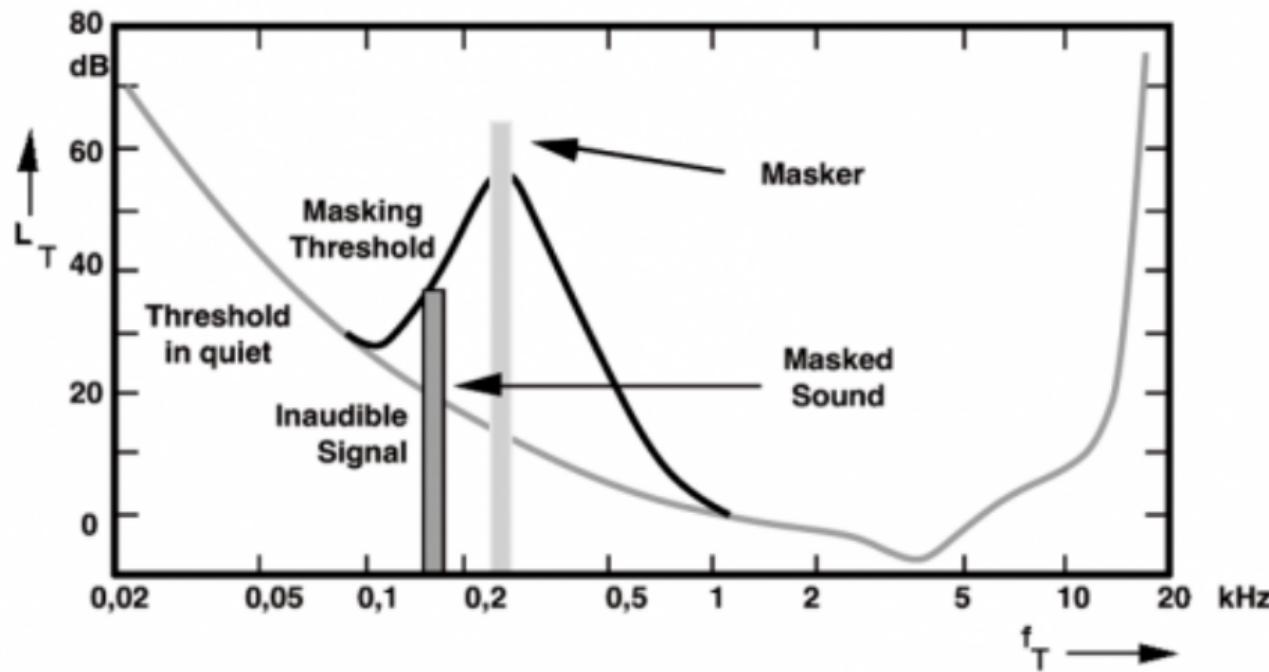
from https://link.springer.com/chapter/10.1007/978-3-662-55004-5_33

Bark and Mel frequency scales



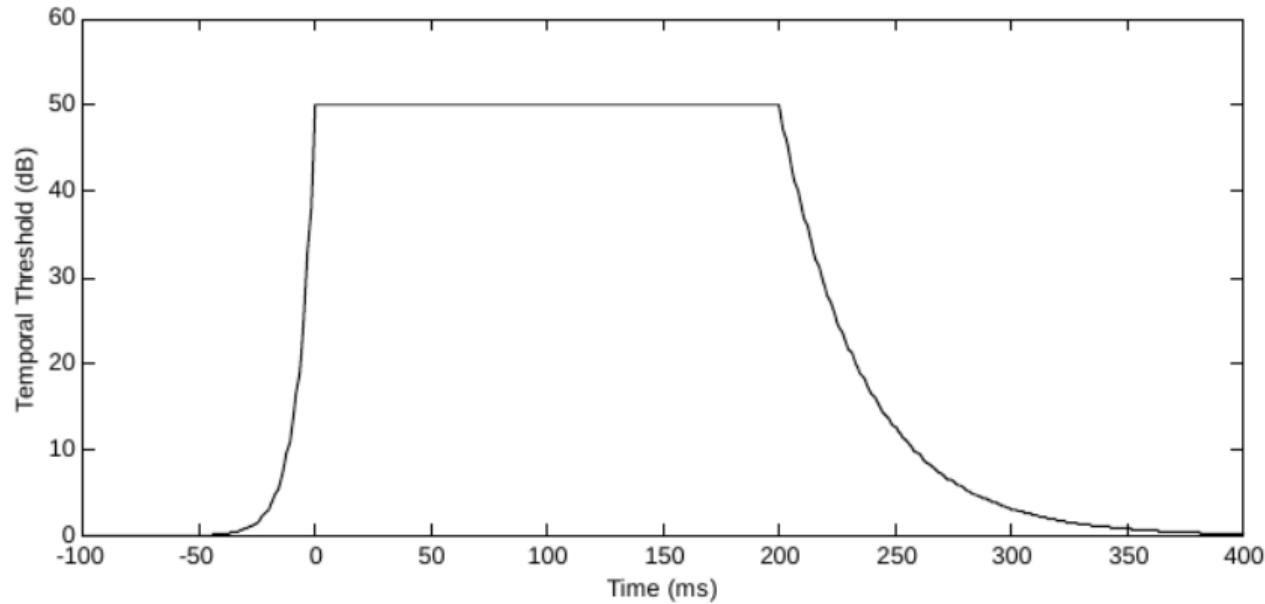
from Huang, Acero and Hon's book

Masking in frequency



from <http://hephaestusaudio.com/delphi/category/psychoacoustics/>

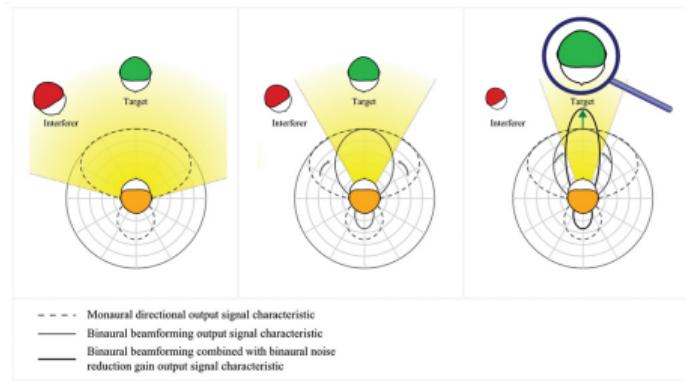
Masking in time



from Huang, Acero and Hon's book

Binaural hearing

- mainly used for localization
- main cue: inter-aural time difference (ITD)
- main cue: inter-aural level difference (ILD)
- filtering from ear pinna
- head-related transfer function HRTF



Check <http://auditoryneuroscience.com/spatial-hearing/binaural-cues>

Outline

1 Speech Production

- Source/Filter Model

2 Speech Perception

3 Challenges

Challenges — Variability

Between speakers

- Age
- Gender
- Anatomy
- Dialect

Within speaker

- Stress
- Emotion
- Health condition
- Read vs Spontaneous
- Adaptation to environment (Lombard effect)
- Adaptation to listener

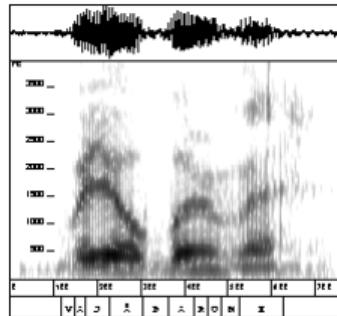
Environment

- Noise
- Room acoustics
- Microphone distance
- Microphone, telephone
- Bandwidth

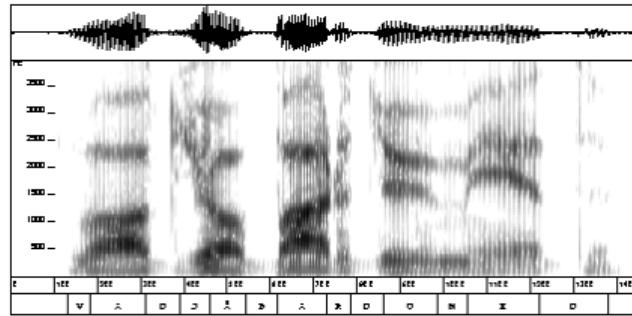
Listener

- Age
- Mother tongue
- Hearing loss
- Known / unknown
- Human / Machine

Example: spontaneous vs hyper-articulated



Va jobbaru me



Vad jobbar du med

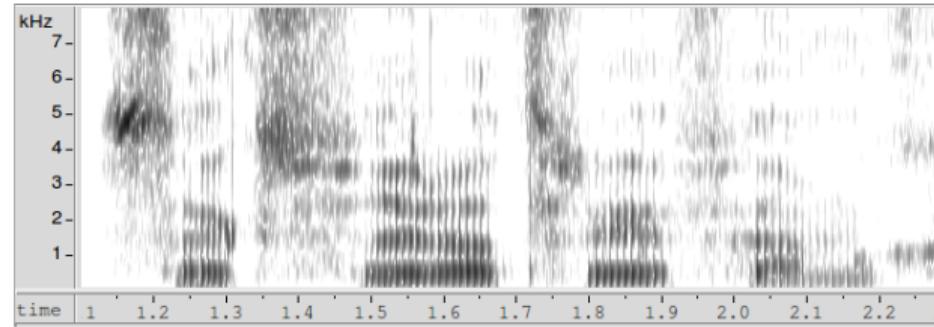
“What is your occupation”
 (“What work you with”)

Examples of reduced pronunciation

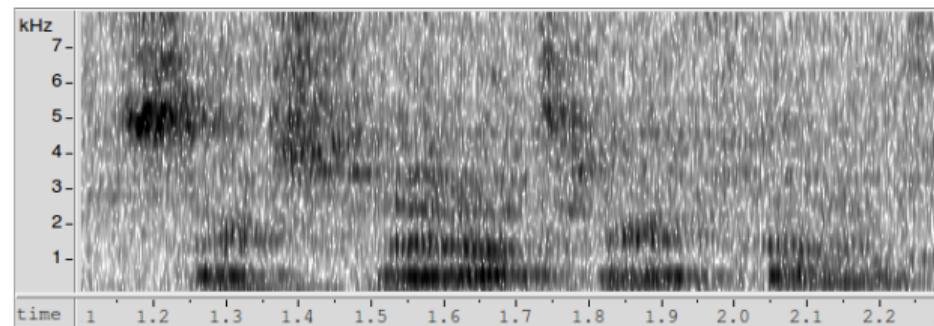
Spoken	Written	In English
Tesempel	Till exempel	for example
åhamba	och han bara	and he just
bafatt	bara för att	just because
javende	jag vet inte	I don't know

Microphone distance

Headset



2 m distance



Speech Analysis and Feature Extraction

TTT4185 Machine Learning for Signal Processing

Giampiero Salvi

Department of Electronic Systems
NTNU

HT2020

Outline

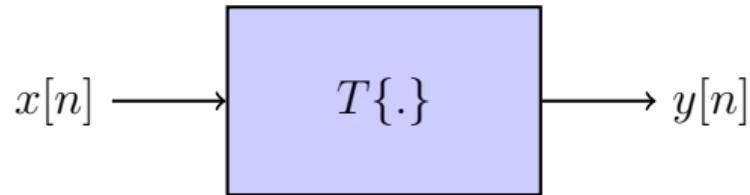
1 Speech Signal Representations

- Signal Processing Reminder
- Sampling and Quantization
- Pre-Emphasis
- Windowing
- Discrete Fourier Transform

2 Feature Extraction

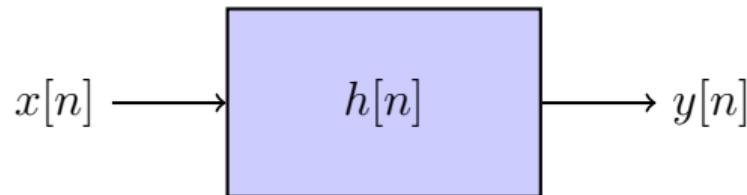
- Linear Prediction Analysis (LPA)
- Cepstrum
- Perceptually Motivated Features

Assuming that you know



- linear time-invariant systems (continuous and discrete time case)
- convolution
- impulse response
- Fourier transform and transfer function

Convolution and Impulse Response



$$y[n] = T\{x[n]\} = \sum_{k=-\infty}^{\infty} x[k]h[n-k] = x[n] * h[n]$$

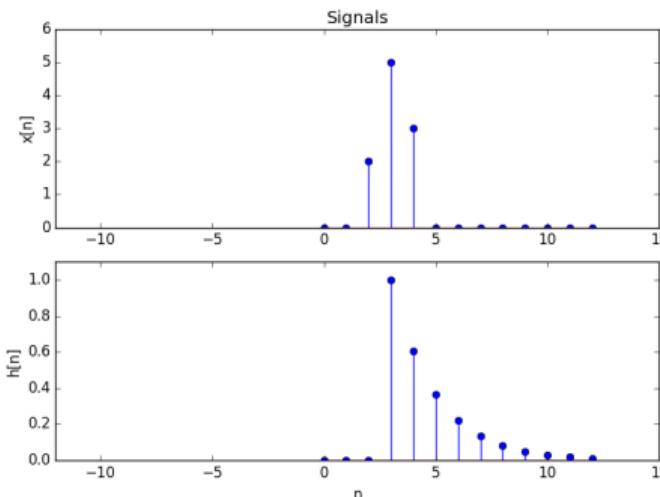
Properties:

- linearity: $(a_1x_1 + a_2x_2) * h = a_1(x_1 * h) + a_2(x_2 * h)$
- symmetry: $x * h = h * x$

Kind of complicated to interpret.

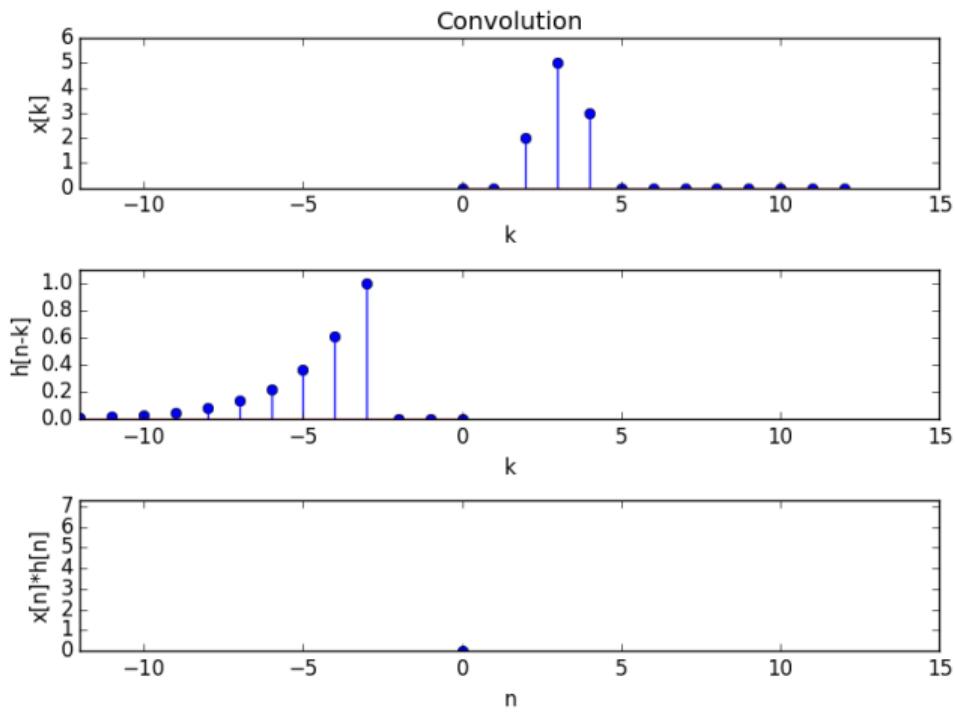
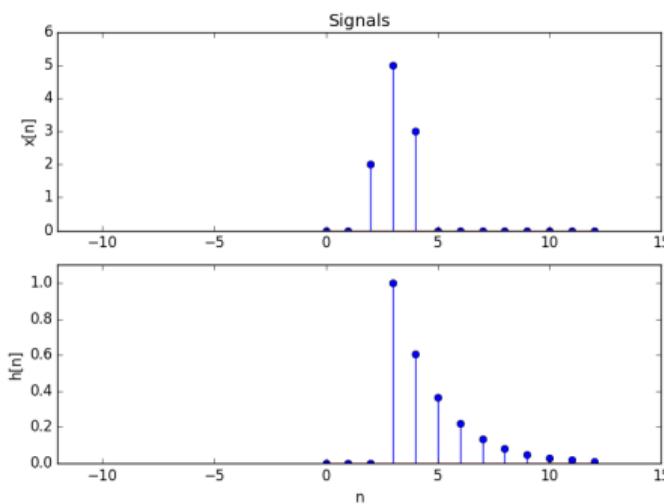
Convolution: Illustration

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



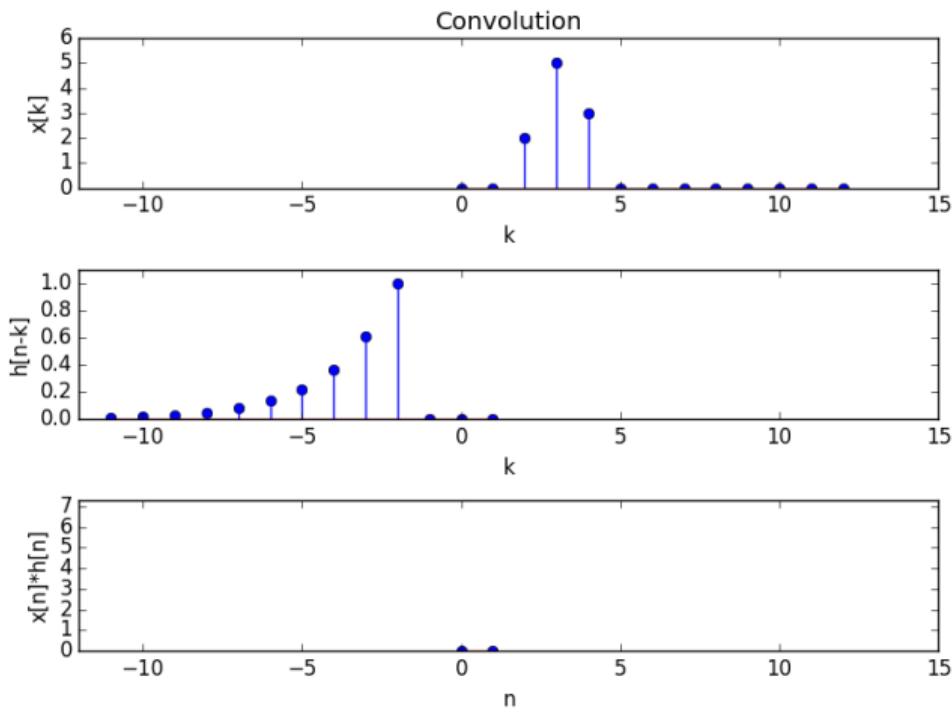
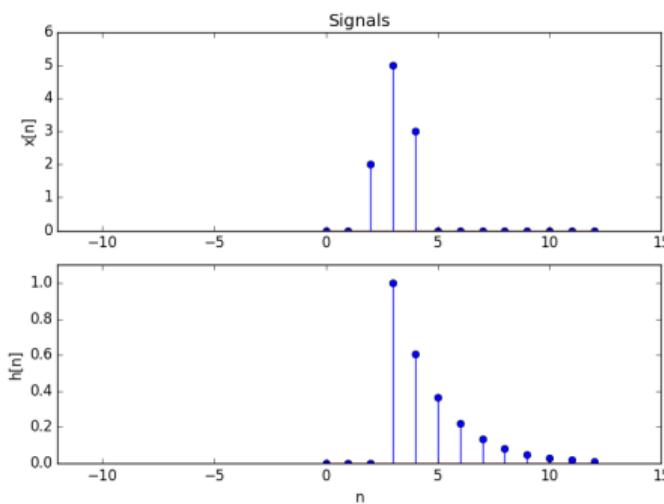
Convolution: Illustration

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



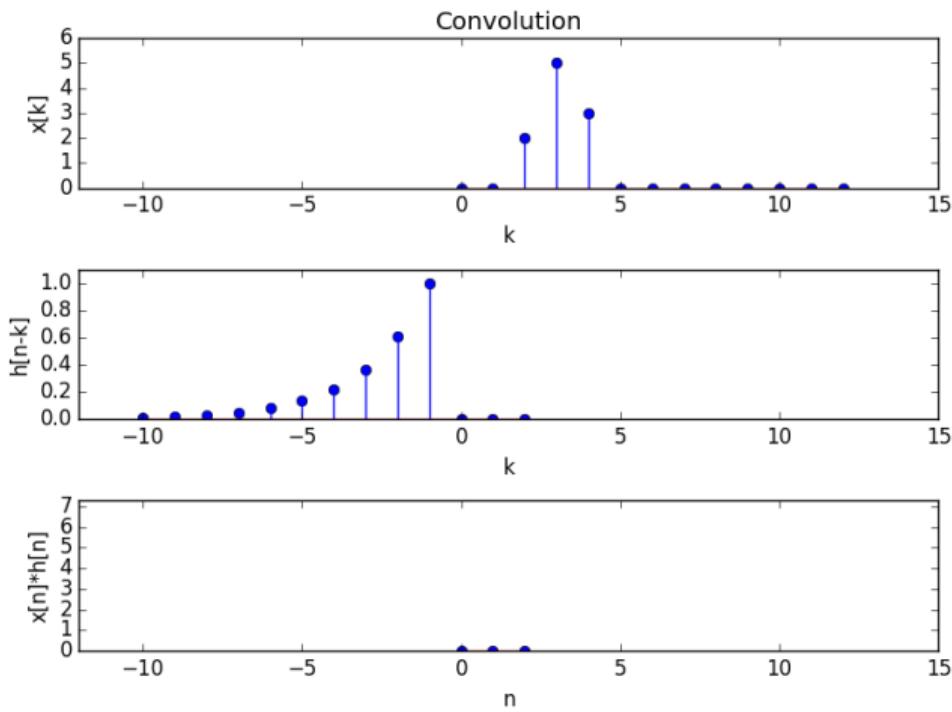
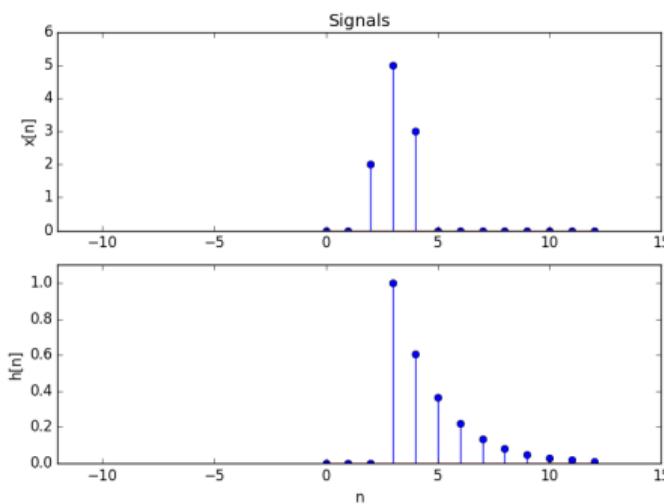
Convolution: Illustration

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



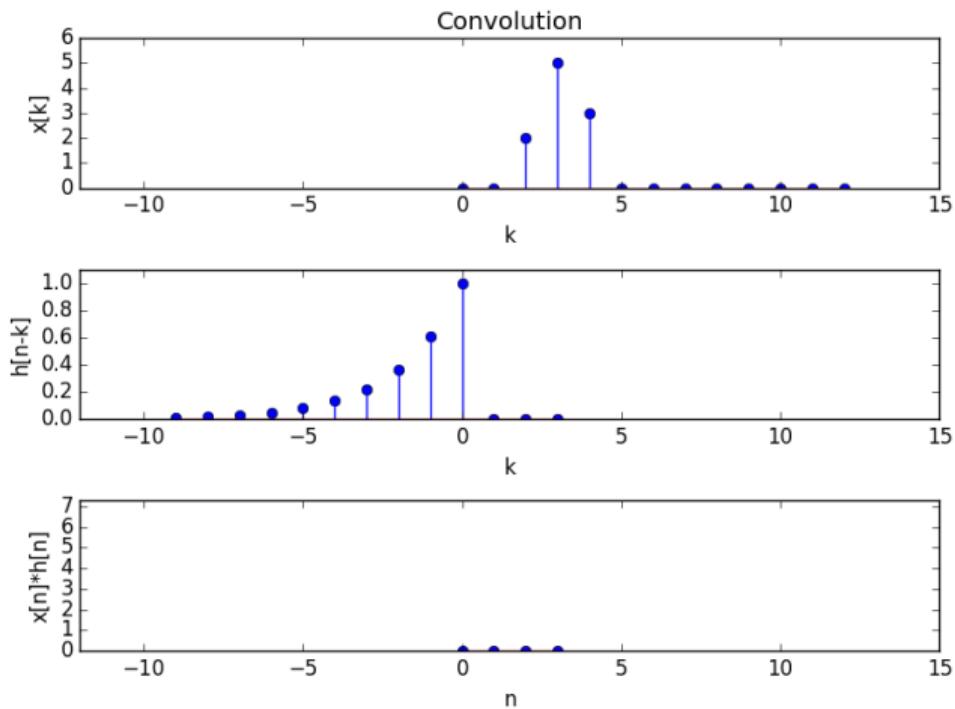
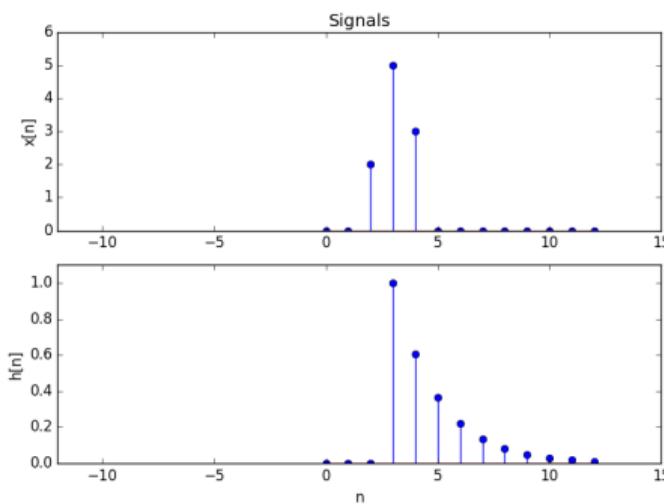
Convolution: Illustration

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



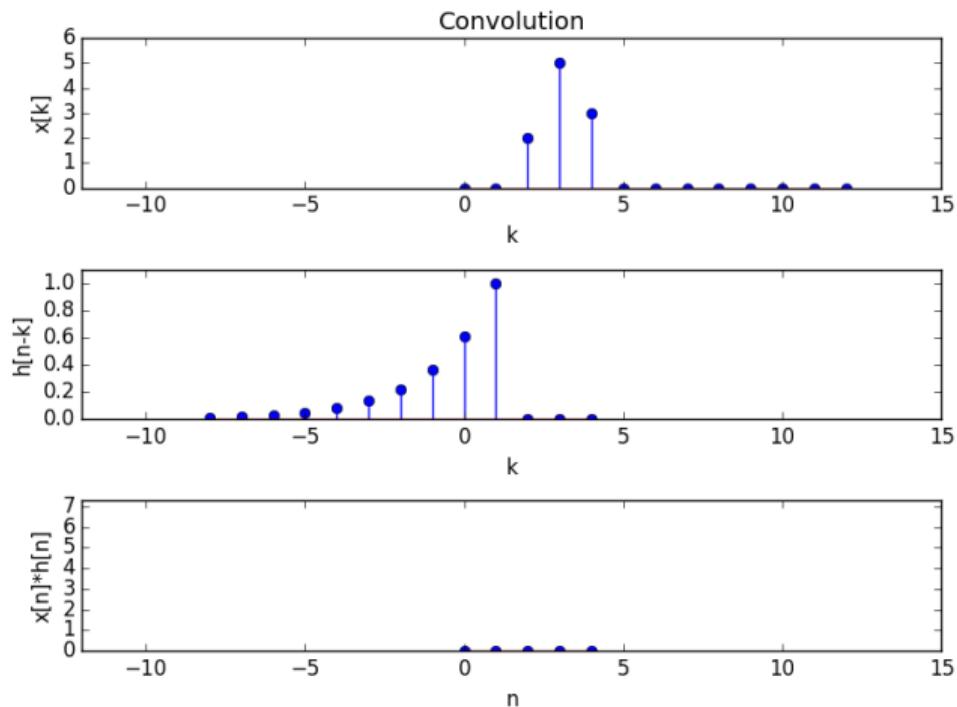
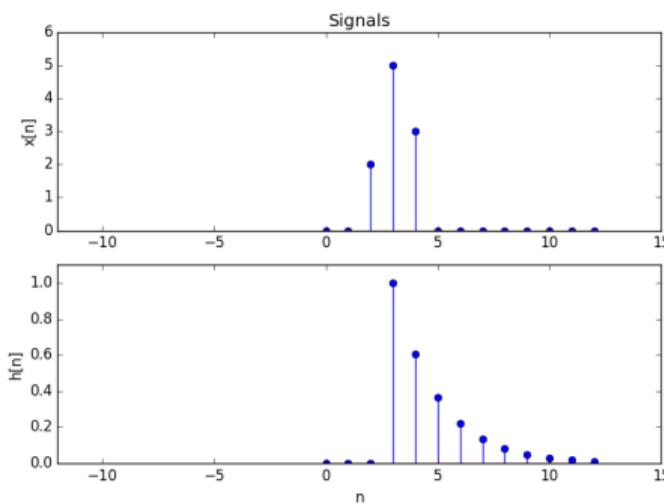
Convolution: Illustration

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



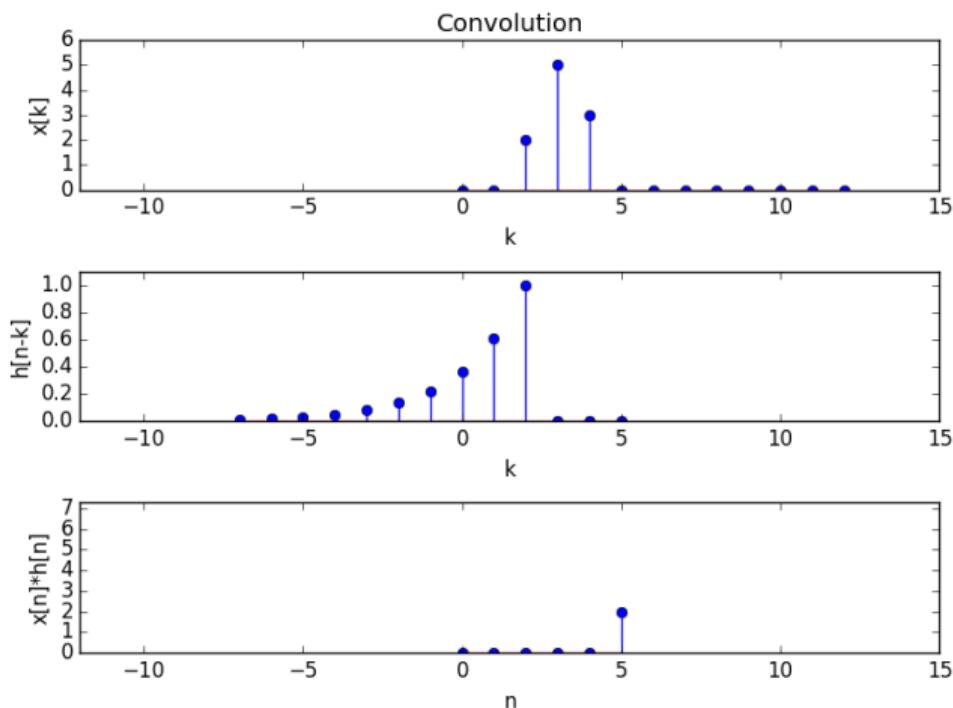
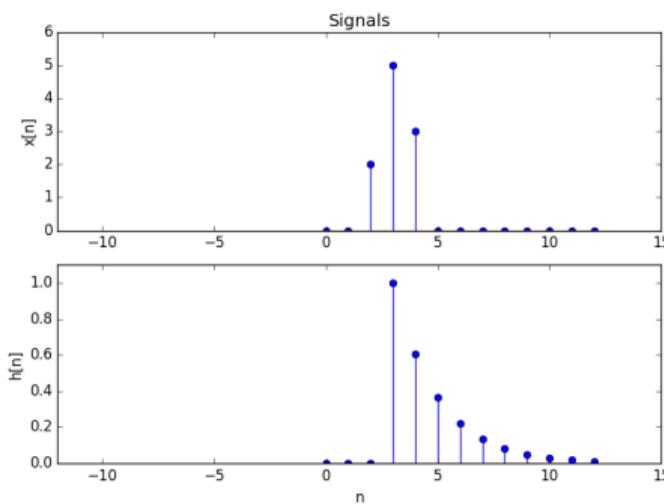
Convolution: Illustration

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



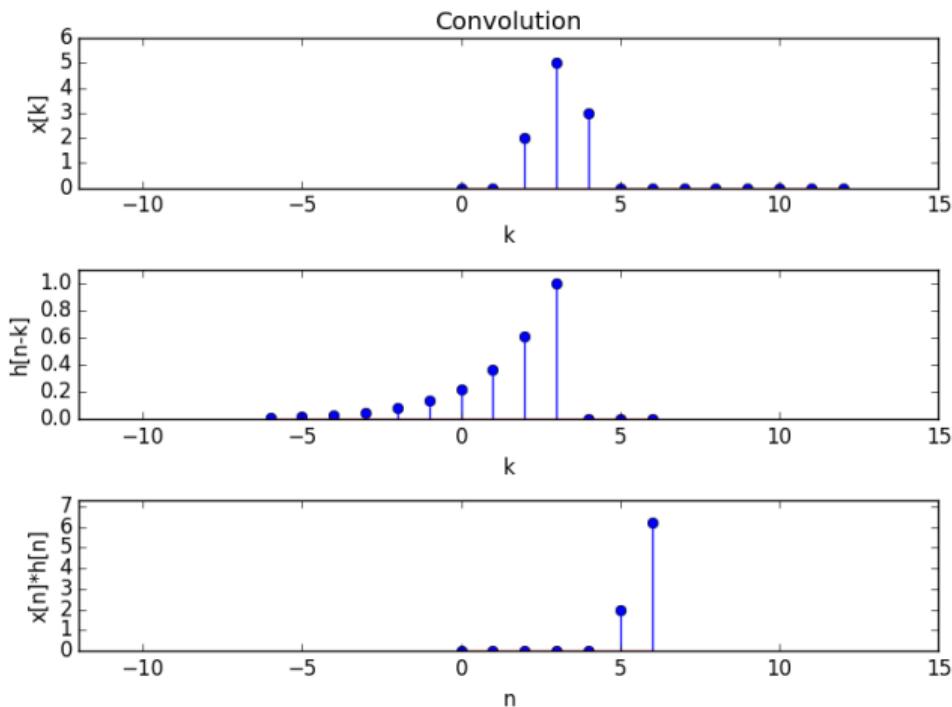
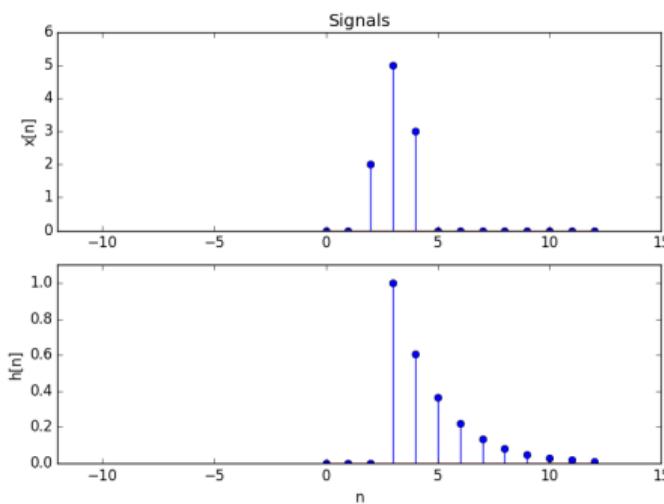
Convolution: Illustration

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



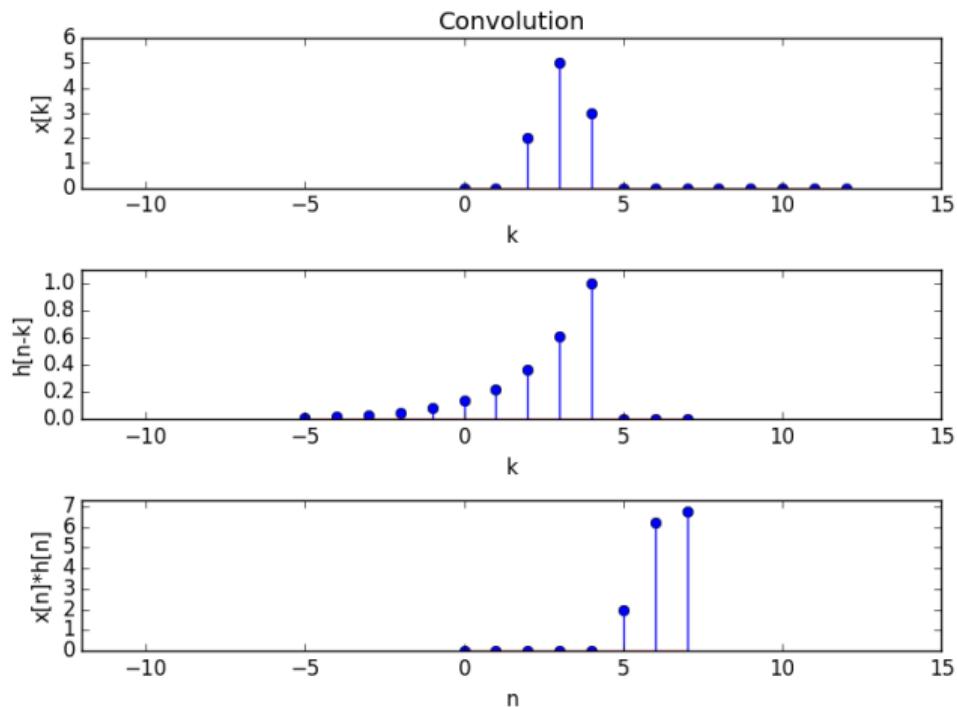
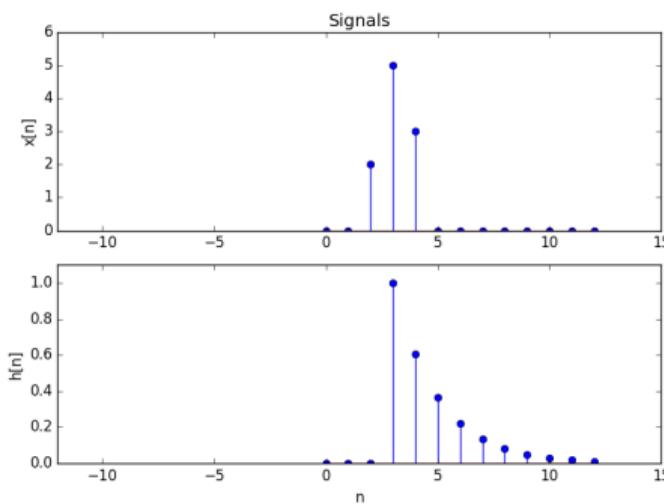
Convolution: Illustration

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



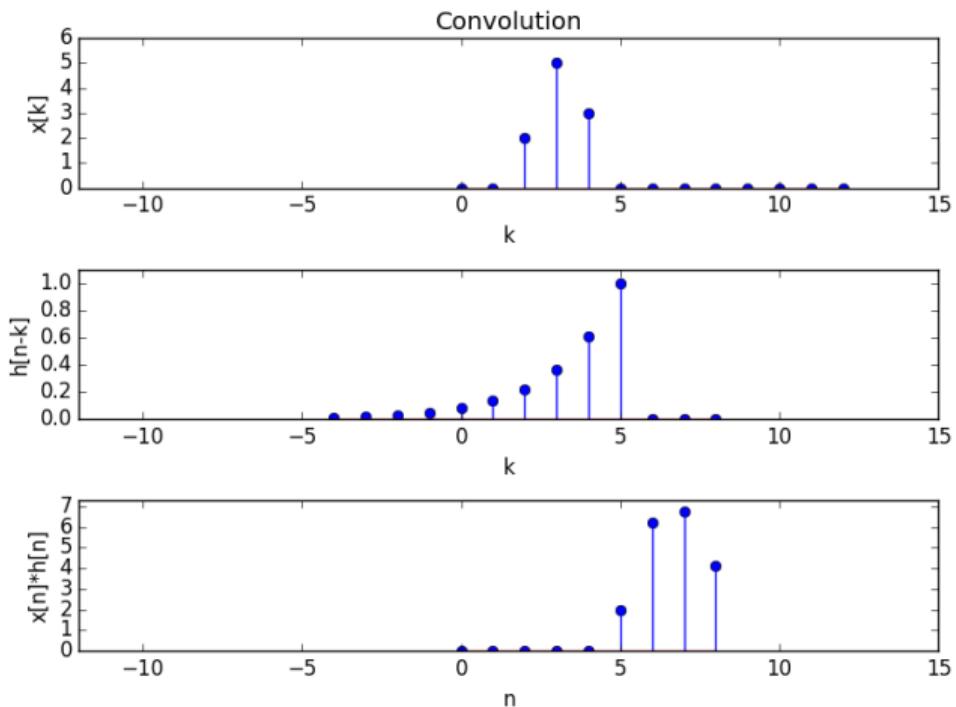
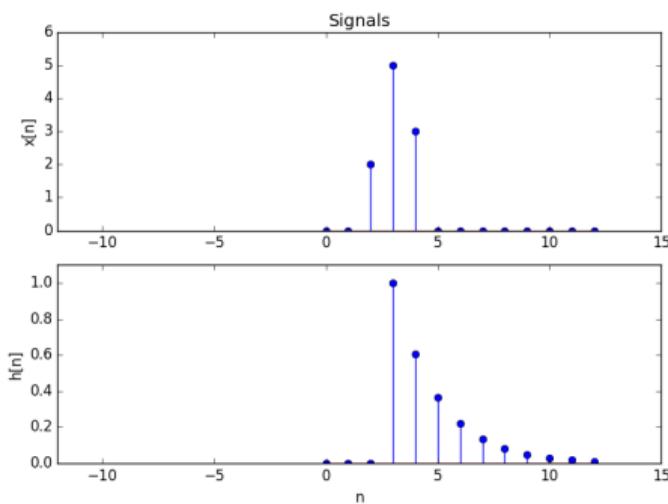
Convolution: Illustration

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



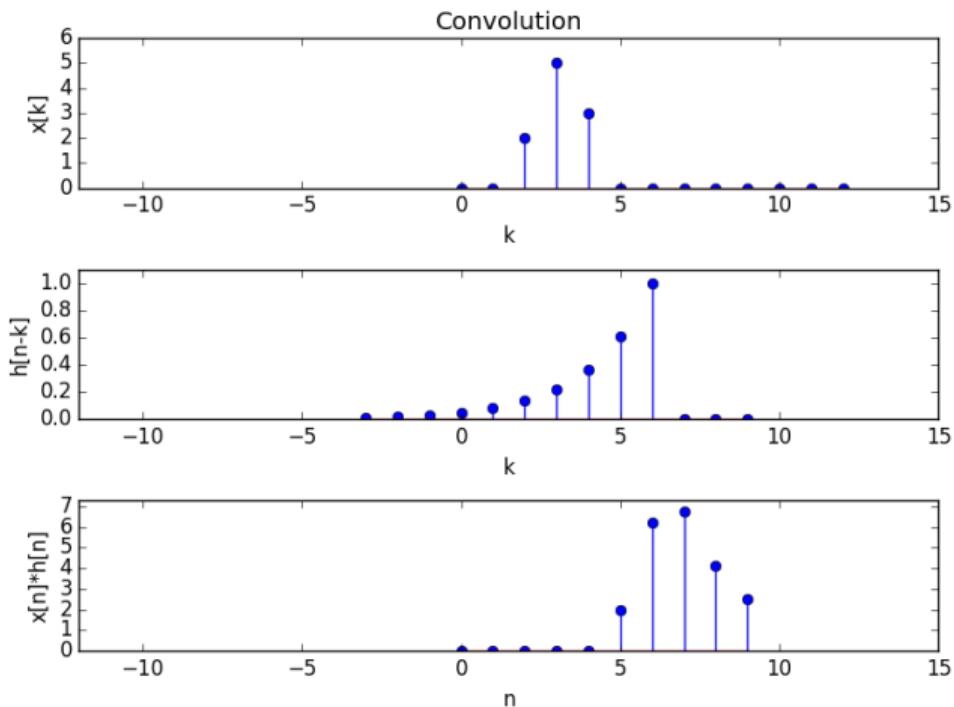
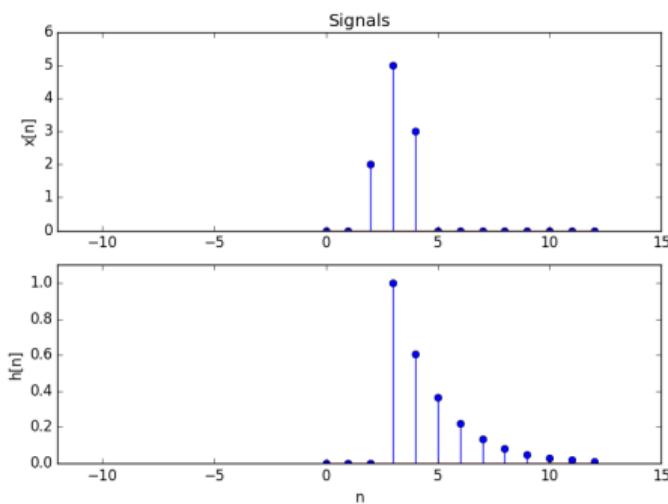
Convolution: Illustration

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



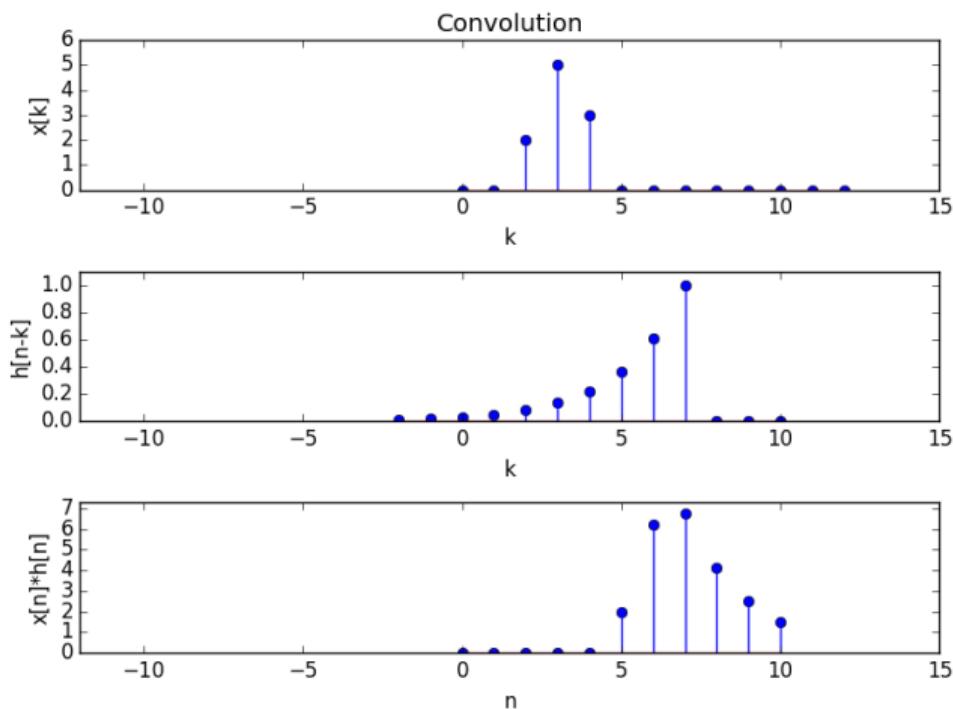
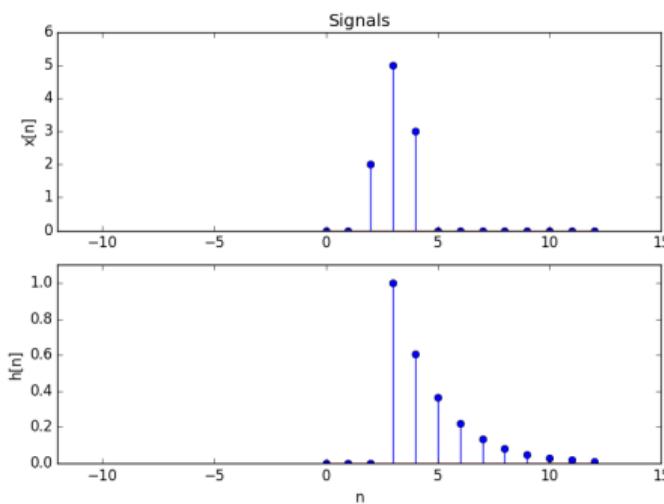
Convolution: Illustration

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



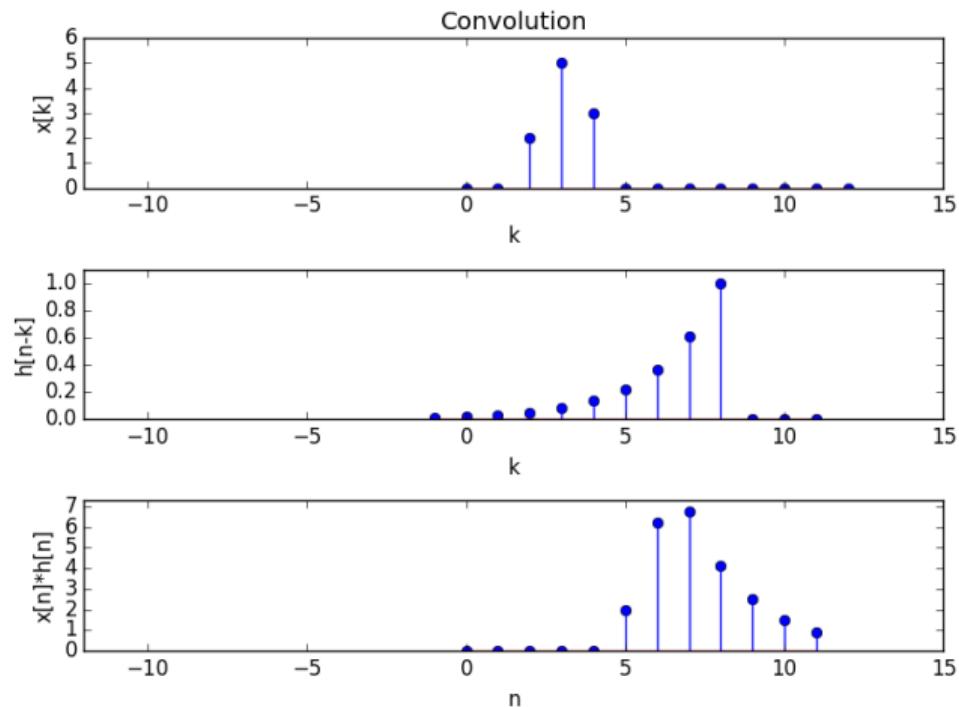
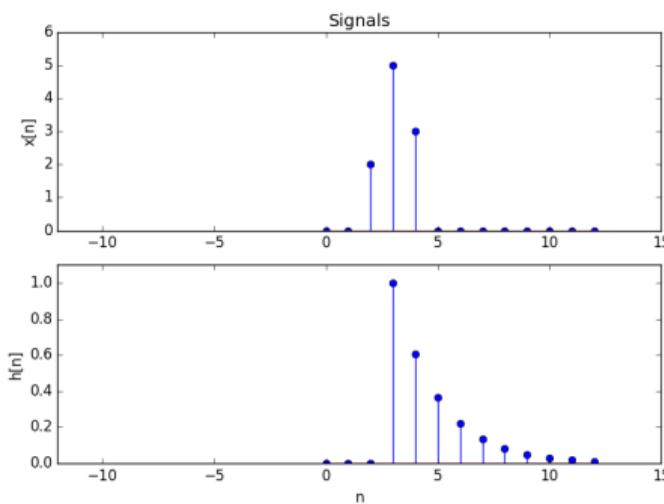
Convolution: Illustration

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



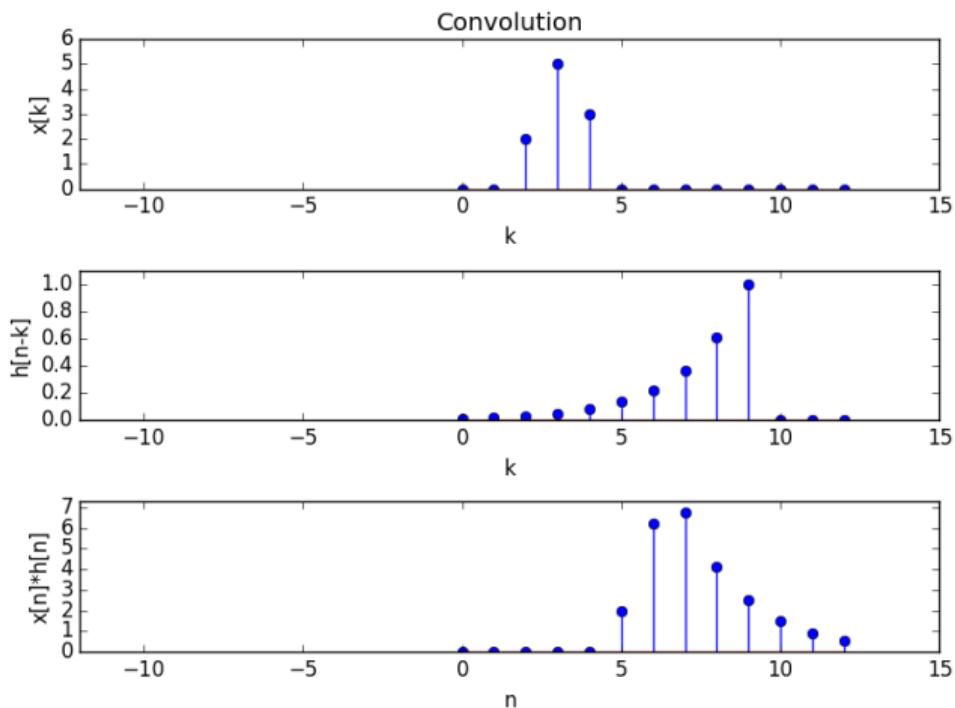
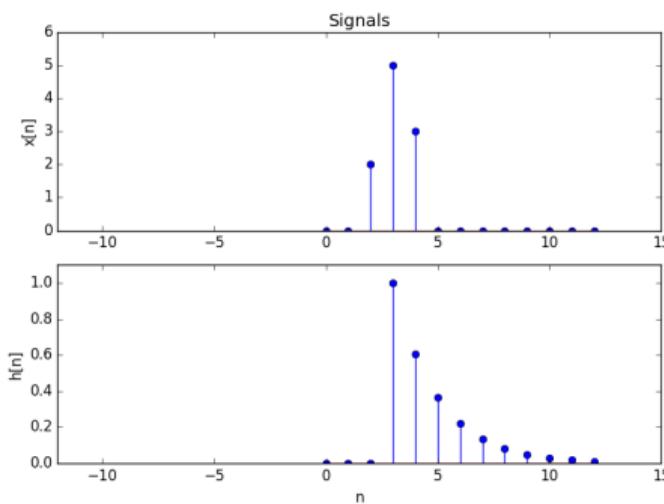
Convolution: Illustration

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



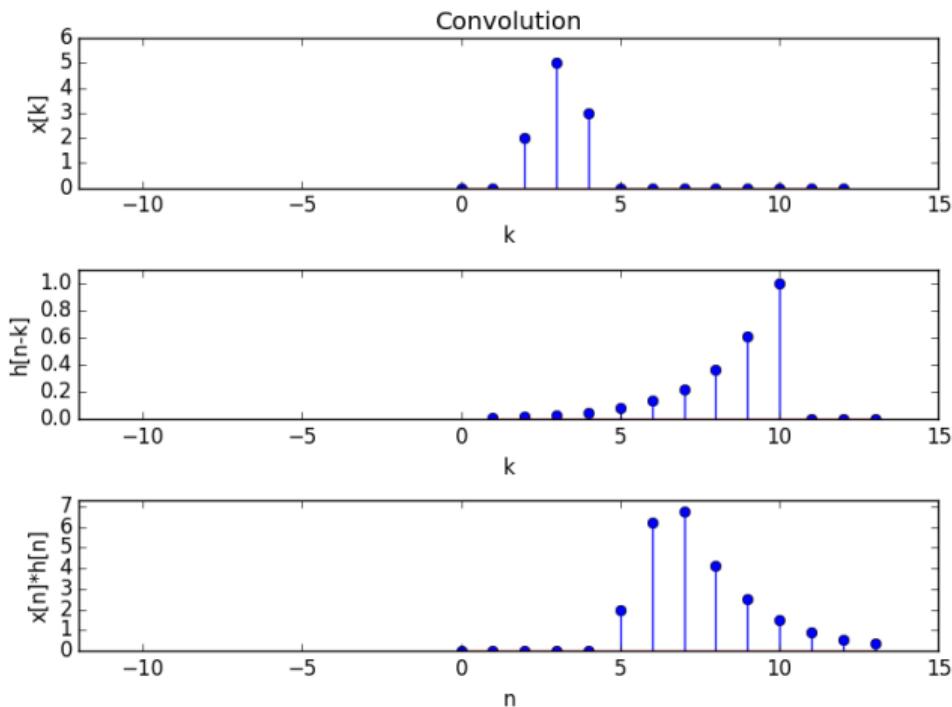
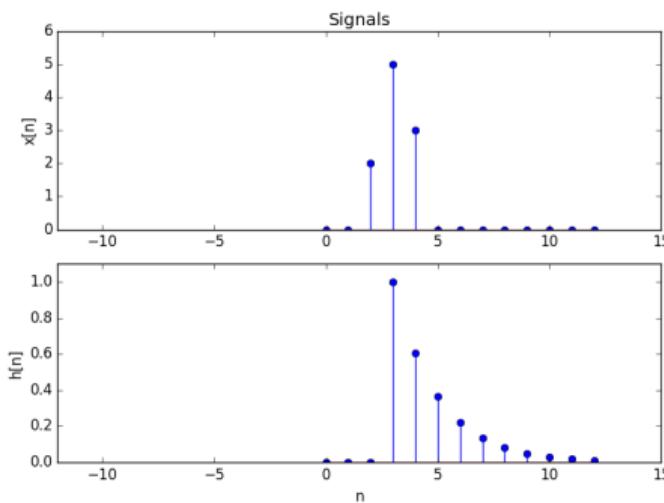
Convolution: Illustration

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



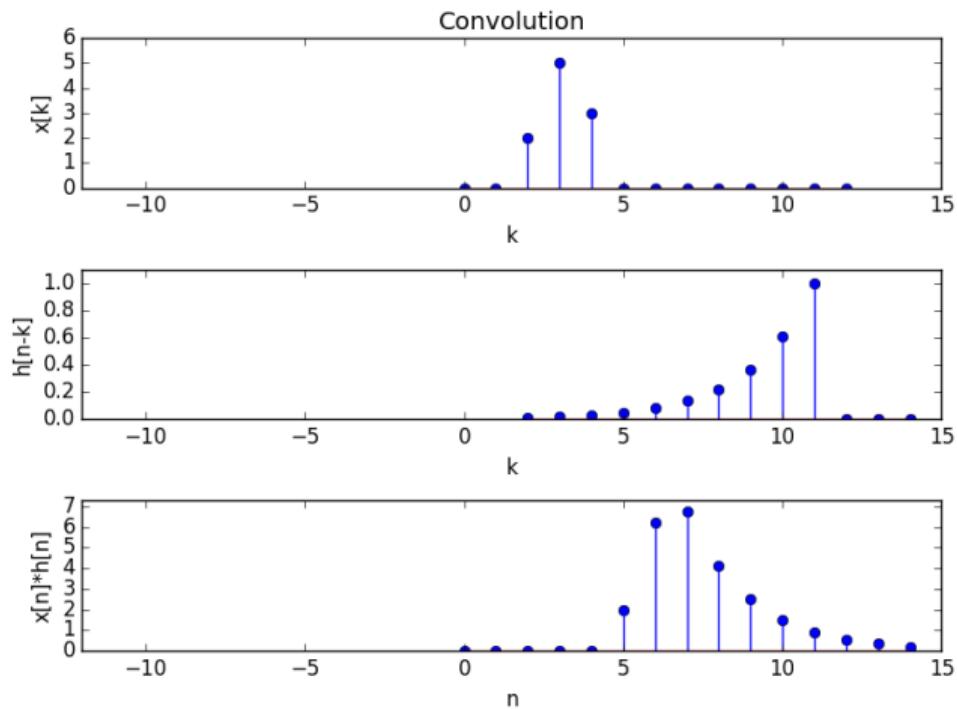
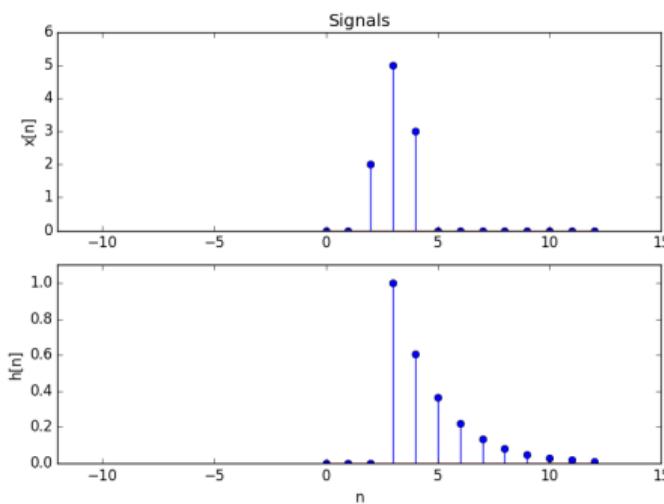
Convolution: Illustration

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



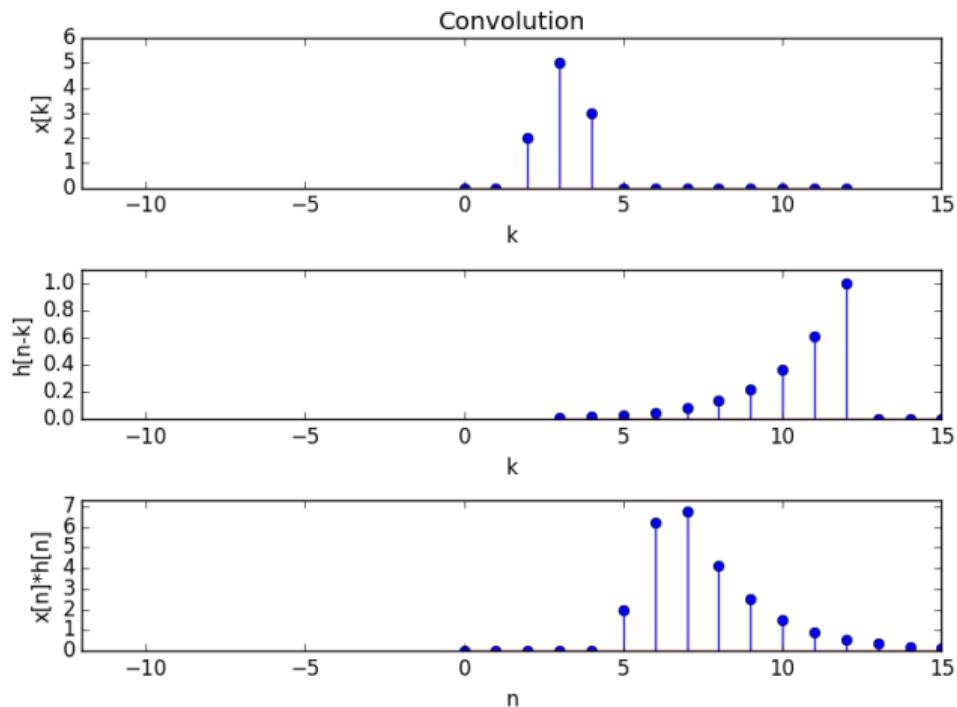
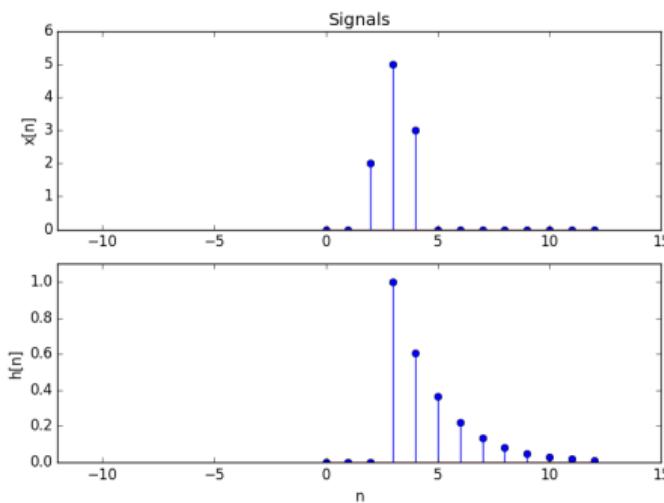
Convolution: Illustration

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



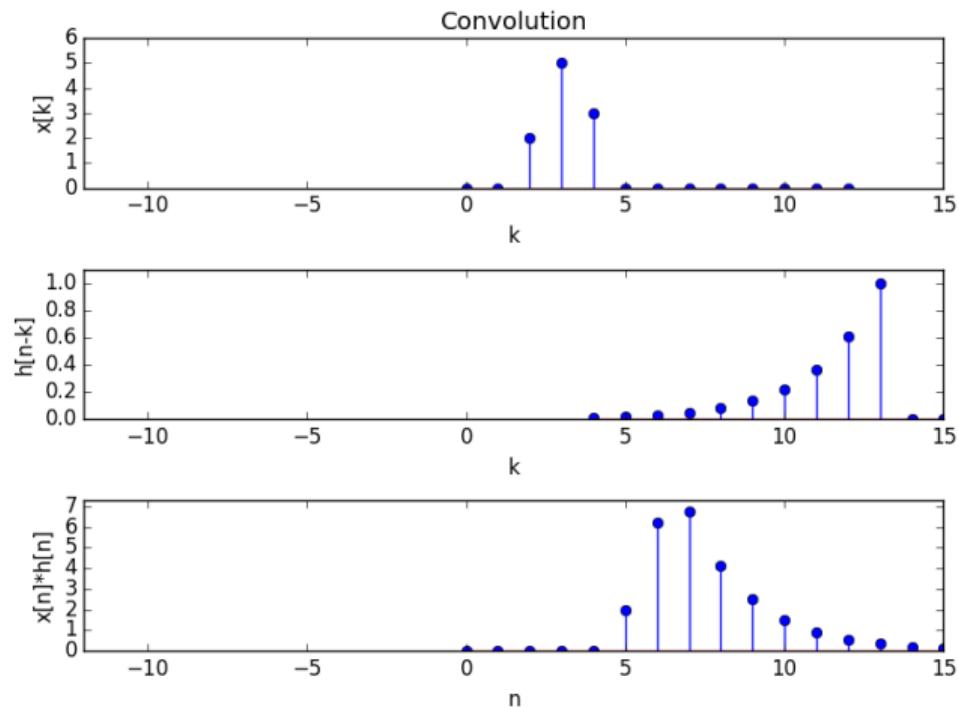
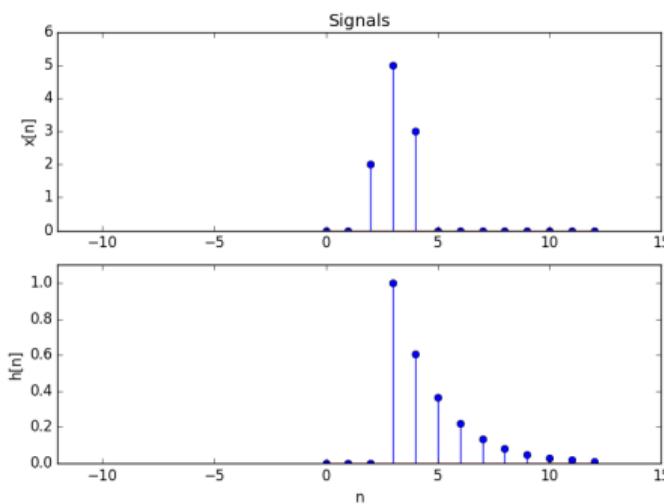
Convolution: Illustration

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



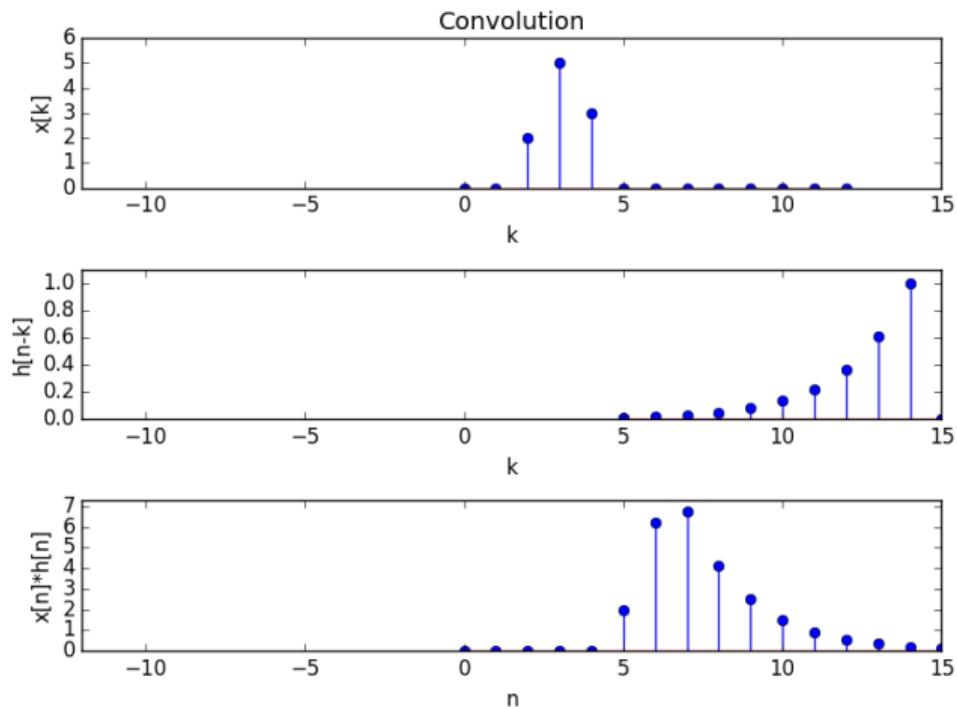
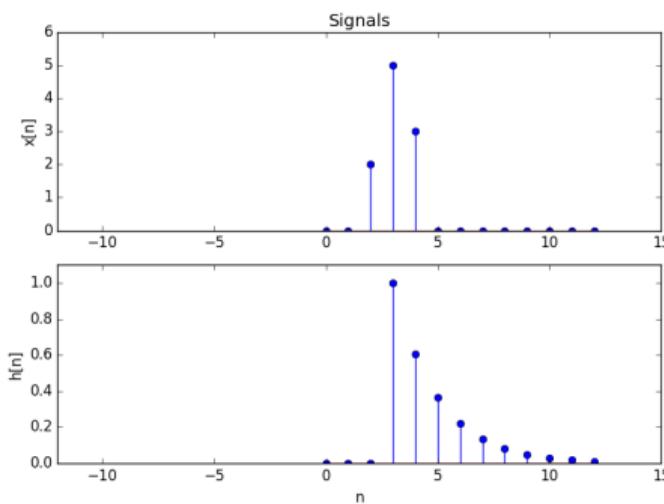
Convolution: Illustration

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



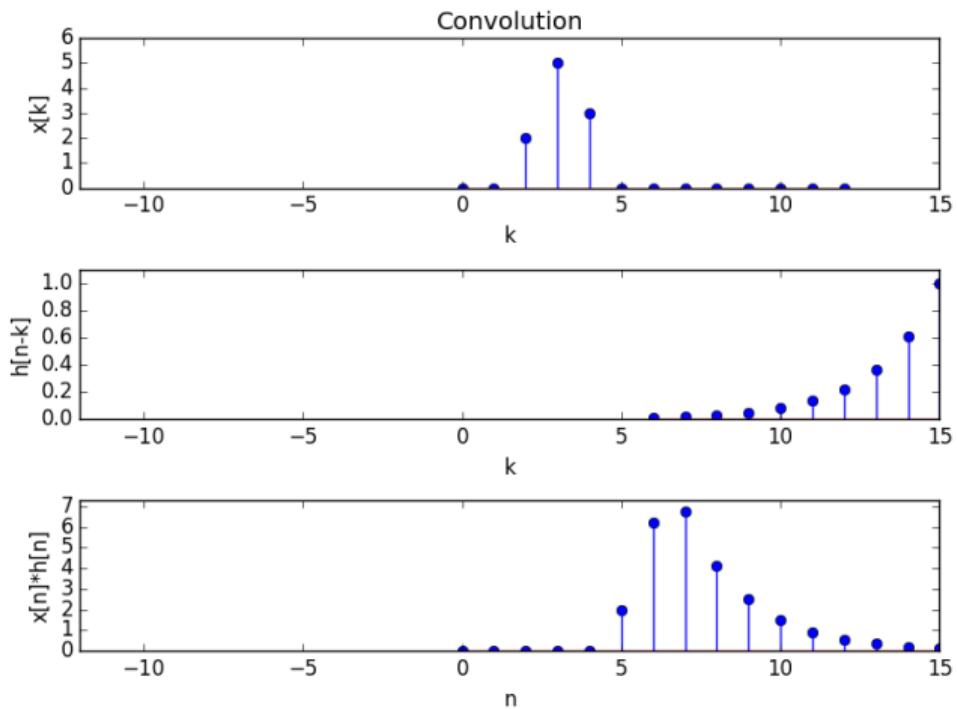
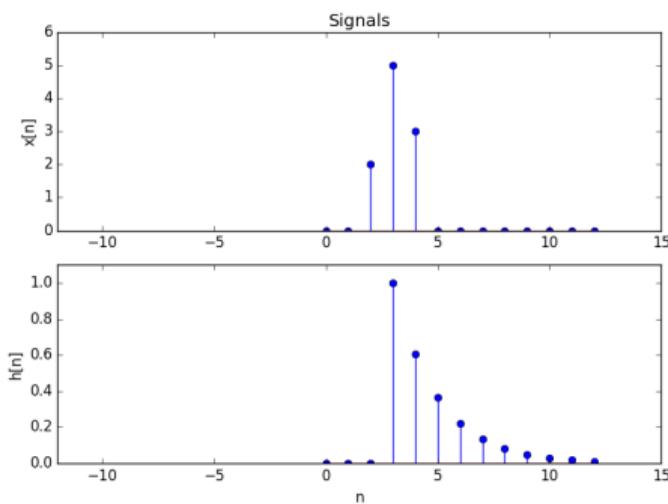
Convolution: Illustration

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



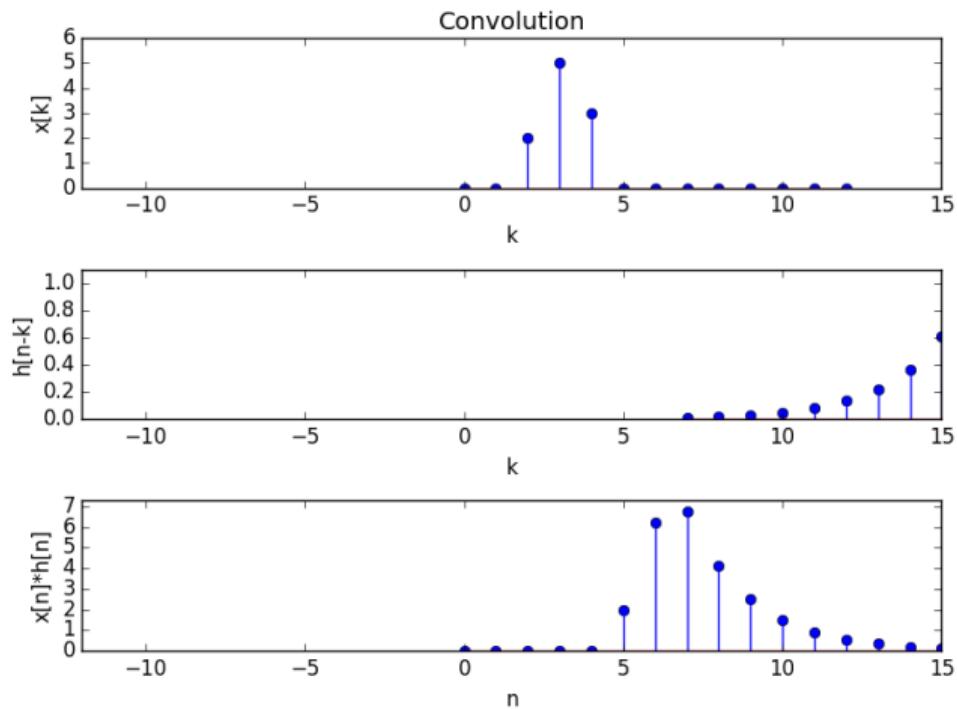
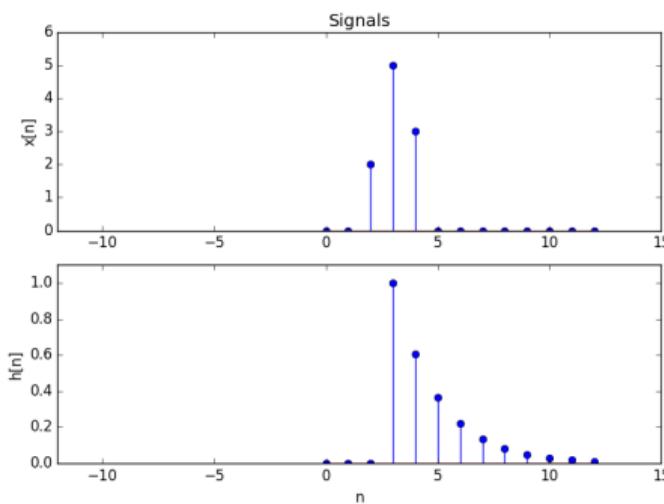
Convolution: Illustration

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



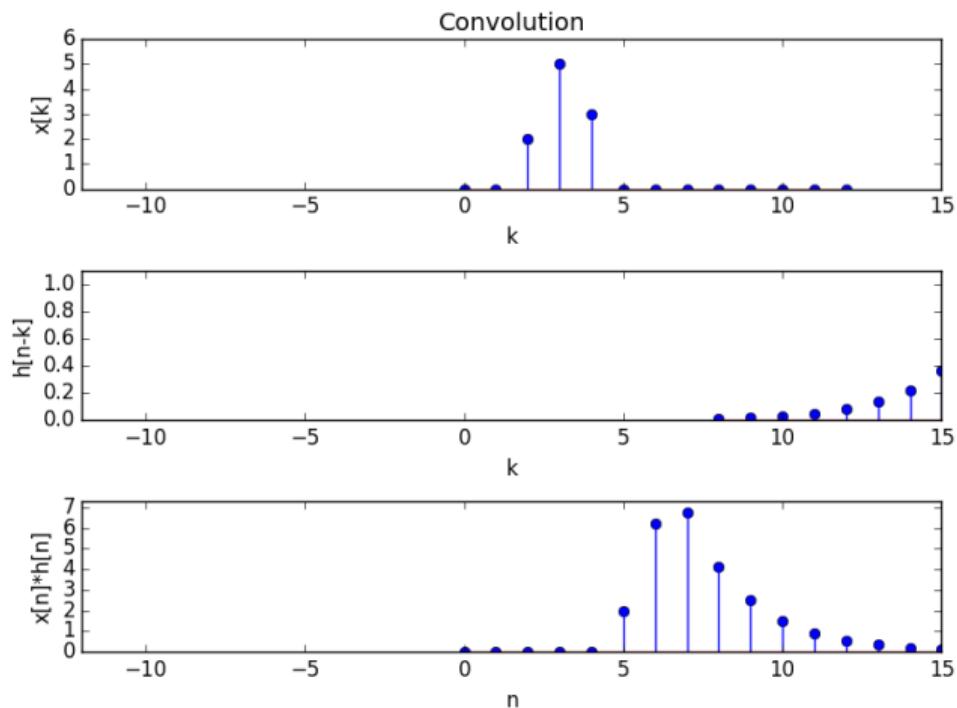
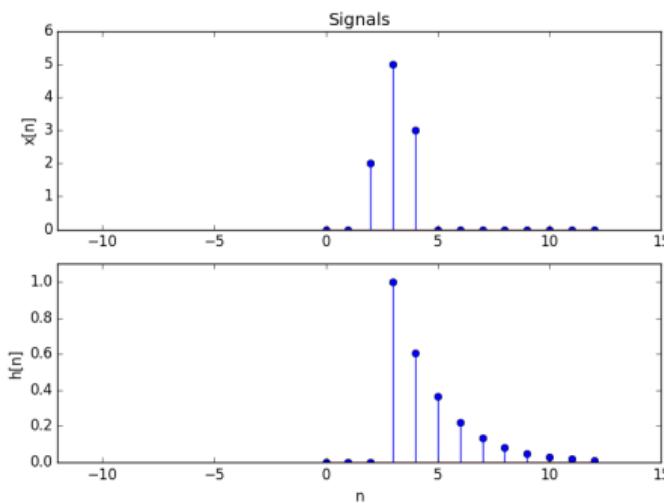
Convolution: Illustration

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



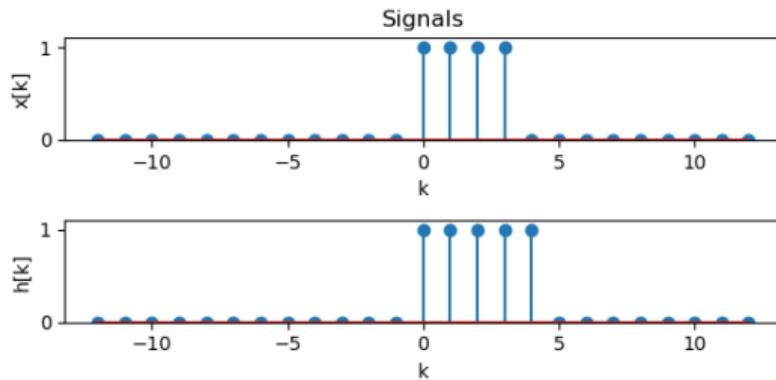
Convolution: Illustration

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

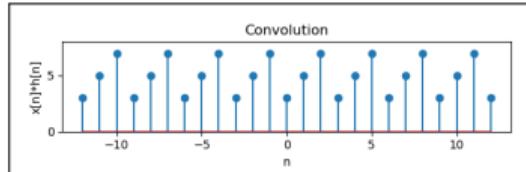


Question: Which is the convolution of the signals

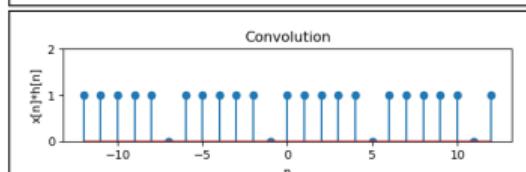
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



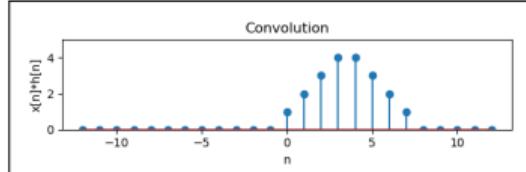
a)



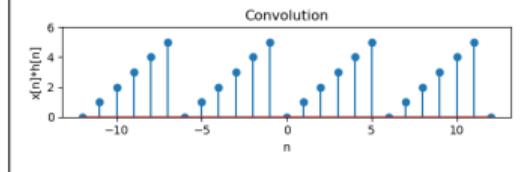
b)



c)

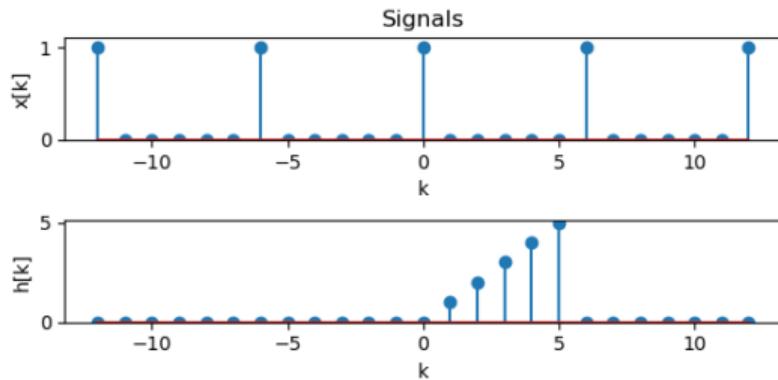


d)

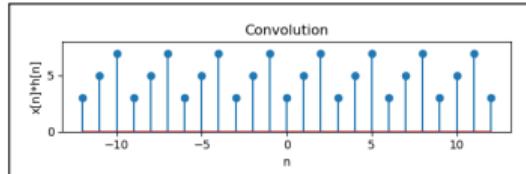


Question: Which is the convolution of the signals

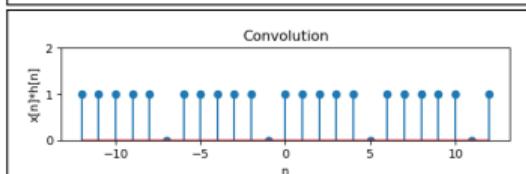
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



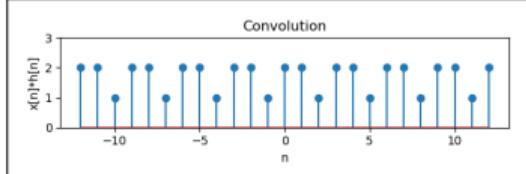
a)



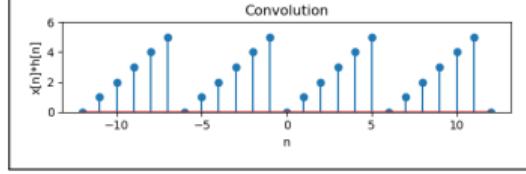
b)



c)

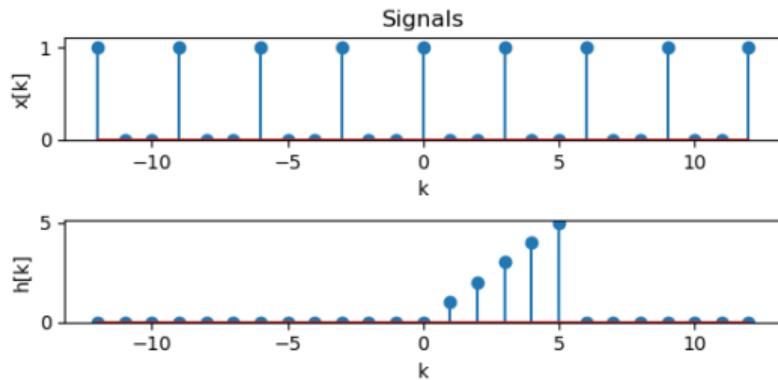


d)

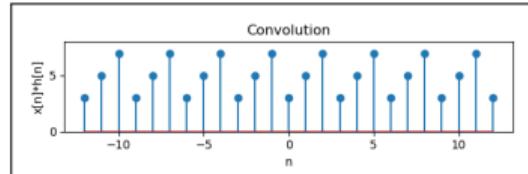


Question: Which is the convolution of the signals

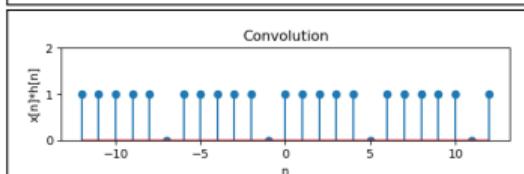
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



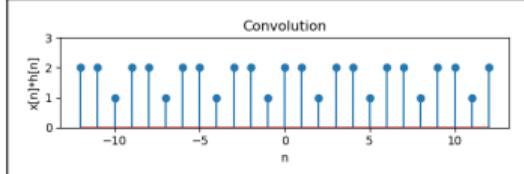
a)



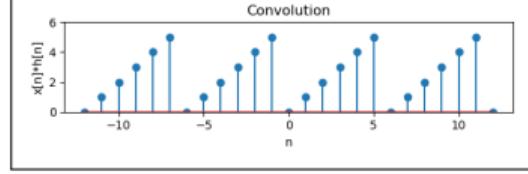
b)



c)



d)



Linear filter in general (Infinite Impulse Response)

`scipy.signal.lfilter(b, a, x, ...)`

$$\begin{aligned}y[n] &= \frac{1}{a_0} \left(b_0 x[n] + b_1 x[n-1] + \cdots + b_P x[n-P] + \right. \\&\quad \left. - a_1 y[n-1] - a_2 y[n-2] - \cdots + a_Q y[n-Q] \right) \\&= \frac{1}{a_0} \left(\sum_{i=0}^P b_i x[n-i] - \sum_{j=1}^Q a_j y[n-j] \right)\end{aligned}$$

$$a = [a_0, a_1, \dots, a_Q]$$

$$b = [b_0, b_1, \dots, b_P]$$

Fourier Transforms

Fourier transform of continuous signals

$$X(\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t} dt$$

Fourier transform of discrete signals

$$X(\omega) = \sum_{k=-\infty}^{\infty} x[k]e^{-j\omega k}$$

Discrete Fourier Transform (and Fast Fourier Transform)

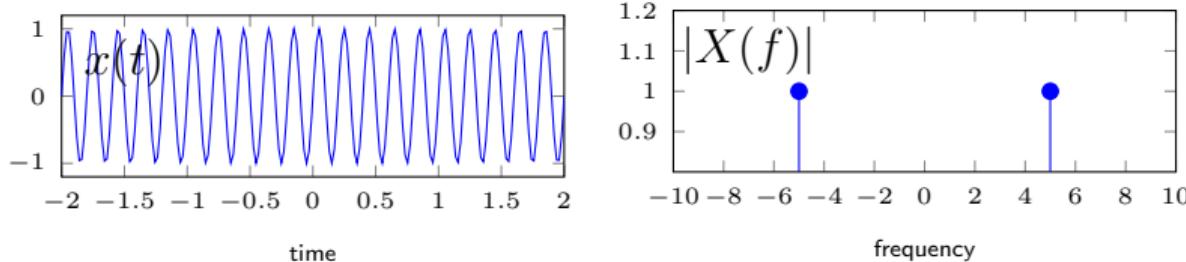
$$X[n] = \sum_{k=0}^{N-1} x[k]e^{-j2\pi \frac{n}{N} k}$$

Properties of Fourier Transform

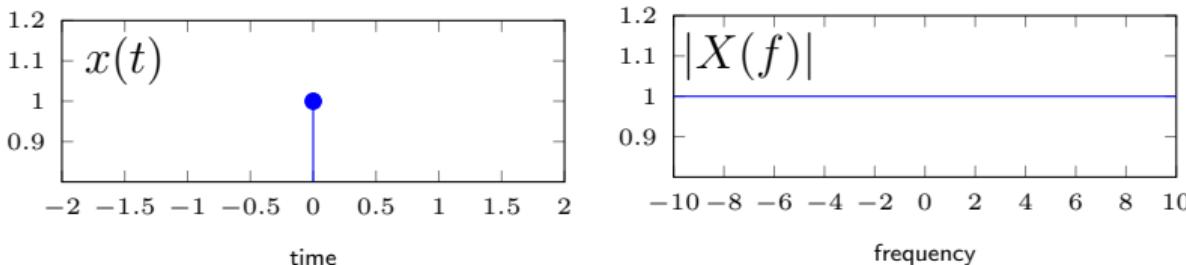
Property	Time domain	Frequency domain
Linearity	$ax[n] + by[n]$	$\iff aX(\omega) + bY(\omega)$
Time-shift	$x[n - k]$	$\iff X(\omega)e^{-j\omega nT_s}$
Convolution	$x[n] * h[n]$	$\iff X(\omega)H(\omega)$
Multiplication	$x[n]w[n]$	$\iff X(\omega) * W(\omega)$

Fourier Transform of Useful Signals

Sinusoidal at frequency $f = 5\text{Hz}$

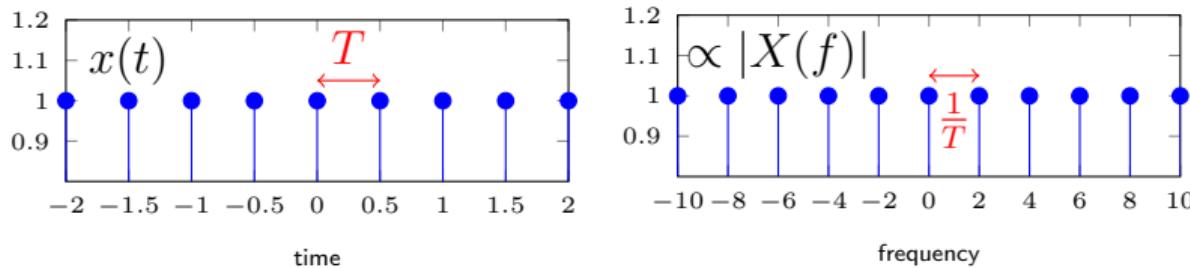


Single (Ideal) Impulse

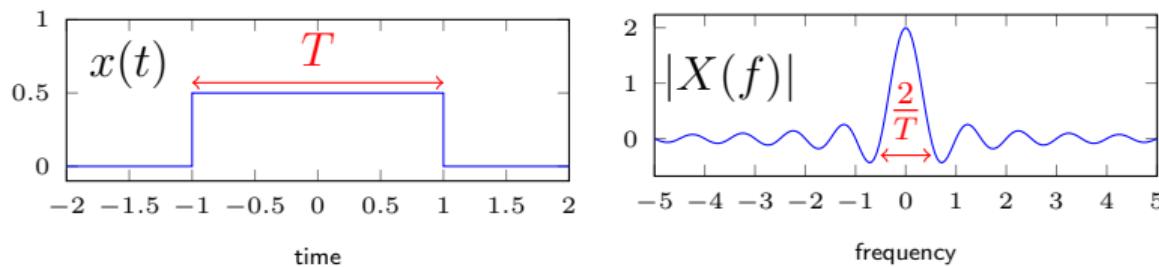


Fourier Transform of Useful Signals

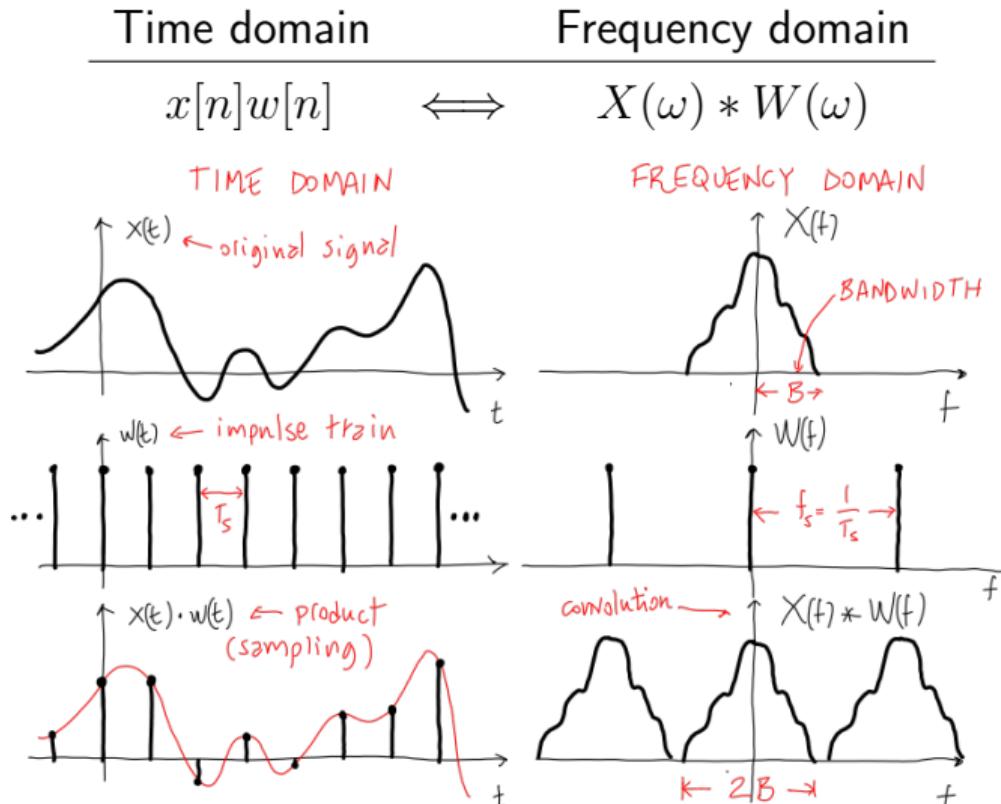
Train of (Ideal) Impulses



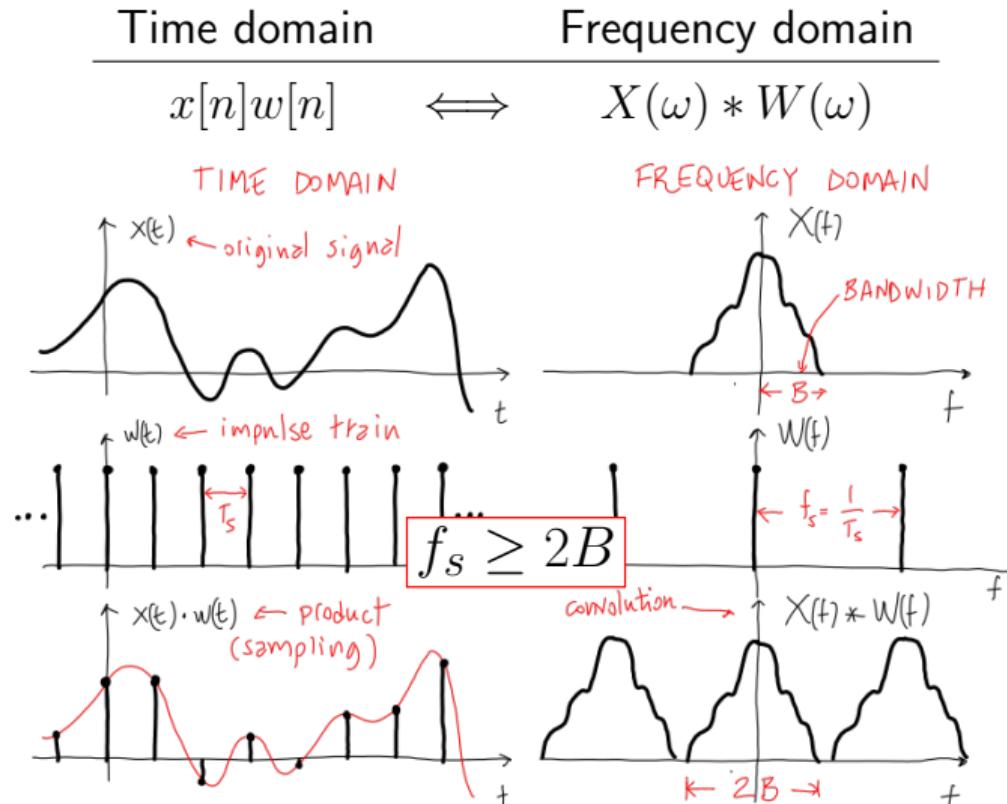
Square function



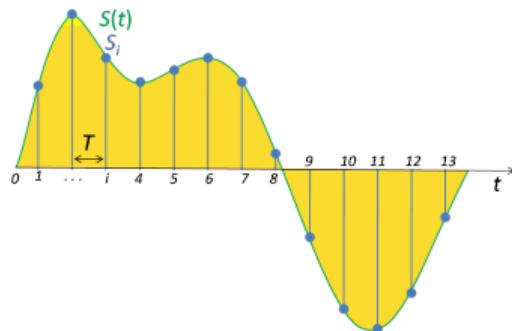
Properties of Fourier Transform — Example: sampling



Properties of Fourier Transform — Example: sampling



Sampling Theorem (Nyquist-Shannon)



If $x(t)$ contains energy up to B_x , in order to reconstruct the signal we need to sample with

$$f_s > 2B_x$$

Aliasing

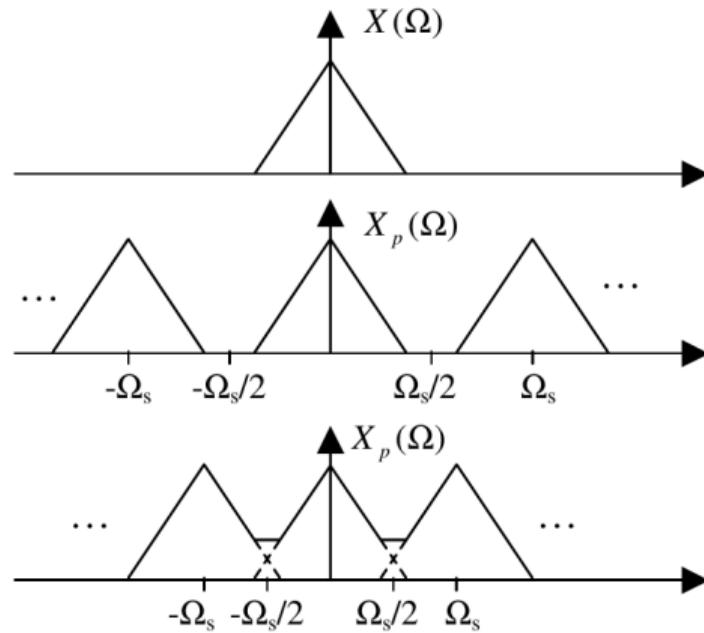


Figure from Huang, Acero and Hon (2001)

Aliasing: Illustration



Video from <https://youtu.be/usN47Jvy9PY> (unfortunately removed)

First step: represent speech signal

Sampling

- **Nyquist-Shannon Theorem:** sample at twice the band
- 8kHz (4kHz band, telephone), 16kHz (8 kHz band, high quality)
- TIDIGITS sampled at 20kHz
- TIMIT sampled at 16kHz

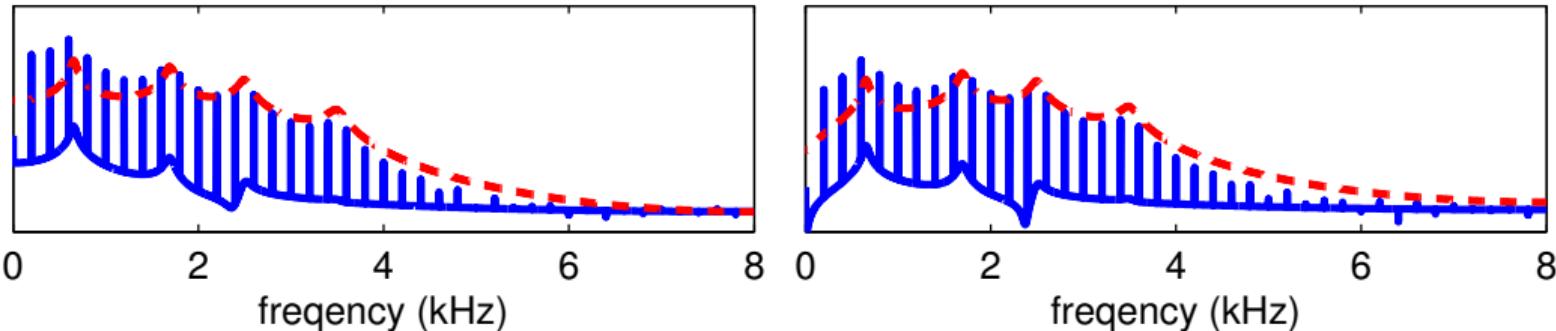
Quantisation

- Type of quantisation: linear, a-law, μ -law
- 8, 16 bits (more rare 32, floating point)
- TIDIGITS and TIMIT are quantised with 16 bits linear

Pre-emphasis

Compensate for the 6db/octave drop (glottal shape - radiation at the lips)

$$y[n] = x[n] - \alpha x[n - 1]$$

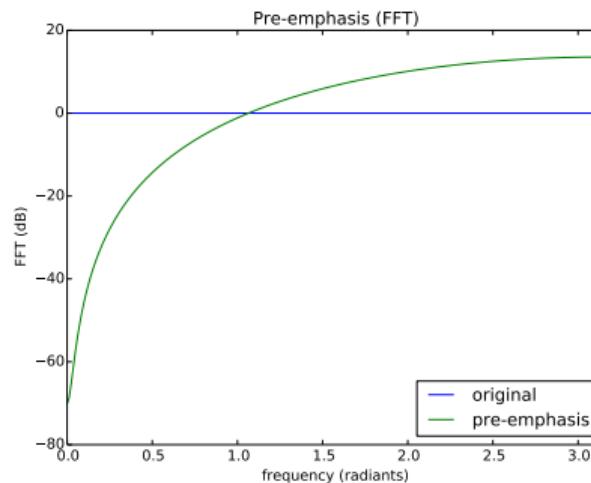
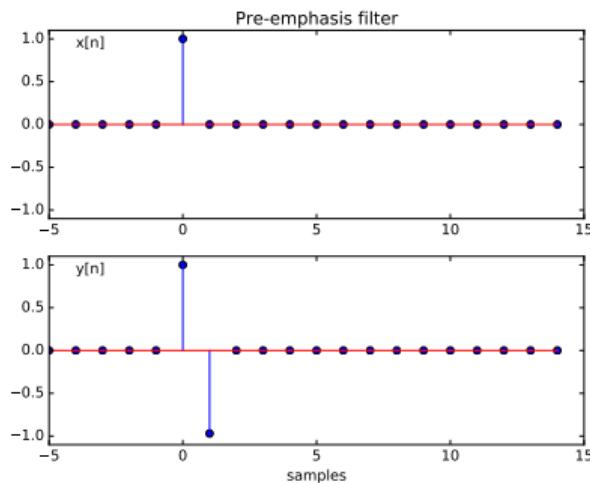


α is usually 0.95–0.97

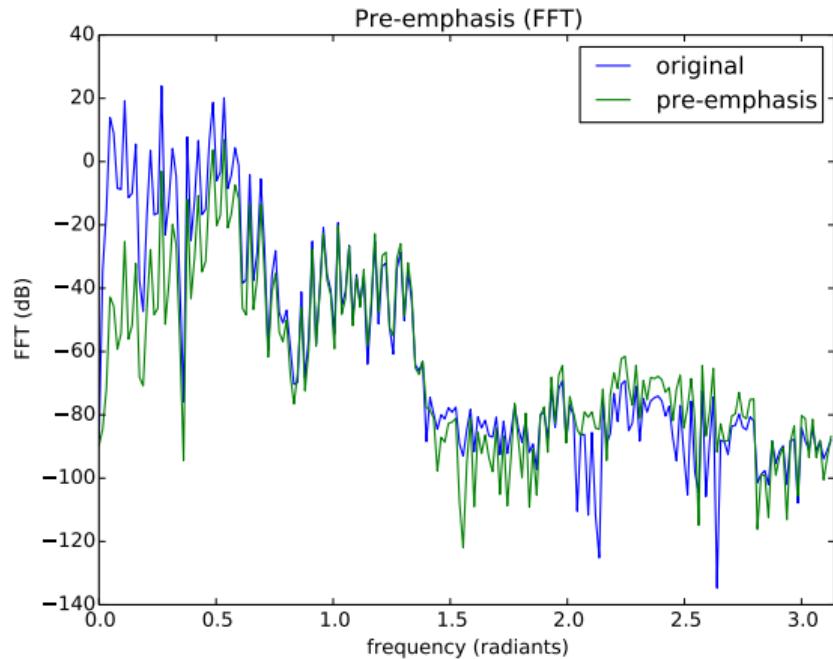
Pre-Emphasis as Linear Filter

Time domain	Frequency domain
$x[n] * h[n]$	$\iff X(\omega)H(\omega)$

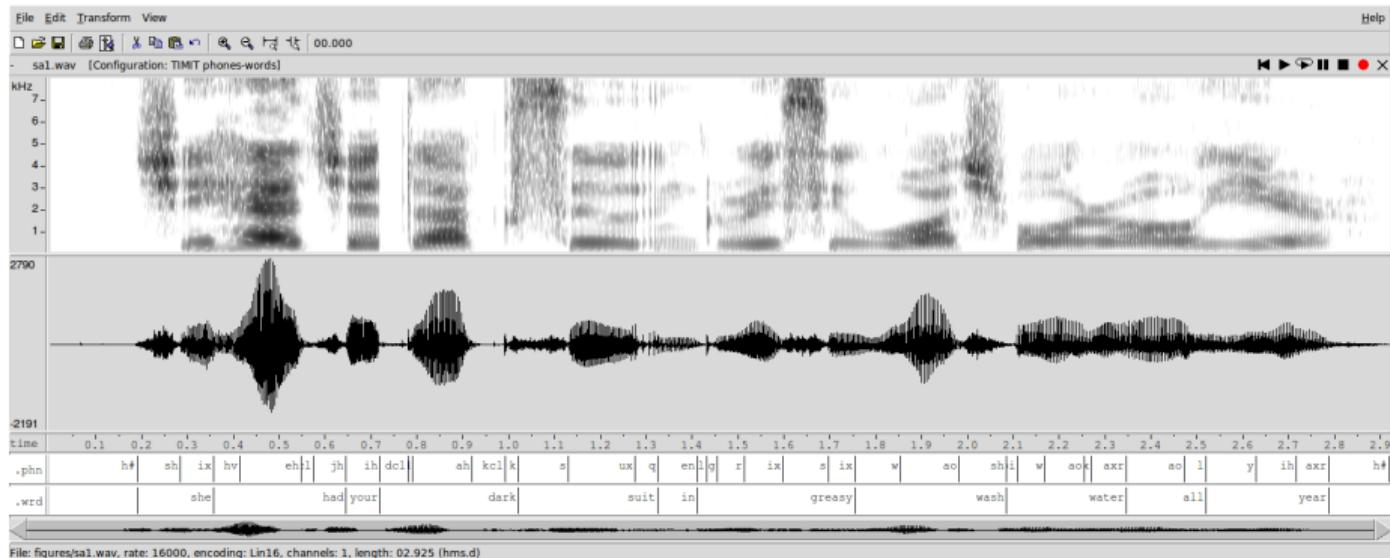
$$y[n] = x[n] - \alpha x[n-1], \quad \text{with } \alpha = 0.97$$



Pre-emphasis applied to vowel

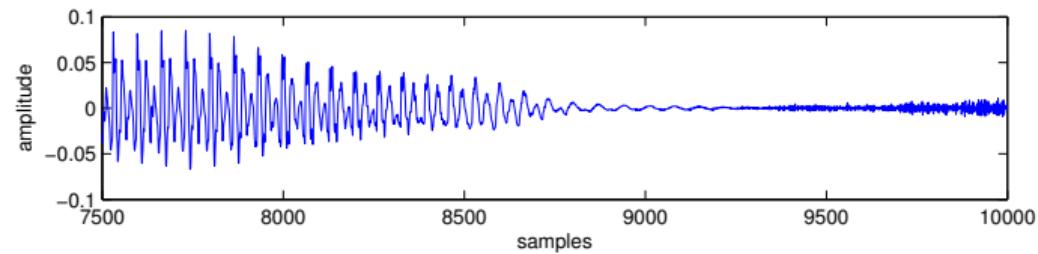


A time varying signal

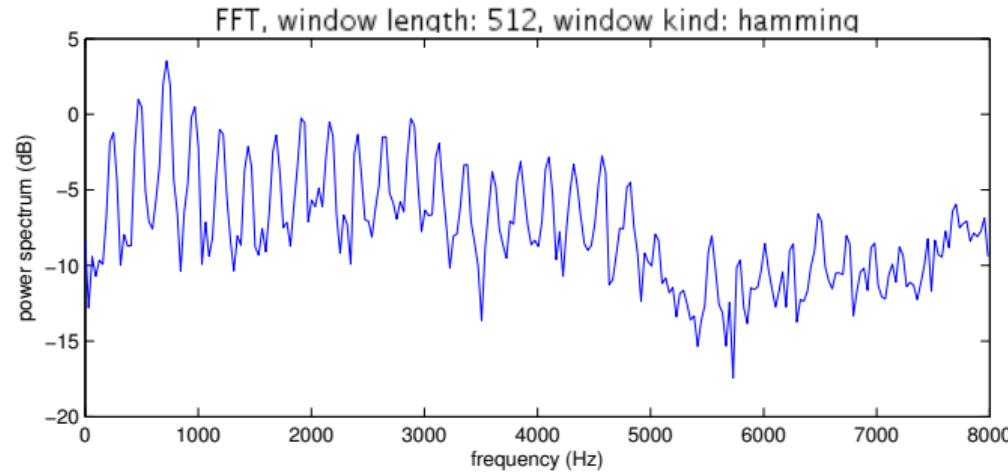
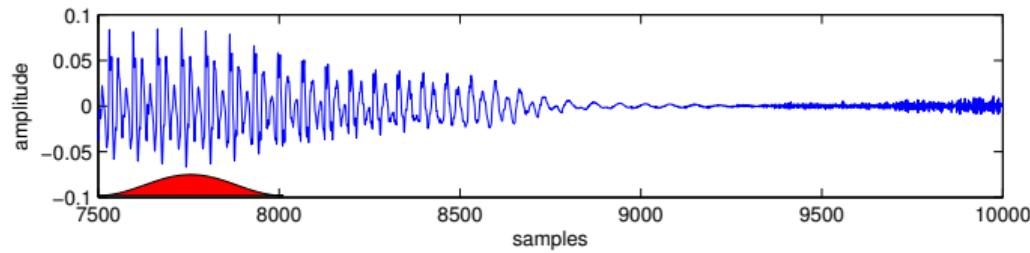


- speech is time varying
- short segment are quasi-stationary
- use short time analysis

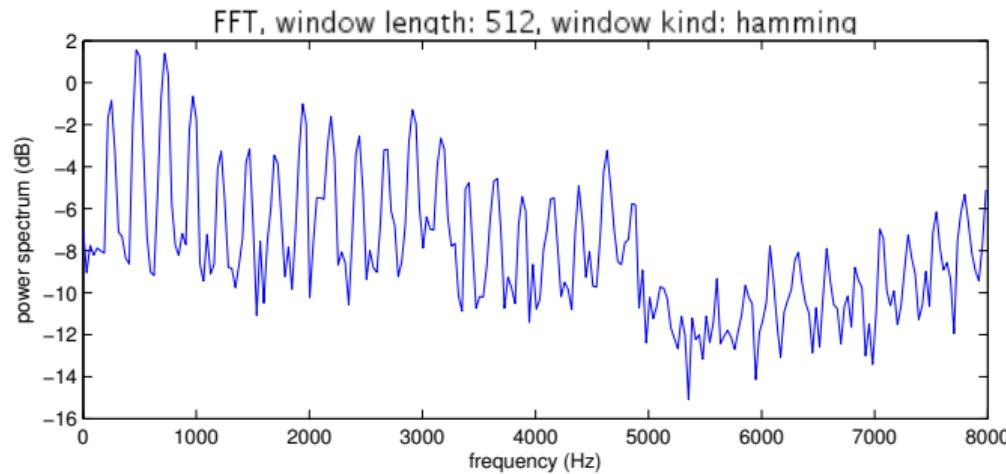
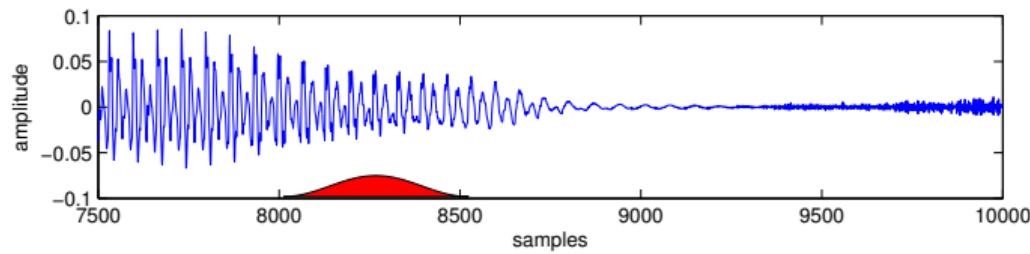
Short-Time Fourier Analysis



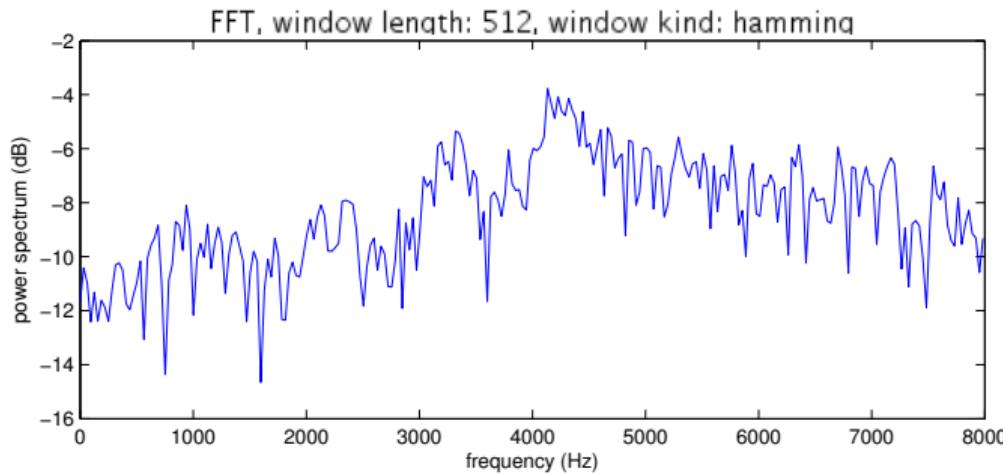
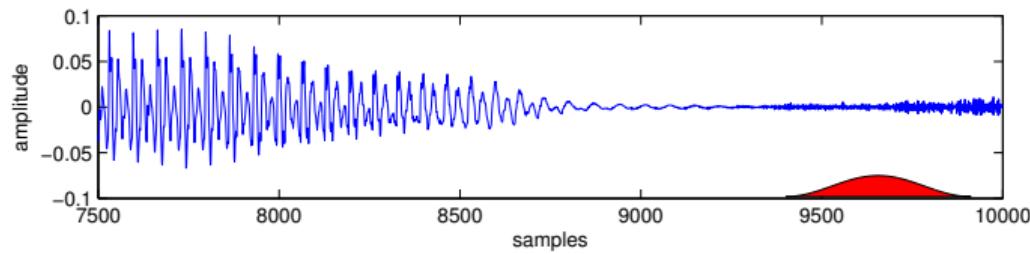
Short-Time Fourier Analysis



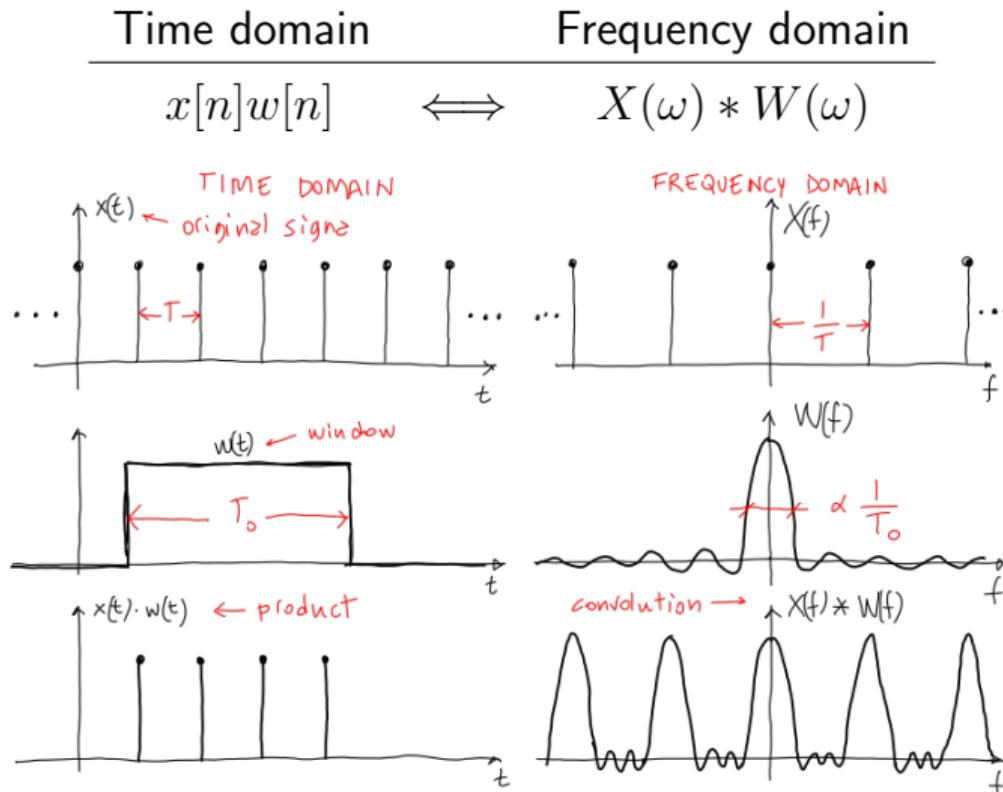
Short-Time Fourier Analysis



Short-Time Fourier Analysis

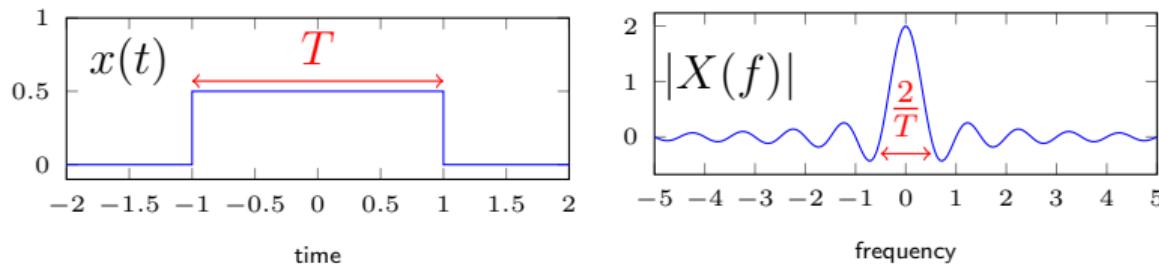


Properties of Fourier Transform — Example: Windowing

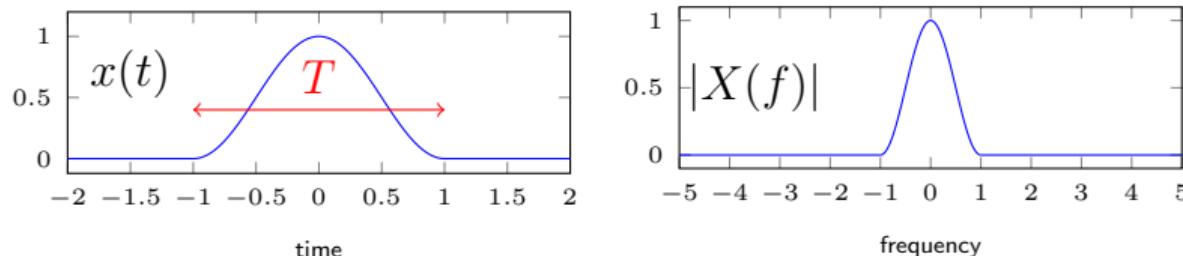


Effect of Windowing

Square window

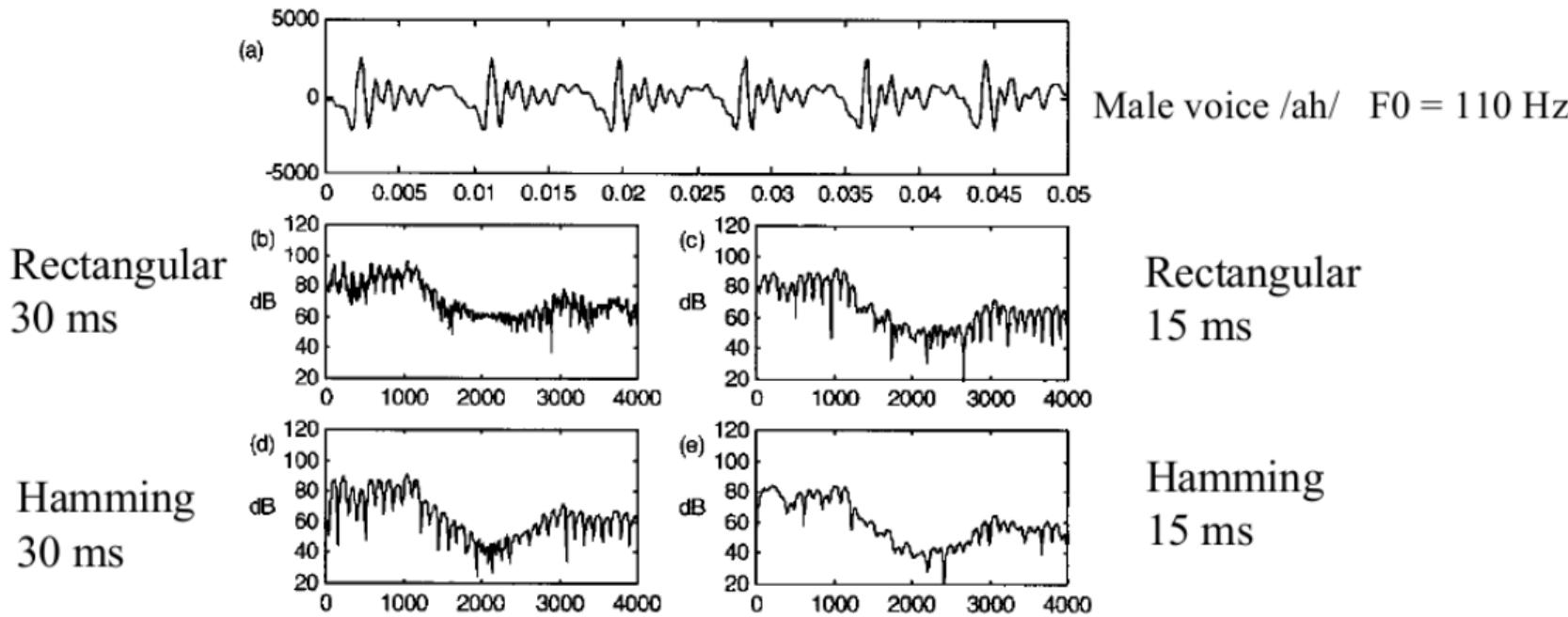


Hamming window



Effect of Windowing on Speech

Effect of different window functions



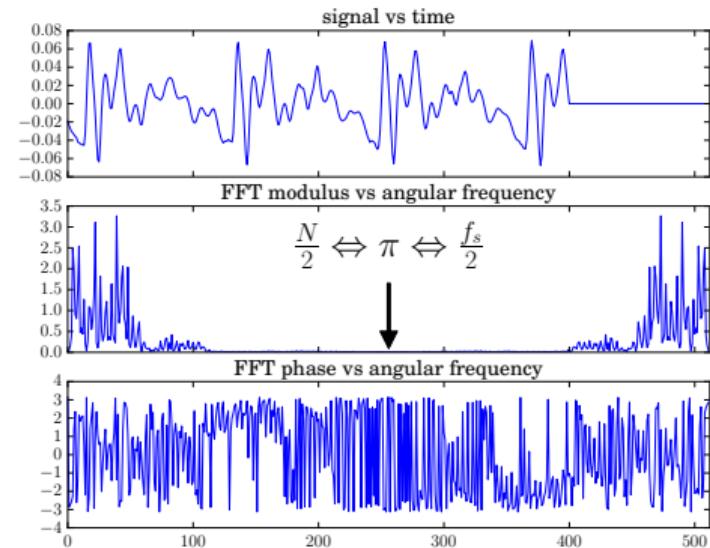
Windowing, typical values

- signal sampling frequency: 8–20kHz
- analysis window: 10–50ms
- frame step: 10–25ms (100–40Hz)

Fast Fourier Transform (FFT) ($N = 512$)

```
scipy.fftpack.fft(x, n=512, ...)
```

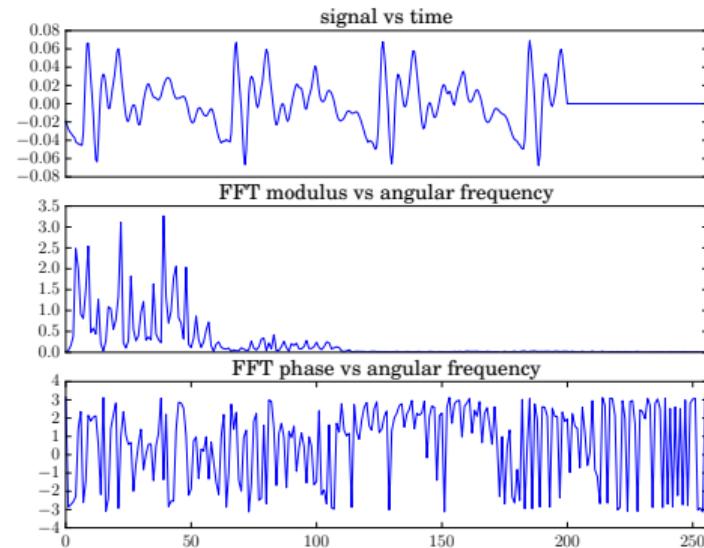
$$X[n] = \sum_{k=0}^{N-1} x[k] e^{-j2\pi \frac{n}{N} k}$$



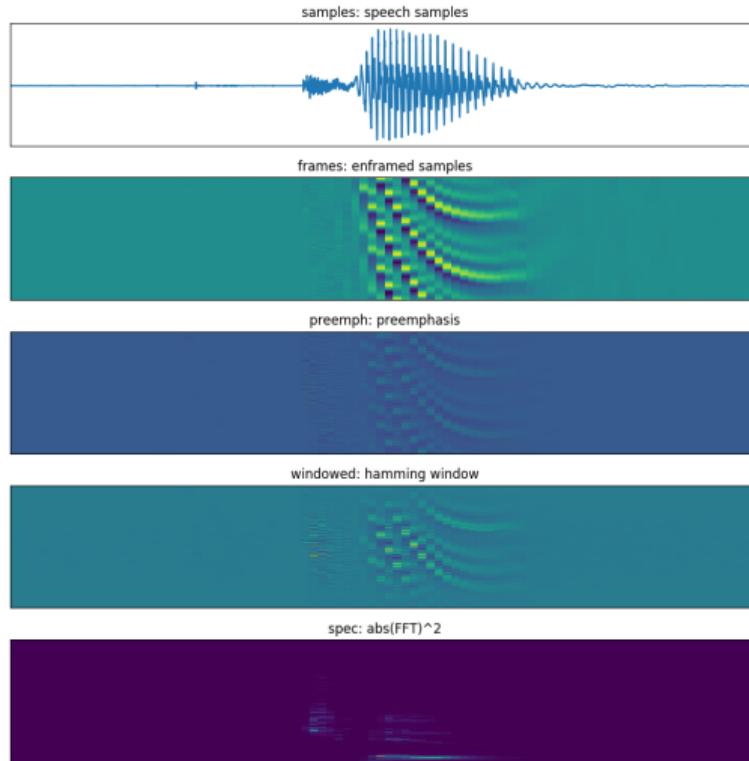
Fast Fourier Transform (FFT) ($N = 512$)

```
scipy.fftpack.fft(x, n=512, ...)
```

$$X[n] = \sum_{k=0}^{N-1} x[k]e^{-j2\pi \frac{n}{N} k}$$



Speech signal processing in practice



Outline

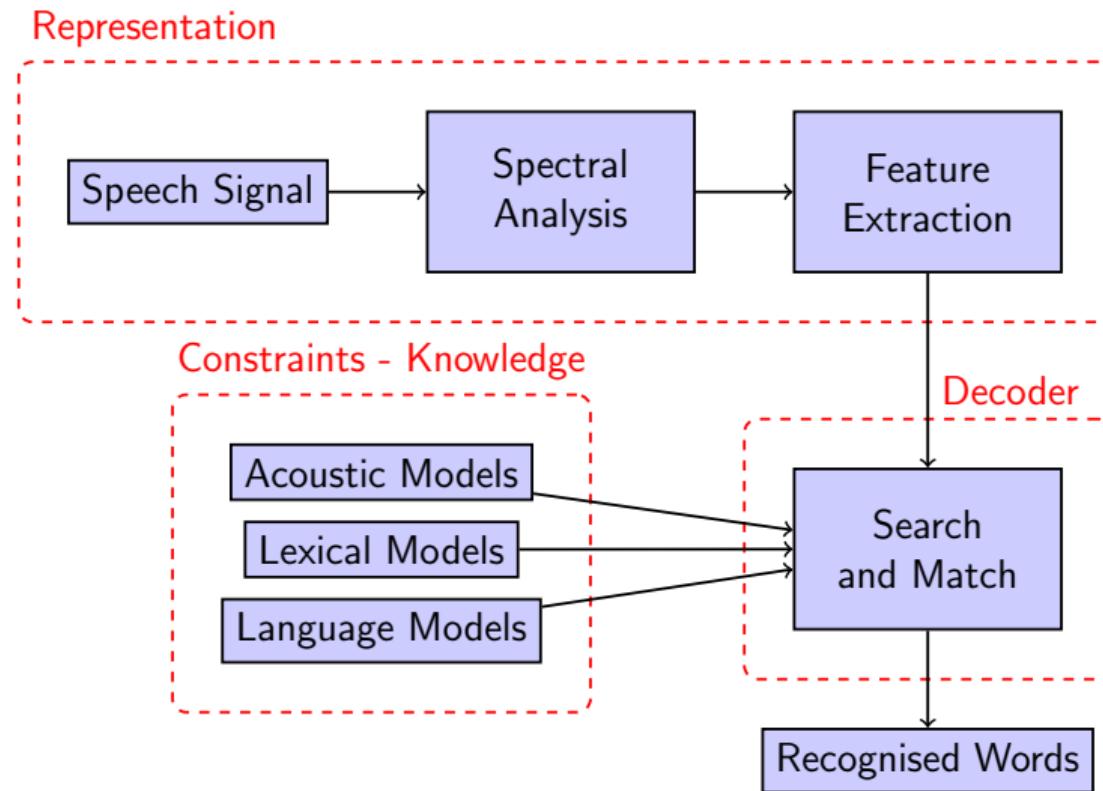
1 Speech Signal Representations

- Signal Processing Reminder
- Sampling and Quantization
- Pre-Emphasis
- Windowing
- Discrete Fourier Transform

2 Feature Extraction

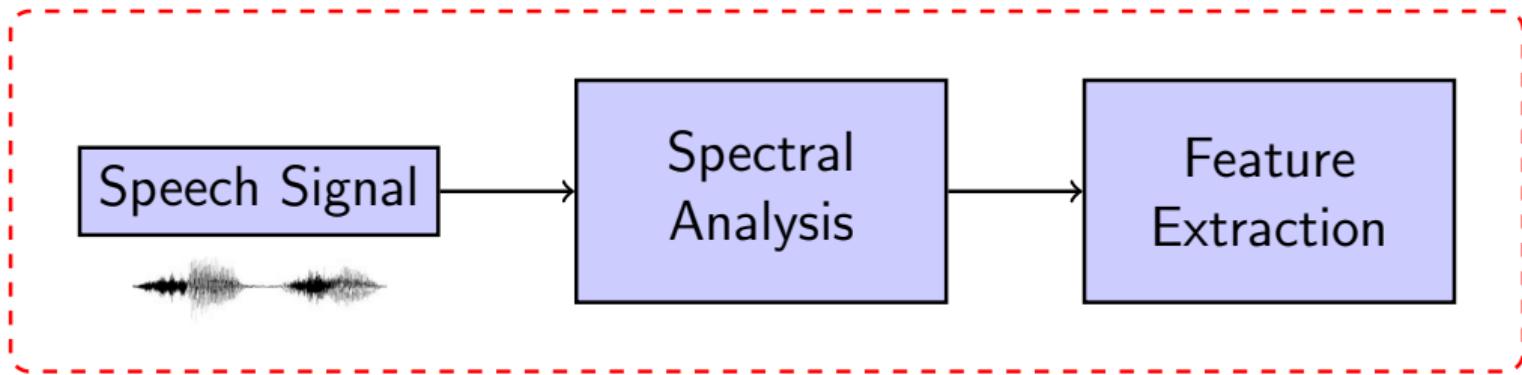
- Linear Prediction Analysis (LPA)
- Cepstrum
- Perceptually Motivated Features

Components of ASR System



Speech Signal Representations

Representation

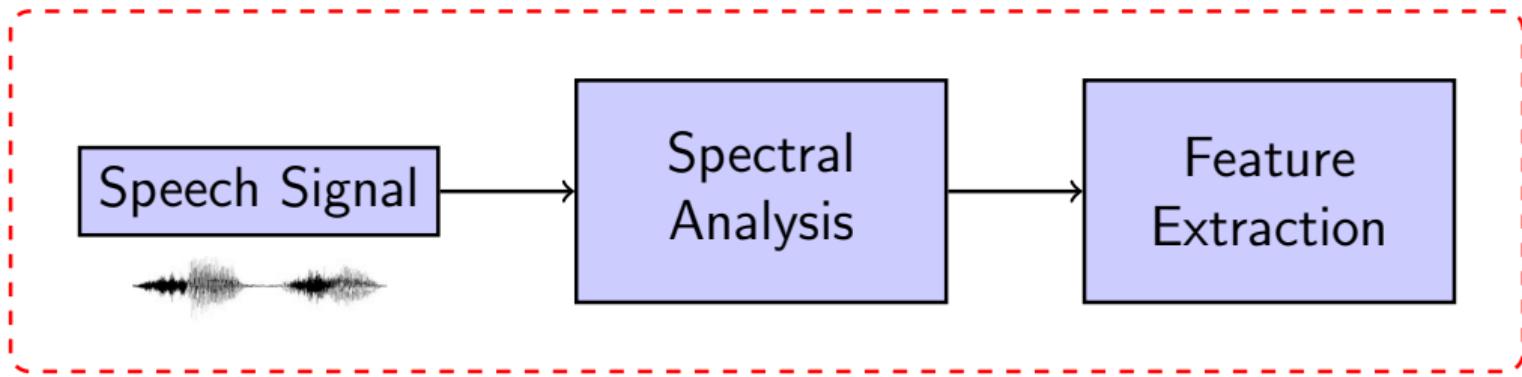


Goals:

- disregard irrelevant information
- optimise relevant information for modelling

Speech Signal Representations

Representation



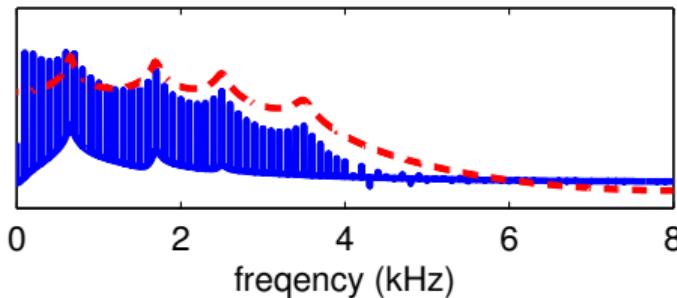
Means:

- try to model essential aspects of speech production
- imitate auditory processes
- consider properties of statistical modelling

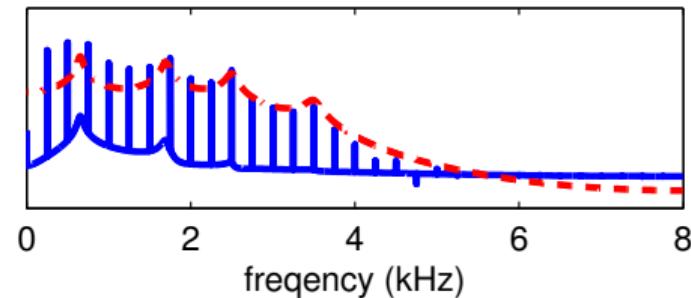
F_0 and Formants

- Varying F_0 (vocal fold oscillation rate)

spectrum (log) $f_0 = 100\text{Hz}$

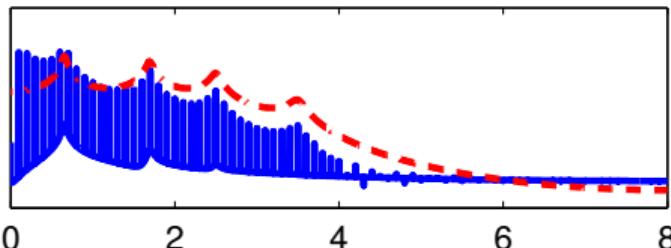


spectrum (log) $f_0 = 250\text{Hz}$

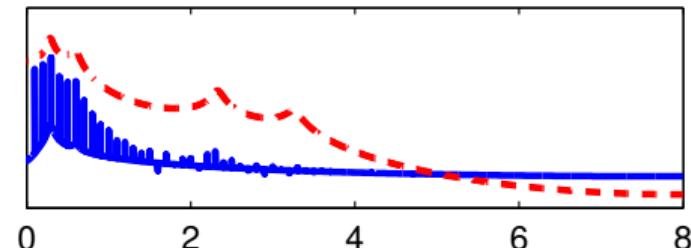


- Varying Formants (vocal tract shape)

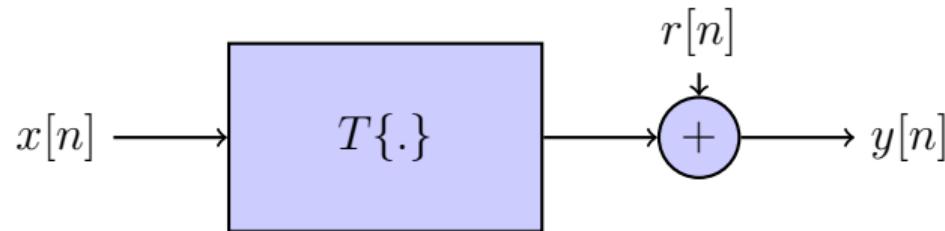
spectrum (log) vowel [ɛ]



spectrum (log) vowel [u]



Linear Prediction Coefficients (LPC)



approximate $y[n]$ as a linear combination of p previous samples:

$$\hat{y}[n] = \sum_{k=1}^p a_k y[n - k]$$

The error is called **residual**: $r[n] = \hat{y}[n] - y[n]$

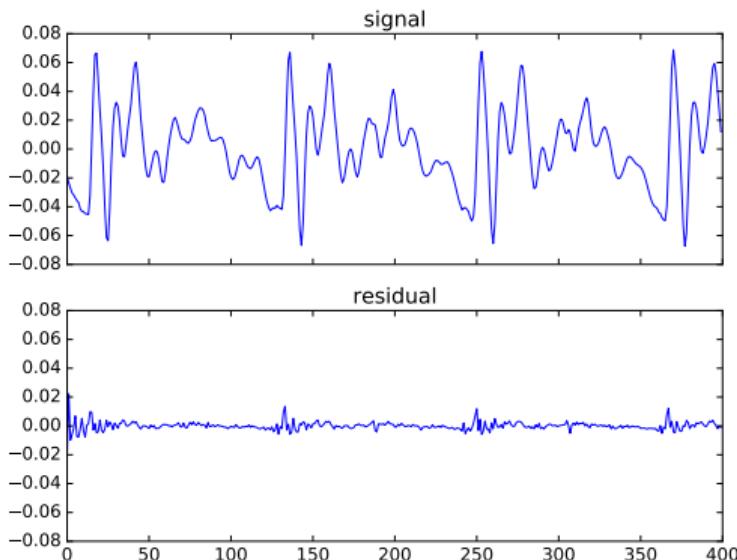
The output of the signal is:

$$y[n] = \sum_{k=1}^p a_k y[n - k] + r[n]$$

LPC and Speech coding

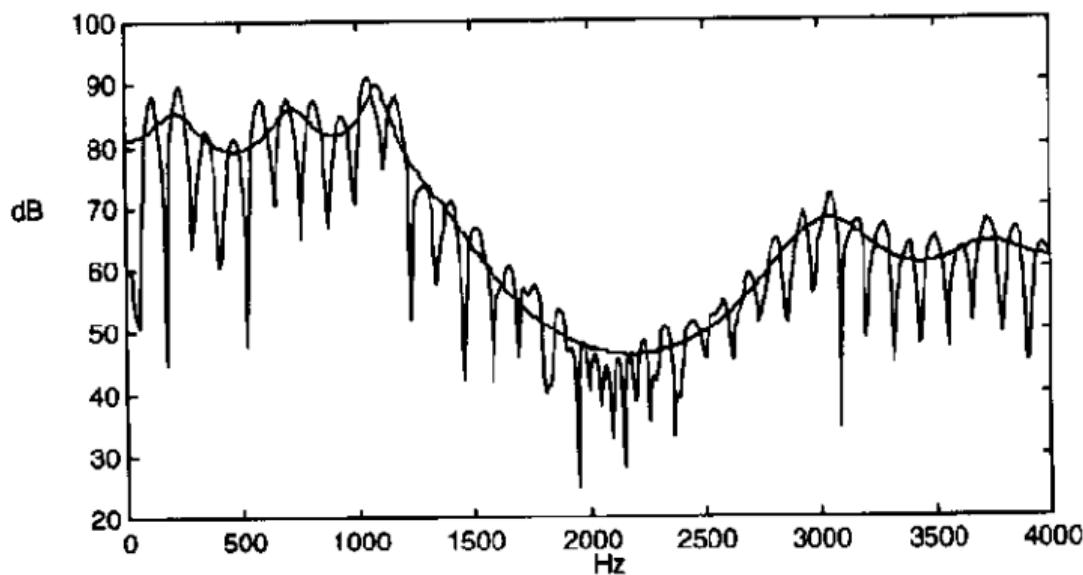
$$y[n] = \sum_{k=1}^p a_k y[n - k] + r[n]$$

- We only need to send a_1, \dots, a_p and $r[n]$.
- $r[n]$ can be coded with fewer bits



LPC Example

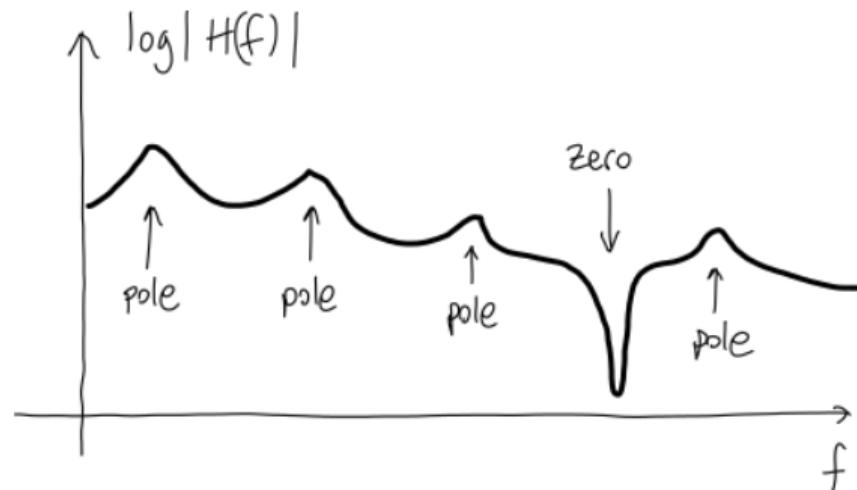
$$y[n] = \sum_{k=1}^p a_k y[n - k] + r[n]$$



Infinite Impulse Response (IIR) Systems

In general y depends on (delayed) samples of the input, as well as the output at previous times (feedback)

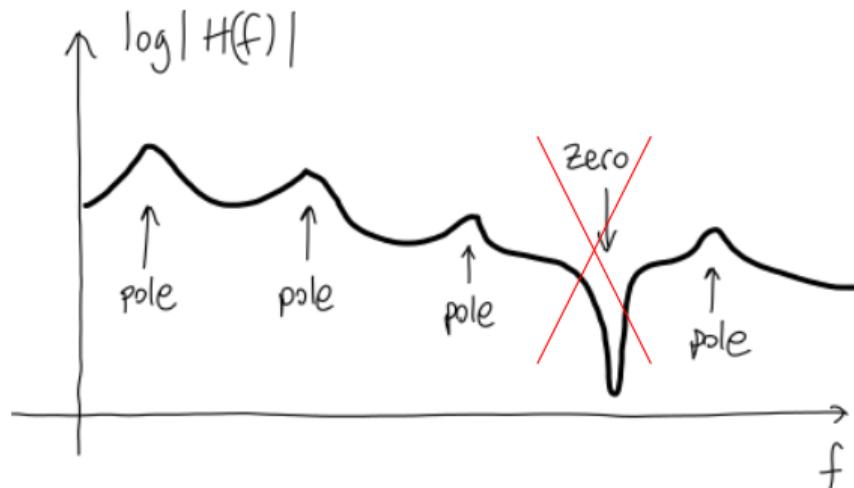
$$\begin{aligned}y[n] &= \frac{1}{a_0} \sum_{h=0}^P b_h x[n-h] && \leftarrow \text{zeros} \\&\quad - \frac{1}{a_0} \sum_{k=1}^Q a_k y[n-k] && \leftarrow \text{poles}\end{aligned}$$



Linear Prediction

Auto regressive (AR): only depends on current input and the output at previous times (feedback)

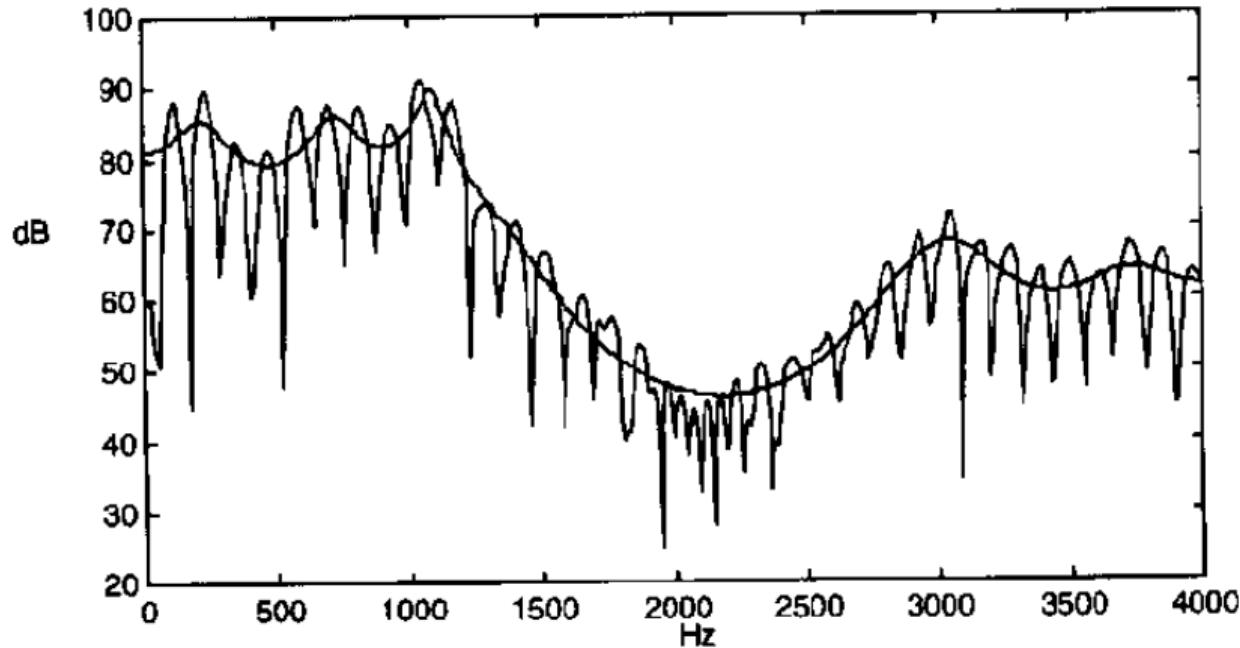
$$y[n] = \frac{b_0}{a_0} x[n] \quad \leftarrow \text{no zeros}$$
$$- \frac{1}{a_0} \sum_{k=1}^Q a_k y[n-k] \quad \leftarrow \text{poles}$$



See also [poles_and_zeros.pdf](#) in Blackboard

LPC Limitations

- better match at spectral peaks than at valleys (all-pole model)
- not accurate if transfer function contains zeros (nasals, fricatives...)



Cepstrum Rationale (Homomorphic Transformation)

- signals combined in a convolutive way: $a[n] * b[n] * c[n]$
- in the spectral domain: $A(z)B(z)C(z)$
- taking the log: $\log(A(z)) + \log(B(z)) + \log(C(z))$
- to analyse the different contribution perform Fourier transform (DCT if not interested in phase information).

Cepstrum Rationale (Homomorphic Transformation)

- signals combined in a convolutive way: $a[n] * b[n] * c[n]$
- in the spectral domain: $A(z)B(z)C(z)$
- taking the log: $\log(A(z)) + \log(B(z)) + \log(C(z))$
- to analyse the different contribution perform Fourier transform (DCT if not interested in phase information).
- Terminology:
 - frequency vs quefrency
 - spectrum vs cepstrum
 - filter vs lifter
 - ...

Cepstrum Definition

Complex Cepstrum

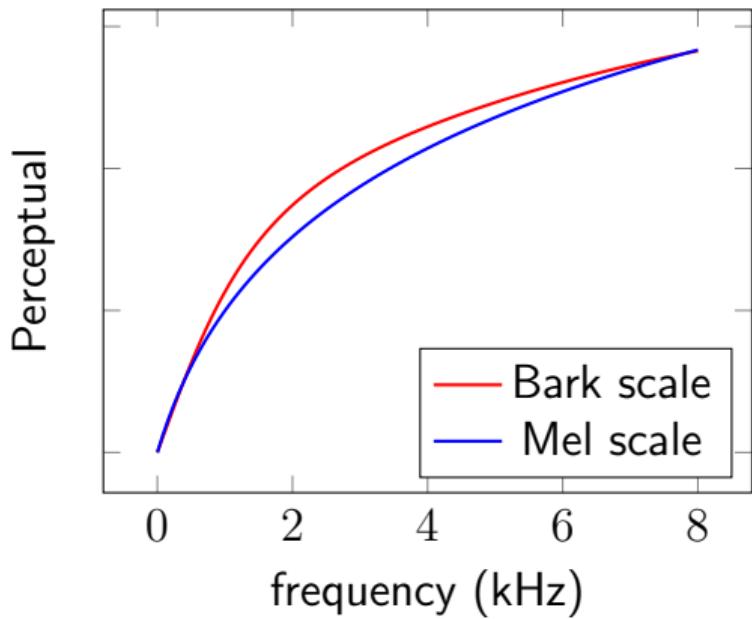
$$\hat{x}[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \ln X(\omega) e^{j\omega n} d\omega$$

Real Cepstrum

$$c[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \ln |X(\omega)| e^{j\omega n} d\omega$$

Mel Filterbank: Motivation

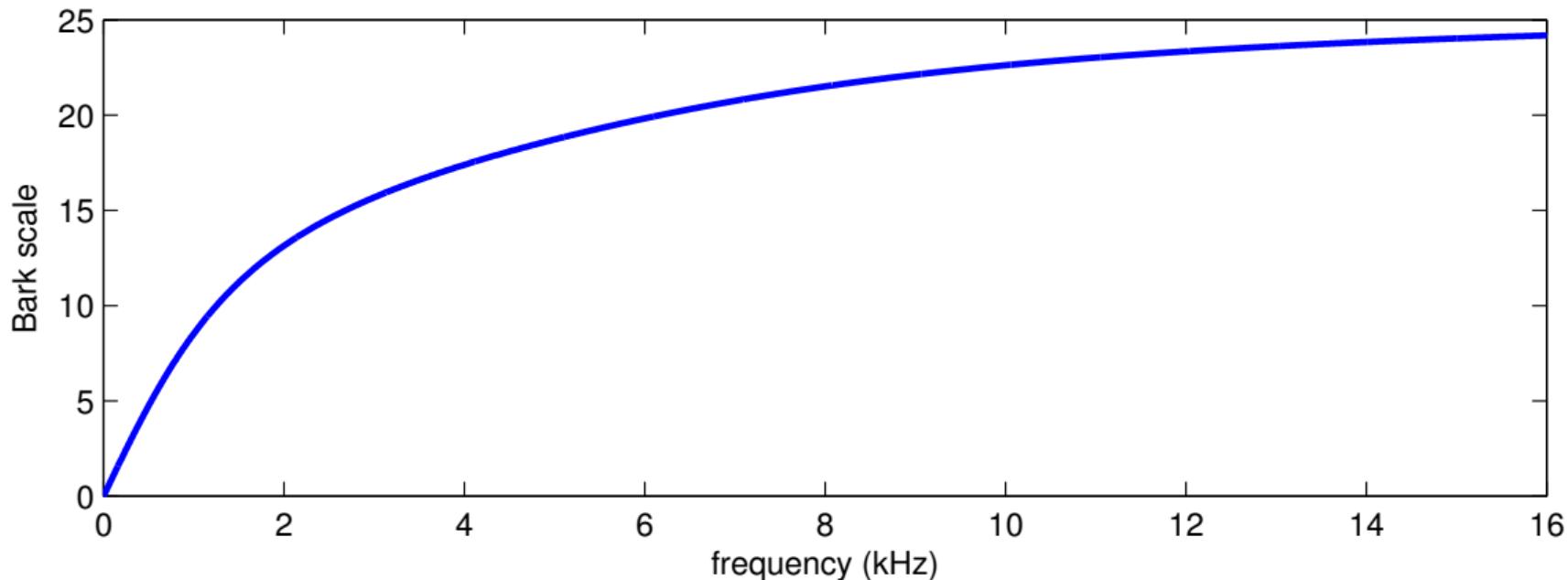
- Perception of frequencies is logarithmic: Mel and Bark scales
- Perception of amplitude (or energy, or loudness) is logarithmic



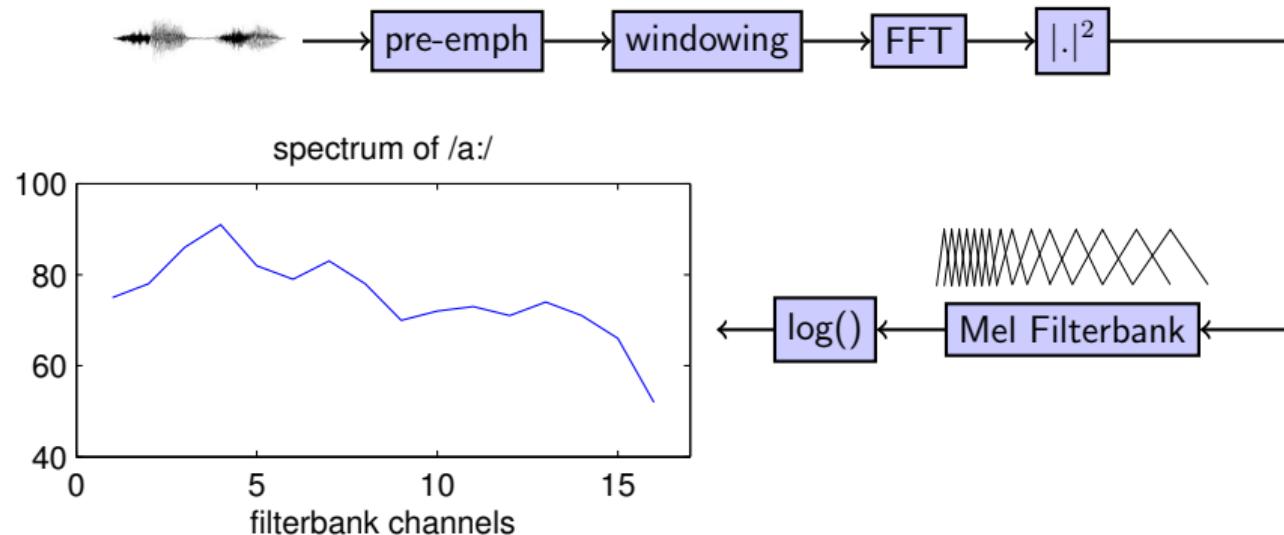
$$\begin{aligned}\text{bark}(f) &= 13 \arctan(0.00076 * f) + \\ &+ 3.5 \arctan \left[\left(\frac{f}{7500} \right)^2 \right] \\ \text{mel}(f) &= 1125 \ln \left(1 + \frac{f}{700} \right)\end{aligned}$$

Perceptual Linear Prediction

- Transform to the Bark frequency scale before computing the LPC coefficients
- Cubic root of energy instead of logarithm



Mel Filterbank: Calculation



Mel Filterbank: Pros and Cons

Cons:

- coefficients are correlated (inefficient use of information)
- difficult to model statistically (e.g. multivariate Gaussian distribution)

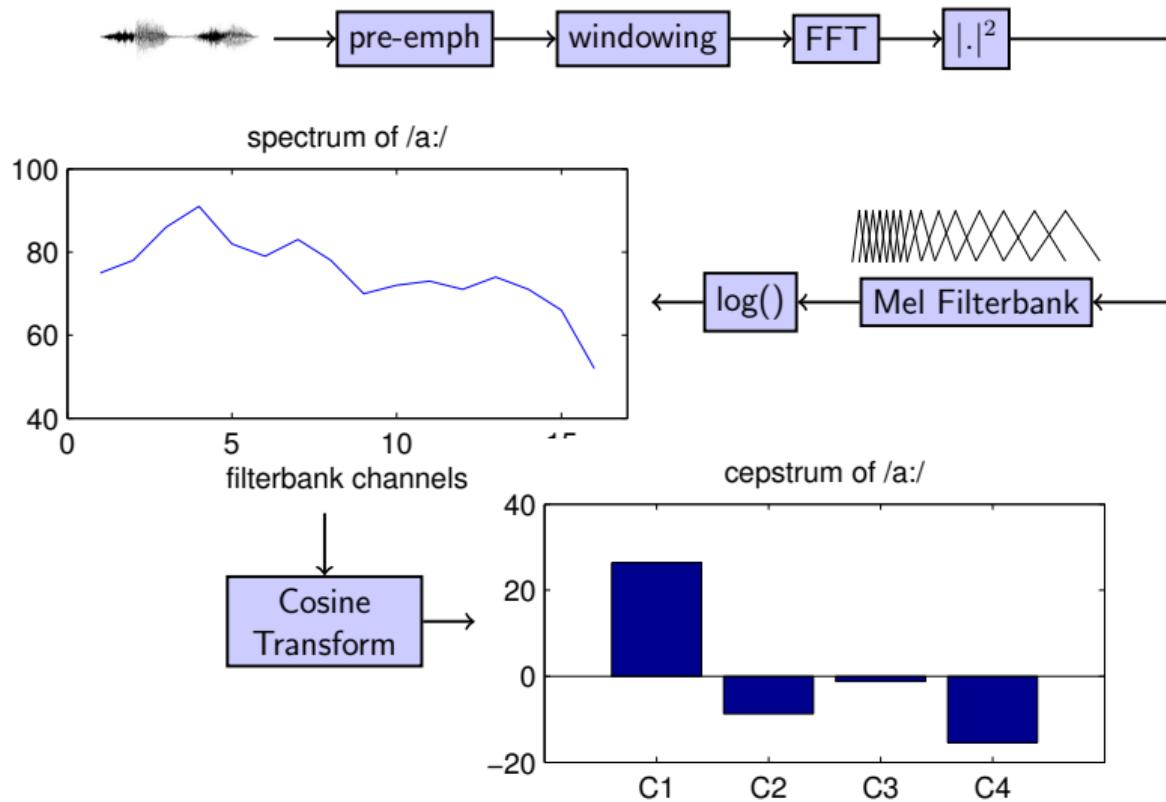
Pros:

- easy to interpret (smooth spectrum)
- works well with neural networks (no problem with correlations)

Mel Frequency Cepstrum Coefficients

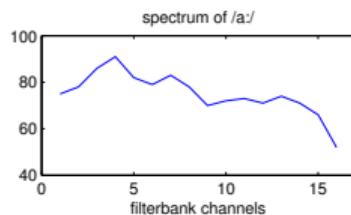
- *de facto* standard in ASR
- imitate aspects of auditory processing
- does not assume all-pole model of the spectrum
- uncorrelated: easier to model statistically

MFCCs Calculation

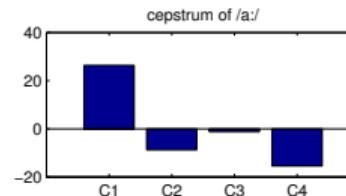
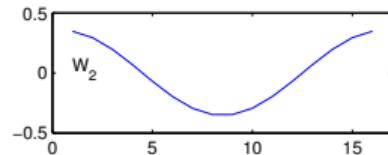
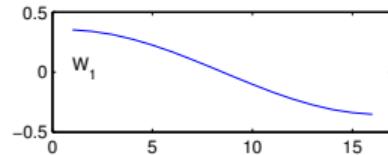
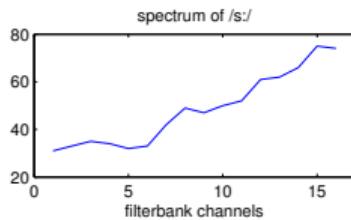


MFCC: Cosine Transform (scipy.fftpack.realtransforms.dct)

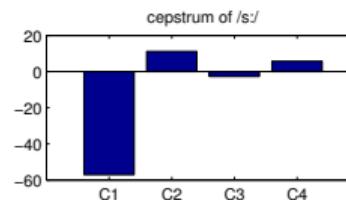
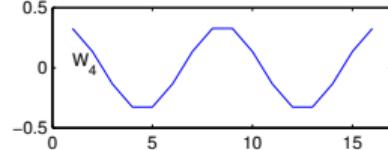
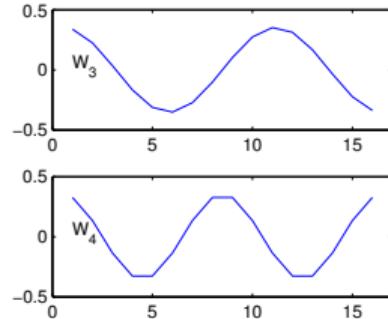
$$C_j = \sqrt{\frac{2}{N}} \sum_{i=1}^N A_i \cos\left(\frac{j\pi(i-0.5)}{N}\right)$$



A_i



C_j



MFCC Advantages

- fairly uncorrelated coefficients (simpler statistical models)
- high phonetic discrimination (empirically shown)
- do not assume all-pole model
- low number of coeff. enough to capture coarse structure of spectrum
- Cepstral Mean Subtraction corresponds to channel removal

Dynamic Features

Concatenate static MFCCs (or LPCs) to Δ and $\Delta\Delta$ vectors.

Δ_n computed as weighted sum of $d_k(n)$

$$\Delta_n = \frac{\sum_{k=1}^K w_k d_k(n)}{\sum_{k=1}^K w_k}$$

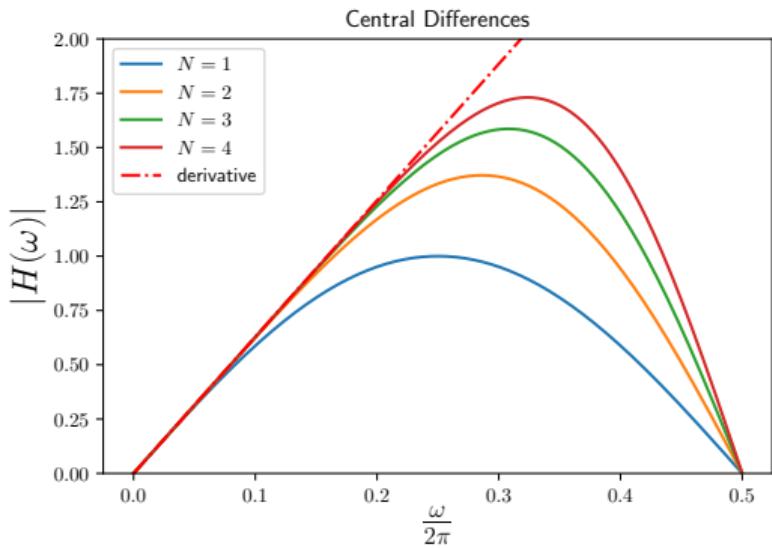
$d_k(n)$: finite differences centered around n with interval $2k$:

$$d_k(n) = \frac{c_{n+k} - c_{n-k}}{2k}$$
$$w_k = 2k^2$$

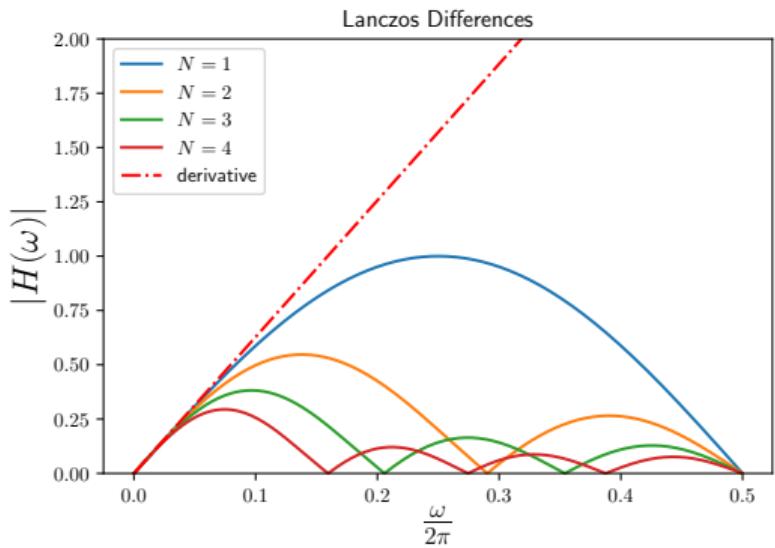
Similarly for $\Delta\Delta_n$

Dynamic Features: Motivation

Central Differences



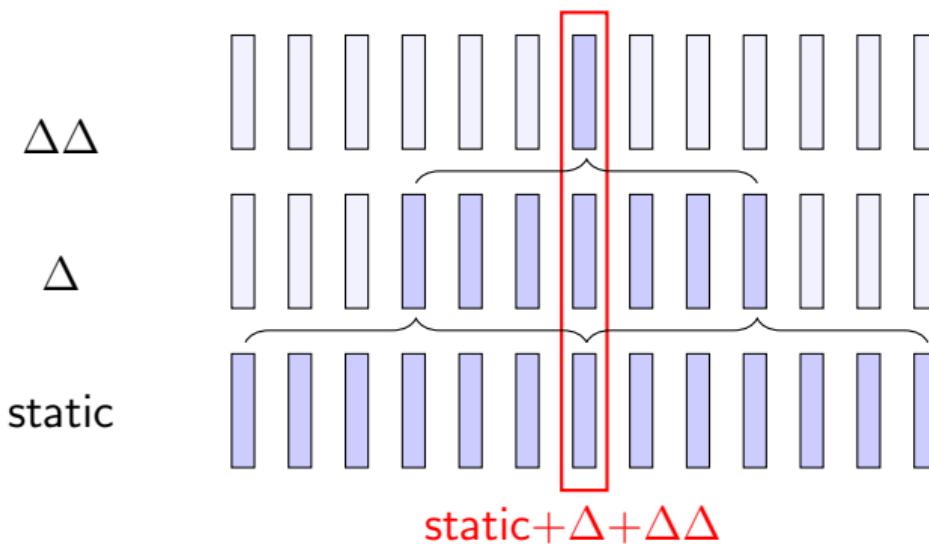
Lanczos Differences



Polynomial fit with or without error

Dynamic Features: Common values

- Usually k goes from 1 to 3
- to compute static+ Δ + $\Delta\Delta$ we need 13 consecutive static vectors (around 130 msec).



MFCCs: typical values

- 12 Coefficients C1–C12
- Energy (could be C0)
- Delta coefficients (derivatives in time)
- Delta-delta (second order derivatives)
- total: 39 coefficients per frame (analysis window)

Introduction to Machine Learning

TTT4185 Machine Learning for Signal Processing

Giampiero Salvi

Department of Electronic Systems
NTNU

HT2021

Examples of applications

Google self driving



IBM congestion fees



autonomous ships



Voice assistants



DeepMind AlphaGo



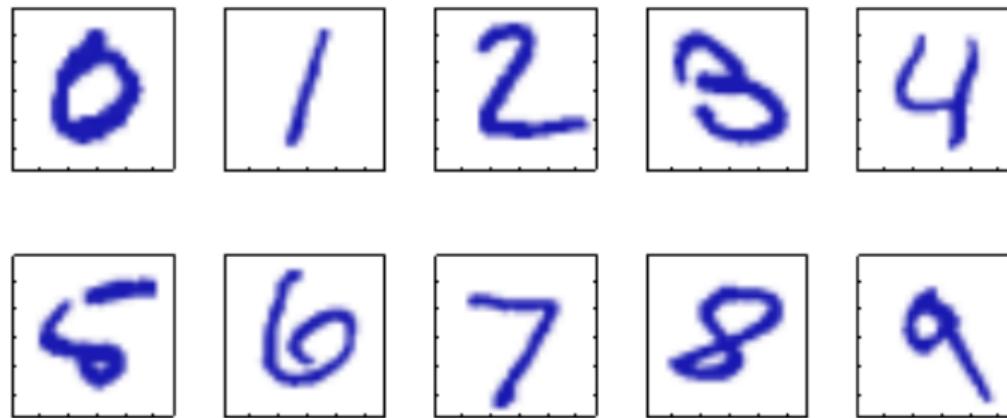
smart buildings



Machine Learning Objectives

- ① automatic discovery of **regularities** in data through computer algorithms
 - similar to statistics
- ② use knowledge acquired to take **actions**

Example: Written digit recognition



MNIST (figure from Bishop)

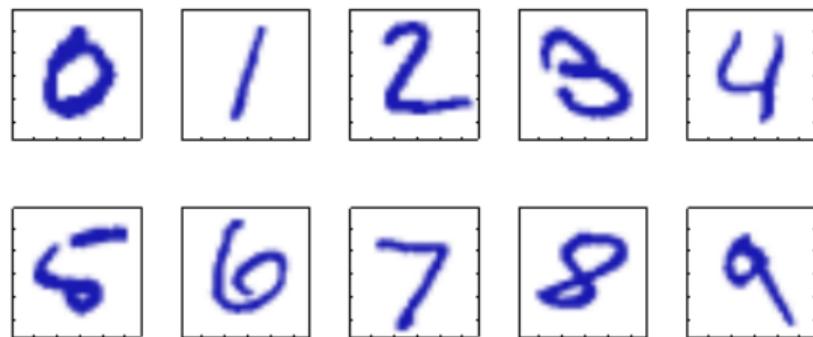
Example: Written digit recognition

Data:

- 28×28 grayscale pixels [0, 255]
- pre-processing: centering and normalization
- fixed length representation (784 dim)

Task

- from pixels classify one of 10 discrete digits



Formalization (Supervised Classification)

Training data:

- set of observations $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, $\mathbf{x}_i \in \mathbb{R}^D$
- set of target values $\{t_1, \dots, t_N\}$, $t_i \in \{c_1, \dots, c_K\}$



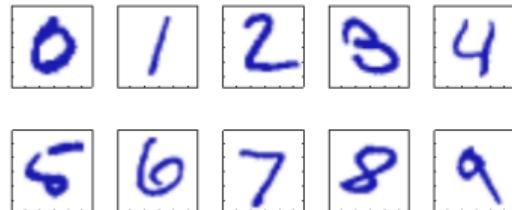
Goal

- find a function $y : \mathbb{R}^D \rightarrow \{c_1, \dots, c_K\}$ such that
- $y(\mathbf{x})$ gives correct answer for any unseen observation \mathbf{x}



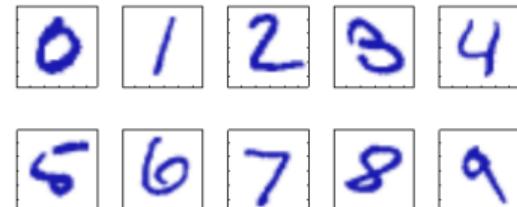
Key aspects

- Training data is **incomplete**
- Evaluation must be performed on unseen observations (test set)
- We need to ensure generalization
- data generation → measurements → feature extraction



Feature Extraction

- ➊ disregard irrelevant information
- ➋ reduce the dimensionality (complexity)



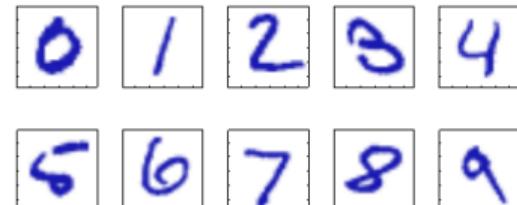
Classification vs Regression

Input x_i can be:

- discrete,
- continuous (\mathbb{R}),
- D dimensional (\mathbb{R}^D)

Classification:

- discrete targets: $t_i \in \{c_1, \dots, c_K\}$



Regression:

- continuous targets $t_i \in \mathbb{R}$
- can also be multi-dimensional

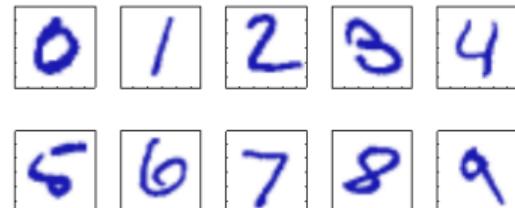
Supervised vs Unsupervised Learning

Unsupervised Learning

- we don't know the value of t_i
- data collection is cheap, but annotations are expensive
- find regularities in data

Applications

- Clustering: group data points according to distance metric
- Density estimation: find parametric model of complex distributions



Reinforcement Learning

- agent
- environment
- actions
- states
- reward

Differences from Supervised Learning

- reward not as detailed as targets
- reward can be delayed
- need to find responsibility of each actions to the reward

Other forms of Learning (Judea Pearl)

Levels	Activities
1) Associations	Seeing, hearing

Other forms of Learning (Judea Pearl)

Levels	Activities
1) Associations	Seeing, hearing
2) Intervention	Doing
3) Counterfactual	Imagining Retrospecting

In this course

- Supervised
 - Classification
 - Regression
- Unsupervised
 - Clustering
 - Density estimation
- Combined Supervised/Unsupervised
 - Example: Hidden Markov Models

Example: polynomial fitting

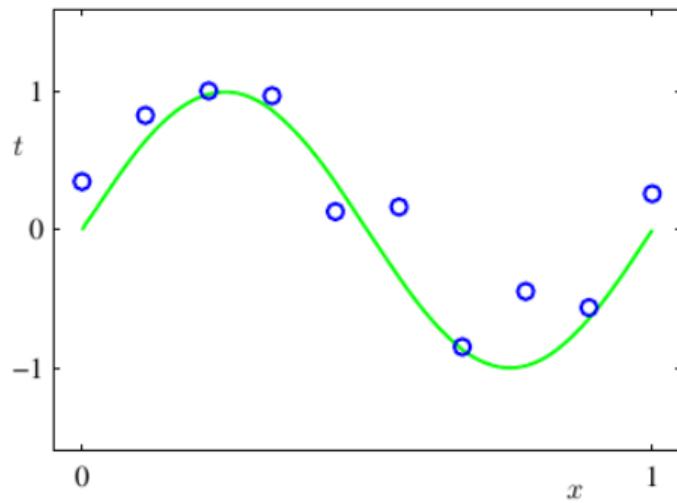
Data generation:

- $t = \sin(2\pi x) + \text{noise}$
- underlying regularity (sin)
- uncertainty (noise)

Model: polynomial

$$\begin{aligned}y(x, \omega) &= w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M \\&= \sum_{j=0}^M w_j x^j\end{aligned}$$

- non-linear in x
- linear in w



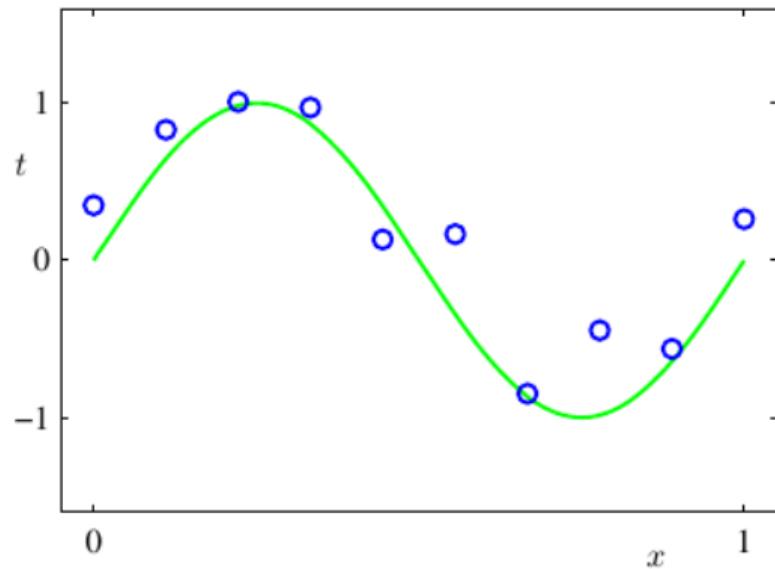
Example: polynomial fitting

Principled methods

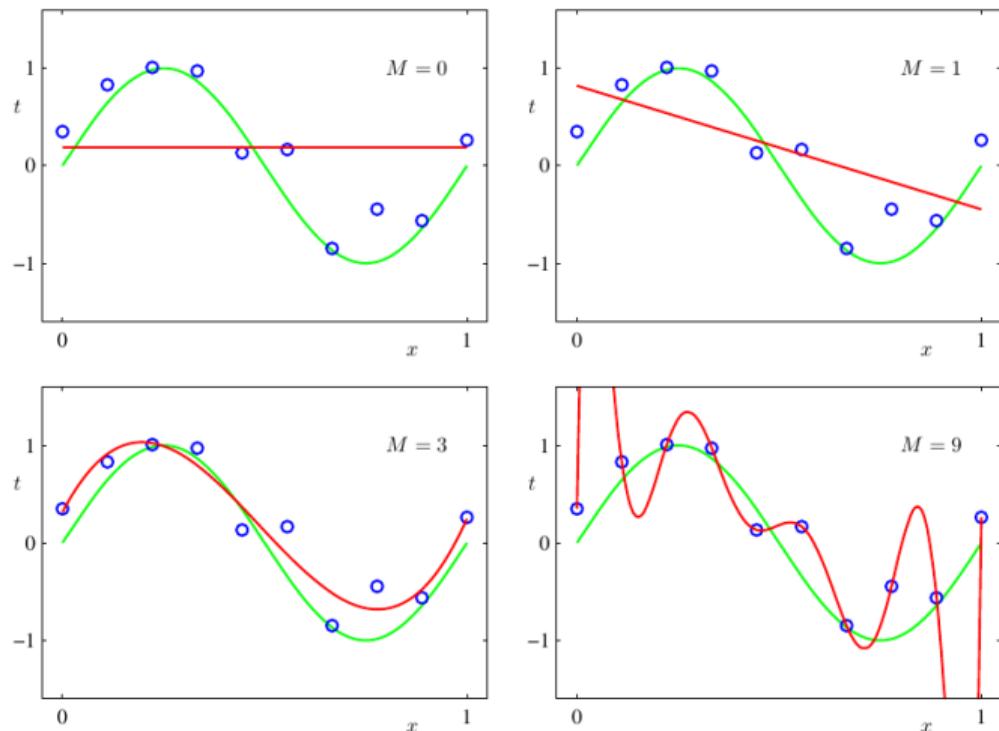
- backed up by a general theory
- in ML: probability theory

Heuristic methods

- based on common sense



Order of the polynomial (from Bishop)



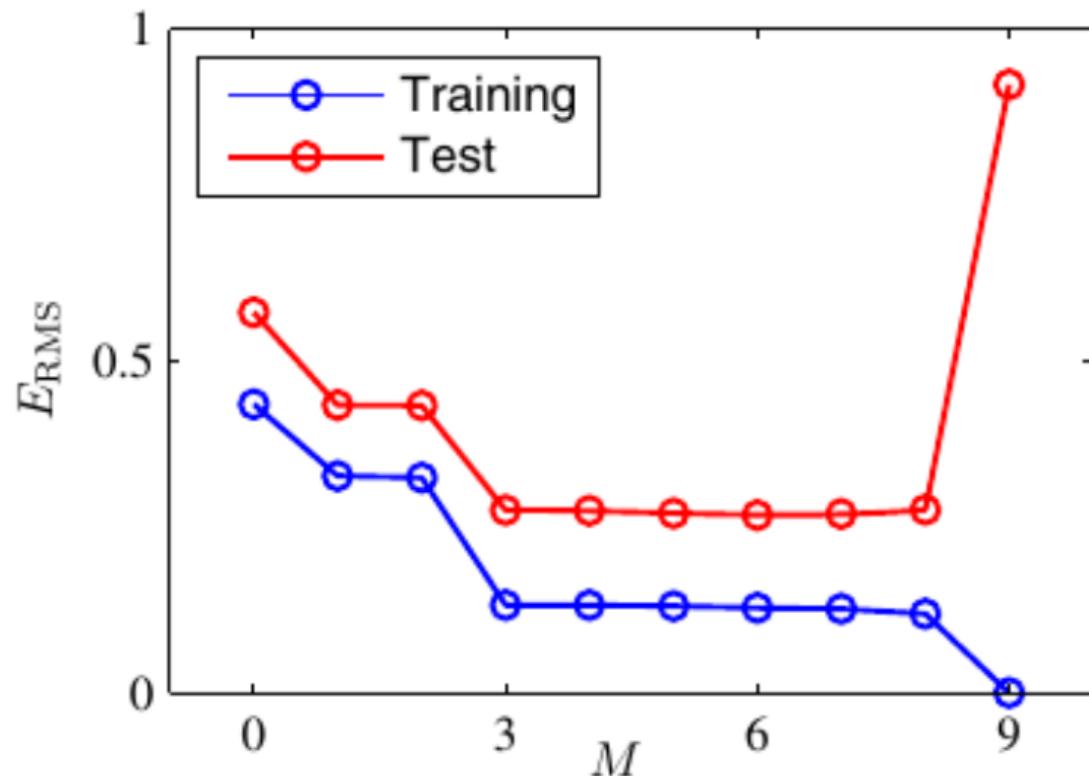
Model parameters (from Bishop)

	$M = 0$	$M = 1$	$M = 6$	$M = 9$
w_0^*	0.19	0.82	0.31	0.35
w_1^*		-1.27	7.99	232.37
w_2^*			-25.43	-5321.83
w_3^*			17.37	48568.31
w_4^*				-231639.30
w_5^*				640042.26
w_6^*				-1061800.52
w_7^*				1042400.18
w_8^*				-557682.99
w_9^*				125201.43

Overfitting: Training and Test set (from Bishop)

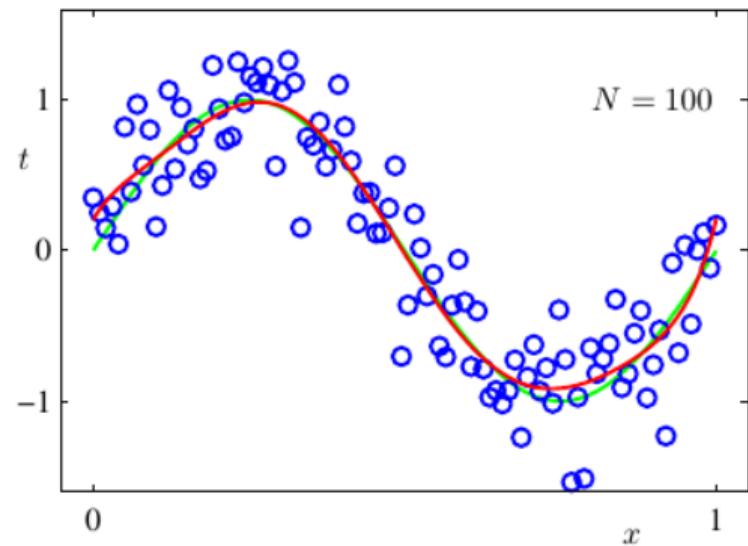
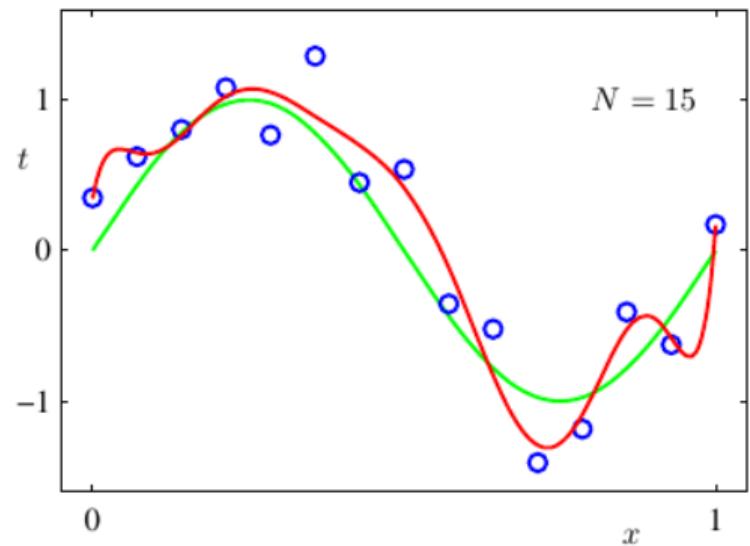
Root Mean Square Error

$$E_{\text{RMS}} = \sqrt{\frac{2E(w)}{N}}$$



Increasing training set size

parameters = 10



Increasing training set size

Problems:

- annotating data is expensive
- # parameters not equal to complexity
- we would like complexity of model to correspond to complexity of underlying phenomenon

Model Selection

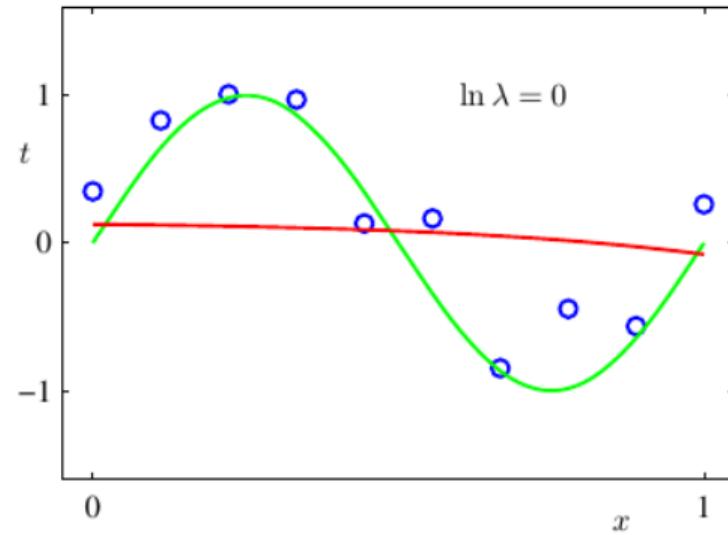
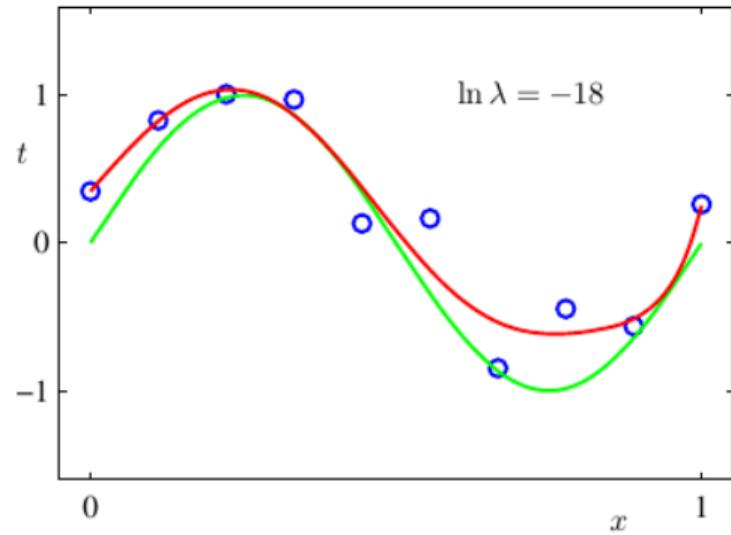
Choose the right complexity

Regularization

- Methods to reduce overfitting
- Heuristics: force model parameters to have small values
- Principled methods: use a priori information

Ridge Regression

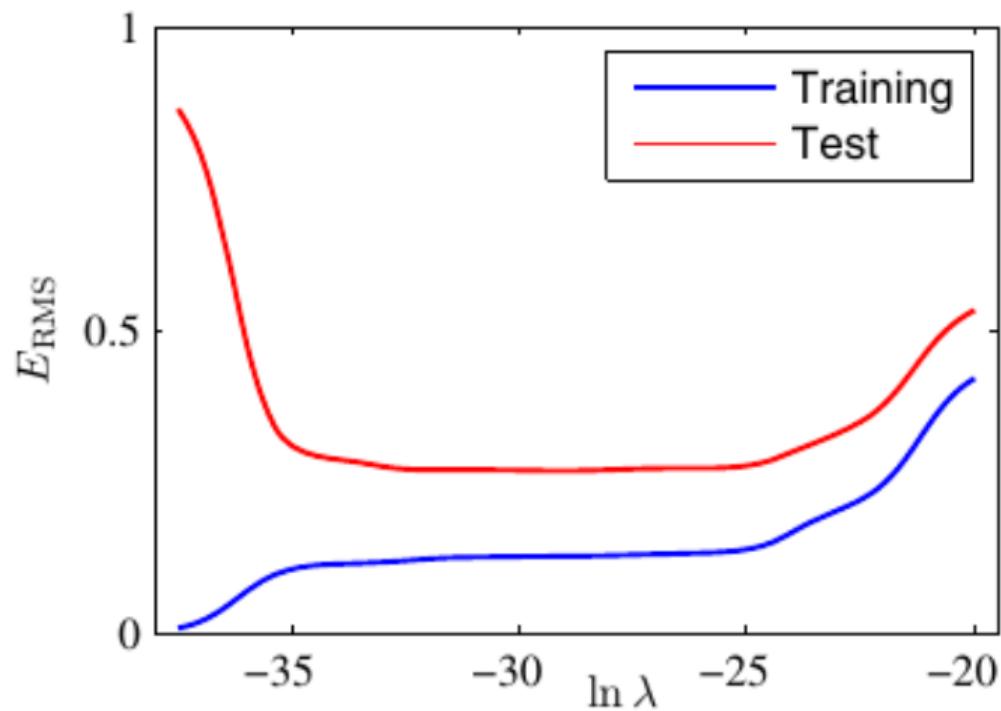
parameters = # data points



Ridge Regression

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
w_0^*	0.35	0.35	0.13
w_1^*	232.37	4.74	-0.05
w_2^*	-5321.83	-0.77	-0.06
w_3^*	48568.31	-31.97	-0.05
w_4^*	-231639.30	-3.89	-0.03
w_5^*	640042.26	55.28	-0.02
w_6^*	-1061800.52	41.32	-0.01
w_7^*	1042400.18	-45.95	-0.00
w_8^*	-557682.99	-91.53	0.00
w_9^*	125201.43	72.68	0.01

Ridge Regression



Probability, Decision and Information Theory

TTT4185 Machine Learning for Signal Processing

Giampiero Salvi

Department of Electronic Systems
NTNU

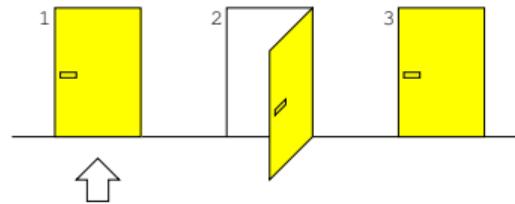
HT2021

Probability Theory in ML

incorporate probabilistic thinking at all levels

- start with incomplete knowledge (uncertainty)
- use observations to reduce uncertainty
- belief propagation

probability distributions as carriers of information¹



¹E T Jaynes. *Probability theory: The logic of science*. Ed. by G Larry Bretthorst. Cambridge university press, June 2003.

Engineering vs Science

Engineering:

- ML as collection of methods
- fine tune aspects to boost the results

Science:

- define unified theory
- give the deepest possible interpretation to the results

Engineering vs Science

Engineering:

- ML as collection of methods
- fine tune aspects to boost the results

Science:

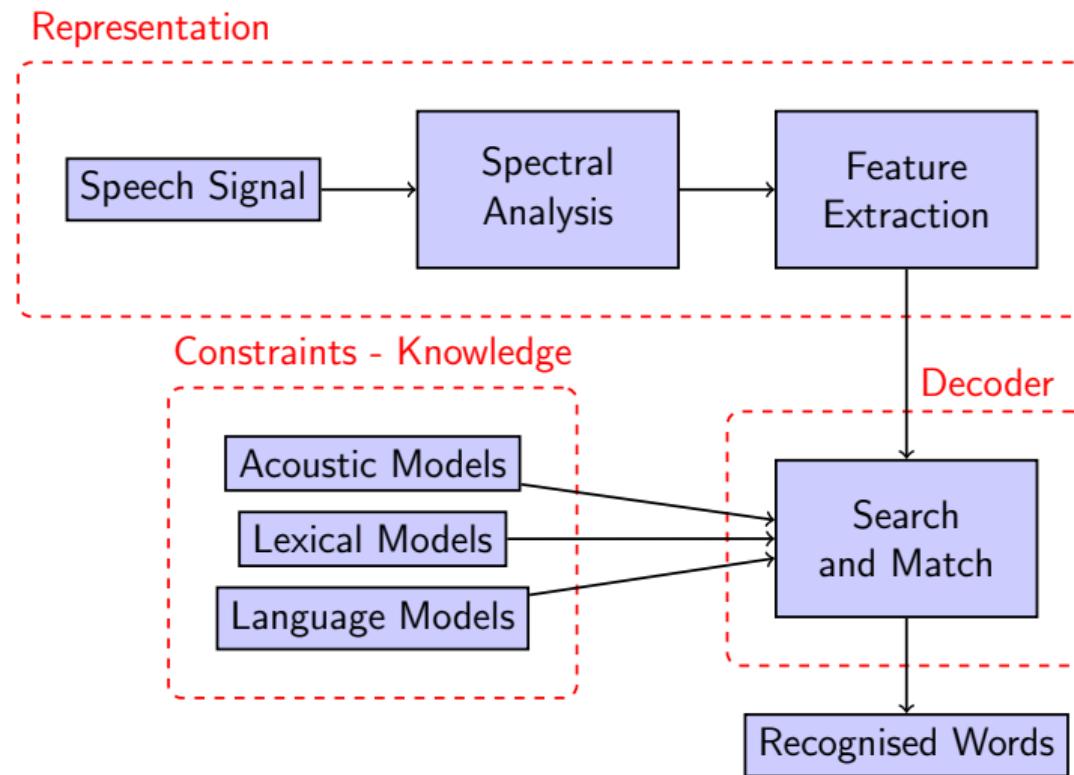
- define unified theory
- give the deepest possible interpretation to the results

reality not 100% clear cut

Advantages of Probability Based Methods

- **Results are interpretable.** More transparent and mathematically rigorous than methods such as *ANN*, *Evolutionary methods*.
- **Tool for interpreting other methods.** And make the assumptions explicit — *concept learning, least squares*.
- **Work with sparse training data.** More powerful than deterministic methods when training data is sparse (framework for including prior knowledge).
- **Belief Propagation:** Easy to merge different parts of a complex system and to update current knowledge with new observations.

Example: Automatic Speech Recognition



Advantages of Probability Based Methods, ctnd.

- **Shape a way of thinking.** All aspects of learning, modelling and inference can be cast under the same theory.

Disadvantages of Probability Based Methods

- **Often hard to derive closed solutions.** Need to resort to heuristic approximations.
- **Inefficient for large data sets.** But many argue that the need for large data set is a flaw in the methods.

Outline

1 Probability Theory Reminder

- Axioms and Properties
- Common Distributions
- Moments

2 Probabilistic Machine Learning

- Supervised Learning, General Definition
- Regression
- Classification
- Bayes decision theory

3 Information Theory

Outline

1 Probability Theory Reminder

- Axioms and Properties
- Common Distributions
- Moments

2 Probabilistic Machine Learning

- Supervised Learning, General Definition
- Regression
- Classification
- Bayes decision theory

3 Information Theory

Different views on probabilities

Axiomatic defines axioms and derives properties

Classical number of ways something can happen over total number of things that can happen (e.g. dice)

Logical same, but weight the different ways

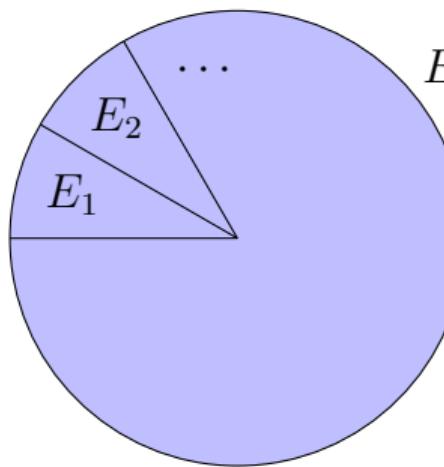
Frequency frequency of success in repeated experiments

Subjective degree of belief (basis for Bayesian statistics)

Axiomatic definition of probabilities (Kolmogorov)

Given an event E in a event space F

- ① $P(E) \geq 0$ for all $E \in F$
- ② sure event Ω : $P(\Omega) = 1$
- ③ E_1, E_2, \dots countable sequence of pairwise disjoint events, then

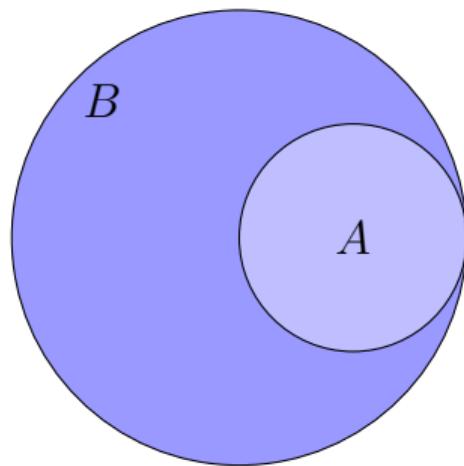


$$E_1 \cup E_2 \cup \dots$$

$$P(E_1 \cup E_2 \cup \dots) = \sum_{i=1}^{\infty} P(E_i)$$

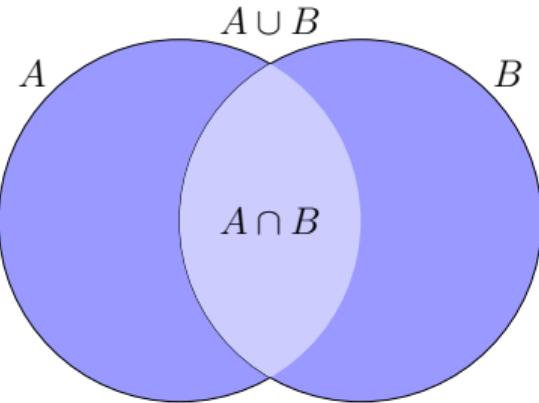
Consequences

- ① Monotonicity: $P(A) \leq P(B)$ if $A \subseteq B$
Example: $A = \{3\}$, $B = \{\text{odd}\}$
- ② Empty set \emptyset : $P(\emptyset) = 0$
Example:
 $P(A \cap B)$ where $A = \{\text{odd}\}, B = \{\text{even}\}$
- ③ Bounds: $0 \leq P(E) \leq 1$ for all $E \in F$



More Consequences: Addition

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

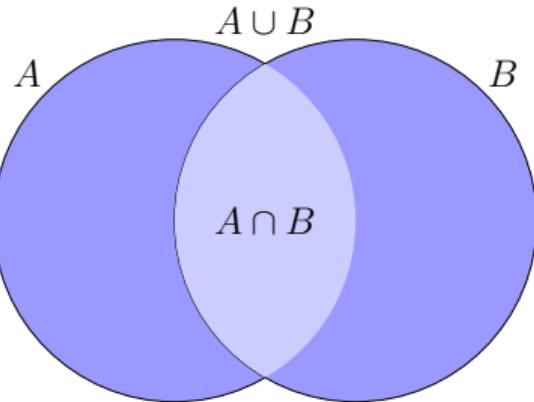


Example:

$$\begin{array}{lll} A & = & \{1, 3, 5\}, \quad P(A) & = & \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = \frac{1}{2} \\ B & = & \{5, 6\}, \quad P(B) & = & \frac{1}{6} + \frac{1}{6} = \frac{1}{3} \end{array}$$

More Consequences: Addition

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

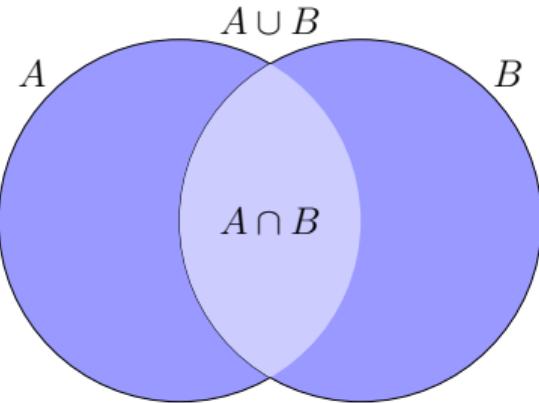


Example:

$$\begin{array}{llll} A & = & \{1, 3, 5\}, & P(A) = \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = \frac{1}{2} \\ B & = & \{5, 6\}, & P(B) = \frac{1}{6} + \frac{1}{6} = \frac{1}{3} \\ A \cap B & = & \{5\} & P(A \cap B) = \frac{1}{6} \end{array}$$

More Consequences: Addition

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$



Example:

$$A = \{1, 3, 5\}, \quad P(A) = \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = \frac{1}{2}$$

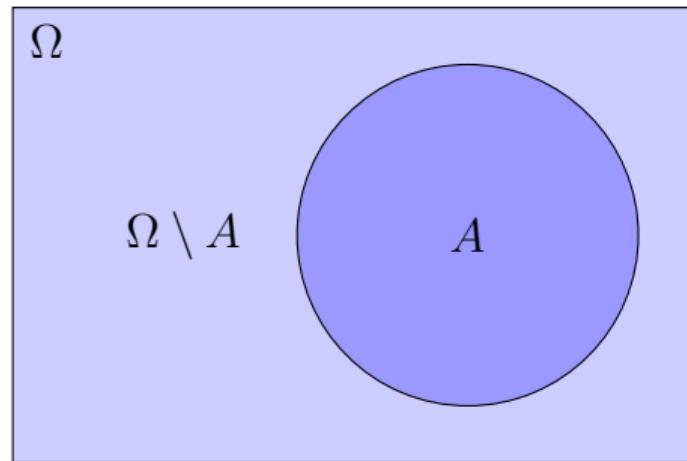
$$B = \{5, 6\}, \quad P(B) = \frac{1}{6} + \frac{1}{6} = \frac{1}{3}$$

$$A \cap B = \{5\} \quad P(A \cap B) = \frac{1}{6}$$

$$A \cup B = \{1, 3, 5, 6\} \quad P(A \cup B) = \frac{1}{6} + \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = \frac{2}{3}$$

More Consequences: Negation

$$P(\bar{A}) = P(\Omega \setminus A) = 1 - P(A)$$



Example:

$$\begin{aligned} A &= \{1, 2\}, & P(A) &= \frac{1}{6} + \frac{1}{6} = \frac{1}{3} \\ \bar{A} &= \{3, 4, 5, 6\}, & P(\bar{A}) &= \frac{1}{6} + \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = 1 - \frac{1}{3} \end{aligned}$$

Random (Stochastic) Variables

A random variable is a **function** that assigns a number x to the outcome of an experiment

- the result of flipping a coin,
- the result of measuring the temperature

The *probability distribution* $P(x)$ of a random variable (r.v.) captures the fact that

- the r.v. will have different values when observed **and**
- some values occur more than others.

Formal definition of RVs

$$RV = \{f : \mathcal{S}_a \rightarrow \mathcal{S}_b, P(x)\}$$

where:

\mathcal{S}_a = set of possible outcomes of the experiment

\mathcal{S}_b = domain of the variable

$f : \mathcal{S}_a \rightarrow \mathcal{S}_b$ = function mapping outcomes to values x

$P(x)$ = probability distribution function

Examples of RVs

Dice:

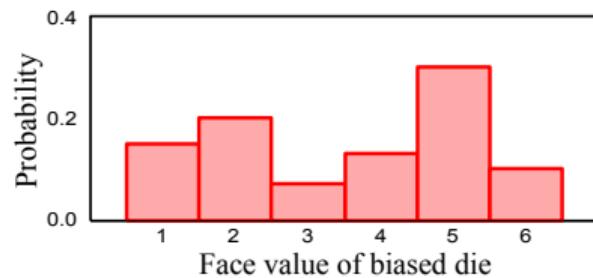
- $\mathcal{S}_a =$ the dice lands on one of the sides
- $\mathcal{S}_b =$ integer numbers $\{1, 2, 3, 4, 5, 6\}$
- $f : \mathcal{S}_a \rightarrow \mathcal{S}_b =$ assigns each side to a number
- $P(x) =$ uniform distribution for a fair dice

Temperature:

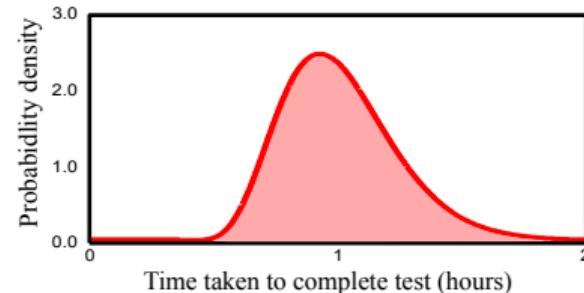
- $\mathcal{S}_a =$ degrees of expansion of mercury
- $\mathcal{S}_b =$ real numbers \mathbb{R}
- $f : \mathcal{S}_a \rightarrow \mathcal{S}_b =$ maps expansion to a real number (different maps for Celsius, Fahrenheit or absolute temperature)
- $P(x) =$ depends on the application

Types of Random Variables

- A **discrete random variable** takes values from a predefined set.
- For a **Boolean discrete random variable** this predefined set has two members - $\{0, 1\}$, $\{\text{yes}, \text{no}\}$ etc.
- A **continuous random variable** takes values that are real numbers.



discrete pdf



continuous pdf

Figures taken from **Computer Vision: models, learning and inference** by Simon Prince.

Examples of Random Variables



- Discrete events: either 1, 2, 3, 4, 5, or 6.
- Discrete probability distribution $p(x) = P(d = x)$
- $P(d = 1) = 1/6$ (fair dice)



- Any real number (theoretically infinite)
- Probability Density Function (PDF) $f(x)$ (**NOT PROBABILITY!!!**)
- $P(t = 36.6) = 0$
- $P(36.6 < t < 36.7) = 0.1$

Joint Probabilities

- Consider two random variables x and y .
- Observe multiple paired instances of x and y . Some paired outcomes will occur more frequently.
- This information is encoded in the joint probability distribution $P(x, y)$.
- $P(x)$ denotes the joint probability of $x = (x_1, \dots, x_K)$.



Joint Probabilities (cont.)

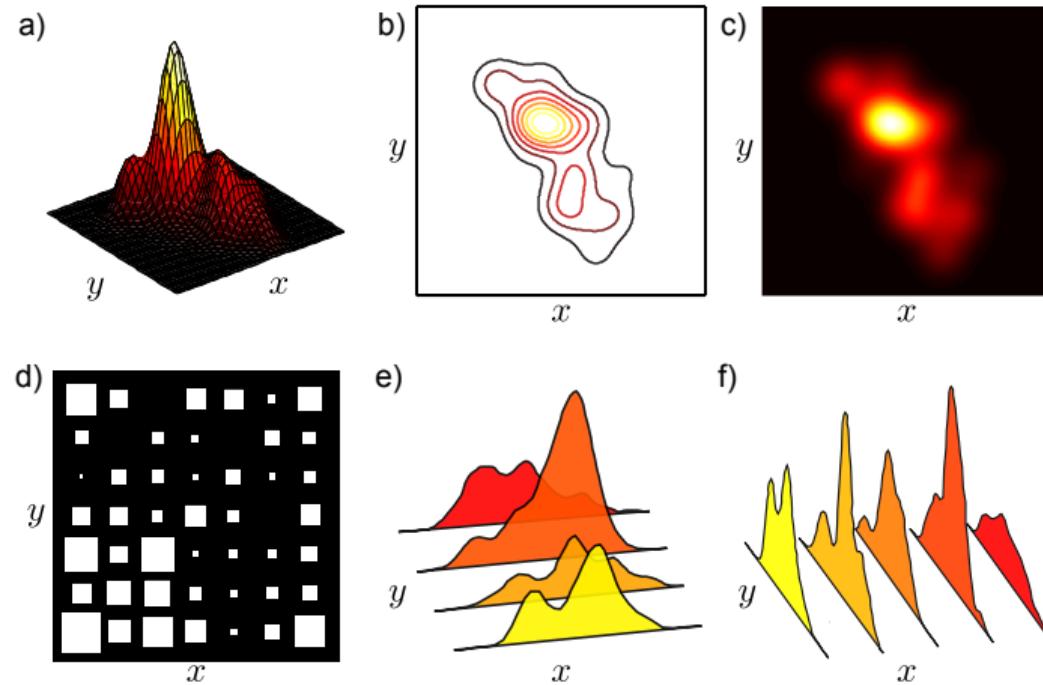


Figure from **Computer Vision: models, learning and inference** by Simon Prince.

Marginalization

The probability distribution of any single variable can be recovered from a joint distribution by summing for the discrete case

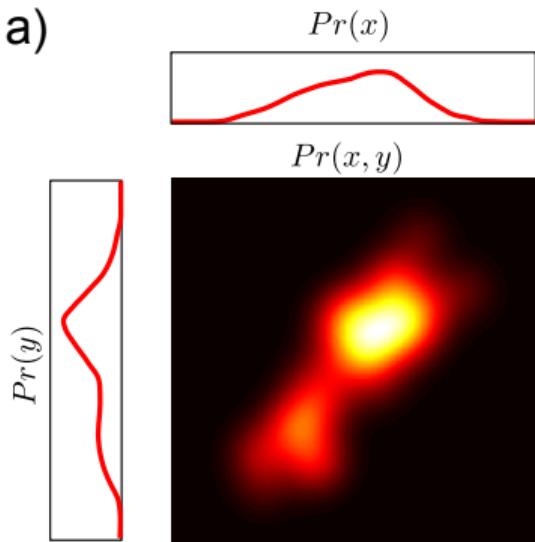
$$P(x) = \sum_y P(x, y)$$

and integrating for the continuous case

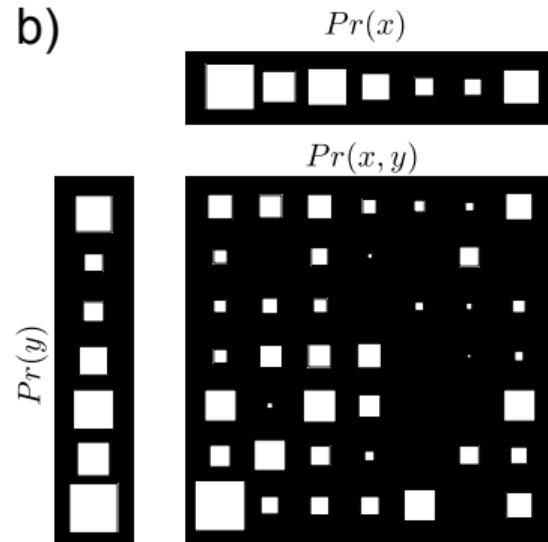
$$P(x) = \int_y P(x, y) dy$$

Marginalization (cont.)

a)



b)



c)

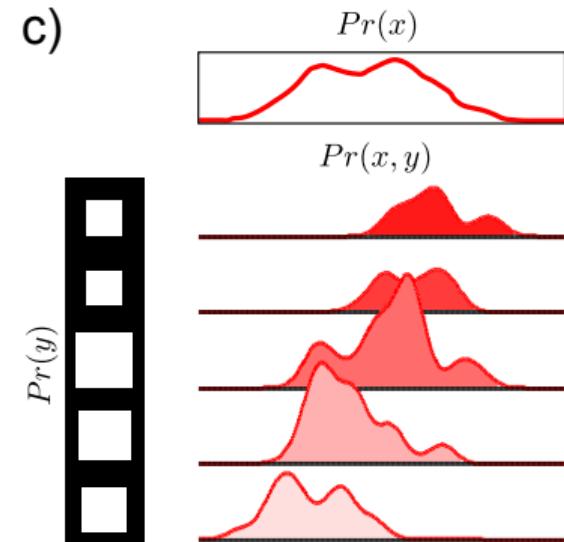


Figure from **Computer Vision: models, learning and inference** by Simon Prince.

Conditional Probabilities

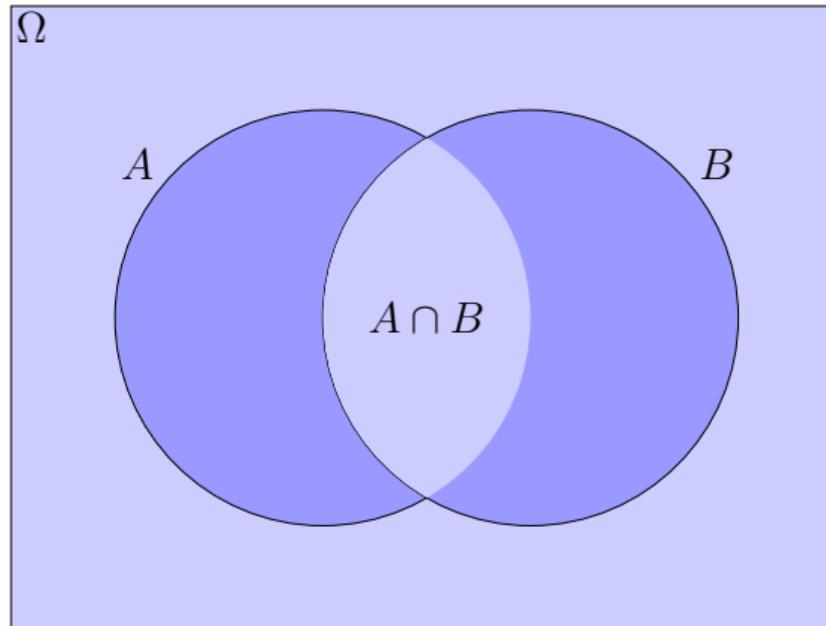
$$P(A|B)$$

The probability of event A when we *know* that event B has happened

Note: different from the probability that event A *and* event B will happen

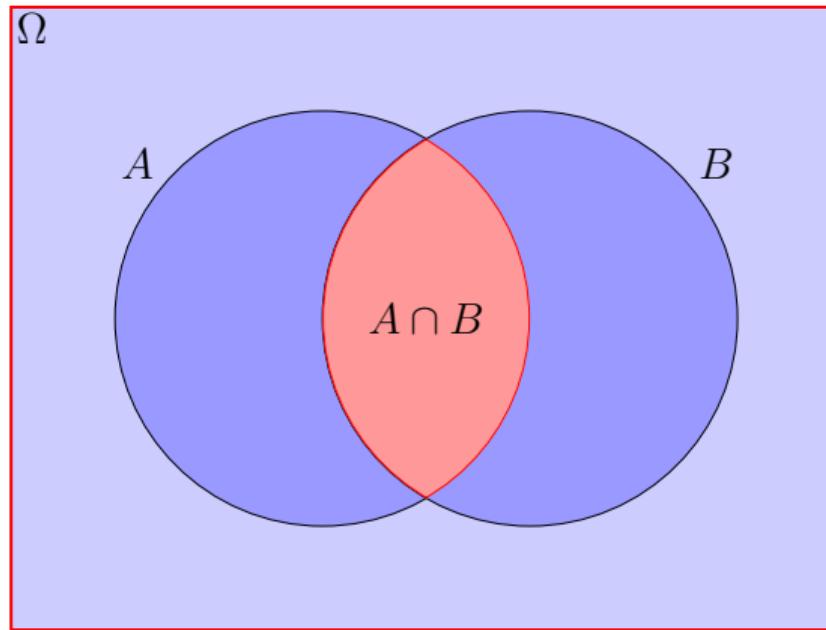
Conditional Probabilities

$$P(A|B) \neq P(A \cap B)$$



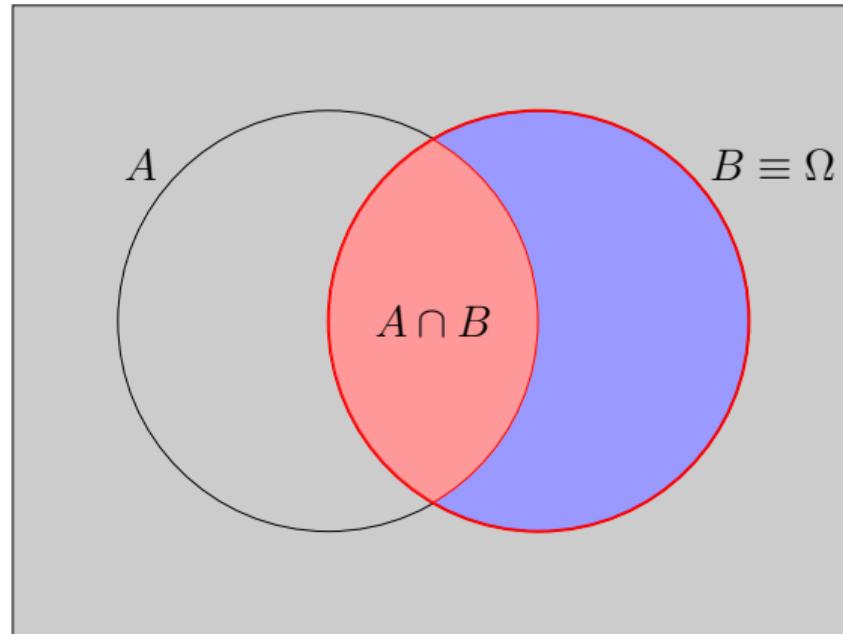
Conditional Probabilities

$$P(A|B) \neq P(A \cap B)$$



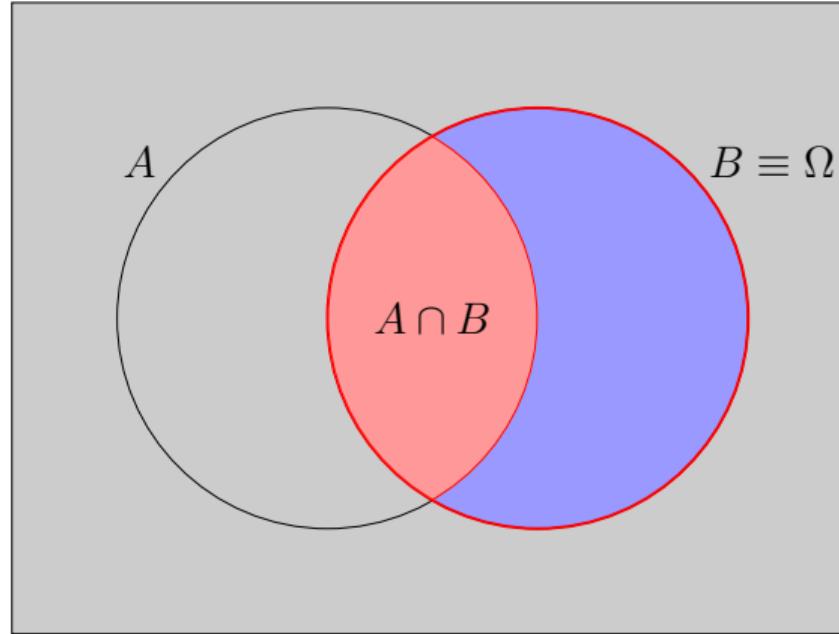
Conditional Probabilities

$$P(A|B) \neq P(A \cap B)$$



Conditional Probabilities

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

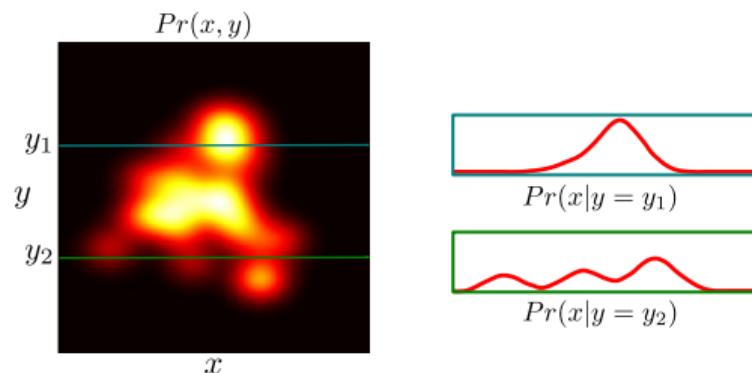


Conditional Probability (Random Variables)

- The conditional probability of x given that y takes value y^* indicates the different values of r.v. x which we'll observe given that y is fixed to value y^* .
- The conditional probability can be recovered from the joint distribution $P(x, y)$:

$$P(x | y = y^*) = \frac{P(x, y = y^*)}{P(y = y^*)} = \frac{P(x, y = y^*)}{\int_x P(x, y = y^*) dx}$$

- Extract an appropriate slice, and then normalize it.



Independence

- two events are independent if the joint distribution can be factorized:
 $P(A \cap B) = P(A)P(B)$
- this means that:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A)P(B)}{P(B)} = P(A)$$

Independence

- two events are independent if the joint distribution can be factorized:
 $P(A \cap B) = P(A)P(B)$
- this means that:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A)P(B)}{P(B)} = P(A)$$

knowing that B has happened does not tell us anything about A

Bayes' Rule

if

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

then

$$P(A \cap B) = P(A|B)P(B)$$

Bayes' Rule

if

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

then

$$P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$$

Bayes' Rule

if

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

then

$$P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$$

and

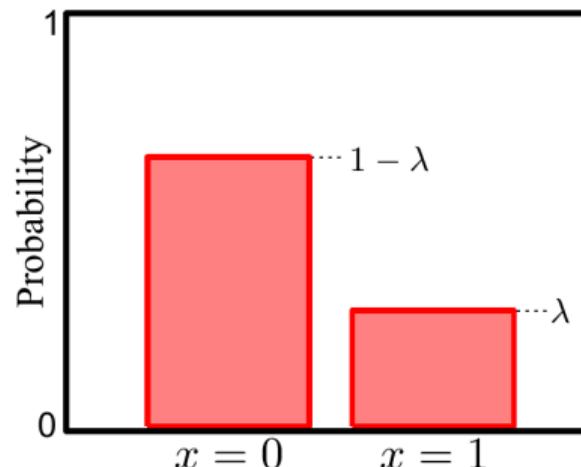
$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Bernoulli: binary variables

- Domain: binary variables ($x \in \{0, 1\}$)
- Parameters: $\lambda = Pr(x = 1)$, $\lambda \in [0, 1]$

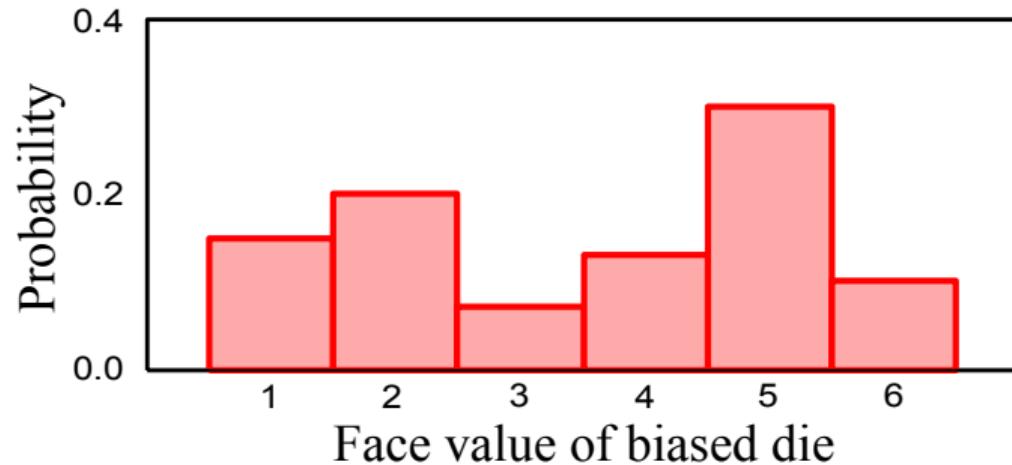
Then $Pr(x = 0) = 1 - \lambda$, and

$$Pr(x) = \lambda^x(1 - \lambda)^{1-x} = \begin{cases} \lambda, & \text{if } x = 1, \\ 1 - \lambda, & \text{if } x = 0 \end{cases}$$



Categorical

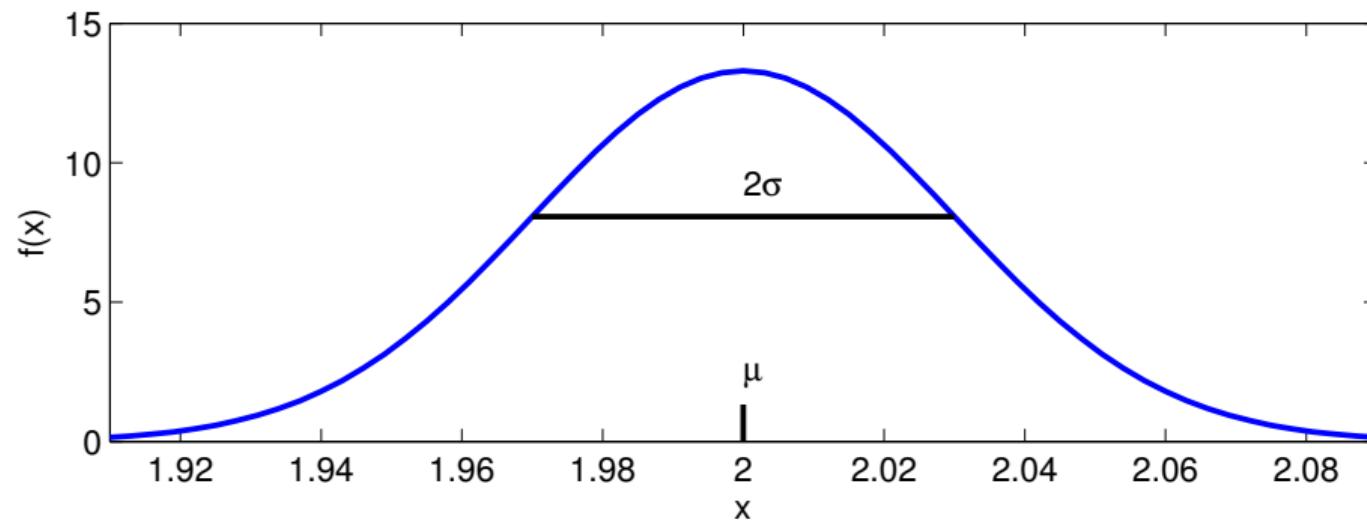
- Domain: discrete variables ($x \in \{x_1, \dots, x_K\}$)
- Parameters: $\lambda = [\lambda_1, \dots, \lambda_K]$
- with $\lambda_k \in [0, 1]$ and $\sum_{k=1}^K \lambda_k = 1$



Gaussian distributions: One-dimensional

- aka univariate normal distribution
- Domain: real numbers ($x \in \mathbb{R}$)

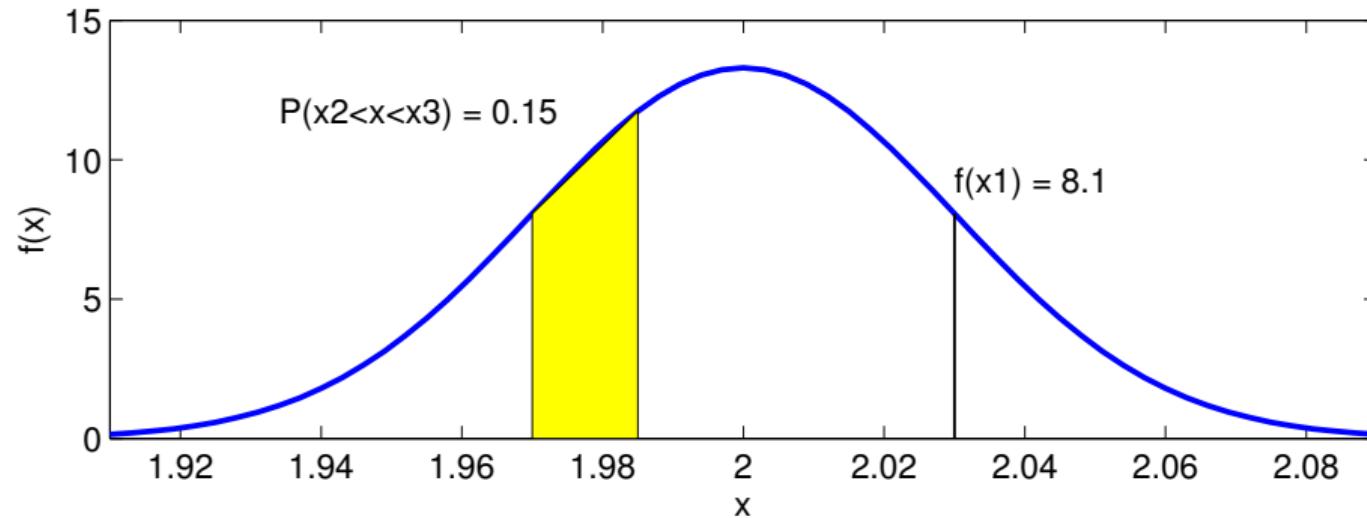
$$f(x|\mu, \sigma^2) = N(\mu, \sigma^2) = N(\mu, \beta^{-1}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$



Gaussian distributions: One-dimensional

- aka univariate normal distribution
- Domain: real numbers ($x \in \mathbb{R}$)

$$f(x|\mu, \sigma^2) = N(\mu, \sigma^2) = N(\mu, \beta^{-1}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$



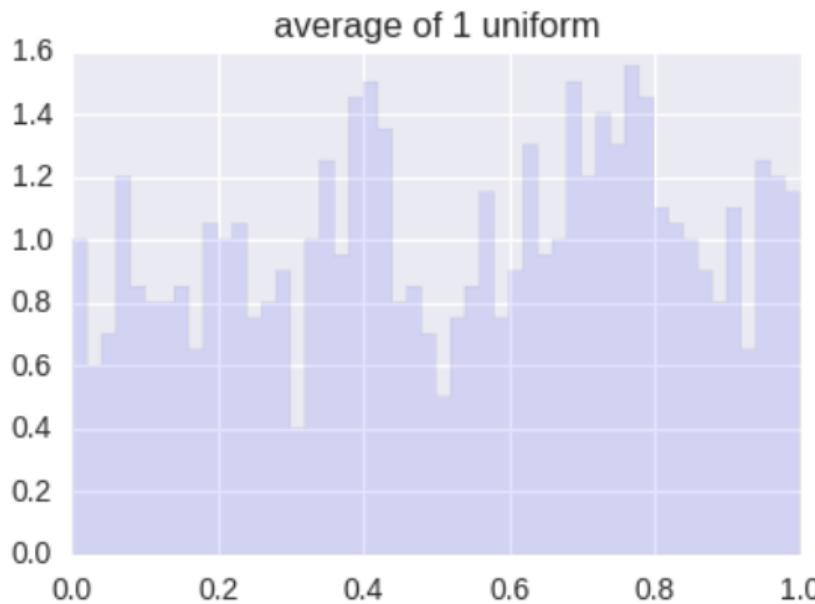
Why Gaussian: Central Limit Theorem

Galton Board (Sir Francis Galton, 1822-1911)



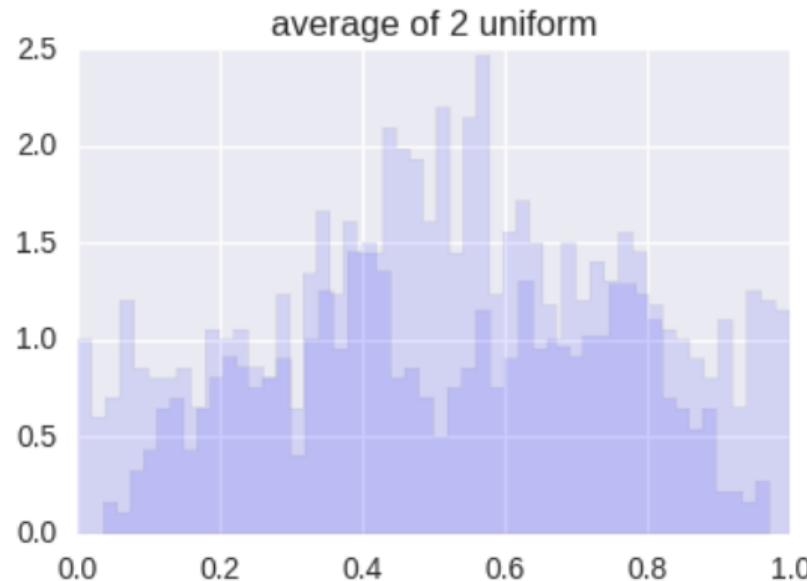
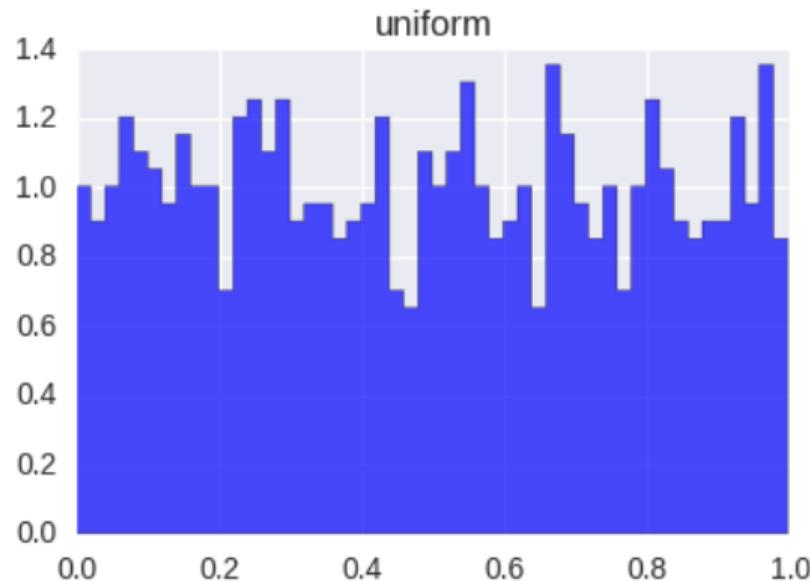
Why Gaussian: Central Limit Theorem

The distribution of the sum (or average) of a large number of independent, identically distributed variables will be approximately normal, regardless of the underlying distribution.².



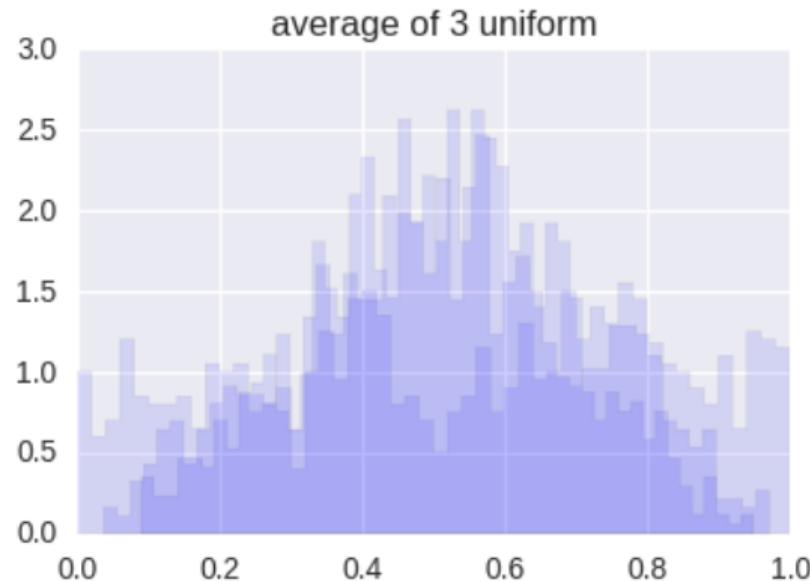
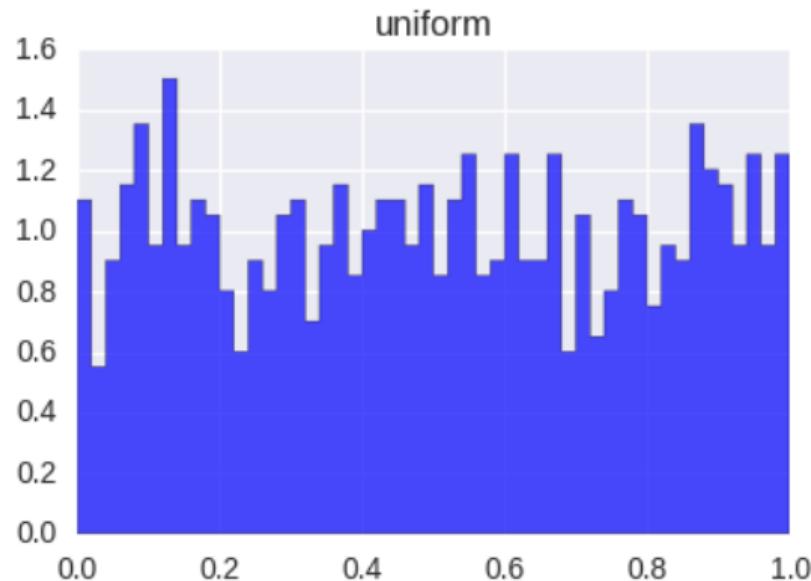
Why Gaussian: Central Limit Theorem

The distribution of the sum (or average) of a large number of independent, identically distributed variables will be approximately normal, regardless of the underlying distribution.².



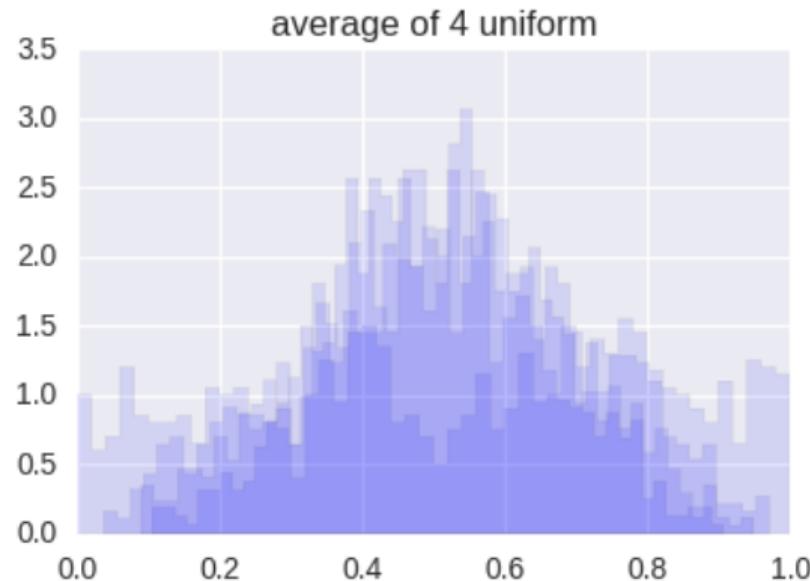
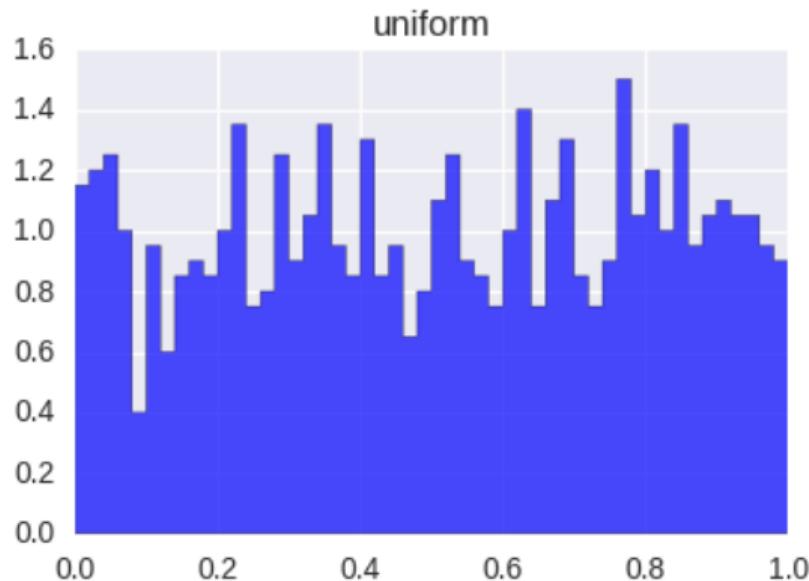
Why Gaussian: Central Limit Theorem

The distribution of the sum (or average) of a large number of independent, identically distributed variables will be approximately normal, regardless of the underlying distribution.².



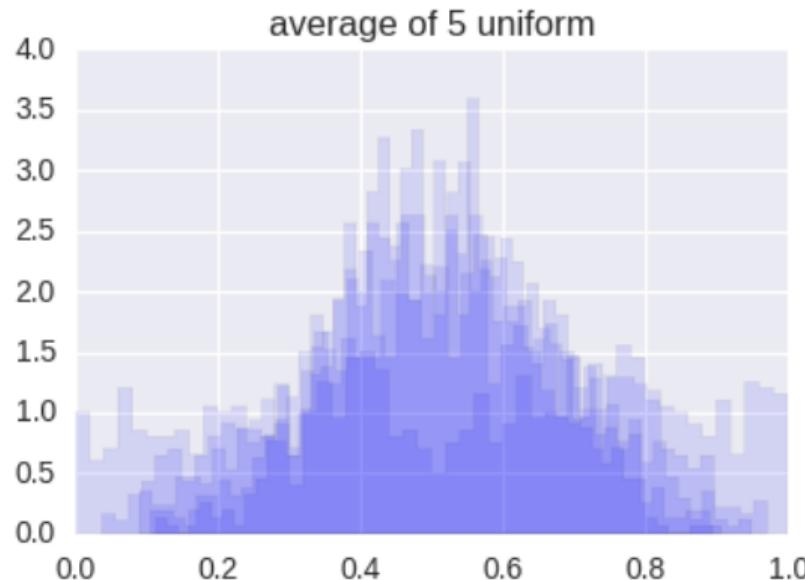
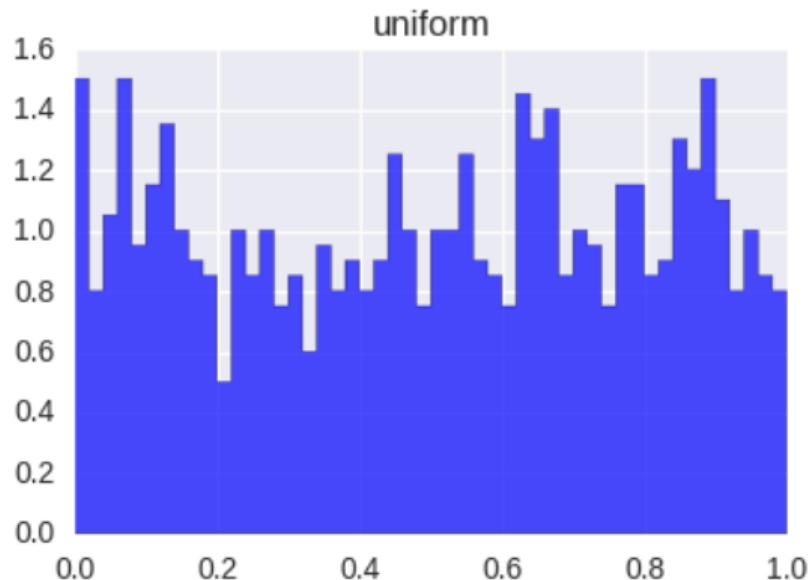
Why Gaussian: Central Limit Theorem

The distribution of the sum (or average) of a large number of independent, identically distributed variables will be approximately normal, regardless of the underlying distribution.².



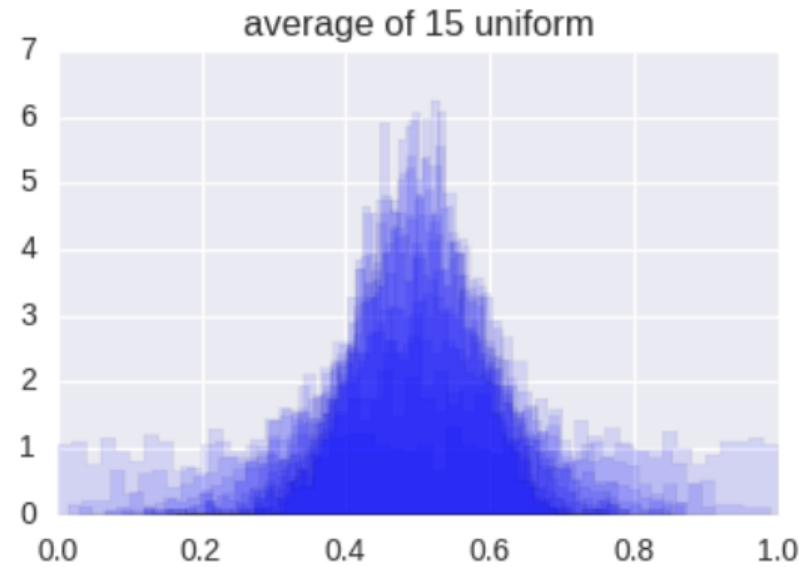
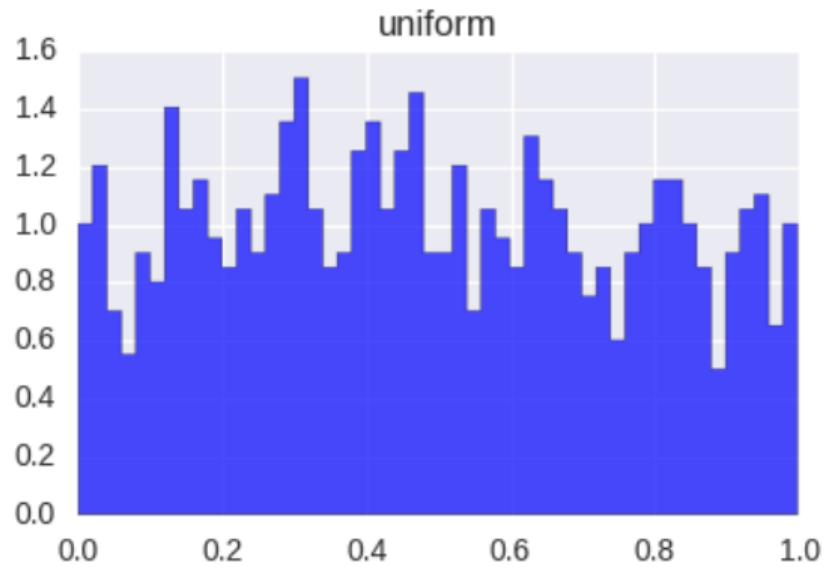
Why Gaussian: Central Limit Theorem

The distribution of the sum (or average) of a large number of independent, identically distributed variables will be approximately normal, regardless of the underlying distribution.².



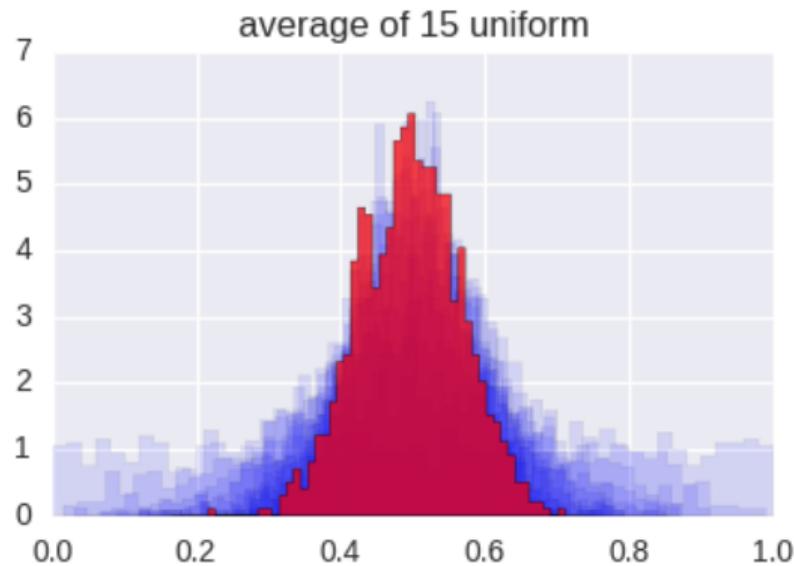
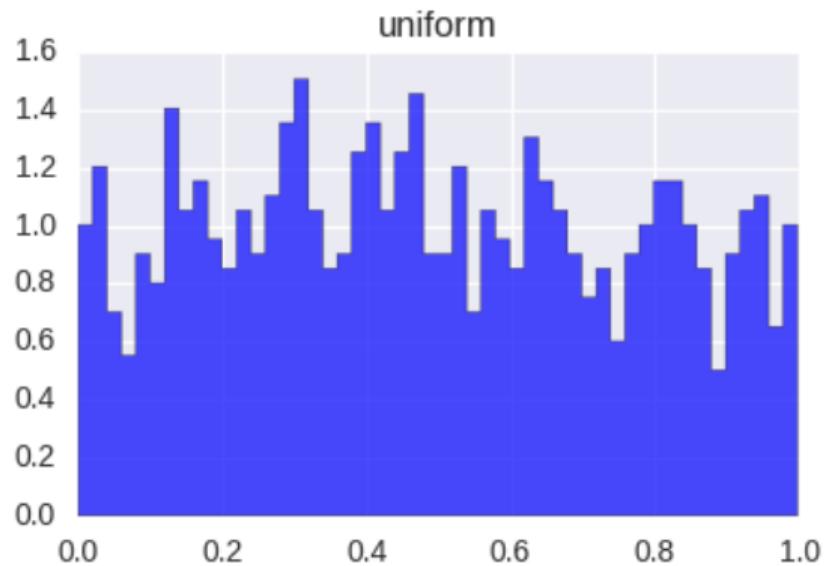
Why Gaussian: Central Limit Theorem

The distribution of the sum (or average) of a large number of independent, identically distributed variables will be approximately normal, regardless of the underlying distribution.².



Why Gaussian: Central Limit Theorem

The distribution of the sum (or average) of a large number of independent, identically distributed variables will be approximately normal, regardless of the underlying distribution.².



Gaussian distributions: D Dimensions

- aka multivariate normal distribution
- Domain: real numbers ($\mathbf{x} \in \mathbb{R}^D$)

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_D \end{bmatrix} \quad \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \dots \\ \mu_D \end{bmatrix} \quad \Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \dots & \sigma_{1D} \\ \sigma_{21} & \sigma_2^2 & \dots & \\ \dots & & & \\ \sigma_{D1} & \dots & & \sigma_D^2 \end{bmatrix}$$

$$f(\mathbf{x}|\mu, \Sigma) = \frac{\exp\left[-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)\right]}{(2\pi)^{\frac{D}{2}} |\Sigma|^{\frac{1}{2}}}$$

Gaussian distributions

$$f(\mathbf{x}|\mu, \Sigma) = \frac{\exp\left[-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)\right]}{(2\pi)^{\frac{D}{2}} |\Sigma|^{\frac{1}{2}}}$$

Eigenvalue decomposition of the covariance matrix:

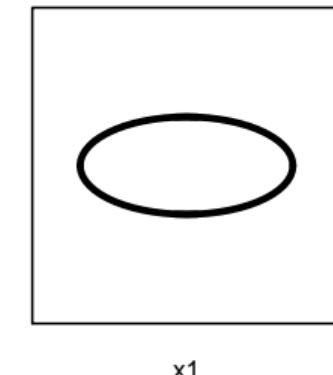
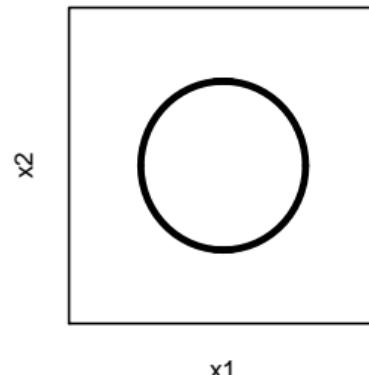
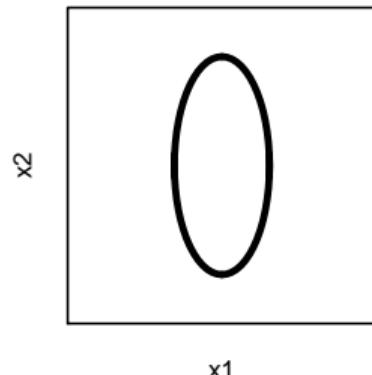
$$\Sigma = \lambda \ R \ \Sigma_{\text{diag}} \ R^T$$

Gaussian distributions

$$f(\mathbf{x}|\mu, \Sigma) = \frac{\exp\left[-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)\right]}{(2\pi)^{\frac{D}{2}} |\Sigma|^{\frac{1}{2}}}$$

Eigenvalue decomposition of the covariance matrix:

$$\Sigma = \lambda \ R \ \textcolor{red}{\Sigma_{\text{diag}}} \ R^T$$

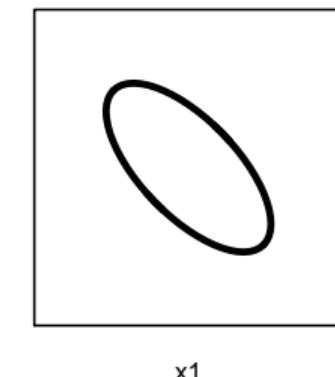
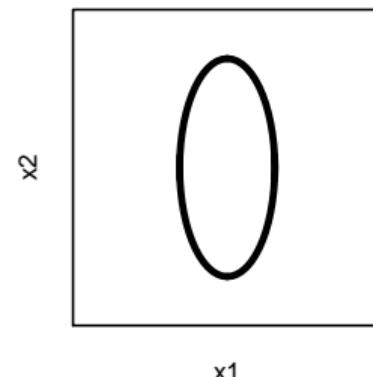
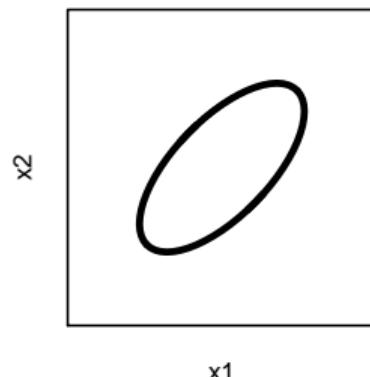


Gaussian distributions

$$f(\mathbf{x}|\mu, \Sigma) = \frac{\exp\left[-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)\right]}{(2\pi)^{\frac{D}{2}} |\Sigma|^{\frac{1}{2}}}$$

Eigenvalue decomposition of the covariance matrix:

$$\Sigma = \lambda \textcolor{red}{R} \Sigma_{\text{diag}} \textcolor{red}{R}^T$$

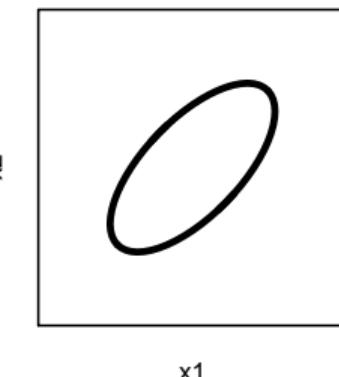
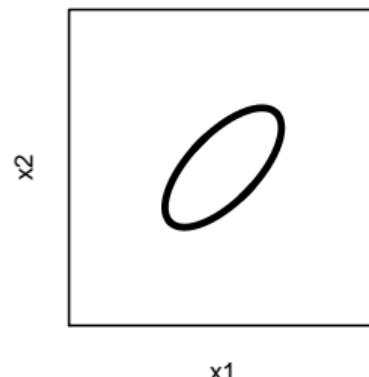
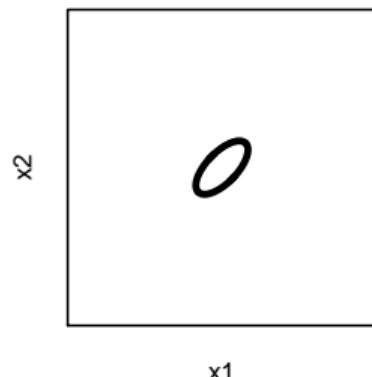


Gaussian distributions

$$f(\mathbf{x}|\mu, \Sigma) = \frac{\exp\left[-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)\right]}{(2\pi)^{\frac{D}{2}} |\Sigma|^{\frac{1}{2}}}$$

Eigenvalue decomposition of the covariance matrix:

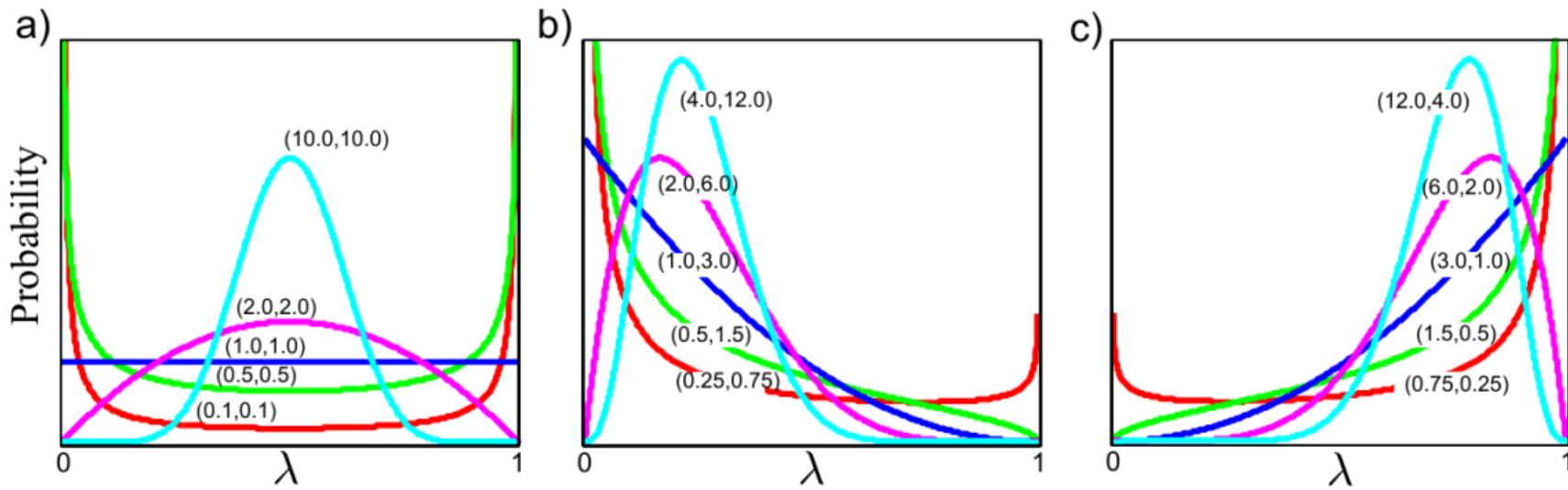
$$\Sigma = \lambda R \Sigma_{\text{diag}} R^T$$



Beta and Dirichlet (PDF over Probabilities)

Beta

- Domain: real numbers, bounded ($\lambda \in [0, 1]$)
- Parameters: $\alpha, \beta \in \mathbb{R}_+$
- describes probability of parameter λ in Bernoulli



Beta and Dirichlet (PDF over Probabilities)

Beta

- Domain: real numbers, bounded ($\lambda \in [0, 1]$)
- Parameters: $\alpha, \beta \in \mathbb{R}_+$
- describes probability of parameter λ in Bernoulli

Dirichlet

- Domain: K real numbers, bounded ($\lambda_1, \dots, \lambda_K \in [0, 1]$)
- Parameters: $\alpha_1, \dots, \alpha_K \in \mathbb{R}_+$
- describes probability of parameters λ_k in Categorical

Expected value

$$\mathbb{E}[\mathbf{x}] = \mu(\mathbf{x}) = \int \mathbf{x} p(\mathbf{x}) d\mathbf{x}$$

- Shows the “center of gravity” of a distribution
- Sampled expected value (mean)

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_i^N \mathbf{x}_i$$

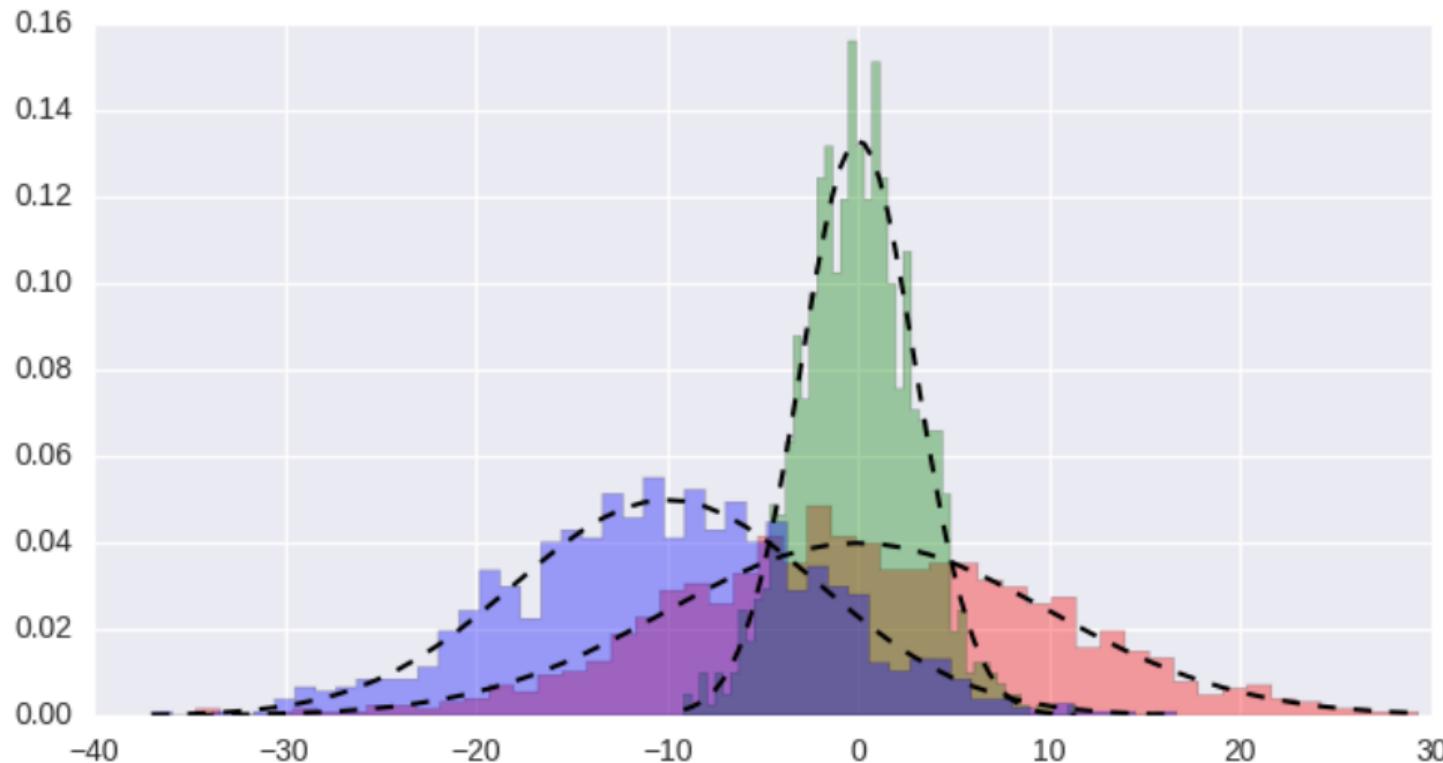
Variance

$$\sigma^2(\mathbf{x}) = \text{var}[\mathbf{x}] = \mathbb{E}[(\mathbf{x} - \mathbb{E}[\mathbf{x}])^2]$$

- Shows the “spread” of a distribution
- Sample variance

$$\overline{\sigma^2(\mathbf{x})} = \frac{1}{N-1} \sum_i^N (\mathbf{x}_i - \mu(\mathbf{x}_i))^2$$

Examples



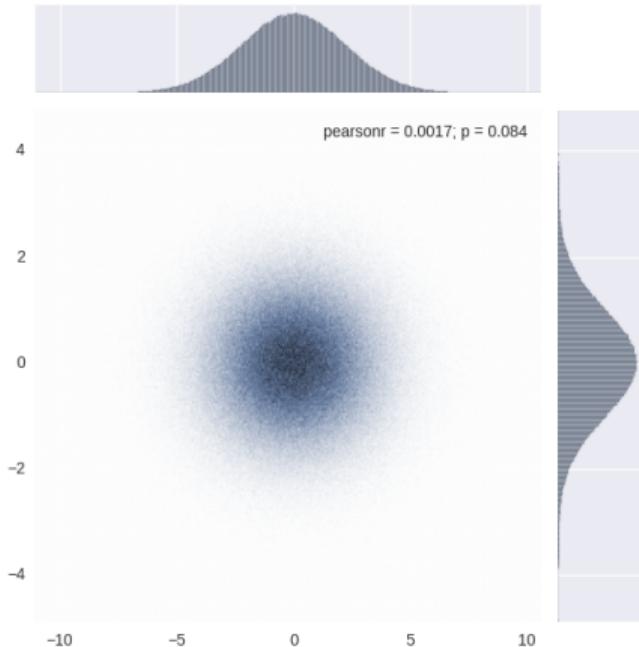
Covariance

$$\sigma(\mathbf{x}, \mathbf{y}) = \mathbb{E}[(\mathbf{x} - \mathbb{E}[\mathbf{x}])(\mathbf{y} - \mathbb{E}[\mathbf{y}])]$$

- Shows how the “spread” of how two variables vary *together*
- Sample co-variance

$$\overline{\sigma(\mathbf{x}, \mathbf{y})} = \frac{1}{N-1} \sum_i^N (\mathbf{x}_i - \mu(\mathbf{x}_i))(\mathbf{y}_i - \mu(\mathbf{y}))$$

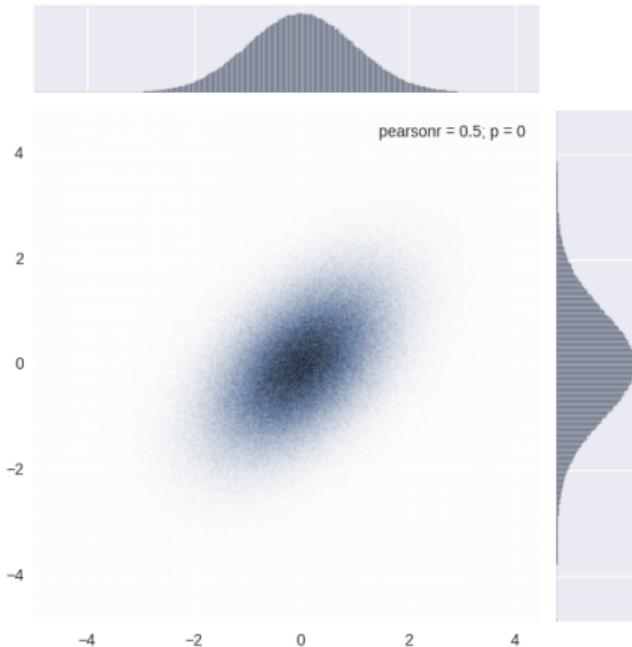
Examples



4

⁴Script by C.E. Ek

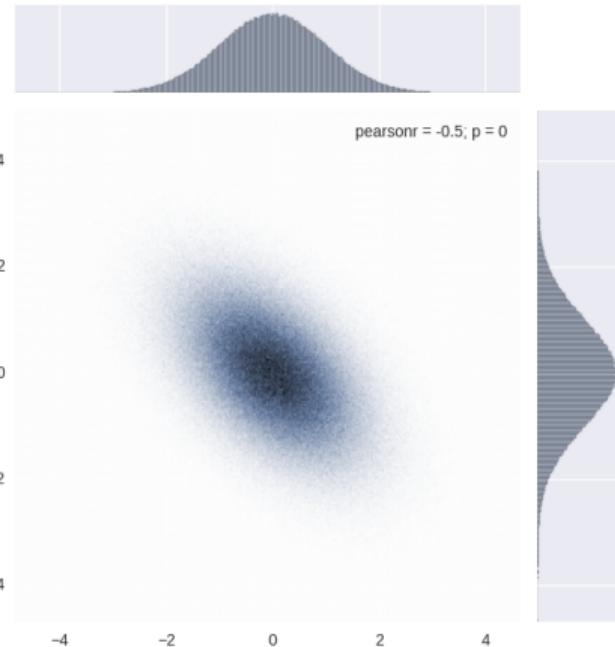
Examples



4

⁴Script by C.E. Ek

Examples

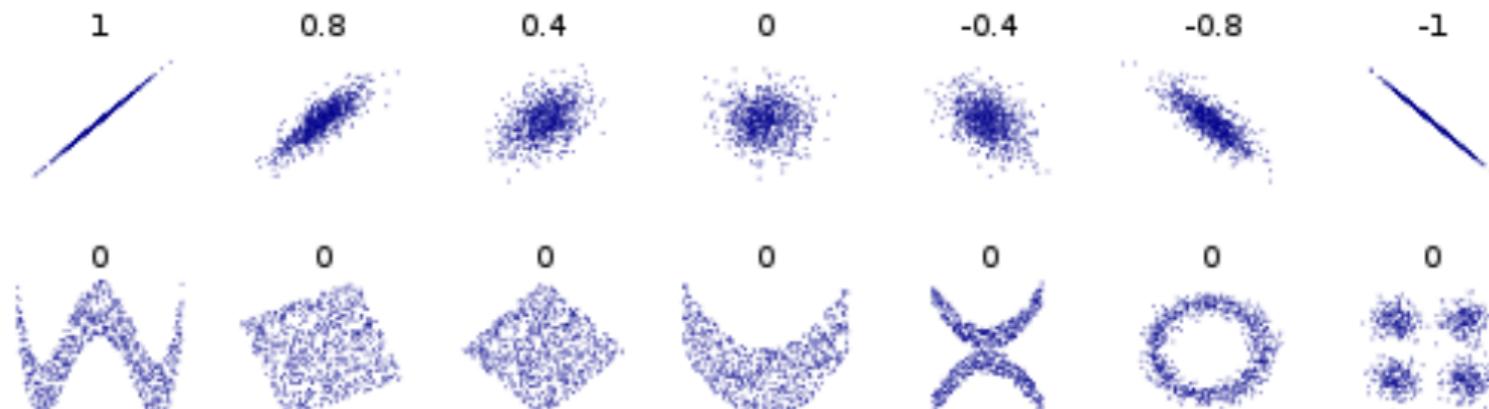


4

⁴Script by C.E. Ek

Covariance and Independence

- covariance is “linear” dependency
- dependent variables may have zero covariance
- in some distributions zero covariance is equivalent to independence



5

⁵Figure adapted from Wikipedia

Covariance and Independence (Gaussian)

- covariance is “linear” dependency
- dependent variables may have zero covariance
- in Gaussian (and few other distribution) zero covariance is equivalent to independence

$$f(\mathbf{x}|\mu, \Sigma) = \frac{\exp\left[-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)\right]}{(2\pi)^{\frac{D}{2}} |\Sigma|^{\frac{1}{2}}}$$

Outline

1 Probability Theory Reminder

- Axioms and Properties
- Common Distributions
- Moments

2 Probabilistic Machine Learning

- Supervised Learning, General Definition
- Regression
- Classification
- Bayes decision theory

3 Information Theory

General ML problem (supervised learning)

Data:

$$\{(\mathbf{x}^1, t^1), (\mathbf{x}^2, t^2), \dots, (\mathbf{x}^n, t^n)\}$$

Where \mathbf{x} are features, and t is the answer (target)

- if t is discrete: classification
- if t is continuous: regression

General ML problem (supervised learning)

Data:

$$\{(\mathbf{x}^1, t^1), (\mathbf{x}^2, t^2), \dots, (\mathbf{x}^n, t^n)\}$$

Where \mathbf{x} are features, and t is the answer (target)

- if t is discrete: classification
- if t is continuous: regression

Learning: we observe several examples of \mathbf{x} and we know t

- we can estimate $P(t)$ and $P(\mathbf{x}|t)$

Inference: we want to know t' given a new \mathbf{x}'

- we want to estimate $P(t'|\mathbf{x}')$

Bayes' Rule

$$P(t | \mathbf{x}) = \frac{P(\mathbf{x} | t)P(t)}{P(\mathbf{x})}$$

- $P(\mathbf{x} | t)$ ← **Likelihood** represents the probability of observing data \mathbf{x} given the hypothesis t .
- $P(t)$ ← **Prior** represents the knowledge on hypothesis t before any observation.
- $P(t | \mathbf{x})$ ← **Posterior** represents the probability of hypothesis t after the data \mathbf{x} has been observed.
- $P(\mathbf{x})$ ← **Evidence** encodes the quality of the underlying model.

$$P(\mathbf{x}) = \begin{cases} \sum_t P(\mathbf{x} | t)P(t) & \text{classification} \\ \int_t P(\mathbf{x} | t)P(t) & \text{regression} \end{cases}$$

Probabilistic Regression

Regression as conditional probability

- use multivariate joint distribution between \mathbf{x} and t
- find posterior $p(t|\mathbf{x}, \theta)$ of t by conditioning on \mathbf{x}

Explicit regression model:

- define a deterministic model $t = y(\mathbf{x}, \theta) + \epsilon$
- describe probability distribution of error ϵ

Implicit model: Gaussian Processes

Example: bivariate Normal distribution

Define joint probability distribution function

$$\text{pdf}(x, t) = \mathcal{N}(x, t | \mu, \Sigma)$$

Where:

$$\mu = \begin{bmatrix} \mu_x \\ \mu_t \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \sigma_x^2 & \rho\sigma_x\sigma_t \\ \rho\sigma_x\sigma_t & \sigma_t^2 \end{bmatrix}$$

and ρ is the correlation coefficient

Conditional probability distribution function still Normal:

$\text{pdf}(t|x, \mu, \Sigma) = \mathcal{N}(t, \mu_{t|x}, \sigma_{t|x}^2)$, with:

$$\mu_{t|x} = \mu_t + \rho \frac{\sigma_t}{\sigma_x} (x - \mu_x) = w_0 + w_1 x$$

$$\sigma_{t|x}^2 = (1 - \rho^2)\sigma_t^2 \quad (\text{constant wrt. } x)$$

Explicit Regression Model

Model (deterministic):

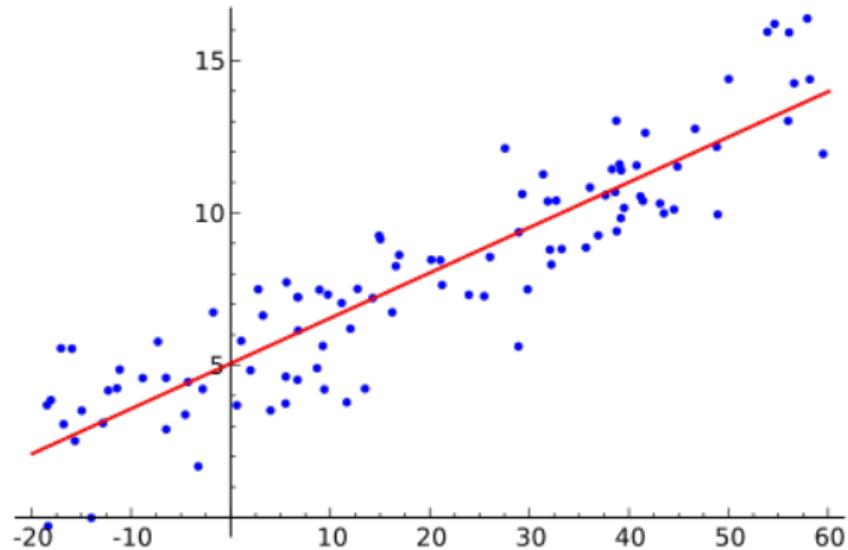
$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon$$

But now:

$$\epsilon \sim \mathcal{N}(0, \sigma^2)$$

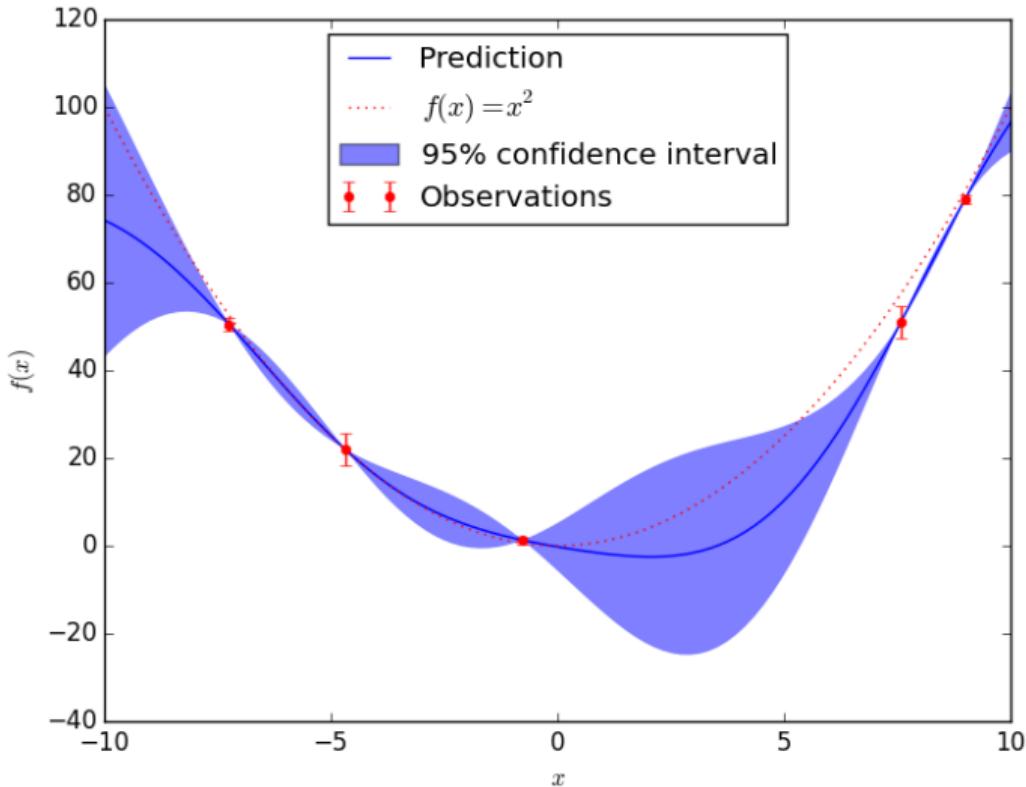
Therefore:

$$\begin{aligned} t &\sim \mathcal{N}(\mu_T(\mathbf{x}), \sigma_T^2(\mathbf{x})) \\ &= \mathcal{N}(y(\mathbf{x}, \mathbf{w}), \sigma^2) \end{aligned}$$



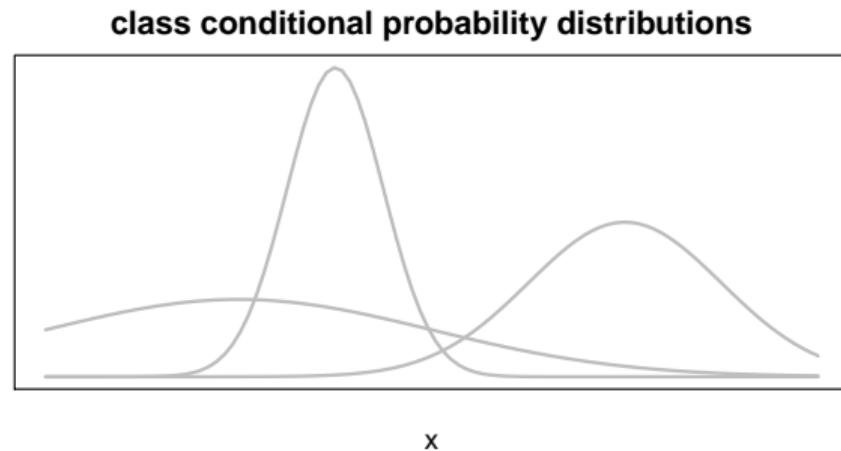
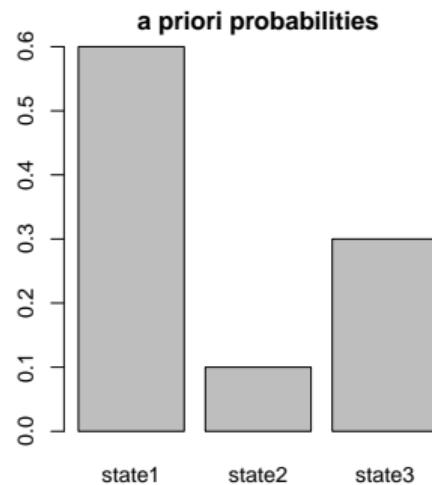
Gaussian Processes (advanced topic)

- non-parametric model
- covariance between t and x depends on observed data \mathcal{D}



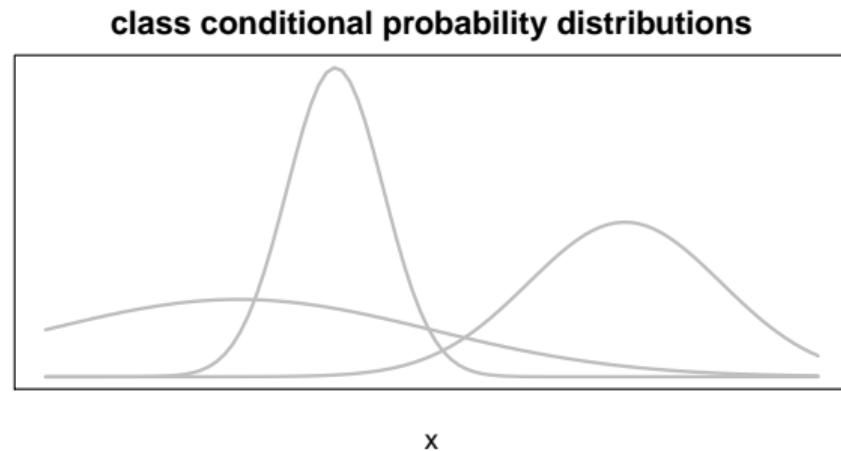
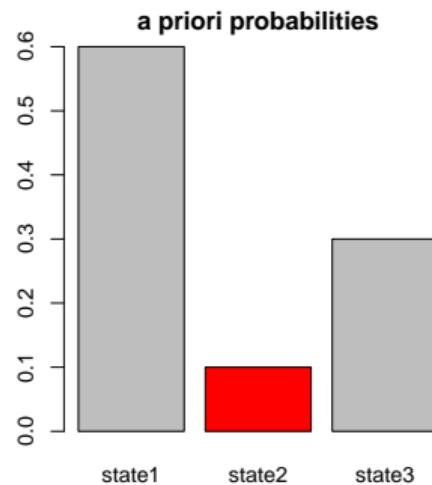
The Probabilistic Model of Classification

- one of k states t_j is selected with *a priori* probability $P(t_j)$
- When in state t_j , some observations $\hat{\mathbf{x}}$ are generated with distribution $p(\mathbf{x}|t_j)$



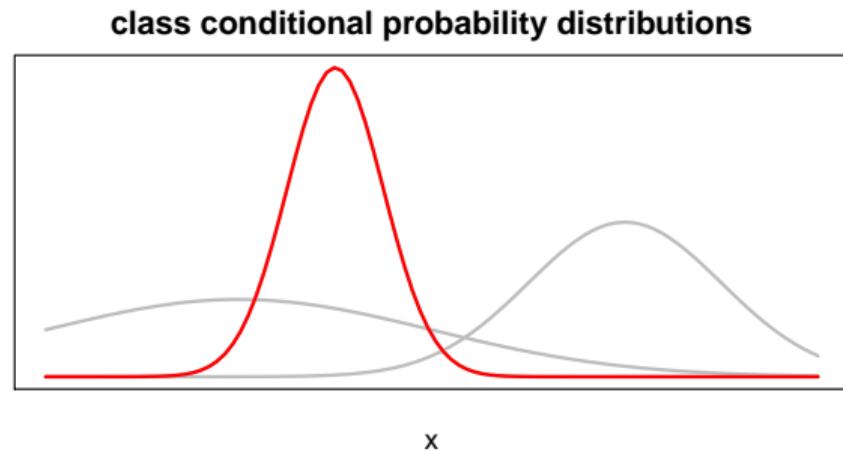
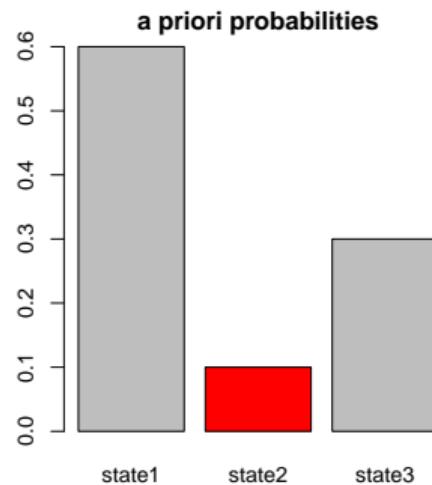
The Probabilistic Model of Classification

- one of k states t_j is selected with *a priori* probability $P(t_j)$
- When in state t_j , some observations $\hat{\mathbf{x}}$ are generated with distribution $p(\mathbf{x}|t_j)$



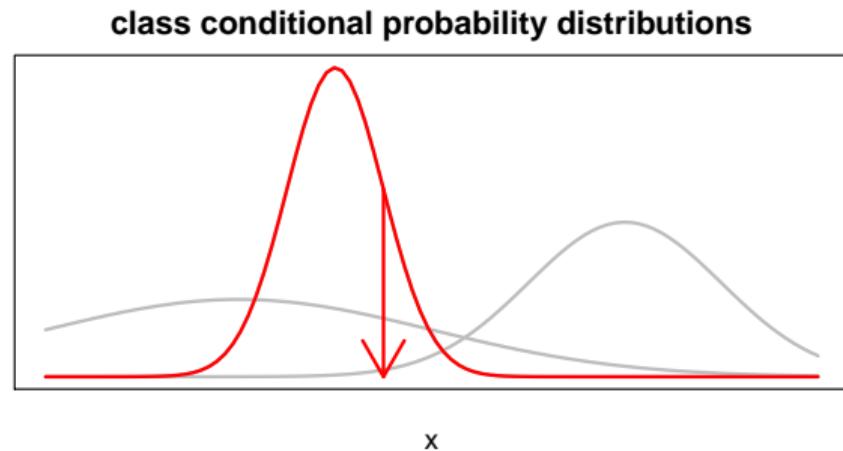
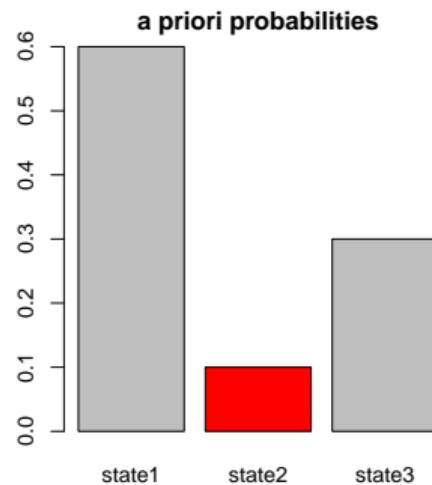
The Probabilistic Model of Classification

- one of k states t_j is selected with *a priori* probability $P(t_j)$
- When in state t_j , some observations $\hat{\mathbf{x}}$ are generated with distribution $p(\mathbf{x}|t_j)$



The Probabilistic Model of Classification

- one of k states t_j is selected with *a priori* probability $P(t_j)$
- When in state t_j , some observations $\hat{\mathbf{x}}$ are generated with distribution $p(\mathbf{x}|t_j)$



Problem

- If I observe a new $\hat{\mathbf{x}}$
- and I know $P(t_j)$ and $p(\mathbf{x}|t_j)$ for each class t_j
- what can I say about the state of the problem (class) t_j ?
- equivalent to divideing feature space in K regions $\{\mathcal{R}_1, \dots, \mathcal{R}_K\}$

Minimizing probability of errors

Example two classes:

$$\begin{aligned} P(\text{error}) &= P(\mathbf{x} \in \mathcal{R}_1, c_2) + P(\mathbf{x} \in \mathcal{R}_2, c_1) \\ &= \int_{\mathcal{R}_1} p(\mathbf{x}, c_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, c_1) d\mathbf{x} \end{aligned}$$

Minimizing probability of errors

Example two classes:

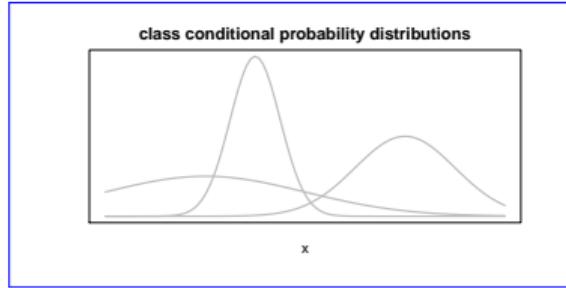
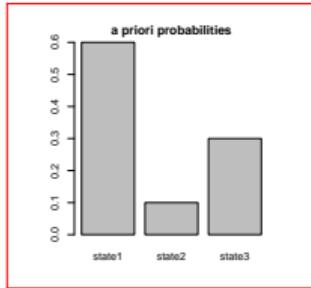
$$\begin{aligned} P(\text{error}) &= P(\mathbf{x} \in \mathcal{R}_1, c_2) + P(\mathbf{x} \in \mathcal{R}_2, c_1) \\ &= \int_{\mathcal{R}_1} p(\mathbf{x}, c_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, c_1) d\mathbf{x} \end{aligned}$$

Example K classes (maximize correct):

$$P(\text{correct}) = \sum_{k=1}^K P(\mathbf{x} \in \mathcal{R}_k, c_k) = \sum_{k=1}^K \int_{\mathcal{R}_k} p(\mathbf{x}, c_k) d\mathbf{x}$$

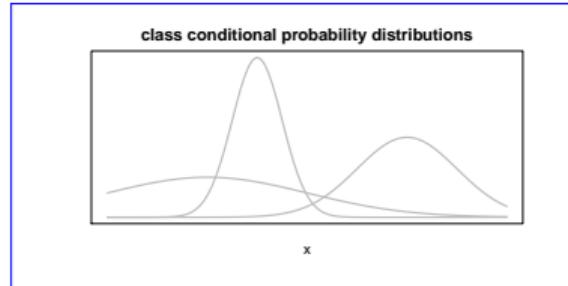
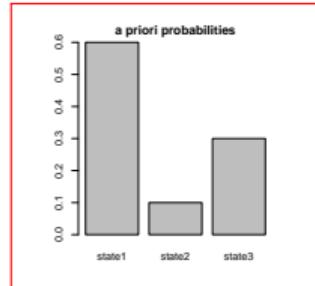
In both cases equivalent to Maximum a posteriori (MAP)

Bayes decision theory



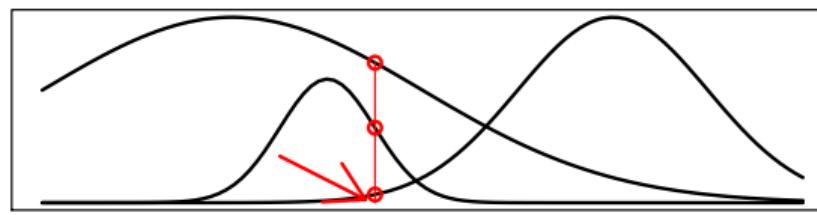
$$P(t_j | \hat{\mathbf{x}}) = \frac{p(\hat{\mathbf{x}} | t_j) P(t_j)}{p(\hat{\mathbf{x}})}$$

Bayes decision theory

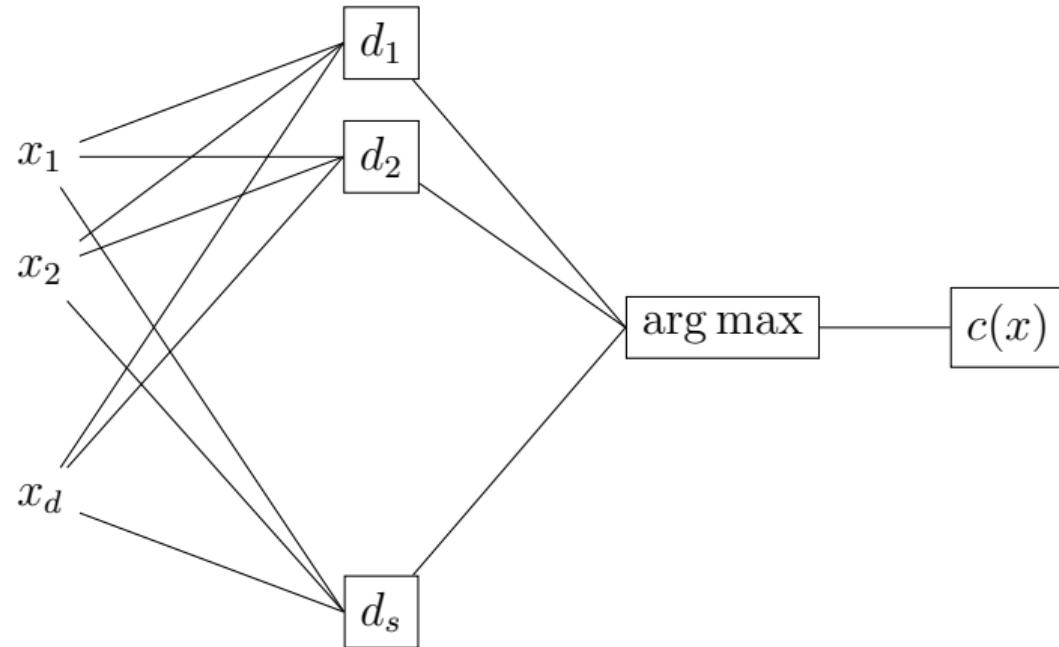


$$P(t_j|\hat{\mathbf{x}}) = \frac{p(\hat{\mathbf{x}}|t_j) P(t_j)}{p(\hat{\mathbf{x}})}$$

posterior probabilities



Classifiers: Discriminant Functions



$$d_i(\mathbf{x}) = p(\mathbf{x}|t_i) P(t_i)$$

Loss Function (classification)

L_{kj} , $k = \text{true class}, j = \text{classification}$

- different consequences for different kinds of errors:

Minimize expected loss

$$\mathbb{E}[L] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj} p(x, c_k) dx$$

Loss Function (regression)

$L(t, y(\mathbf{x}))$, $t = \text{true value}$, $y(\mathbf{x}) = \text{predicted value}$

Expected loss:

$$\mathbb{E}[L] = \int \int L(t, y(\mathbf{x})) p(\mathbf{x}, t) d\mathbf{x} dt$$

Examples

$$L(t, y(\mathbf{x})) = (y(\mathbf{x}) - t)^2 \quad \text{square loss}$$

$$L(t, y(\mathbf{x})) = |y(\mathbf{x}) - t|^q \quad \text{Minkowski loss}$$

$q = 2 \rightarrow \text{conditional mean}$, $q = 1 \rightarrow \text{conditional median}$, $q = 0 \rightarrow \text{conditional mode}$

Example: Which Gender?

Task: Determine the gender of a person given their measured hair length.

Example: Which Gender?

Task: Determine the gender of a person given their measured hair length.

Notation:

- Let $g \in \{'f', 'm'\}$ be a r.v. denoting the gender of a person.
- Let x be the measured length of the hair.

Example: Which Gender?

Task: Determine the gender of a person given their measured hair length.

Notation:

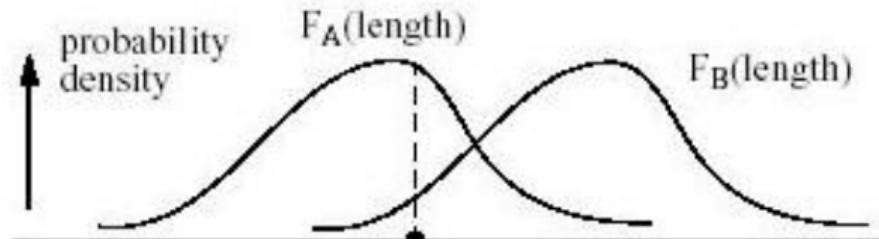
- Let $g \in \{'f', 'm'\}$ be a r.v. denoting the gender of a person.
- Let x be the measured length of the hair.

Information given:

- The hair length observation was made at a boy's school thus

$$P(g = 'm') = .95, \quad P(g = 'f') = .05$$

- Knowledge of the likelihood distributions $P(x | g = 'f')$ and $P(x | g = 'm')$



Example: Which Gender?

Task: Determine the gender of a person given their measured hair length \implies calculate $P(g | x)$.

Solution:

Apply Bayes' Rule to get

$$\begin{aligned} P(g = 'm' | x) &= \frac{P(x | g = 'm')P(g = 'm')}{P(x)} \\ &= \frac{P(x | g = 'm')P(g = 'm')}{P(x | g = 'f')P(g = 'f') + P(x | g = 'm')P(g = 'm')} \end{aligned}$$

Can calculate $P(g = 'f' | x) = 1 - P(g = 'm' | x)$

Selecting the most probable hypothesis

- **Maximum A Posteriori (MAP) Estimate:**

Hypothesis with highest probability given observed data

$$\begin{aligned} t_{\text{MAP}} &= \arg \max_{t \in \mathcal{T}} P(t | \mathbf{x}) \\ &= \arg \max_{t \in \mathcal{T}} \frac{P(\mathbf{x} | t) P(t)}{P(\mathbf{x})} \\ &= \arg \max_{t \in \mathcal{T}} P(\mathbf{x} | t) P(t) \end{aligned}$$

Selecting the most probable hypothesis

- **Maximum A Posteriori (MAP) Estimate:**

Hypothesis with highest probability given observed data

$$\begin{aligned} t_{\text{MAP}} &= \arg \max_{t \in \mathcal{T}} P(t | \mathbf{x}) \\ &= \arg \max_{t \in \mathcal{T}} \frac{P(\mathbf{x} | t) P(t)}{P(\mathbf{x})} \\ &= \arg \max_{t \in \mathcal{T}} P(\mathbf{x} | t) P(t) \end{aligned}$$

- **Maximum Likelihood Estimate (MLE):**

Hypothesis with highest likelihood of generating observed data.

$$t_{\text{MLE}} = \arg \max_{t \in \mathcal{T}} P(\mathbf{x} | t)$$

Useful if we do not know prior distribution or if it is uniform.

Example: Cancer or Not?

Scenario:

A patient takes a lab test and the result comes back positive. The test returns a correct positive result in only 98% of the cases in which the disease is actually present, and a correct negative result in only 97% of the cases in which the disease is not present. Furthermore, 0.8% of the entire population have cancer.

Example: Cancer or Not?

Scenario:

A patient takes a lab test and the result comes back positive. The test returns a correct positive result in only 98% of the cases in which the disease is actually present, and a correct negative result in only 97% of the cases in which the disease is not present. Furthermore, 0.8% of the entire population have cancer.

Scenario in probabilities:

- **Priors:**

$$P(\text{disease}) = .008 \quad P(\text{not disease}) = .992$$

- **Likelihoods:**

$$P(+ | \text{disease}) = .98$$

$$P(- | \text{disease}) = .02$$

$$P(+ | \text{not disease}) = .03$$

$$P(- | \text{not disease}) = .97$$

Example: Cancer or Not?

Find MAP estimate:

When test returned a positive result,

$$t_{\text{MAP}} = \arg \max_{t \in \{\text{disease, not disease}\}} P(t | +) = \arg \max_{t \in \{\text{disease, not disease}\}} P(+ | t) P(t)$$

Example: Cancer or Not?

Find MAP estimate:

When test returned a positive result,

$$t_{\text{MAP}} = \arg \max_{t \in \{\text{disease, not disease}\}} P(t | +) = \arg \max_{t \in \{\text{disease, not disease}\}} P(+ | t) P(t)$$

Substituting in the correct values get

$$P(+ | \text{disease}) P(\text{disease}) = .98 \times .008 = .0078$$

$$P(+ | \text{not disease}) P(\text{not disease}) = .03 \times .992 = .0298$$

Therefore $y_{\text{MAP}} = \text{not disease}$.

Example: Cancer or Not?

Find MAP estimate:

When test returned a positive result,

$$t_{\text{MAP}} = \arg \max_{t \in \{\text{disease, not disease}\}} P(t | +) = \arg \max_{t \in \{\text{disease, not disease}\}} P(+ | t) P(t)$$

Substituting in the correct values get

$$P(+ | \text{disease}) P(\text{disease}) = .98 \times .008 = .0078$$

$$P(+ | \text{not disease}) P(\text{not disease}) = .03 \times .992 = .0298$$

Therefore $y_{\text{MAP}} = \text{not disease}$.

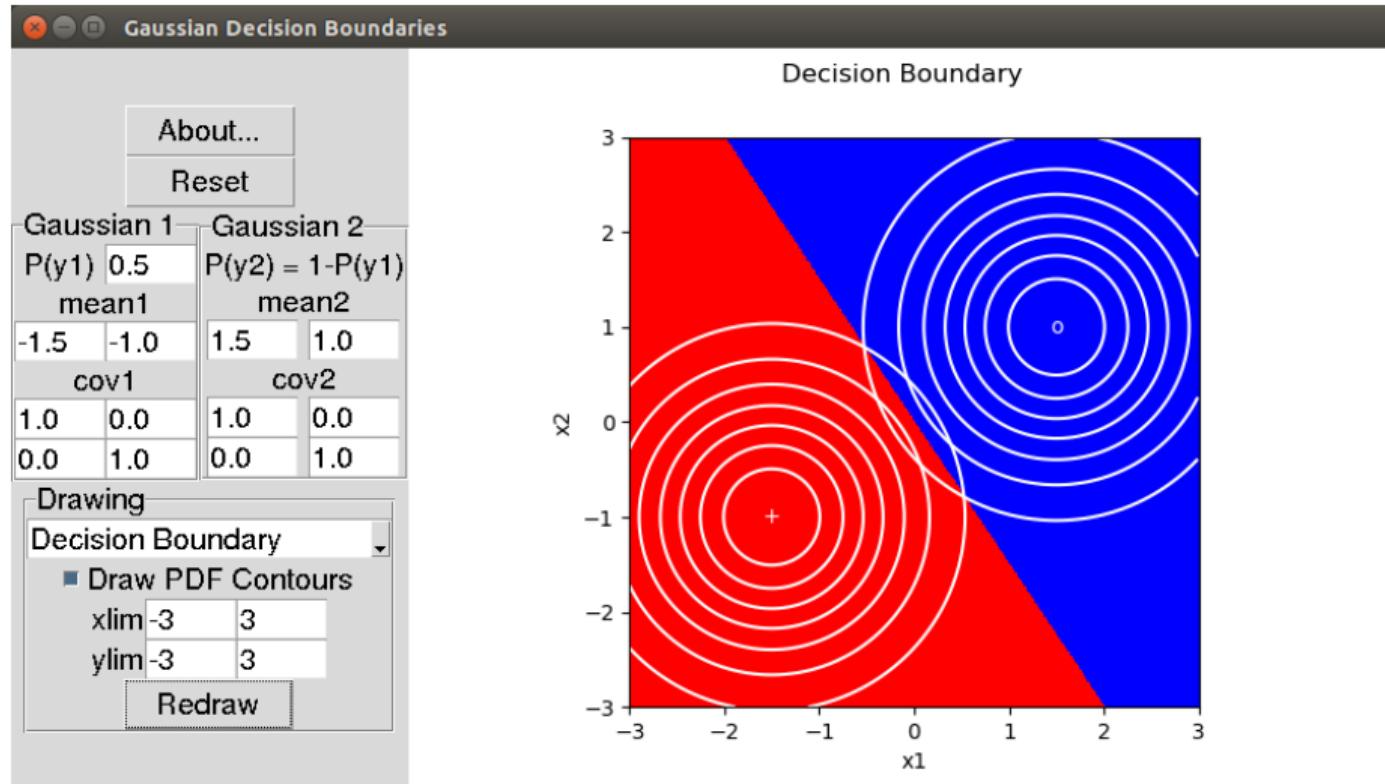
The Posterior probabilities:

$$P(\text{disease} | +) = \frac{.0078}{(.0078 + .0298)} = .21$$

$$P(\text{not disease} | +) = \frac{.0298}{(.0078 + .0298)} = .79$$

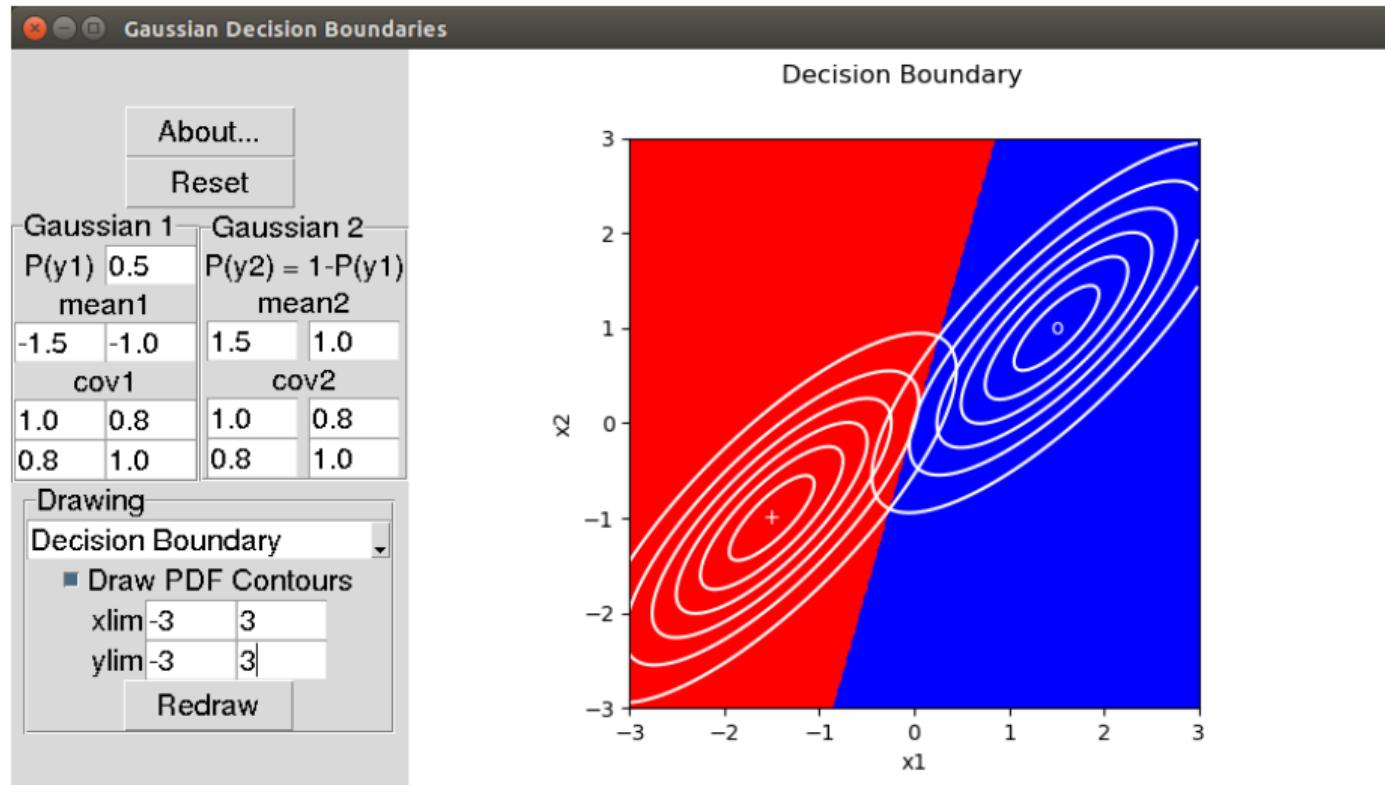
Demo: Gaussian Decision Boundaries

<https://github.com/giampierosalvi/GaussianDecisionBoundaries>



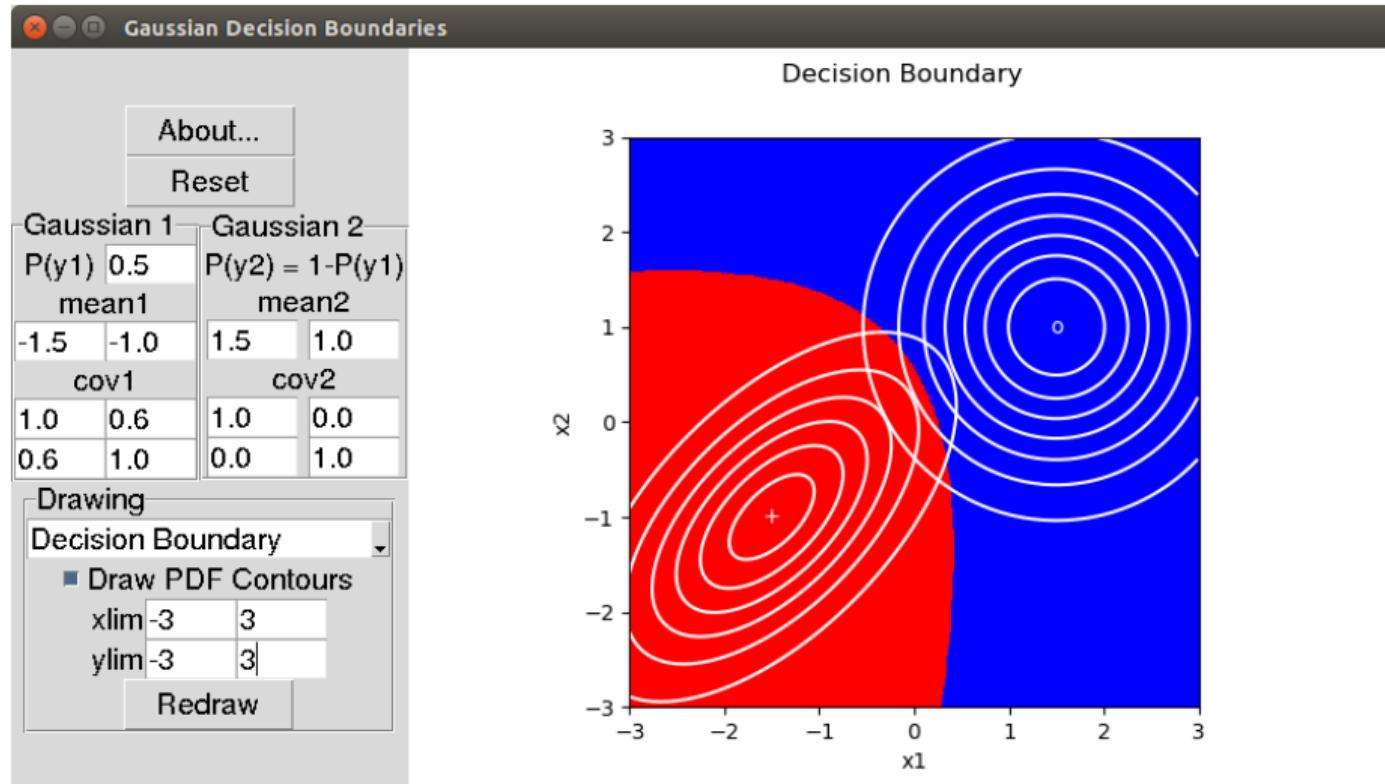
Demo: Gaussian Decision Boundaries

<https://github.com/giampierosalvi/GaussianDecisionBoundaries>



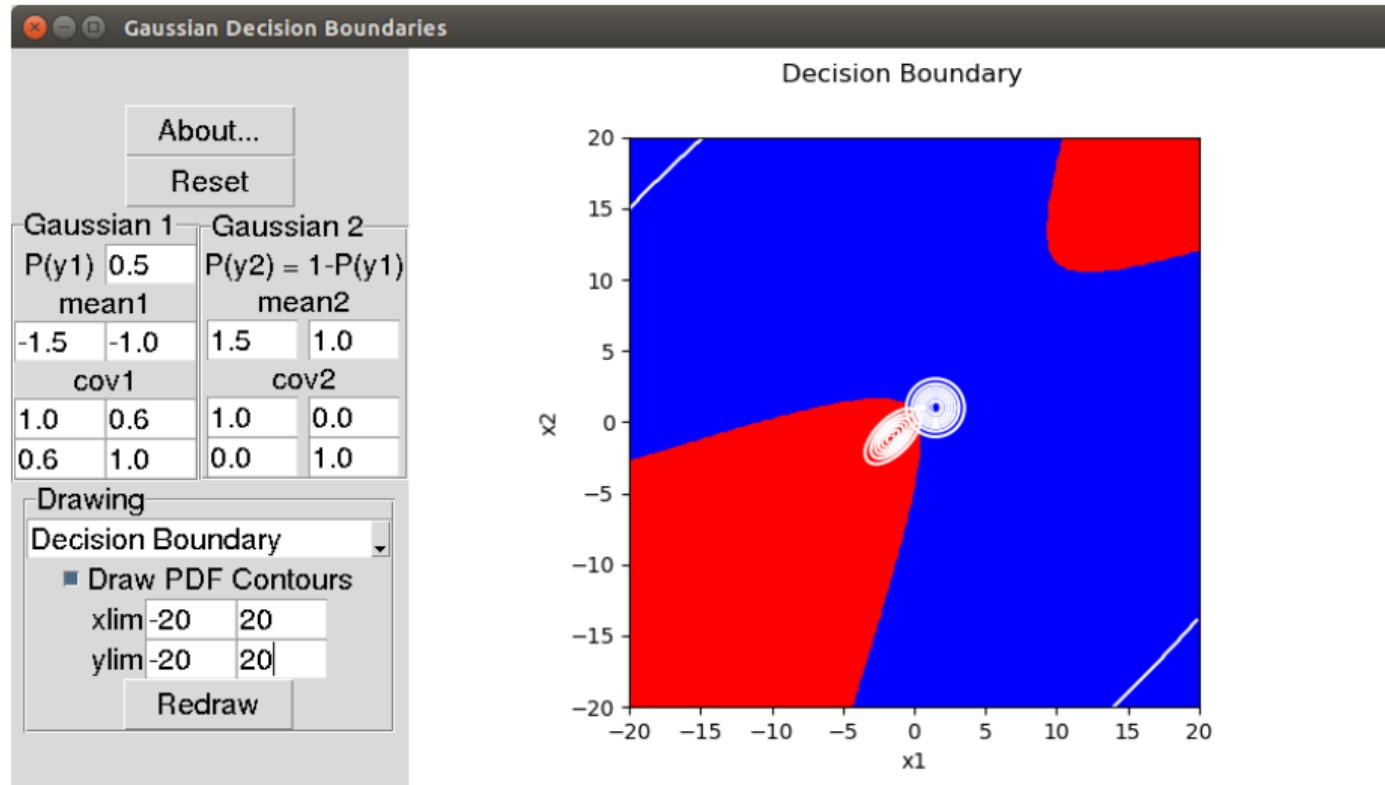
Demo: Gaussian Decision Boundaries

<https://github.com/giampierosalvi/GaussianDecisionBoundaries>



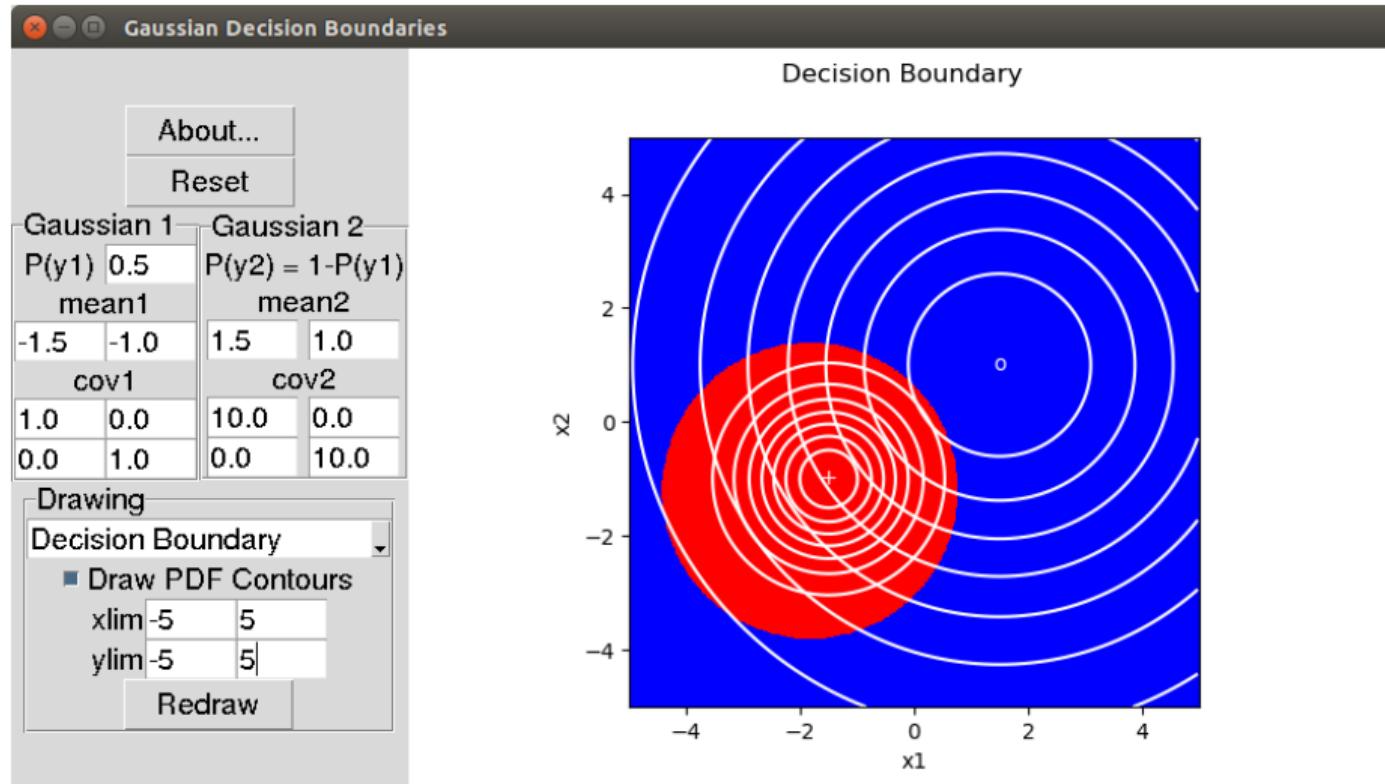
Demo: Gaussian Decision Boundaries

<https://github.com/giampierosalvi/GaussianDecisionBoundaries>



Demo: Gaussian Decision Boundaries

<https://github.com/giampierosalvi/GaussianDecisionBoundaries>



Inference vs Decision

- Inference: estimate posterior $p(c_k|\mathbf{x})$
- Decision: use $p(c_k|\mathbf{x})$ to make class assignments
- we could go directly from x to c_k with discriminant functions

Generative vs Discriminative

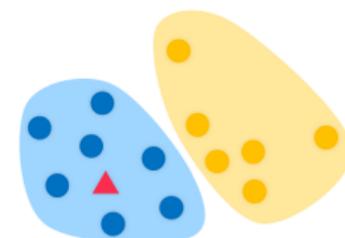
Generative:

- estimate $p(x|c_k)$ and $p(c_k)$
- compute $p(c_k|x) = \frac{p(x|c_k)p(c_k)}{\sum_j p(x|c_j)p(c_j)}$
- use decision theory

Discriminative:

- estimate posterior $p(c_k|x)$ directly
- use decision theory
- we could also use discriminant functions $f_k(x)$ with no probabilistic interpretation

Generative



Discriminative

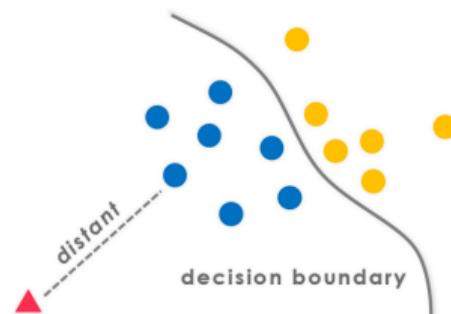


Figure from Nguyen *et al.* 2015. <http://www.evolvingai.org/fooling>

Outline

1 Probability Theory Reminder

- Axioms and Properties
- Common Distributions
- Moments

2 Probabilistic Machine Learning

- Supervised Learning, General Definition
- Regression
- Classification
- Bayes decision theory

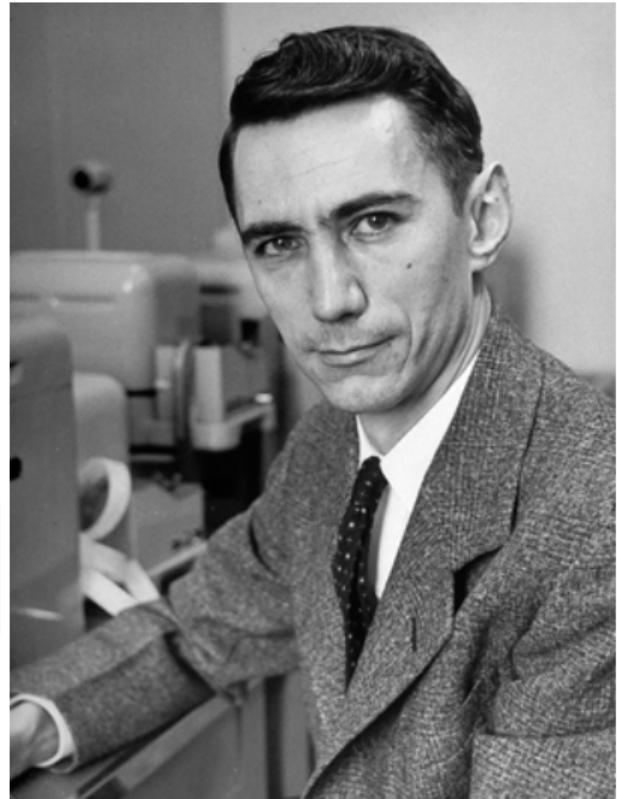
3 Information Theory

Information Theory

- Pioneered by Shannon in the 1940s
- probabilities describe uncertainty of events
- we are interested in the **information** gained observing the outcome of an event

Information

$$h(x) = -\log_2 p(x)$$



Properties of Information

$$h(x) = -\log_2 p(x)$$

- ① If x and y are independent variables:

$$\begin{aligned} p(x, y) &= p(x)p(y) \\ h(x, y) &= h(x) + h(y) \end{aligned}$$

- ② The sure event Ω associated with zero information:

$$p(\Omega) = 1 \Rightarrow h(\Omega) = 0$$

- ③ If $p(x) < p(y) \Rightarrow h(x) > h(y)$

Information and transmissions

Suppose we want to send the outcome of $x \in \{x_1, x_2, \dots\}$ (finite or countable) with distribution $p(x)$

Entropy

The expected number of bits of information is

$$\mathbb{E}[h(x)] = \sum_i p(x_i)h(x_i) = - \sum_i p(x_i) \log_2 p(x_i) = H(x)$$

Example

Suppose we want to send the outcome of $x \in \{a, b, c, d, e, f, g, h\}$

Uniform distribution: $p(x_i) = \frac{1}{8}$

Then $H(x) = - \sum_{i=1}^8 \frac{1}{8} \log_2 \frac{1}{8} = 3$ bits

Possible code:

$$a = 000$$

$$b = 001$$

...

$$h = 111$$

Example

Suppose we want to send the outcome of $x \in \{a, b, c, d, e, f, g, h\}$

Non uniform distribution: $p(x) = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}\}$

Then $H(x) = \frac{1}{2} \log_2 2 + \frac{1}{4} \log_2 2^2 + \frac{1}{8} \log_2 2^3 + \frac{1}{16} \log_2 2^4 + 4 \frac{1}{64} \log_2 2^6 = 2$ bits

Possible code:

$$a = 0$$

$$b = 10$$

$$c = 110$$

$$d = 1110$$

$$e = 111100$$

$$f = 111101$$

$$g = 111110$$

$$h = 111111$$

Noiseless coding theorem (Shannon 1948)

Theorem

The entropy is the lower bound to the number of bits required on average to transmit the state of a random variable.

Other interpretations:

- Thermodynamics: measure of equilibrium
- Statistical mechanics: measure of disorder

Units:

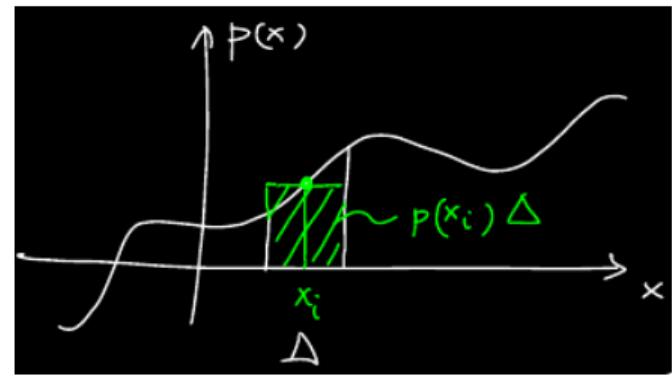
- $\log_2 \rightarrow$ bits
- $\ln \rightarrow$ nats (from natural log)

Entropy and continuous variables

If $x \in \mathbb{R}$ and $p(x) \in C^0$ is a continuous function in \mathbb{R}

$$\begin{aligned} H_\Delta &= - \sum_i p(x_i) \Delta \ln(p(x_i) \Delta) \\ &= - \sum_i p(x_i) \Delta \ln p(x_i) - \ln \Delta \xrightarrow{\Delta \rightarrow 0} \infty \end{aligned}$$

Mean value theorem



$$\int_{i\Delta}^{(i+1)\Delta} p(x) dx = p(x_i) \Delta$$

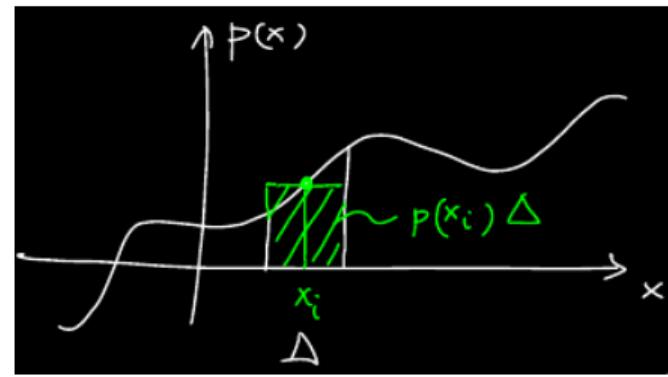
Entropy and continuous variables

If $x \in \mathbb{R}$ and $p(x) \in C^0$ is a continuous function in \mathbb{R}

$$\begin{aligned} H_\Delta &= - \sum_i p(x_i) \Delta \ln(p(x_i) \Delta) \\ &= - \sum_i p(x_i) \Delta \ln p(x_i) - \ln \Delta \xrightarrow{\Delta \rightarrow 0} \infty \end{aligned}$$

Infinite information to transmit a real number!

Mean value theorem



$$\int_{i\Delta}^{(i+1)\Delta} p(x) dx = p(x_i) \Delta$$

Differential Entropy

$$H_{\Delta} = - \sum_i p(x_i) \Delta \ln p(x_i) - \ln \Delta \xrightarrow{\Delta \rightarrow 0} \infty$$

- remove the offending term $\ln \Delta$
- take the limit for $\Delta \rightarrow 0$

Differential Entropy

$$H(x) = \lim_{\Delta \rightarrow 0} - \sum_i p(x_i) \Delta \ln p(x_i) = - \int p(x) \ln p(x) dx$$

Maximum Entropy

What are the distributions that give maximum (differential) entropy?

- Discrete case (k values): uniform distribution,

$$H = \ln k$$

- Continuous case: Gaussian distribution,

$$H = \frac{1}{2} [1 + \ln(2\pi\sigma^2)]$$

Conditional Entropy

Given two variables x and y :

$$\begin{aligned} H(y|x) &= - \int \int p(x,y) \ln p(y|x) dx dy \\ &= H(x,y) - H(x) \end{aligned}$$

Relative Entropy (Kullback-Leibler divergence)

Given two distributions $p(x)$ and $q(x)$

Cross entropy:

$$\mathbb{E}_{x \sim p}[-\ln q] = - \int p(x) \ln q(x) dx$$

Kullback-Leibler divergence:

$$\begin{aligned}\text{KL}(p||q) &= \mathbb{E}_{x \sim p}[-\ln q] - \mathbb{E}_{x \sim q}[-\ln p] \\ &= - \int \int p(x) \ln q(x) dx + \int \int q(x) \ln p(x) dx \\ &= - \int \int p(x) \ln \frac{q(x)}{p(x)} dx\end{aligned}$$

Mutual Information

Given two variables x and y , are they independent?

Mutual information:

$$\begin{aligned} I(x, y) &= \text{KL}(p(x, y) || p(x)p(y)) \\ &= - \int \int p(x, y) \ln \frac{p(x)p(y)}{p(x, y)} dx dy \end{aligned}$$

- $I(x, y) \geq 0$
- $I(x, y) = 0 \Leftrightarrow x \text{ and } y \text{ are independent}$

$$I(x, y) = H(x) - H(x|y) = H(y) - H(y|x)$$

Learning as Inference

TTT4185 Machine Learning for Signal Processing

Giampiero Salvi

Department of Electronic Systems
NTNU

HT2021

Outline

1 Learning as Inference

2 Point Estimates

- Maximum Likelihood Estimation
- Maximum a Posteriori Estimation

3 Bayesian Methods

4 Curse of Dimensionality

Outline

1 Learning as Inference

2 Point Estimates

- Maximum Likelihood Estimation
- Maximum a Posteriori Estimation

3 Bayesian Methods

4 Curse of Dimensionality

Probabilistic Classification and Regression

- In both cases estimate posterior

$$P(t \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid t)P(t)}{P(\mathbf{x})}$$

- Classification: t is discrete
- Regression: t is continuous

Probabilistic Classification and Regression

- In both cases estimate posterior

$$P(t \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid t)P(t)}{P(\mathbf{x})}$$

- Classification: t is discrete
- Regression: t is continuous

Until now we assumed we knew:

- $P(t) \leftarrow \text{Prior}$
- $P(\mathbf{x} \mid t) \leftarrow \text{Likelihood}$
- $P(\mathbf{x}) \leftarrow \text{Evidence}$

Probabilistic Classification and Regression

- In both cases estimate posterior

$$P(t \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid t)P(t)}{P(\mathbf{x})}$$

- Classification: t is discrete
- Regression: t is continuous

Until now we assumed we knew:

- $P(t) \leftarrow \text{Prior}$
- $P(\mathbf{x} \mid t) \leftarrow \text{Likelihood}$
- $P(\mathbf{x}) \leftarrow \text{Evidence}$

How can we obtain this information from observations (data)?

Learning as Inference

Given:

- the training data $\mathcal{D} = \{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \dots, (\mathbf{x}_N, t_N)\}$
- a new observation \mathbf{x}

Estimate the posterior probability of the answer t :

$$P(t|\mathbf{x}, \mathcal{D})$$

Discriminative vs Generative Models

Discriminative:

- learn the posterior $P(t|\mathbf{x}, \mathcal{D})$ directly
- examples: linear regression, logistic regression

Generative:

- learn a model of data generation: priors $P(t|\mathcal{D})$ and likelihoods $P(\mathbf{x}|t, \mathcal{D})$
- use Bayes rule to obtain posterior $P(t|\mathbf{x}, \mathcal{D})$
- example: classification

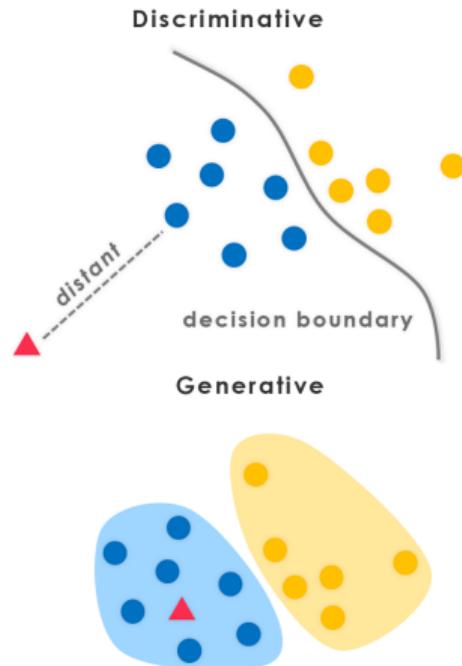


Figure from Nguyen et al.

Parametric vs Non-parametric Inference

Parametric:

- First make the model parameters explicit: $P(t|\mathbf{x}) = P(t|\mathbf{x}, \theta)$
- estimate the optimal parameters $\hat{\theta}$ using the data (point estimate)
- compute the posterior $P(t|\mathbf{x}, \hat{\theta})$

Learning corresponds to finding $\hat{\theta}$

Parametric vs Non-parametric Inference

Parametric:

- First make the model parameters explicit: $P(t|\mathbf{x}) = P(t|\mathbf{x}, \theta)$
- estimate the optimal parameters $\hat{\theta}$ using the data (point estimate)
- compute the posterior $P(t|\mathbf{x}, \hat{\theta})$

Learning corresponds to finding $\hat{\theta}$

Non-Parametric:

- Use a parametric model as before: $P(t|\mathbf{x}) = P(t|\mathbf{x}, \theta)$
- but estimate the posterior of the parameters given the data: $P(\theta|\mathcal{D})$
- Compute the posterior $P(t|\mathbf{x}, \mathcal{D})$ by marginalizing out the parameters θ

The number of parameters can grow with the data!

Three Approaches

Parametric:

- Maximum Likelihood (ML)
- Maximum A Posteriori (MAP)

Non-parametric:

- Bayesian methods

Fundamental Assumption: i.i.d.

Observations are **independent and identically distributed**:

$$\mathcal{D} = \{\mathbf{o}_1, \dots, \mathbf{o}_N\}$$

The likelihood of the whole data set can be factorized:

$$P(\mathcal{D}) = P(\mathbf{o}_1, \dots, \mathbf{o}_N) = \prod_{i=1}^N P(\mathbf{o}_i)$$

And the log-likelihood becomes:

$$\log P(\mathcal{D}) = \sum_{i=1}^N \log P(\mathbf{o}_i)$$

Outline

1 Learning as Inference

2 Point Estimates

- Maximum Likelihood Estimation
- Maximum a Posteriori Estimation

3 Bayesian Methods

4 Curse of Dimensionality

Maximum Likelihood Estimate

- define parametric form for the distributions:

$$P(\mathbf{x}|t) \equiv P(\mathbf{x}|t, \theta) \quad \text{or} \quad P(t|\mathbf{x}) \equiv P(t|\mathbf{x}, \theta)$$

- find optimal value for the parameter θ_{ML} by maximizing the likelihood of the data:

$$\theta_{\text{ML}} = \arg \max_{\theta} P(\mathcal{D}|\theta)$$

- approximate the distribution given the data with this distribution:

$$P(\mathbf{x}|t, \mathcal{D}) \approx P(\mathbf{x}|t, \theta_{\text{ML}}) \quad \text{or} \quad P(t|\mathbf{x}, \mathcal{D}) \approx P(t|\mathbf{x}, \theta_{\text{ML}})$$

Parameter Estimation vs Decision Theory

Decision theory:

- x and θ are known
- maximize likelihood or posterior to find t

Parameter Estimation:

- x and t are known (supervised learning)
- maximize likelihood or posterior to find θ

Parameter Estimation vs Decision Theory

Decision theory:

- x and θ are known
- maximize likelihood or posterior to find t

Parameter Estimation:

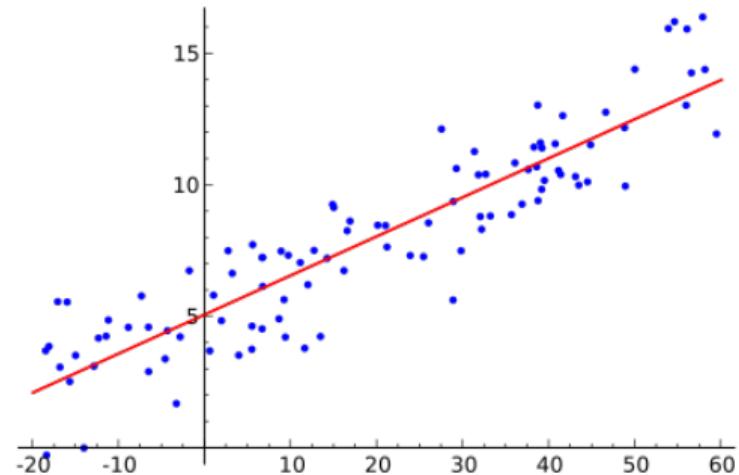
- x and t are known (supervised learning)
- maximize likelihood or posterior to find θ

Same models and same kind of optimization

Classical Linear Regression

Model (deterministic):

$$\begin{aligned}\hat{t} = y(\mathbf{x}, \mathbf{w}) &= w_0 + w_1 x_1 + \cdots + w_d x_d \\ &= [w_0 \quad w_1 \quad \dots \quad w_d] \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} \\ &= \mathbf{w}^T \mathbf{x}\end{aligned}$$



Minimize sum of square errors

$$\mathbf{w}_{\text{opt}} = \arg \min_{\mathbf{w}} \sum_{i=1}^N (t_i - y(\mathbf{x}_i, \mathbf{w}))^2 = \arg \min_{\mathbf{w}} \sum_{i=1}^N (t_i - \mathbf{w}^T \mathbf{x}_i)^2$$

Probabilistic Linear Regression

Model (deterministic):

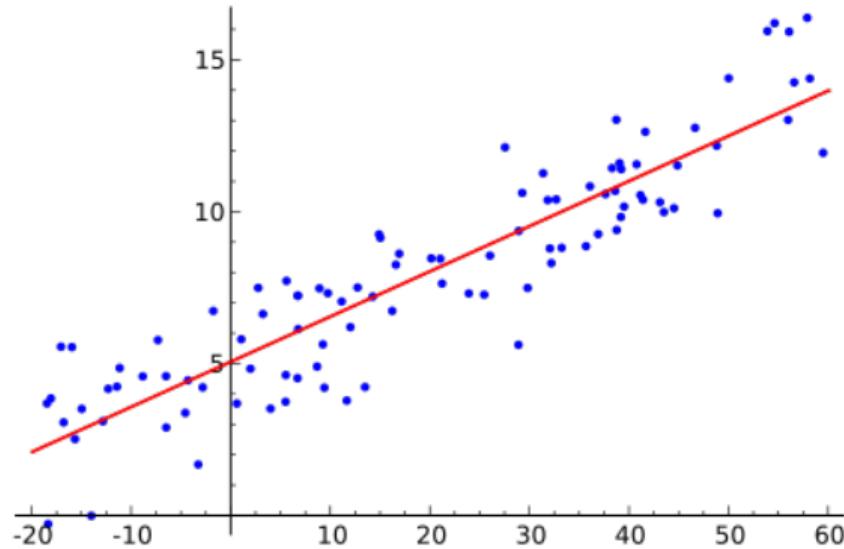
$$\hat{t} = y(\mathbf{x}, \mathbf{w}) + \epsilon = \mathbf{w}^T \mathbf{x} + \epsilon$$

But now:

$$\epsilon \sim \mathcal{N}(0, \sigma^2)$$

Therefore:

$$\begin{aligned} t &\sim \mathcal{N}(\mu_T(\mathbf{x}), \sigma_T^2(\mathbf{x})) \\ &= \mathcal{N}(\mathbf{w}^T \mathbf{x}, \sigma^2) \end{aligned}$$



Learning: find \mathbf{w} that maximizes $P(T|X, \mathbf{w}, \sigma^2)$

Maximize the posterior directly \implies discriminative method

MLE for Probabilistic Linear Regression

$$\log p(T|X, \mathbf{w}, \sigma^2) = \log \prod_i p(t_i|\mathbf{x}_i, \mathbf{w}, \sigma^2)$$

MLE for Probabilistic Linear Regression

$$\begin{aligned}\log p(T|X, \mathbf{w}, \sigma^2) &= \log \prod_i p(t_i|\mathbf{x}_i, \mathbf{w}, \sigma^2) \\ &= \sum_i \log p(t_i|\mathbf{x}_i, \mathbf{w}, \sigma^2)\end{aligned}$$

MLE for Probabilistic Linear Regression

$$\begin{aligned}\log p(T|X, \mathbf{w}, \sigma^2) &= \log \prod_i p(t_i|\mathbf{x}_i, \mathbf{w}, \sigma^2) \\ &= \sum_i \log p(t_i|\mathbf{x}_i, \mathbf{w}, \sigma^2) \\ &= \sum_i \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(t_i - \mathbf{w}^T \mathbf{x}_i)^2}{2\sigma^2}} \right]\end{aligned}$$

MLE for Probabilistic Linear Regression

$$\begin{aligned}\log p(T|X, \mathbf{w}, \sigma^2) &= \log \prod_i p(t_i|\mathbf{x}_i, \mathbf{w}, \sigma^2) \\ &= \sum_i \log p(t_i|\mathbf{x}_i, \mathbf{w}, \sigma^2) \\ &= \sum_i \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(t_i - \mathbf{w}^T \mathbf{x}_i)^2}{2\sigma^2}} \right] \\ &= \sum_i \left[-\frac{1}{2} \log(2\pi\sigma^2) - \frac{(t_i - \mathbf{w}^T \mathbf{x}_i)^2}{2\sigma^2} \right]\end{aligned}$$

MLE for Probabilistic Linear Regression

$$\begin{aligned}\log p(T|X, \mathbf{w}, \sigma^2) &= \log \prod_i p(t_i|\mathbf{x}_i, \mathbf{w}, \sigma^2) \\&= \sum_i \log p(t_i|\mathbf{x}_i, \mathbf{w}, \sigma^2) \\&= \sum_i \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(t_i - \mathbf{w}^T \mathbf{x}_i)^2}{2\sigma^2}} \right] \\&= \sum_i \left[-\frac{1}{2} \log(2\pi\sigma^2) - \frac{(t_i - \mathbf{w}^T \mathbf{x}_i)^2}{2\sigma^2} \right]\end{aligned}$$

$$\arg \max_{\mathbf{w}} [p(T|X, \mathbf{w}, \sigma^2)] = \arg \min_{\mathbf{w}} \sum_i (t_i - \mathbf{w}^T \mathbf{x}_i)^2$$

Maximizing $p(T|X, \mathbf{w}, \sigma^2)$ equivalent to minimizing sum of squares!

Source of confusion

We did Maximum a Posteriori (MAP) regression

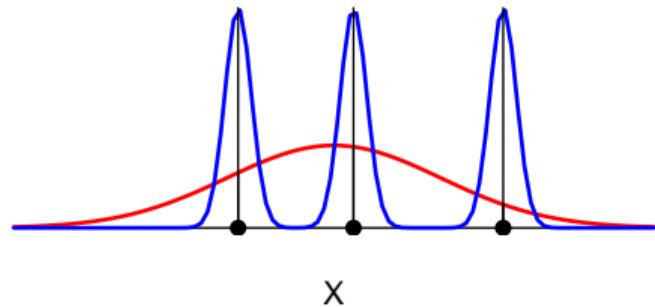
$$t_{\text{MAP}} = \arg \max_t p(t|\mathbf{x}, \theta_{\text{ML}})$$

with parameters θ estimated by Maximum Likelihood (ML):

$$\theta_{\text{ML}} = \arg \max_{\theta} p(D|\theta) = \arg \max_{\theta} \prod_i p(\mathbf{x}_i|t_i, \theta)$$

ML and overfitting

- same solution as sum of squares
- \Rightarrow same problems with overfitting
- we would like regularization



Maximum a posteriori

- assume that parameter θ is stochastic variable
- define a **prior** distribution over θ
- maximize posterior $P(\theta|\mathcal{D})$ over the parameter

Maximum a Posteriori Estimation

$$\theta_{\text{MAP}} = \arg \max_{\theta} p(\theta | \mathcal{D})$$

Maximum a Posteriori Estimation

$$\begin{aligned}\theta_{\text{MAP}} &= \arg \max_{\theta} p(\theta | \mathcal{D}) \\ &= \arg \max_{\theta} \frac{p(\theta) p(\mathcal{D} | \theta)}{p(\mathcal{D})}\end{aligned}$$

Maximum a Posteriori Estimation

$$\begin{aligned}\theta_{\text{MAP}} &= \arg \max_{\theta} p(\theta | \mathcal{D}) \\ &= \arg \max_{\theta} \frac{p(\theta) p(\mathcal{D} | \theta)}{p(\mathcal{D})} \\ &= \arg \max_{\theta} p(\theta) p(\mathcal{D} | \theta)\end{aligned}$$

Maximum a Posteriori Estimation

$$\begin{aligned}\theta_{\text{MAP}} &= \arg \max_{\theta} p(\theta | \mathcal{D}) \\&= \arg \max_{\theta} \frac{p(\theta) p(\mathcal{D} | \theta)}{p(\mathcal{D})} \\&= \arg \max_{\theta} p(\theta) p(\mathcal{D} | \theta) \\&= \arg \max_{\theta} \left[p(\theta) \prod_{i=1}^N p(\mathbf{o}_i | \theta) \right]\end{aligned}$$

Maximum a Posteriori Estimation

$$\begin{aligned}\theta_{\text{MAP}} &= \arg \max_{\theta} p(\theta | \mathcal{D}) \\&= \arg \max_{\theta} \frac{p(\theta) p(\mathcal{D} | \theta)}{p(\mathcal{D})} \\&= \arg \max_{\theta} p(\theta) p(\mathcal{D} | \theta) \\&= \arg \max_{\theta} \left[p(\theta) \prod_{i=1}^N p(\mathbf{o}_i | \theta) \right] \\&= \arg \max_{\theta} \left[\log p(\theta) + \sum_{i=1}^N \log p(\mathbf{o}_i | \theta) \right]\end{aligned}$$

Maximum a Posteriori Estimation

$$\begin{aligned}\theta_{\text{MAP}} &= \arg \max_{\theta} p(\theta | \mathcal{D}) \\&= \arg \max_{\theta} \frac{p(\theta) p(\mathcal{D} | \theta)}{p(\mathcal{D})} \\&= \arg \max_{\theta} p(\theta) p(\mathcal{D} | \theta) \\&= \arg \max_{\theta} \left[p(\theta) \prod_{i=1}^N p(\mathbf{o}_i | \theta) \right] \\&= \arg \max_{\theta} \left[\log p(\theta) + \sum_{i=1}^N \log p(\mathbf{o}_i | \theta) \right]\end{aligned}$$

- $\log p(\theta)$ works as regularization

MAP for Linear Regression

Model (deterministic):

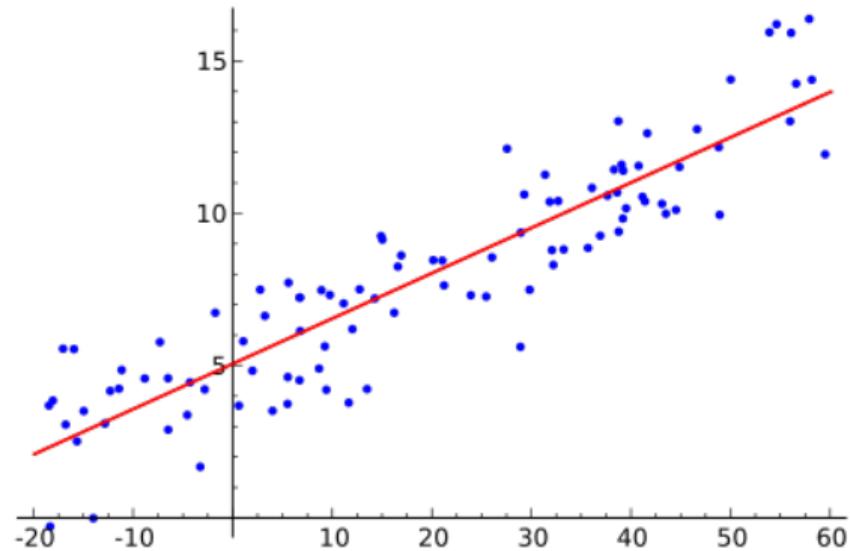
$$t = \mathbf{w}^T \mathbf{x} + \epsilon$$

With:

$$\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$$

Therefore:

$$t \sim \mathcal{N}(\mathbf{w}^T \mathbf{x}, \sigma_\epsilon^2)$$



MAP for Linear Regression

Model (deterministic):

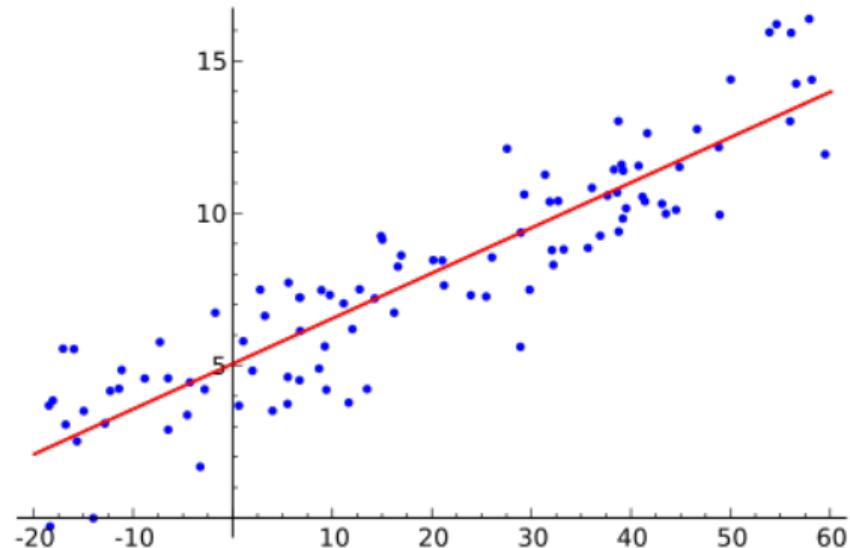
$$t = \mathbf{w}^T \mathbf{x} + \epsilon$$

With:

$$\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$$

Therefore:

$$t \sim \mathcal{N}(\mathbf{w}^T \mathbf{x}, \sigma_\epsilon^2)$$



But now we define the a priori probability over \mathbf{w} : $p(\mathbf{w})$

Example: zero-mean spherical Gaussian prior

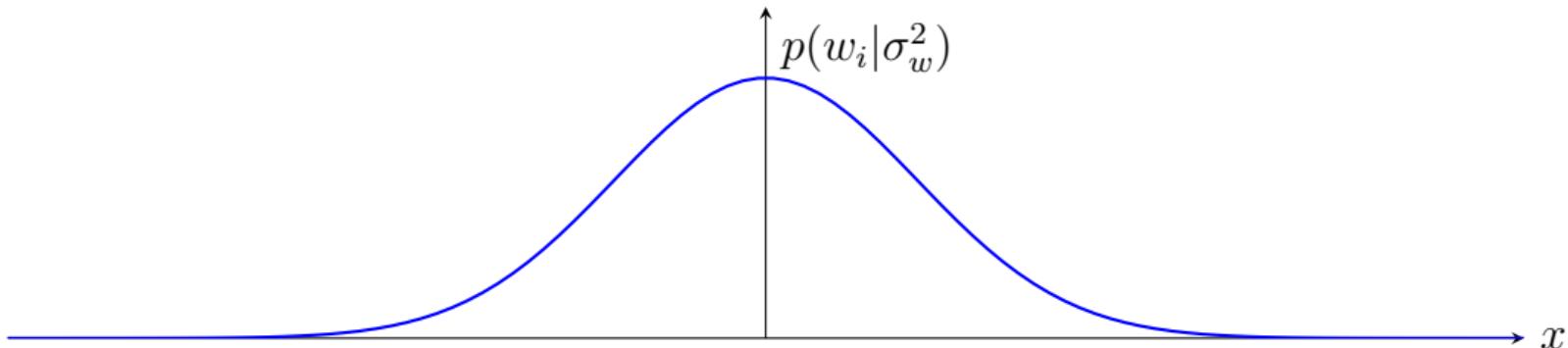
Example: zero-mean spherical Gaussian on $\mathbf{w} = [w_0, \dots, w_{d-1}]$

$$p(\mathbf{w}|\sigma_w^2) = \mathcal{N}(0, \sigma_w^2 \mathbf{I}) = \frac{1}{(2\pi\sigma_w^2)^{\frac{d}{2}}} \exp\left(-\frac{\mathbf{w}^T \mathbf{w}}{2\sigma_w^2}\right)$$

Example: zero-mean spherical Gaussian prior

Example: zero-mean spherical Gaussian on $\mathbf{w} = [w_0, \dots, w_{d-1}]$

$$\begin{aligned} p(\mathbf{w} | \sigma_w^2) &= \mathcal{N}(0, \sigma_w^2 \mathbf{I}) = \frac{1}{(2\pi\sigma_w^2)^{\frac{d}{2}}} \exp\left(-\frac{\mathbf{w}^T \mathbf{w}}{2\sigma_w^2}\right) = \\ &= \prod_{i=1}^d \frac{1}{\sqrt{2\pi\sigma_w^2}} \exp\left(-\frac{w_i^2}{2\sigma_w^2}\right) \end{aligned}$$



MAP estimate with zero-mean spherical Gaussian prior

Instead of $\log p(T|X, \mathbf{w})$ as in MLE, we optimize $\log p(\mathbf{w}|T, X)$:

$$\mathbf{w}_{\text{MAP}} = \arg \max_{\mathbf{w}} \log p(\mathbf{w}|T, X) = \arg \max_{\mathbf{w}} \log [p(T|X, \mathbf{w})p(\mathbf{w})]$$

MAP estimate with zero-mean spherical Gaussian prior

Instead of $\log p(T|X, \mathbf{w})$ as in MLE, we optimize $\log p(\mathbf{w}|T, X)$:

$$\begin{aligned}\mathbf{w}_{\text{MAP}} &= \arg \max_{\mathbf{w}} \log p(\mathbf{w}|T, X) = \arg \max_{\mathbf{w}} \log [p(T|X, \mathbf{w})p(\mathbf{w})] \\ \dots &= \arg \max_{\mathbf{w}} \left[\sum_n \log p(t_n|\mathbf{x}_n, \mathbf{w}) + \log p(\mathbf{w}) \right] =\end{aligned}$$

MAP estimate with zero-mean spherical Gaussian prior

Instead of $\log p(T|X, \mathbf{w})$ as in MLE, we optimize $\log p(\mathbf{w}|T, X)$:

$$\mathbf{w}_{\text{MAP}} = \arg \max_{\mathbf{w}} \log p(\mathbf{w}|T, X) = \arg \max_{\mathbf{w}} \log [p(T|X, \mathbf{w})p(\mathbf{w})]$$

$$\dots = \arg \max_{\mathbf{w}} \left[\sum_n \log p(t_n | \mathbf{x}_n, \mathbf{w}) + \log p(\mathbf{w}) \right] =$$

$$\dots = \arg \min_{\mathbf{w}} \left[\underbrace{\sum_n (t_n - \mathbf{w}^T \mathbf{x}_n)^2}_{\text{fit to the data (ML)}} + \underbrace{\frac{\sigma_\epsilon^2}{\sigma_w^2} \mathbf{w}^T \mathbf{w}}_{\text{keep } \mathbf{w} \text{ simple}} \right]$$

MAP estimate with zero-mean spherical Gaussian prior

Instead of $\log p(T|X, \mathbf{w})$ as in MLE, we optimize $\log p(\mathbf{w}|T, X)$:

$$\begin{aligned}\mathbf{w}_{\text{MAP}} &= \arg \max_{\mathbf{w}} \log p(\mathbf{w}|T, X) = \arg \max_{\mathbf{w}} \log [p(T|X, \mathbf{w})p(\mathbf{w})] \\ \dots &= \arg \max_{\mathbf{w}} \left[\sum_n \log p(t_n | \mathbf{x}_n, \mathbf{w}) + \log p(\mathbf{w}) \right] = \\ \dots &= \arg \min_{\mathbf{w}} \left[\underbrace{\sum_n (t_n - \mathbf{w}^T \mathbf{x}_n)^2}_{\text{fit to the data (ML)}} + \underbrace{\frac{\sigma_\epsilon^2}{\sigma_w^2} \mathbf{w}^T \mathbf{w}}_{\text{keep } \mathbf{w} \text{ simple}} \right]\end{aligned}$$

Equivalent to **ridge regression** with $\lambda = \frac{\sigma_\epsilon^2}{\sigma_w^2}$

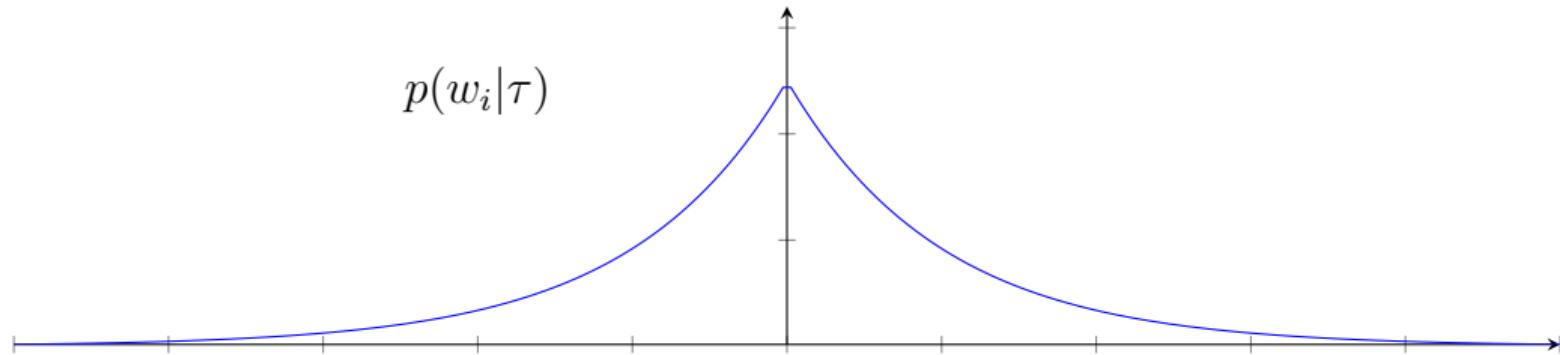
Example: Prior for LASSO

- LASSO: Least Absolute Shrinkage and Selection Operator
- We want the regularization to be $\lambda \sum_i |w_i|$ instead of $\lambda \sum_i w_i^2$.

Example: Prior for LASSO

- LASSO: Least Absolute Shrinkage and Selection Operator
- We want the regularization to be $\lambda \sum_i |w_i|$ instead of $\lambda \sum_i w_i^2$.
- Following the same arguments as before, we will need a product of zero-mean Laplace priors:

$$p(\mathbf{w}|\tau) = \prod_i \text{Laplace}(w_i, 0, \tau) = \prod_i \frac{1}{2\tau} \exp\left(-\frac{|w_i|}{\tau}\right)$$



Conjugate Prior

Definition:

if posterior and prior in the same family of functions

Examples:

Likelihood	Conjugate prior
Bernoulli	Beta
Binomial	Beta
Categorical	Dirichlet
Normal	Normal
Normal	Normal-inverse Gamma

Conjugate Priors and Iterative learning

- we start with prior $p(\theta)$
- we use a data set \mathcal{D}_1 to estimate posterior $p(\theta|\mathcal{D}_1)$

If new data \mathcal{D}_2 becomes available:

- we can use $p(\theta|\mathcal{D}_1)$ as prior
- and use \mathcal{D}_2 to estimate new posterior $p(\theta|\mathcal{D}_1, \mathcal{D}_2)$

Conjugate Priors and Iterative learning

- we start with prior $p(\theta)$
- we use a data set \mathcal{D}_1 to estimate posterior $p(\theta|\mathcal{D}_1)$

If new data \mathcal{D}_2 becomes available:

- we can use $p(\theta|\mathcal{D}_1)$ as prior
- and use \mathcal{D}_2 to estimate new posterior $p(\theta|\mathcal{D}_1, \mathcal{D}_2)$

Notes:

- It is simple because $p(\theta|\mathcal{D}_1)$ has the same shape as $p(\theta)$
- we need to keep the whole posterior, not only point estimate θ_{MAP}

Outline

1 Learning as Inference

2 Point Estimates

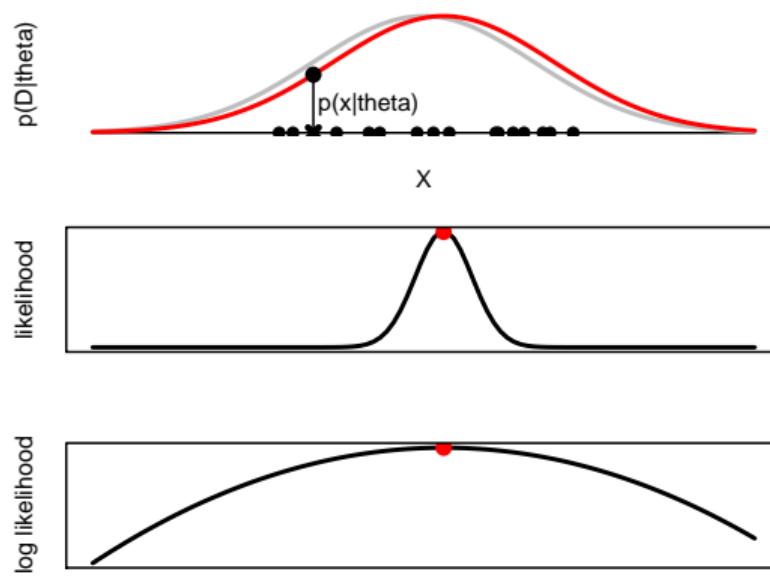
- Maximum Likelihood Estimation
- Maximum a Posteriori Estimation

3 Bayesian Methods

4 Curse of Dimensionality

ML, MAP and Point Estimates

- Both ML and MAP produce point estimates of θ
- Assumption: there is a **true** value for θ
- advantage: once $\hat{\theta}$ is found, everything is known



Limitations of MAP Estimate

- shift problem to defining the parameters of the prior (λ in Ridge and LASSO regression)
- uncertainty in the posterior $p(t|\mathbf{x}, \mathbf{w}_{\text{OPT}})$ is still σ_ϵ^2 and is independent of \mathbf{x}

Bayesian estimation (non-parametric models)

$$\begin{array}{lll} \text{ML: } & \mathcal{D} & \rightarrow \theta_{\text{ML}} \rightarrow P(\mathbf{o}_{\text{new}} | \theta_{\text{ML}}) \\ \text{MAP: } & \mathcal{D}, \textcolor{red}{P(\theta)} & \rightarrow \theta_{\text{MAP}} \rightarrow P(\mathbf{o}_{\text{new}} | \theta_{\text{MAP}}) \\ \text{Bayes: } & \mathcal{D}, \textcolor{red}{P(\theta)} & \rightarrow \textcolor{blue}{P(\theta | \mathcal{D})} \rightarrow P(\mathbf{o}_{\text{new}} | \mathcal{D}) \end{array}$$

- ① consider θ as a random variable (same as MAP)
- ② characterize θ with the posterior distribution $P(\theta | \mathcal{D})$ given the data
- ③ compute new predictive posterior $P(\mathbf{o}_{\text{new}} | \mathcal{D})$ marginalizing over θ (predictive posterior)

$$P(\mathbf{o}_{\text{new}} | \mathcal{D}) = \int_{\theta \in \Theta} P(\mathbf{o}_{\text{new}} | \theta) \textcolor{blue}{P(\theta | \mathcal{D})} d\theta$$

Bayesian Linear Regression

Setup:

$$\mathcal{D} = \{(\mathbf{x}_1, t_1), \dots, (\mathbf{x}_N, t_N)\}$$

Model (same as MAP):

- t_1, \dots, t_n independent given \mathbf{w}
- $t_i \sim \mathcal{N}(\mathbf{w}^T \mathbf{x}_i, \sigma_\epsilon^2)$
- $\mathbf{w} \sim \mathcal{N}(0, \sigma_w^2 \mathbf{I})$, $\mathbf{w} = \{w_0, w_1, \dots, w_d\}$
- we assume σ_ϵ^2 and σ_w^2 are known: $\theta = \{\mathbf{w}\}$

Goal:

Estimate $p(t_{\text{new}} | \mathbf{x}_{\text{new}}, \mathcal{D})$

Bayesian Linear Regression

$$\begin{aligned} p(t_{\text{new}} | \mathbf{x}_{\text{new}}, \mathcal{D}) &= \int_{\mathbf{w} \in W} p(t_{\text{new}} | \mathbf{x}_{\text{new}}, \mathcal{D}, \mathbf{w}) p(\mathbf{w} | \mathbf{x}_{\text{new}}, \mathcal{D}) d\mathbf{w} \\ &= \int_{\mathbf{w} \in W} p(t_{\text{new}} | \mathbf{x}_{\text{new}}, \mathbf{w}) p(\mathbf{w} | \mathcal{D}) d\mathbf{w} \end{aligned}$$

Results obtained with many passages:

- if prior $p(\mathbf{w})$ is Gaussian, then posterior $p(\mathbf{w} | \mathcal{D})$ is still Gaussian
- because the likelihood $p(t_{\text{new}} | \mathbf{x}_{\text{new}}, \mathbf{w})$ is Gaussian, the predictive posterior $p(t_{\text{new}} | \mathbf{x}_{\text{new}}, \mathcal{D})$ is Gaussian as well.
- all the results can be obtained in closed form (in this case)

Complete Derivations

From **mathematicalmonk**'s YouTube channel:

- problem and model definition

<https://youtu.be/1WvnpjljKXA>

- posterior $p(\mathbf{w}|\mathcal{D})$, part 1–2

<https://youtu.be/nrd4AnDLR3U>

<https://youtu.be/qz2U8coNwV4>

- predictive posterior $p(t_{\text{new}}|\mathbf{x}_{\text{new}}, \mathcal{D})$, part 1–3

<https://youtu.be/xyuSiKXttxw>

<https://youtu.be/vTcsacTqlfQ>

<https://youtu.be/LCISTY9S6SQ>

Closed Form Solutions

Posterior $p(\mathbf{w}|\mathcal{D}) = \mathcal{N}(\mu, \Sigma)$, with:

$$\begin{aligned}\Sigma &= \frac{1}{\sigma_\epsilon^2} X^T X + \frac{1}{\sigma_w^2} \mathbf{I} \\ \mu &= \frac{1}{\sigma_\epsilon^2} \Sigma^{-1} X^T T\end{aligned}$$

Closed Form Solutions

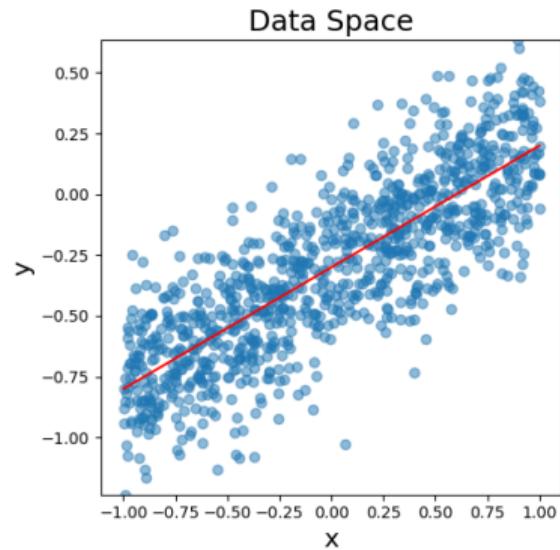
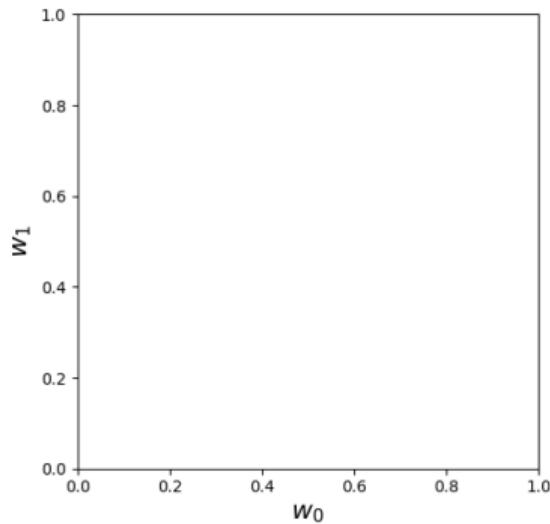
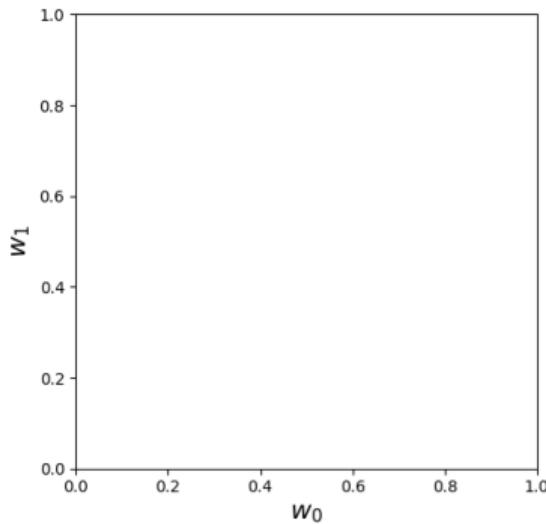
Posterior $p(\mathbf{w}|\mathcal{D}) = \mathcal{N}(\mu, \Sigma)$, with:

$$\begin{aligned}\Sigma &= \frac{1}{\sigma_\epsilon^2} X^T X + \frac{1}{\sigma_w^2} \mathbf{I} \\ \mu &= \frac{1}{\sigma_\epsilon^2} \Sigma^{-1} X^T T\end{aligned}$$

Predictive posterior

$$p(t_{\text{new}} | \mathbf{x}_{\text{new}}, \mathcal{D}) = \mathcal{N}(\mu^T \mathbf{x}_{\text{new}}, \sigma_\epsilon^2 + \mathbf{x}_{\text{new}}^T \Sigma \mathbf{x}_{\text{new}})$$

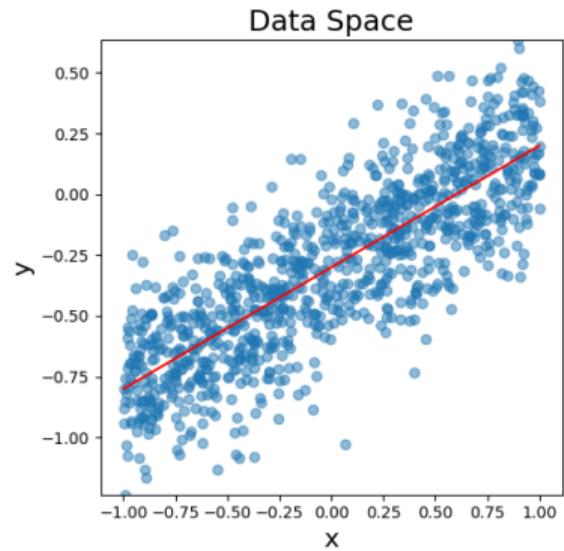
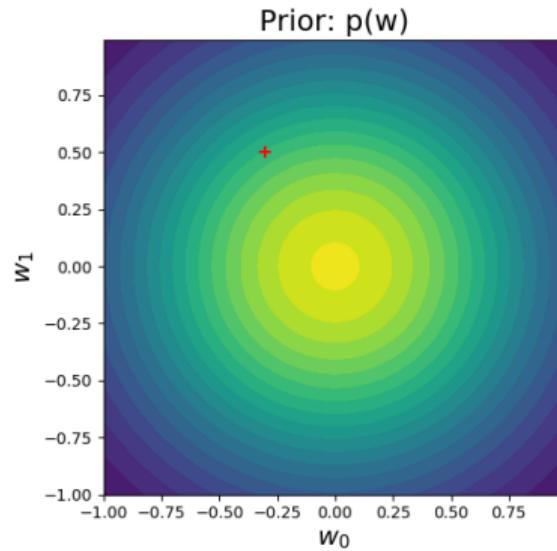
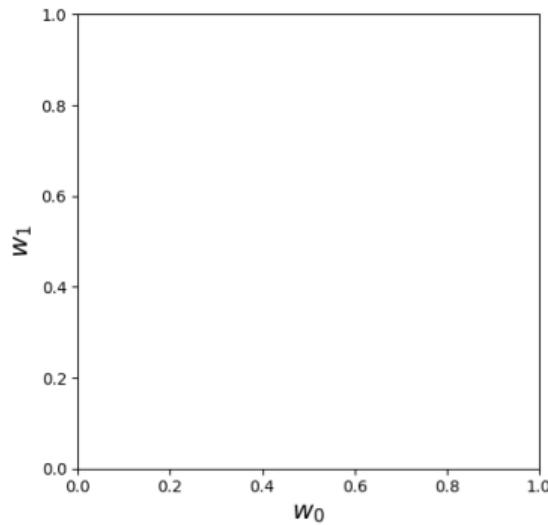
Bayesian Linear Regression: Example



Largely adapted from <https://zjost.github.io/bayesian-linear-regression/>

Inspired by Fig 3.7 in Bishop's Pattern Recognition and Machine Learning

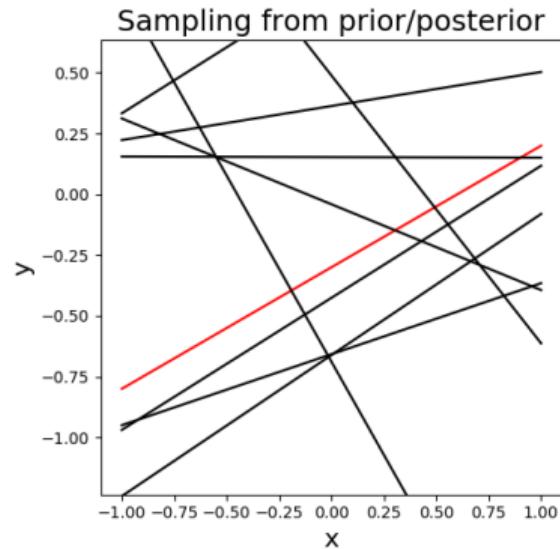
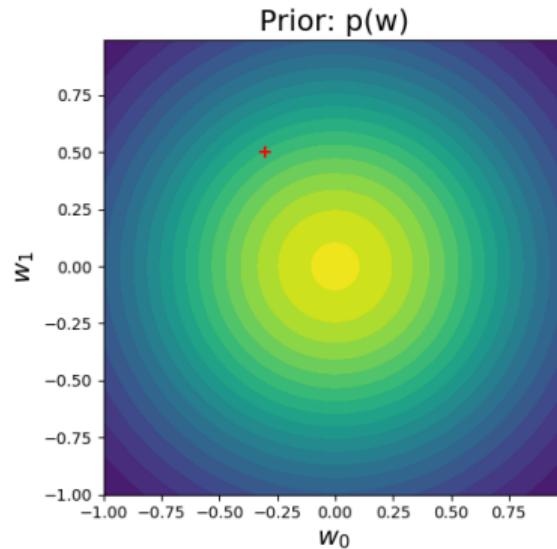
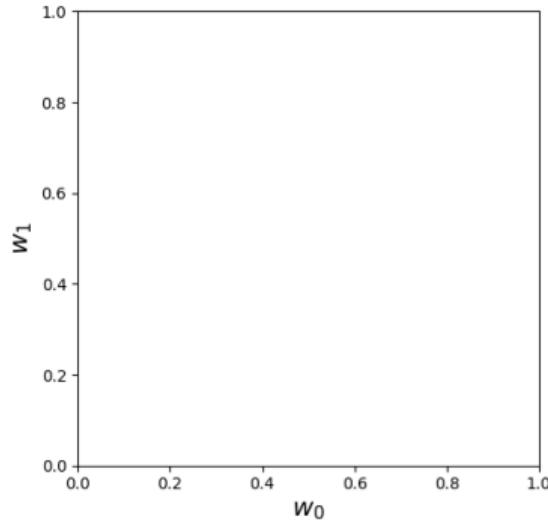
Bayesian Linear Regression: Example



Largely adapted from <https://zjost.github.io/bayesian-linear-regression/>

Inspired by Fig 3.7 in Bishop's Pattern Recognition and Machine Learning

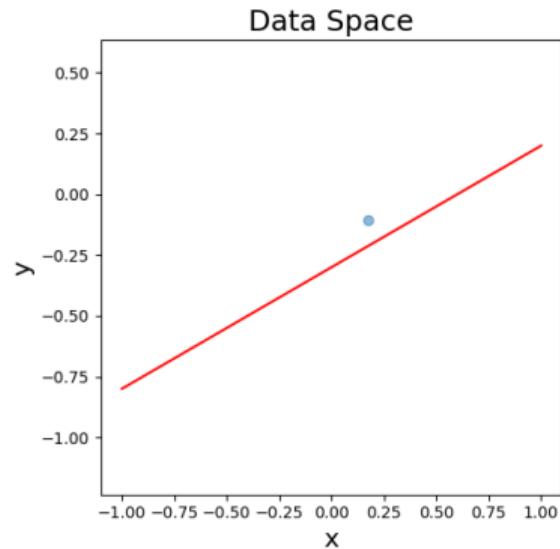
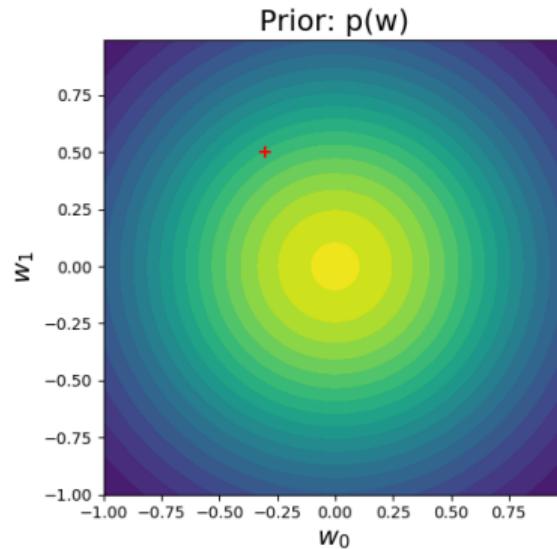
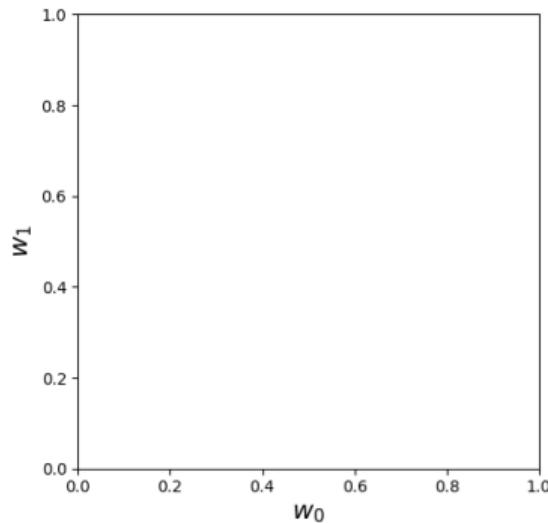
Bayesian Linear Regression: Example



Largely adapted from <https://zjost.github.io/bayesian-linear-regression/>

Inspired by Fig 3.7 in Bishop's Pattern Recognition and Machine Learning

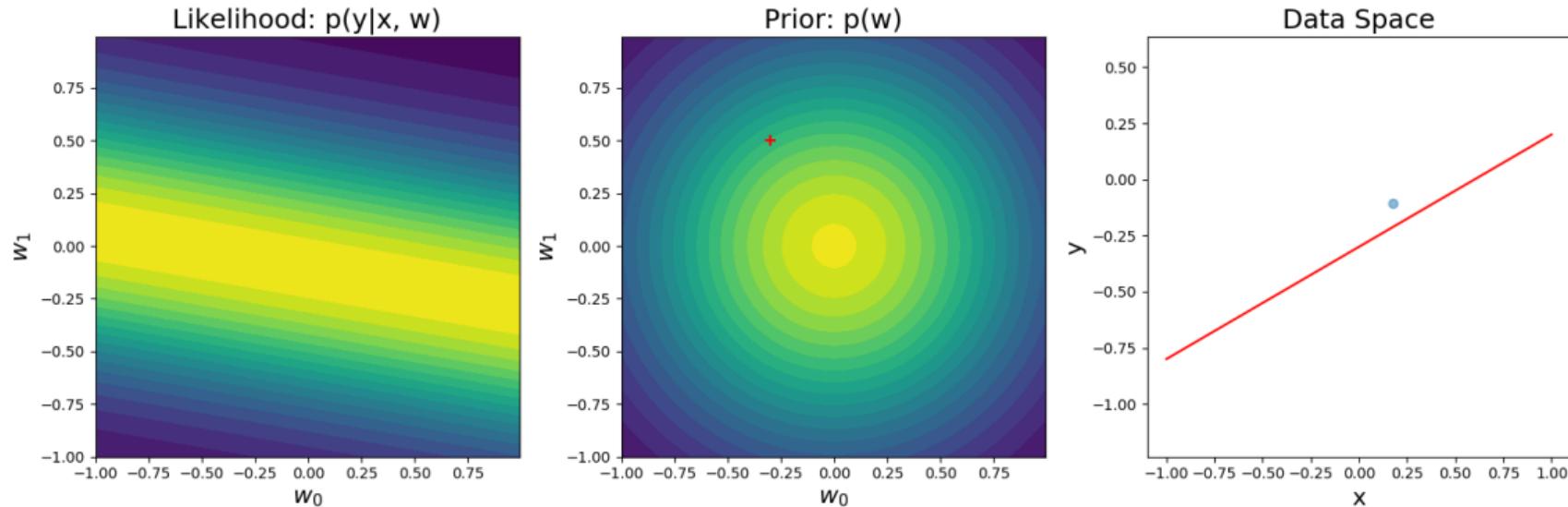
Bayesian Linear Regression: Example



Largely adapted from <https://zjost.github.io/bayesian-linear-regression/>

Inspired by Fig 3.7 in Bishop's Pattern Recognition and Machine Learning

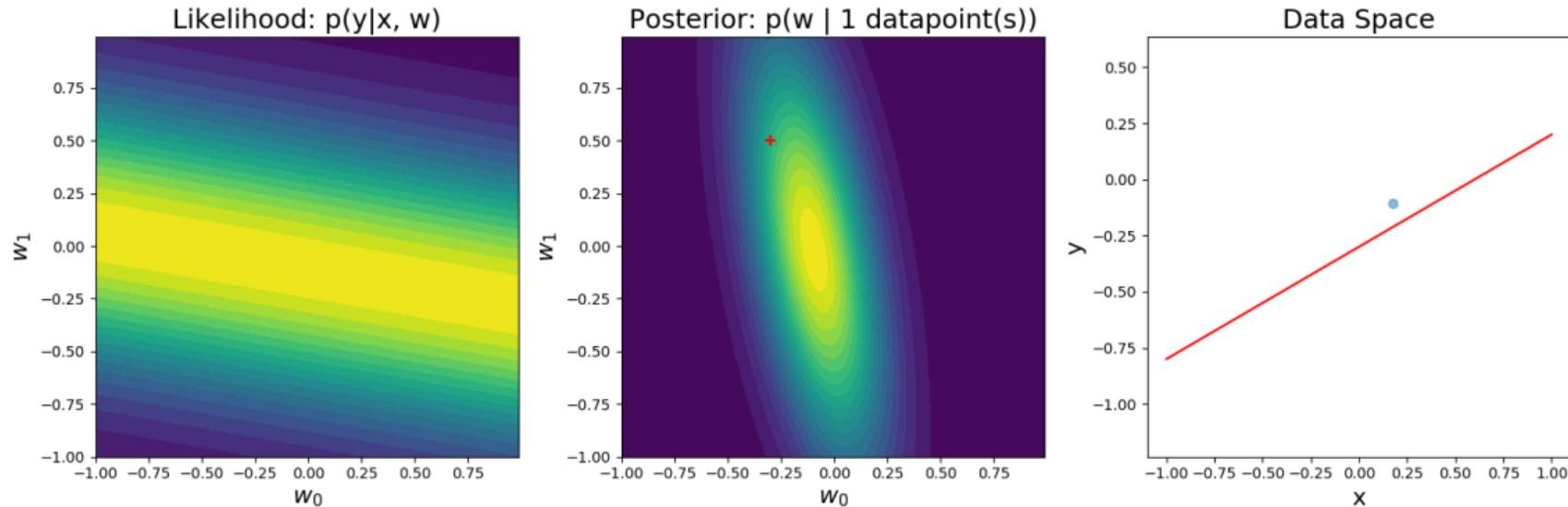
Bayesian Linear Regression: Example



Largely adapted from <https://zjost.github.io/bayesian-linear-regression/>

Inspired by Fig 3.7 in Bishop's Pattern Recognition and Machine Learning

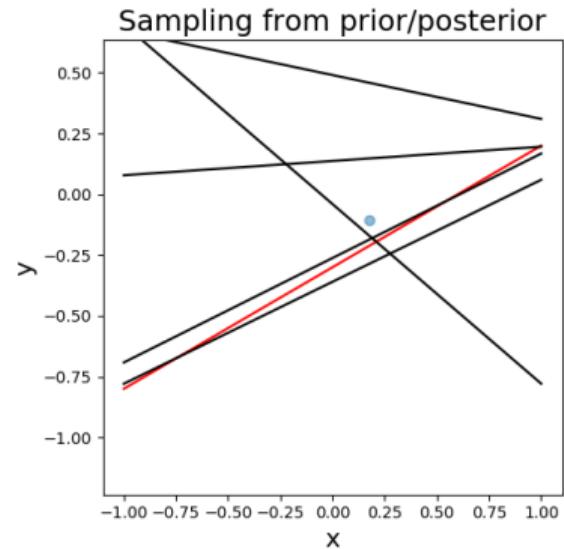
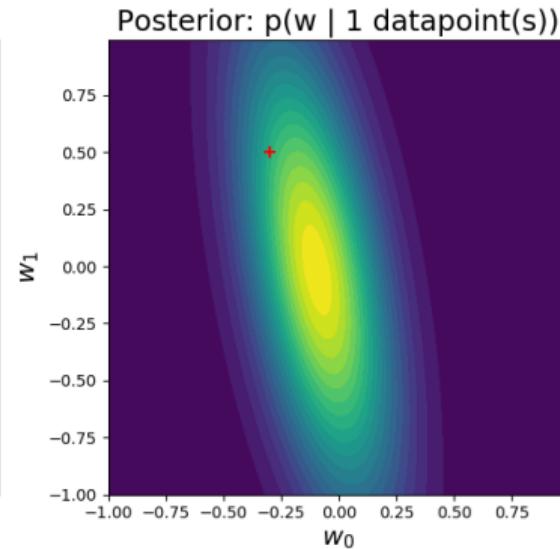
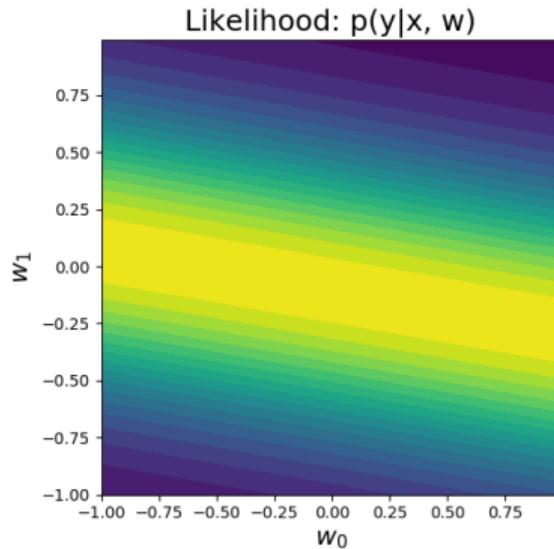
Bayesian Linear Regression: Example



Largely adapted from <https://zjost.github.io/bayesian-linear-regression/>

Inspired by Fig 3.7 in Bishop's Pattern Recognition and Machine Learning

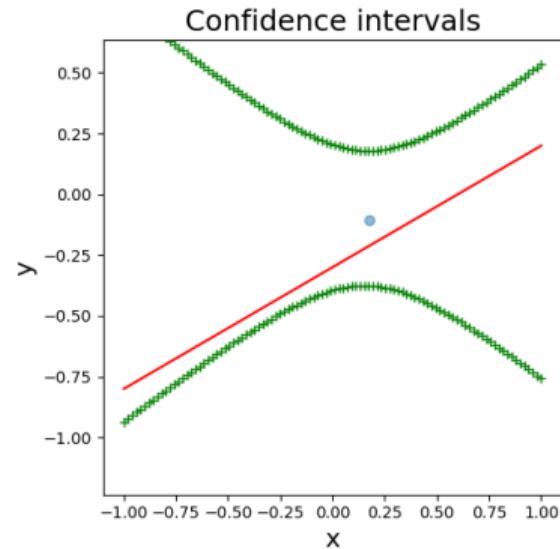
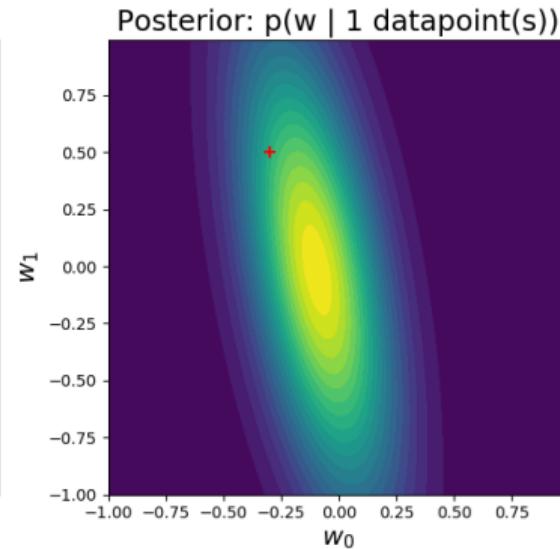
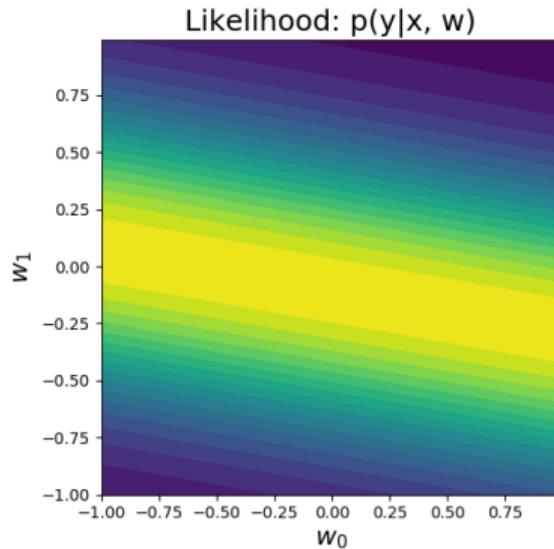
Bayesian Linear Regression: Example



Largely adapted from <https://zjost.github.io/bayesian-linear-regression/>

Inspired by Fig 3.7 in Bishop's Pattern Recognition and Machine Learning

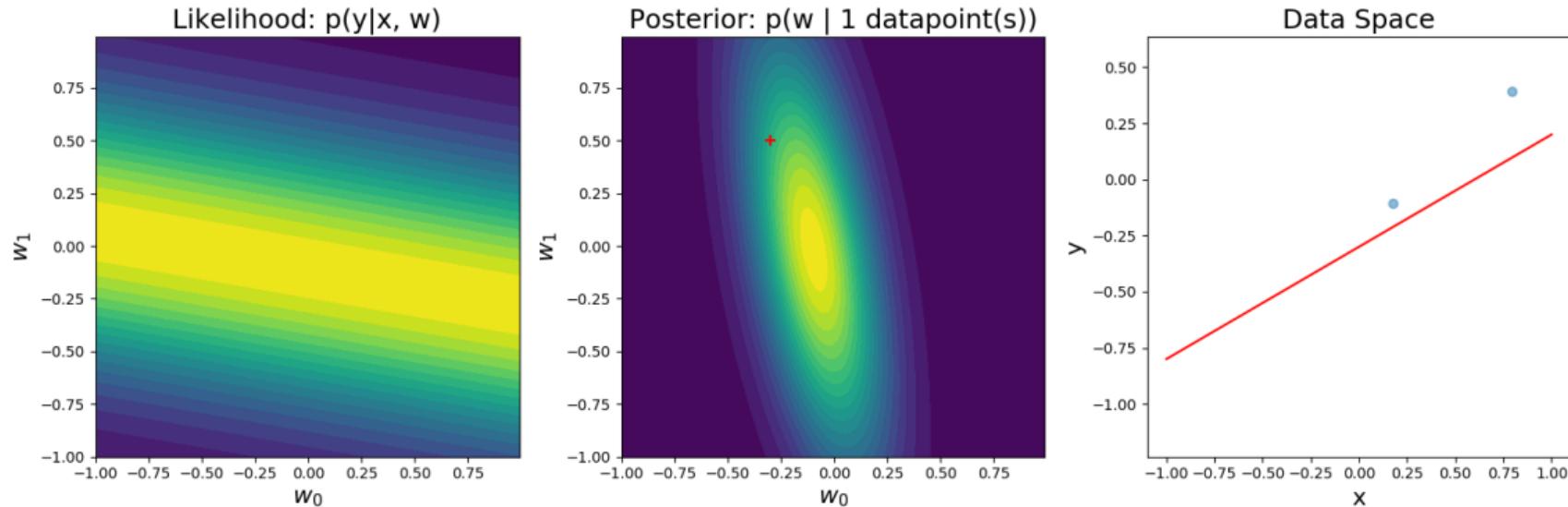
Bayesian Linear Regression: Example



Largely adapted from <https://zjost.github.io/bayesian-linear-regression/>

Inspired by Fig 3.7 in Bishop's Pattern Recognition and Machine Learning

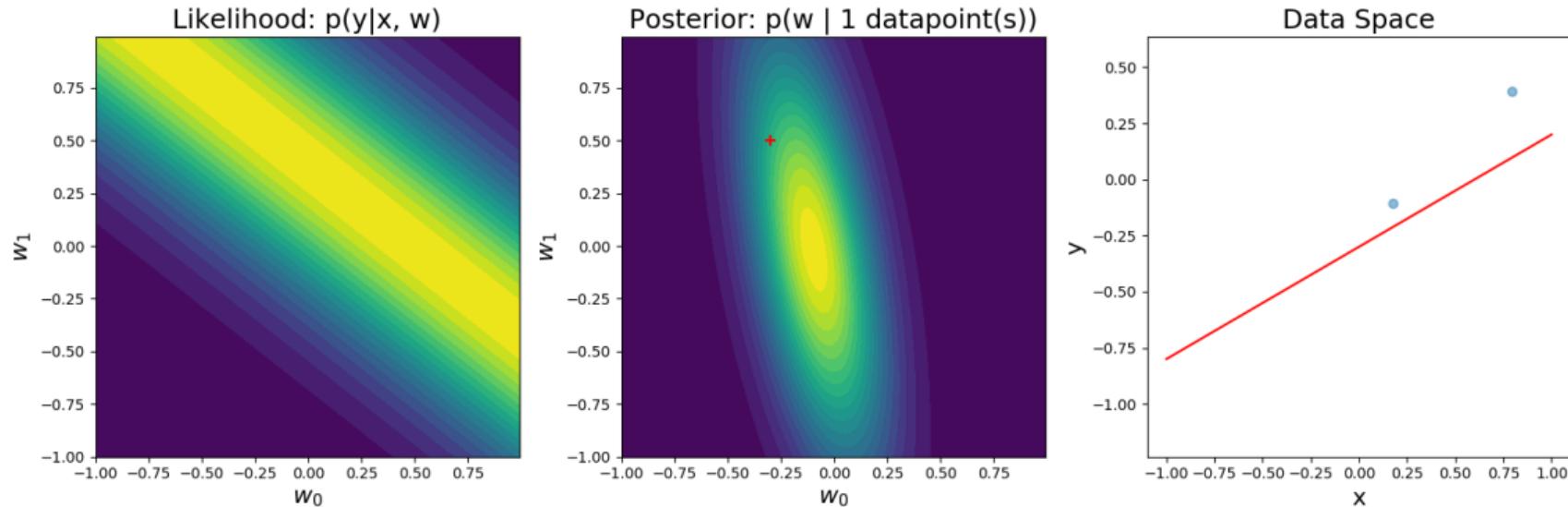
Bayesian Linear Regression: Example



Largely adapted from <https://zjost.github.io/bayesian-linear-regression/>

Inspired by Fig 3.7 in Bishop's Pattern Recognition and Machine Learning

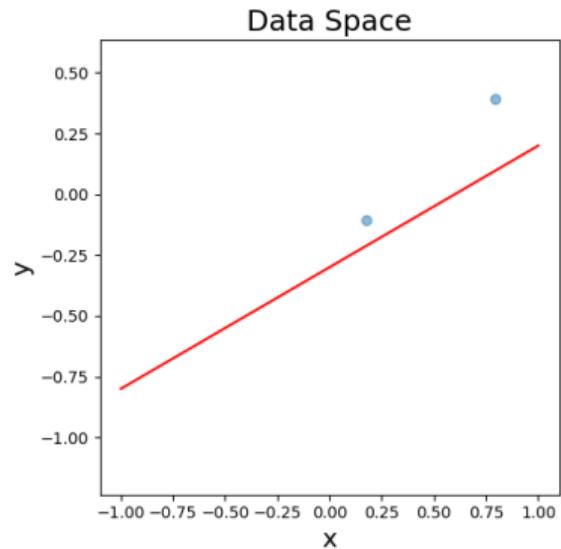
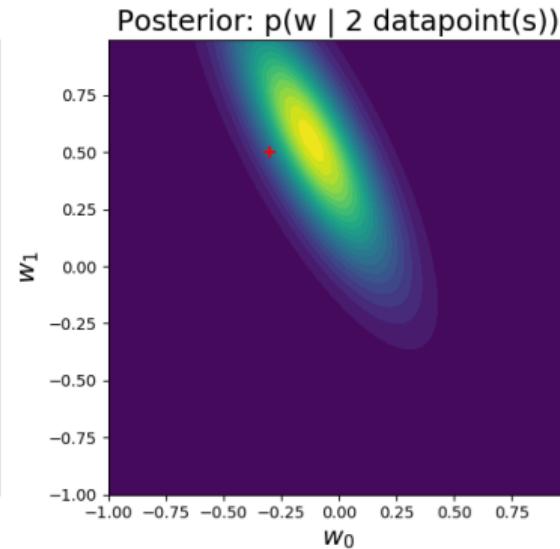
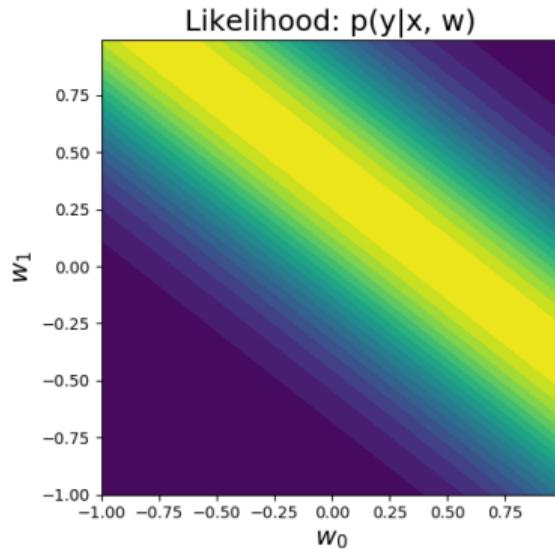
Bayesian Linear Regression: Example



Largely adapted from <https://zjost.github.io/bayesian-linear-regression/>

Inspired by Fig 3.7 in Bishop's Pattern Recognition and Machine Learning

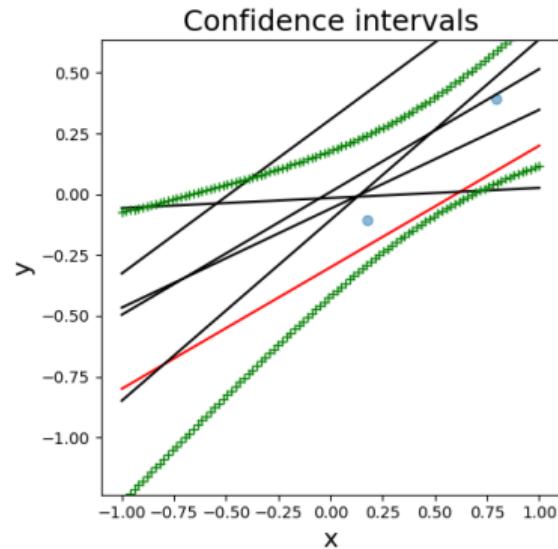
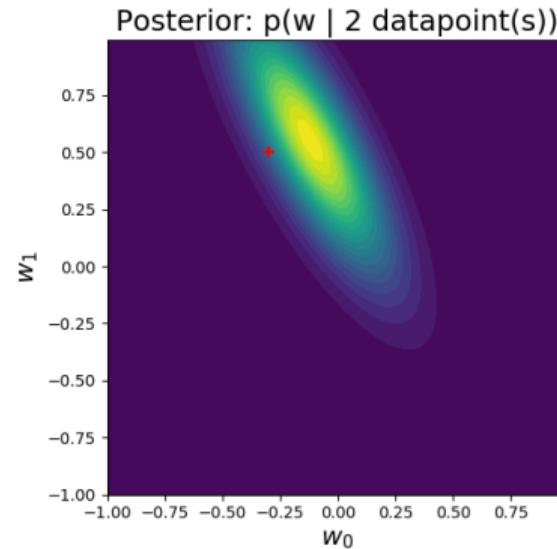
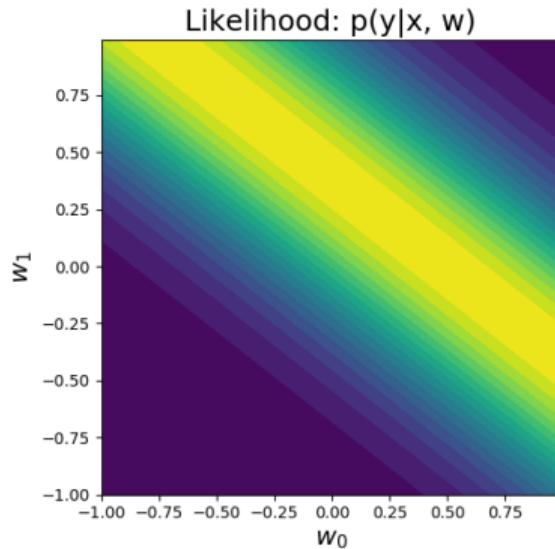
Bayesian Linear Regression: Example



Largely adapted from <https://zjost.github.io/bayesian-linear-regression/>

Inspired by Fig 3.7 in Bishop's Pattern Recognition and Machine Learning

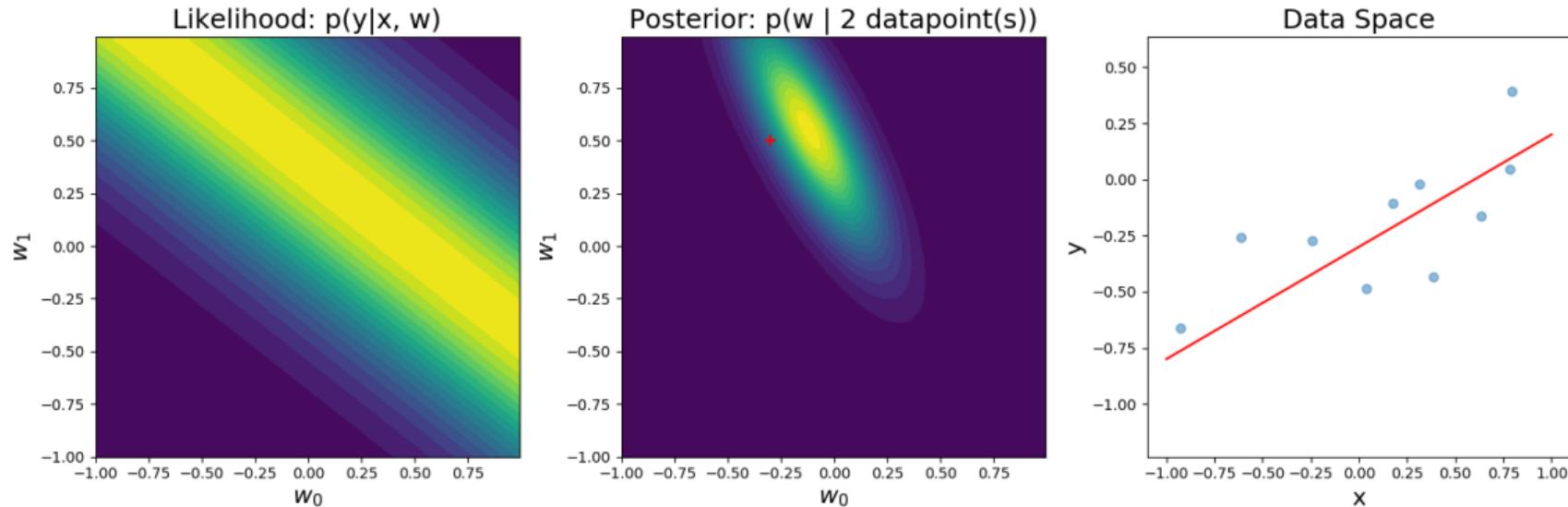
Bayesian Linear Regression: Example



Largely adapted from <https://zjost.github.io/bayesian-linear-regression/>

Inspired by Fig 3.7 in Bishop's Pattern Recognition and Machine Learning

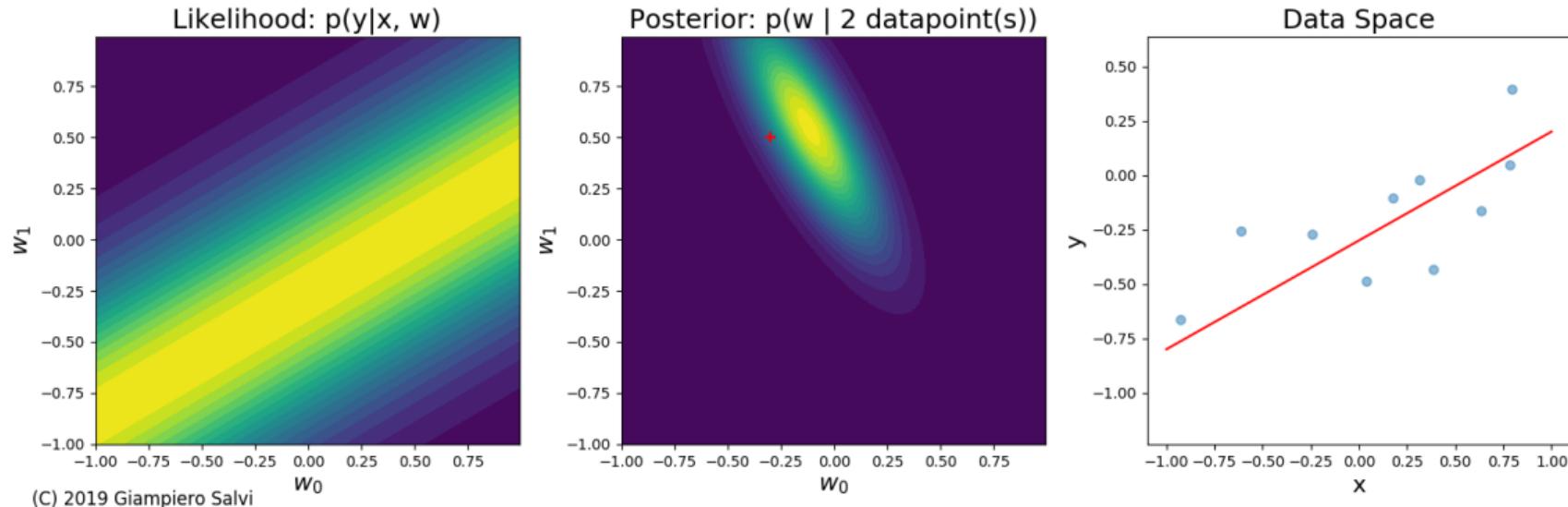
Bayesian Linear Regression: Example



Largely adapted from <https://zjost.github.io/bayesian-linear-regression/>

Inspired by Fig 3.7 in Bishop's Pattern Recognition and Machine Learning

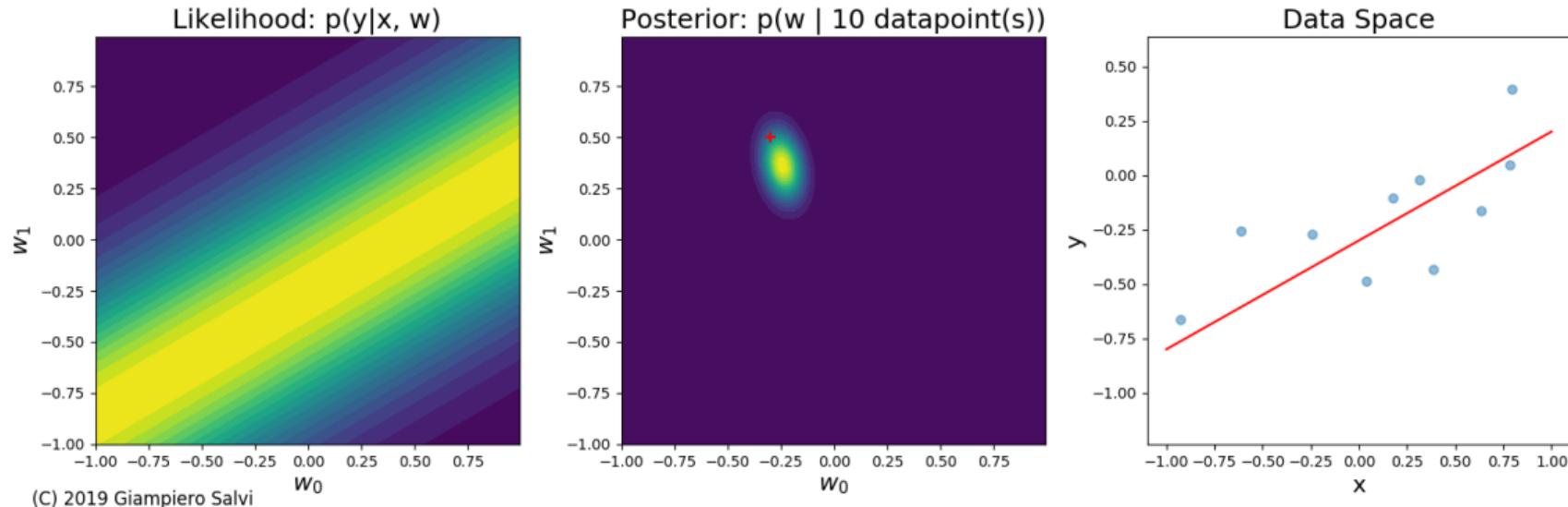
Bayesian Linear Regression: Example



Largely adapted from <https://zjost.github.io/bayesian-linear-regression/>

Inspired by Fig 3.7 in Bishop's Pattern Recognition and Machine Learning

Bayesian Linear Regression: Example

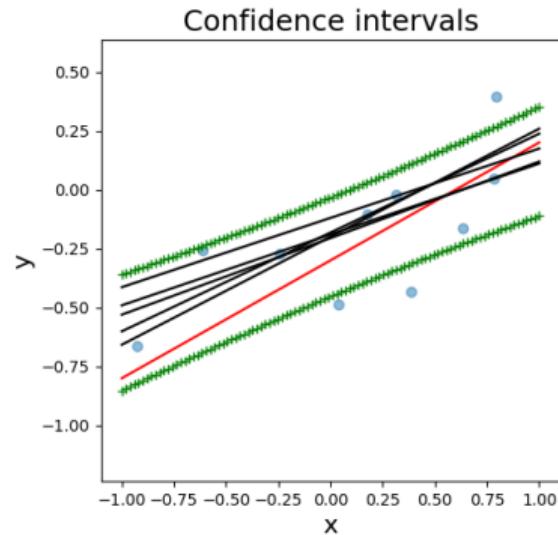
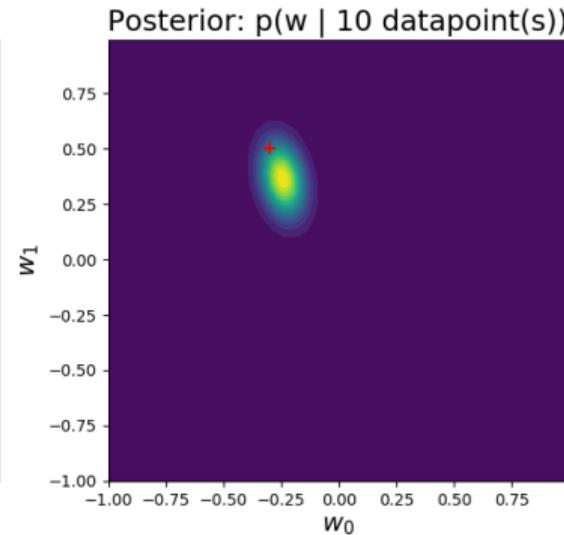
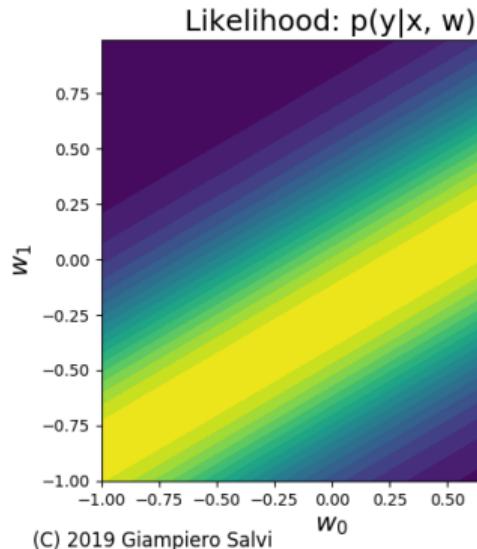


(C) 2019 Giampiero Salvi

Largely adapted from <https://zjost.github.io/bayesian-linear-regression/>

Inspired by Fig 3.7 in Bishop's Pattern Recognition and Machine Learning

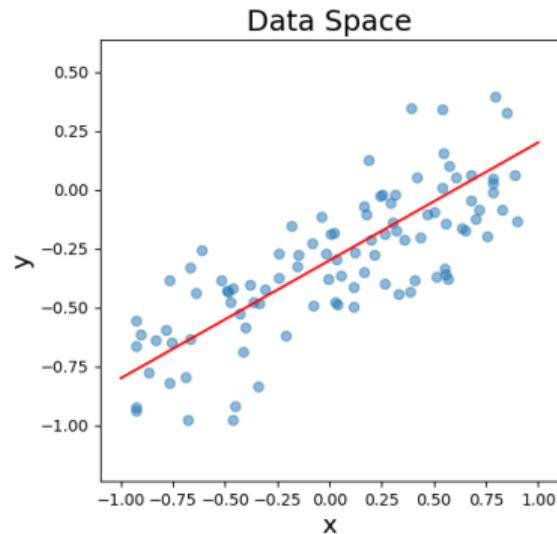
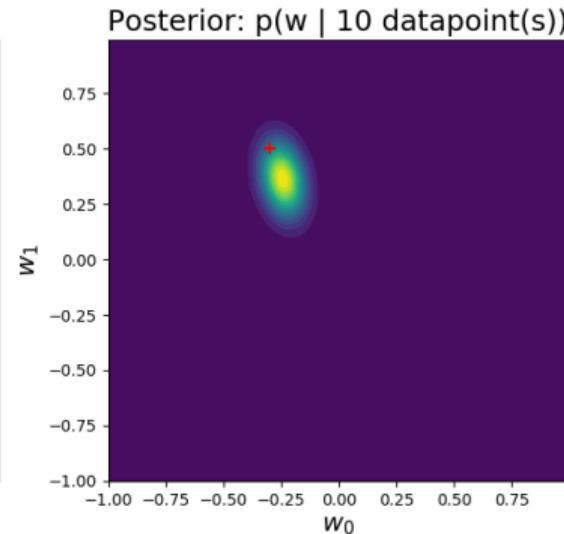
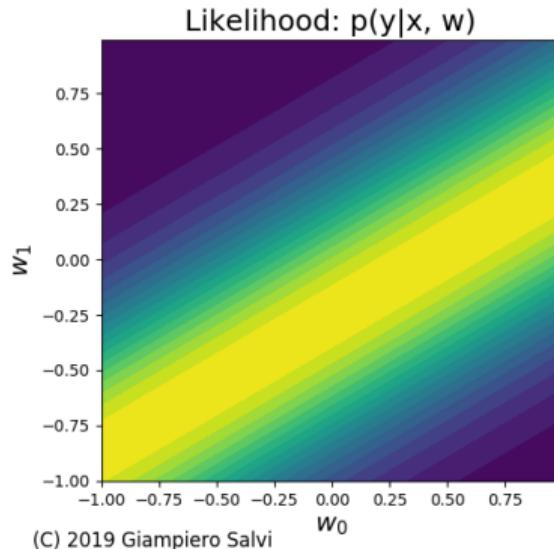
Bayesian Linear Regression: Example



Largely adapted from <https://zjost.github.io/bayesian-linear-regression/>

Inspired by Fig 3.7 in Bishop's Pattern Recognition and Machine Learning

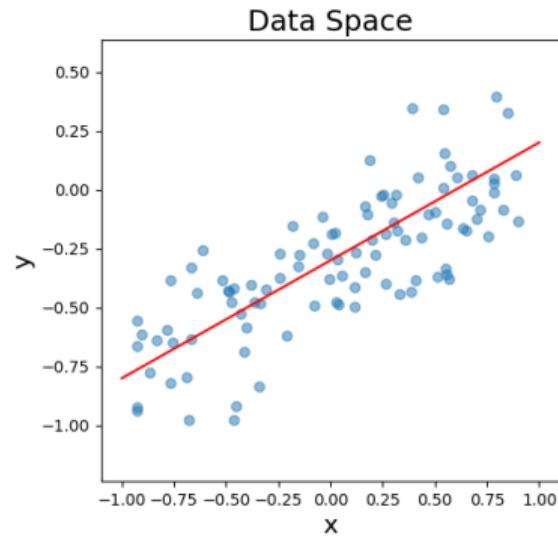
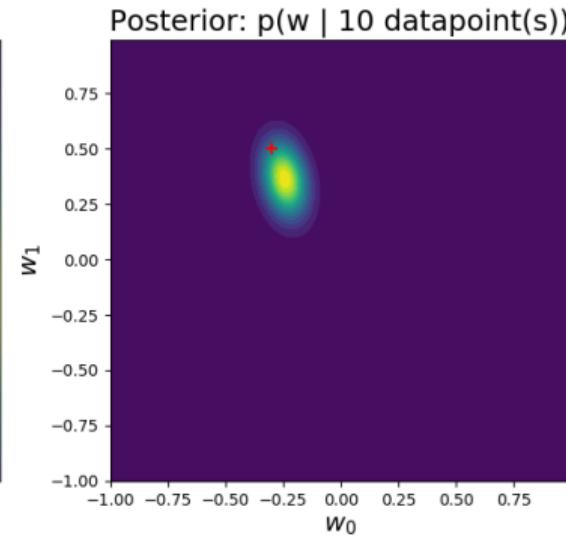
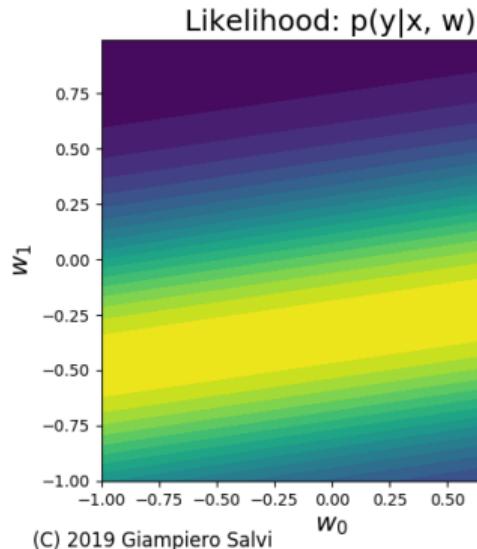
Bayesian Linear Regression: Example



Largely adapted from <https://zjost.github.io/bayesian-linear-regression/>

Inspired by Fig 3.7 in Bishop's Pattern Recognition and Machine Learning

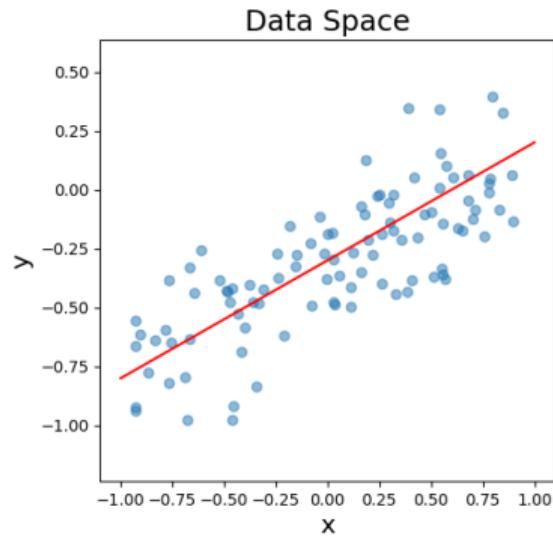
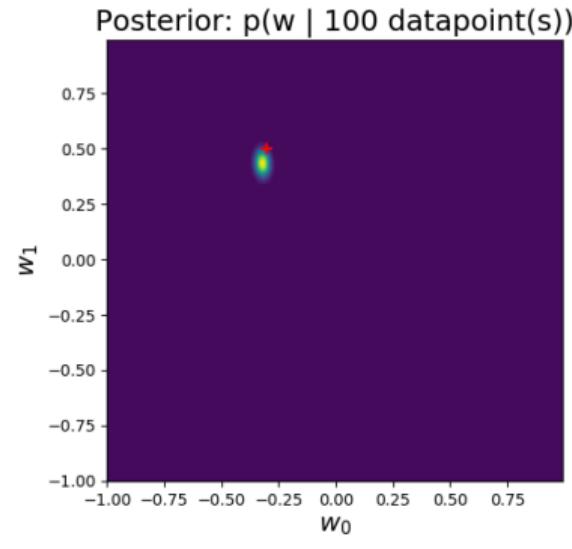
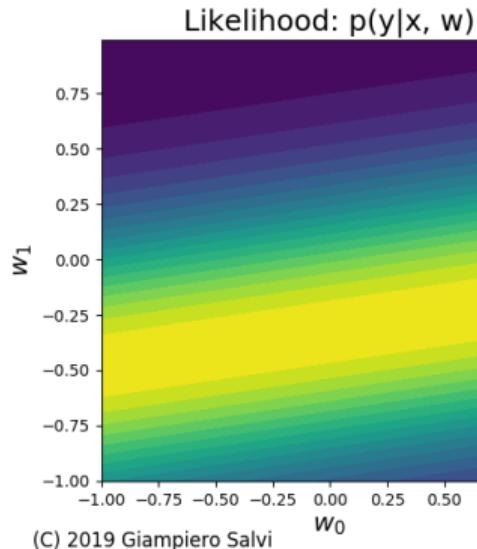
Bayesian Linear Regression: Example



Largely adapted from <https://zjost.github.io/bayesian-linear-regression/>

Inspired by Fig 3.7 in Bishop's Pattern Recognition and Machine Learning

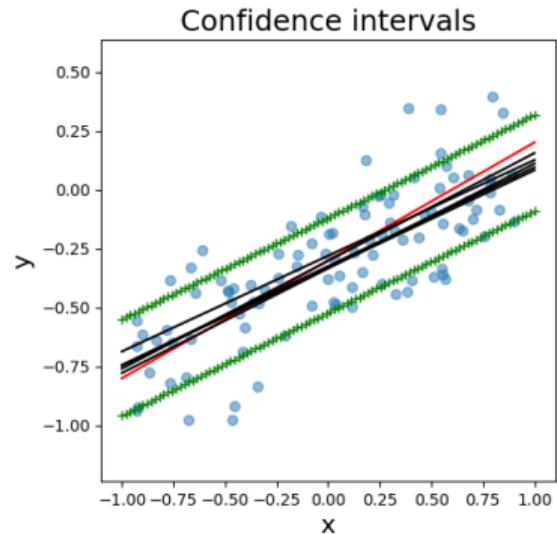
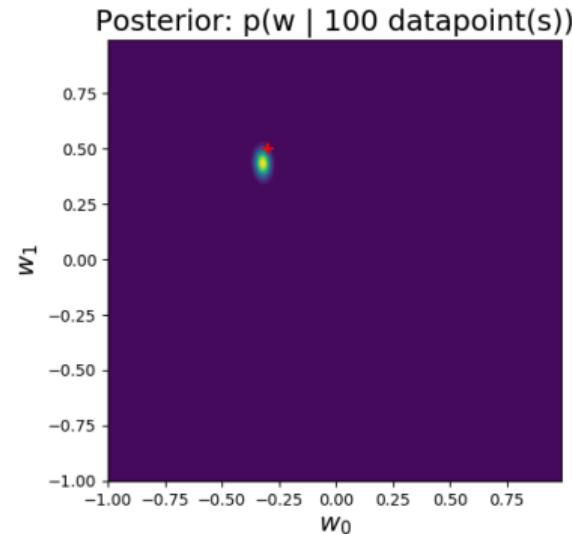
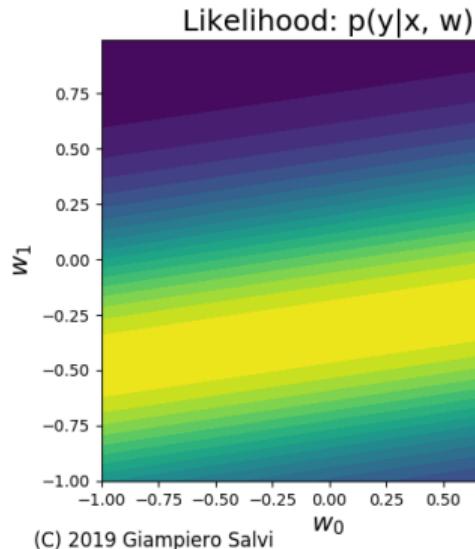
Bayesian Linear Regression: Example



Largely adapted from <https://zjost.github.io/bayesian-linear-regression/>

Inspired by Fig 3.7 in Bishop's Pattern Recognition and Machine Learning

Bayesian Linear Regression: Example

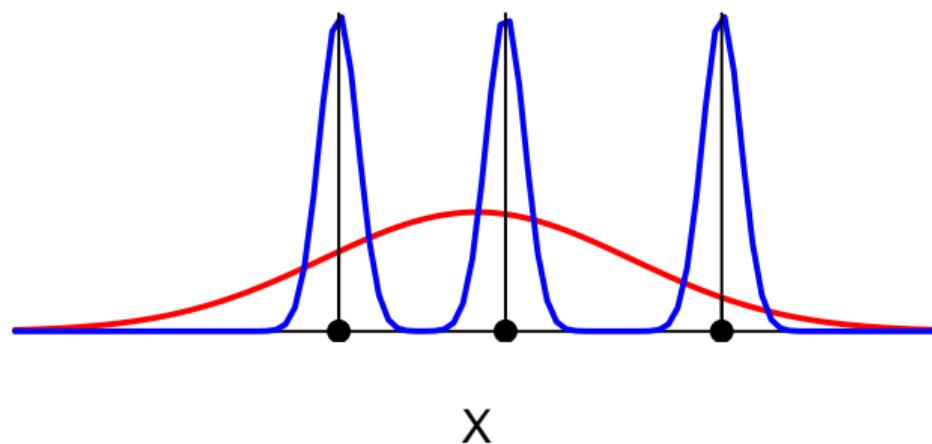


Largely adapted from <https://zjost.github.io/bayesian-linear-regression/>

Inspired by Fig 3.7 in Bishop's Pattern Recognition and Machine Learning

Overfitting and Maximum Likelihood

we can make the likelihood **arbitrary large** by increasing the number of parameters



Occam's Razor and Bayesian Learning

Remember that:

$$p(y_{\text{new}} | \mathbf{x}_{\text{new}}, \mathcal{D}) = \int_{\theta \in \Theta} p(y_{\text{new}} | \mathbf{x}_{\text{new}}, \theta) p(\theta | \mathcal{D}) d\theta$$

Occam's Razor and Bayesian Learning

Remember that:

$$p(y_{\text{new}} | \mathbf{x}_{\text{new}}, \mathcal{D}) = \int_{\theta \in \Theta} p(y_{\text{new}} | \mathbf{x}_{\text{new}}, \theta) p(\theta | \mathcal{D}) d\theta$$

Intuition:

More complex models fit the data very well (large $p(\mathcal{D}|\theta)$ and $p(\theta|\mathcal{D})$) but only for small regions of the parameter space Θ .

Limitations

- not always possible to compute posterior (**conjugate priors**)
- approximations with high computational cost (sampling methods) or complex solutions (variational methods)
- sometimes we want to have **non-informative priors**
- for unbounded continuous variables this can be difficult

Outline

1 Learning as Inference

2 Point Estimates

- Maximum Likelihood Estimation
- Maximum a Posteriori Estimation

3 Bayesian Methods

4 Curse of Dimensionality

Curse of dimensionality

1-dimension

$$y(x, w) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

(4 parameters)

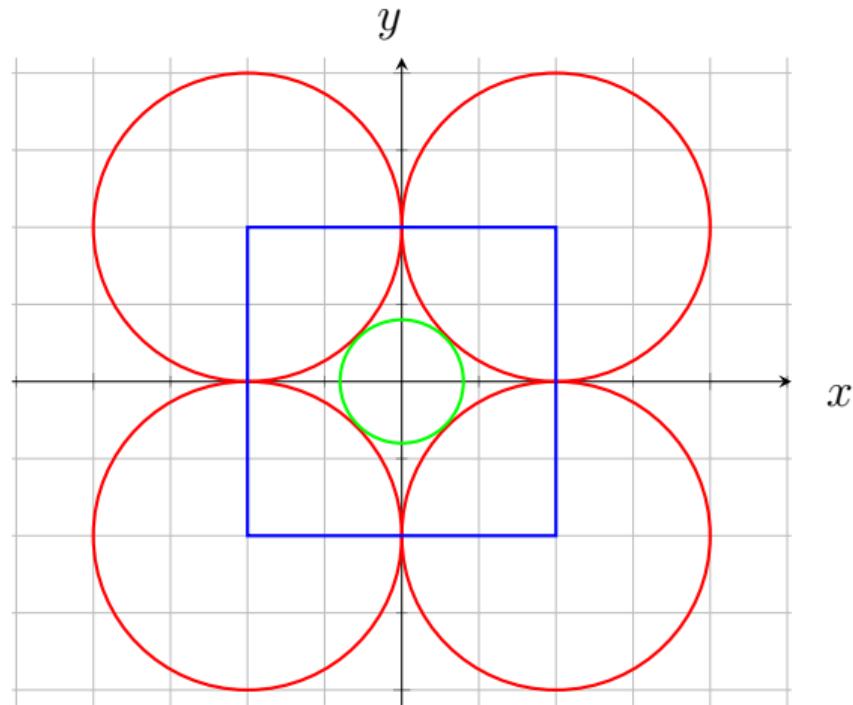
D -dimension

$$y(x, w) = w_0 + \sum_{i=1}^D w_i x_i + \sum_{i=1}^D \sum_{j=1}^D w_{ij} x_i x_j + \sum_{i=1}^D \sum_{j=1}^D \sum_{k=1}^D w_{ijk} x_i x_j x_k$$

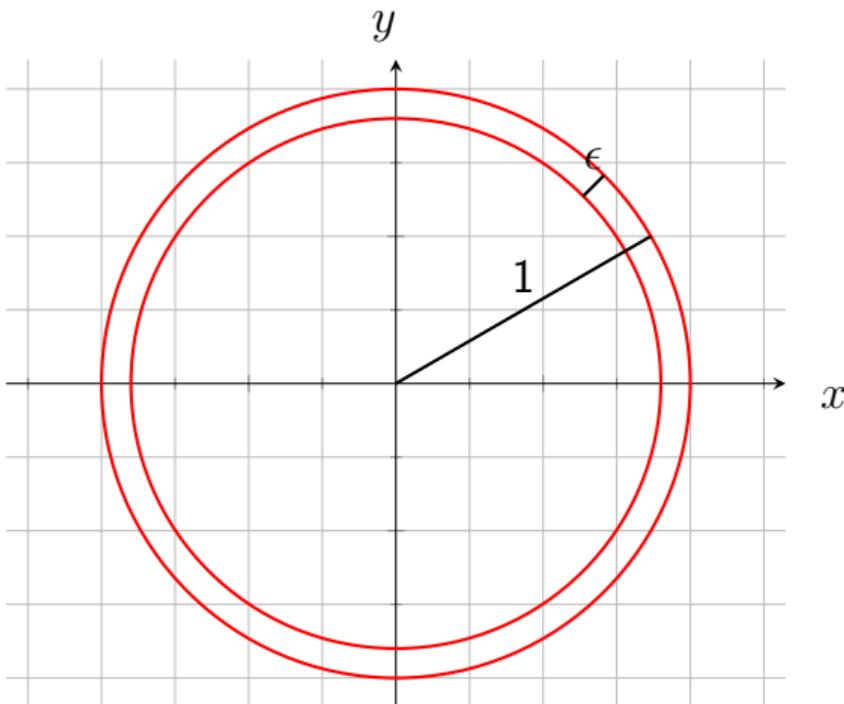
(1 + D + D^2 + D^3 parameters)

High dimensions and intuition

- radius of red circles = 1
- side of blue square = 2
- what is the radius of the green circle?
- what is the radius of the sphere in 3D?
- how about higher dimensions?



High dimensions and intuition

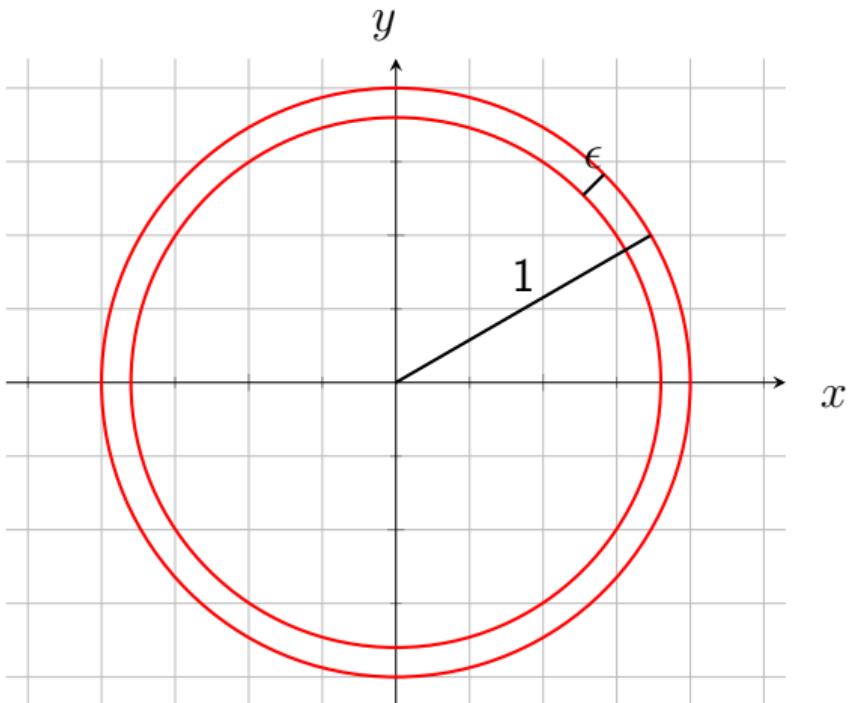


- What is ratio between the volume between the spheres and the volume of the large sphere?

$$\frac{V_D(1) - V_D(1 - \epsilon)}{V_D(1)} = \dots$$

- In D dimensions $V_D(r) = K_D r^D$
- Examples:
 - 2D: $K_2 = \pi$
 - 3D: $K_3 = \frac{4}{3}\pi$
 - ...

High dimensions and intuition



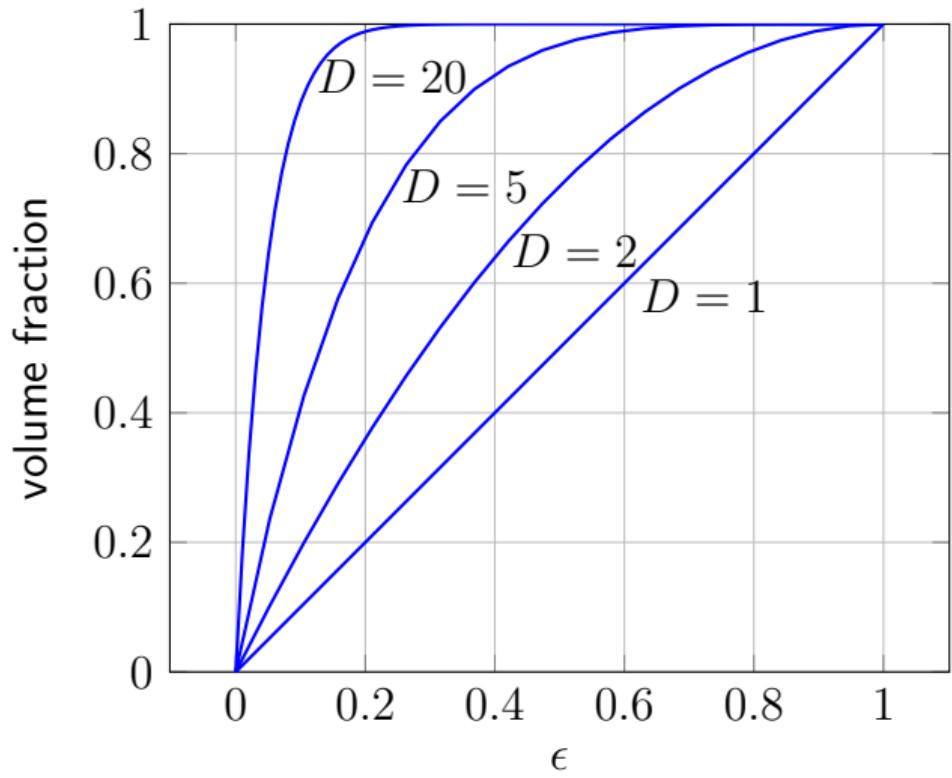
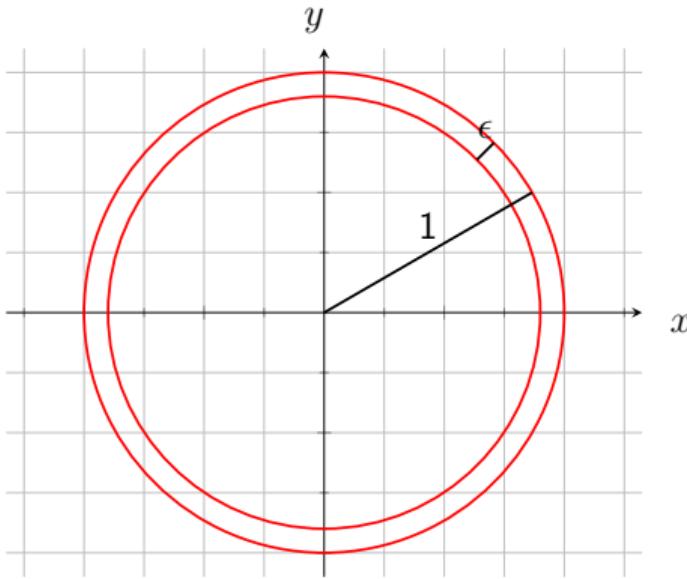
- What is ratio between the volume between the spheres and the volume of the large sphere?

$$\frac{V_D(1) - V_D(1 - \epsilon)}{V_D(1)} = \dots$$

- In D dimensions $V_D(r) = K_D r^D$

$$\dots = \frac{K_D 1^D - K_D (1 - \epsilon)^D}{K_D 1^D} = 1 - (1 - \epsilon)^D$$

High dimensions and intuition



Example: Euclidean Distance

Two points in D dimensions:

$$\mathbf{a} = (a_1, a_2, \dots, a_D)$$

$$\mathbf{b} = (b_1, b_2, \dots, b_D)$$

Euclidean square distance

$$d^2(\mathbf{a}, \mathbf{b}) = (a_1 - b_1)^2 + (a_2 - b_2)^2 + \dots + (a_D - b_D)^2$$

If $D = 1000$ it is enough that just a few coordinates differ.

Linear Models for Regression

TTT4185 Machine Learning for Signal Processing

Giampiero Salvi

Department of Electronic Systems
NTNU

HT2021

Outline

1 Curse of Dimensionality

- Manifolds and Dimensionality Reduction

2 Basis Functions

- Maximum Likelihood

3 Bias/Variance trade-off

4 Bayesian Model Selection

Outline

1 Curse of Dimensionality

- Manifolds and Dimensionality Reduction

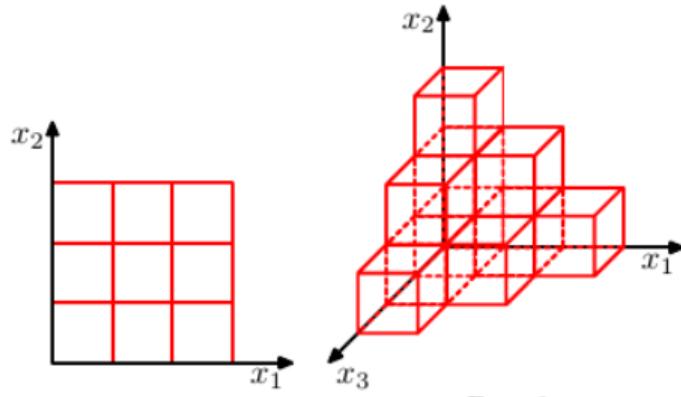
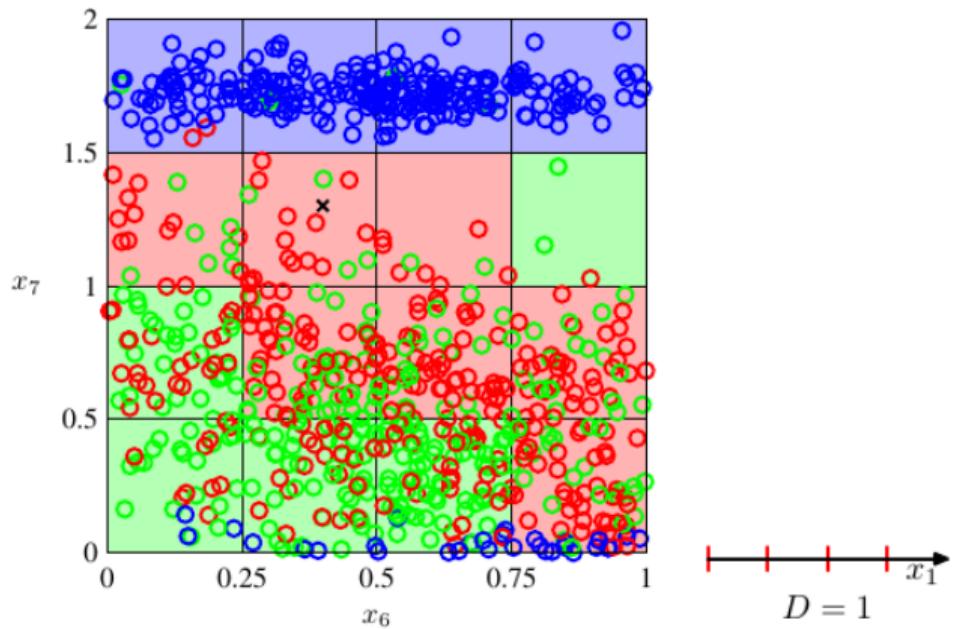
2 Basis Functions

- Maximum Likelihood

3 Bias/Variance trade-off

4 Bayesian Model Selection

Curse of dimensionality (non-parametric case)



Figures from Bishop

Curse of dimensionality (parametric case)

1-dimension $x \in \mathbb{R}$, third order polynomial

$$y(x, w) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

(4 parameters)

D -dimension $\mathbf{x} = \{x_1, \dots, x_D\} \in \mathbb{R}^D$, third order polynomial

$$y(x, w) = w_0 + \sum_{i=1}^D w_i x_i + \sum_{i=1}^D \sum_{j=1}^D w_{ij} x_i x_j + \sum_{i=1}^D \sum_{j=1}^D \sum_{k=1}^D w_{ijk} x_i x_j x_k$$

(1 + D + D^2 + D^3 parameters)

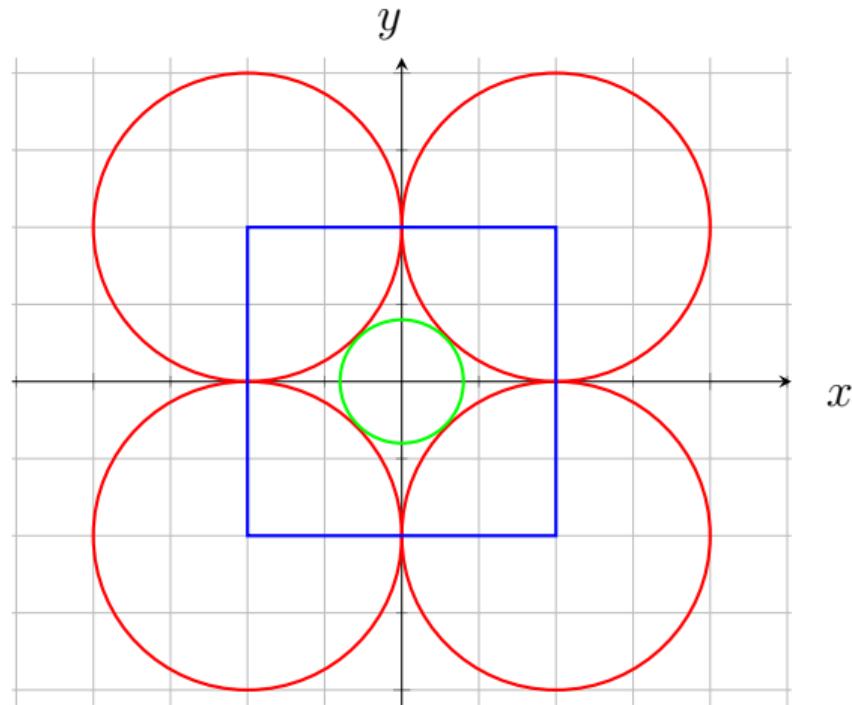
Example 28×28 images (MNIST): $D = 784$, # parameters = 482.505.745

High dimensions and intuition

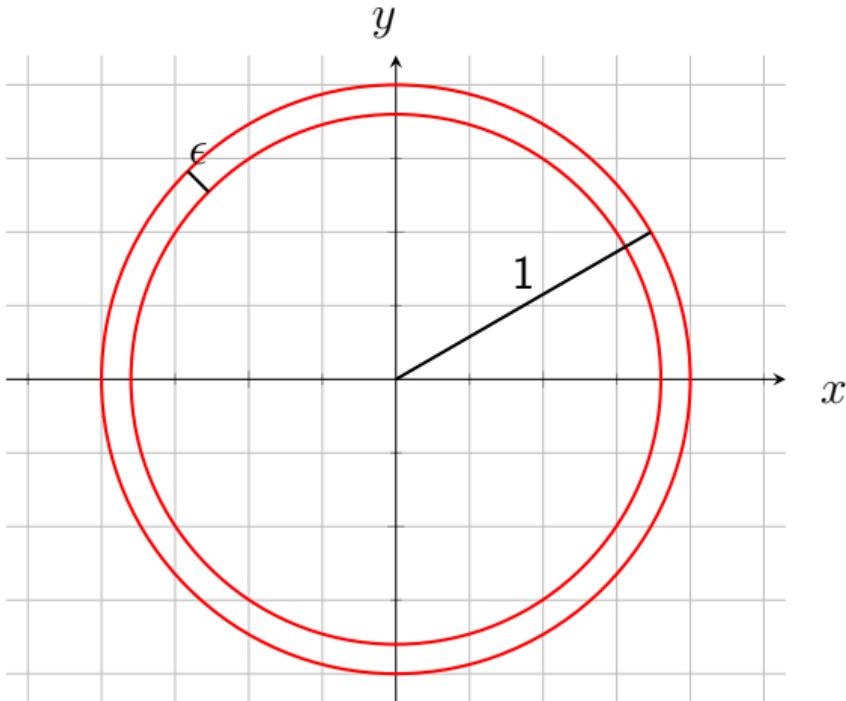
- radius of red circles = 1
- side of blue square = 2
- what is the radius of the green circle?
- what is the radius of the sphere in 3D?
- how about higher dimensions?

3Blue1Brown

<https://youtu.be/zwAD6dRSVyI>



High dimensions and intuition

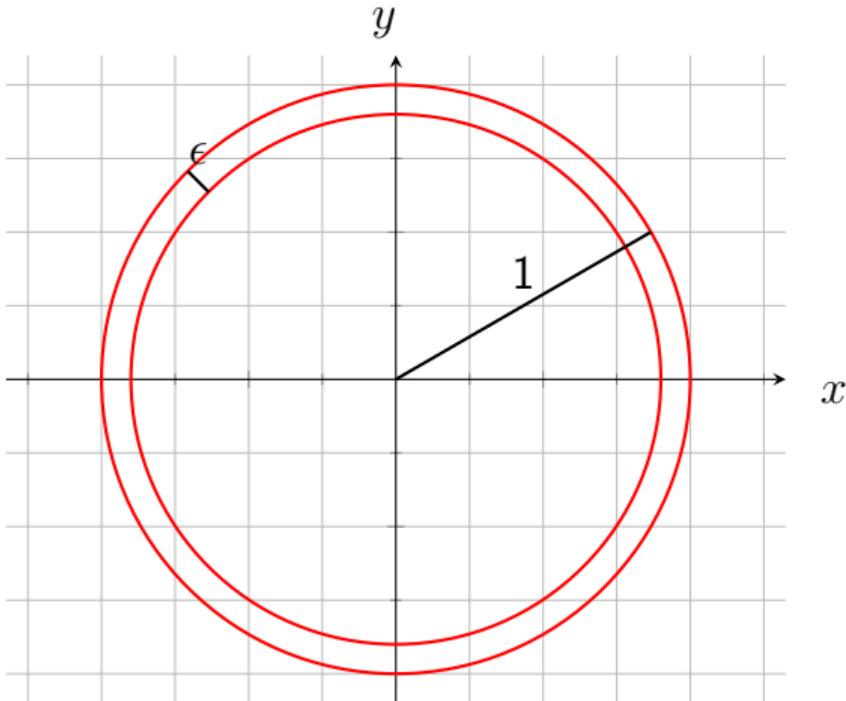


- What is ratio between the volume between the spheres and the volume of the large sphere?

$$\frac{V_D(1) - V_D(1 - \epsilon)}{V_D(1)} = \dots$$

- In D dimensions $V_D(r) = K_D r^D$
- Examples:
 - 2D: $K_2 = \pi$
 - 3D: $K_3 = \frac{4}{3}\pi$
 - ...

High dimensions and intuition



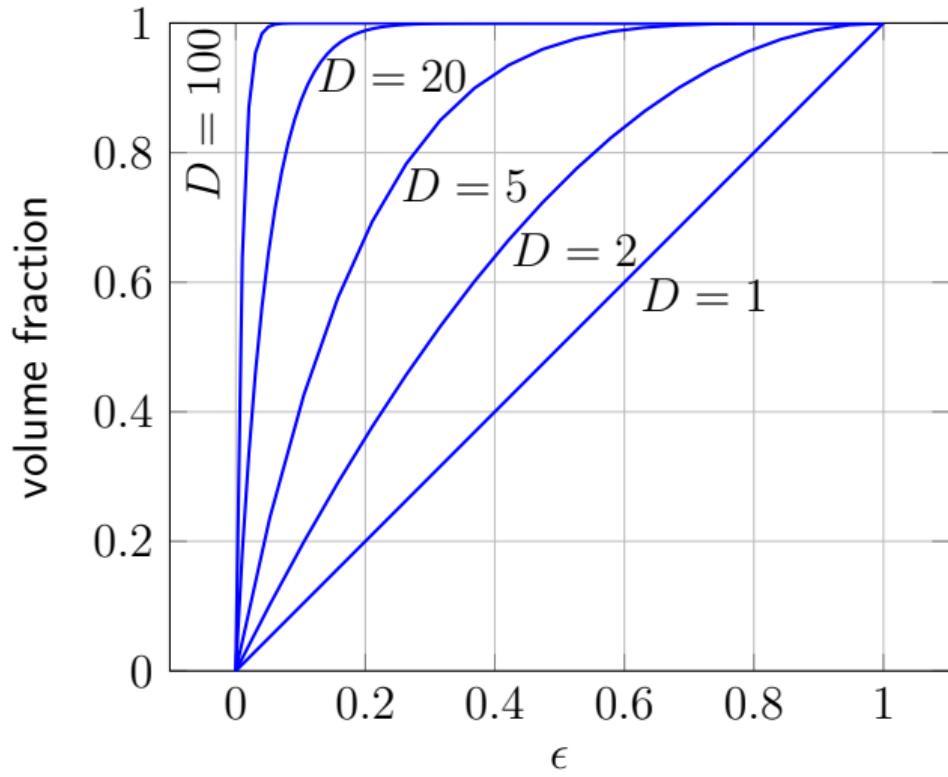
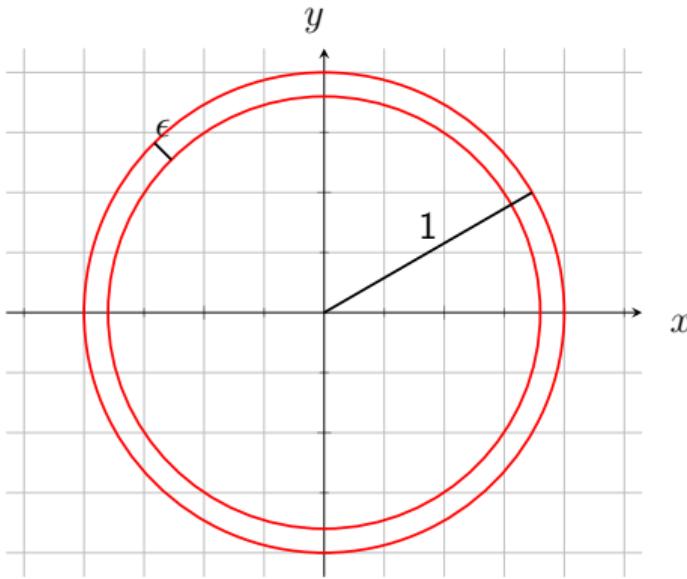
- What is ratio between the volume between the spheres and the volume of the large sphere?

$$\frac{V_D(1) - V_D(1 - \epsilon)}{V_D(1)} = \dots$$

- In D dimensions $V_D(r) = K_D r^D$

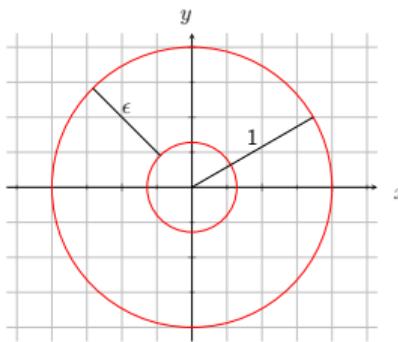
$$\dots = \frac{K_D 1^D - K_D (1 - \epsilon)^D}{K_D 1^D} \\ = 1 - (1 - \epsilon)^D$$

High dimensions and intuition

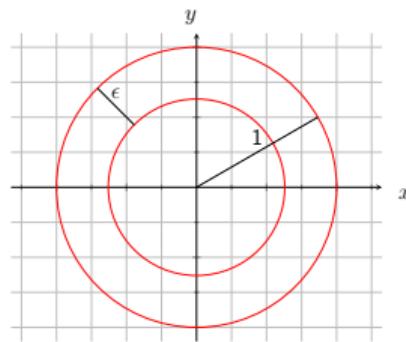


Where is 90% of the Volume?

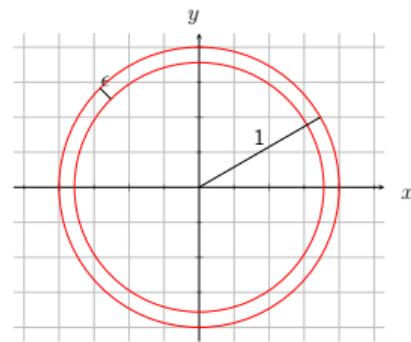
$$D = 2, \epsilon = 0.68$$



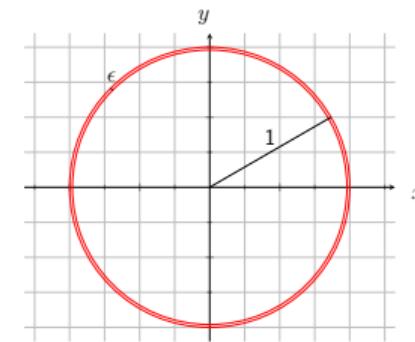
$$D = 5, \epsilon = 0.37$$



$$D = 20, \epsilon = 0.11$$



$$D = 100, \epsilon = 0.02$$



Example: Euclidean Distance

Two points in D dimensions:

$$\mathbf{a} = (a_1, a_2, \dots, a_D)$$

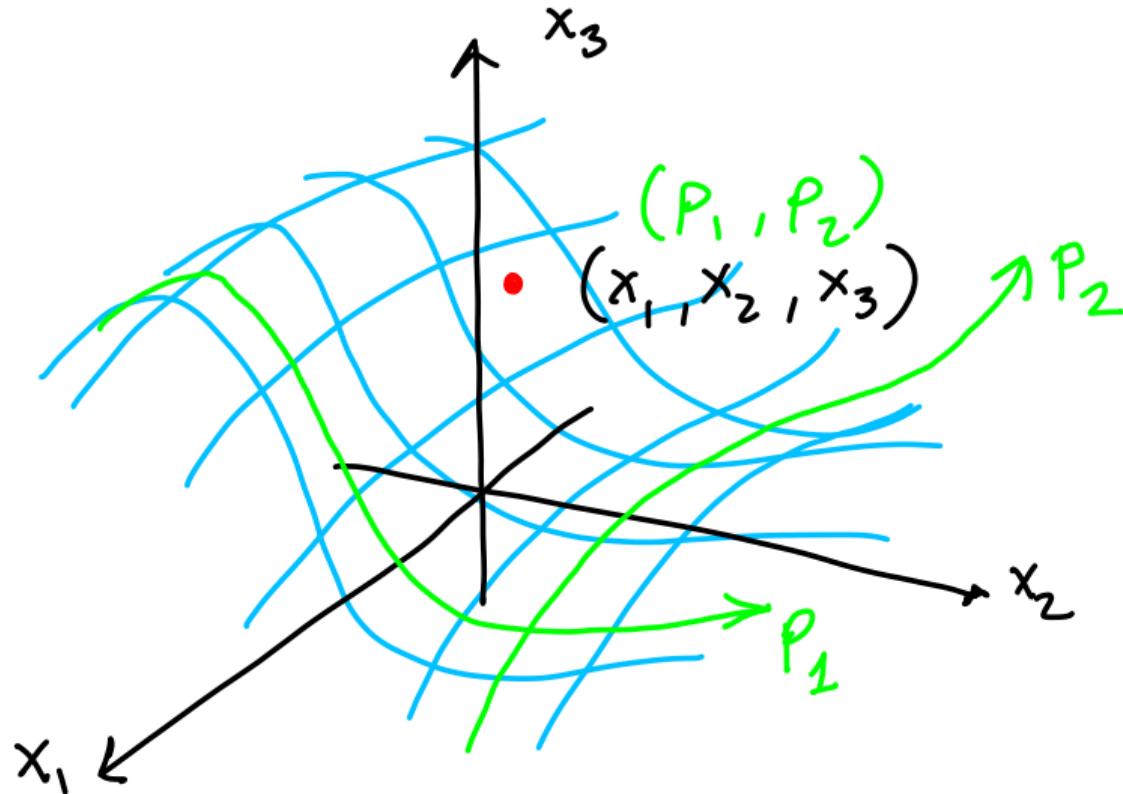
$$\mathbf{b} = (b_1, b_2, \dots, b_D)$$

Euclidean square distance

$$d^2(\mathbf{a}, \mathbf{b}) = (a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_D - b_D)^2$$

If $D = 1000$ it is enough that just a few coordinates differ.

Manifolds and Dimensionality Reduction



Outline

1 Curse of Dimensionality

- Manifolds and Dimensionality Reduction

2 Basis Functions

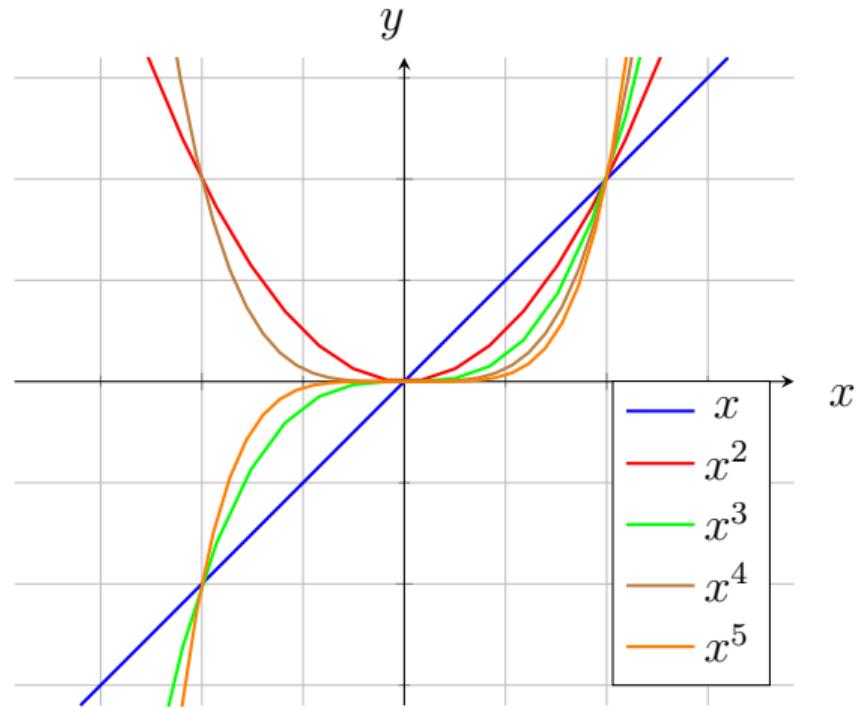
- Maximum Likelihood

3 Bias/Variance trade-off

4 Bayesian Model Selection

Linear Regression with Polynomials

$$y(x, \mathbf{w}) = w_0 + w_1x + \cdots + w_{M-1}x^{M-1}$$



Linear Regression with Basis Functions

$$\begin{aligned}y(\mathbf{x}, \mathbf{w}) &= w_0 + w_1\phi(\mathbf{x}) + \cdots + w_{M-1}\phi_{M-1}(\mathbf{x}) \\&= \sum_{j=0}^{M-1} w_j\phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})\end{aligned}$$

with:

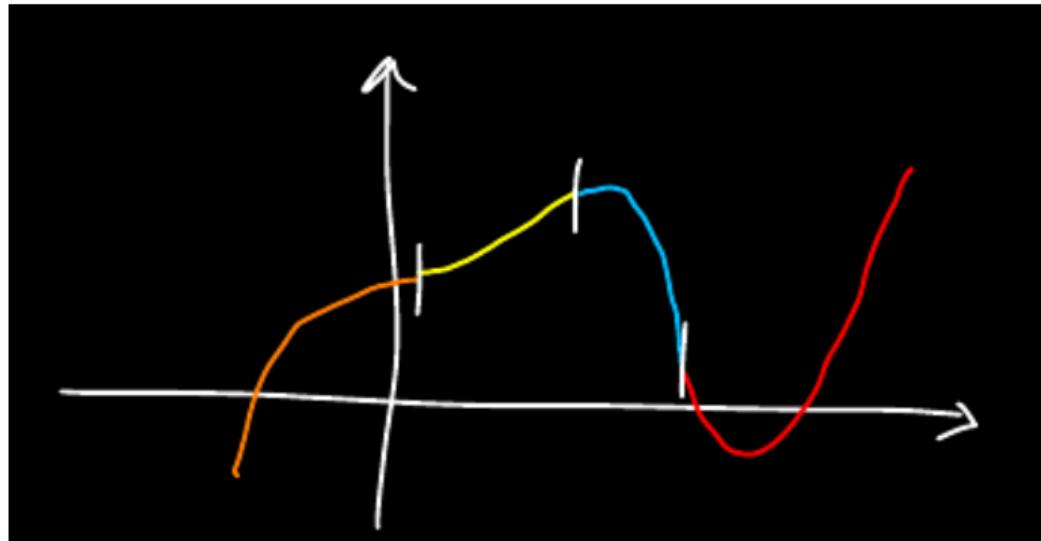
$$\phi_j(\mathbf{x}) : \mathbb{R}^D \rightarrow \mathbb{R}$$

$$\phi_0(\mathbf{x}) = 1, \forall \mathbf{x}$$

$$\boldsymbol{\phi}(\mathbf{x}) = [\phi_0(\mathbf{x}) \dots \phi_{M-1}(\mathbf{x})]^T$$

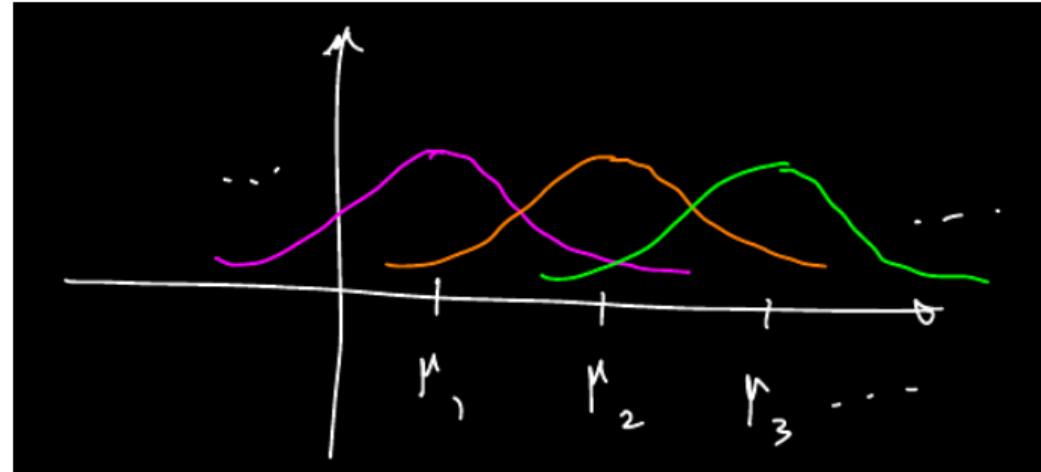
Example: Spline

- Piece-wise polynomial
- continuous up to first derivative



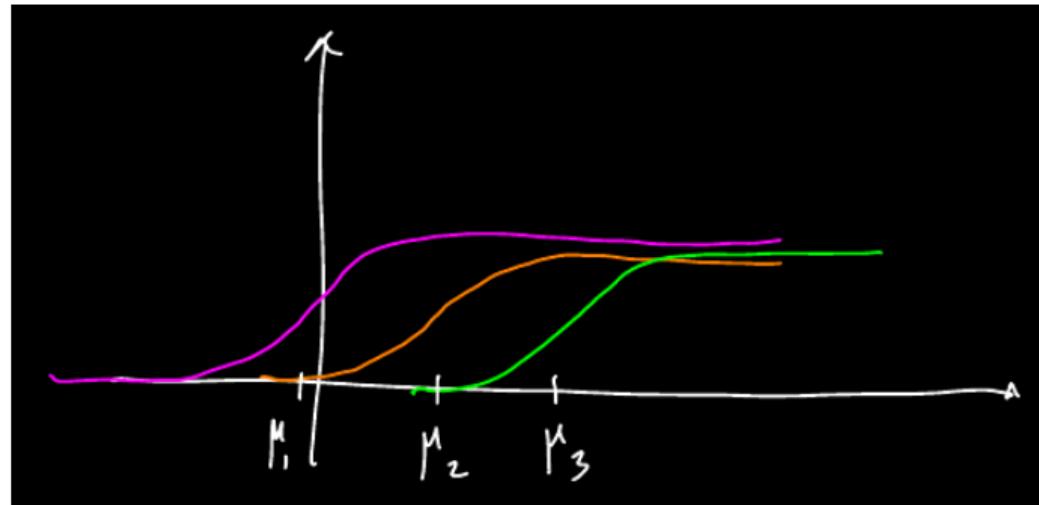
Example: Gaussian

$$\phi_j(x) = \exp \left\{ -\frac{(x - \mu_j)^2}{2\sigma^2} \right\}$$

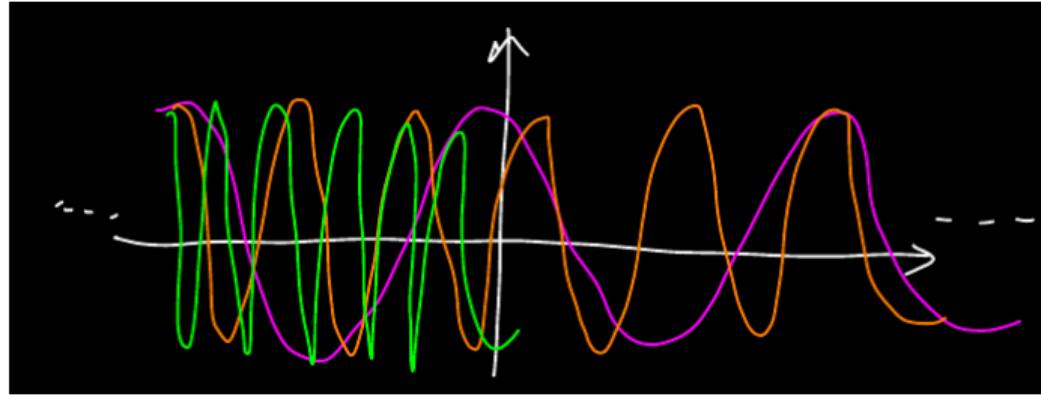


Example: Sigmoid

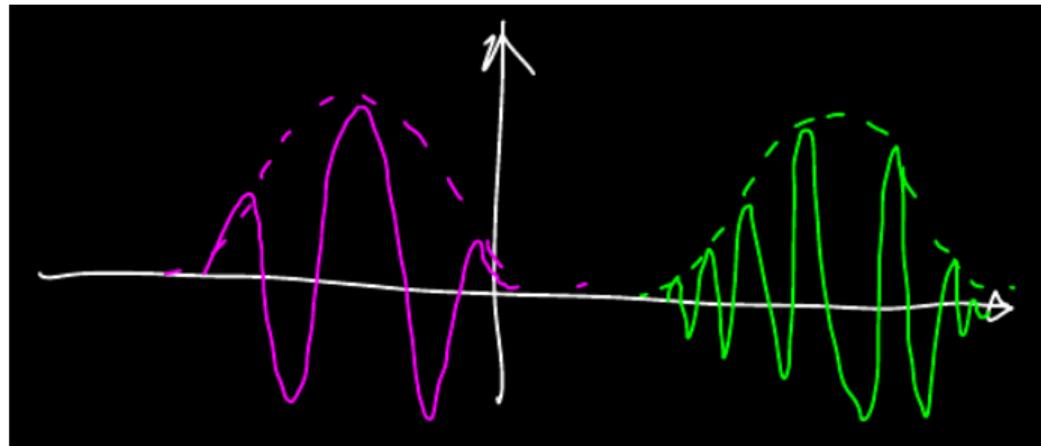
$$\phi_j(x) = \sigma\left(\frac{x - \mu_j}{s}\right), \text{ where } \sigma(a) = \frac{1}{1 + \exp(-a)}$$



Example: Fourier



Example: Wavelets



Basis Functions: Likelihood

Model:

$$\begin{aligned}t &= y(\mathbf{x}, \mathbf{w}) + \epsilon \\y(\mathbf{x}, \mathbf{w}) &= \mathbf{w}^T \phi(\mathbf{x}) \\p(t|\mathbf{x}, \mathbf{w}, \beta) &= \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1})\end{aligned}$$

Data:

$$\begin{aligned}\mathbf{X} &= \{\mathbf{x}_1, \dots, \mathbf{x}_N\} \\\mathbf{t} &= \{t_1, \dots, t_N\}\end{aligned}$$

Likelihood:

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^N \mathcal{N}(t_n | \mathbf{w}^T \phi(\mathbf{x}_n), \beta^{-1})$$

Basis Functions: Maximum Likelihood Solution

$$\mathbf{w}_{\text{ML}} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{t},$$

by defining the **design matrix**

$$\Phi = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \dots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \dots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \dots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}$$

Basis Functions: Maximum Likelihood Solution

Equivalent to the linear regression solution in $\mathbf{x} \in \mathbb{R}^D$:

$$\mathbf{w}_{\text{ML}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{t},$$

with

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{ND} \end{pmatrix}.$$

Basis Functions: Maximum Likelihood Solution

Equivalent to the linear regression solution in $\mathbf{x} \in \mathbb{R}^D$:

$$\mathbf{w}_{\text{ML}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{t},$$

with

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{ND} \end{pmatrix}.$$

The basis functions $\phi_j(\mathbf{x}_N)$ act as feature extraction!

Basis Functions

- equivalent to linear models using Φ instead of \mathbf{X}
- all other results hold:
 - overfitting of ML
 - regularization (MAP)
 - Bayesian models

Outline

1 Curse of Dimensionality

- Manifolds and Dimensionality Reduction

2 Basis Functions

- Maximum Likelihood

3 Bias/Variance trade-off

4 Bayesian Model Selection

Bias/Variance Decomposition

- Maximum Likelihood (least squares) leads to overfitting
- limiting the complexity of the model risks to miss trends in data
- regularization helps, but we need to find value for λ

Decision theory

Under L^2 loss, best decision is conditional expectation

$$h(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}] = \int t p(t|\mathbf{x}) dt,$$

where $p(t|\mathbf{x})$ is the true (unknown) distribution

Expected Loss (theoretical distribution)

If we predict the answer with $y(\mathbf{x})$, the expected (square) loss is:

$$\begin{aligned}\mathbb{E}[L] &= \iint L(t, y(\mathbf{x})) p(\mathbf{x}, t) d\mathbf{x} dt = \\ &= \iint \{y(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt \quad (\text{square loss})\end{aligned}$$

We can compare this to the theoretically optimal estimation $h(\mathbf{x})$

Expected Loss (theoretical distribution)

$$\begin{aligned}\mathbb{E}[L] &= \dots \\ &= \int \{y(\mathbf{x}) - h(\mathbf{x})\}^2 p(\mathbf{x}) d\mathbf{x} + \quad \leftarrow \text{sub-optimal inference} \\ &\quad + \iint \{h(x) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt \quad \leftarrow \text{intrinsic noise}\end{aligned}$$

Expected Loss from Data

- we do not know $p(\mathbf{x}, t)$
- we imagine we have many data sets drawn from $p(\mathbf{x}, t)$
- for every data set \mathcal{D} we obtain:
 - a model $y(\mathbf{x}, \mathcal{D})$
 - an expected loss $\mathbb{E}_{\mathcal{D}}[L]$
- then we can average over data sets.

Bias and Variance (single input value)

For a single value of \mathbf{x}

$$\begin{aligned}\mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}) - h(\mathbf{x})\}^2] &= \dots \\ &= \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}, \mathcal{D})] - h(\mathbf{x})\}^2 + && (\text{bias})^2 \\ &\quad + \mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}, \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}, \mathcal{D})]\}^2] && \text{variance}\end{aligned}$$

Bias and Variance (general case)

Integrating over all possible values of \mathbf{x} :

$$\text{expected loss} = (\text{bias})^2 + \text{variance} + \text{noise}$$

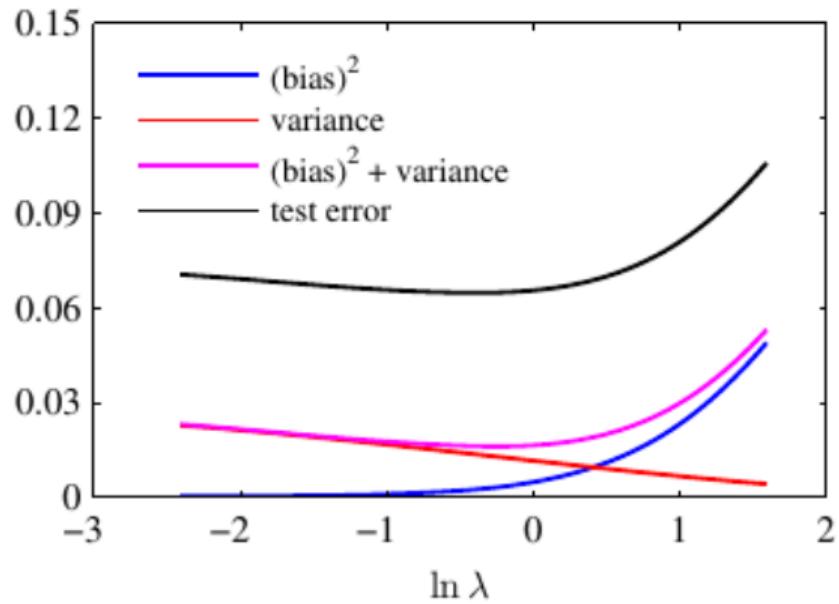
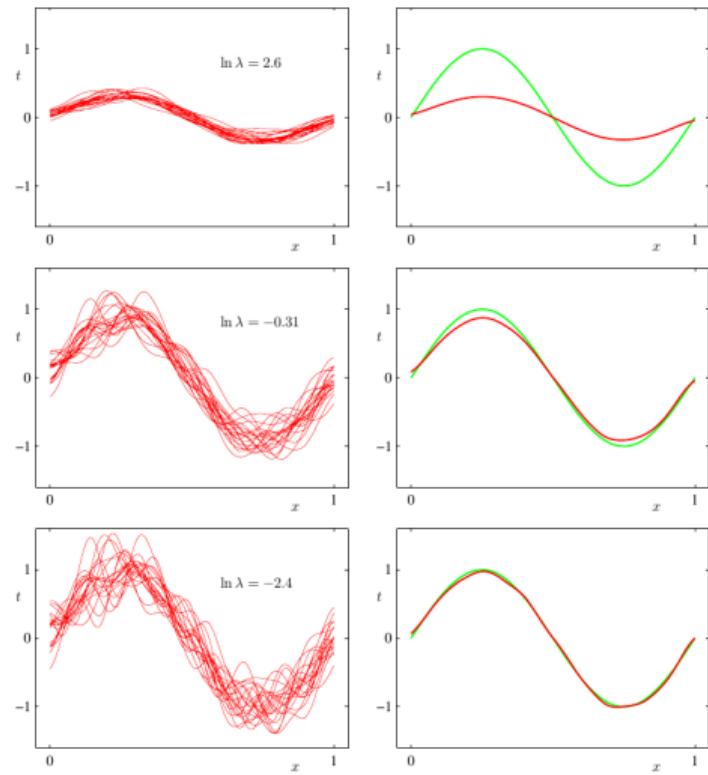
Where:

$$(\text{bias})^2 = \int \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}, \mathcal{D})] - h(\mathbf{x})\}^2 p(\mathbf{x}) d\mathbf{x}$$

$$\text{variance} = \int \mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}, \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}, \mathcal{D})]\}^2] p(\mathbf{x}) d\mathbf{x}$$

$$(\text{noise}) = \int \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt$$

Bias/Variance Example



Outline

1 Curse of Dimensionality

- Manifolds and Dimensionality Reduction

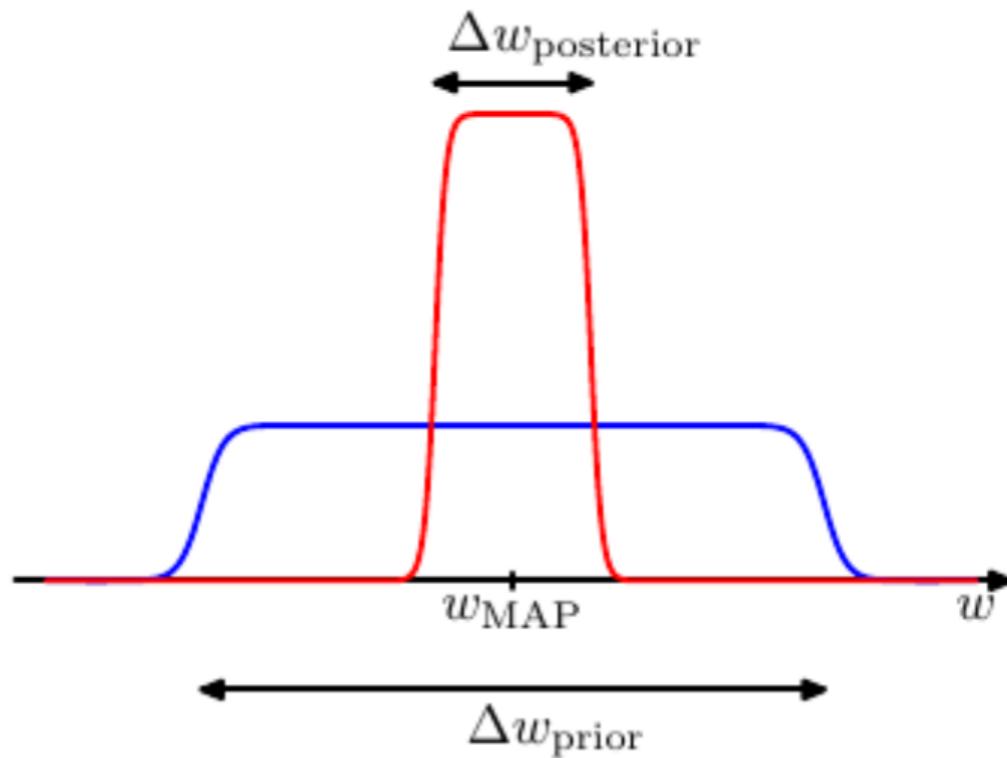
2 Basis Functions

- Maximum Likelihood

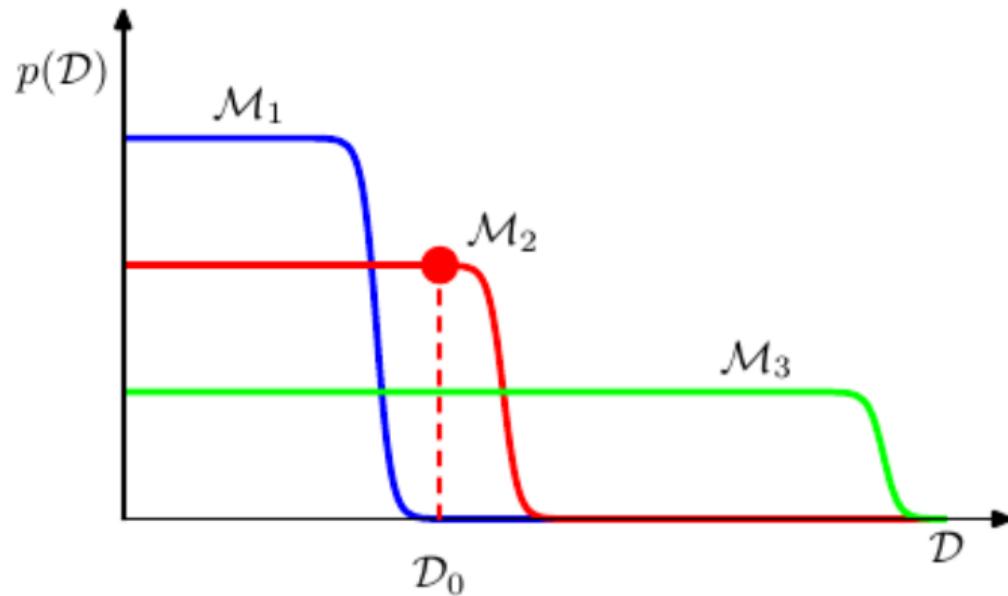
3 Bias/Variance trade-off

4 Bayesian Model Selection

Bayesian Model Evidence



Bayesian Model Selection



Limitation of Linear Models

- basis functions $\phi_J(\mathbf{x})$ are fixed (not trained)
- the number of basis functions grow with dimensionality of input \mathbf{x}

Solution: exploit manifold

- dimesionality reduction methods
- support vector machines
- neural networks

Linear Models for Classification

TTT4185 Machine Learning for Signal Processing

Giampiero Salvi

Department of Electronic Systems
NTNU

HT2021

Outline

1 Discriminant Functions

2 Learning Parameters

- Least-squares
- Fisher's discriminant
- Perceptron

3 Probabilistic Models

- Generative Approach
- Discriminative Approach

Outline

1 Discriminant Functions

2 Learning Parameters

- Least-squares
- Fisher's discriminant
- Perceptron

3 Probabilistic Models

- Generative Approach
- Discriminative Approach

Discriminant Functions

Classification

$$\mathbf{x} \in \mathbb{R}^M, \quad t \in \{\mathcal{C}_1, \dots, \mathcal{C}_K\}$$

Then

$$y(\mathbf{x}) : \mathbb{R}^M \rightarrow \{\mathcal{C}_1, \dots, \mathcal{C}_K\}$$

Decision regions $\mathcal{R}_1, \dots, \mathcal{R}_K \in \mathbb{R}^M$

Decision Boundaries:

- linear models: $\mathbf{x} \in \mathbb{R}^{M-1}$ ($M - 1$ hyperplanes)

Discriminant Functions

Linear in parameters and inputs

$$y(\mathbf{x}, \mathbf{w}) = f(\mathbf{w}^T \mathbf{x} + w_0)$$

$f(\cdot)$ is called

- activation function (machine learning)
- inverse link function (statistics)

Decision surfaces:

$$y(\mathbf{x}, \mathbf{w}) = \text{constant} \Leftrightarrow \mathbf{w}^T \mathbf{x} + w_0 = \text{constant}$$

Even if $f(\cdot)$ is non-linear.

Example: Two Classes

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

$$y(\mathbf{x}) \geq 0 \rightarrow \mathbf{x} \in \mathcal{R}_1 \quad (\mathcal{C}_1)$$

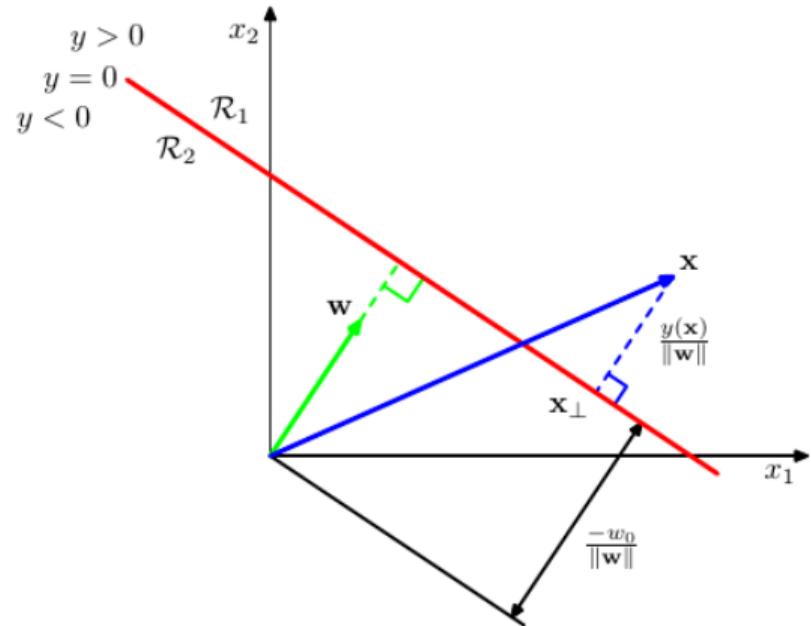
$$y(\mathbf{x}) < 0 \rightarrow \mathbf{x} \in \mathcal{R}_2 \quad (\mathcal{C}_2)$$

(To be precise: $y(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} + w_0)$)

Equal values perpendicular to \mathbf{w} :

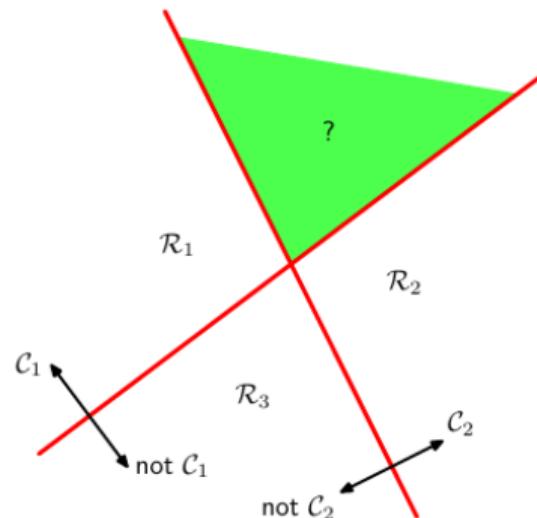
$$\begin{aligned} y(\mathbf{x}_A) = y(\mathbf{x}_B) &\Rightarrow \mathbf{w}^T(\mathbf{x}_A - \mathbf{x}_B) = 0 \\ &\Leftrightarrow \mathbf{w} \perp \text{boundary} \end{aligned}$$

Decision boundary for $\mathbf{w}^T \mathbf{x} = -w_0$

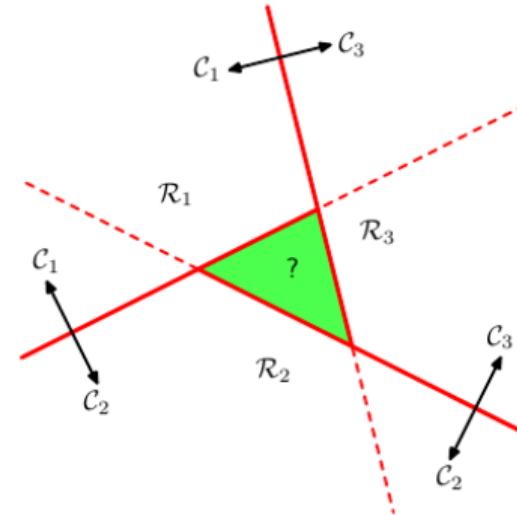


Multiple Classes

Problem combining 2-class decision functions:



one-versus-the-rest



one-versus-one

Multiple Classes

k -class discriminant

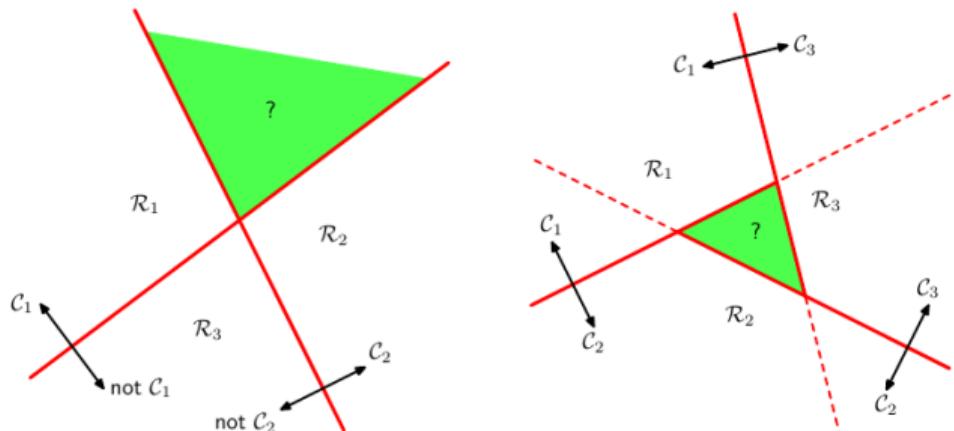
$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}$$

$$\mathbf{x} \in C_k \Leftrightarrow y_k(\mathbf{x}) > y_j(\mathbf{x}) \forall j \neq k$$

Decision boundaries:

$$y_k(\mathbf{x}) = y_j(\mathbf{x}) \Rightarrow$$

$$(\mathbf{w}_k - \mathbf{w}_j)^T (\mathbf{x}) + (w_{k0} - w_{j0}) = 0$$



Simplified Notation and Basis Functions

Simplified notation

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \mathbf{x}$$

where

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_M \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_M \end{bmatrix}$$

All results are equivalent if we use basis functions:

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x})$$

Outline

1 Discriminant Functions

2 Learning Parameters

- Least-squares
- Fisher's discriminant
- Perceptron

3 Probabilistic Models

- Generative Approach
- Discriminative Approach

Least Squares

Data $\mathcal{D} = \{(\mathbf{x}_1, t_1), \dots, (\mathbf{x}_N, t_n)\}$

Minimize:

$$E_{\mathcal{D}} = \frac{1}{2} \sum_{n=1}^N (y(\mathbf{x}_n, \mathbf{w}) - t_n)^2$$

Notation:

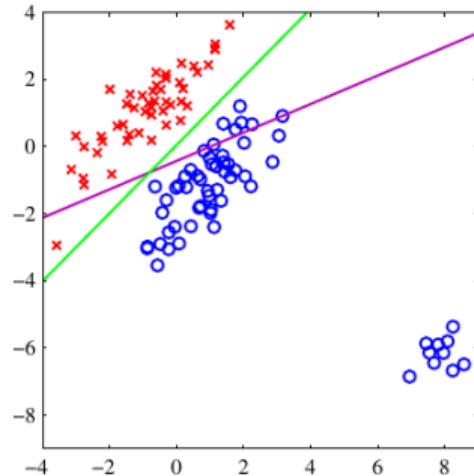
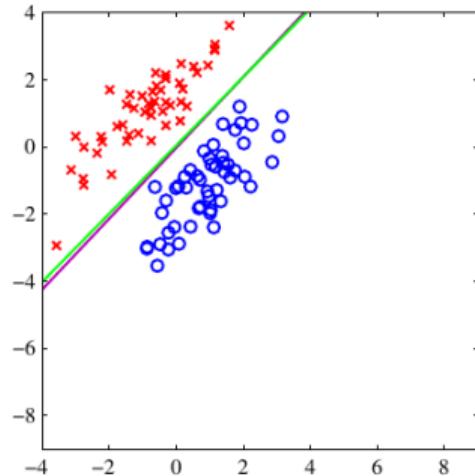
1-of- K binary encoding for t :

$$t_n = (0, 0, 0, 1, 0, \dots, 0)$$

Least Squares

Problems

- least-squares suffers from outliers (figure)
- corresponds to ML in the assumption of Gaussian conditional distributions



Fisher's Linear Discriminant

- linear classification viewed as dimensionality reduction
- select projection that maximizes class separation

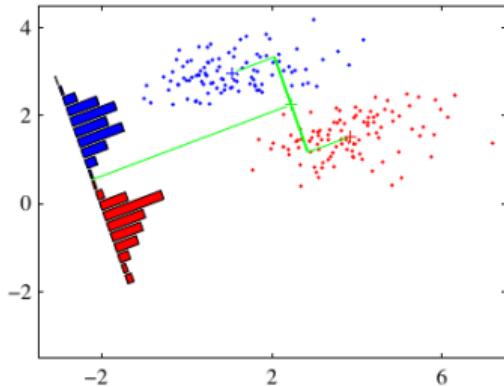
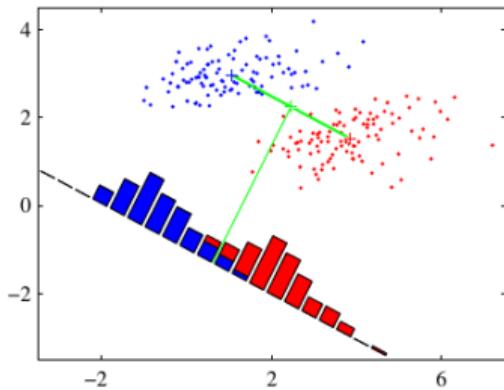
Example: 2 classes, C_1, N_1, C_2, N_2

$$\mathbf{y} = \mathbf{w}^T \mathbf{x}$$

Measure of separation (to be maximized)

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$$

- S_B : between-class covariance
- S_W : within-class covariance



Fisher's Linear Discriminant

- linear classification viewed as dimensionality reduction
- select projection that maximizes class separation

Example: 2 classes, C_1, N_1, C_2, N_2

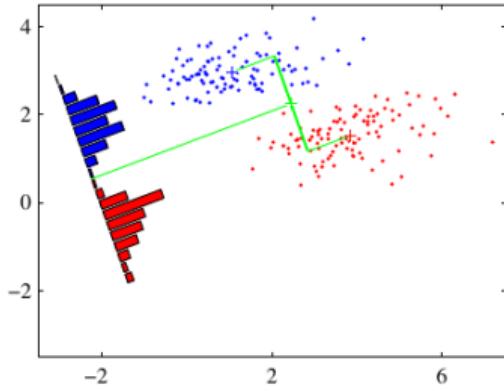
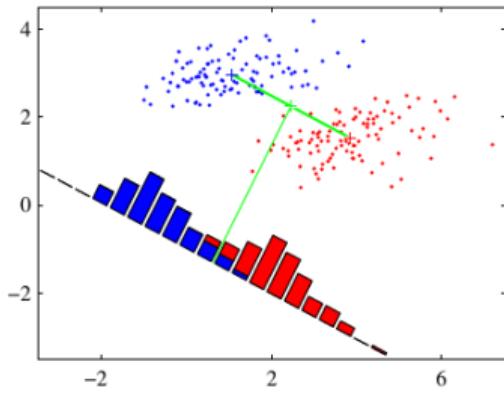
$$\mathbf{y} = \mathbf{w}^T \mathbf{x}$$

Measure of separation (to be maximized)

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$$

Differentiating $J(\mathbf{w})$:

$$\mathbf{w} \propto S_W^{-1} (\mathbf{m}_2 - \mathbf{m}_1)$$



Perceptron (Rosenblatt 1962)

$$y(\mathbf{x}) = f(\mathbf{w}^T \mathbf{x}) \quad f(a) = \begin{cases} +1 & \text{if } a \geq 0 \\ -1 & \text{if } a < 0 \end{cases}$$

- in the book $\phi(\mathbf{x})$ instead of \mathbf{x}
- encode t with $+1$ for \mathcal{C}_1 and -1 for \mathcal{C}_2
- classify \mathbf{x} according to sign of $\mathbf{w}^T \mathbf{x}$
- ideally for the training data $(\mathbf{w}^T \mathbf{x}_n)t_n > 0, \quad \forall n$

Error function

$$E_p(\mathbf{w}) = - \sum_{n \in \mathcal{M}} \mathbf{w}^T \mathbf{x}_n t_n \quad \mathcal{M} = \text{miss-classified examples}$$

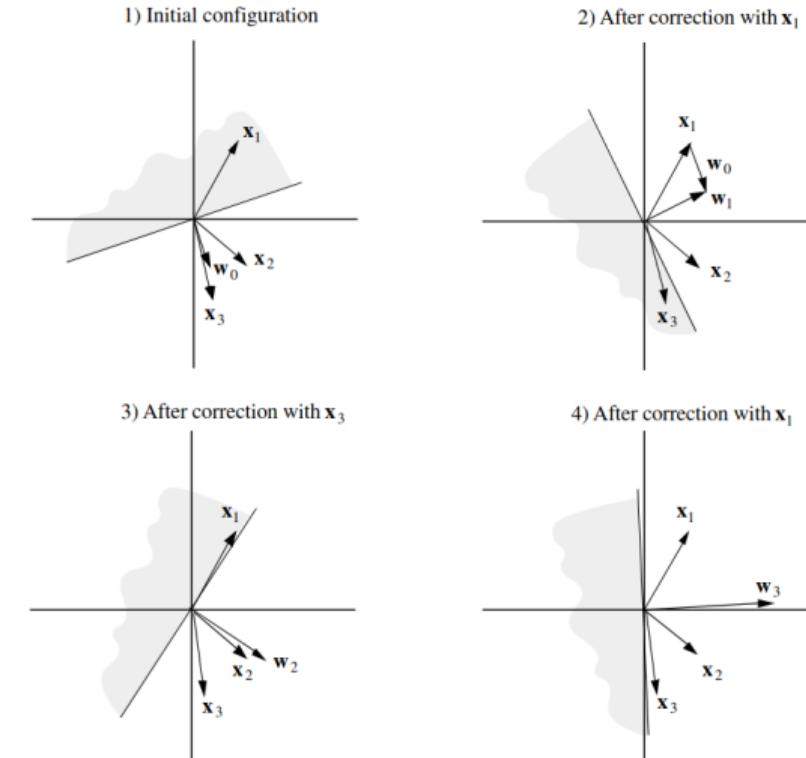
Stochastic Gradient descent

Parameter update for every miss-classified data point $(x_n, t_n) \in \mathcal{M}$

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla E_p(\mathbf{w}) = \mathbf{w}^{(t)} + \eta \mathbf{x}_n t$$

η = learning rate = 1

Not guaranteed to reduce error at every update.



Perception: Convergence Theorem

Convergence Theorem

If there exists a solution (linearly separable data)
then the perception will find it in a finite number of steps.

Problems:

- the number of steps could be substantial
- no way to know if the problem is linearly separable before convergence
- there usually are several solutions
- if not linearly separable, it never converges

Outline

1 Discriminant Functions

2 Learning Parameters

- Least-squares
- Fisher's discriminant
- Perceptron

3 Probabilistic Models

- Generative Approach
- Discriminative Approach

Probabilistic Models

2 classes:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} = \frac{1}{1 + \exp(-a)} \quad (\text{sigmoid})$$

where

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \quad \text{log odds}$$

K classes:

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)} \quad (\text{softmax})$$

where

$$a_k = \ln p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)$$

Generative Approach

Given data $\mathcal{D} = \{(\mathbf{x}_1, t_1), \dots, (\mathbf{x}_N, t_N)\}$ maximize log likelihood:

$$\ln \mathcal{L} = \sum_{n=1}^N p(\mathbf{x}_n, t_n | \theta) = \sum_{k=1}^K \sum_{n: \{t_n=k\}} \left[\ln p(\mathbf{x} | \mathcal{C}_k, \theta_k) + \ln p(\mathcal{C}_k | \theta_k) \right]$$

with $\theta = \{\theta_1, \dots, \theta_K\}$, and θ_k are the class dependent model parameters.

Example:

- Categorical priors: $p(\mathcal{C}_k | \theta_k) = \pi_k$, with $\sum_k \pi_k = 1$
- Gaussian class conditional likelihoods $p(\mathbf{x} | \mathcal{C}_k, \theta_k) = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
- $\theta = \underbrace{\{\pi_1, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \dots, \pi_K, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_K\}}_{\theta_1} \underbrace{\{\pi_1, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \dots, \pi_K, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_K\}}_{\theta_K}$

Class-conditional optimization

Given data $\mathcal{D} = \{(\mathbf{x}_1, t_1), \dots, (\mathbf{x}_N, t_N)\}$ maximize log likelihood:

$$\ln \mathcal{L} = \sum_{n=1}^N p(\mathbf{x}_n, t_n | \theta) = \sum_{k=1}^K \sum_{n: \{t_n=k\}} \left[\ln p(\mathbf{x} | \mathcal{C}_k, \theta_k) + \ln p(\mathcal{C}_k | \theta_k) \right]$$

Differentiating w.r.t. θ_k , only contributions from (\mathbf{x}_n, t_n) for which $t_n = k$!

- ① split data into K subset according to class label t_n
- ② optimize $p(\mathbf{x} | \mathcal{C}_k, \theta_k)$ and $p(\mathcal{C}_k | \theta_k)$ independently using ML

Example: Gaussian class-conditional likelihoods

Categorical priors

$$p(\mathcal{C}_k | \theta_k) = \pi_k, \text{ with } \sum_k \pi_k = 1$$

Maximum likelihood: $\pi_k = \frac{1}{N} \sum_{n:\{t_n=k\}} 1 = \frac{N_k}{N}$

Gaussian class conditional likelihoods

$$p(\mathbf{x} | \mathcal{C}_k, \theta_k) = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Maximum likelihood:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n:\{t_n=k\}} \mathbf{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n:\{t_n=k\}} (\mathbf{x}_n - \boldsymbol{\mu}_k)^T (\mathbf{x}_n - \boldsymbol{\mu}_k)$$

Special Case: Gaussians with Equal Covariance

2-class problem with $\Sigma_1 = \Sigma_2 = \Sigma$

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{1}{1 + \exp(-a)} = \sigma(a) \quad \text{with } a = \ln \frac{\pi \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_1, \Sigma)}{(1-\pi)\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_2, \Sigma)} \quad \text{log odds}$$

Equal normalization term in $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_1, \Sigma)$ and $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_2, \Sigma)$, expanding and simplifying:

$$a = \mathbf{w}^T \mathbf{x} + w_0, \quad \text{with}$$

$$\mathbf{w} = \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2), \quad \text{and}$$

$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^T \Sigma^{-1} \boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T \Sigma^{-1} \boldsymbol{\mu}_2 + \ln \frac{\pi}{1-\pi}$$

Linear classifier!!

Discriminative Approach: Logistic Regression

- 2-class problem: posterior as sigmoid
- use ML to maximize $P(\mathcal{C}_k|\mathbf{x})$ directly.

$$P(\mathcal{C}_k|\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}$$

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^N \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}$$

$$\nabla E(\mathbf{w}) = \sum_{n=1}^N (y_n - t_n) x_n$$

- non-linear in $\mathbf{w} \Rightarrow$ no closed form solution
- but $E(\mathbf{w})$ is convex \Rightarrow single global minimum
- can be optimized with iterative algorithm

Number of Parameters, 2-class problem, M dimensional \mathbf{x}

Generative model

(equal covariance matrices):

Parameter	Degrees of freedom
μ_1	M
μ_2	M
Σ	$\frac{M(M+1)}{2}$
$P(C_1)$	1
Total	$\frac{M(M+5)}{2} + 1$

Discriminative model
(logistic regression)

Parameter	Degrees of freedom
\mathbf{w}	$M + 1$
Total	$M + 1$

Kernel Methods

TTT4185 Machine Learning for Signal Processing

Giampiero Salvi

Department of Electronic Systems
NTNU

HT2021

Outline

1 Kernel Methods

- Memory-Based Methods
- Dual Representation
- Constructing Kernels
- Gaussian Processes

Outline

1 Kernel Methods

- Memory-Based Methods
- Dual Representation
- Constructing Kernels
- Gaussian Processes

Parametric Models

- ① define a parametric model, for example

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \mathbf{x}$$

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 \dots$$

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x})$$

$$p(\mathbf{x}, \mathcal{C}_k) = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

...

- ② fit model parameters to data \mathcal{D} (ML, MAP)
- ③ throw away the data and use the model for predictions

Special Case: Full Bayesian Methods

$$p(t|\mathcal{D}) = \int_{\theta} p(t|\theta, \mathcal{D})p(\theta|\mathcal{D})d\theta$$

- marginalize out the value of the parameters
- but still define an underlying parametric model

Memory-Based Methods

- use the training data \mathcal{D} directly to make predictions
- example: k -nearest neighbours
- we need a metric (e.g. Euclidean distance) to determine similarity

Dual Representation: Kernel

- many linear parametric models can be recast
- representation of observations \mathbf{x}_n is not important singularly
- what matters is pair-wise relationship between points $k(x, x')$ (kernel)
- as long as we can compute $k(., .)$ the dimensionality of the input does not matter

Example: Linear Regression

Linear model:

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x}) \quad \text{with} \quad \phi(\mathbf{x}) = [\phi_1(\mathbf{x}) \quad \dots \quad \phi_M(\mathbf{x})]^T$$

MAP estimation with zero-mean isotropic prior:

$$J(\mathbf{w}) = \underbrace{\frac{1}{2} \sum_{n=1}^N \{ \mathbf{w}^T \phi(\mathbf{x}_n) - t_n \}^2}_{\text{least squares}} + \underbrace{\frac{\lambda}{2} \mathbf{w}^T \mathbf{w}}_{\text{regularization}}$$

Differentiating:

$$\nabla J(\mathbf{w}) = 0 \quad \Rightarrow \quad \mathbf{w} = -\frac{1}{\lambda} \sum_{n=1}^N \{ \mathbf{w}^T \phi(\mathbf{x}_n) - t_n \} \phi(\mathbf{x}_n)$$

Linear Prediction: Dual representation

$$\mathbf{w} = -\frac{1}{\lambda} \sum_{n=1}^N \{\mathbf{w}^T \phi(\mathbf{x}_n) - t_n\} \phi(\mathbf{x}_n) = \sum_{n=1}^N a_n \phi(\mathbf{x}_n) = \Phi^T \mathbf{a}$$

Where we have defined:

$$\mathbf{a} = [a_1, \dots, a_N]^T, \quad a_n = -\frac{1}{\lambda} \{\mathbf{w}^T \phi(\mathbf{x}_n) - t_n\}$$
$$\Phi = \begin{bmatrix} \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \dots & \phi_M(\mathbf{x}_1) \\ \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \dots & \phi_M(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(\mathbf{x}_N) & \phi_2(\mathbf{x}_N) & \dots & \phi_M(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} \phi(\mathbf{x}_1)^T \\ \phi(\mathbf{x}_2)^T \\ \vdots \\ \phi(\mathbf{x}_N)^T \end{bmatrix} \quad \text{design matrix}$$

Linear Prediction: Dual representation Properties

Parameter space:

$$\mathbf{w} = [w_1, \dots, w_M]^T \in \mathbb{R}^M \quad \text{same dimensionality as the features}$$

Dual space:

$$\mathbf{a} = [a_1, \dots, a_N]^T \in \mathbb{R}^N \quad \text{number of data points}$$

$$a_n = -\frac{1}{\lambda} \{\mathbf{w}^T \phi(\mathbf{x}_n) - t_n\} \quad \text{is the prediction error for the } n\text{th data point}$$

Regularized sum of squares: Dual representation

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^T \Phi \Phi^T \Phi \Phi^T \mathbf{a} - \mathbf{a}^T \Phi \Phi \mathbf{t} + \frac{1}{2}\mathbf{t}^T \mathbf{t} + \frac{\lambda}{2}\mathbf{a}^T \Phi \Phi^T \mathbf{a}$$

Gram matrix

$$\mathbf{K} = \Phi \Phi^T \quad \text{that is } K_{nm} = \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m)$$

Then

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^T \mathbf{K} \mathbf{K} \mathbf{a} - \mathbf{a}^T \mathbf{K} \mathbf{t} + \frac{1}{2}\mathbf{t}^T \mathbf{t} + \frac{\lambda}{2}\mathbf{a}^T \mathbf{K} \mathbf{a}$$

Solving for $\nabla J(\mathbf{a}) = 0$:

$$\mathbf{a} = (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t}$$

Linear prediction model: Dual representation

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x}) = \mathbf{a}^T \Phi \phi(\mathbf{x}) = k(\mathbf{x})^T (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t}$$

where

$$k(\mathbf{x})^T = [k(\mathbf{x}_1, \mathbf{x}) \quad k(\mathbf{x}_2, \mathbf{x}) \quad \dots \quad k(\mathbf{x}_N, \mathbf{x})]$$

- to express $y(\mathbf{x})$ in \mathbf{w} we needed to invert an $M \times M$ matrix
- now we need to invert an $N \times N$ matrix
- usually $N \gg M$!

Dual representation: advantages

- no need to define $\phi(\mathbf{x})$ explicitly
- it can work even with very high dimensional features (even ∞)
- no need to use all the N training points (support vector machines)

Constructing Kernels

Indirect approach (use basis functions

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}') = \sum_{i=1}^M \phi_i(\mathbf{x}) \phi_i(\mathbf{x}')$$

Direct approach:

- use scalar product in some space

Example: Quadratic Kernel

Direct approach: use scalar product in some space

Example: $\mathbf{x}, \mathbf{z} \in \mathbb{R}^2$,

$$\begin{aligned} k(\mathbf{x}, \mathbf{z}) &= (\mathbf{x}^T \mathbf{z})^2 = (x_1 z_1 + x_2 z_2)^2 = \\ &= x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2 = \\ &= (x_1^2, \sqrt{2}x_1 x_2, x_2^2)(z_1^2, \sqrt{2}z_1 z_2, z_2^2)^T = \\ &= \phi(x)^T \phi(z) \end{aligned}$$

therefore:

$$\phi(x) = (x_1^2, \sqrt{2}x_1 x_2, x_2^2)^T \in \mathbb{R}^3$$

Quadratic Kernel Remarks

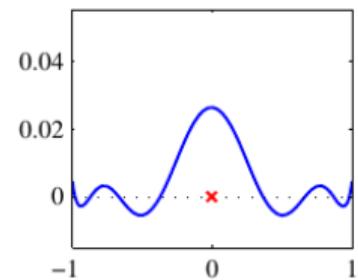
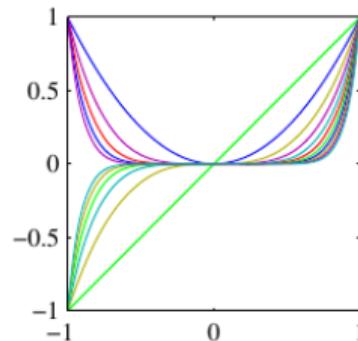
- input space is 2 dimensional ($\mathbf{x} \in \mathbb{R}^2$)
- feature space is 3 dimensional ($\phi(\mathbf{x}) \in \mathbb{R}^3$)
- all we need for inference is $k(\mathbf{x}, \mathbf{z}) \Rightarrow$ we still work in 2 dimensions
- ... but have the advantages of 3 dimensions
- more in general for $\mathbf{x} \in \mathbb{R}^D$, the features are $\frac{D(D+1)}{2}$ dimensional

Practical Example

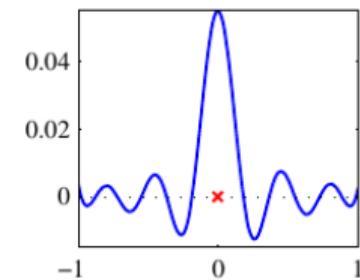
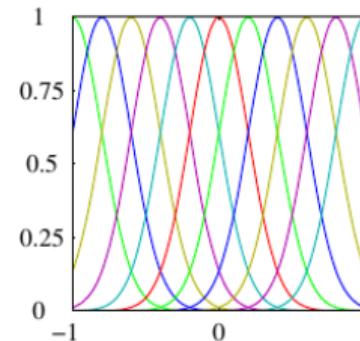
- We need to do regression based on 1000 training images
- each image is 32×32 pixels (1024 dimensions)
- the quadratic kernel gives us features in 524.800 dimensions
- but we only need to invert a 1000×1000 matrix.

Basis Functions and Kernels

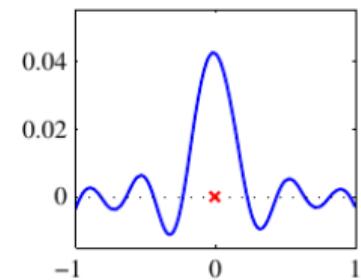
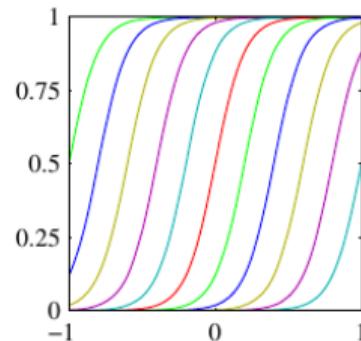
One dimensional $x \in \mathbb{R}$, kernel $k(x, x')$ for $x' = 0$



polynomial



Gaussian



sigmoid

Kernels as Building Blocks

If $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$ are valid kernels, the following are also valid:

$$k(\mathbf{x}, \mathbf{x}') = c k_1(\mathbf{x}, \mathbf{x}')$$
 scaled version, $c > 0$

$$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}')$$
 any function $f(\cdot)$

$$k(\mathbf{x}, \mathbf{x}') = q(k_1(\mathbf{x}, \mathbf{x}'))$$
 $q(\cdot)$ polynomial n.n. coeff.

$$k(\mathbf{x}, \mathbf{x}') = \exp(k_1(\mathbf{x}, \mathbf{x}'))$$
 exponential

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$$
 sum

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}')$$
 product

$$k(\mathbf{x}, \mathbf{x}') = k_1(\phi(\mathbf{x}), \phi(\mathbf{x}'))$$
 change of basis

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{A} \mathbf{x}'$$
 \mathbf{A} symm. positive semidef.

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b)$$
 subspace sum

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a)k_b(\mathbf{x}_b, \mathbf{x}'_b)$$
 subspace product

Polynomial Kernels

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}')^2 \quad \text{only square terms}$$

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + c)^2, c > 0 \quad \text{also linear terms}$$

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}')^M \quad \text{polynomial order } M$$

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + c)^M, c > 0 \quad \text{also linear terms}$$

Gaussian Kernel

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$

- not interpreted as probability distribution
- check validity

$$\|\mathbf{x} - \mathbf{x}'\|^2 = \mathbf{x}^T \mathbf{x} + (\mathbf{x}')^T \mathbf{x}' - 2\mathbf{x}^T \mathbf{x}'$$

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(\frac{\mathbf{x}^T \mathbf{x}}{2\sigma^2}\right) \exp\left(\frac{(\mathbf{x}')^T \mathbf{x}'}{2\sigma^2}\right) \exp\left(\frac{-2\mathbf{x}^T \mathbf{x}'}{2\sigma^2}\right)$$

- features $\phi(\mathbf{x})$ associated with Gaussian kernel have infinite dimensionality!!

Generality of Kernels

- we are not interested in the representation for x
- we can define kernels on discrete objects (non-vectorial spaces)

Example 1: A_1 and A_2 are subsets of a finite set

$$k(A_1, A_2) = 2^{|A_1 \cap A_2|}$$

Example 2: strings (variable length sequences of discrete symbols)

- text classification, spam filters
- gene analysis

Kernels over Generative Models

Given a generative model $p(\mathbf{x})$, define kernel as

$$k(\mathbf{x}, \mathbf{x}') = p(\mathbf{x})p(\mathbf{x}')$$
 simple distribution

$$k(\mathbf{x}, \mathbf{x}') = \sum_i p(\mathbf{x}|i)p(\mathbf{x}'|i)p(i)$$
 mixture of distributions

- combining discriminative and generative ideas

Example (Hidden Markov Model)

$\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_L\}$ observation sequence

$\mathbf{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_L\}$ latent variable

$k(\mathbf{X}, \mathbf{X}') = \sum_{\mathbf{Z}} p(\mathbf{X}|\mathbf{Z})p(\mathbf{X}'|\mathbf{Z})p(\mathbf{Z})$ sequences of equal length

- can be extended to sequences of variable length

Gaussian Processes

Note: this is big topic, we only mention it here

Standard regression model:

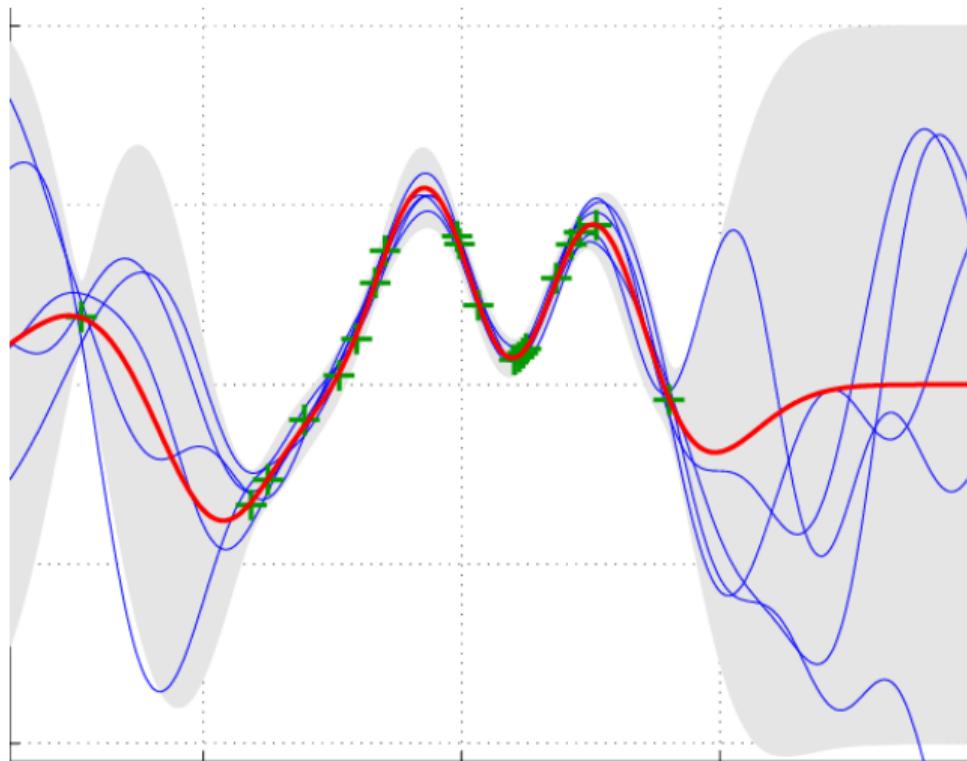
$$\begin{aligned}y(\mathbf{x}, \mathbf{w}) &= \mathbf{w}^T \phi(\mathbf{x}) && M \text{ basis functions } \phi_i(\mathbf{x}) \\p(\mathbf{w}) &= \mathcal{N}(\mathbf{w} | \mathbf{0}, \alpha^{-1} \mathbf{I}) && \text{parameter prior}\end{aligned}$$

Gaussian process

probability distribution over functions $y(\mathbf{x})$ such that if we consider an arbitrary set of points $\mathbf{x}_1, \dots, \mathbf{x}_N$ the joint probability $p(\mathbf{x}_1, \dots, \mathbf{x}_N)$ is Gaussian.

Solved with kernels

Gaussian Processes Regression Example



Source: [10.1109/MSP.2013.2250352 \(DOI\)](https://doi.org/10.1109/MSP.2013.2250352)

2020-09-28

- sparse Kernel methods
- maximum margin classifier
- support vector machines
 - linearly separable data
 - overlapping data

PRML Ch. 7

Maximum margin classifier

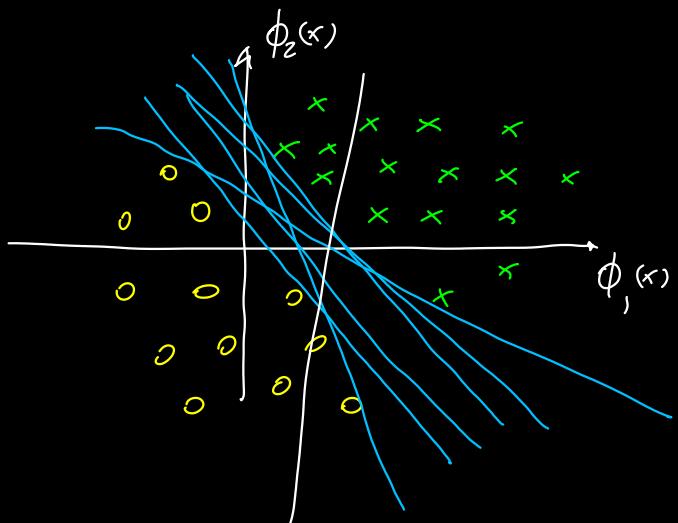
two-class problem

$$y(x) = w^T \phi(x) + b$$

training data

$$\{x_1, \dots, x_N ; x_n \in \mathbb{R}^D\}$$

$$\{t_1, \dots, t_N ; t_n \in \{-1, +1\}\}$$

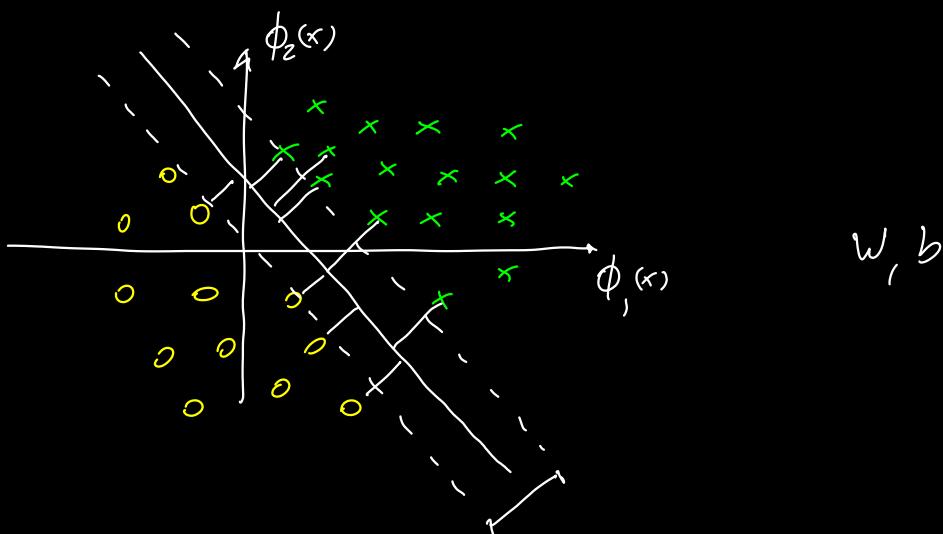


$$\left. \begin{array}{l} \exists w, b \text{ s.t. } y(x_n) > 0 \Leftrightarrow t_n = +1 \\ y(x_n) < 0 \Leftrightarrow t_n = -1 \end{array} \right\} \Rightarrow y(x_n) \cdot t_n > 0 \quad \forall n$$

Goal: minimize generalization error

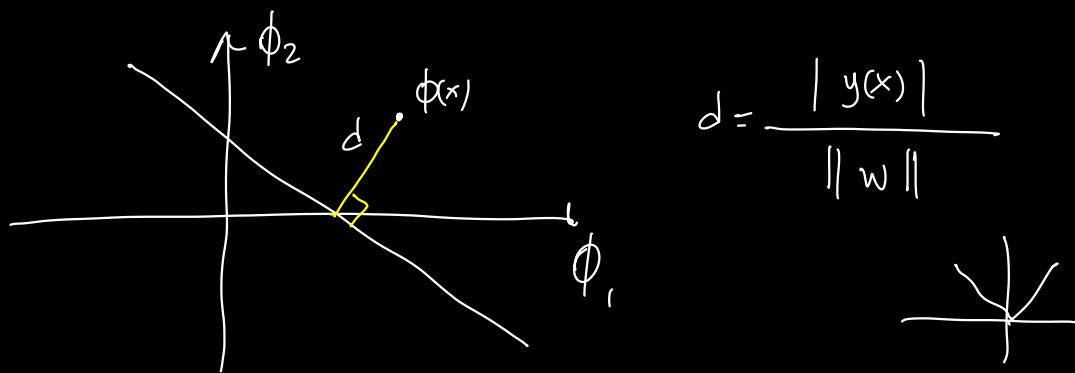
We only have the training data (do not know the underlying distribution)

Heuristic solution: maximize margin

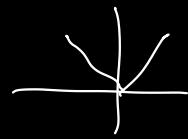


Recall:

- if \$x\$ is on decision boundary \$\Rightarrow y(x) = 0\$



$$d = \frac{|y(x)|}{\|w\|}$$



- we are only interested in no error $\Rightarrow t_n y(x_n) > 0$

$$d = \frac{t_n y(x_n)}{\|w\|} = \frac{t_n (w^T \phi(x_n) + b)}{\|w\|} \quad \forall n \in [1, N]$$

Optimization:

$$\underset{w, b}{\operatorname{argmax}} \left\{ \frac{1}{\|w\|} \min_n \left[t_n (w^T \phi(x_n) + b) \right] \right\}$$

↑
independent of n

*hard to
solve
directly*

$$w, b \rightarrow K w, K b$$

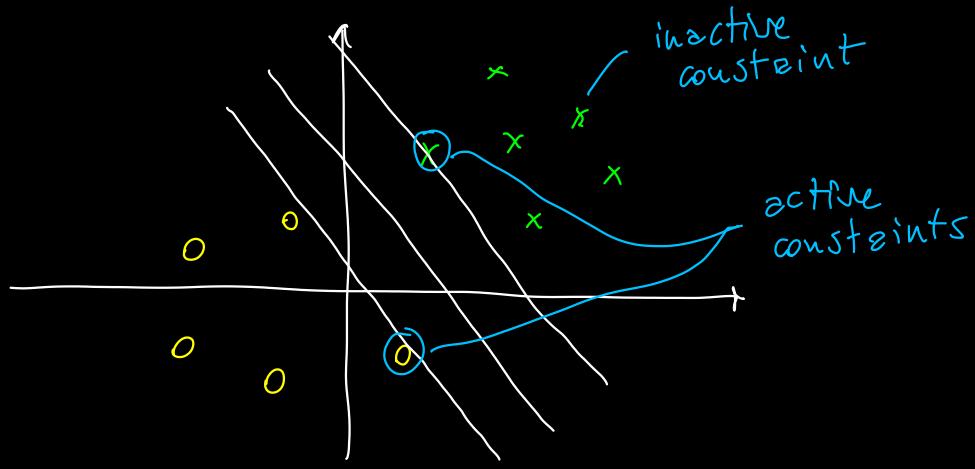
$$\frac{t_n (K w^T \phi(x) + K b)}{\|K w\|} = \frac{t_n (w^T \phi(x) + b)}{\|w\|}$$

Canonical representation of the decision hyperplane

$t_n (w^T \phi(x_n) + b) = 1$ for the point that is closest to the hyperplane

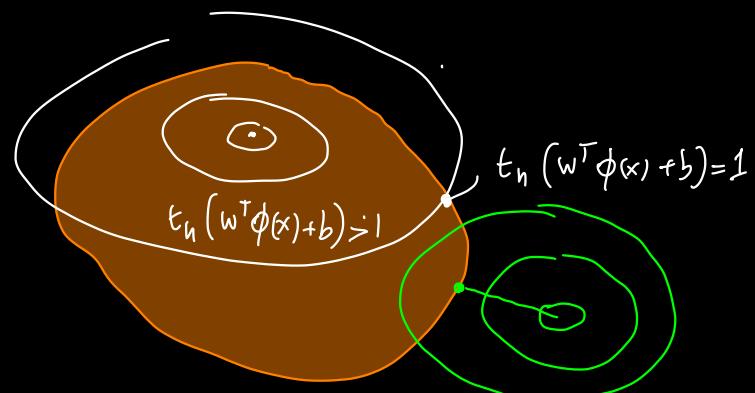
Then

$$t_n (w^T \phi(x_n) + b) \geq 1 \quad \forall n \in [1, N]$$



$$\underset{w, b}{\operatorname{argmax}} \left\{ \frac{1}{\|w\|} \underbrace{\min_n \left[t_n (w^T \phi(x) + b) \right]}_1 \right\} = \\ = \underset{w, b}{\operatorname{argmax}} \left\{ \frac{1}{\|w\|} \right\} = \underset{w, b}{\operatorname{argmin}} \frac{1}{2} \|w\|^2$$

subject to the constraint $\underbrace{t_n (w^T \phi(x) + b) \geq 1}_{\forall n \in [1, N]}$



Appendix E

quadratic programming

$$a = (a_1, \dots, a_N)^T$$

Lagrange multipliers

$$a_n \geq 0 \quad \forall \text{ each data point}$$

$$L(w, b, a) = \frac{1}{2} \|w\|^2 - \sum_{n=1}^N a_n \left\{ \underbrace{t_n (w^T \phi(x_n) + b)}_{\geq 0} - 1 \right\}$$

↑
minimize ↑
 ≥ 0

$$\frac{\partial L}{\partial w} = 0 \quad w = \sum_{n=1}^N a_n t_n \phi(x_n) \quad \leftarrow$$

$$\frac{\partial L}{\partial b} = 0 \quad \boxed{0 = \sum_{n=1}^N a_n t_n}$$

$$\tilde{L}(a) = \frac{1}{2} \left\| \sum_{n=1}^N a_n t_n \phi(x_n) \right\|^2 - \sum_{n=1}^N a_n \left\{ t_n \left(\underbrace{\sum_{m=1}^N a_m t_m \phi^\top(x_m)}_{w^\top} \right) \phi(x_n) \right\}$$

$$- b \sum_{n=1}^N a_n t_n + \sum_{n=1}^N a_n =$$

$$= \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m \phi^\top(x_n) \phi(x_m) - \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m \underbrace{\phi^\top(x_n) \phi(x_m)}$$

$$+ \sum_{n=1}^N a_n = - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m \underbrace{\phi^\top(x_n) \phi(x_m)}_{K(x_n, x_m)} + \sum_{n=1}^N a_n$$

dual representation for the optimality criterion

$$\begin{aligned} a_n &\geq 0 \\ \sum_{n=1}^N a_n t_n &= 0 \quad n \in [1, N] \end{aligned}$$

Quadratic problem in M variables $\Rightarrow \underline{O(M^3)}$

$$L(\underbrace{w, b}_M, a)$$

$\tilde{L}(a)$ N variables $\Rightarrow O(N^3)$

$$y(x) = w^\top \phi(x) + b = \sum_{n=1}^N a_n t_n \underbrace{\phi^\top(x_n) \phi(x)}_{K(x, x_n)} + b$$

$$= \sum_{n \in N_S} a_n t_n K(x, x_n) + b$$

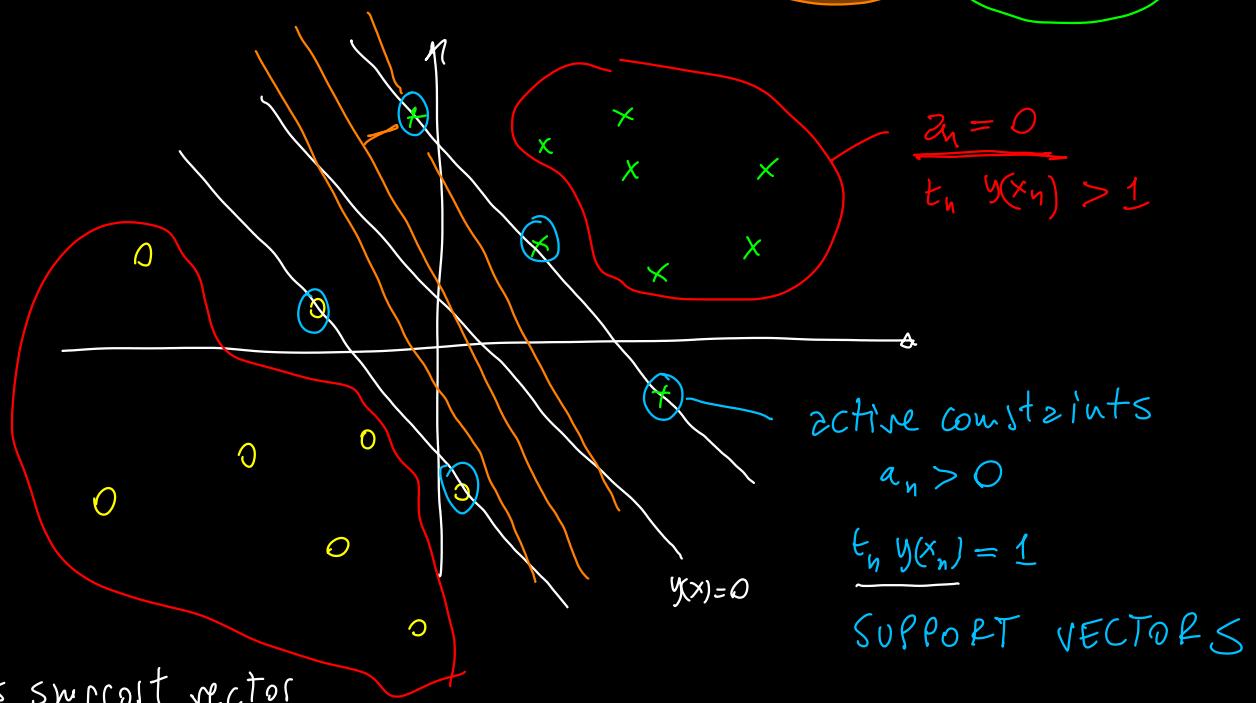
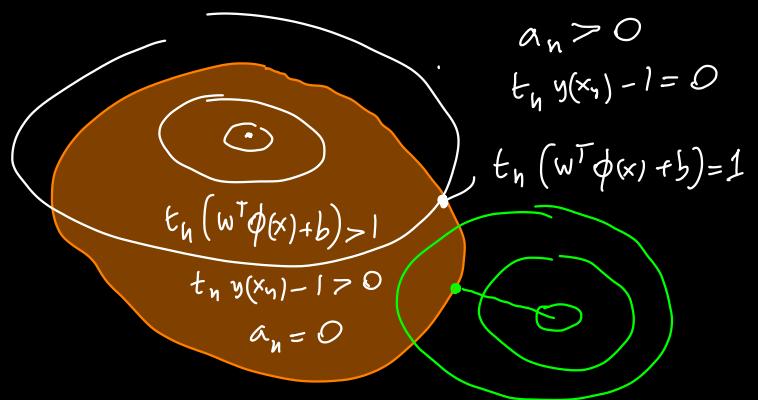
Karush-Kuhn-Tucker KKT conditions:

$$\alpha_n \geq 0$$

$$t_n y(x_n) - 1 \geq 0$$

$$\alpha_n \{ t_n y(x_n) - 1 \} = 0$$

=



if x_n is support vector

$$\Rightarrow 1 = t_n y(x_n) = t_n \left(\sum_{m \in S} \alpha_m t_m K(x_n, x_m) + b \right)$$

$+1, -1$

$$t_n = t_n^2 \left(\begin{array}{c} \\ \\ \parallel \\ 1 \end{array} \right) N_S b$$

$$\sum_{n \in S} t_n = \sum_{n \in S} \sum_{m \in S} \alpha_m t_m K(x_n, x_m) + \sum_{n \in S} b$$

$$w = \sum_{n \in S} a_n t_n \phi(x_n)$$

$$b = \frac{1}{N_S} \sum_{n \in S} \left[t_n - \sum_{m \in S} a_m t_m K(x_n, x_m) \right]$$

Sparse Kernel Methods

TTT4185 Machine Learning for Signal Processing

Giampiero Salvi

Department of Electronic Systems
NTNU

HT2021

Outline

1 Linearly Separable Classes

- Maximum Margin Classifier
- Canonical Representation
- Dual Representation

2 Overlapping Classes

3 Multi-Class SVM

Outline

1 Linearly Separable Classes

- Maximum Margin Classifier
- Canonical Representation
- Dual Representation

2 Overlapping Classes

3 Multi-Class SVM

Two-Class Classification Problem

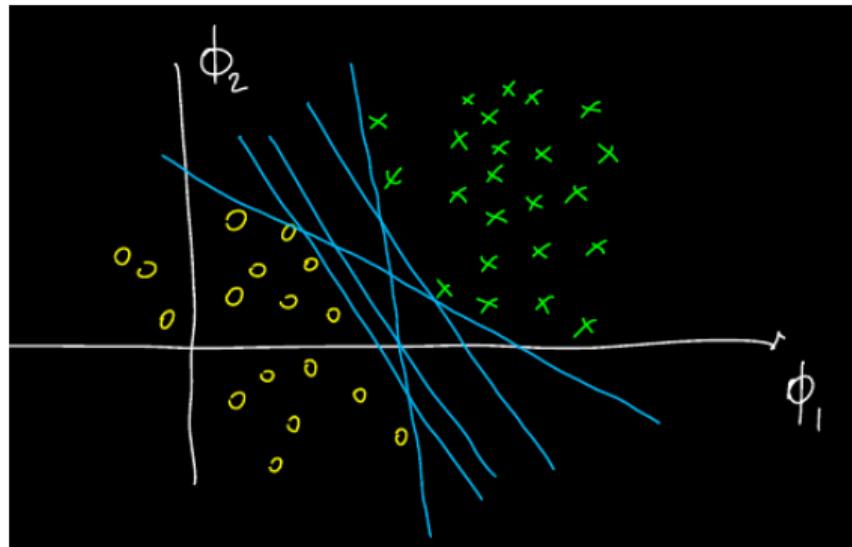
Model

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x}) + b$$

Training data (assume linearly separable)

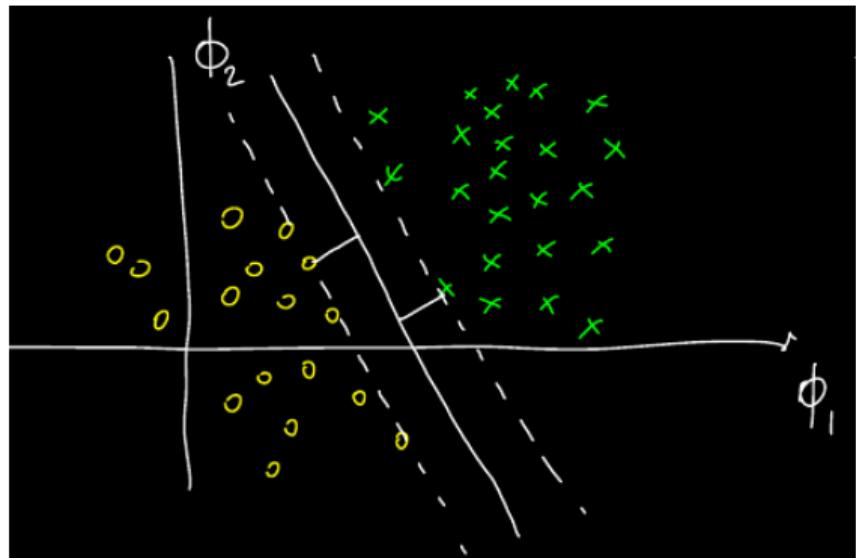
$$\{\mathbf{x}_1, \dots, \mathbf{x}_N\}, \{t_1, \dots, t_N\}$$

how to choose the best solution?



Maximum Margin

- Goal: minimize generalization error
- Problem: we can not use the test data
- Solution (Heuristics): choose decision boundary as far as possible from data



Optimization

- Only interested in solutions with no misclassifications:

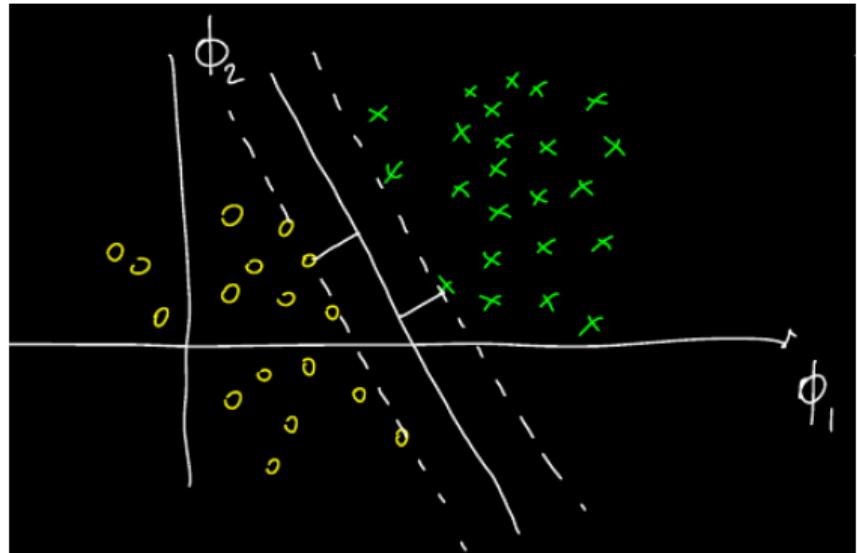
$$t_n y(\mathbf{x}_n) > 0$$

- if \mathbf{x} on the decision boundary then

$$y(\mathbf{x}) = 0$$

- Distance between a point and the decision boundary:

$$\frac{|y(\mathbf{x})|}{\|\mathbf{w}\|}$$

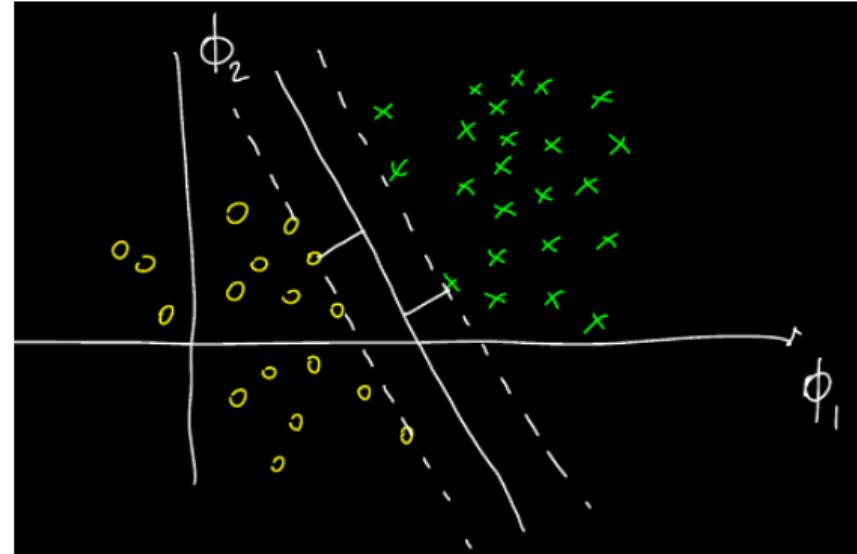


Optimization

- We can rewrite the distance as

$$\frac{t_n y(\mathbf{x}_n)}{\|\mathbf{w}\|} = \frac{t_n (\mathbf{w}^T \phi(\mathbf{x}) + b)}{\|\mathbf{w}\|}$$

- we want to maximize the distance of the closest point:



$$\arg \max_{\mathbf{w}, b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n \left[t_n (\mathbf{w}^T \phi(\mathbf{x}) + b) \right] \right\}$$

Canonical Representation

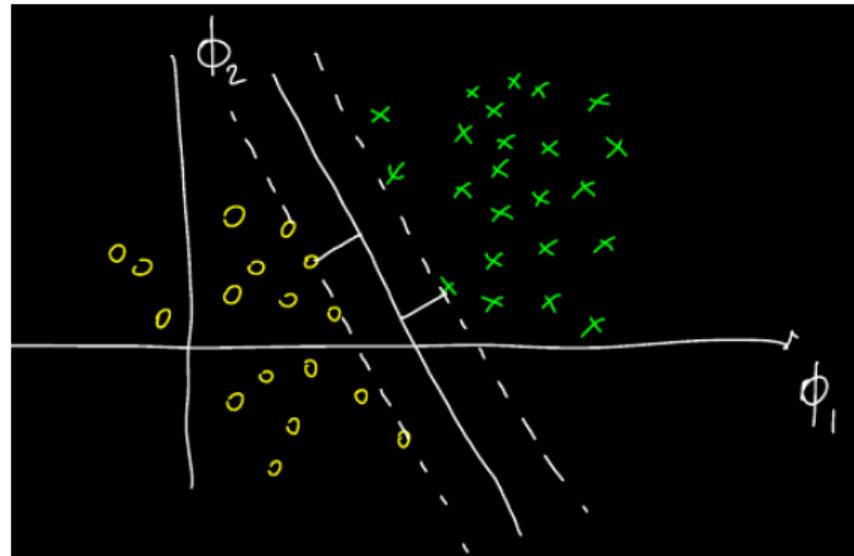
- Rescaling \mathbf{w}, b does not change distances
- Define \mathbf{w}, b such that:

$$t_n (\mathbf{w}^T \phi(\mathbf{x}) + b) = 1$$

for the point that is closest to the decision boundary

- then

$$t_n (\mathbf{w}^T \phi(\mathbf{x}) + b) \geq 1, \forall n \in [1, N]$$



Optimization Problem (Quadratic Programming)

$$\arg \max_{\mathbf{w}, b} \left\{ \frac{1}{\|\mathbf{w}\|} \right\} = \arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2$$

subject to the constraints

$$t_n (\mathbf{w}^T \phi(\mathbf{x}) + b) \geq 1, \forall n \in [1, N]$$

- can be solved with Lagrange multipliers

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N a_n \{ t_n (\mathbf{w}^T \phi(\mathbf{x}) + b) - 1 \}$$

Dual Representation

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

subject to

$$a_n \geq 0$$

$$\sum_{n=1}^N t_n a_n = 0$$

Dual Representation for Prediction

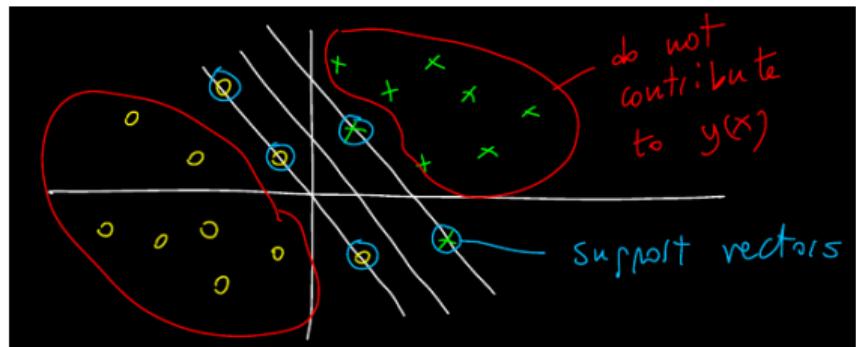
$$y(\mathbf{x}) = \sum_{n=1}^N a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b$$

subject to the constraints
(Karish-Kuhn-Tucker, KKT)

$$a_n \geq 0$$

$$t_n y(\mathbf{x}_n) - 1 \geq 0$$

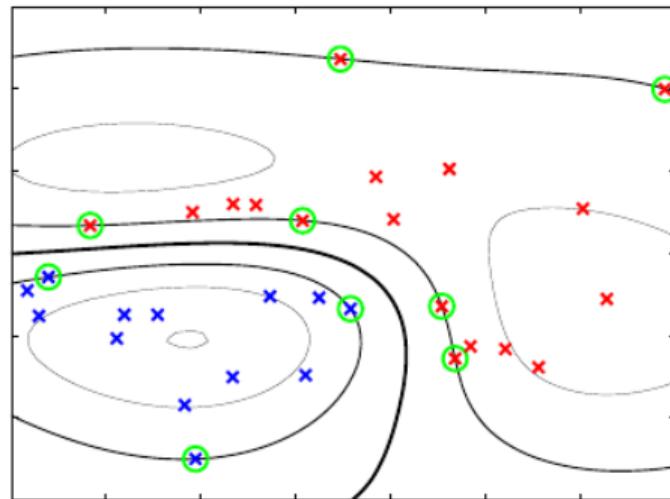
$$a_n \{t_n y(\mathbf{x}_n) - 1\} = 0$$



Properties

- complexity of quadratic programming in M variables: $O(M^3)$
- with dual representation we have N variables (usually $N \gg M$)
- but is convex optimization: global optimum
- and we can work in arbitrary large dimensions

Example



Example

<https://playground.tensorflow.org>

Outline

1 Linearly Separable Classes

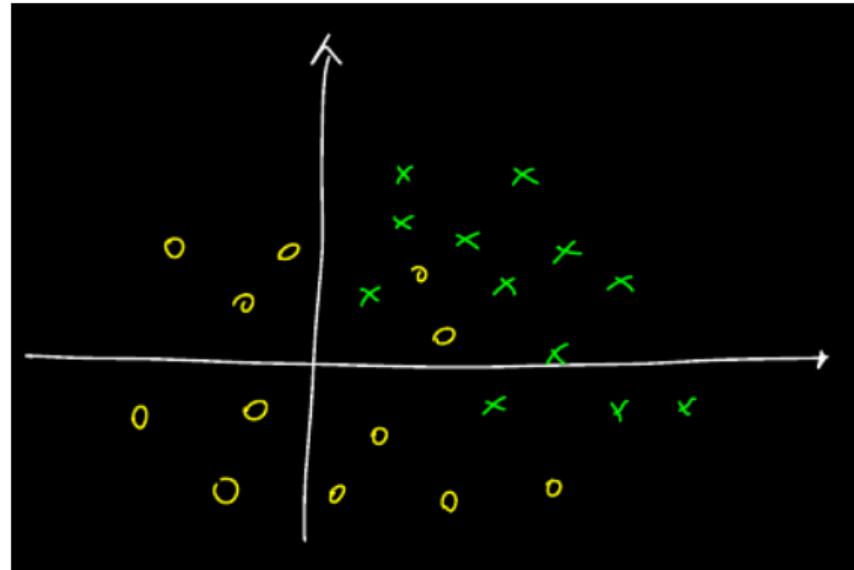
- Maximum Margin Classifier
- Canonical Representation
- Dual Representation

2 Overlapping Classes

3 Multi-Class SVM

Overlapping Classes

- we need to allow some misclassifications



Loss Function

- previously assumed zero errors + canonical representation
- \Rightarrow loss function is $\frac{1}{2}||\mathbf{w}||^2$
- equivalent to

$$\sum_{n=1}^N E_\infty(t_n y(\mathbf{x}_n) - 1) + \lambda ||\mathbf{w}||^2,$$

where

$$E_\infty(z) = \begin{cases} 0 & \text{if } z \geq 0 \\ \infty & \text{otherwise.} \end{cases}$$

- we need to modify this to allow for finite errors

Slack Variables

$$\xi_n \geq 0,$$

$$\forall n \in [1, N]$$

$$\xi_n = 0,$$

inside the margin*

$$\xi_n = |t_n - y(\mathbf{x}_n)|,$$

outside the margin*

we substitute the constraint $t_n y(\mathbf{x}_n) \geq 0$ with:

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n,$$

$$\forall n \in [1, N]$$

* with respect to the class

Loss Function

Goal: maximize margin by minimizing error:

$$C \sum_{n=1}^N \xi_n + \frac{1}{2} \|\mathbf{w}\|^2$$

- C controls trade-off between slack variable penalty and margin
- for misclassified points: $\xi_n > 1$
- $\Rightarrow \sum_{n=1}^N \xi_n$ upper bound to # errors
- $C \rightarrow \infty$ gives same solution as for separable data

Lagrange Multipliers

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n - \sum_{n=1}^N a_n \{t_n y(\mathbf{x}_n) - 1 + \xi_n\} - \sum_{n=1}^N \mu_n \xi_n$$

KKT conditions:

$$a_n \geq 0$$

$$t_n y(\mathbf{x}_n) - 1 + \xi_n \geq 0$$

$$a_n (t_n y(\mathbf{x}_n) - 1 + \xi_n) = 0$$

$$\mu_n \geq 0$$

$$\xi_n \geq 0$$

$$\mu_n \xi_n \geq 0$$

Dual Representation

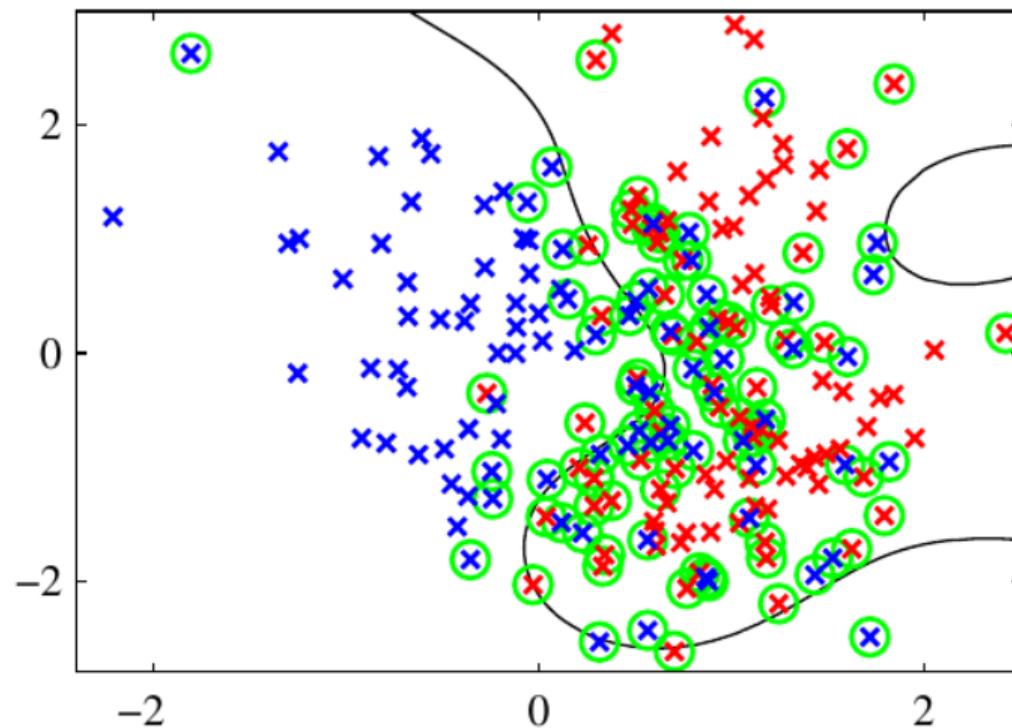
$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

Same as before but with different constraints

$$0 \leq a_n \leq C$$

$$\sum_{n=1}^N a_n t_n = 0$$

Example



Outline

1 Linearly Separable Classes

- Maximum Margin Classifier
- Canonical Representation
- Dual Representation

2 Overlapping Classes

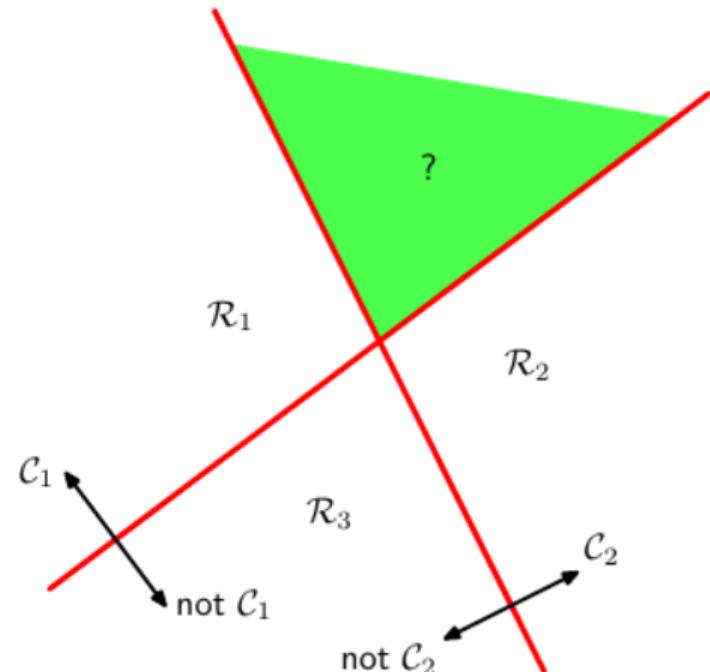
3 Multi-Class SVM

Multi-Class SVM

- SVM is a two-class classifier
- Several approaches to solve the K -class problem

Multi-Class SVM: Vapnik 1998

- Construct K different classifiers $y_k(\mathbf{x})$
- for each use data from C_k as positive examples and remaining classes as negative
- known as one-versus-the-rest approach
- problem in figure



Multi-Class SVM: Possible Solution

- Instead of one-versus-the-rest use:

$$y(\mathbf{x}) = \max_k y_k(\mathbf{x})$$

- new problem: every $y_k(\mathbf{x})$ is trained on a different task
- no guarantee that have same scale
- other problem: if e.g. $K = 10$, for each $y_k(\mathbf{x})$, 10% positive and 90% negative examples

Multi-Class SVM: Possible Solution

- Instead of one-versus-the-rest use:

$$y(\mathbf{x}) = \max_k y_k(\mathbf{x})$$

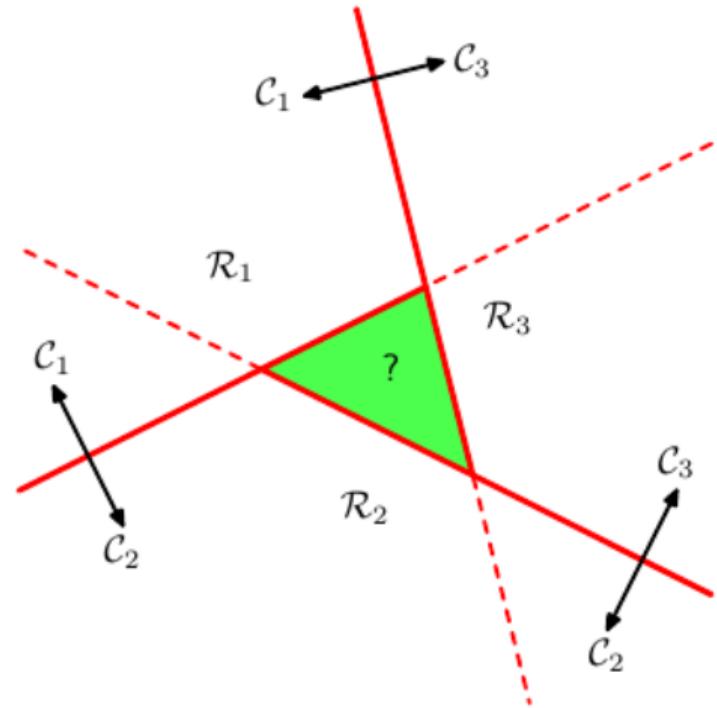
- new problem: every $y_k(\mathbf{x})$ is trained on a different task
- no guarantee that have same scale
- other problem: if e.g. $K = 10$, for each $y_k(\mathbf{x})$, 10% positive and 90% negative examples
- Solution (Lee et al 2001): scale targets
 - +1 for the positive class
 - $\frac{-1}{K-1}$ for the negative class

Multi-Class SVM: Weston and Watkins 1999

- objective function for training all $y_k(\mathbf{x})$ simultaneously
- but computationally expensive:
- instead of K problems over N data points $O(KN^2)$
- single problem of size $(K - 1)N$ which is $O(K^2N^2)$

Multi-Class SVM: one-versus-one

- train $\frac{K(K-1)}{2}$ classifiers
(one-versus-one)
- this is used in `sklearn.svm.SVC`
(assignment 2)





NTNU | Norwegian University of
Science and Technology

DEEP LEARNING DEEP NEURAL NETWORKS

Tor Andre Myrvoll
2020-10-05

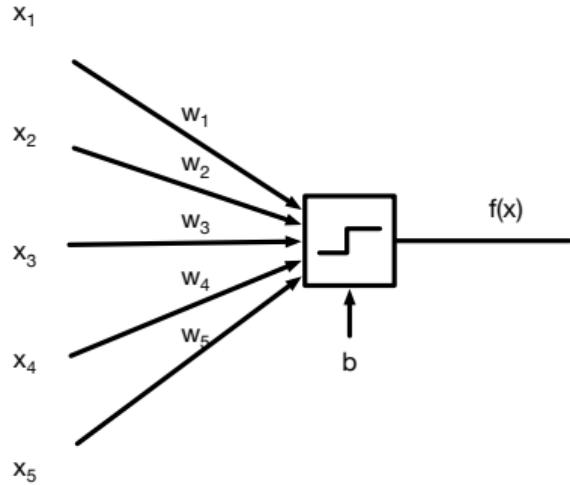
Introduction - Outline

- ▶ Overview of the coming lectures
- ▶ What are deep neural networks (DNN) ?
- ▶ Examples of DNN applications
- ▶ Learning DNN parameters

Introduction - Overview

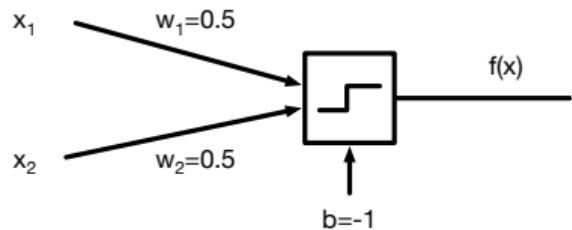
- ▶ Lecture 1: Multilayer perceptrons and back-propagation
- ▶ Lecture 2: Training DNNs - best practice
- ▶ Lecture 3: Convolutional Neural Networks (CNN)
- ▶ Lecture 4: Generative Adversarial Networks (GAN). Recursive Neural Networks (RNN)

Introduction - First there was the perceptron...



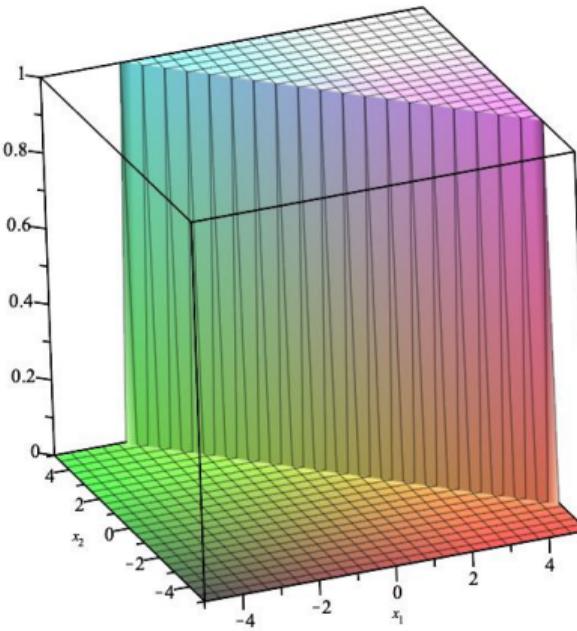
$$f(x) = \begin{cases} 1 & \sum_i w_i x_i + b \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Introduction - First there was the perceptron...



$$f(x) = \begin{cases} 1 & 0.5(x_1 + x_2) - 1 > 0 \\ 0 & \text{otherwise} \end{cases}$$

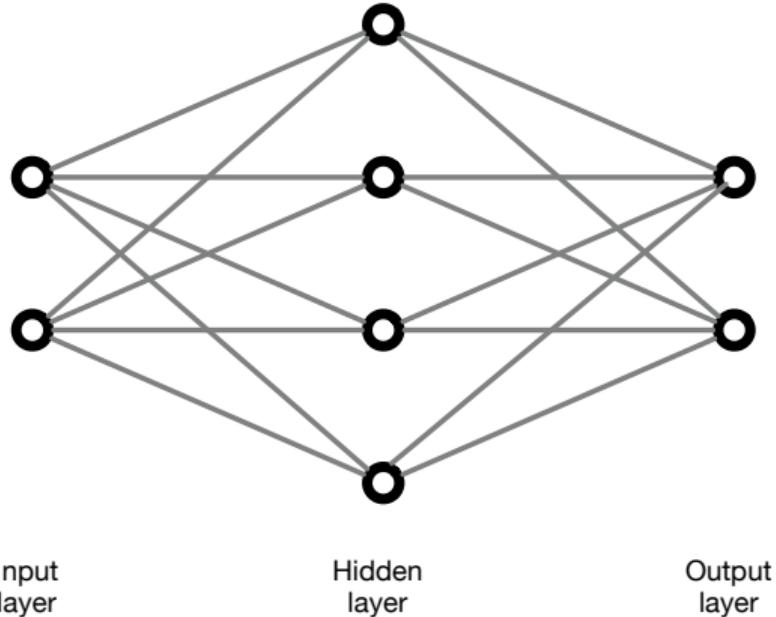
Introduction - First there was the perceptron...



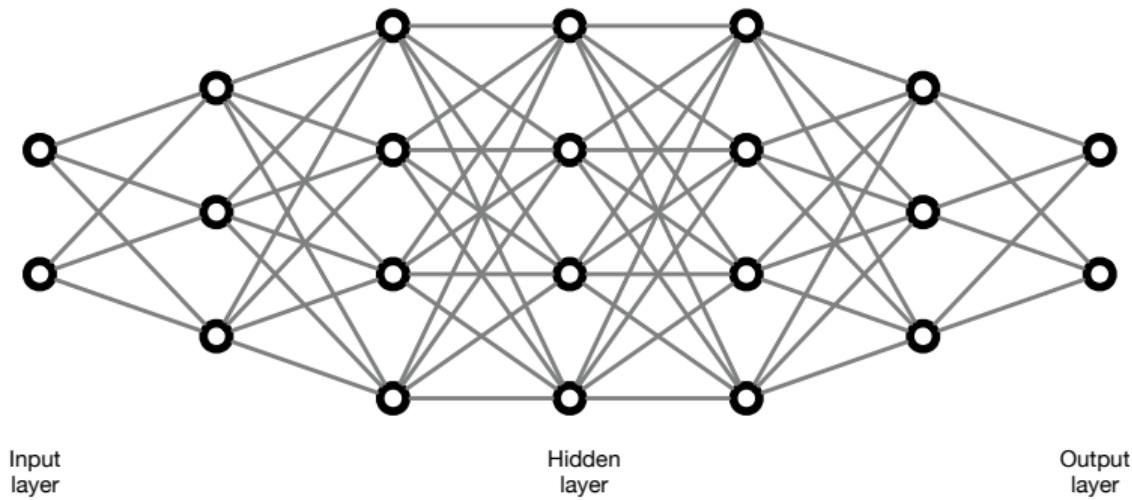
Introduction - First there was the perceptron...

- ▶ The perceptron was very popular in the 1950s, but problems with generalization to more complex problems made it fall out of vogue
- ▶ Over the years the two major ideas made the use of neural networks feasible:
 - ▶ Multiple layers of perceptrons made the networks more powerful
 - ▶ Differentiable processing functions made optimization easier
- ▶ Today one of the most commonly used neural networks is the *multilayer perceptron*

Introduction - What are deep neural networks



Introduction - What are deep neural networks



Introduction - What are deep neural networks

- ▶ Deep Neural Networks are in essence functions/mappings
- ▶ The mappings are defined by the parameters of the DNN
- ▶ Parameters are learned from data using a training algorithm
- ▶ DNNs can express a wide variety of mappings
 - ▶ Regression problems: $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$
 - ▶ Posterior distributions: $f : \mathbb{R}^m \rightarrow \{p_1, p_2, \dots, p_n\}$, $p_i > 0 \forall i$, $\sum p_i = 1$
 - ▶ Multilabels: $f : \mathbb{R}^m \rightarrow [0, 1]^n$, where $p \in [0, 1]$ indicates to what extend a label is present

Simple network structures like the one shown in the previous slide are often called feed-forward networks or multilayer perceptrons

Introduction - What are deep neural networks

Multilayer perceptrons with one or more layers are *universal approximators*:

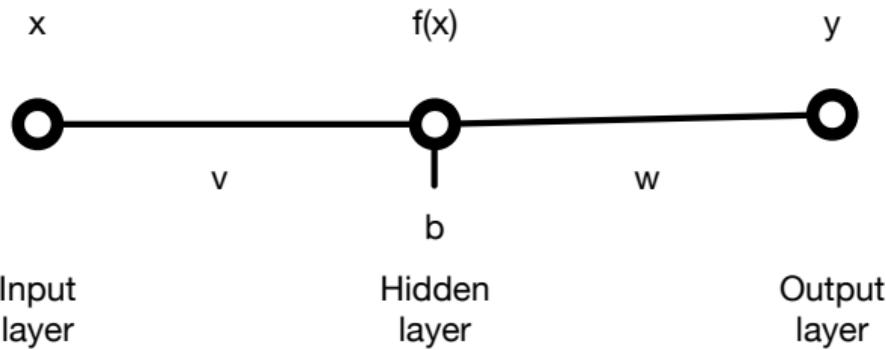
- ▶ Assume $x \in E \subset \mathbb{R}^m$, where E is a closed and bounded subset.
- ▶ Let $f : E \rightarrow \mathbb{R}^n$ be a smooth function defined on E .
- ▶ It can then be shown that for every $\varepsilon > 0$ there always exists a neural network i with N parameters implementing a function $g : E \rightarrow \mathbb{R}^n$, so that $|f(x) - g(x)| < \varepsilon$.

This means that we can be sure that whatever relationship our data represents, our neural network can model it given enough parameters.

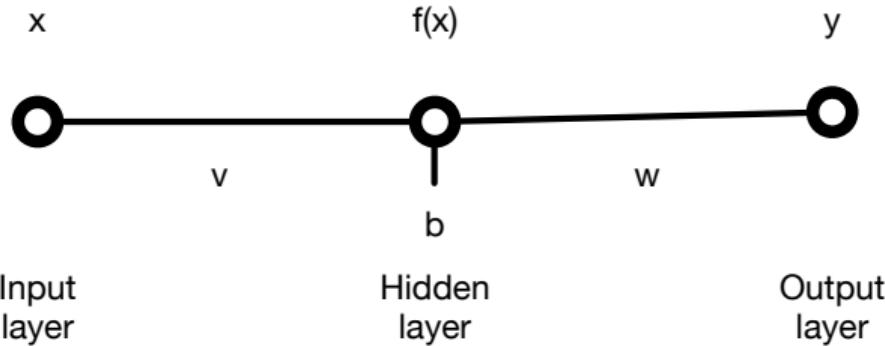
Introduction - What are deep neural networks

- ▶ In practice however, this theorem *seems* of little use, as a single layer network will grow exponentially when we increase the dimensionality of the domain, \mathbb{R}^m .
- ▶ In other words, going from \mathbb{R}^m to \mathbb{R}^{m+1} , will on average increase the number of parameters to obtain an accuracy within $\varepsilon > 0$ by a factor $C > 1$.
- ▶ This may not sound bad if $C = 1 + \delta$, but consider classifying a 512×512 pixel image?! $((C + \delta)^{512 \times 512} = (C + \delta)^{262144}$ will be a large number unless $C \approx 1$)
- ▶ This is where the power of *deep neural networks* enters. Experiments and theoretical analysis shows that increasing the *depth* of a neural network, will at some point dramatically decrease the number of parameters needed to approximate a given function.

Multilayer perceptrons - Simplest MLP ever



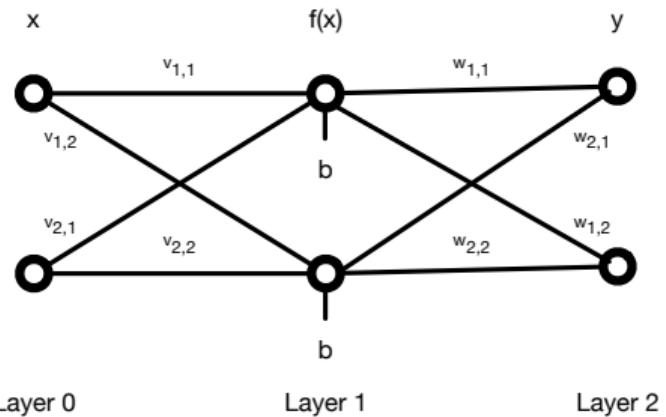
Multilayer perceptrons - Simplest MLP ever



The input-output relationship of this network is:

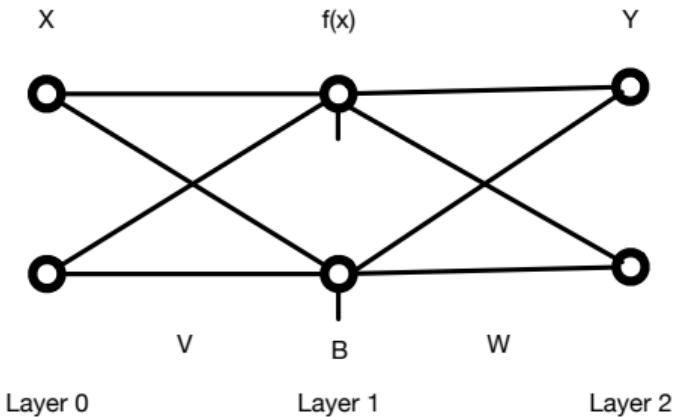
$$y = w \cdot f(v \cdot x + b)$$

Multilayer perceptrons - Slightly more complicated MLP



The input-output relationship of this network is already much more complicated.

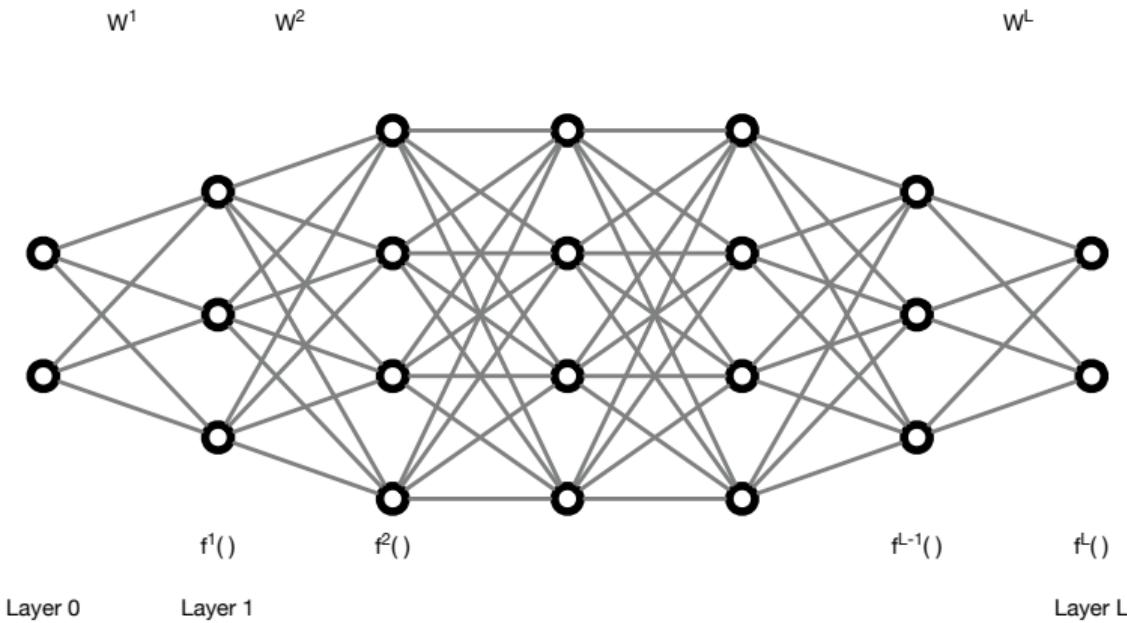
Multilayer perceptrons - Slightly more complicated MLP



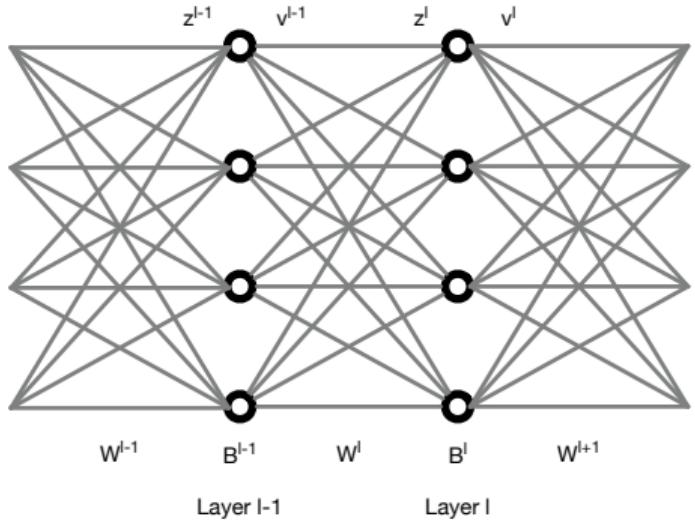
Using matrix notation:

$$Y = W^T \cdot f(V^T \cdot X + B)$$

Multilayer perceptrons - General MLP



Multilayer perceptrons - General MLP closeup



- ▶ Weight matrix W^I
- ▶ Bias vector B^I
- ▶ Excitation vector
$$z^I = W^{I,T} \cdot v^{I-1} + B^I$$
- ▶ Activation vector $v^I = f(z^I)$

Activation functions - Hidden layer functions

Activation functions:

- ▶ The sigmoid:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

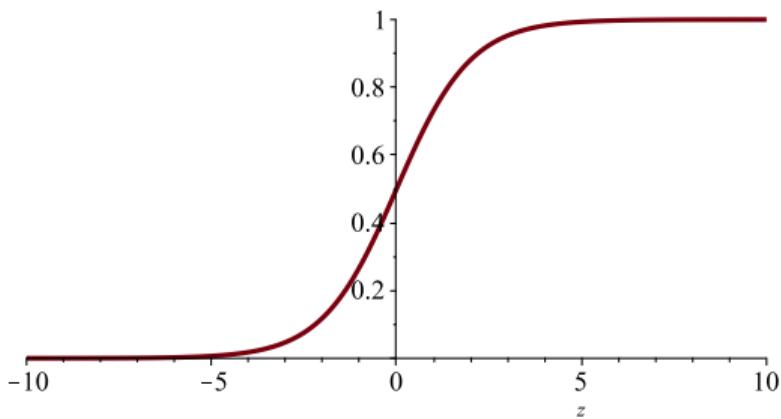
- ▶ The hyperbolic tangent:

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

- ▶ The rectified linear unit:

$$\text{ReLU}(z) = \begin{cases} 0, & z < 0 \\ z, & 0 \leq z \end{cases}$$

Activation functions - Sigmoid

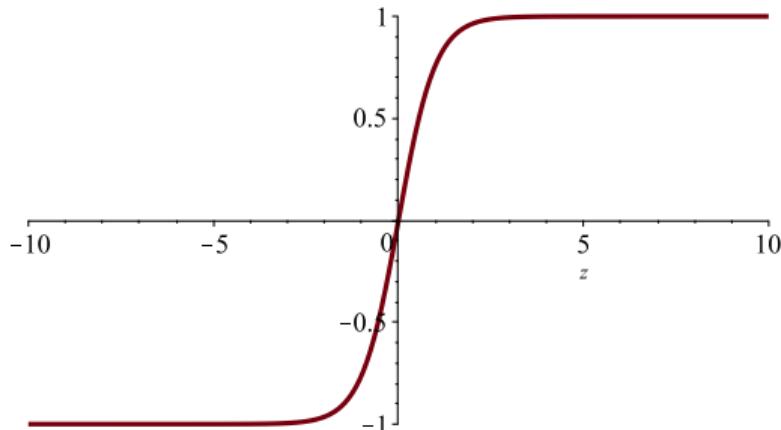


- ▶ Function: $\sigma(z) = \frac{1}{1+e^{-z}}$
- ▶ Continuous approximation to the classical perceptron function

$$\text{per}(z) = \begin{cases} 0, & z < 0 \\ 1, & 0 \leq z \end{cases}$$

- ▶ Used to be the most popular activation function for MLPs

Activation functions - Hyperbolic tangent

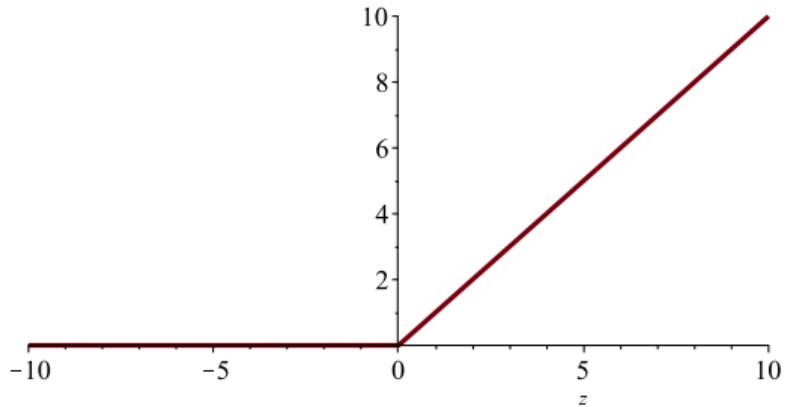


- ▶ Function: $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
- ▶ Symmetric version of the sigmoid

$$\text{per}(z) = \begin{cases} 0, & z < 0 \\ 1, & 0 \leq z \end{cases}$$

- ▶ Popular alternative to the sigmoid

Activation functions - Rectified Linear Unit



- ▶ Function: $\text{ReLU}(z) = \begin{cases} 0, & z < 0 \\ z, & 0 \leq z \end{cases}$
- ▶ The most common activation function today
- ▶ Training ReLU based networks faster and easier

Activation functions - Final layer functions

The functions used in the final layer, the output, of the network, varies from the functions used in the hidden layers. Examples:

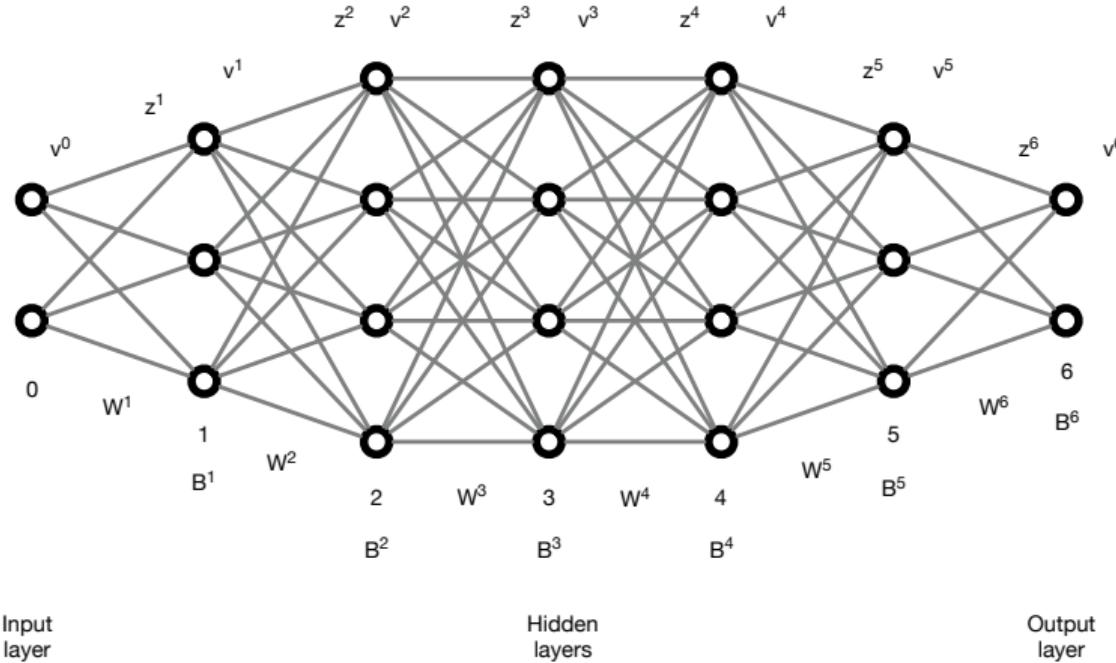
- ▶ Linear output: $f(z) = z$. Mostly used in regression where one typically wants unbounded output.
- ▶ Softmax: Let $\{z_i\}$ be the elements of the exitation vector z . Then the i th output from the softmax function is

$$f_i(z) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

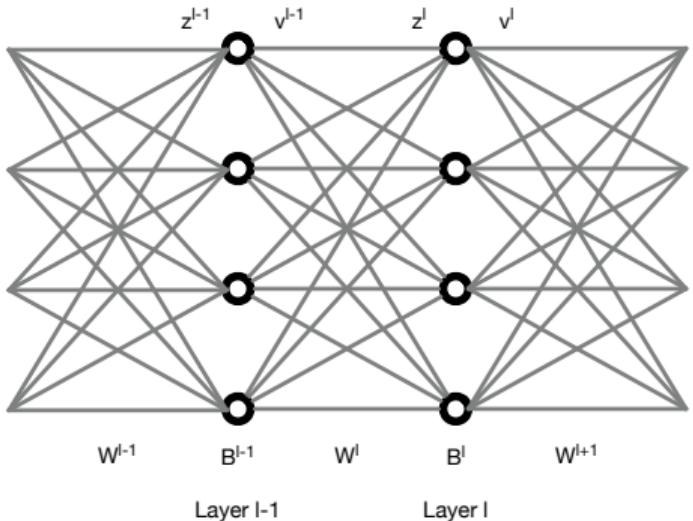
We see that the elements of the softmax function are positive and sums to one. Used for classification problems.

- ▶ Sigmoid: Same as for hidden layers. Used in multi-label classification.

Training the network - The forward computation



Training the network - The forward computation



- ▶ Given an input $x \in \mathbb{R}^m$, we want to compute the output, $f(x)$ of the neural network.
- ▶ This is done using a simple forward loop over the layers of the networks
- ▶ As we do the forward computation we will store the intermediate results for use with the *backpropagation algorithm* that we will be discussing after this

Training the network - The forward computation

```
1:  $v^0 \leftarrow x$ 
2: for  $l \leftarrow 1, L$  do
3:    $z^l \leftarrow W^l v^{l-1} + B^l$ 
4:    $v^l \leftarrow f^l(z^l)$ 
5: end for
6: if Regression then
7:    $v^L \leftarrow z^L$ 
8: else
9:    $v^L \leftarrow \text{softmax}(z^L)$ 
10: end if
```

Training the network - The objective function

The MLP is determined by the following:

- ▶ The network topology
 - ▶ Number of layers
 - ▶ Number of nodes per layer
- ▶ The choice of nonlinearities $f(x)$
- ▶ The weights $\{W^l\}$ and biases $\{B^l\}$

The network topology and nonlinearity is usually chosen in advance and held fixed while we optimize the network parameters.

Training the network - The objective function

Given a training set $X = \{x_n\}$, $Y = \{y_n\}$. To train an MLP we need to define an *objective function* that measures the discrepancy, or error, between the output of an MLP, $\hat{y}_n = f(x_n)$, and the *true value* from the training set, y_n .

- ▶ We define a *loss function* $I(y, \hat{y})$, that measures the loss when we predict \hat{y} as opposed to y .
- ▶ Examples of loss functions are:
 - ▶ The squared error: $I(y, \hat{y}) = \|y - \hat{y}\|^2$. This is a good choice for regression problems
 - ▶ The cross entropy loss: $I(y, \hat{y}) = \sum_i -y_i \log \hat{y}_i$. This is the most common choice for classification problems where y is a vector of 0 – 1 labels.

Training the network - Learning the parameters

- Given a loss function we can define the *total loss* for the training problem:

$$L(W, B, X, Y) = \frac{1}{|X|} \sum_n l(y_n, f(x_n; W, B))$$

- Our goal is now to find the network parameters W, B so that

$$\hat{W}, \hat{B} = \arg \min_{W, B} L(W, B, X, Y)$$

- It is not possible to *solve* this in closed form, or even approximately.
- The solution is to use *numerical search*, or more specifically, *gradient descent*

Training the network - Gradient descent

Assume we have a function $f(x)$ that we want to minimize with respect to x . That is, we want to find

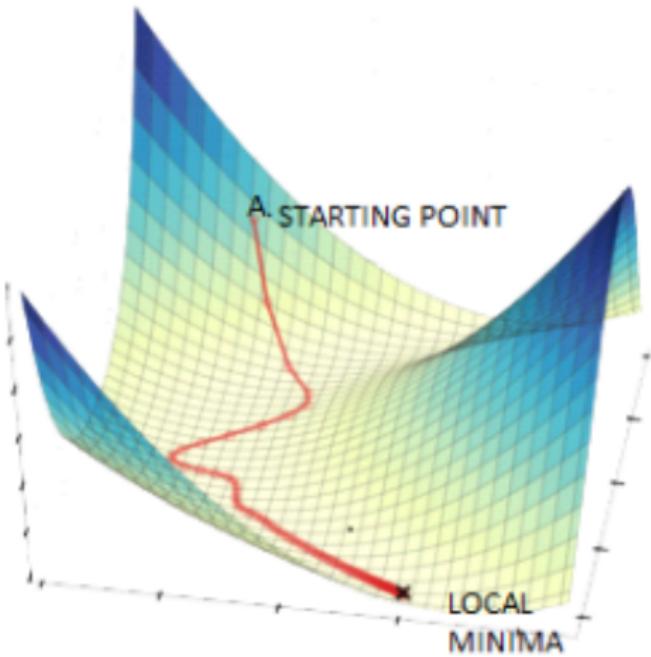
$$x^* = \arg \min_x f(x)$$

The function is too complicated to solve directly, however, we can compute the gradient at any point x . Let $f(x^{(0)})$ be an initial guess of the minimum of the function. We know that the gradient $\nabla_x f(x)$ points towards larger $f(x)$, while the negative gradient points towards smaller $f(x)$. Let our next guess for x^* be

$$x^{(1)} = x^{(0)} - \epsilon \nabla_x f(x)$$

For small ϵ we are guaranteed that $f(x^{(1)}) < f(x^{(0)})$, and so we can create a sequence of updates that may converge to the optimal value.

Training the network - Gradient descent



Training the network - Gradient descent

We plan to optimize the MLP parameters in the following way:

1. Initialize $W^{(0)}, B^{(0)}$, eg. by using Gaussian noise
2. Iterate until convergence:

$$\begin{aligned} W^{(i)} &= W^{(i-1)} - \epsilon \nabla_W L(W, B, X, Y) \\ B^{(i)} &= B^{(i-1)} - \epsilon \nabla_B L(W, B, X, Y) \end{aligned}$$

To achieve this we need to be able to compute $\nabla_W L(W, B, X, Y)$ and $\nabla_B L(W, B, X, Y)$.

$$\nabla_W L(W, B, X, Y) = \frac{1}{|X|} \sum_n \nabla_W I(y_n, f(x_n; W, B))$$

(B is computed in a similar manner)

Training the network - The backward propagation algorithm

It turns out that the gradients can be computed in an efficient manner using *back propagation*

1. Initialize the network
2. For each iteration
 - 2.1 Do the forward computation
 - 2.2 Compute the gradient wrt. the last set of weights, W^L for each training example x_n, y_n
 - 2.3 It can now be shown that the gradient wrt. W^l depends on the gradient wrt. W^{l+1} . Hence, we can start at the end and propagate the gradient backward to the beginning of the network

Training the network - Back propagation

In a little more detail (discussing W only, B is similar):

- ▶ For each layer l we can compute

$$\nabla_{W_t^l} L(W, B, X, Y) = \left[f'(z_t^l) \cdot e_t^l \right] (v_t^{l-1})^T$$

where

$$e_t^l = (W_t^l)^T (f'(z_t^l \cdot e_t^{l+1}))$$

starting with

$$e_t^L = (V_t^L - y)$$

Summary - Wrapping up

- ▶ Neural networks are functions
- ▶ Universal approximation means they can model anything given by a mapping
- ▶ They can be efficiently trained using back propagation

Summary - Next time

- ▶ Best practices for neural network training
- ▶ Batch, mini-batch, stochastic gradient descent
- ▶ Regularization, drop-out
- ▶ Normalization
- ▶ And more...

Thank you for your attention



NTNU | Norwegian University of
Science and Technology



NTNU | Norwegian University of
Science and Technology

DEEP LEARNING DEEP NEURAL NETWORKS

Tor Andre Myrvoll
2020-10-12

Introduction - Outline

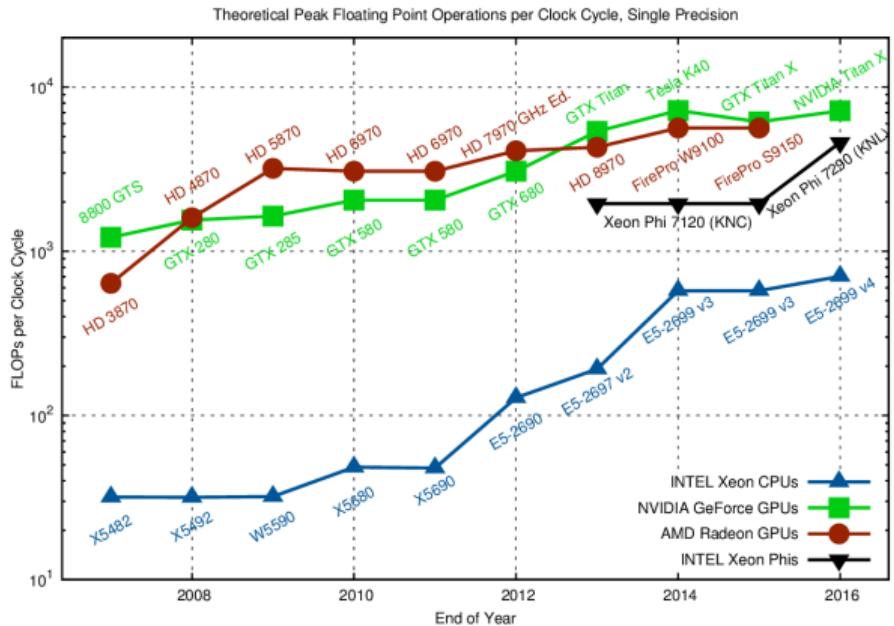
- ▶ Training DNNs – Common techniques
 - ▶ Batch, mini-batch and stochastic gradient descent
 - ▶ Normalization
 - ▶ Dropout
 - ▶ Variations on the gradient search
- ▶ Best practice
 - ▶ Training, validation and test data
 - ▶ Accuracy, precision, recall

Common techniques - Training on a GPU

Training a deep neural network using a GPU

- ▶ Training any reasonably sized DNN demands a massive amount of processing power
- ▶ Even the latest CPUs are no match for this challenge
- ▶ Luckily, the problem is *massively parallel*
- ▶ This means that we can use a large number of simple, specialized processing units, instead of a few very powerful, but generic ones
- ▶ A source of a large number of small, but efficient processing units is your graphics card (GPU)

Common techniques - Training on a GPU



Common techniques - Training on a GPU

- ▶ Another strength of a GPU is the very fast memory that it contains
- ▶ Deep learning problems are often very large, and accessing main memory of any computer system is very slow (relatively speaking)
- ▶ GPUs, on the other hand, have very fast RAM, so if we can put the data and the model onto the GPU, we will have a significant speedup
- ▶ To make use of a GPU we should therefore
 - ▶ Have a parallel formulation of the problem
 - ▶ Use proper memory management

Common techniques - Batch, mini-batch, SGD

We remember that the objective function, our total loss, was written

$$L(W, B, X, Y) = \frac{1}{|X|} \sum_n l(y_n, f(x_n; W, B))$$

which in turn gave us the following expression of the *gradient* of the objective function

$$\nabla_W L(W, B, X, Y) = \frac{1}{|X|} \sum_n \nabla_W l(y_n, f(x_n; W, B))$$

We see that the gradient is the sum of contributions for each training sample $\{x_n, y_n\}$. This means that we can compute each contribution independently and *in parallel*.

Common techniques - Batch, mini-batch, SGD

We define three approaches to computing the gradient $\nabla_W L(W, B, X, Y)$ (note: a complete pass through the training set is called *an epoch*):

- ▶ Batch: Compute all elements of the gradient and sum them.
 - ▶ Parallelizable
 - ▶ Slow convergence per epoch
- ▶ Mini-batch: Pick a random subset of training samples and compute an approximation to the true gradient
 - ▶ Parallelizable
 - ▶ Fast convergence per epoch
- ▶ Stochastic Gradient Descent: Pick a single training example at random and compute an approximation to the gradient
 - ▶ Not parallelizable
 - ▶ Fast convergence per epoch

Common techniques - Batch, mini-batch, SGD

- ▶ Batch should never be used as it's convergence is very slow. The reason for this is that we will only do one update of the model every we go through the entire training set.
- ▶ Mini-batch is the most popular approach by far. Selecting the number of examples used carefully, one can get a good approximation to the gradient, and update the model many times per epoch.
- ▶ The SGD gradient is very noisy, but on average the model will improve more per epoch than when using regular batch. Not parallelizable, but still used for some particular problems.

Common techniques - Normalization

Although feature extraction can be avoided in many cases for DNNs, some pre-processing is still in order:

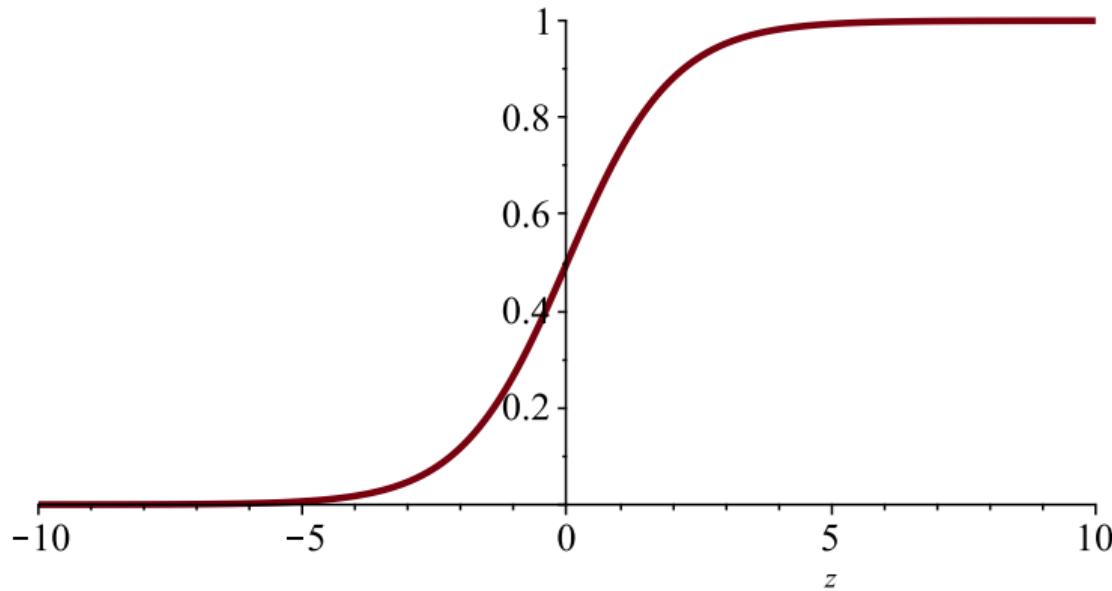
- ▶ Global normalization:
 - ▶ Mean and variance: Compute new features

$$x'_n = \frac{x_n - \bar{x}}{\bar{\sigma}}$$

where \bar{x} and $\bar{\sigma}$ is the sample mean and standart deviation respectively.

- ▶ Global normalization is used to mitigate numerical problems, in particular those involving gradients.

Common techniques - Normalization



Common techniques - Normalization

Local Normalization

- ▶ Sometimes there are variation within the dataset that is detractive to learning
- ▶ Example: When recording speech from a number of speakers, the microphones and recording conditions may vary.
- ▶ This will in turn yield an additive mean value to the feature vectors that is unique per speaker
- ▶ In this case it will be useful to compute the mean per speaker and subtract this

Common techniques - Regularization

Overtraining is one of the biggest problems across all machine learning methods, and DNNs are subject to this as well.

- ▶ We can control the overtraining by adjusting the number of free parameters, that is nodes and layers
- ▶ This is not very efficient however, as we need to retrain the network for every topology and test for overtraining.
- ▶ A better approach is to use *regularization*
- ▶ Regularization techniques were originally constructed to address ill-posed problems, for instance solving systems of linear equations $Ax = y$ where the matrix A is almost non-invertible.
- ▶ Regularization techniques can also be used to control the flexibility of a machine learning system

Common techniques - Regularization

L_p -regularization:

- ▶ In L_p -regularization we add a *penalty* to our optimization criterion

$$L(W, B, X, Y) = \frac{1}{|X|} \sum_n I(y_n, f(x_n; W, B)) + \lambda_W \|W\|_p + \lambda_B \|B\|_p$$

- ▶ The most common values for p are $p = 1, 2$, which corresponds to absolute values or sum of squares.
- ▶ Adding the L_p term will penalize the expression into avoiding large values for w_{ij}^l and b_i^l .
- ▶ This will reduce the flexibility of the network and avoid overfitting

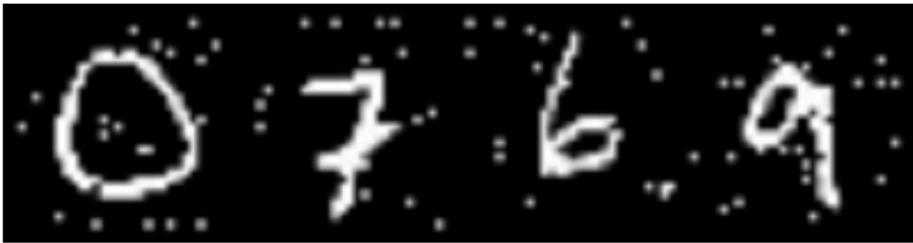
Common techniques - Data augmentation

- ▶ Sometimes the reason for overtraining is the lack of data, in the sense that the data is not spanning the space of possible observations
- ▶ We can *augment* the data set by adding carefully (and sometimes not so carefully), perturbed versions of existing data



Common techniques - Noisy augmentation

- ▶ We can also augment the data by adding noise to the training examples



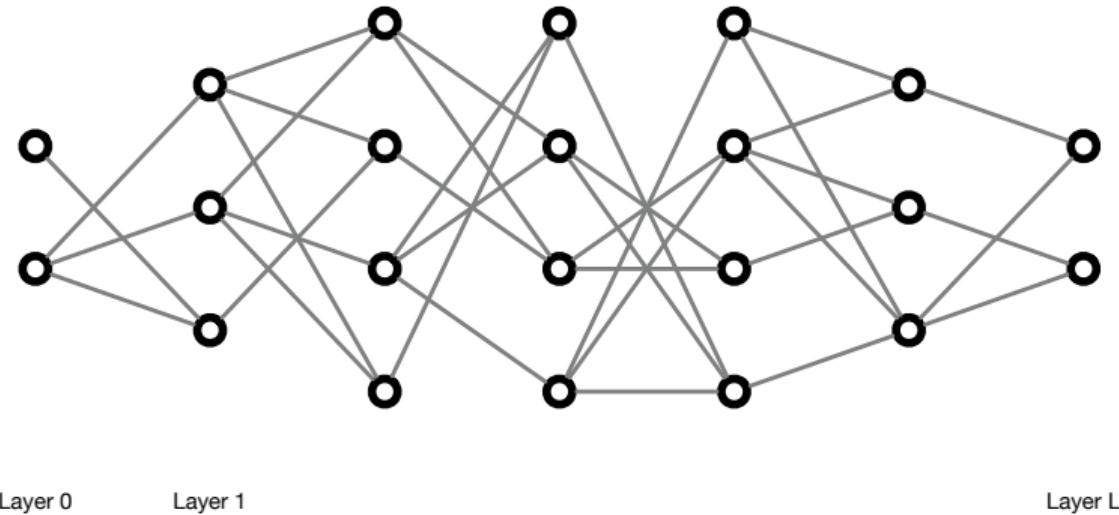
- ▶ Adding noise may seem counter productive, but if done correctly it will provide some *smoothing* of the learning machine, as it will need to fit the model to a large number of noisy examples instead of tuning the model to a small number of clean examples.

Common techniques - Dropout

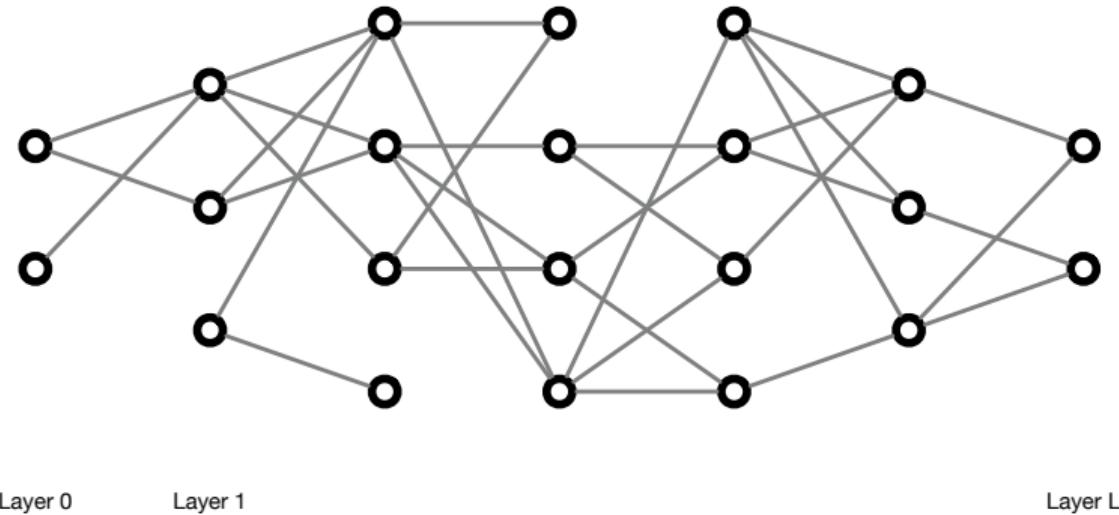
One of the most popular regularization methods is *dropout*

- ▶ For every iteration where the model is to be updated, remove a fraction c of the weights, so that they don't contribute
- ▶ Choose the weights to remove randomly every iteration

Common techniques - Dropout



Common techniques - Dropout



Common techniques - Dropout

- ▶ Dropout is in a sense a type of additive noise, behaving similarly to the noise augmentation training
- ▶ In addition, dropout *breaks dependencies*
- ▶ In principle, if two weights are summed at a processing node, the network may become dependent on the *difference* $w_{ij}x_i + w_{kj}x_k$.
- ▶ Dependencies like this are unwanted for a number of reasons. The weights may for instance become unbounded, while still computing a valid difference *for the training set*

Common techniques - Gradient search

There are many variations on the concept of gradient search:

- ▶ Simple gradient search

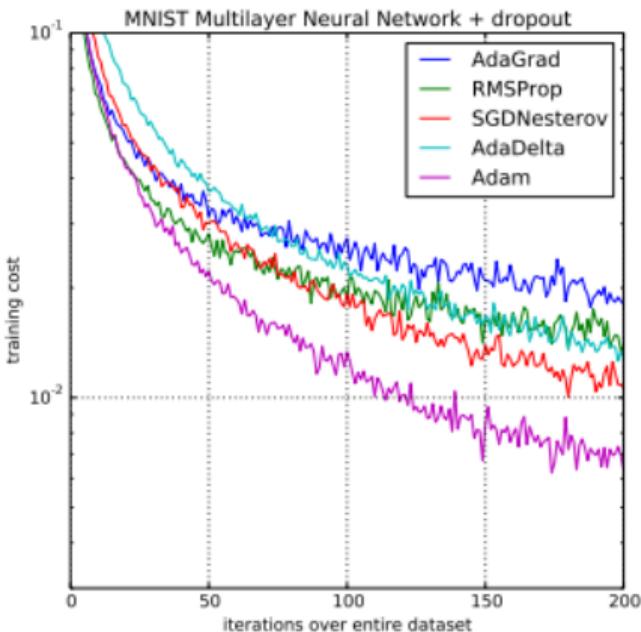
$$x^{i+1} = x^i - \epsilon \nabla_x f(x)$$

- ▶ Time dependent gradient search

$$x^{i+1} = x^i - \epsilon_i \nabla_x f(x)$$

- ▶ Gradient search with momentum (Nesterov)
- ▶ Adaptive learning algorithms: Adam, AdaGrad, AdaDelta...

Common techniques - Gradient search



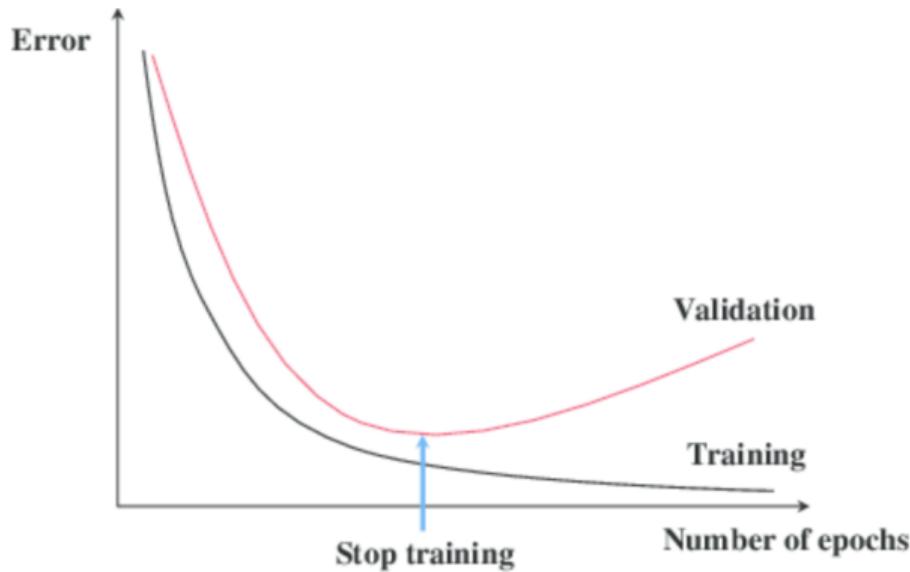
Best practice - Train, validation and test

One of the most important practices when doing any machine learning work is to define and use the following data sets correctly:

- ▶ Training data: This is the data that the model is based on directly. We use this data to compute gradients and update the model.
- ▶ Test data: This is the ultimate indicator of the performance of our model. This data should never be used in training, nor should it be used to optimize any hyper-parameters of the model.
- ▶ Validation data: This data set is to tune hyper-parameters, like model topology, learning rates, dropout rates and so on. It is never used to compute gradients and as such directly contributing to the model.

Best practice - Use of validation set - early stopping

The validation set is often used to tune regularization parameters. One example is *early stopping*



Best practice - Cross-validation

If resources permit it, it is beneficial to use *N-fold cross-validation*

- ▶ Take the training set and divide it into N equal parts
- ▶ Use one of the parts as validation data set
- ▶ Perform whatever performance tuning needed using the other $N - 1$ parts as the training set, measuring the performance on the validation set.
- ▶ Do this N times.
- ▶ Choose optimal parameters by eg. majority vote, train the system using all available data and test on the test set

Best practice - Accuracy, precision, recall

For many machine learning tasks, the data may be severely skewed

		Predicted/Classified	
		Negative	Positive
Actual	Negative	998	0
	Positive	1	1

In this case, even a classifier that *always* guesses negative, will be right in 99.8% of the cases.

But is this a good classifier?

Best practice - Accuracy, precision, recall

Let us define accuracy, precision and recall in terms of the confusion matrix

		Predicted	
		Negative	Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

Best practice - Accuracy, precision, recall

Definitions:

- ▶ Accuracy

$$\text{Accuracy} = \frac{\text{True positive} + \text{True negative}}{\text{Number of test examples}}$$

- ▶ Precision:

$$\text{Precision} = \frac{\text{True positive}}{\text{True positive} + \text{False positive}} = \frac{\text{True positive}}{\text{Total positive predictions}}$$

- ▶ Recall:

$$\text{Recall} = \frac{\text{True positive}}{\text{True positive} + \text{False negative}} = \frac{\text{True positive}}{\text{Total positive examples}}$$

Best practice - Accuracy, precision, recall

		Predicted/Classified	
		Negative	Positive
Actual	Negative	998	0
	Positive	1	1

- ▶ Accuracy: 0.999 (99.9 %)
- ▶ Precision: 1.0 (100 %)
- ▶ Recall: 0.5 (50 %)

Thank you for your attention



NTNU | Norwegian University of
Science and Technology



DEEP LEARNING CONVOLUTIONAL NEURAL NETWORKS

Tor Andre Myrvoll
2020-10-16

Introduction - Outline

Convolutional Neural Networks (CNNs)

- ▶ Motivation
- ▶ Basic CNNs
- ▶ Transfer learning

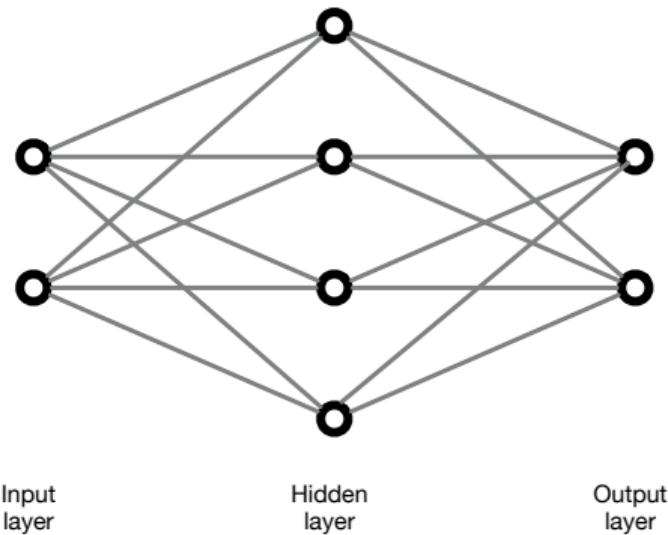
Introduction - Motivation



Introduction - Motivation

MLP Specification

- ▶ Binary classifier
- ▶ 512×512 images
- ▶ One hidden layer with as many nodes as there are inputs



Introduction - Motivation

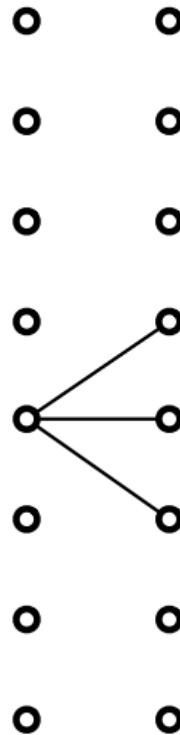
Reality catches up...

- ▶ We stack the image into a columnvector of dimension $512 \times 512 = 262144$. Large, but doable
- ▶ The number of hidden nodes will also be 262144 dimensions
- ▶ The number of connections from input to output will be $262144^2 = 68719476736$
- ▶ Which is roughly 268 GB of data is using float
- ▶ Game over?

Introduction - Solutions

Only use a subset of the connections

- ▶ Using $k \ll N$ inputs will decrease number of parameters
- ▶ Ex. $k = 3 \Rightarrow k \cdot N = 786432$, a reduction of factor 90000



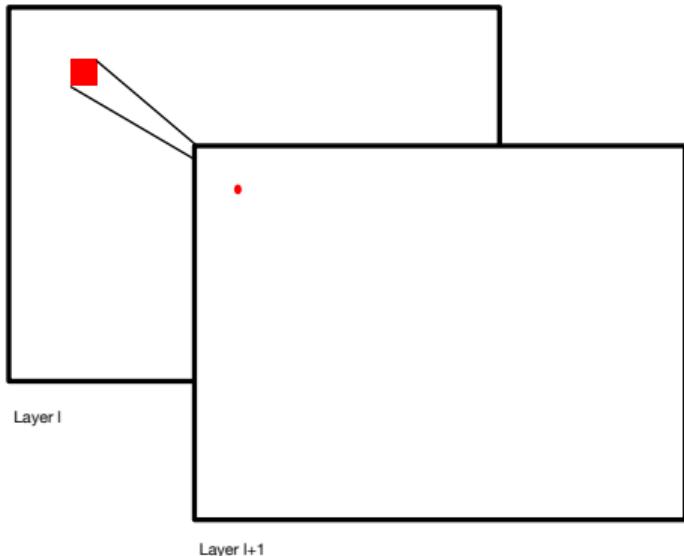
Introduction - The CNN

For images it makes sense to use local information

- ▶ The excitation in the $l + q$ layer is given as

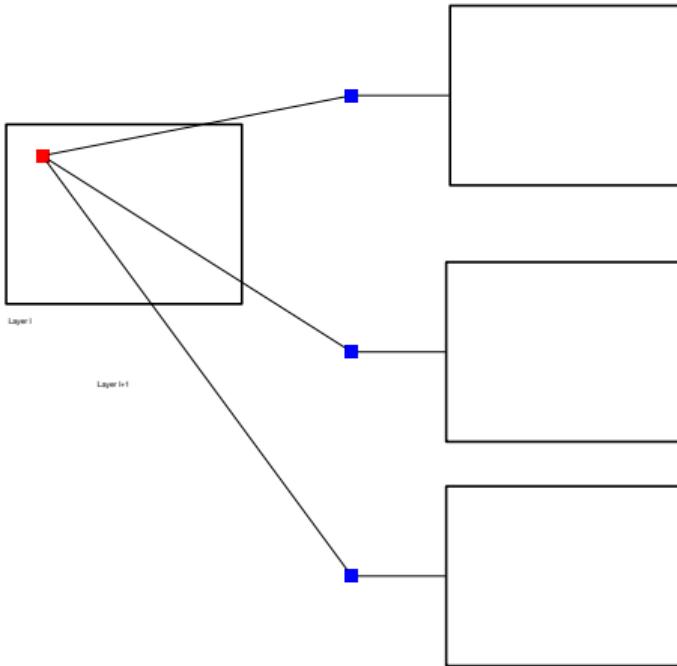
$$z_{i,j}^{l+1} = \sum_{m,n=-L/2}^{L/2} k_{m,n} v_{i-m,j-n}^l$$

- ▶ We see that this corresponds to a *convolution*, hence the naming scheme



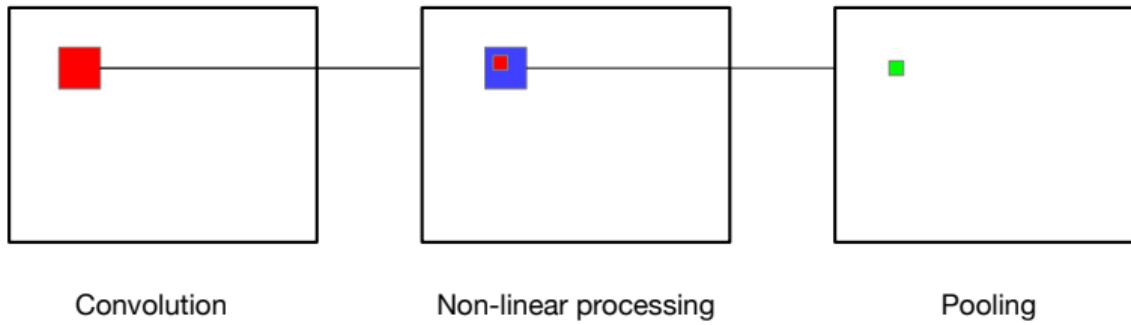
Introduction - The CNN

We can use multiple filters



Introduction - The CNN

In addition to convolution we do non-linear processing and *pooling*



Introduction - The CNN

The three processing steps that defines a CNN *layer* are:

- ▶ Convolution

$$z_{i,j}^{l+1} = \sum_{m,n=-L/2}^{L/2} k_{m,n} v_{i-m,j-n}^l$$

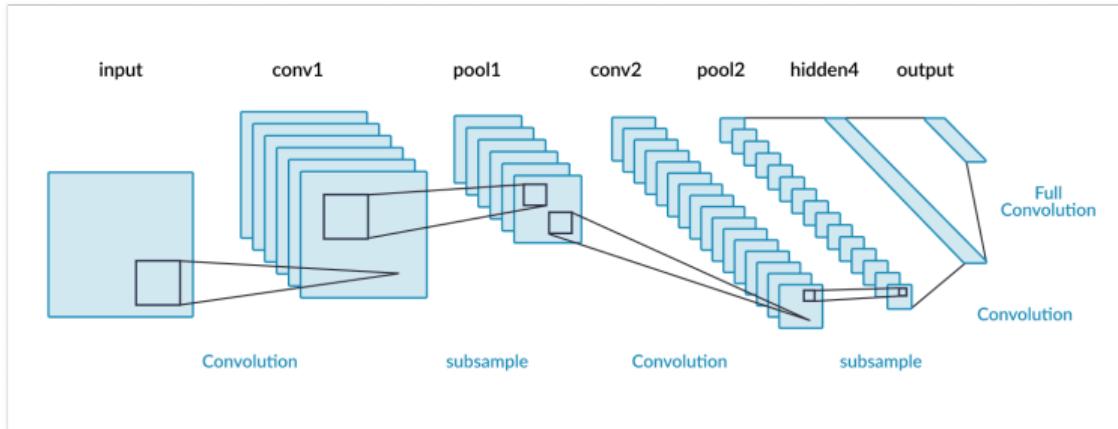
- ▶ Non-linear processing: ReLU, Sigmoid, etc.
- ▶ Pooling: Representing local areas by some statistic
 - ▶ Examples: Mean value, median, maximum value, power

$$z_{i,j}^l = \max\{v_{i-m,j-n}\}_{m,n=-L/2}^{L/2}$$

- ▶ Pooling leads to dimensionality reduction, and implicit translation and rotational invariance

Introduction - CNN example - LeNet

Early example of CNN – LeNet-5. Handwritten character recognition:



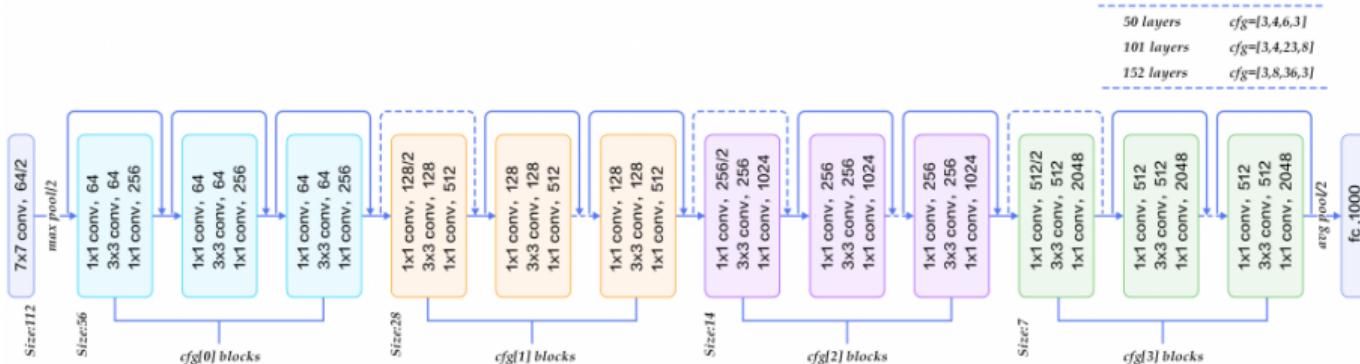
Introduction - CNN example – LeNet

Structure:

- ▶ Layer 1: Convolution. Six 5×5 kernels
- ▶ Layer 2: Pooling: $32 \times 32 \rightarrow 14 \times 14$
- ▶ Layer 3: Convolution. 16 5×5 kernels
- ▶ Layer 4: Pooling: $10 \times 10 \rightarrow 5 \times 5$
- ▶ Layer 5: Convolution: 120 5×5 kernels
- ▶ Layer 6: Fully connecter. 84 features output.

Introduction - CNN example - ResNet

Massive 152 layer network for image classification



Introduction - Transfer learning

Training a large CNN is a massive undertaking. Use *transfer learning* to use a previously training CNN.

- ▶ Only use the CNN layers of a model
- ▶ This part of the model is often called a *feature extractor*
- ▶ Define and train your own fully connected layers after the CNN (don't update the CNN parameters)
- ▶ Available from for instance the Keras Applications API
- ▶ Available from Apple when using the COre ML API

Thank you for your attention



NTNU | Norwegian University of
Science and Technology



DEEP LEARNING APPLICATIONS AND ARCHITECTURES

Tor Andre Myrvoll

October 23, 2020

Introduction - Outline

In this lecture we will look at some useful applications of neural networks, as well as some special architectures

- ▶ Recurrent Neural Networks (RNN)
- ▶ Autoencoders
- ▶ Generative Adversarial Networks (GAN)

Recurrent Neural Networks - Basic RNNs

Many problems are not amiable to be modeled using a simple feed-forward network:

- ▶ Language modeling, text generation
- ▶ Machine translation
- ▶ Speech recognition
- ▶ Video tagging
- ▶ Prediction problems: $\hat{y}_n = f(y_{n-1}, w_n)$

Recurrent Neural Networks - Basic RNNs

A simple multilayer perceptron takes an input of fixed dimension and produces an output.

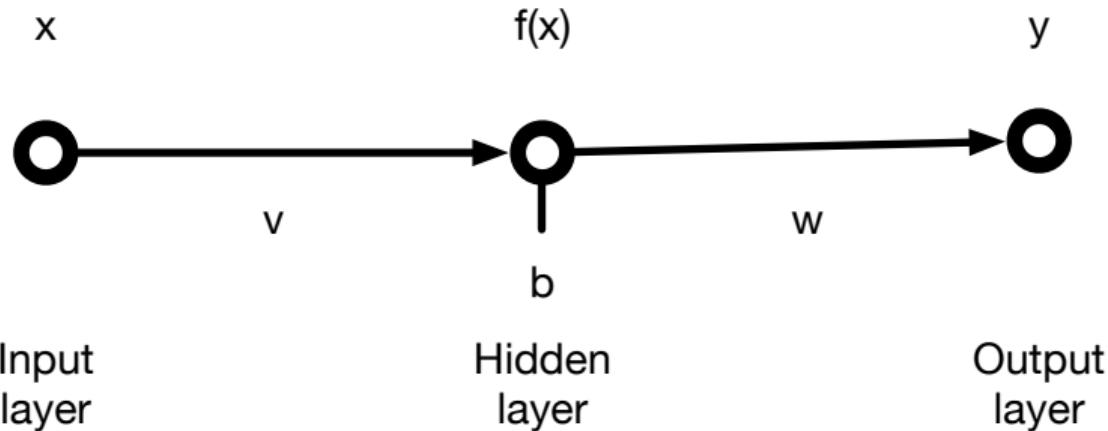
There is no memory involved – sequences of inputs can be presented in arbitrary order, giving the same results.

We need some way to keep the *state* of the system between examples.

This can be achieved by using the hidden layers as memory cells as well as processing units.

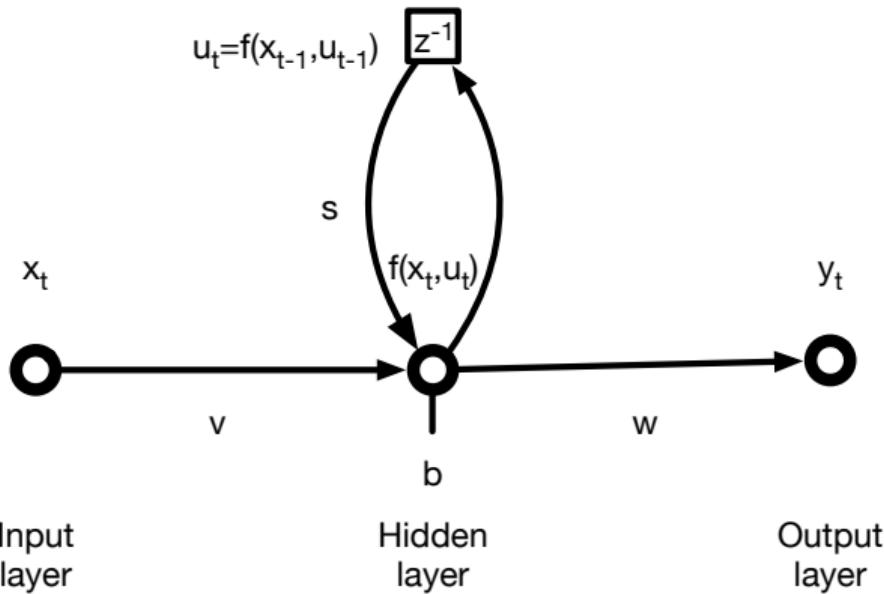
Recurrent Neural Networks - Recursive neural networks

Consider the simplest MLP possible:



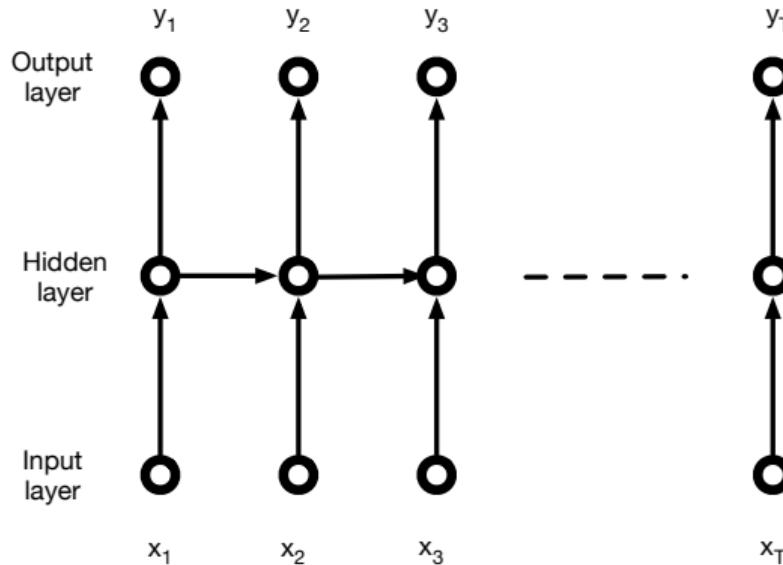
Recurrent Neural Networks - Recursive neural networks

Extending the simplest MLP possible:



Recurrent Neural Networks - Recursive neural networks

Training the basic RNN is done by unfolding the graph, then using back propagation



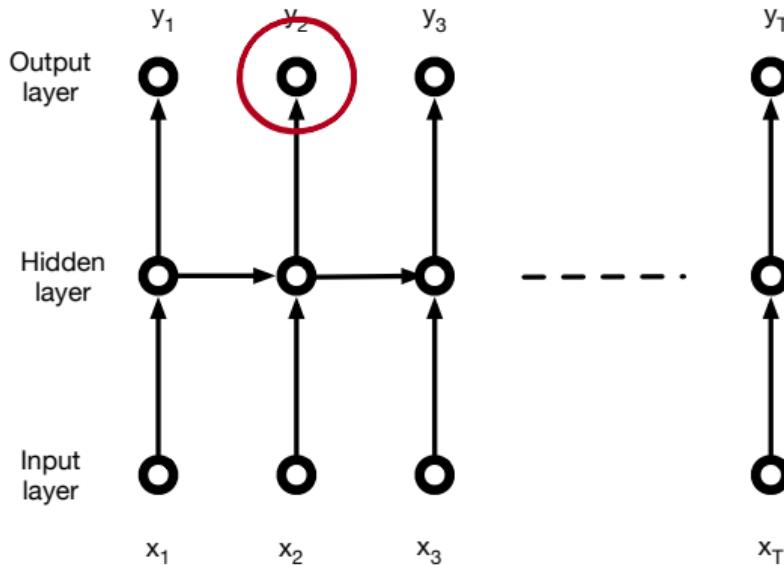
Recurrent Neural Networks - Recursive neural networks

One drawback of the simple recursive network is the inability to utilize future dependencies. This is needed for instance in machine translation, where words can be out of order

- ▶ “I like icecream” ⇒ “Jeg liker iskrem”
- ▶ “I like icecream” ⇒ “Watashi wa aisukurīmu ga suki”

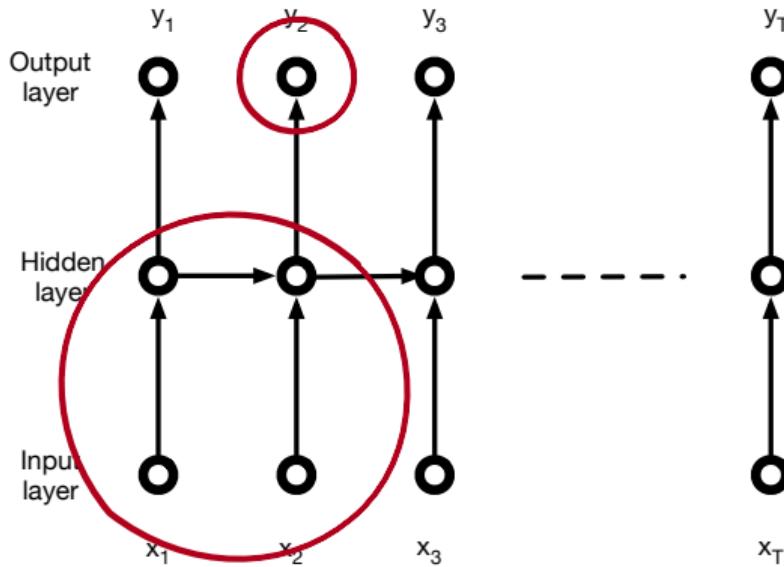
Recurrent Neural Networks - Recursive neural networks

One drawback of the simple recursive network is the inability to utilize future dependencies



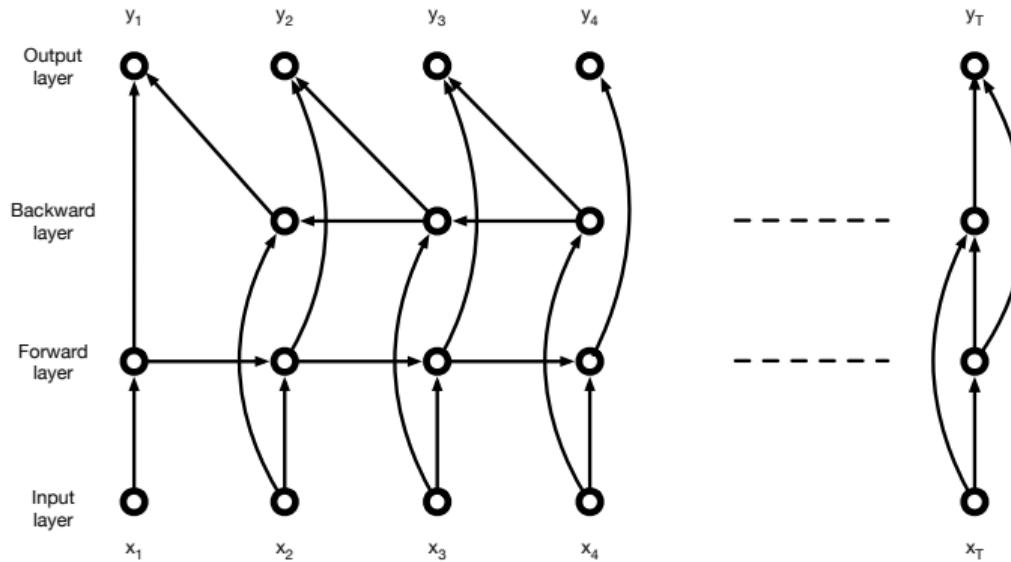
Recurrent Neural Networks - Recursive neural networks

One drawback of the simple recursive network is the inability to utilize future dependencies



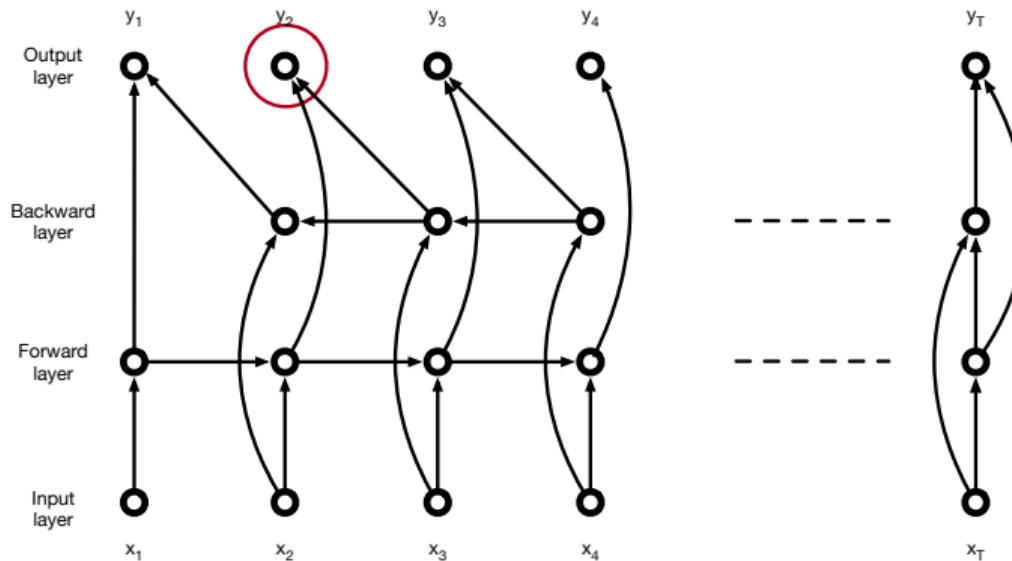
Recurrent Neural Networks - Recursive neural networks

The bi-directional RNN addresses this issue



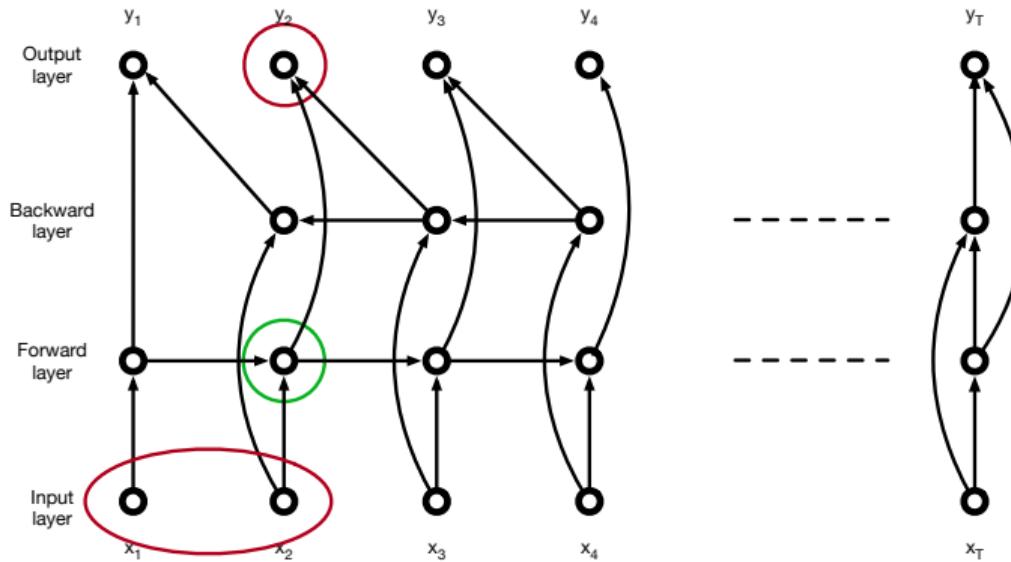
Recurrent Neural Networks - Recursive neural networks

The bi-directional RNN addresses this issue



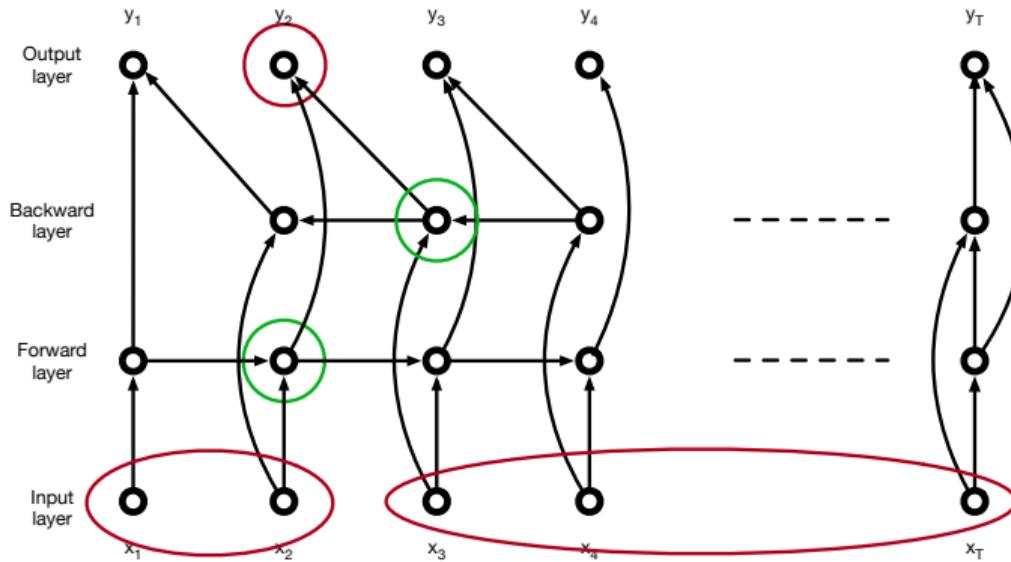
Recurrent Neural Networks - Recursive neural networks

The bi-directional RNN addresses this issue



Recurrent Neural Networks - Recursive neural networks

The bi-directional RNN addresses this issue

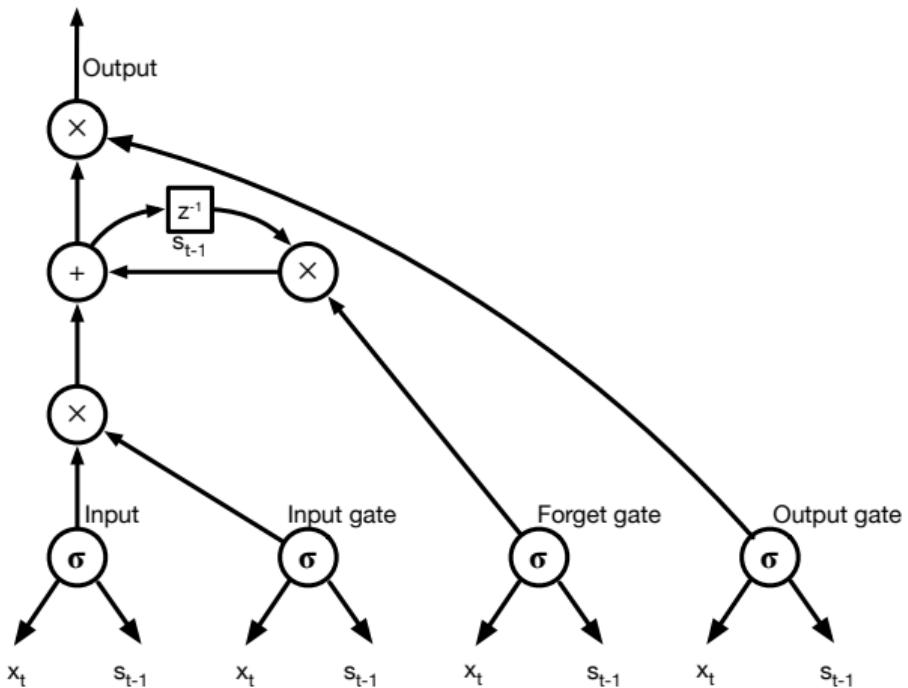


Recurrent Neural Networks - Long short-term memory

The basic recursive neural network has several issues

- ▶ Many problems will have more inputs than outputs or vice versa
 - ▶ Machine translation
 - ▶ Text generation
- ▶ Long dependencies are hard to model
 - ▶ Example – word prediction
 - ▶ “French is the language I am most fluent in as I was born and raised in ...”
 - ▶ Here the most important information for the prediction, “French”, is 15 words ahead of the word to be predicted (most likely “France”)
- ▶ Training is hard due to the vanishing gradient problem
- ▶ The *Long short-term memory* (LSTM) network addresses and solves many of these issues

Recurrent Neural Networks - Long short-term memory



Recurrent Neural Networks - Other architectures

There are a large variety of RNN architectures found in the literature. Some examples:

- ▶ Gated recurrent unit: Simplified LSTM with no output gate. Better performance on some smaller datasets.
- ▶ Encoder-decoder network: Encodes an entire sequence into a neural network “memory”, then generates a new “translated” sequence using a second neural network.
- ▶ Transformer networks: Current state-of-the-art in machine translation loosely based on the encoder-decoder principle. Has mostly replaced LSTM in modern systems.

Autoencoders - Motivation

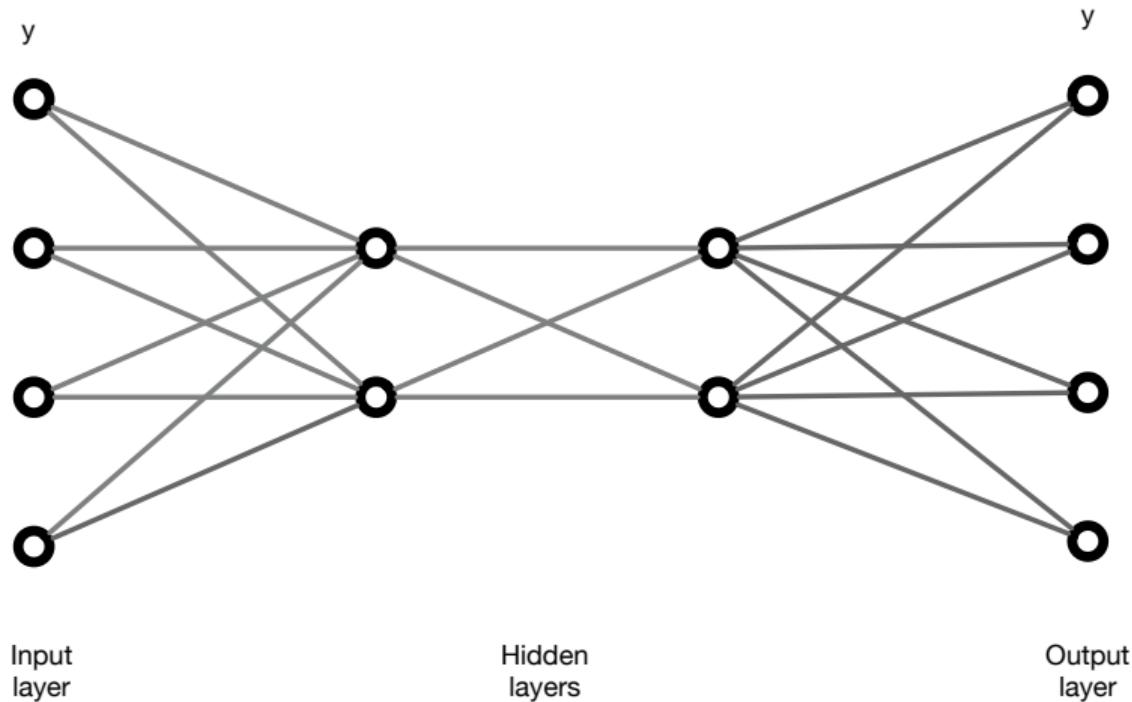
The autoencoder structure is a neural network that learns to estimate

$$\hat{y} = f(y)$$

In other words, it tries to learn a mapping from itself to itself.

What is the point?

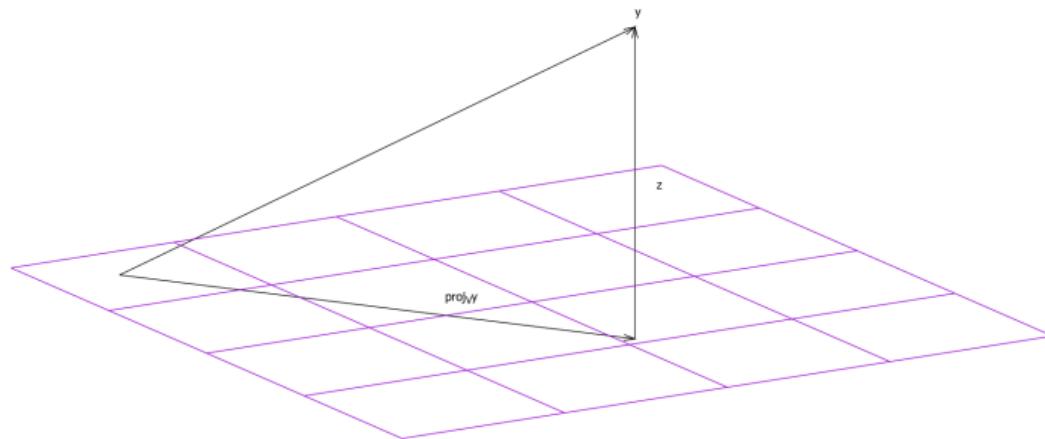
Autoencoders - The bottleneck



Autoencoders - Manifold projection

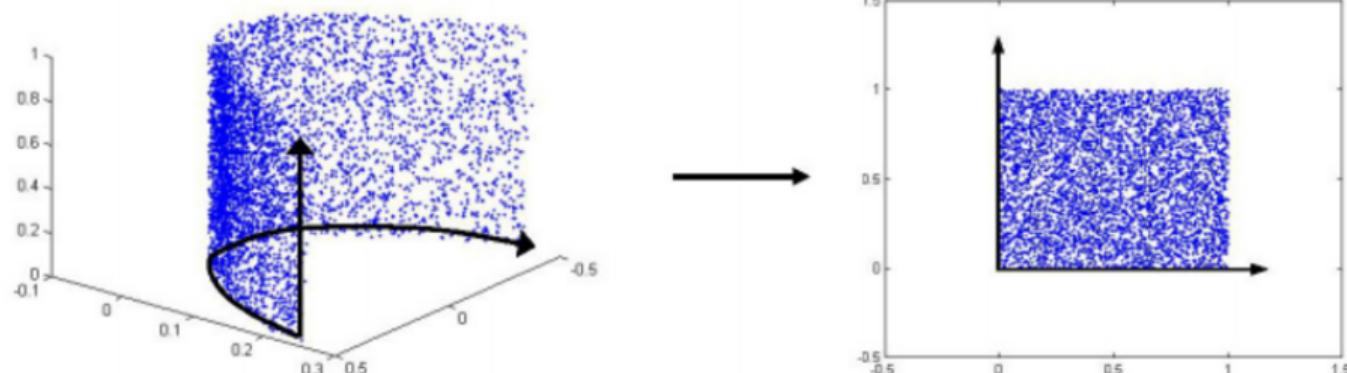
The main idea is to do the nonlinear equivalent of a linear projection

0 —



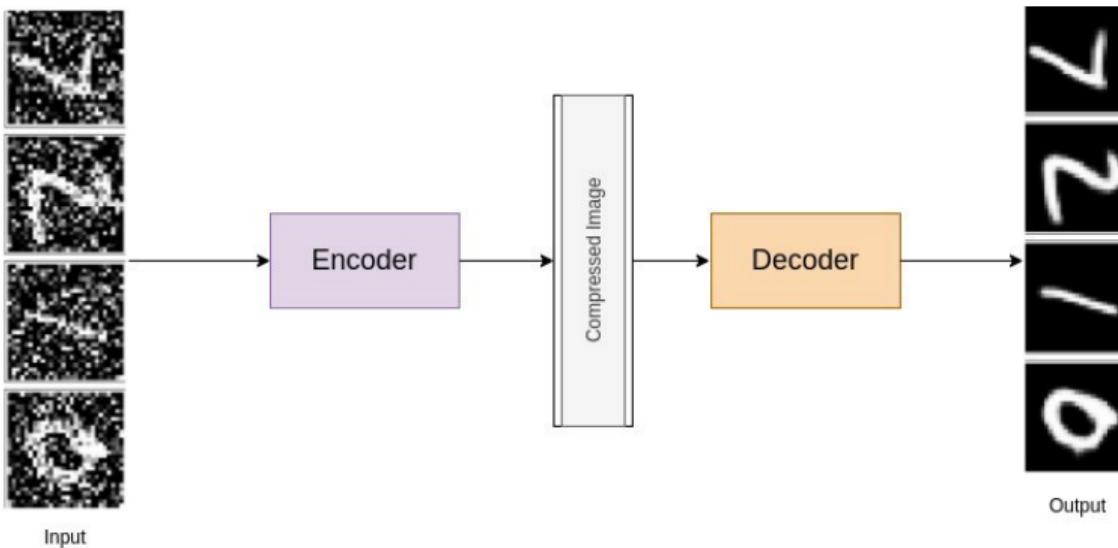
Autoencoders - Manifold projection

The main idea is to do the nonlinear equivalent of a linear projection



Autoencoders - The denoising autoencoder

An alternative to the bottleneck is to add noise to the data and then learn the network to clean it up. This will force the neural network to learn the underlying representation of the signal



Autoencoders - Applications

- ▶ Feature extraction: Use an autoencoder to find a low-level representation of the data. Then use the first half of the network as a feature extractor
- ▶ Speech enhancement: Noisy spectrograms are used as inputs, and the NN learns to recover the clean spectrograms
- ▶ Anomaly detection: Clean data is used to train a predictor $\hat{y}_{n+1} = f(y_1^n)$. If the example data is different from the training data, the variance of the predictor will increase, indicating an anomaly.

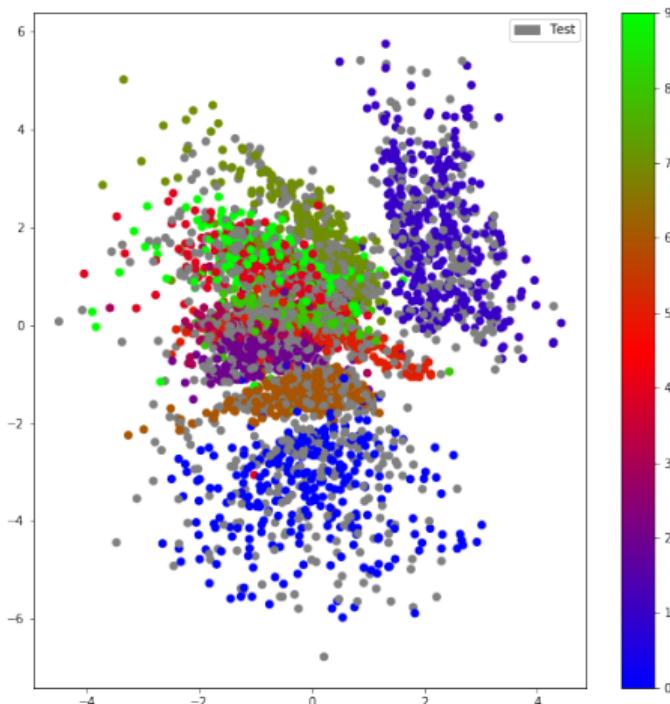
Autoencoders - Example (from kaggle.com)

The MNIST handwritten digits are 28×28 pixels \Rightarrow 784 dimensional input



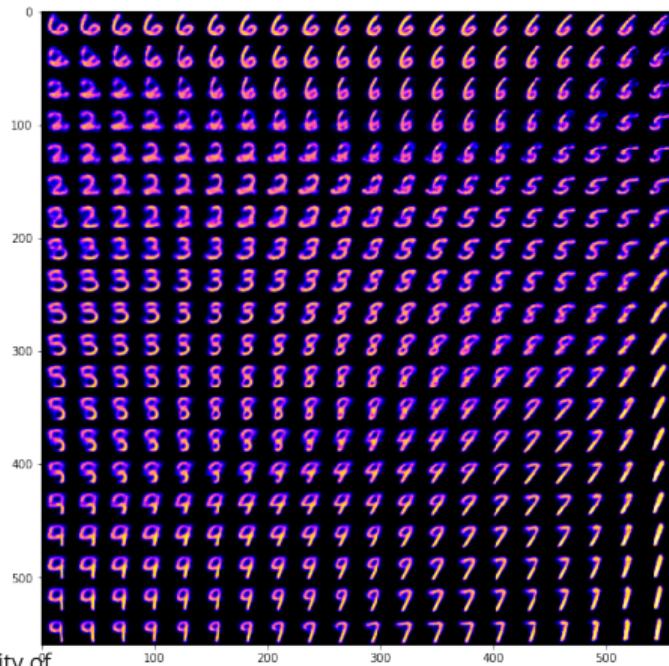
Autoencoders - Example (from kaggle.com)

We can use an autoencode to project it into a 2-dimensional space



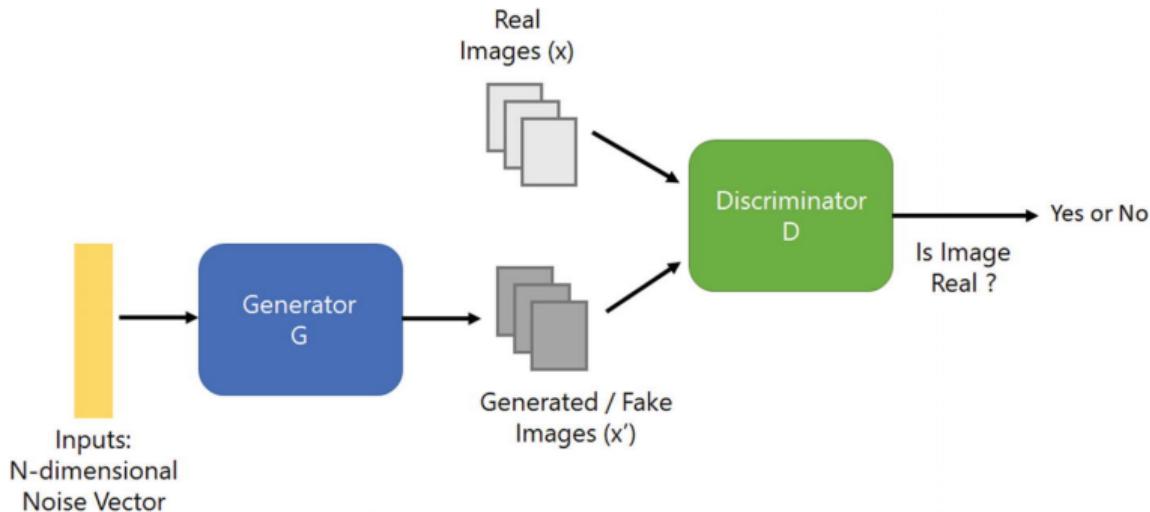
Autoencoders - Example (from kaggle.com)

Using the trained autoencoded we can generate new digits using (x, y) as inputs



Generative Adversarial Networks - Simple GAN

In its simplest form, a GAN learns to generate data from an empirical distribution:



Generative Adversarial Networks - Simple GAN

Training a GAN can be formulated as a game:

- ▶ The true distribution is represented by an empirical distribution given as p_{data}
- ▶ The adversarial distribution is p_g , which is obtained by transforming noise z drawn from $p_z(z)$ by a neural network $G(z; \theta_g)$
- ▶ The detector is a neural network $D(x; \theta_d)$, that outputs the probability of the data x being drawn from the *true* distribution $p_{\text{data}}(x)$, and not $p_g(x)$
- ▶ We train $\{\theta_g, \theta_d\}$ jointly by playing the minimax game

$$\min_G \max_D \mathbb{E}_{x \sim p_{\text{data}}} [\log D(x)] + \mathbb{E}_{z \sim p_z} [\log (1 - D(G(z)))]$$

Thank you for your attention



NTNU | Norwegian University of
Science and Technology

Graphical Models

TTT4185 Machine Learning for Signal Processing

Giampiero Salvi

Department of Electronic Systems
NTNU

HT2021

Course plan (So far)

- Speech Perception and Production
- Speech Analysis and Feature Extraction (Ex. 1)
- Intro to Machine Learning/Probability and Information Theory
- Linear Models for Regression and Classification
- Kernel Methods and Support Vector Machines (Ex. 2)
- Deep Neural Networks (Ex. 3)

Course plan (Rest of the lectures)

- Graphical Models (Bayesian Networks) 2h
- Unsupervised Learning (k-means and Mixture Models) 2h
- Sequences and Hidden Markov Models 4h
- Dimensionality Reduction 2h
- Summary 1h

Guest lecture!

Pablo Ortiz, Telenor Research

2020-11-24

Computer Exercises: Orals

- You will be able to book a time from week 46
- The focus will be on understanding rather than implementation
- More information will be given by the TAs

Graphical Models: Motivation

So far (supervised learning)

- \mathbf{x}_i input, t_i output
- goal: estimate $f(\mathbf{x})$ that approximates relationship between \mathbf{x} and t

We would like to consider probabilistic models more in general

- given a set of random variables $\mathbf{x}_1, \dots, \mathbf{x}_K$
- describe the joint probability distribution $p(\mathbf{x}_1, \dots, \mathbf{x}_K)$

Graphical Models:

- simple way to visualize **structure** in probabilistic models
- insights into the properties of the model (conditional independence)
- complex computations expressed in terms of graphical manipulations

Graphical Models

Definition

- Each node corresponds to a random variable
- Each link (edge or arc) represents probabilistic relationship between variables
- Graph: how to decompose the joint distribution into factors

Bayesian Networks:

- Directed graphs
- useful to express causal relationships

Markov Random Fields:

- Undirected graphs

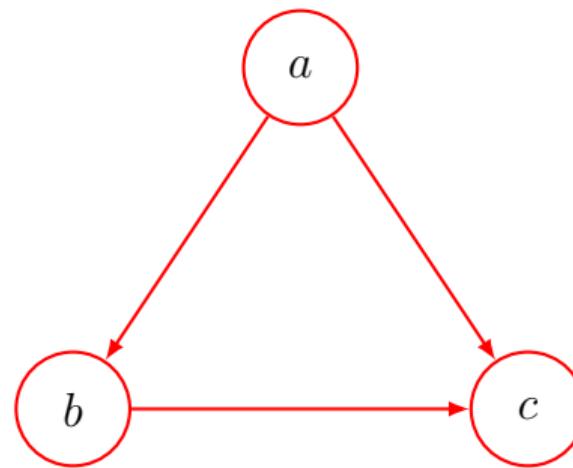
Factor Graphs:

- Mainly used for calculations

Bayesian Network simple example

Three variables a, b , and c .

$$\begin{aligned} p(a, b, c) &= p(c|a, b)p(a, b) \\ &= p(c|a, b)p(b|a)p(a) \end{aligned}$$



Bayesian Networks (example)

$$p(x_1, \dots, x_7) =$$

$$p(x_1)$$

$$p(x_2|x_1)$$

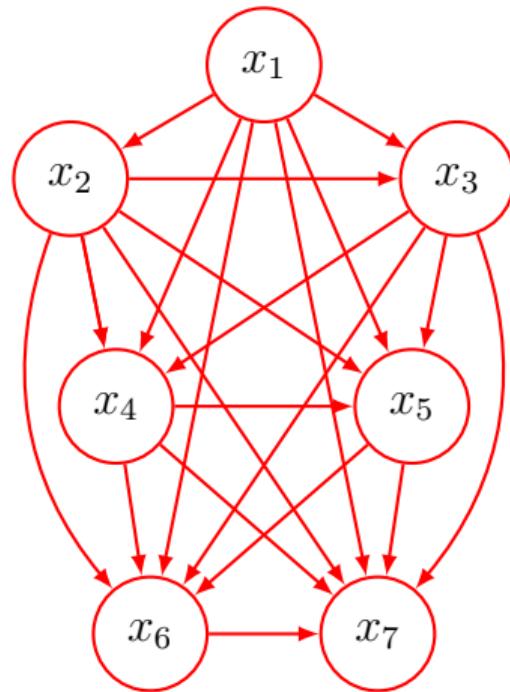
$$p(x_3|x_1, x_2)$$

$$p(x_4|x_1, x_2, x_3)$$

$$p(x_5|x_1, x_2, x_3, x_4)$$

$$p(x_6|x_1, x_2, x_3, x_4, x_5)$$

$$p(x_7|x_1, x_2, x_3, x_4, x_5, x_6)$$



Bayesian Networks (example)

$$p(x_1, \dots, x_7) =$$

$$p(x_1)$$

$$p(x_2|x_1)$$

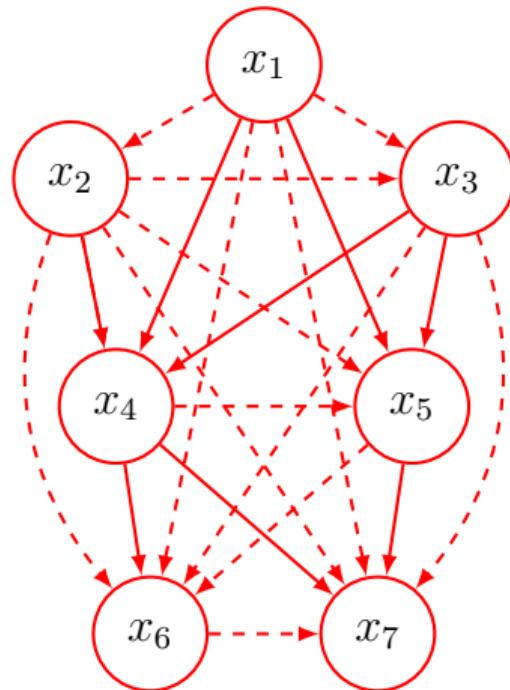
$$p(x_3|x_1, x_2)$$

$$p(x_4|x_1, x_2, x_3)$$

$$p(x_5|x_1, x_2, x_3, x_4)$$

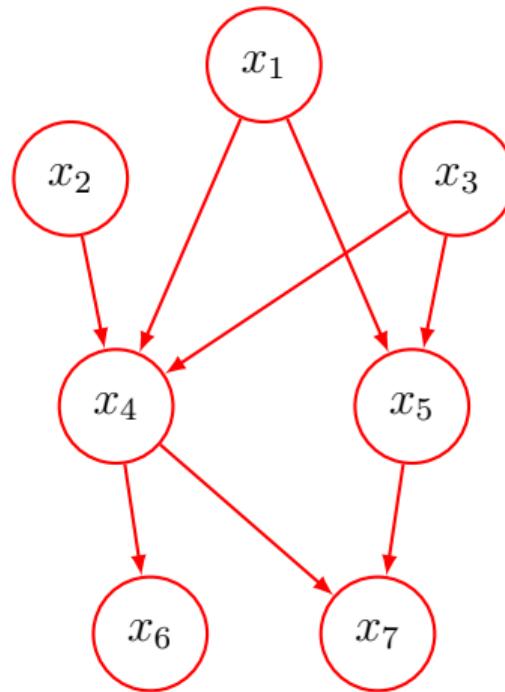
$$p(x_6|x_1, x_2, x_3, x_4, x_5)$$

$$p(x_7|x_1, x_2, x_3, x_4, x_5, x_6)$$



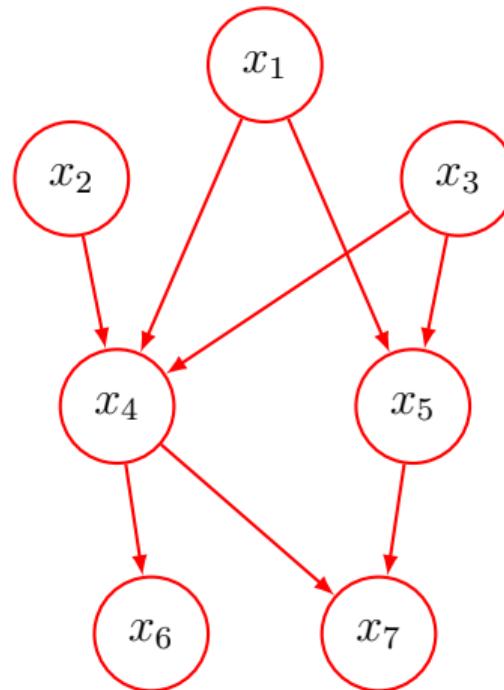
Bayesian Networks (example)

$$\begin{aligned} p(x_1, \dots, x_7) = \\ p(x_1) \\ p(x_2) \\ p(x_3) \\ p(x_4|x_1, x_2, x_3) \\ p(x_5|x_1, x_3) \\ p(x_6|x_4) \\ p(x_7|x_4, x_5) \end{aligned}$$



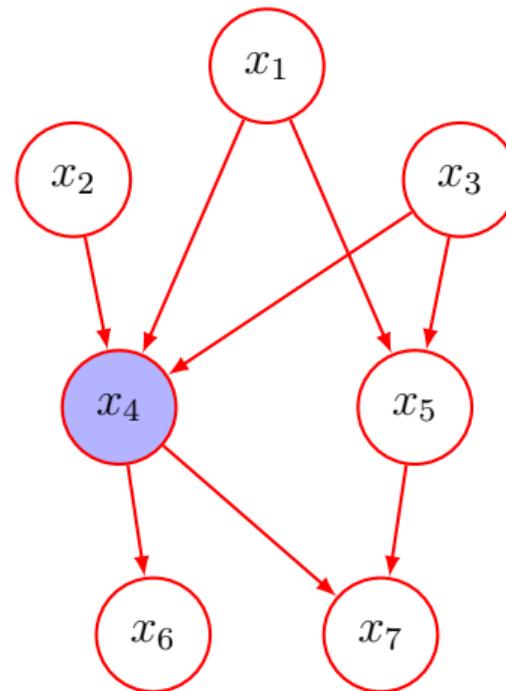
Bayesian Networks (example)

$$p(x_1, \dots, x_7) = \prod_{k=1}^K p(x_k | \text{pa}_k)$$



Bayesian Networks (example)

If we observe x_4 ...



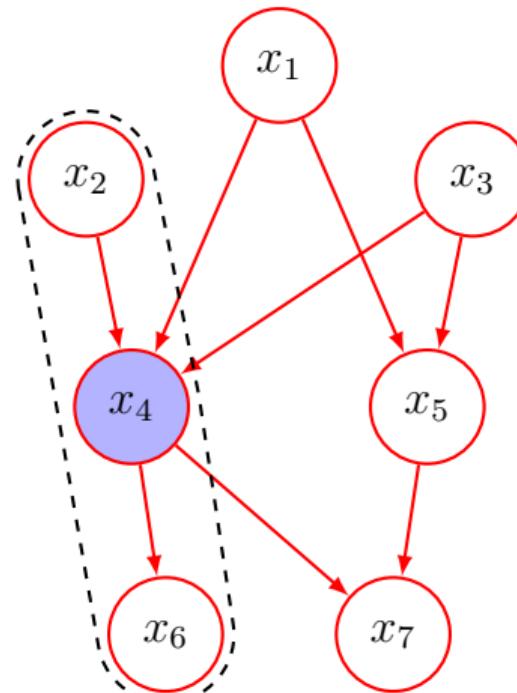
Bayesian Networks (example)

If we observe x_4 ...

d -separation:

Head-to-tail:

x_2 and x_6 conditionally independent



Bayesian Networks (example)

If we observe $x_4 \dots$

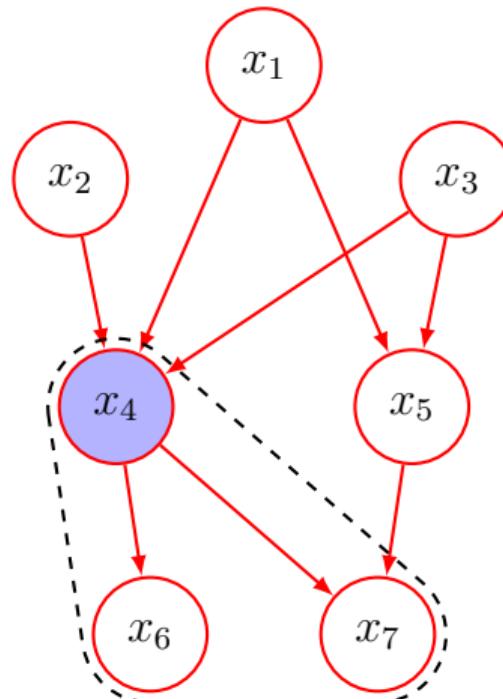
d -separation:

Head-to-tail:

x_2 and x_6 conditionally independent

Tail-to-tail:

x_6 and x_7 conditionally independent



Bayesian Networks (example)

If we observe x_4 ...

d -separation:

Head-to-tail:

x_2 and x_6 conditionally independent

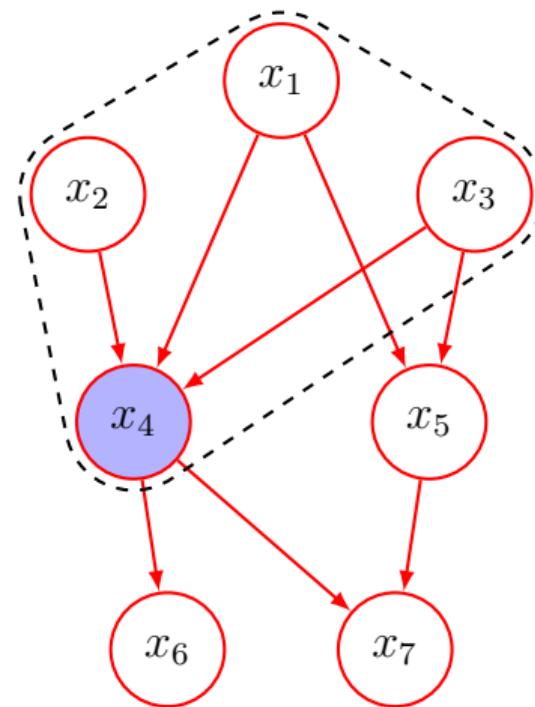
Tail-to-tail:

x_6 and x_7 conditionally independent

Head-to-head:

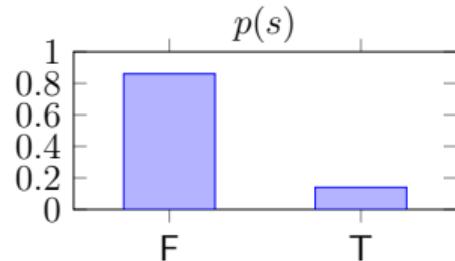
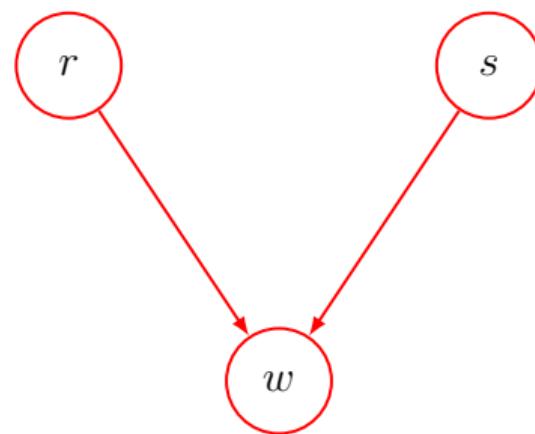
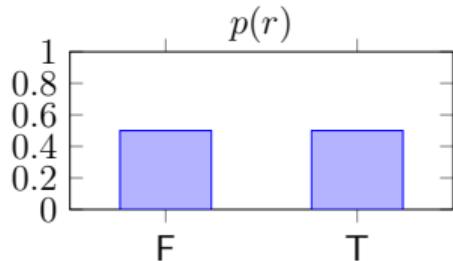
x_1, x_2 and x_3 dependent

(explaining away)



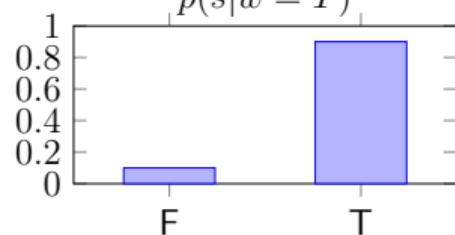
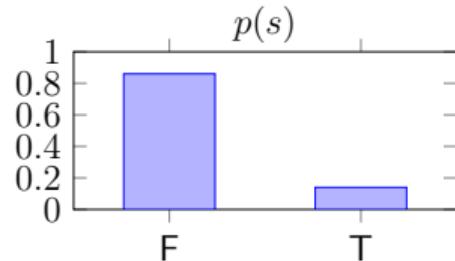
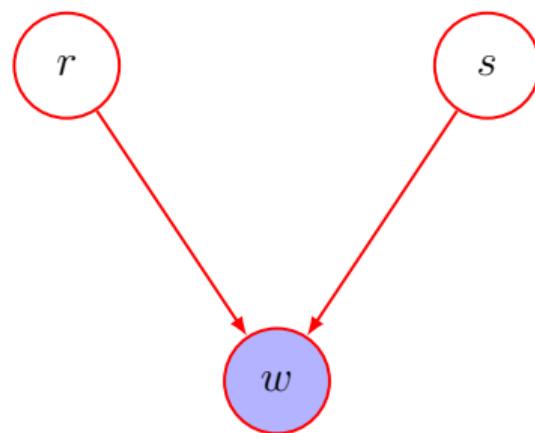
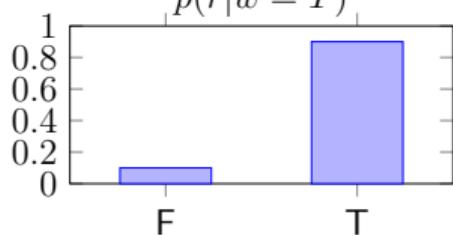
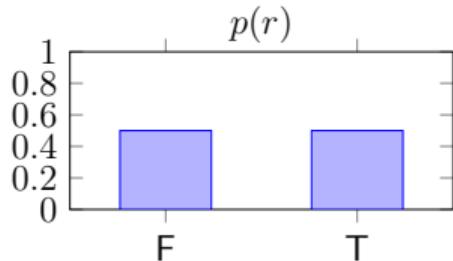
Explaining Away: Intuition

r : it rains, s : sprinkle is on, w : the grass is wet



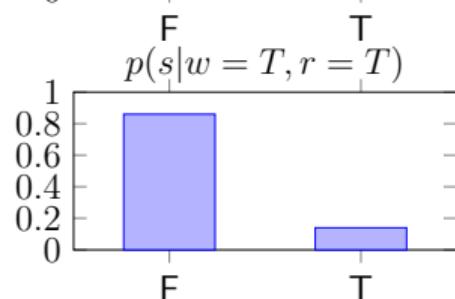
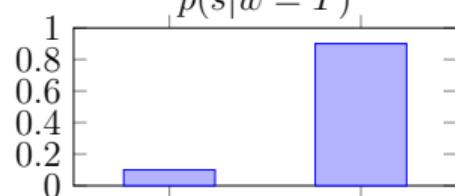
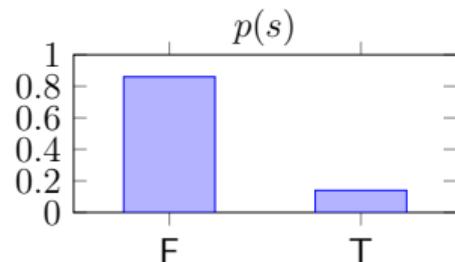
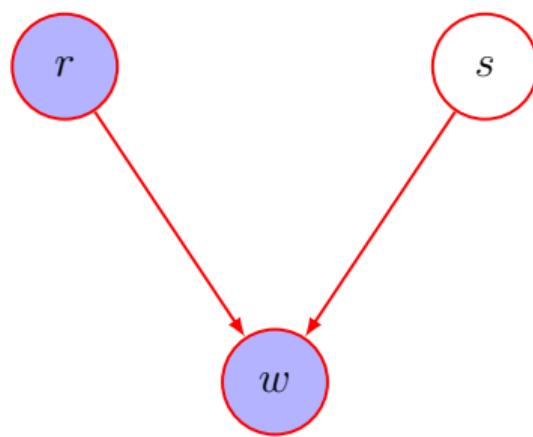
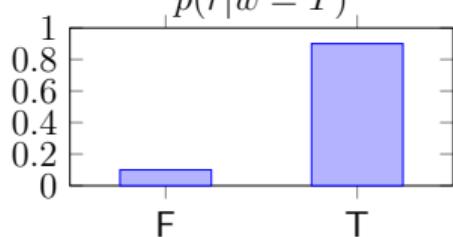
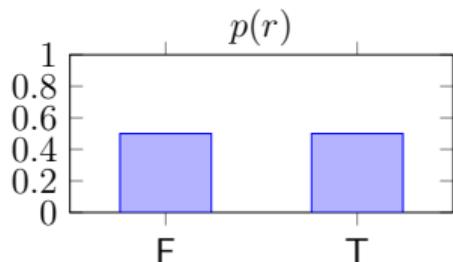
Explaining Away: Intuition

r : it rains, s : sprinkle is on, w : the grass is wet



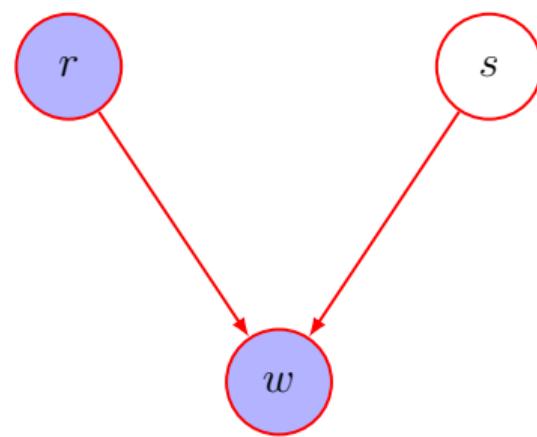
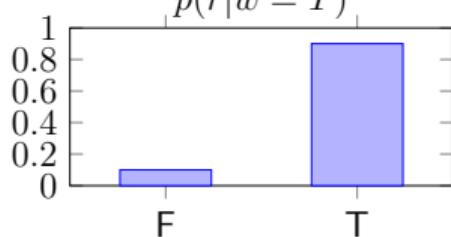
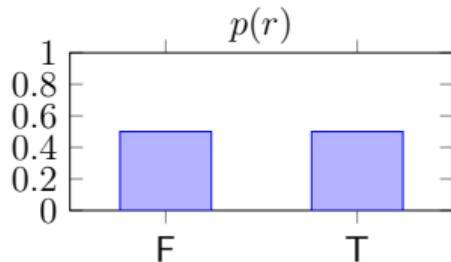
Explaining Away: Intuition

r : it rains, s : sprinkle is on, w : the grass is wet

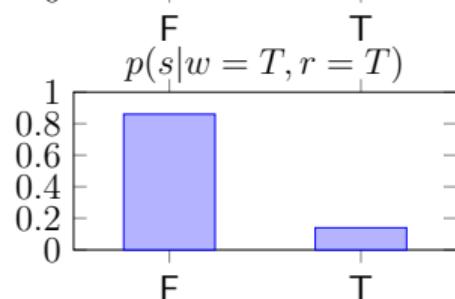
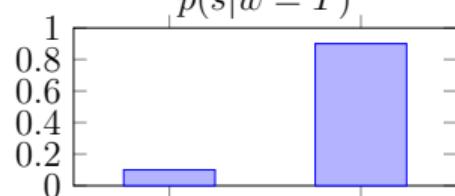
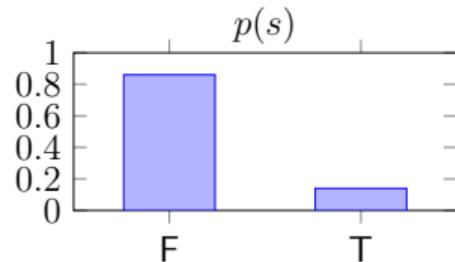


Explaining Away: Intuition

r : it rains, s : sprinkle is on, w : the grass is wet



$$p(s|w = T, r = T) \neq p(s|w = T)$$

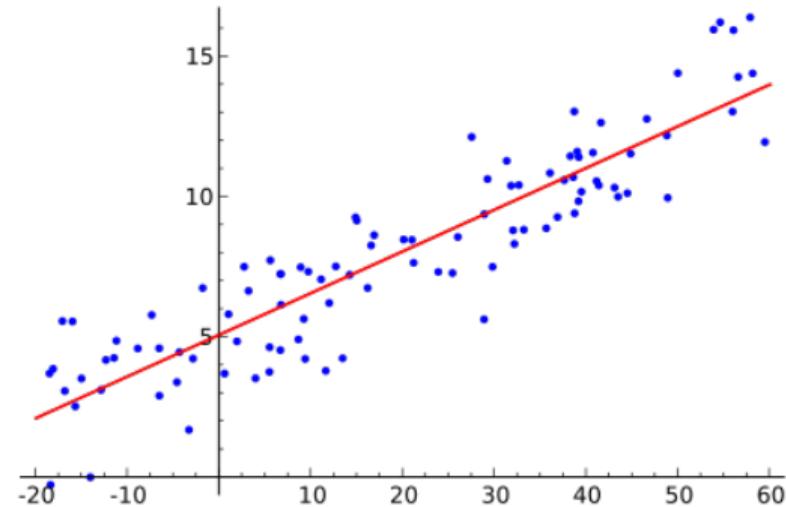


Example: MAP linear regression

Model:

$$p(t|\mathbf{w}, \mathbf{x}, \sigma^2) = \mathcal{N}(\mathbf{w}^T \mathbf{x}, \sigma^2)$$

$$p(\mathbf{w}|\alpha) = \mathcal{N}(0, \frac{1}{\alpha} \mathbf{I})$$

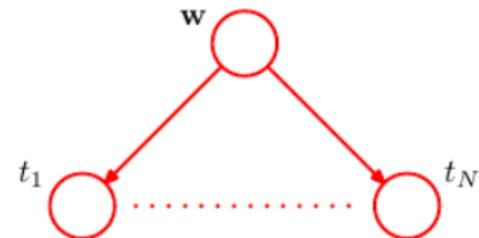


Example: MAP linear regression

$$\mathbf{t} = (t_1, \dots, t_n)^T$$

$$\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^T$$

$$p(\mathbf{t}, \mathbf{w}) = p(\mathbf{w}) \prod_{n=1}^N p(t_n | \mathbf{w})$$

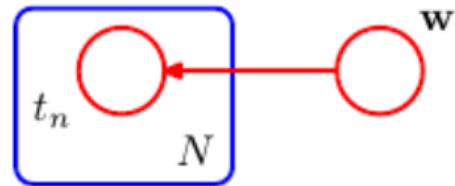


Example: MAP linear regression

$$\mathbf{t} = (t_1, \dots, t_n)^T$$

$$\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^T$$

$$p(\mathbf{t}, \mathbf{w}) = p(\mathbf{w}) \prod_{n=1}^N p(t_n | \mathbf{w})$$



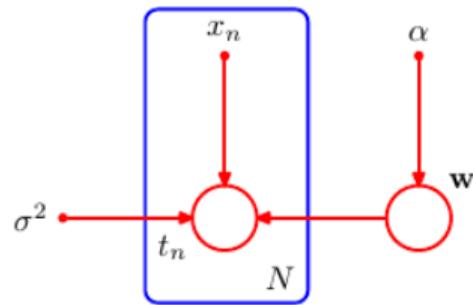
Example: MAP linear regression

$$\mathbf{t} = (t_1, \dots, t_n)^T$$

$$\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^T$$

$$p(\mathbf{t}, \mathbf{w}) = p(\mathbf{w}) \prod_{n=1}^N p(t_n | \mathbf{w})$$

$$p(\mathbf{t}, \mathbf{w} | \mathbf{X}, \alpha, \sigma^2) = p(\mathbf{w} | \alpha) \prod_{n=1}^N p(t_n | \mathbf{w}, \mathbf{x}_n, \sigma^2)$$

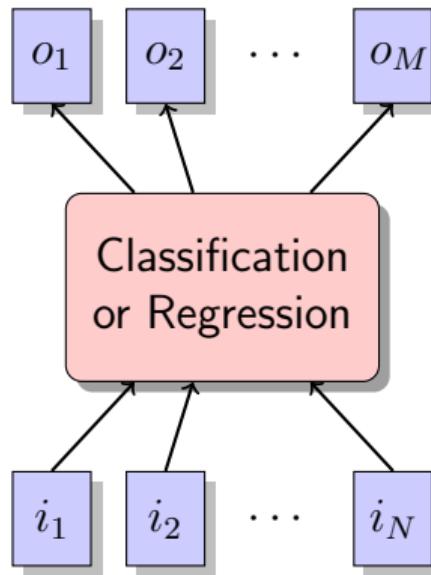


Generative Models: Ancestral Sampling

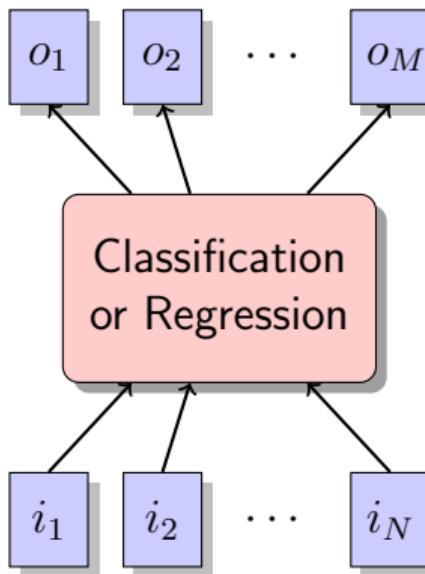
- consider variables x_1, \dots, x_k with joint $p(x_1, \dots, x_k)$
- order them in such a way that there is no link from any node x_i to any node x_j if $j < i$
- we want a sample $\hat{x}_1, \dots, \hat{x}_k$
- first sample \hat{x}_1 from $p(x_1)$
- then for every $n = [2, k]$ sample \hat{x}_n from $p(x_n | \text{pa}_n)$

- ① Given a set of observed variables, compute marginal for the other nodes
 - junction tree algorithm: efficient algorithm based on dynamic programming
- ② given a set of data points eliminate arcs
 - K2 algorithm

Graphical Models and Latent Variables



Graphical Models and Latent Variables

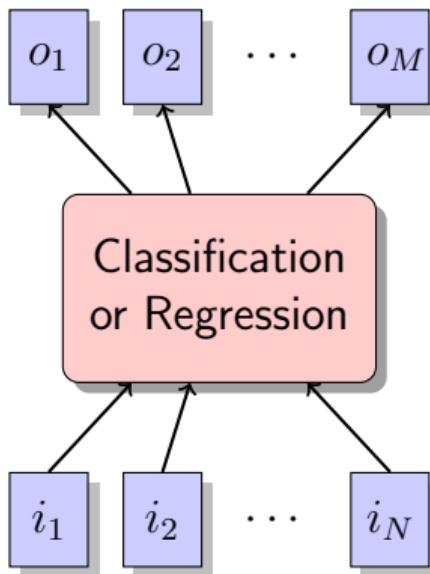


conditional distributions

$$P(o_1, \dots, o_M | i_1, \dots, i_N)$$

$$P(i_1, \dots, i_M | o_1, \dots, o_N)$$

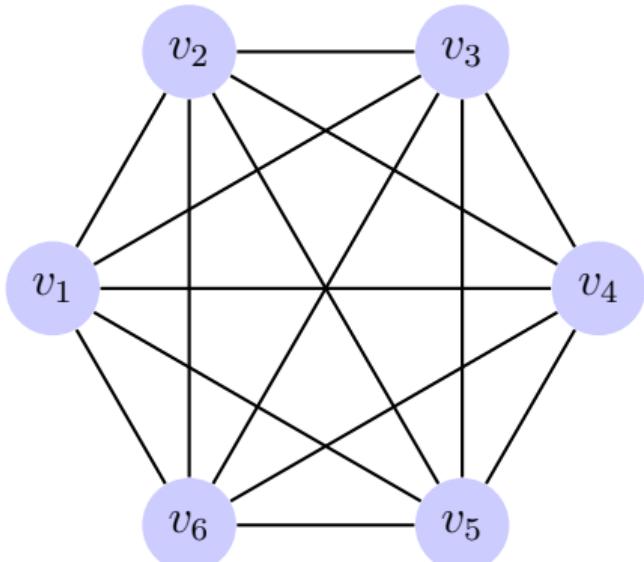
Graphical Models and Latent Variables



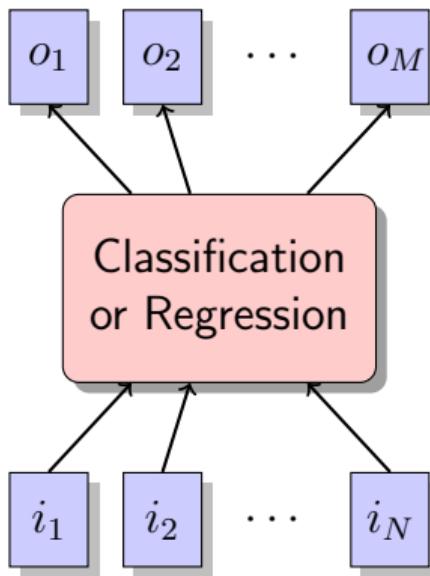
conditional distributions

$$P(o_1, \dots, o_M | i_1, \dots, i_N)$$

$$P(i_1, \dots, i_M | o_1, \dots, o_N)$$

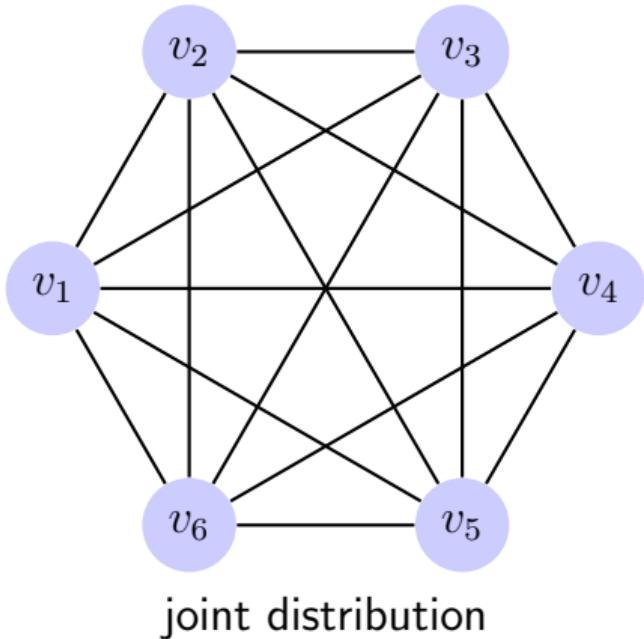


Graphical Models and Latent Variables



conditional distributions

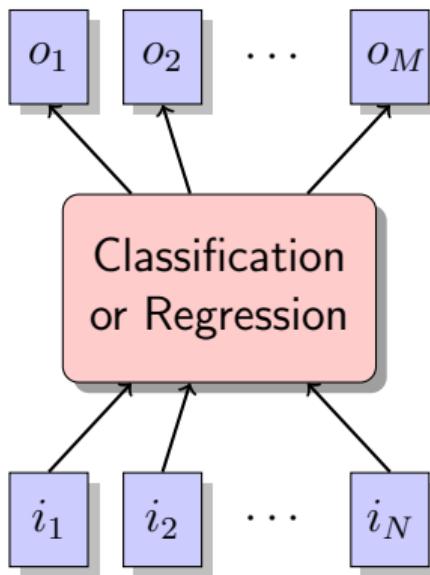
$$P(o_1, \dots, o_M | i_1, \dots, i_N)$$
$$P(i_1, \dots, i_M | o_1, \dots, o_N)$$



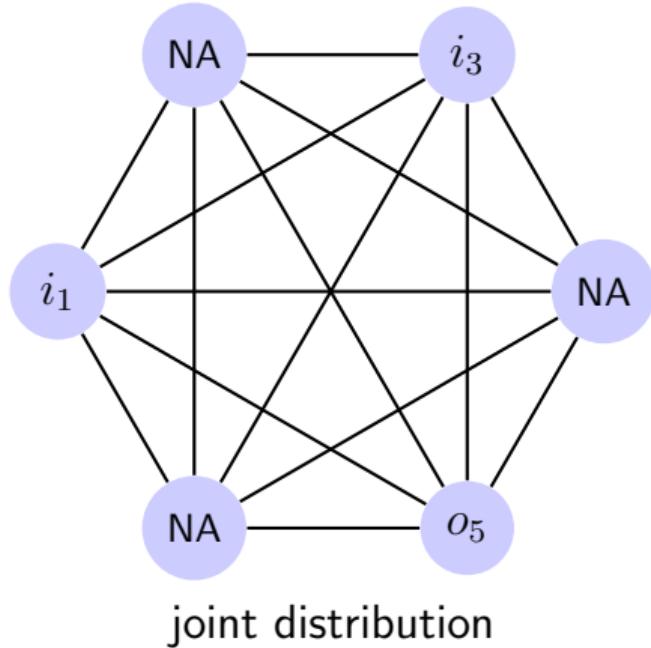
joint distribution

$$P(v_1, v_2, \dots, v_K)$$

Graphical Models and Latent Variables

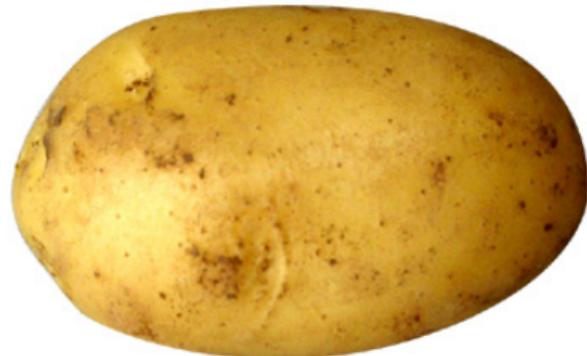
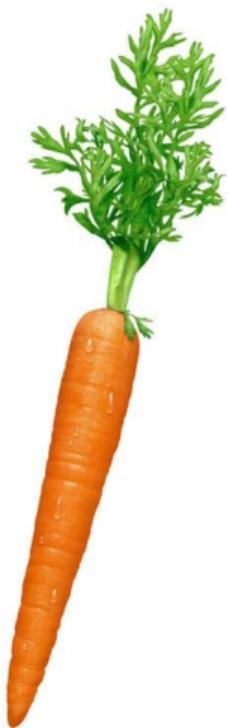


conditional distributions
 $P(o_1, \dots, o_M | i_1, \dots, i_N)$
 $P(i_1, \dots, i_M | o_1, \dots, o_N)$



joint distribution
 $P(v_1, v_2, \dots, v_K)$ marginalisation
 $P(o_5 | i_1, i_3)$

Multi-modality: analogy



Use of Graphical Models

- interpret probabilistic methods
- use directly as ML method (junction tree and K2 algorithms)
- Pros: much more flexible than SVMs and DNNs
- Cons: learning more difficult
 - not feasible for complex (noisy, continuous) perception tasks
 - more suitable for higher cognitive functions (discrete)

Mixture Models and Unsupervised Learning

TTT4185 Machine Learning for Signal Processing

Giampiero Salvi

Department of Electronic Systems
NTNU

HT2021

Supervised vs Unsupervised Learning

Supervised

$$\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$$

$$\mathbf{t} = \{t_1, \dots, t_N\}$$

$$\mathbf{x} \xrightarrow{f} t$$

Unsupervised

$$\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$$

- clustering
- distribution estimation
- dimensionality reduction

Mixture Models

Weighted sum of K simple probability distribution functions:

$$p(\mathbf{x}|\theta) = \sum_{k=1}^K \pi_k p(\mathbf{x}|\theta_k),$$

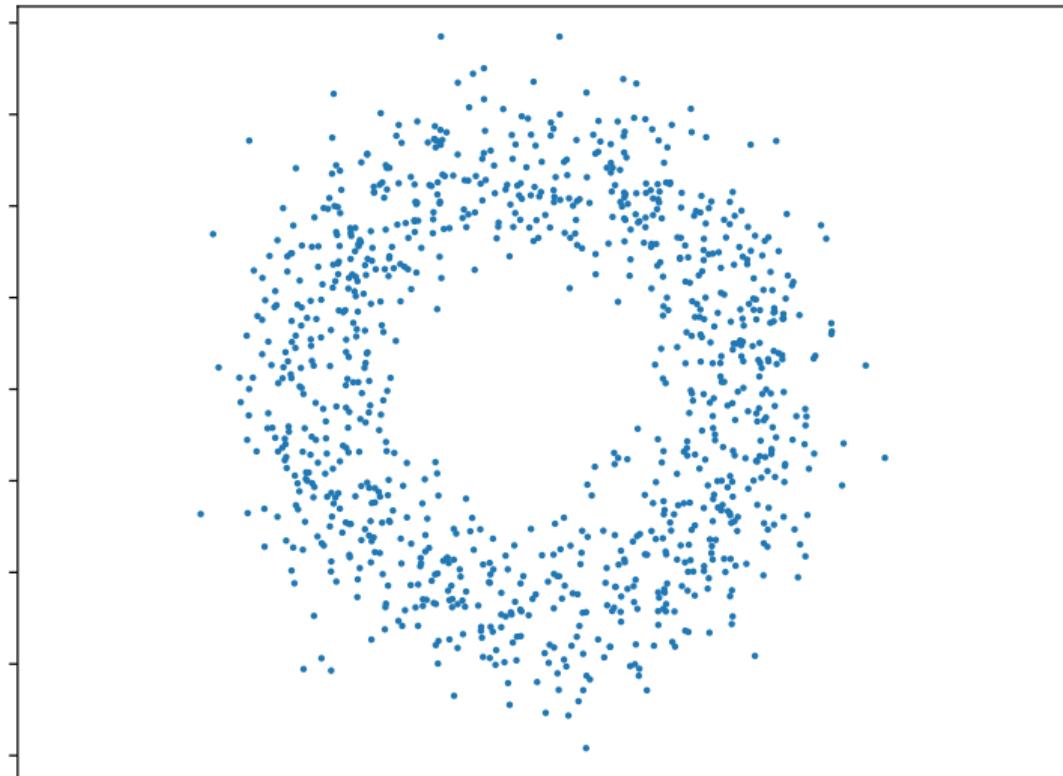
$$\text{with } \theta = \{\pi_1, \dots, \pi_k, \theta_1, \dots, \theta_K\}$$

Two main uses:

- Model complex distributions with simpler pdfs
- Clustering (unsupervised classification)

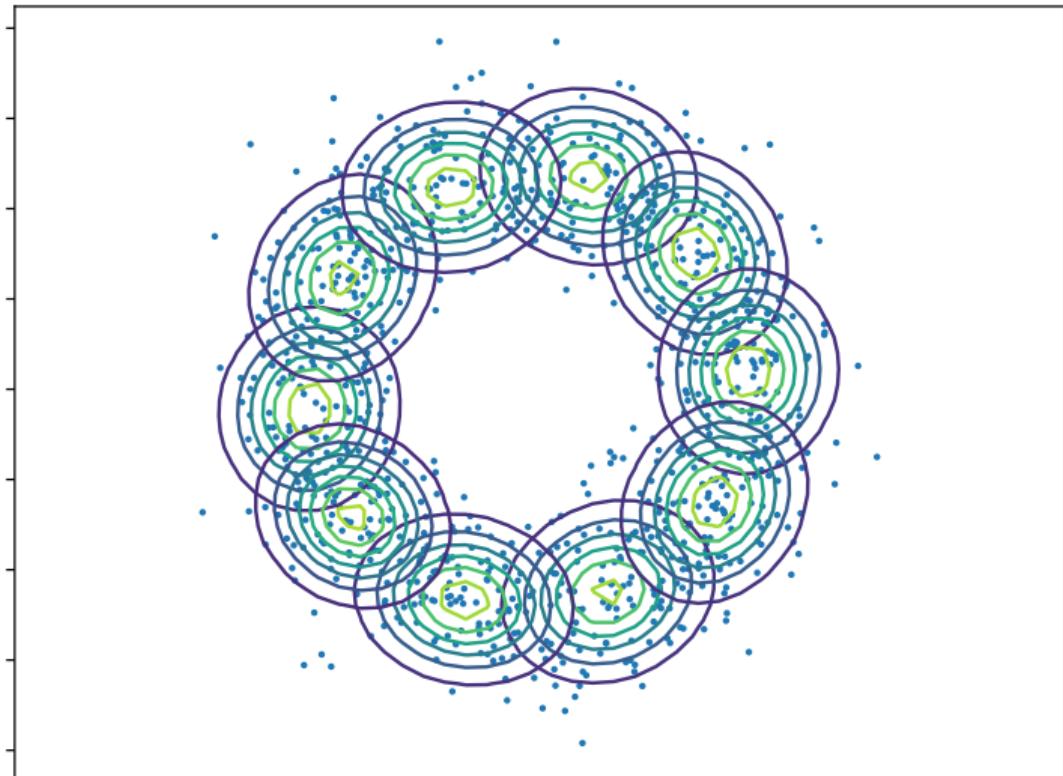
Example: approximating distribution (doughnut data)

$$\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$$



Example: approximating distribution (doughnut data)

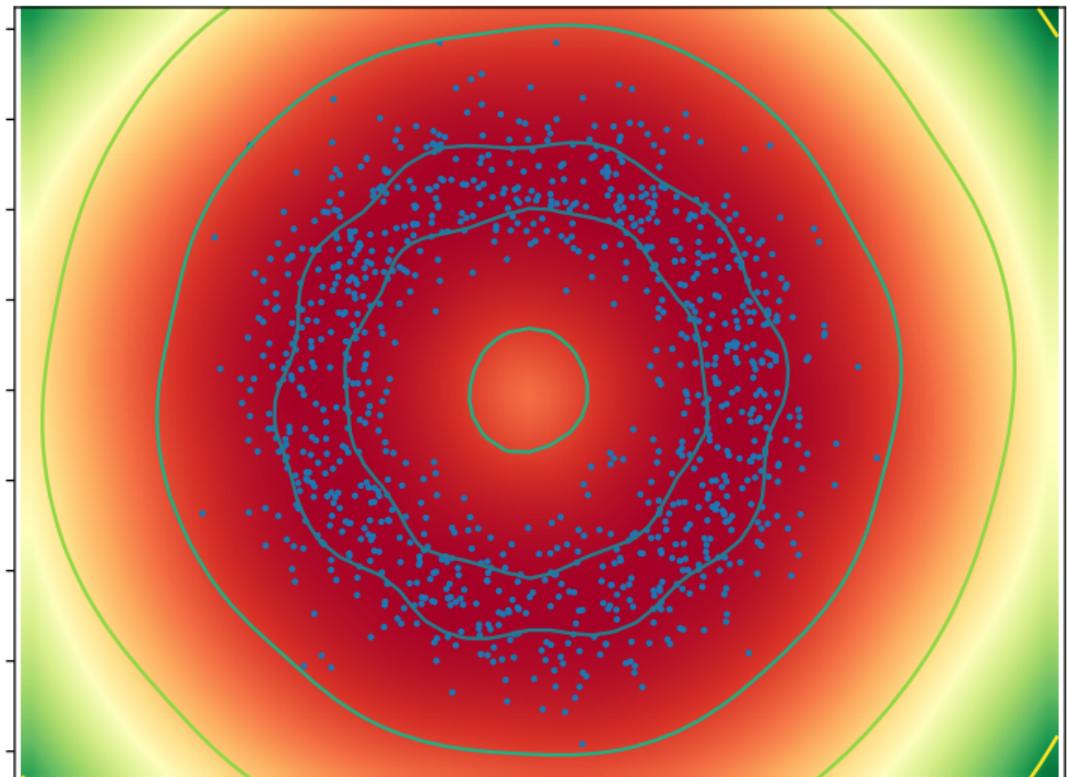
$$\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$$
$$p(\mathbf{x}|\theta_k), \forall k \in [1, K]$$



Example: approximating distribution (doughnut data)

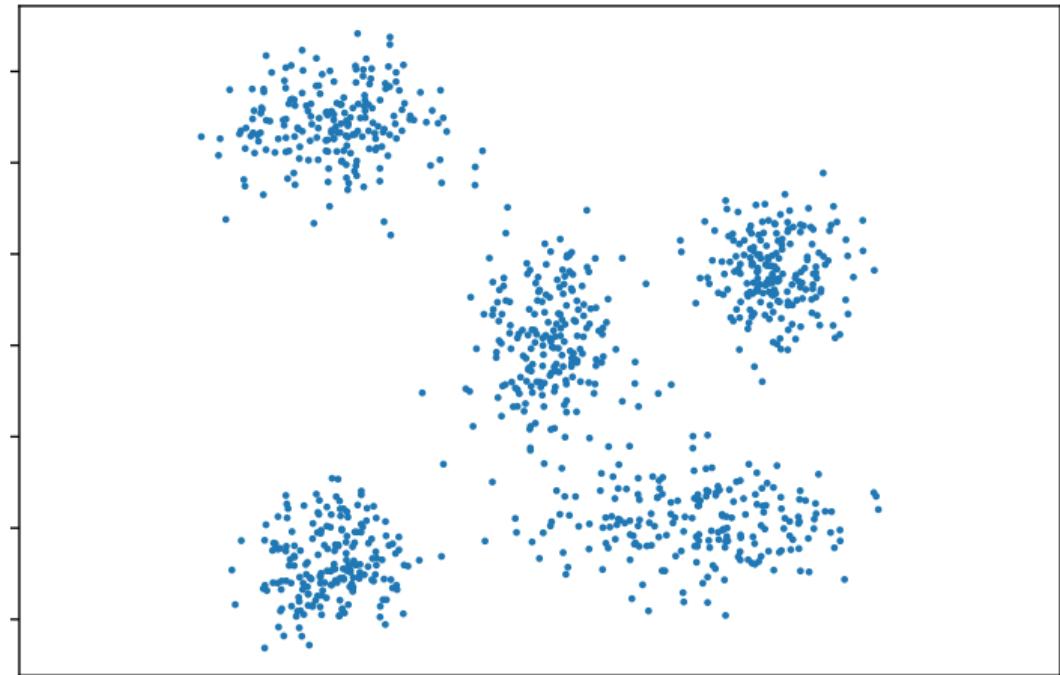
$$\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$$

$$p(\mathbf{x}|\theta) = \sum_{k=1}^K \pi_k p(x|\theta_k)$$



Example: Clustering

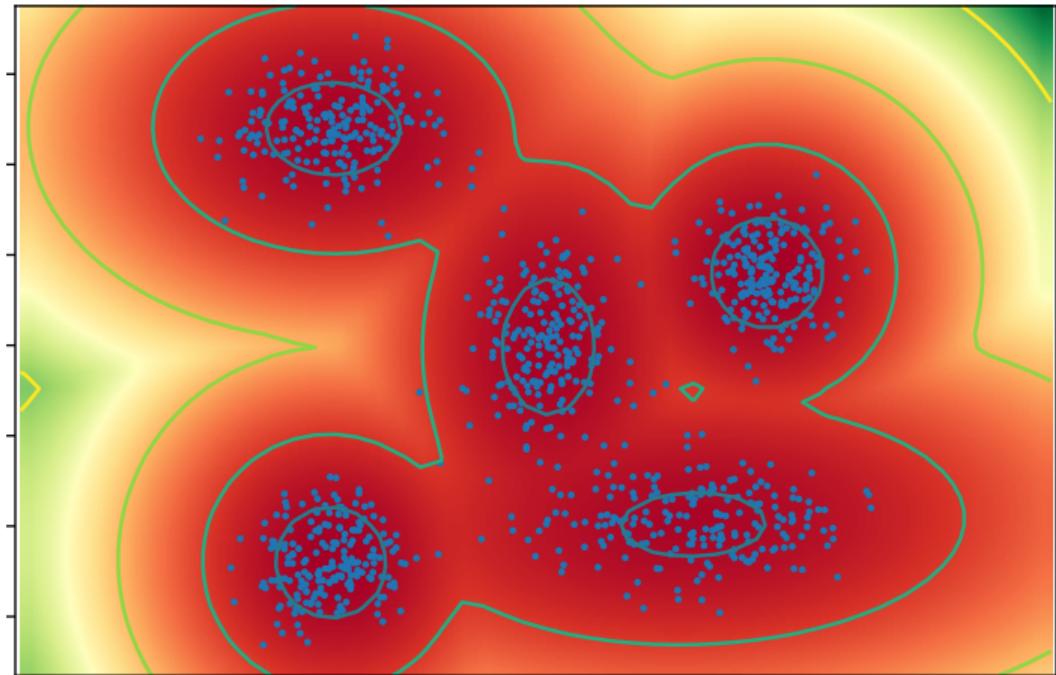
$$\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$$



Example: Clustering

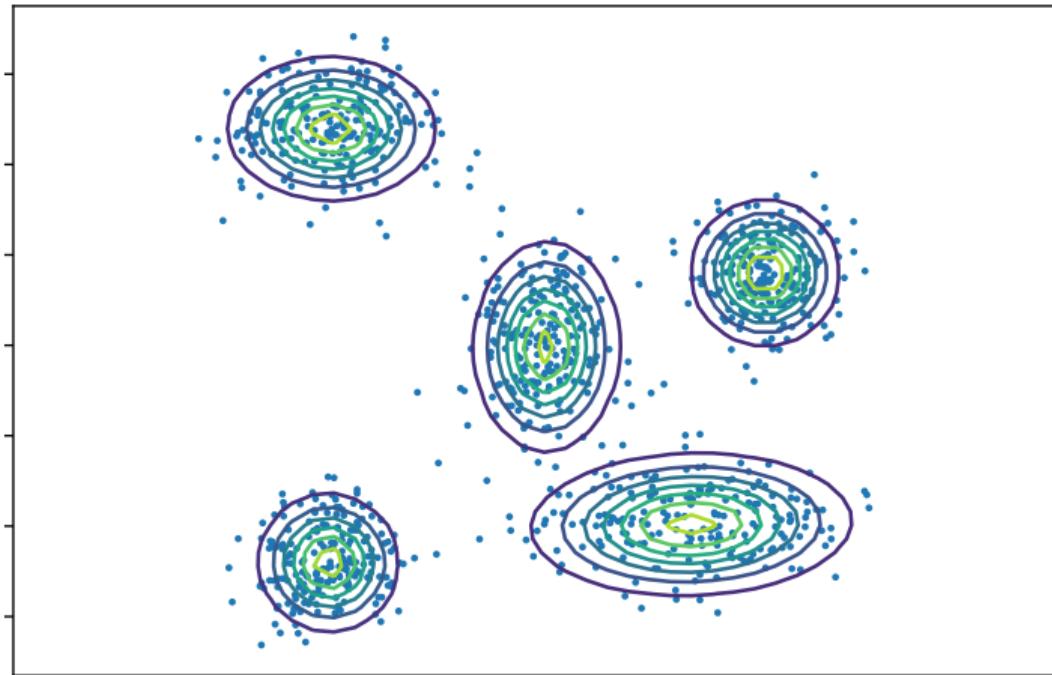
$$\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$$

$$p(\mathbf{x}|\theta) = \sum_{k=1}^K \pi_k p(x|\theta_k)$$



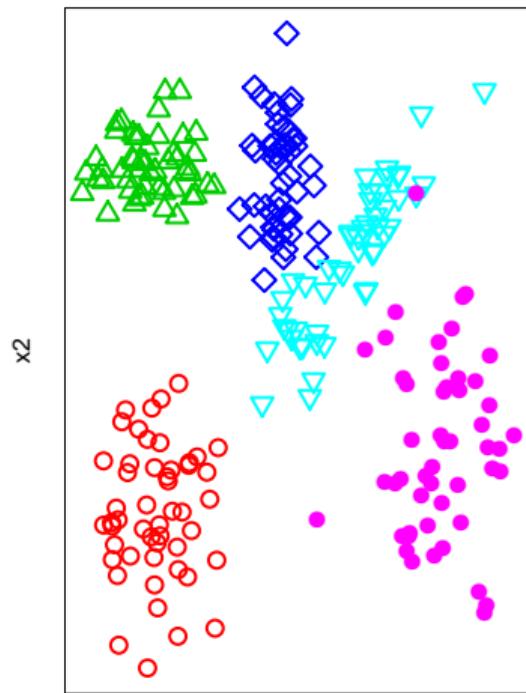
Example: Clustering

$$\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$$
$$p(\mathbf{x}|\theta_k), \forall k \in [1, K]$$

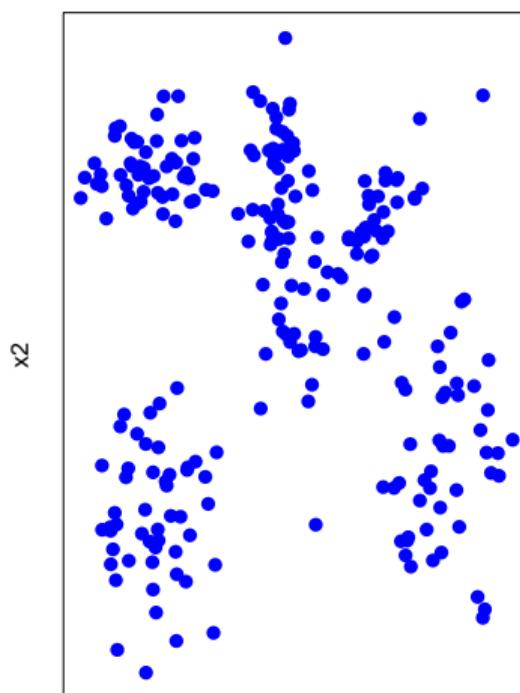


Clustering vs Classification

Classification



Clustering



Fitting Mixture of distributions

We would like to find the maximum likelihood solution:

$$\begin{aligned}\arg \max_{\theta} \ln p(\mathbf{X}|\theta) &= \arg \max \sum_{n=1}^N \ln p(\mathbf{x}_n|\theta) \\ &= \arg \max_{\theta} \sum_{n=1}^N \ln \sum_{k=1}^K \pi_k p(\mathbf{x}_n|\theta_k),\end{aligned}$$

with $\theta = \{\pi_1, \dots, \pi_K, \theta_1, \dots, \theta_K\}$

Fitting Mixture of distributions

We would like to find the maximum likelihood solution:

$$\begin{aligned}\arg \max_{\theta} \ln p(\mathbf{X}|\theta) &= \arg \max \sum_{n=1}^N \ln p(\mathbf{x}_n|\theta) \\ &= \arg \max_{\theta} \sum_{n=1}^N \ln \sum_{k=1}^K \pi_k p(\mathbf{x}_n|\theta_k),\end{aligned}$$

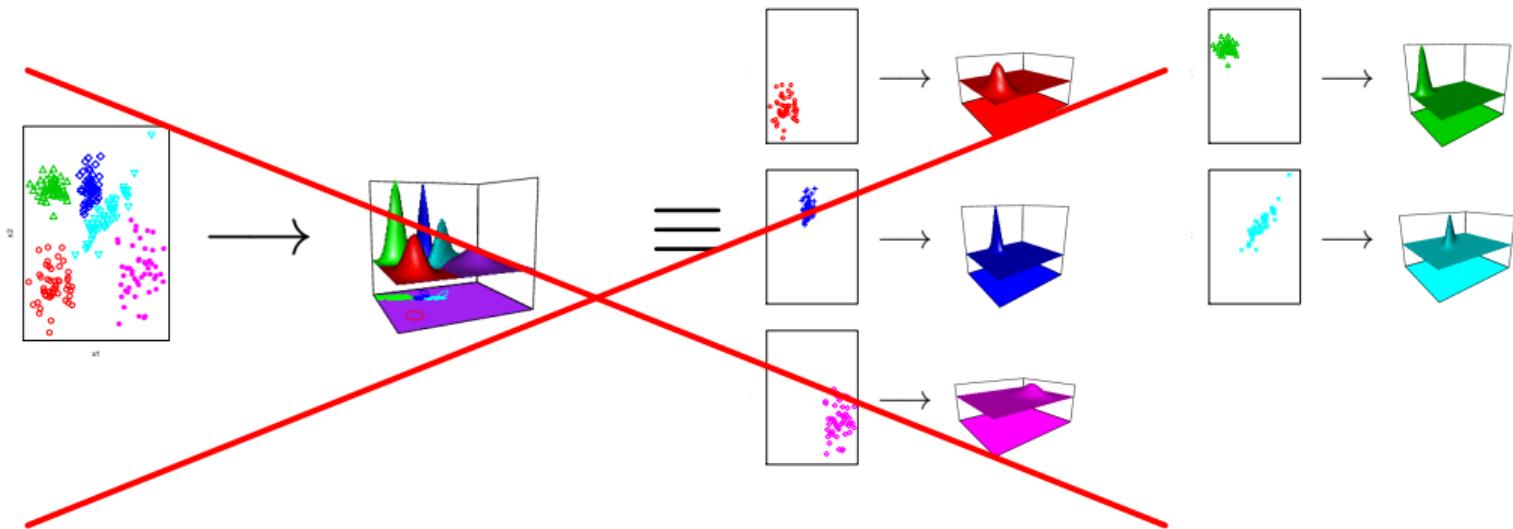
with $\theta = \{\pi_1, \dots, \pi_K, \theta_1, \dots, \theta_K\}$

Problem:

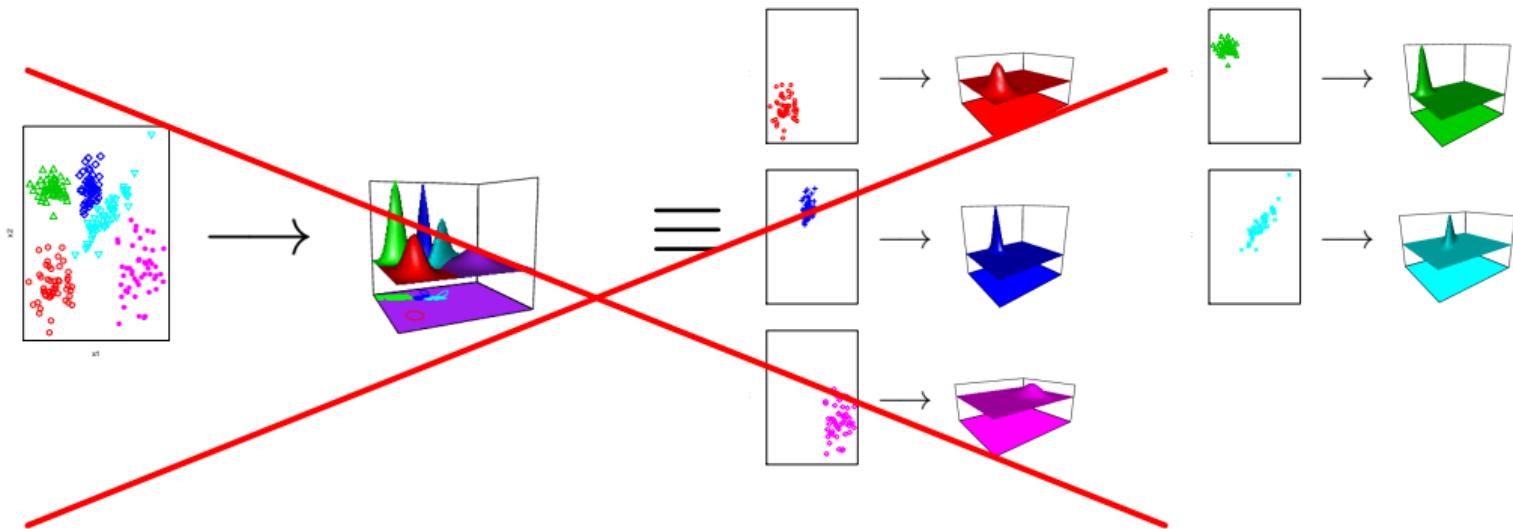
We do not know which point has been generated by which component of the mixture

We cannot optimize $p(\mathbf{X}|\theta)$ directly

No Class Independence Assumption



No Class Independence Assumption



Solution: Expectation Maximization

Expectation Maximization

- introduce **latent variables** to solve the problem
- very general idea (applies to many different probabilistic models)
- easier to explain in terms of clustering than modelling complex distributions
- We will use K -means as introduction

K -means

- given a set of points $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$
- describes each class with a centroid $\boldsymbol{\mu}_k$, $k = 1, \dots, K$
- a point belongs to a class if the corresponding centroid is closest (Euclidean distance)
- define binary indicator variable $r_{nk} \in \{0, 1\}$ (1-of- K coding)
- find $\boldsymbol{\mu}_k$ and r_{nk} by optimizing the distortion measure:

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_n\|^2$$

K -means

- iterative procedure (both steps have closed solution):

- ① optimize J w.r.t. r_{nk} keeping μ_k fixed

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg \min_j \|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2 \\ 0 & \text{otherwise.} \end{cases}$$

- ② optimize J w.r.t. μ_k keeping r_{nk} fixed

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk} \mathbf{x}_n}{\sum_n r_{nk}}$$

- guaranteed to converge
- not guaranteed to find the optimal solution
- used in vector quantization (since the 1950's)

K-means: algorithm

Data: K (number of desired clusters), N data points \mathbf{x}_n

Result: K clusters

initialization: assign initial value to K centroids μ_k ;

repeat

 assign each point \mathbf{x}_n to closest centroid μ_k ;

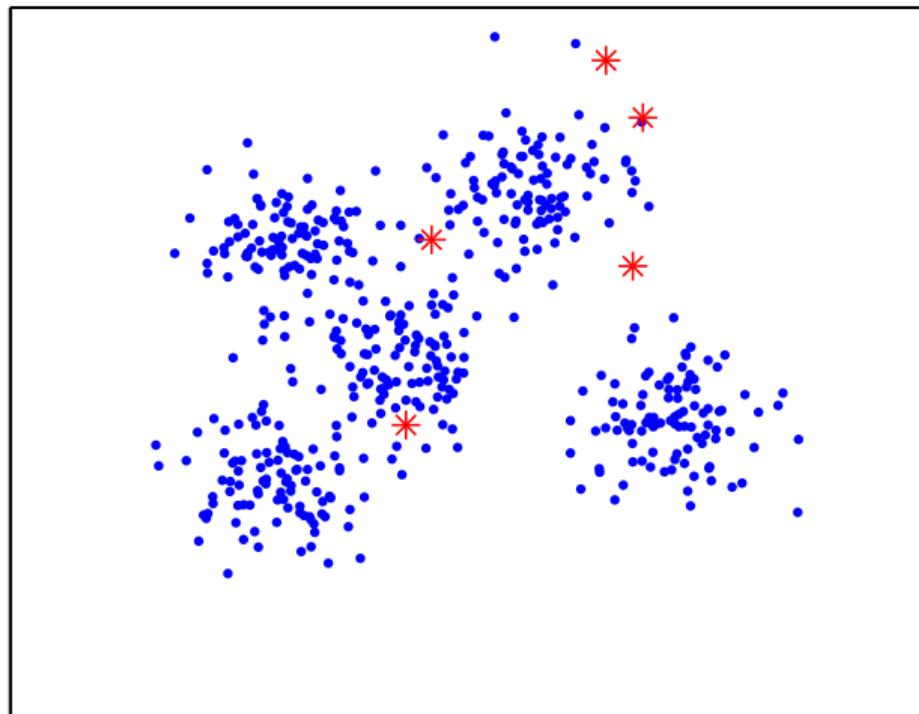
 compute new centroids as mean of each group of points;

until centroids do not change;

return K clusters;

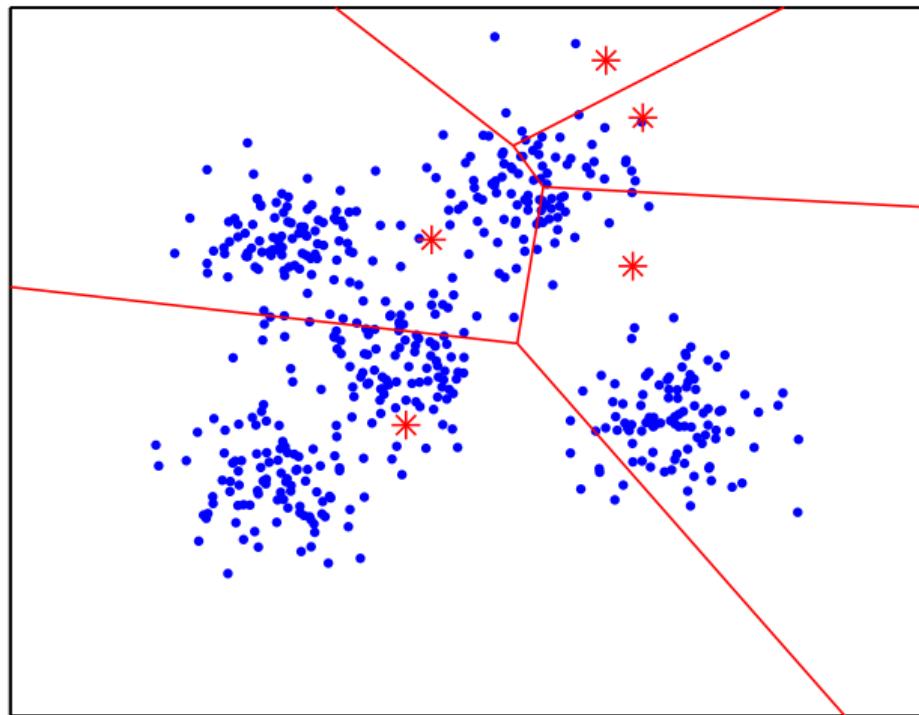
K-means: example

initialization



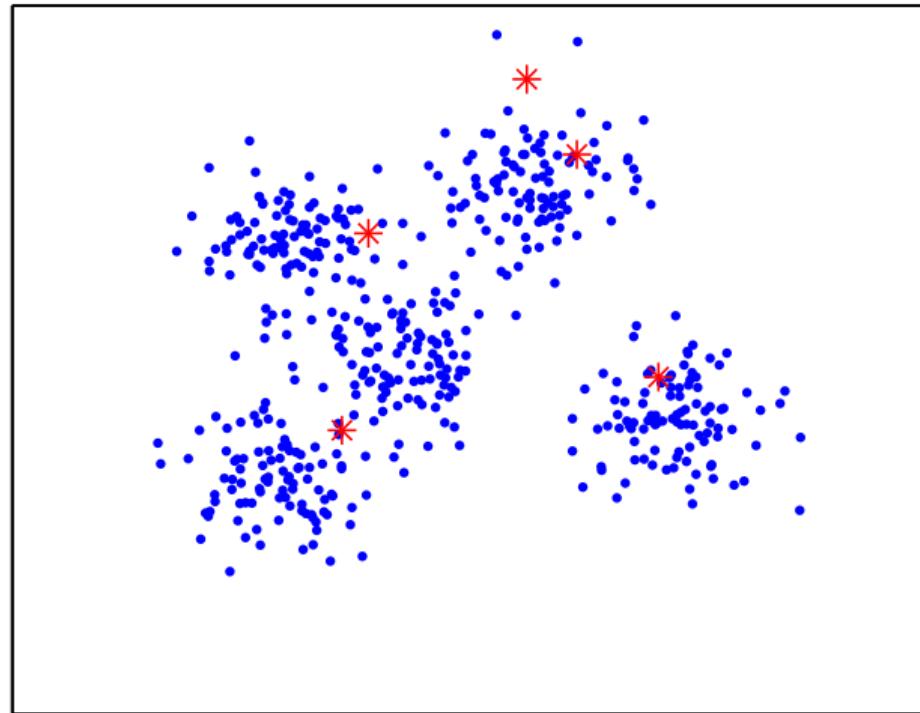
K-means: example

iteration 1, update clusters



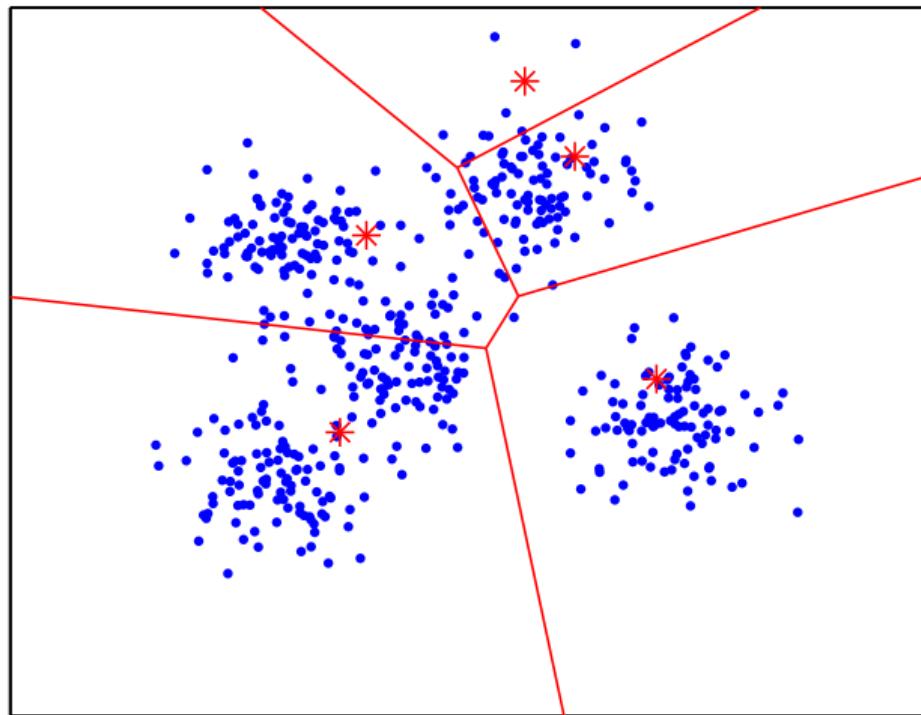
K-means: example

iteration 2, update centroids



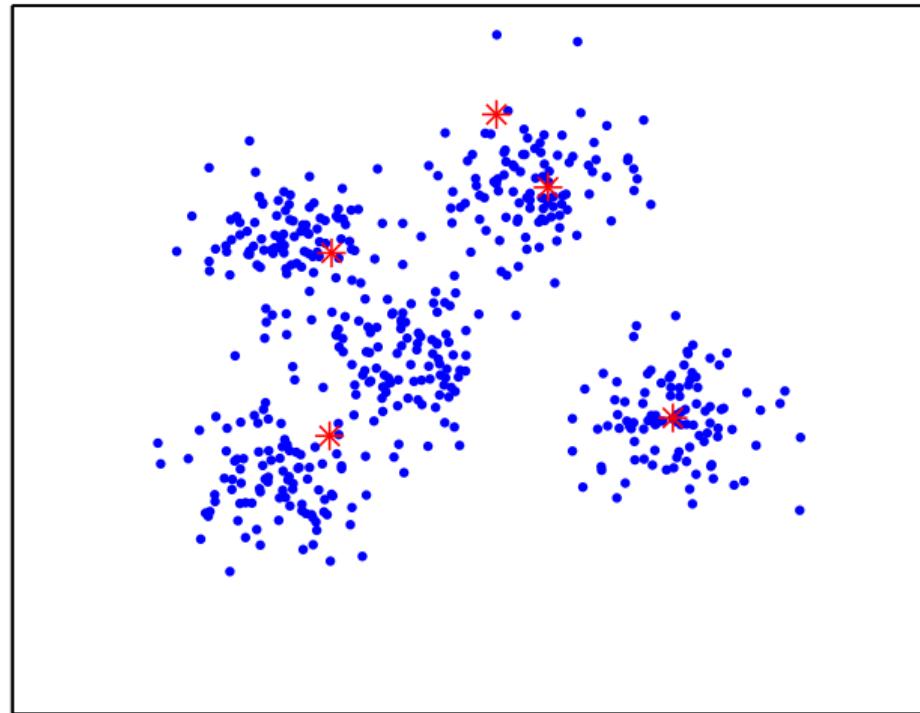
K-means: example

iteration 2, update clusters



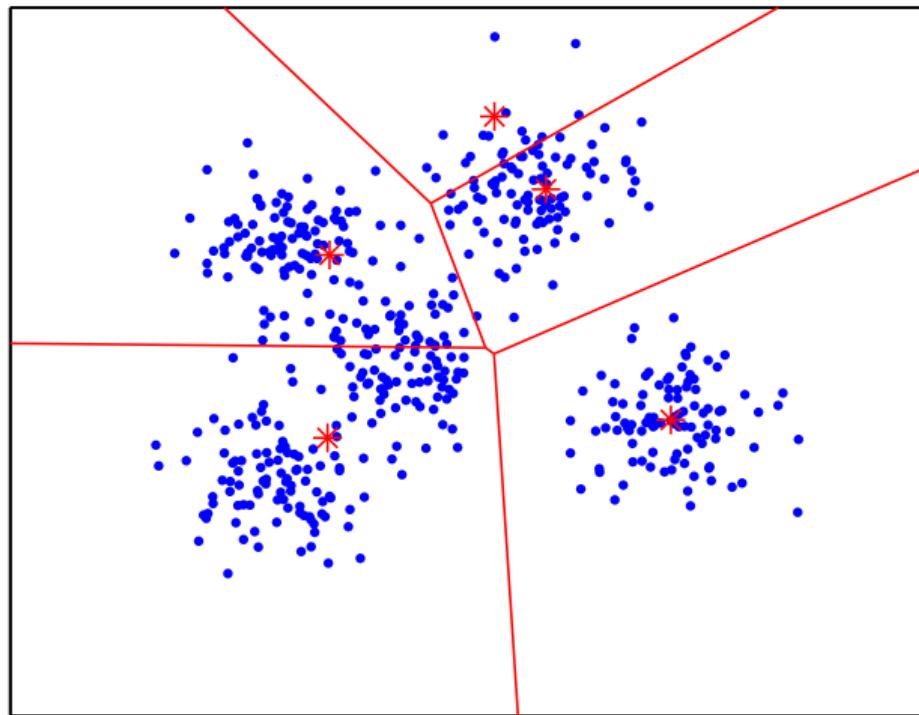
K-means: example

iteration 3, update centroids



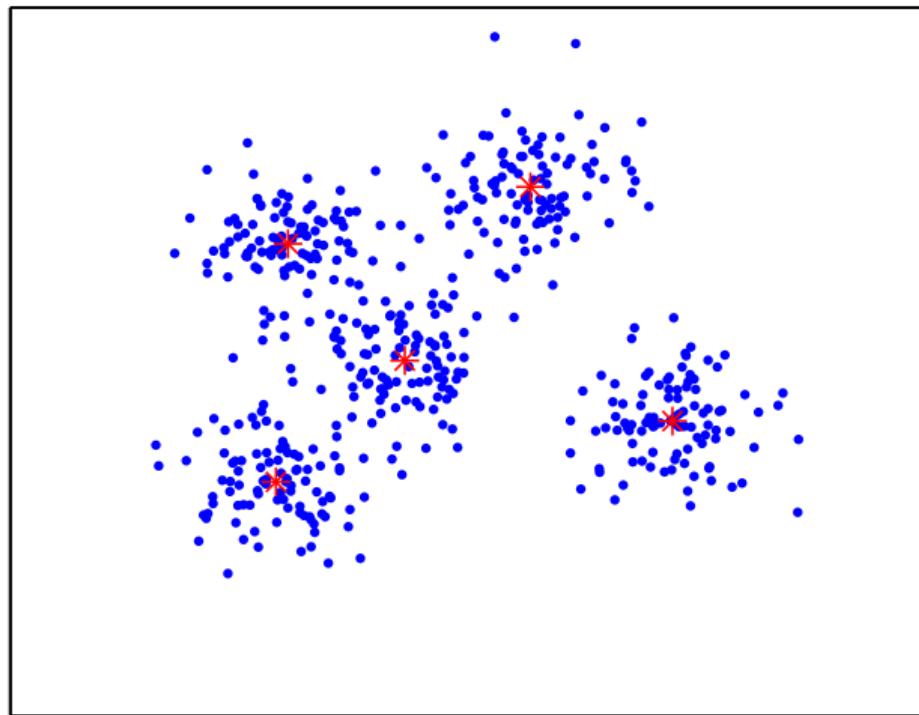
K-means: example

iteration 3, update clusters



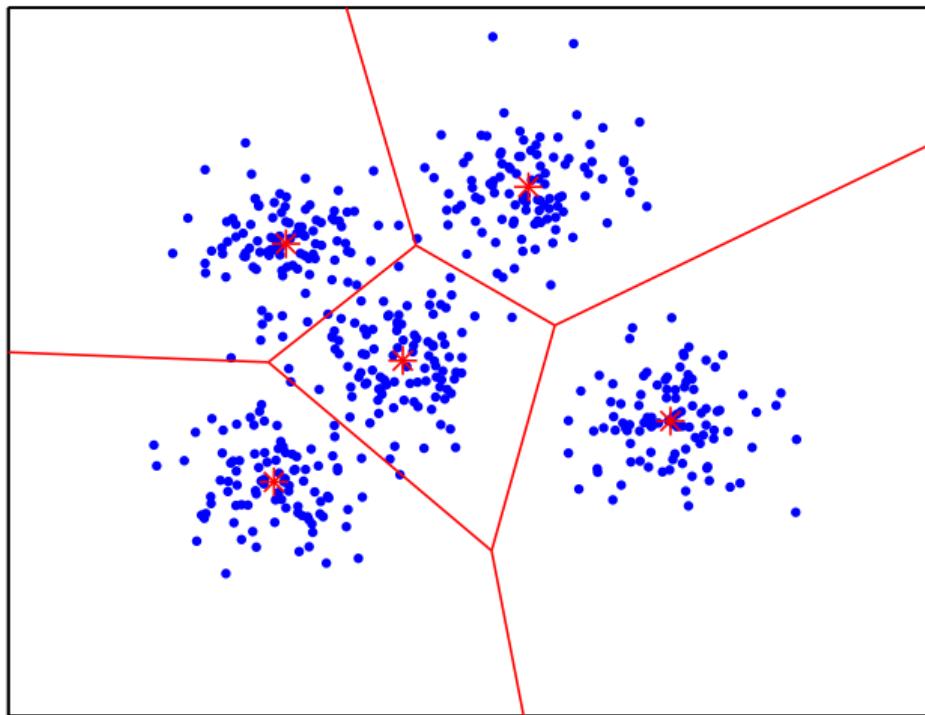
K-means: example

iteration 20, update centroids



K-means: example

iteration 20, update clusters



Example: data compression



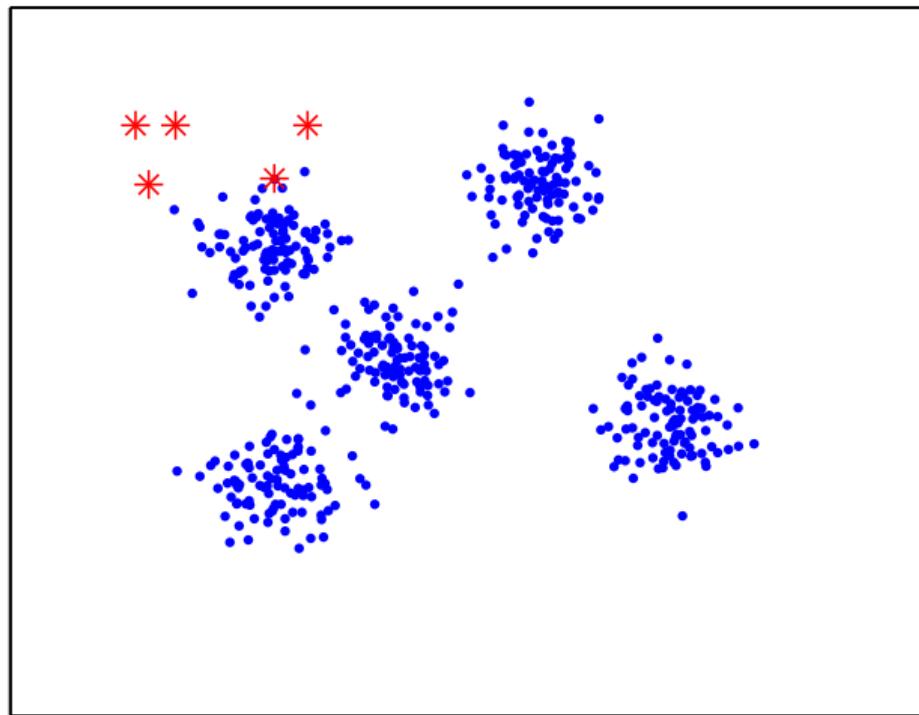
N pixels, 8 bits RGB
 $\rightarrow 24N$ bits

K -means $\rightarrow 24K + N \log_2 K$

If 1 Megapixel,
 $24N = 24$ millions,
 K -means, $K = 10 \rightarrow$
 ~ 3 millions

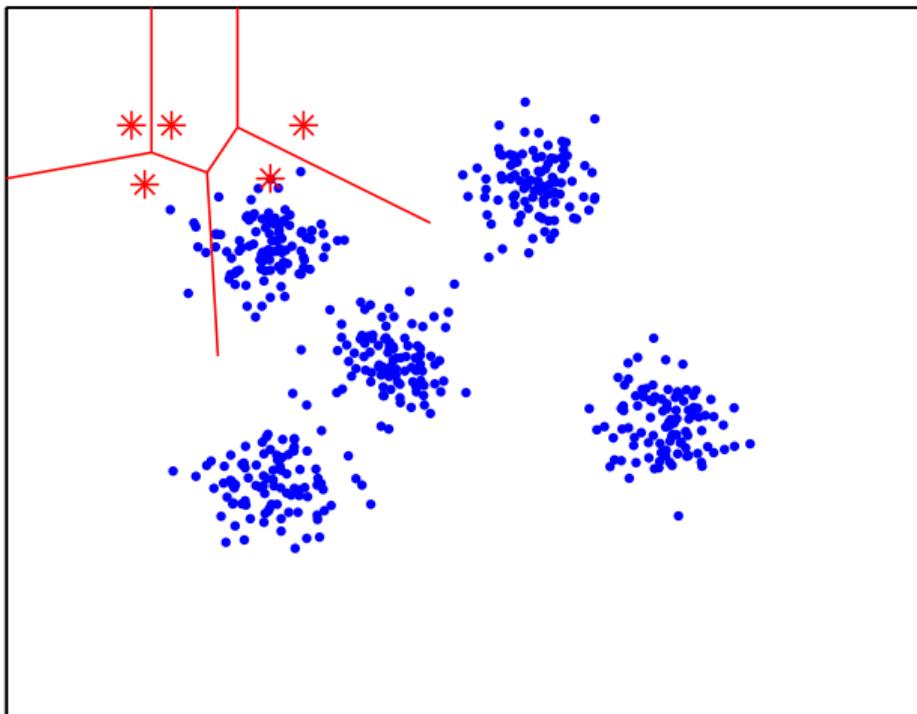
K-means: sensitivity to initial conditions

initialization



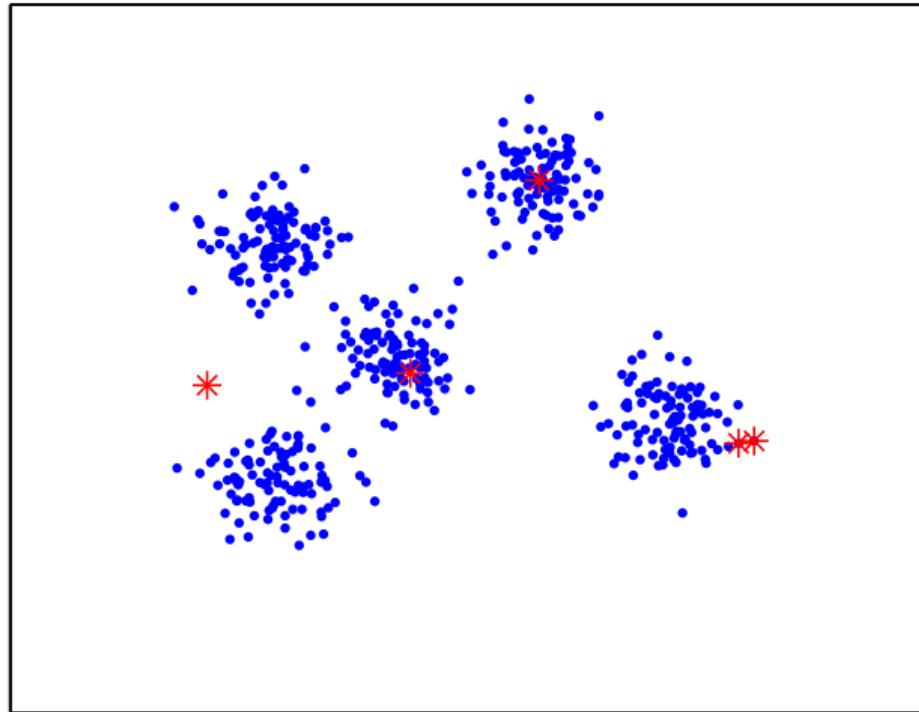
K-means: sensitivity to initial conditions

iteration 1, update clusters



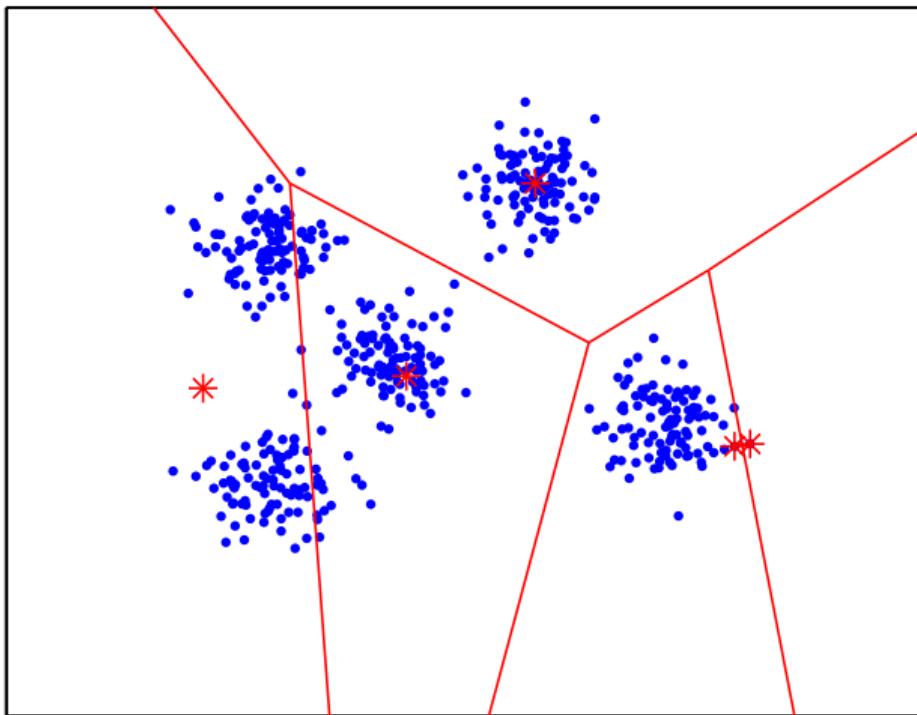
K-means: sensitivity to initial conditions

iteration 2, update centroids



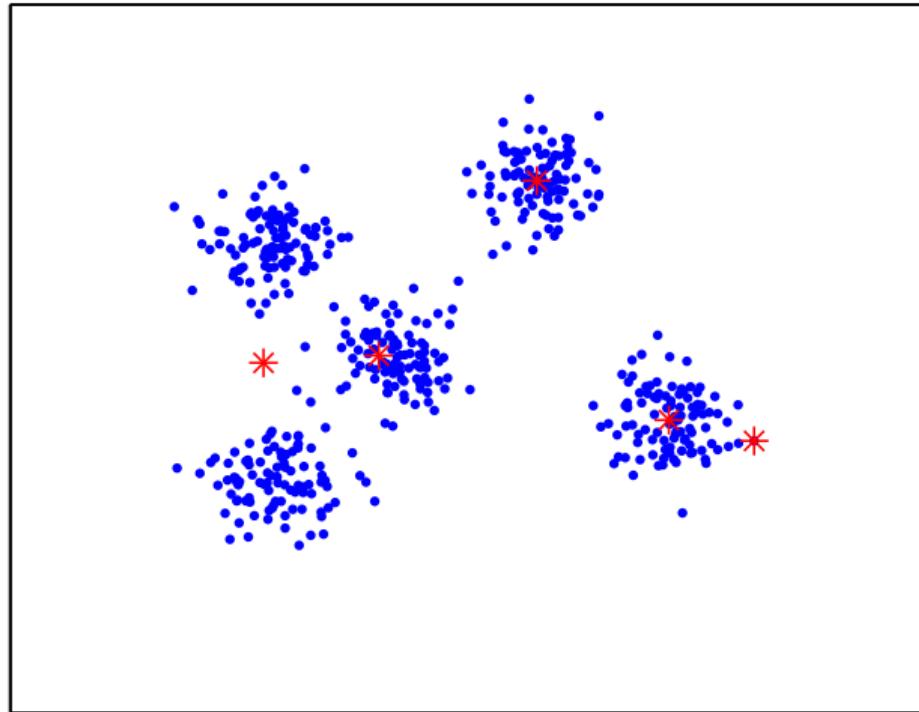
K-means: sensitivity to initial conditions

iteration 2, update clusters



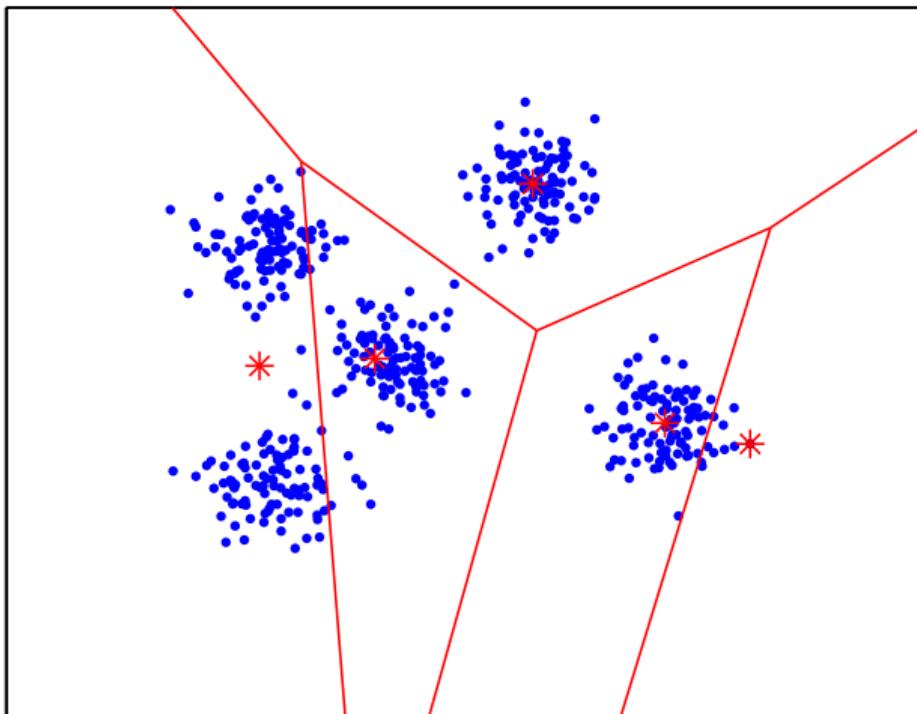
K-means: sensitivity to initial conditions

iteration 3, update centroids



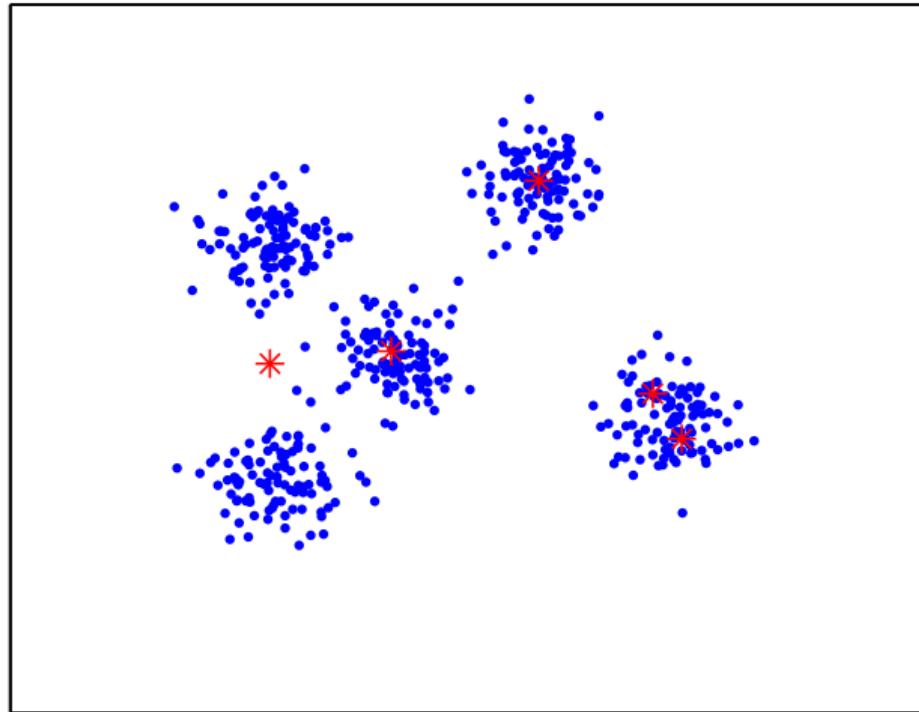
K-means: sensitivity to initial conditions

iteration 3, update clusters



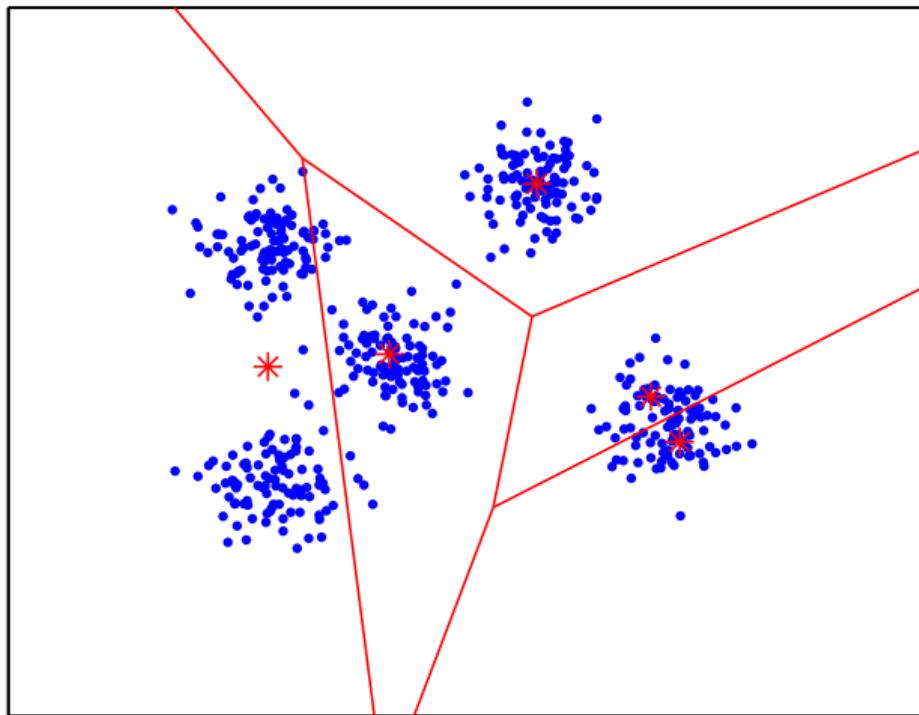
K-means: sensitivity to initial conditions

iteration 20, update centroids



K-means: sensitivity to initial conditions

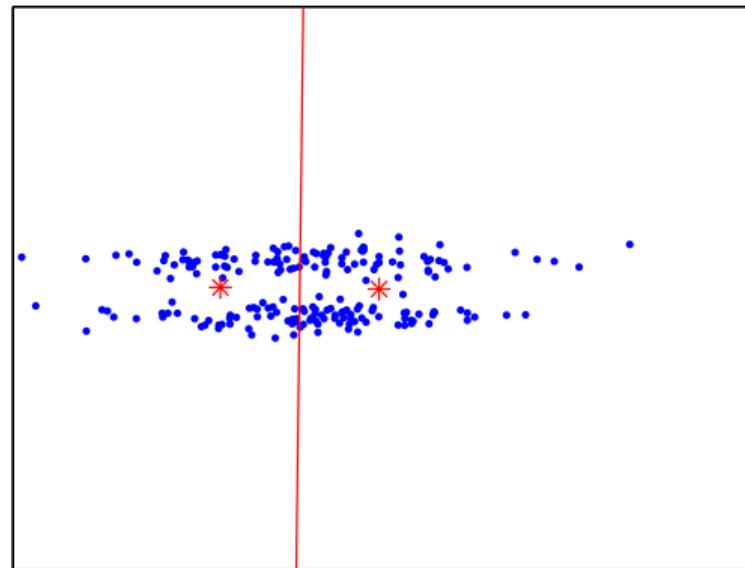
iteration 20, update clusters



K-means: limits of Euclidean distance

- the Euclidean distance is isotropic (same in all directions in \mathbb{R}^p)
- this favours spherical clusters
- the size of the clusters is controlled by their distance

two non-spherical classes



Rethinking Mixture Models: Latent Variables

$$p(\mathbf{x}|\theta) = \sum_{k=1}^K \pi_k p(\mathbf{x}|\theta_k) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

We introduce a *latent* variable \mathbf{z} with a 1-of- K representation:

$$\mathbf{z} \in \{0, 1\}^K, \sum_{k=1}^K z_k = 1.$$



Alternative formulation: $h \in \{1, \dots, K\}$.

Latent Variable: responsibilities

$$p(\mathbf{x}|\theta) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k),$$

Marginal (w.r.t. \mathbf{z}):

$$p(z_k = 1) = \pi_k, \text{ or}$$

$$p(\mathbf{z}) = \prod_{k=1}^K \pi_k^{z_k},$$

Conditional:

$$p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \text{ or}$$

$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^K \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k},$$



Latent Variable: important probabilities

$$p(\mathbf{x}|\theta) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k),$$

Posterior:

$$\begin{aligned}\gamma(z_k) \equiv p(z_k = 1 | \mathbf{x}) &= \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^K p(z_j = 1)p(\mathbf{x}|z_j = 1)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}\end{aligned}$$



This is also called the **responsibility** of the term k in the mixture.

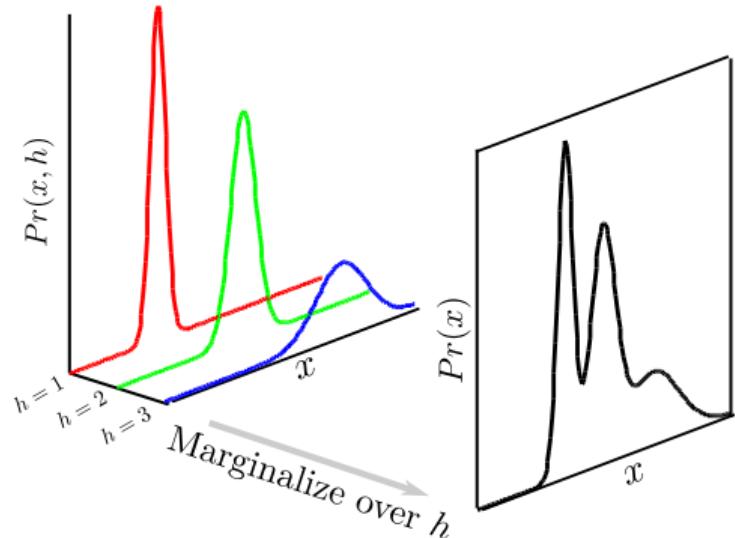
Mixture of Gaussians as a marginalization

Marginal (w.r.t. \mathbf{x}):

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}) = \sum_{k=1}^K p(\mathbf{x}, z_k = 1)$$

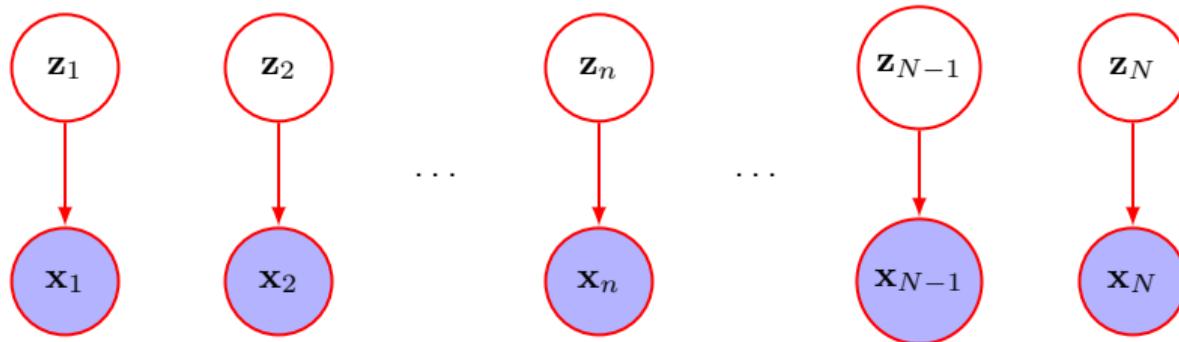
$$= \sum_{k=1}^K p(z_k = 1) p(\mathbf{x} \mid z_k = 1)$$

$$= \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

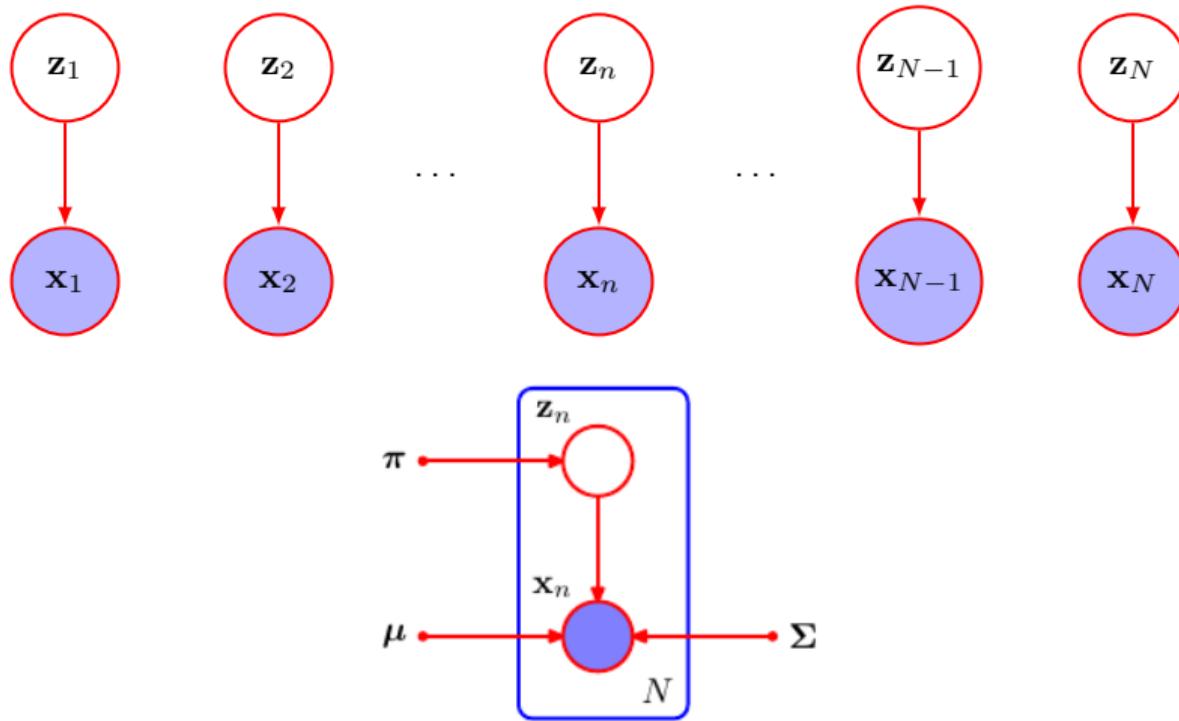


Figures taken from **Computer Vision: models, learning and inference** by Simon Prince.

Set of N i.i.d. points



Set of N i.i.d. points



Ancestral Sampling

We assume the data is generated as follows:

For $n \in [1, N]$, do:

- ① sample the value of \mathbf{z}_n from the distribution $\{\pi_1, \dots, \pi_K\}$
- ② given that $z_{nk} = 1$, sample \mathbf{x}_n from $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$



Maximum Likelihood for Mixture of distributions

We would like to find the maximum likelihood solution:

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \ln \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k),$$

Problems:

- log of sum hard to optimize → Expectation Maximization
- singularities
- identifiability

Singularities

- set a lower threshold for variance,
or
- detect collapsing Gaussian and
reinitialize

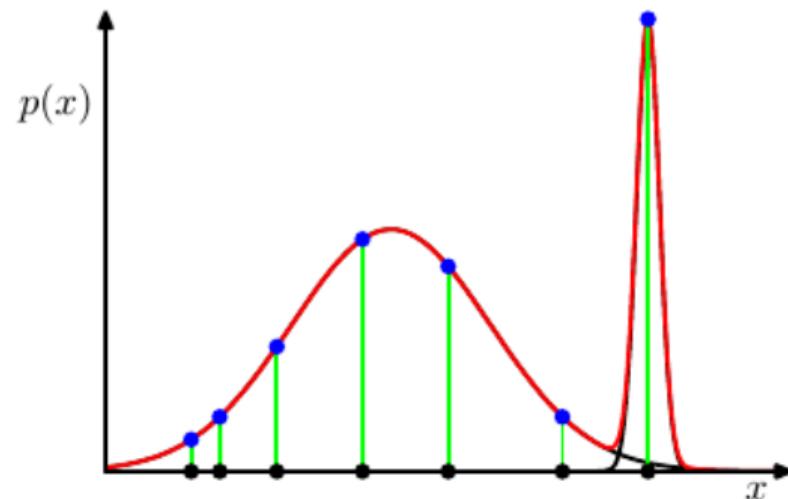


Figure from Bishop

Identifiability

Any permutation of the K distributions is an equivalent solution

Irrelevant if we are interested in $p(\mathbf{x})$ only, but

Important if we are interested in clustering.

Maximum Likelihood

Setting

$$\frac{d \ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{d\boldsymbol{\mu}} = 0$$

we obtain:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n$$

with

$$N_k = \sum_{n=1}^N \gamma(z_{nk})$$

Maximum Likelihood

Similarly for Σ and π :

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n,$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^\top,$$

$$\pi_k = \frac{N_k}{N}, \text{ with}$$

$$N_k = \sum_{n=1}^N \gamma(z_{nk}).$$

Maximum Likelihood

Similarly for Σ and π :

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n,$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^\top,$$

$$\pi_k = \frac{N_k}{N}, \text{ with}$$

$$N_k = \sum_{n=1}^N \gamma(z_{nk}).$$

Not a closed form solution!

We do not know $\gamma(z_{nk})$ until we know $\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}$.

Expectation Maximization

Solve problem with iterative procedure:

- ① Expectation: given π, μ, Σ estimate responsibilities:

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

- ② Maximization: given $\gamma(z_{nk})$, maximise likelihood
(formulae from previous slide)
- very general idea (applies to many different probabilistic models)
 - optimize the Likelihood of the complete data over N data points

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{z}_1, \dots, \mathbf{z}_N | \theta) = p(\mathbf{X}, \mathbf{Z} | \theta)$$

EM and K -means

EM is very similar to K -means, but:

- $\gamma(z_{nk})$ can be non-zero for all $n \Rightarrow$ all points contribute to all distributions
- probability distribution functions are used instead of Euclidean distance:
more flexible cluster shapers

Illustration: EM for two univariate Gaussians

For each sample x_n introduce a *hidden variable* \mathbf{z}_n

$$\mathbf{z}_n = \begin{cases} [1, 0] & \text{if sample } x_n \text{ was drawn from } \mathcal{N}(x|\mu_1, \sigma_1^2) \\ [0, 1] & \text{if sample } x_n \text{ was drawn from } \mathcal{N}(x|\mu_2, \sigma_2^2) \end{cases}$$

Initialize the model parameters (random, or using K -means):

$$\Theta^{(0)} = (\pi_1^{(0)}, \mu_1^{(0)}, \sigma_1^{(0)}, \mu_2^{(0)}, \sigma_2^{(0)})$$

Update the parameters iteratively with Expectation and Maximization steps...

EM for two Gaussians: E-step

Estimate the **responsibility** $\gamma(z_{nk})$ of
 k -th Gaussian **for each sample** x_n
(indicated by the size of the projected
data point)

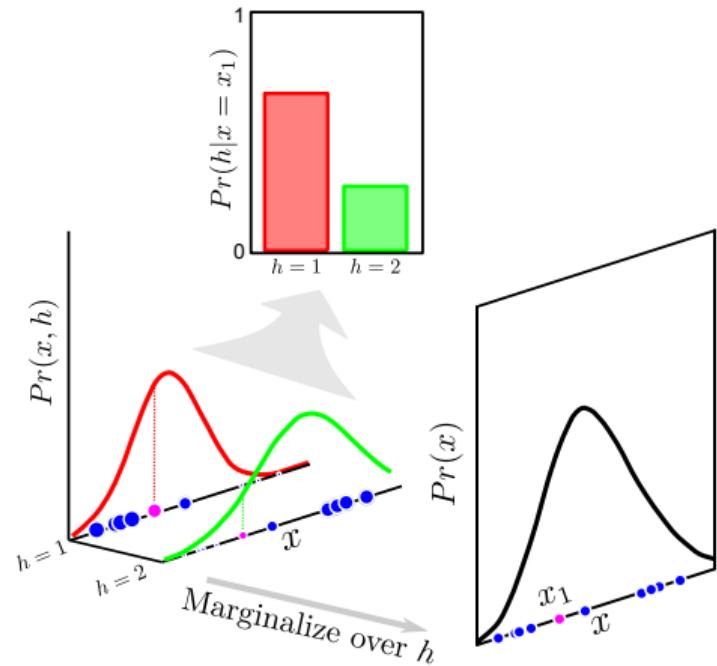


Figure from Prince.

EM for two Gaussians: E-step (cont.)

E-step: Compute the *posterior probability* that x_n was generated by component k given the current estimate of the parameters $\Theta^{(t)}$. (responsibilities)

for $n = 1, \dots, N$

for $k = 1, 2$

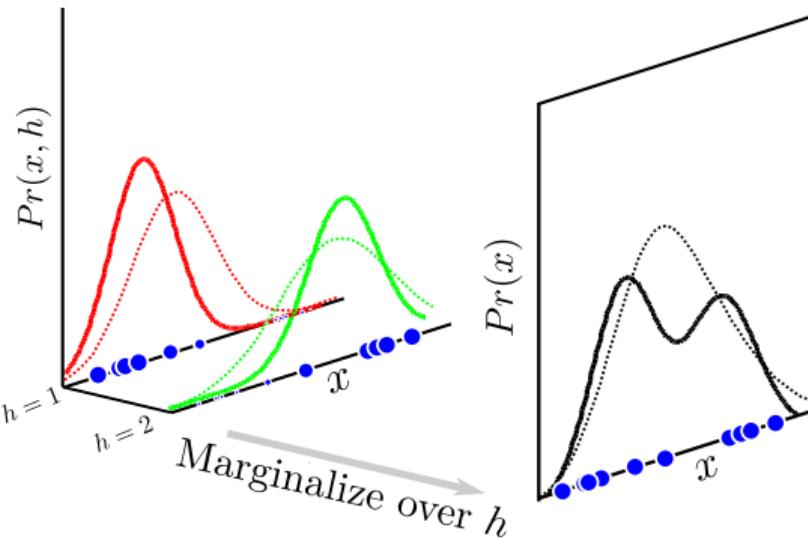
$$\begin{aligned}\gamma_{nk}^{(t)} &= P(z_{nk} = 1 \mid x_n, \Theta^{(t)}) \\ &= \frac{\pi_k^{(t)} \mathcal{N}(x_n \mid \mu_k^{(t)}, \sigma_k^{(t)})}{\pi_1^{(t)} \mathcal{N}(x_n \mid \mu_1^{(t)}, \sigma_1^{(t)}) + \pi_2^{(t)} \mathcal{N}(x_n \mid \mu_2^{(t)}, \sigma_2^{(t)})}\end{aligned}$$

Note: $\gamma_{n1}^{(t)} + \gamma_{n2}^{(t)} = 1$ and $\pi_1 + \pi_2 = 1$

EM for two Gaussians: M-step

Fitting the Gaussian model **for each of k -th constituent.**

Sample x_n contributes according to the responsibility γ_{nk} .



(dashed and solid lines for fit
before and after update)

Figure from Prince.

EM for two Gaussians: M-step (cont.)

M-step: Compute the *Maximum Likelihood* of the parameters of the mixture model given out data's membership distribution, the $\gamma_i^{(t)}$'s:

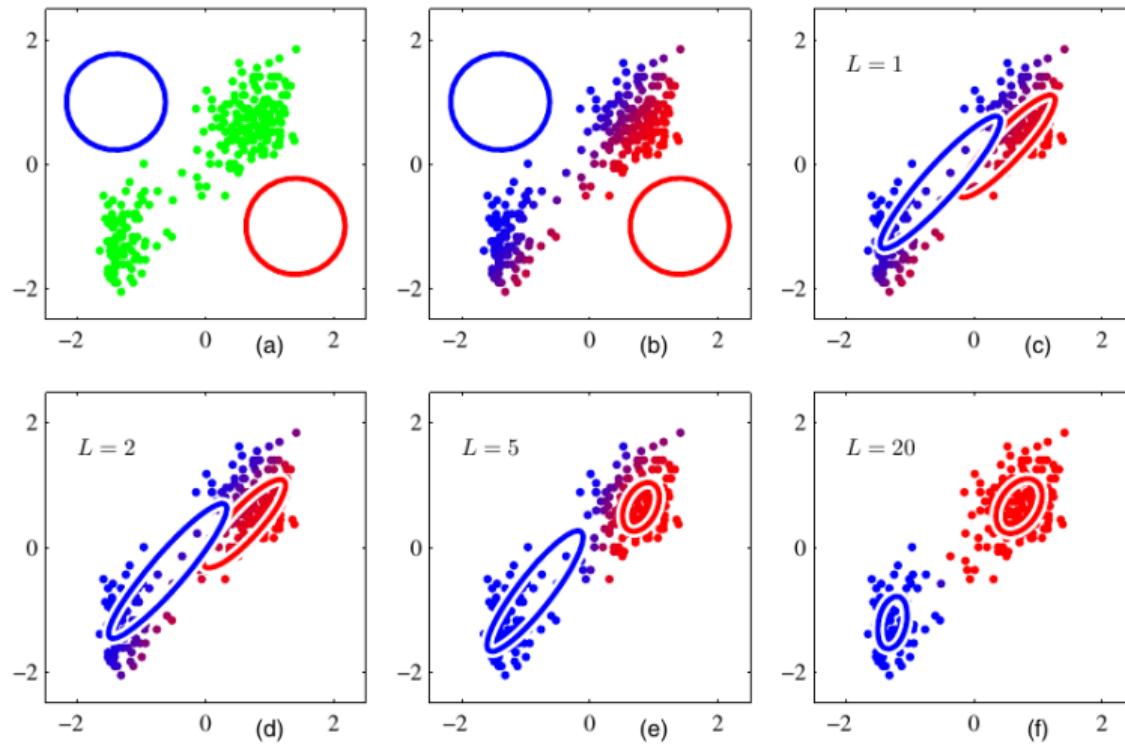
for $k = 1, 2$

$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^n \gamma_{ik}^{(t)} x_i}{\sum_{i=1}^n \gamma_{ik}^{(t)}},$$

$$\sigma_k^{(t+1)} = \sqrt{\frac{\sum_{i=1}^n \gamma_{ik}^{(t)} (x_i - \mu_k^{(t+1)})^2}{\sum_{i=1}^n \gamma_{ik}^{(t)}}},$$

$$\pi_k^{(t+1)} = \frac{\sum_{i=1}^n \gamma_{ik}^{(t)}}{N}.$$

EM in practice



Similar to K-means

- guaranteed to find a **local** maximum of the complete data likelihood
- somewhat sensitive to initial conditions

Better than K-means

- Gaussian distributions can model clusters with different shapes
- all data points are smoothly used to update all parameters

Probabilistic Modelling of Sequences

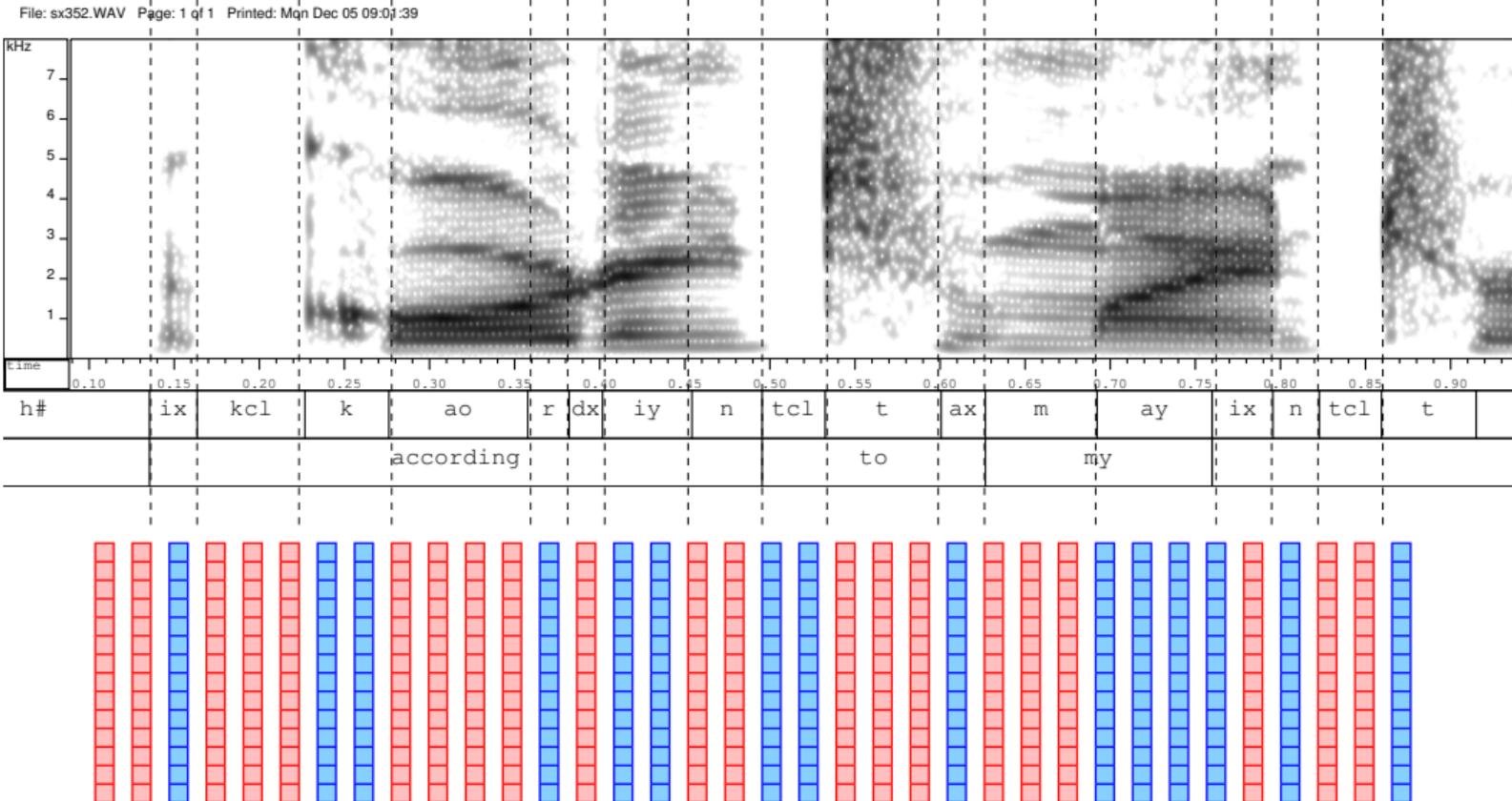
TTT4185 Machine Learning for Signal Processing

Giampiero Salvi

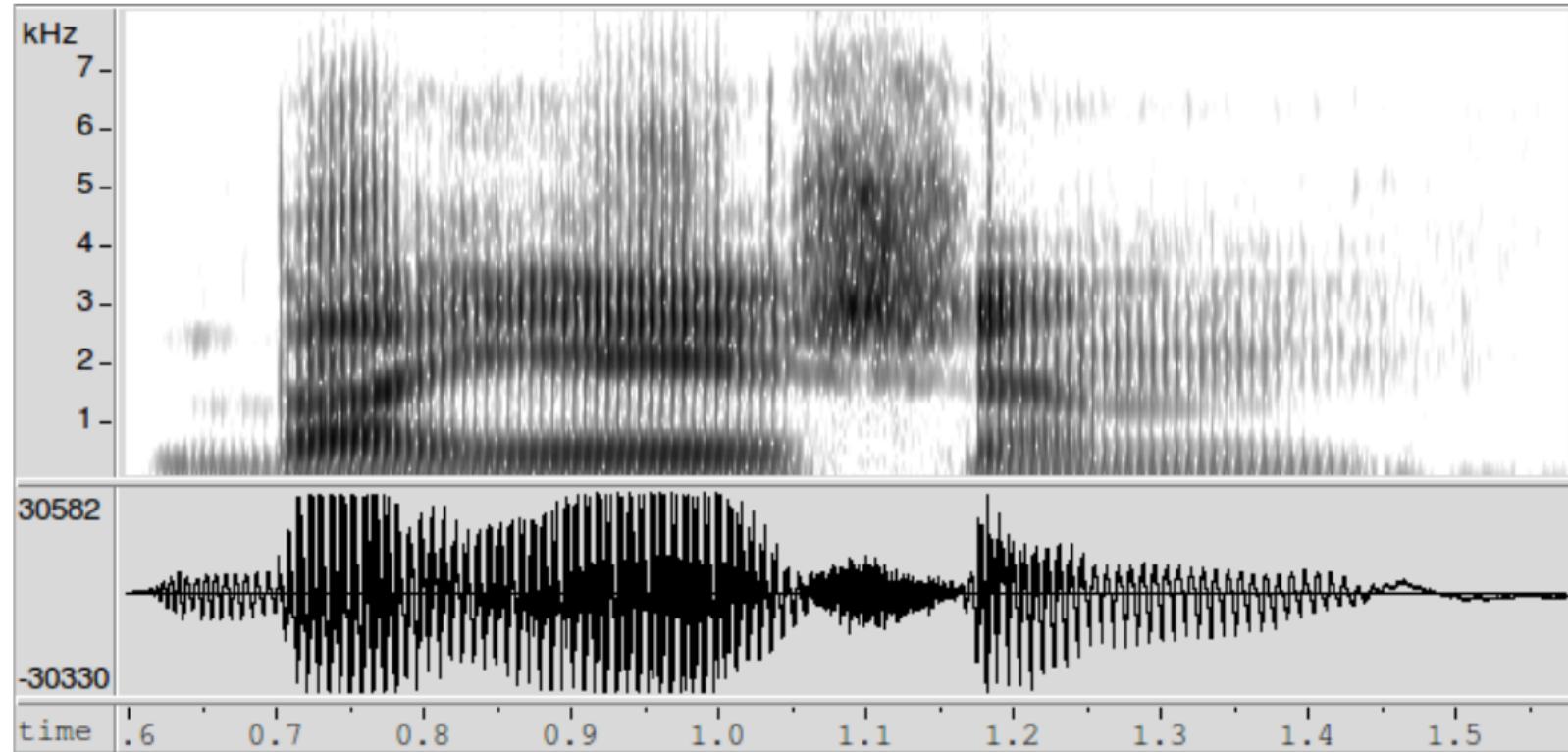
Department of Electronic Systems
NTNU

HT2020

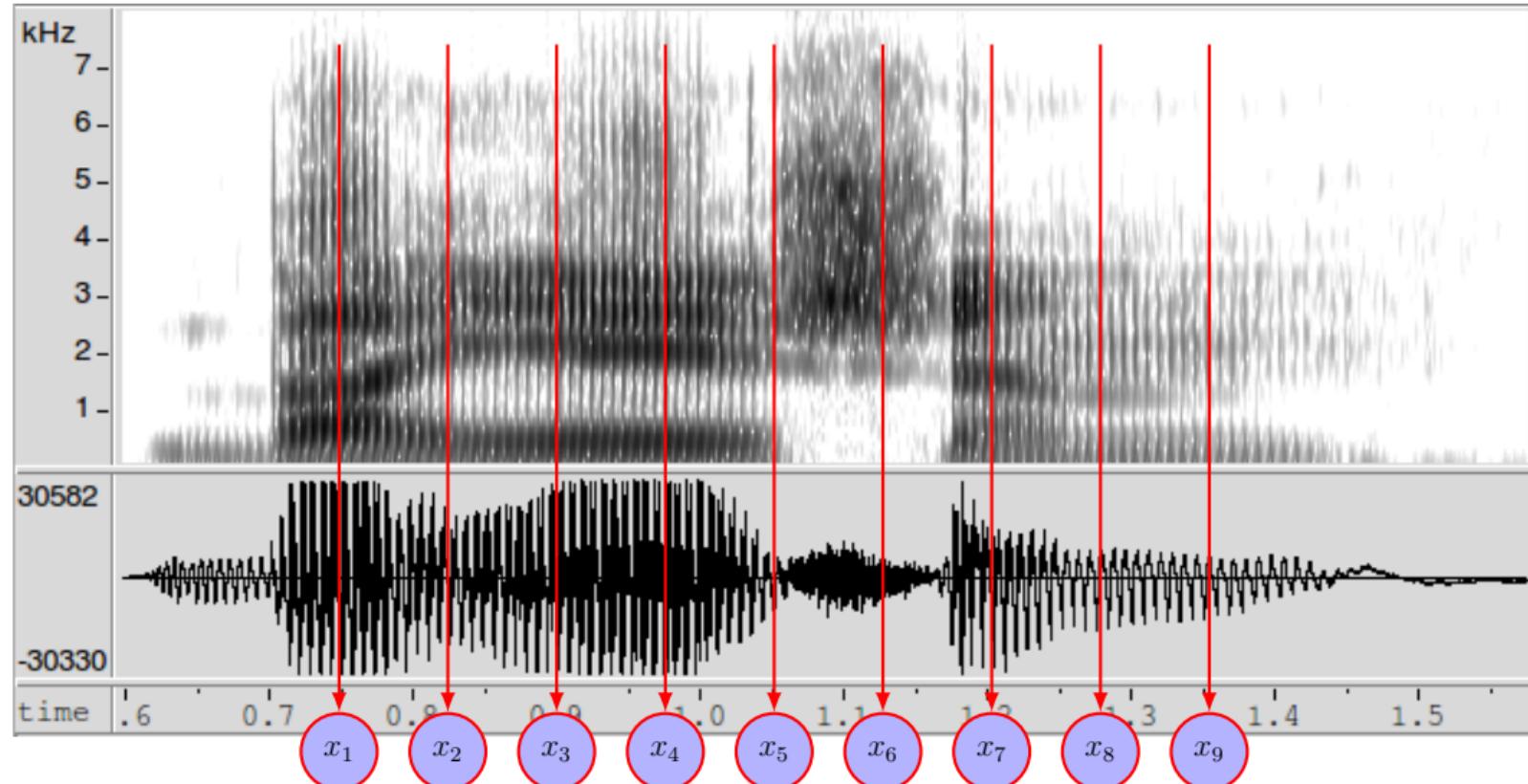
Frame-Based Processing



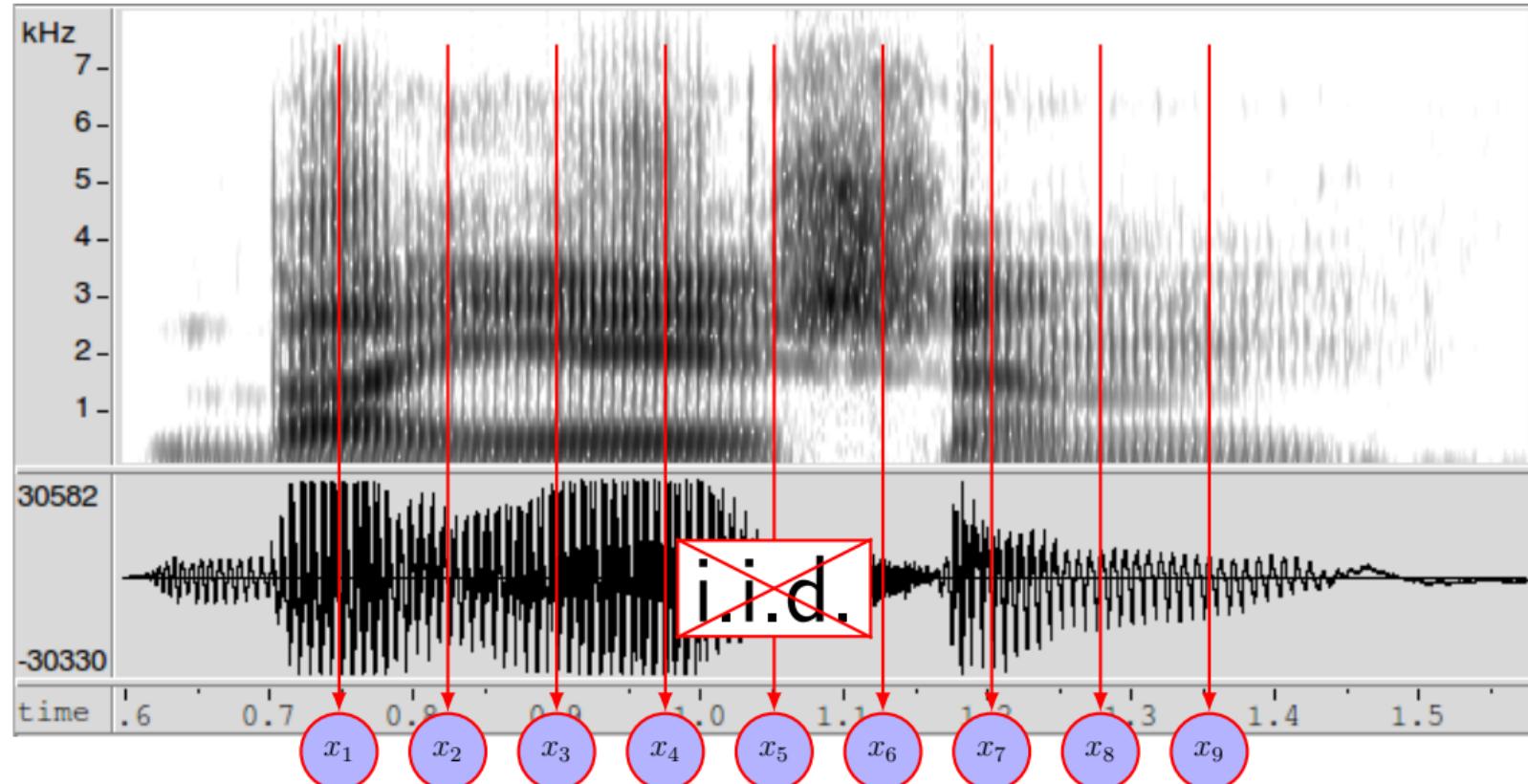
Sequences in Statistical Terms



Sequences in Statistical Terms



Sequences in Statistical Terms

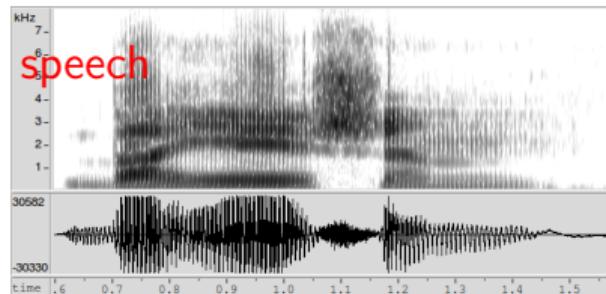


Sequential Data: Not Only Speech

Time sequences

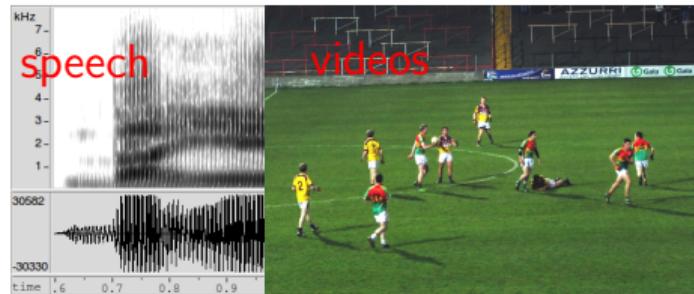
Sequential Data: Not Only Speech

Time sequences



Sequential Data: Not Only Speech

Time sequences



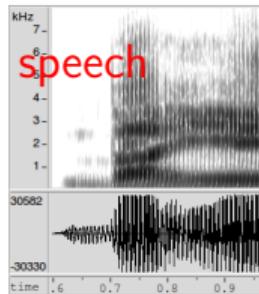
Sequential Data: Not Only Speech

Time sequences



Sequential Data: Not Only Speech

Time sequences

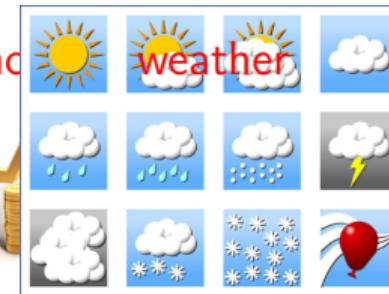


videos

econo

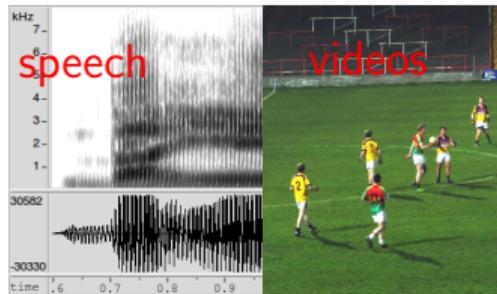
weather

weather



Sequential Data: Not Only Speech

Time sequences



Sequential Data: Not Only Speech

Time sequences

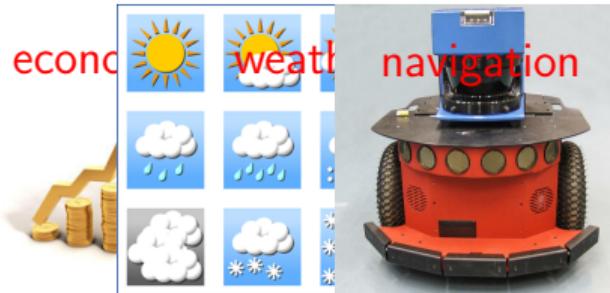
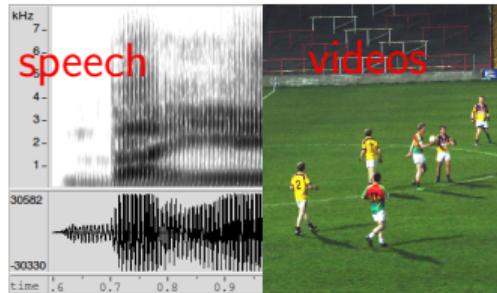


Timeless sequences



Sequential Data: Not Only Speech

Time sequences



Timeless sequences



Nel mezzo del cammin di nostra vita
mi ritrovai per una selva oscura,
ché la diritta via era smarrita.
Ahi quanto a dir qual era è cosa dura
esta selva selvaggia e aspra e forte
che nel pensier rinnova la paura!
Tant' è amara che poco è più morte;
ma per trattar del ben ch' lì vi trovai,
dirò de l'altre cose ch' lì v'ho scorte.
Io non so ben ridir com' i v'intrai,
tant' era pien di sonno a quel punto
che la verace via abbandonai.

Historical Perspective

- Hidden Markov Models first studied in the '60s¹²
- applied to ASR in the mid '70s³
- later seen as special case of Bayesian Networks⁴

¹R. Stratonovich. "Conditional Markov Processes". In: *Theory of Probability and its Applications* 5.2 (1960), pp. 156–178.

²L. E. Baum, T. Petrie, G. Soules, and N. Weiss. "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains". In: *Ann. Math. Statist.* 41.1 (1970), pp. 164–171.

³J. Baker. "The DRAGON system—An overview". In: *IEEE Trans. Acoust., Speech, Signal Process.* 23 (1975), pp. 24–29.

⁴J. Pearl. "Bayesian networks: a model of self-activated memory for evidential reasoning". In: *Proceedings of the 7th Conference of the Cognitive Science Society*. University of California, Irvine, Aug. 1985, pp. 329–334.

Bayesian Networks (reminder)

$$p(x_1, \dots, x_7) =$$

$$p(x_1)$$

$$p(x_2|x_1)$$

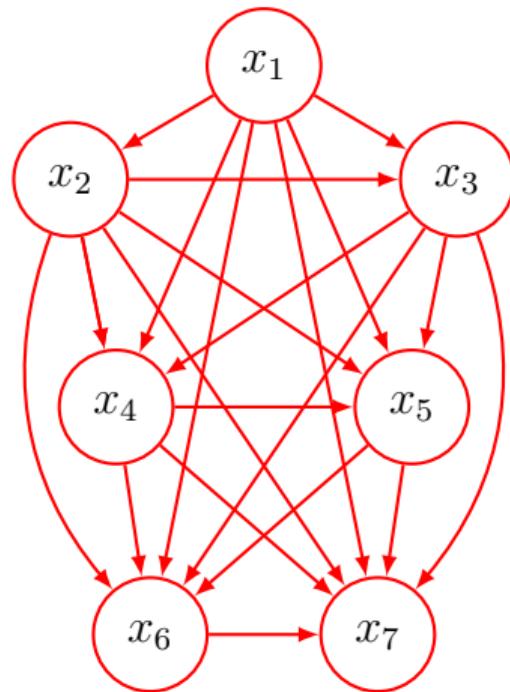
$$p(x_3|x_1, x_2)$$

$$p(x_4|x_1, x_2, x_3)$$

$$p(x_5|x_1, x_2, x_3, x_4)$$

$$p(x_6|x_1, x_2, x_3, x_4, x_5)$$

$$p(x_7|x_1, x_2, x_3, x_4, x_5, x_6)$$



Bayesian Networks (reminder)

$$p(x_1, \dots, x_7) =$$

$$p(x_1)$$

$$p(x_2|x_1)$$

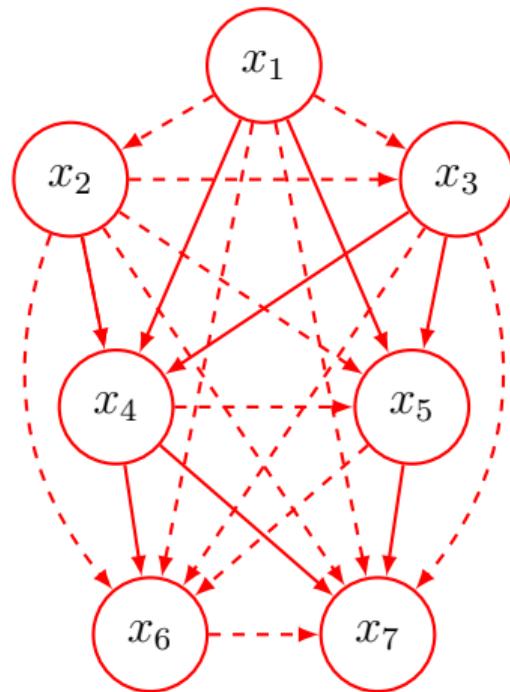
$$p(x_3|x_1, x_2)$$

$$p(x_4|x_1, x_2, x_3)$$

$$p(x_5|x_1, x_2, x_3, x_4)$$

$$p(x_6|x_1, x_2, x_3, x_4, x_5)$$

$$p(x_7|x_1, x_2, x_3, x_4, x_5, x_6)$$



Bayesian Networks (reminder)

$$p(x_1, \dots, x_7) =$$

$$p(x_1)$$

$$p(x_2)$$

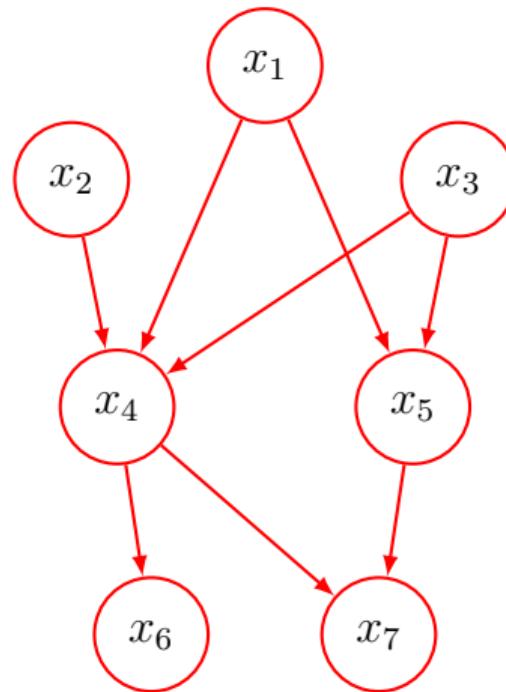
$$p(x_3)$$

$$p(x_4|x_1, x_2, x_3)$$

$$p(x_5|x_1, x_3)$$

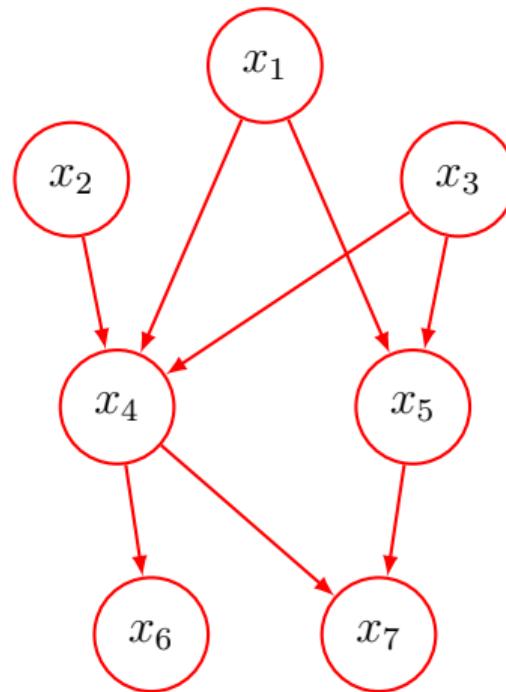
$$p(x_6|x_4)$$

$$p(x_7|x_4, x_5)$$



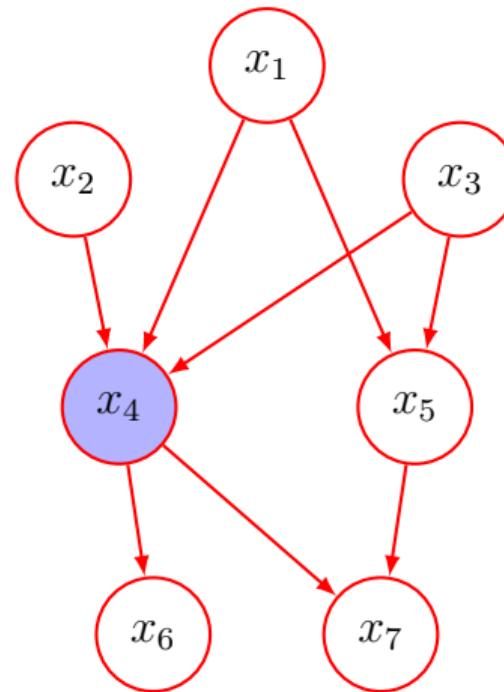
Bayesian Networks (reminder)

$$p(x_1, \dots, x_7) = \prod_{k=1}^K p(x_k | \text{pa}_k)$$



Bayesian Networks (reminder)

If we observe x_4 . . .



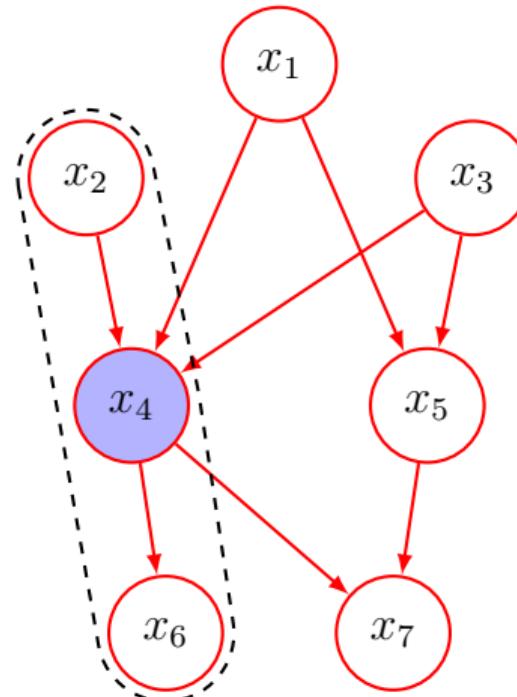
Bayesian Networks (reminder)

If we observe $x_4 \dots$

d -separation:

Head-to-tail:

x_2 and x_6 conditionally independent



Bayesian Networks (reminder)

If we observe $x_4 \dots$

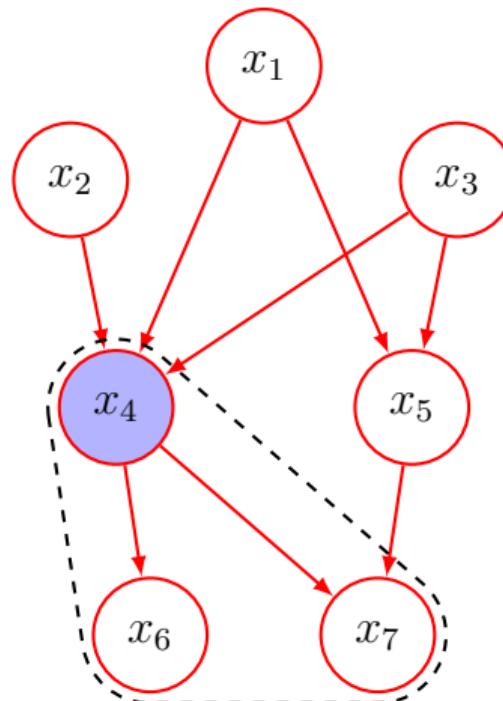
d -separation:

Head-to-tail:

x_2 and x_6 conditionally independent

Tail-to-tail:

x_6 and x_7 conditionally independent



Bayesian Networks (reminder)

If we observe $x_4 \dots$

d -separation:

Head-to-tail:

x_2 and x_6 conditionally independent

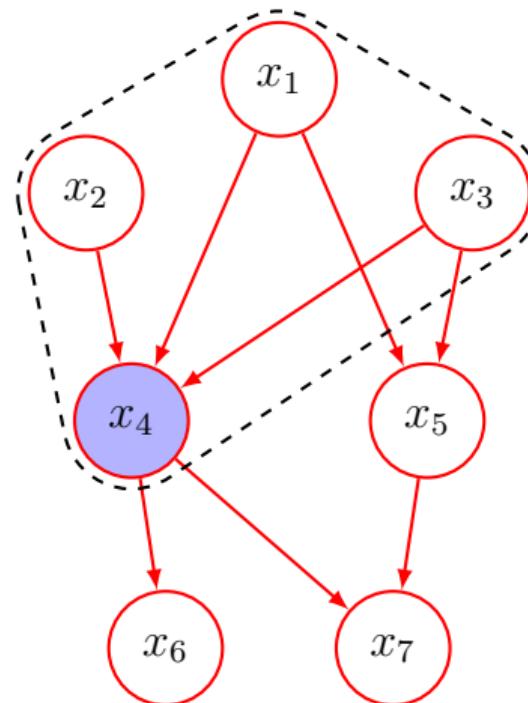
Tail-to-tail:

x_6 and x_7 conditionally independent

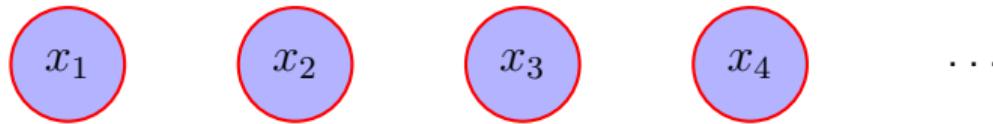
Head-to-head:

x_1, x_2 and x_3 dependent

(explaining away)

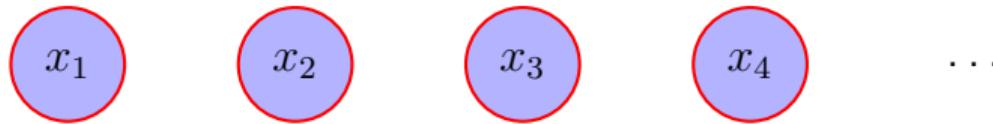


(Dynamic) Bayesian Networks



independence assumption (e.g. i.i.d) not satisfactory

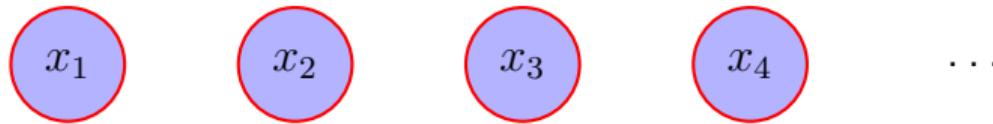
(Dynamic) Bayesian Networks



Most general case, applying chain rule recursively ($p(a, b) = p(a)p(b|a)$)

$$p(x_1, \dots, x_N) = p(x_1)p(x_2, \dots, x_N|x_1)$$

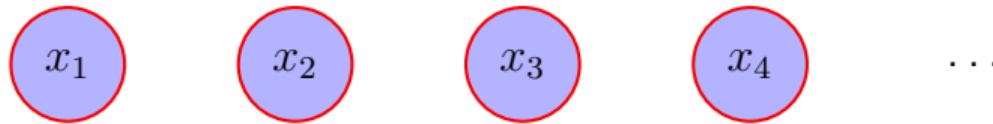
(Dynamic) Bayesian Networks



Most general case, applying chain rule recursively ($p(a, b) = p(a)p(b|a)$)

$$p(x_1, \dots, x_N) = p(x_1)p(x_2|x_1)p(x_3, \dots, x_N|x_1, x_2)$$

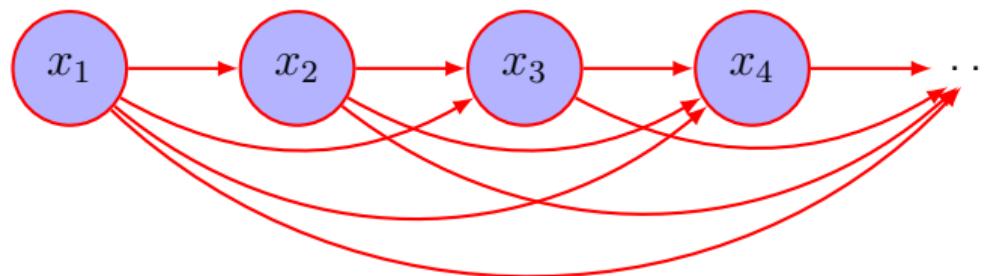
(Dynamic) Bayesian Networks



Most general case, applying chain rule recursively ($p(a, b) = p(a)p(b|a)$)

$$\begin{aligned} p(x_1, \dots, x_N) &= p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)\cdots \\ &\quad \cdots p(x_N|x_1, \dots, x_{N-1}) \end{aligned}$$

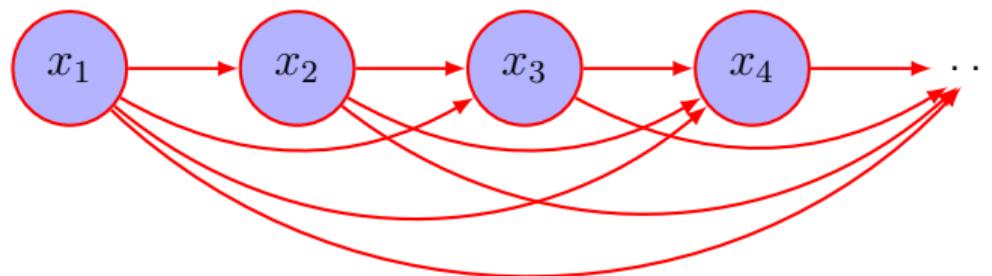
(Dynamic) Bayesian Networks



Most general case, applying chain rule recursively ($p(a, b) = p(a)p(b|a)$)

$$\begin{aligned} p(x_1, \dots, x_N) &= p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)\cdots \\ &\quad \cdots p(x_N|x_1, \dots, x_{N-1}) \end{aligned}$$

(Dynamic) Bayesian Networks

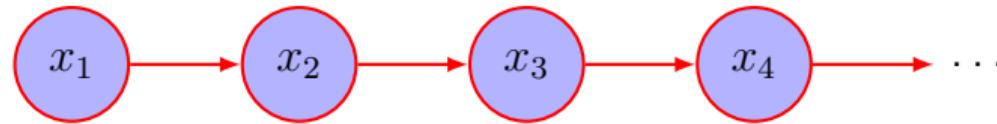


Most general case, applying chain rule recursively ($p(a, b) = p(a)p(b|a)$)

$$\begin{aligned} p(x_1, \dots, x_N) &= p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)\cdots \\ &\quad \cdots p(x_N|x_1, \dots, x_{N-1}) \end{aligned}$$

Grows quadratically with sequence length (N)!!!

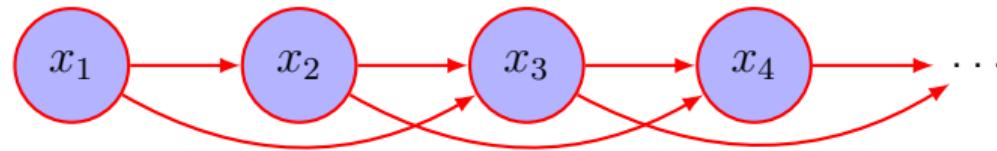
Markov assumption



First order Markov assumption: $p(x_n|x_1, \dots, x_{n-1}) \approx p(x_n|x_{n-1})$

$$p(x_1, \dots, x_N) = p(x_1) \prod_{n=2}^N p(x_n|x_{n-1})$$

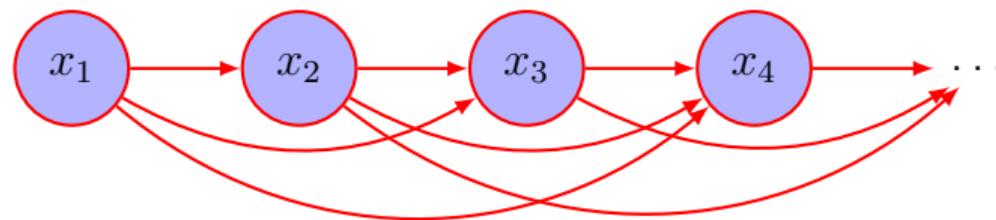
Markov assumption



Second order Markov assumption:

$$p(x_1, \dots, x_N) = p(x_1)p(x_2|x_1) \prod_{n=3}^N p(x_n|x_{n-2}, x_{n-1})$$

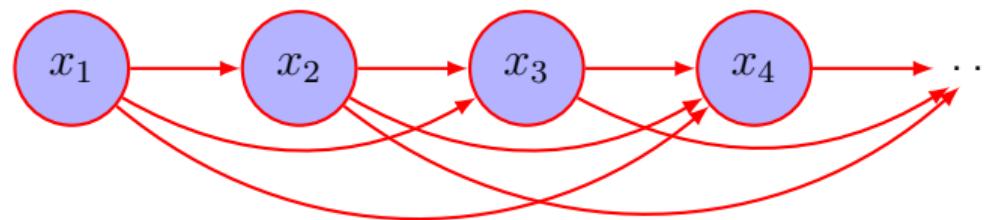
Markov assumption



Third order Markov assumption:

$$p(x_1, \dots, x_N) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \prod_{n=4}^N p(x_n|x_{n-3}, x_{n-2}, x_{n-1})$$

Markov assumption



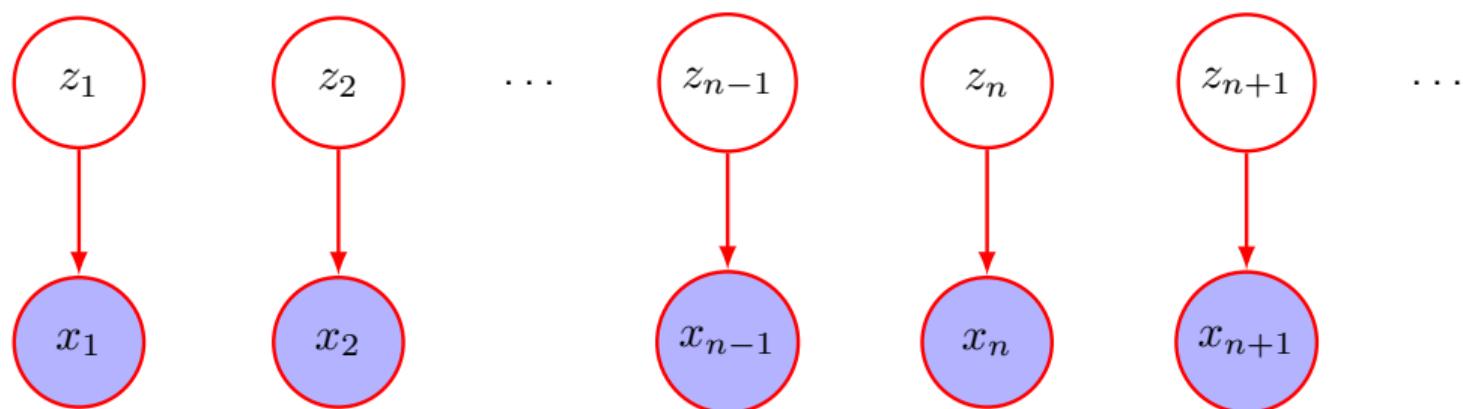
Third order Markov assumption:

$$p(x_1, \dots, x_N) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \prod_{n=4}^N p(x_n|x_{n-3}, x_{n-2}, x_{n-1})$$

Grows quadratically with order!!!

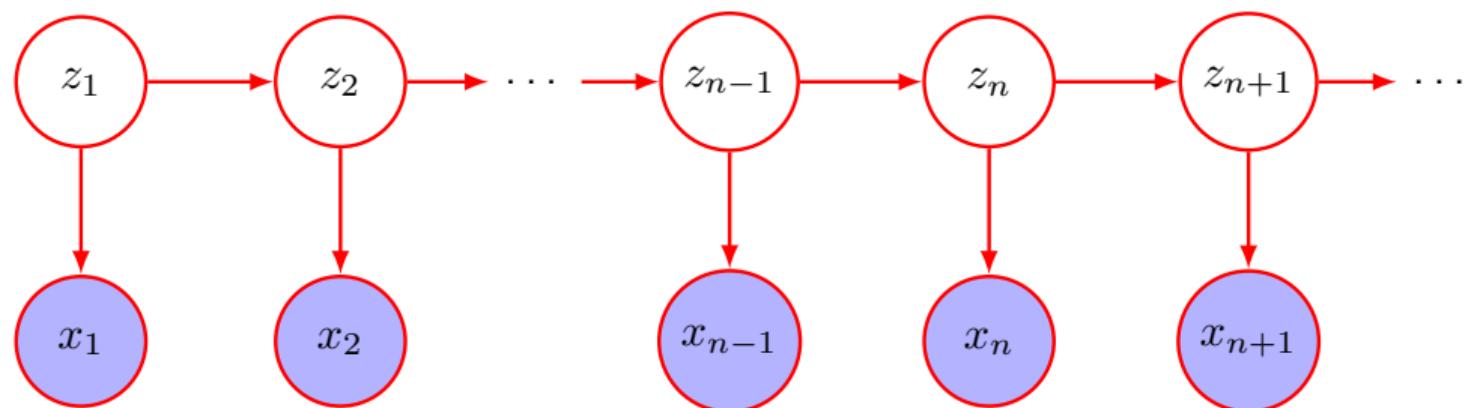
Mixture Models

Adding latent variables z_n

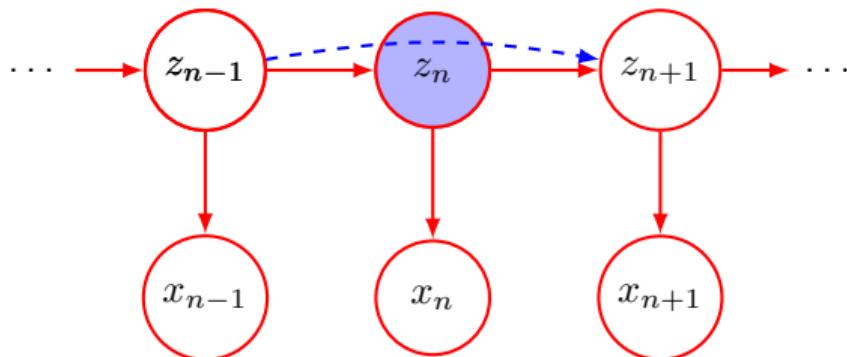


State Space Models

Adding latent variables z_n

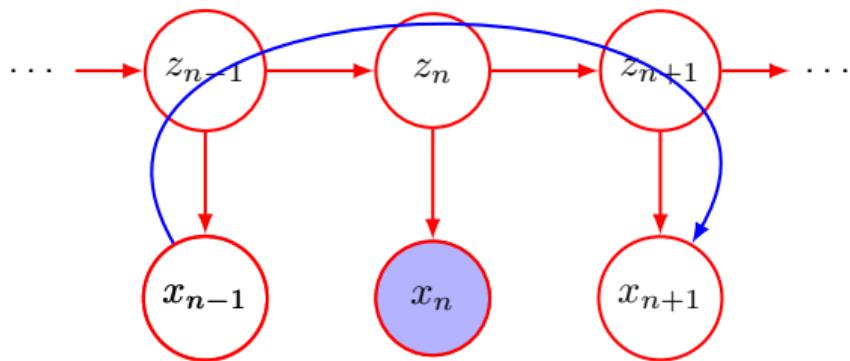


State Space Models: Properties



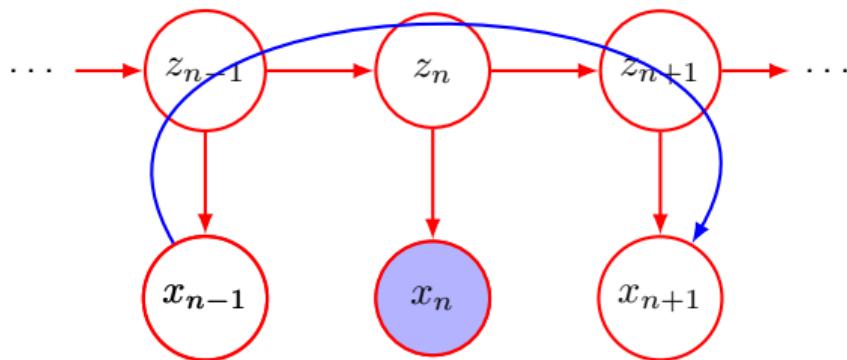
- given z_n , z_{n+1} is independent of z_1, \dots, z_{n-1}
 $p(z_{n+1}|z_1, \dots, z_n) = p(z_{n+1}|z_n)$

State Space Models: Properties



- given z_n , z_{n+1} is independent of z_1, \dots, z_{n-1}
 $p(z_{n+1}|z_1, \dots, z_n) = p(z_{n+1}|z_n)$
- $p(x_{n+1}|x_1, \dots, x_n)$ does not simplify

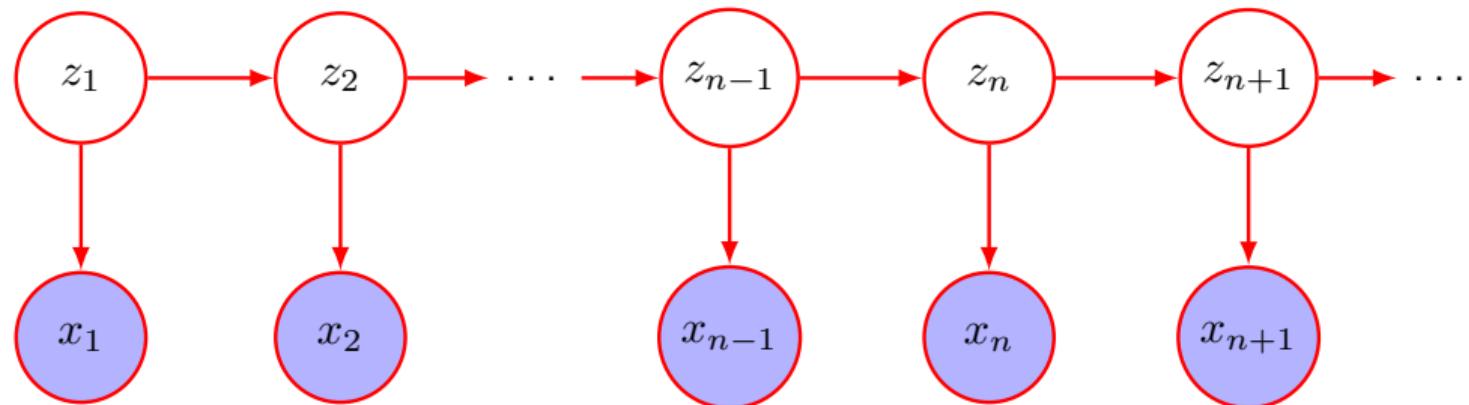
State Space Models: Properties



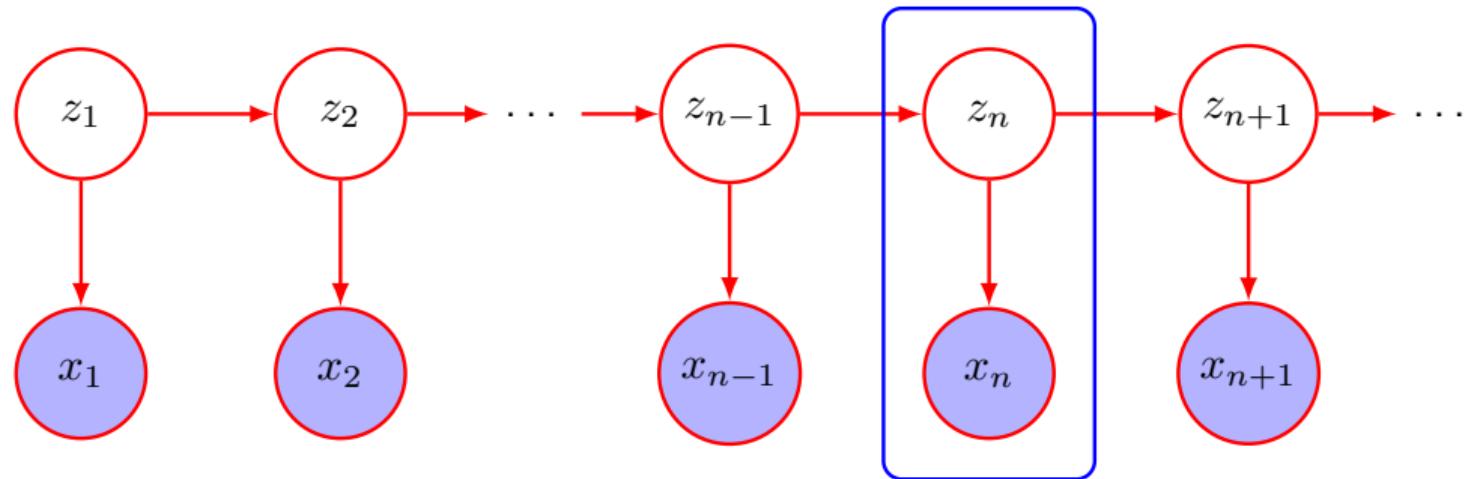
- given z_n , z_{n+1} is independent of z_1, \dots, z_{n-1}
 $p(z_{n+1}|z_1, \dots, z_n) = p(z_{n+1}|z_n)$
- $p(x_{n+1}|x_1, \dots, x_n)$ does not simplify

We have modelled indefinitely long dependencies with a limited set of parameters!

Stationary State Space Models

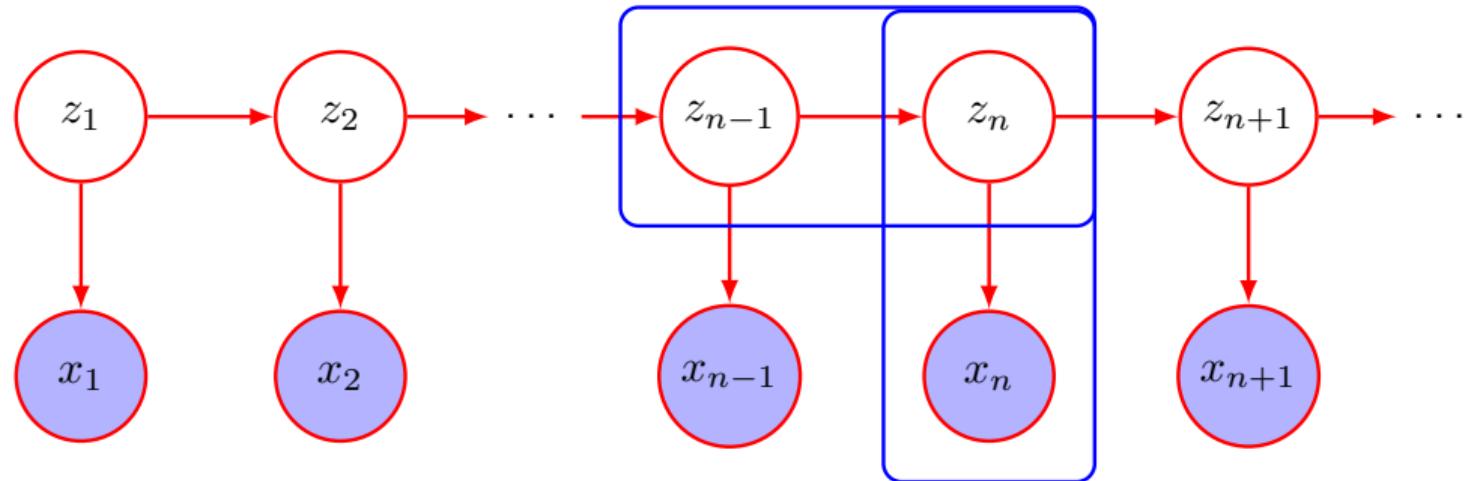


Stationary State Space Models



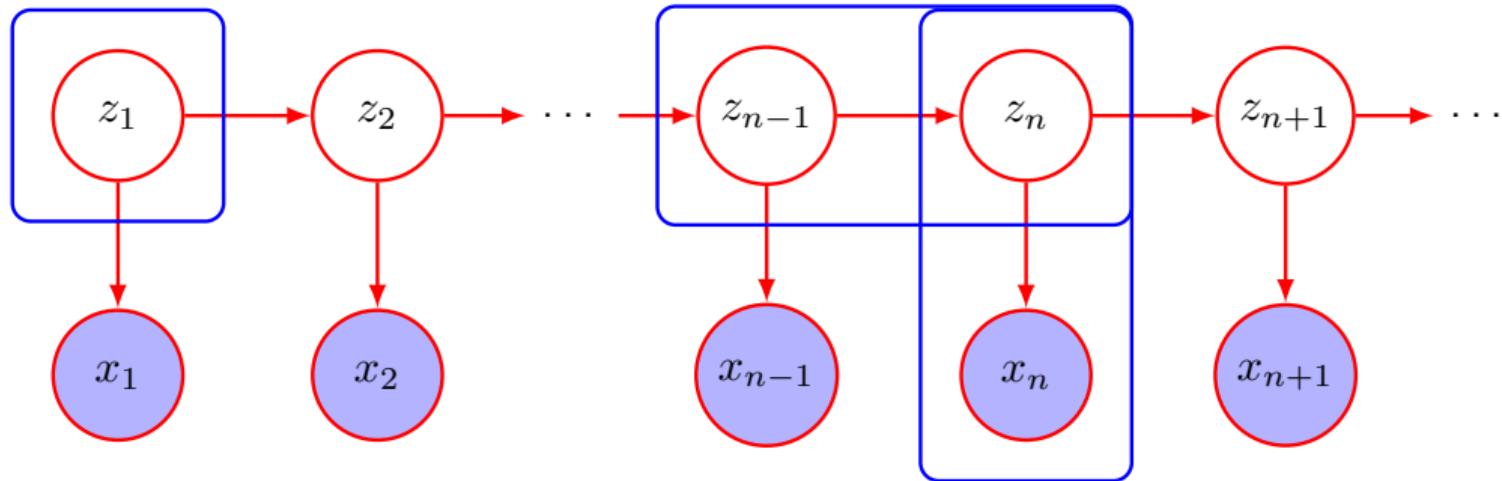
- Emission: $p(x_n|z_n)$

Stationary State Space Models



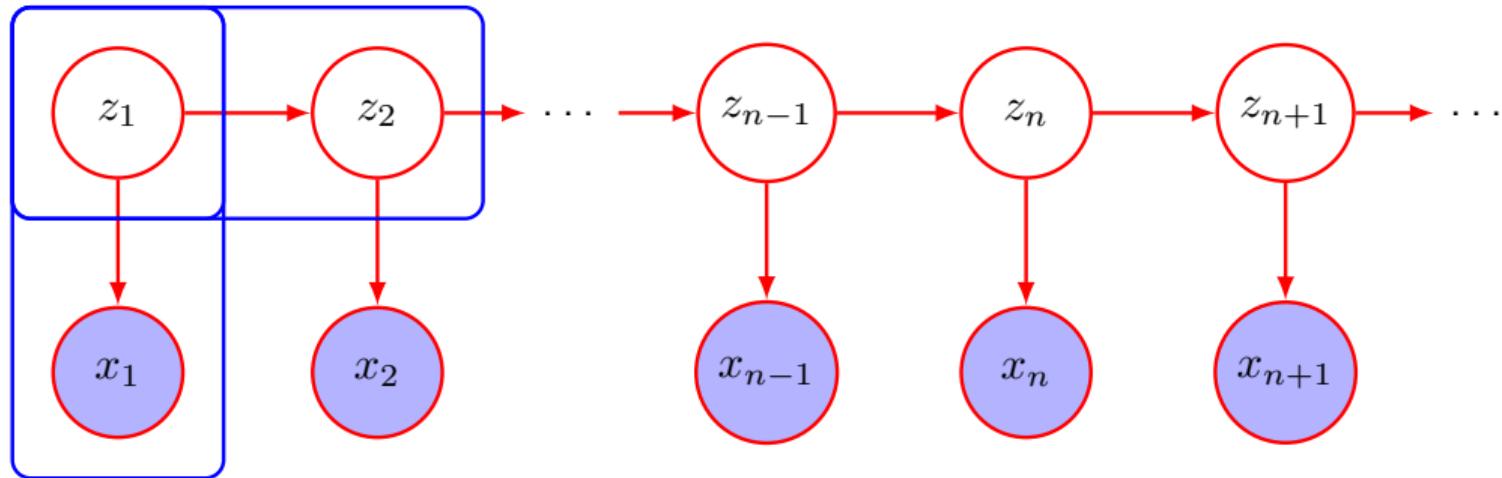
- Emission: $p(x_n|z_n)$
- Transition: $p(z_n|z_{n-1})$

Stationary State Space Models



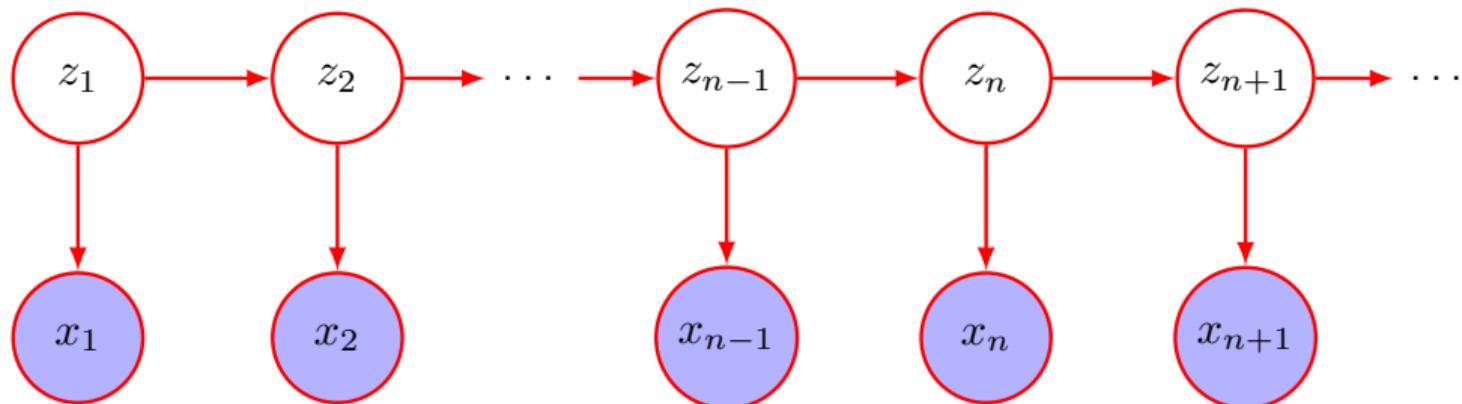
- Emission: $p(x_n|z_n)$
- Transition: $p(z_n|z_{n-1})$
- Initial: $p(z_1)$

Stationary State Space Models



- Emission: $p(x_n|z_n)$
- Transition: $p(z_n|z_{n-1})$
- Initial: $p(z_1)$

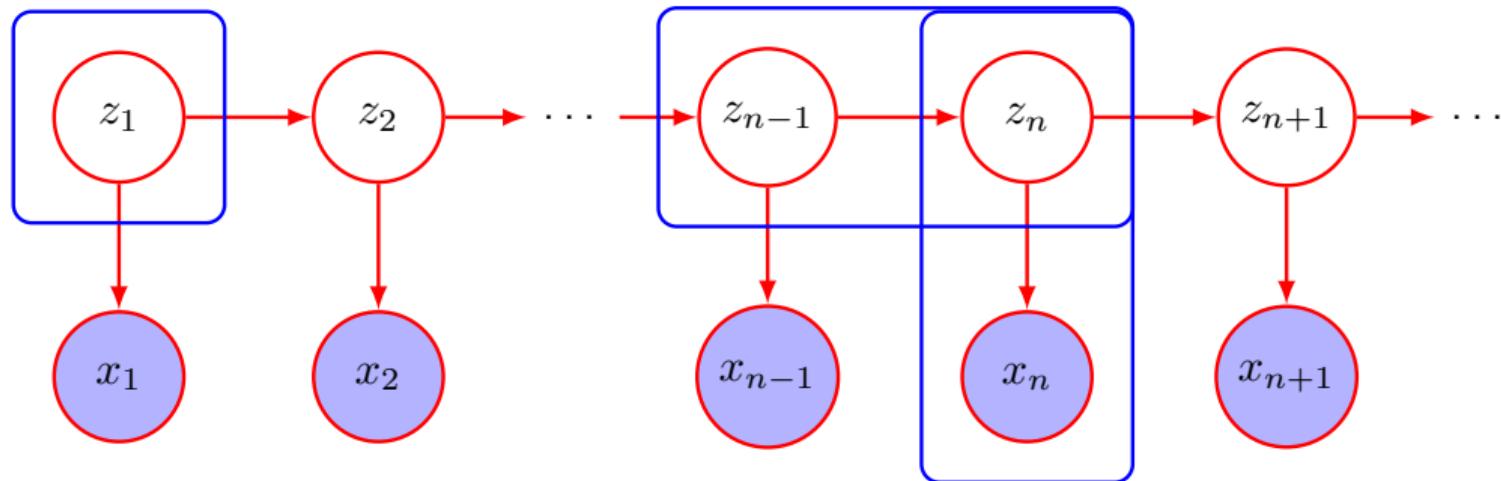
State Space Models Instances



- if z_n are discrete: Hidden Markov Models
- if z_n are continuous: Linear Dynamical Systems

Hidden Markov Models

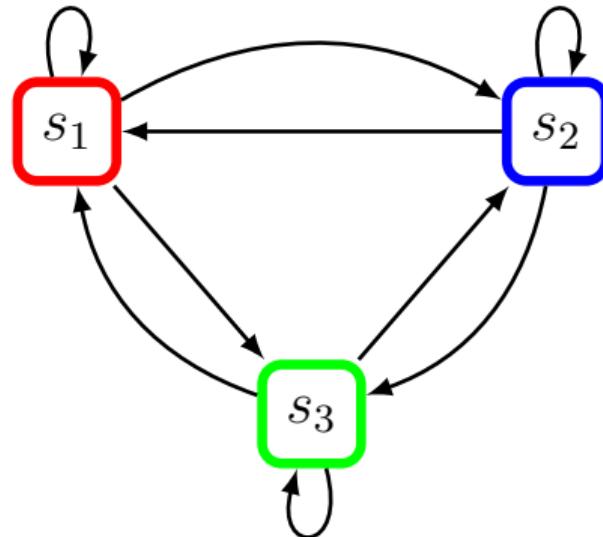
State space models with discrete z_n



- Emission: $p(x_n|z_n) = p(x_n|z_n, \phi)$
equivalent to Mixture Model
- Transition: $p(z_n|z_{n-1}) = p(z_n|z_{n-1}, A)$
- Initial: $p(z_1) = p(z_1|\pi)$

Hidden Markov Models (HMMs)

Ergodic HMM



Elements:

set of states:

$$S = \{s_1, s_2, s_3\}$$

transition probabilities:

$$A(s_a, s_b) = P(s_b, t | s_a, t - 1)$$

prior probabilities:

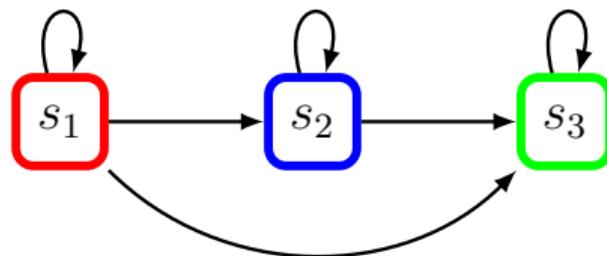
$$\pi(s_a) = P(s_a, t_0)$$

state to observation probs:

$$\phi(o, s_a) = P(o | s_a)$$

Hidden Markov Models (HMMs)

Left-to-right HMM



Elements:

set of states:

$$S = \{s_1, s_2, s_3\}$$

transition probabilities:

$$A(s_a, s_b) = P(s_b, t | s_a, t - 1)$$

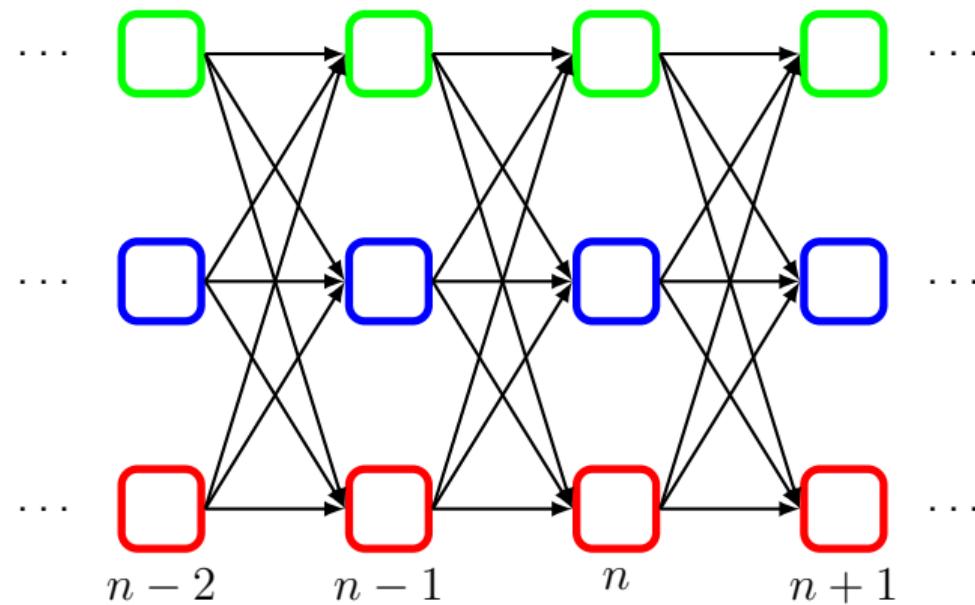
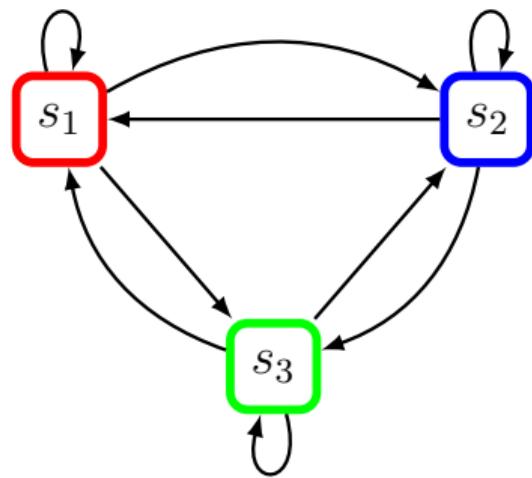
prior probabilities:

$$\pi(s_a) = P(s_a, t_0)$$

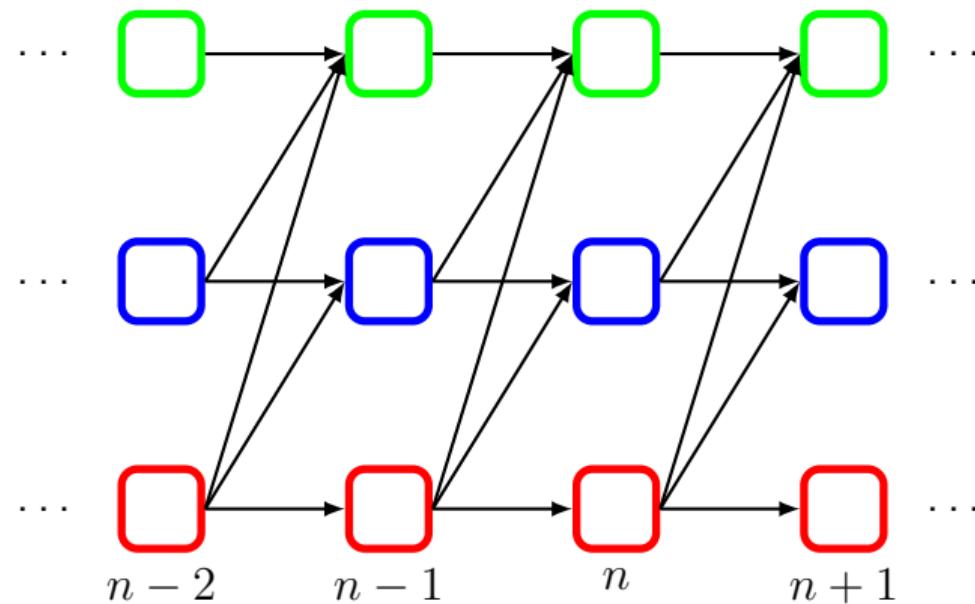
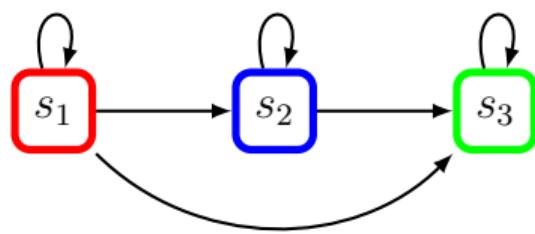
state to observation probs:

$$\phi(o, s_a) = P(o | s_a)$$

HMMs: Trellis (Lattice)



HMMs: Trellis (Lattice)



A probabilistic perspective: Bayes' rule

$$P(\text{words}|\text{sounds}) = \frac{P(\text{sounds}|\text{words})P(\text{words})}{P(\text{sounds})}$$

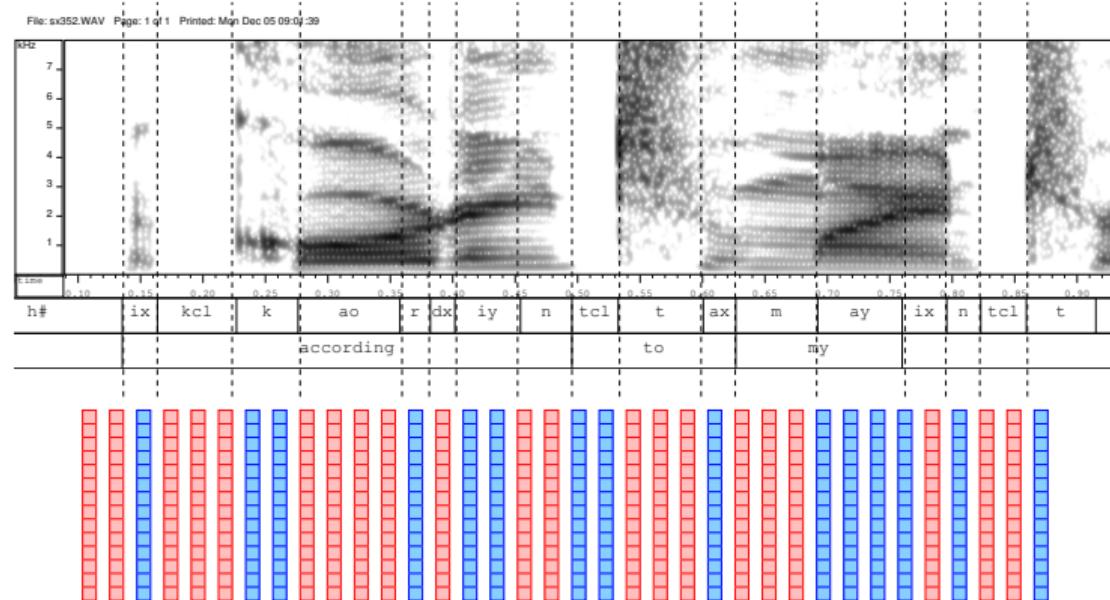
- $P(\text{sounds}|\text{words})$ can be estimated from training data and transcriptions
- $P(\text{words})$: *a priori* probability of the words (Language Model)
- $P(\text{sounds})$: *a priori* probability of the sounds (constant, can be ignored)

Probabilistic Modelling

Problem: How do we model $P(\text{sounds}|\text{words})$?

Probabilistic Modelling

Problem: How do we model $P(\text{sounds}|\text{words})$?

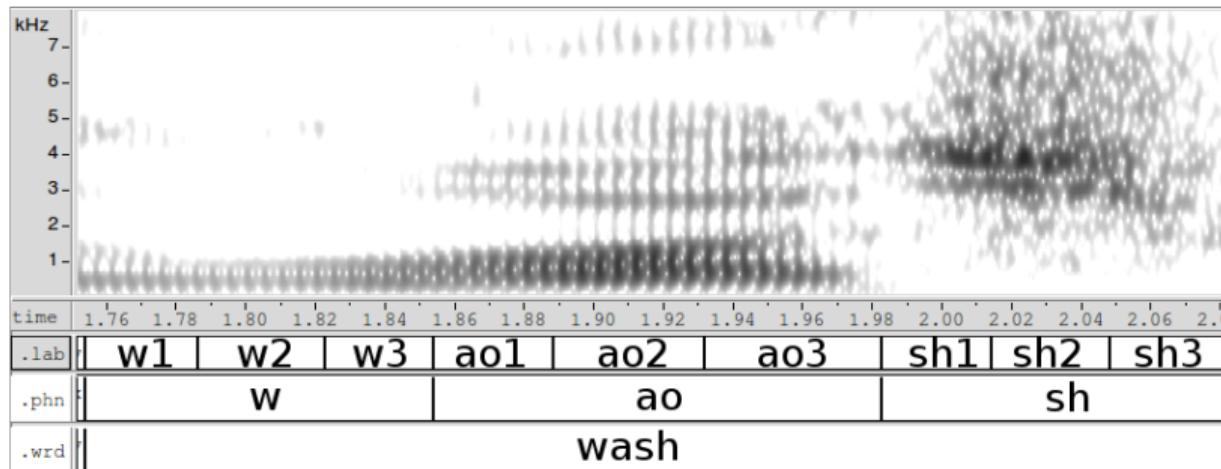


Every feature vector (observation at time t) is a continuous stochastic variable
(e.g. MFCC)

Stationarity

Problem: speech is not stationary

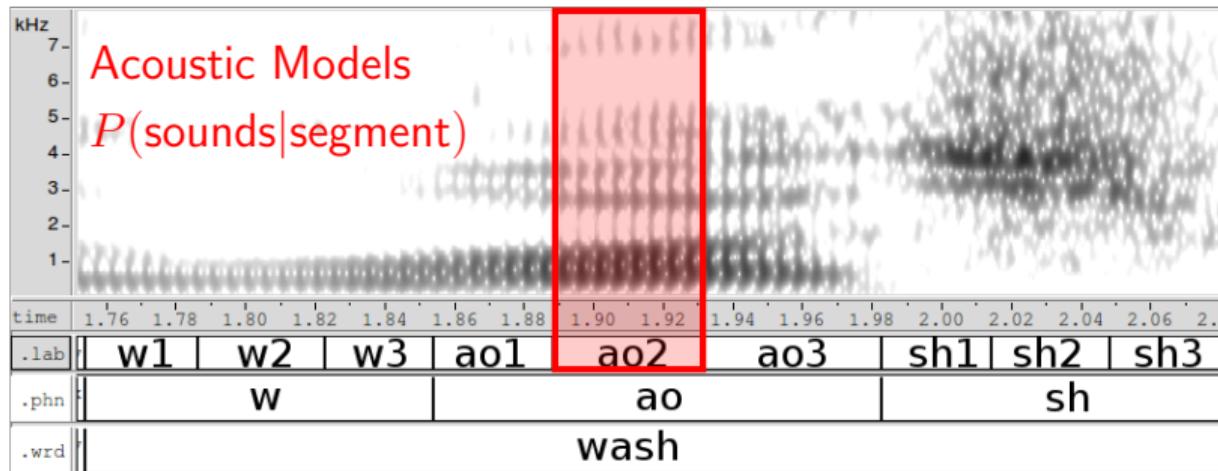
- we need to model short segments independently
- the **fundamental unit** can not be the word, but must be shorter
- usually we model three segments for each phoneme



Stationarity

Problem: speech is not stationary

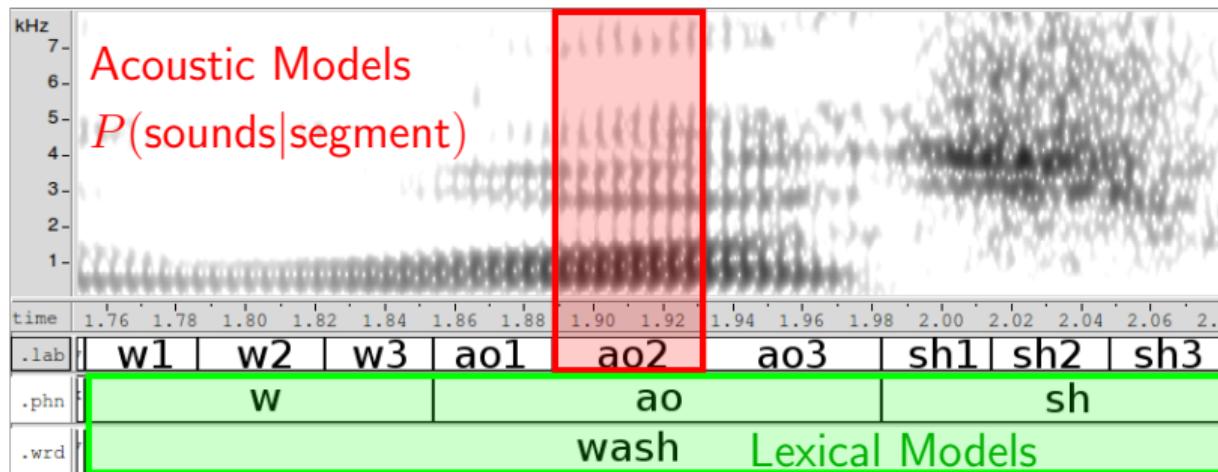
- we need to model short segments independently
- the **fundamental unit** can not be the word, but must be shorter
- usually we model three segments for each phoneme



Stationarity

Problem: speech is not stationary

- we need to model short segments independently
- the **fundamental unit** can not be the word, but must be shorter
- usually we model three segments for each phoneme



Local probabilities (frame-wise)

If **segment** sufficiently short

$$P(\text{sounds}|\text{segment})$$

can be modelled with standard probability distributions

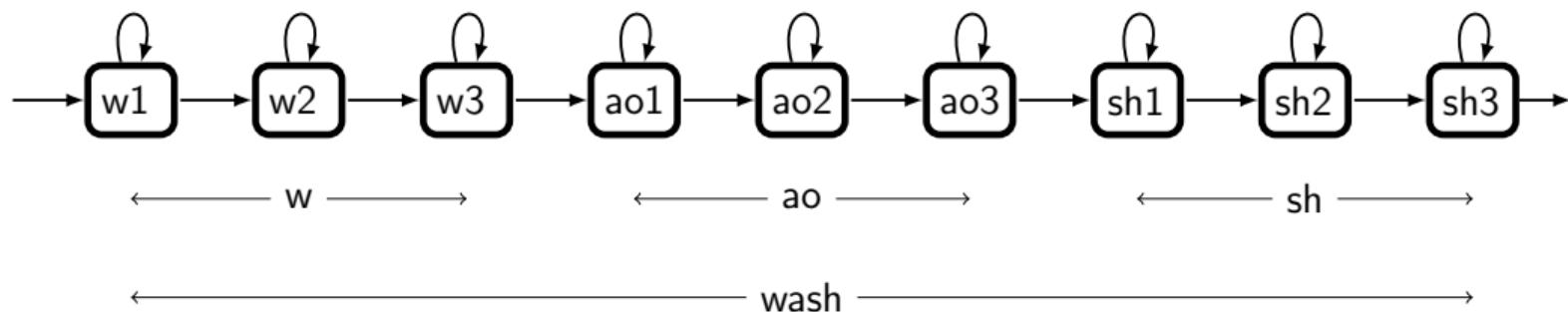
$$\phi_{s_a}(x) = P(x|s_a)$$

Usually Gaussian or Gaussian Mixture

Global Probabilities (utterance)

Problem: How do we combine the different $P(\text{sounds}|\text{segment})$ to form $P(\text{sounds}|\text{words})$?

Answer: Hidden Markov Model (HMM)



HMM-questions (Inference)

- 1 what is the probability that the model has generated the sequence of observations? (isolated word recognition)

⁵A. J. Viterbi. "Error Bounds for Convolutional Codes and an Asymptotically optimum decoding algorithm". In: *IEEE Trans. Inf. Theory* IT-13 (Apr. 1967), pp. 260–269.

⁶L. E. Baum, T. Petrie, G. Soules, and N. Weiss. "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains". In: *Ann. Math. Statist.* 41.1 (1970), pp. 164–171.

HMM-questions (Inference)

- 1 what is the probability that the model has generated the sequence of observations? (isolated word recognition) **forward algorithm**

⁵A. J. Viterbi. "Error Bounds for Convolutional Codes and an Asymptotically optimum decoding algorithm". In: *IEEE Trans. Inf. Theory* IT-13 (Apr. 1967), pp. 260–269.

⁶L. E. Baum, T. Petrie, G. Soules, and N. Weiss. "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains". In: *Ann. Math. Statist.* 41.1 (1970), pp. 164–171.

HMM-questions (Inference)

- ① what is the probability that the model has generated the sequence of observations? (isolated word recognition) **forward algorithm**
- ② what is the most likely state sequence given the observation sequence? (continuous speech recognition)

⁵A. J. Viterbi. "Error Bounds for Convolutional Codes and an Asymptotically optimum decoding algorithm". In: *IEEE Trans. Inf. Theory* IT-13 (Apr. 1967), pp. 260–269.

⁶L. E. Baum, T. Petrie, G. Soules, and N. Weiss. "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains". In: *Ann. Math. Statist.* 41.1 (1970), pp. 164–171.

HMM-questions (Inference)

- ① what is the probability that the model has generated the sequence of observations? (isolated word recognition) **forward algorithm**
- ② what is the most likely state sequence given the observation sequence? (continuous speech recognition) **Viterbi algorithm⁵**

⁵A. J. Viterbi. "Error Bounds for Convolutional Codes and an Asymptotically optimum decoding algorithm". In: *IEEE Trans. Inf. Theory* IT-13 (Apr. 1967), pp. 260–269.

⁶L. E. Baum, T. Petrie, G. Soules, and N. Weiss. "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains". In: *Ann. Math. Statist.* 41.1 (1970), pp. 164–171.

HMM-questions (Inference)

- ① what is the probability that the model has generated the sequence of observations? (isolated word recognition) **forward algorithm**
- ② what is the most likely state sequence given the observation sequence? (continuous speech recognition) **Viterbi algorithm⁵**
- ③ how can the model parameters be estimated from examples? (training)

⁵A. J. Viterbi. "Error Bounds for Convolutional Codes and an Asymptotically optimum decoding algorithm". In: *IEEE Trans. Inf. Theory* IT-13 (Apr. 1967), pp. 260–269.

⁶L. E. Baum, T. Petrie, G. Soules, and N. Weiss. "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains". In: *Ann. Math. Statist.* 41.1 (1970), pp. 164–171.

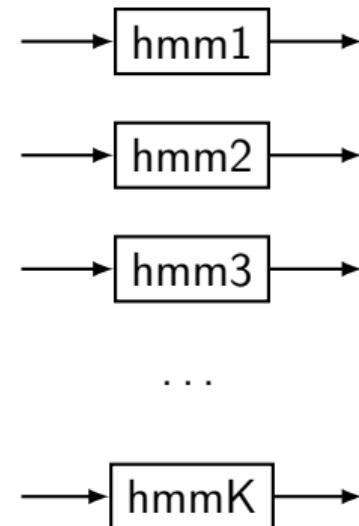
HMM-questions (Inference)

- ① what is the probability that the model has generated the sequence of observations? (isolated word recognition) **forward algorithm**
- ② what is the most likely state sequence given the observation sequence? (continuous speech recognition) **Viterbi algorithm⁵**
- ③ how can the model parameters be estimated from examples? (training)
Baum-Welch⁶

⁵A. J. Viterbi. "Error Bounds for Convolutional Codes and an Asymptotically optimum decoding algorithm". In: *IEEE Trans. Inf. Theory* IT-13 (Apr. 1967), pp. 260–269.

⁶L. E. Baum, T. Petrie, G. Soules, and N. Weiss. "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains". In: *Ann. Math. Statist.* 41.1 (1970), pp. 164–171.

Isolated Words Recognition



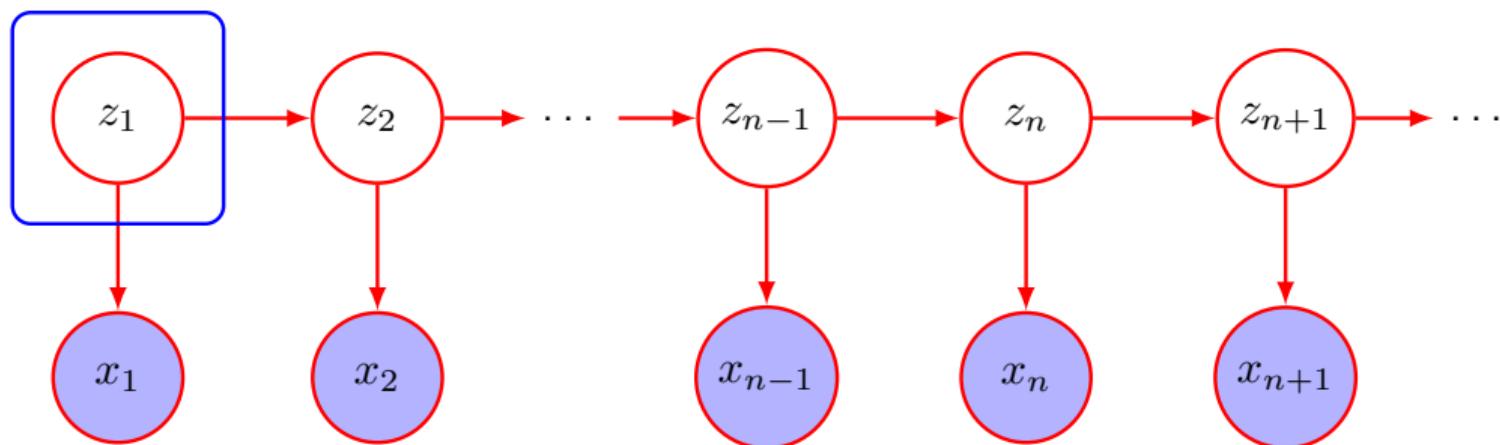
Compare Likelihoods (forward algorithm)

HMM Inference: Joint Distribution

$$X = \{x_1, \dots, x_N\}$$

$$Z = \{z_1, \dots, z_N\}$$

$$P(X, Z | \theta) = p(z_1 | \pi) \left[\prod_{n=2}^N p(z_n | z_{n-1}, A) \right] \prod_{m=1}^N p(x_m | z_m, \phi)$$

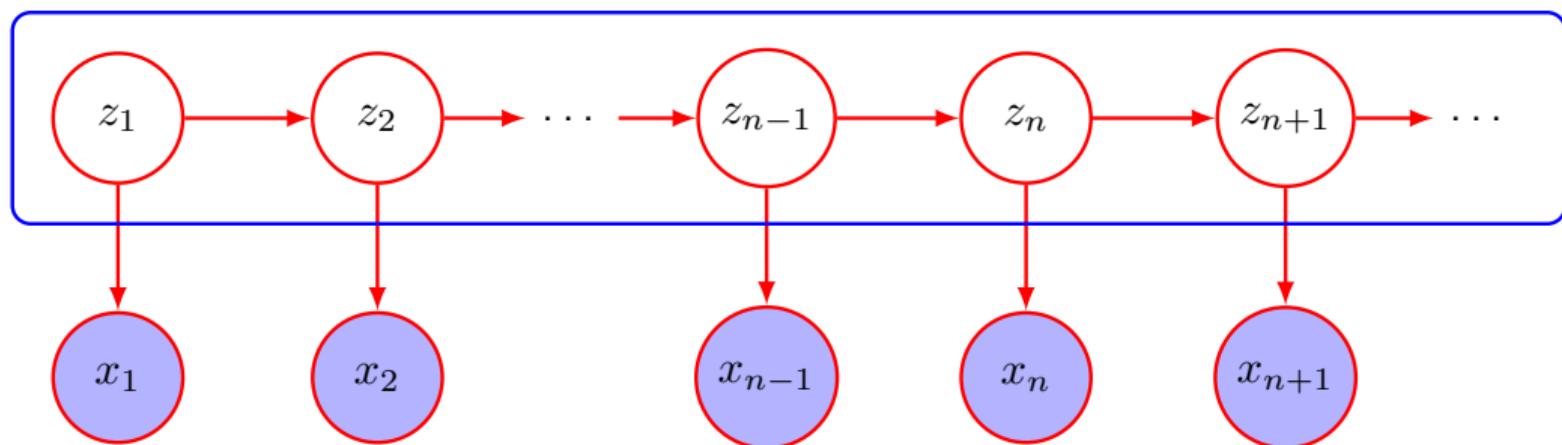


HMM Inference: Joint Distribution

$$X = \{x_1, \dots, x_N\}$$

$$Z = \{z_1, \dots, z_N\}$$

$$P(X, Z | \theta) = p(z_1 | \pi) \left[\prod_{n=2}^N p(z_n | z_{n-1}, A) \right] \prod_{m=1}^N p(x_m | z_m, \phi)$$

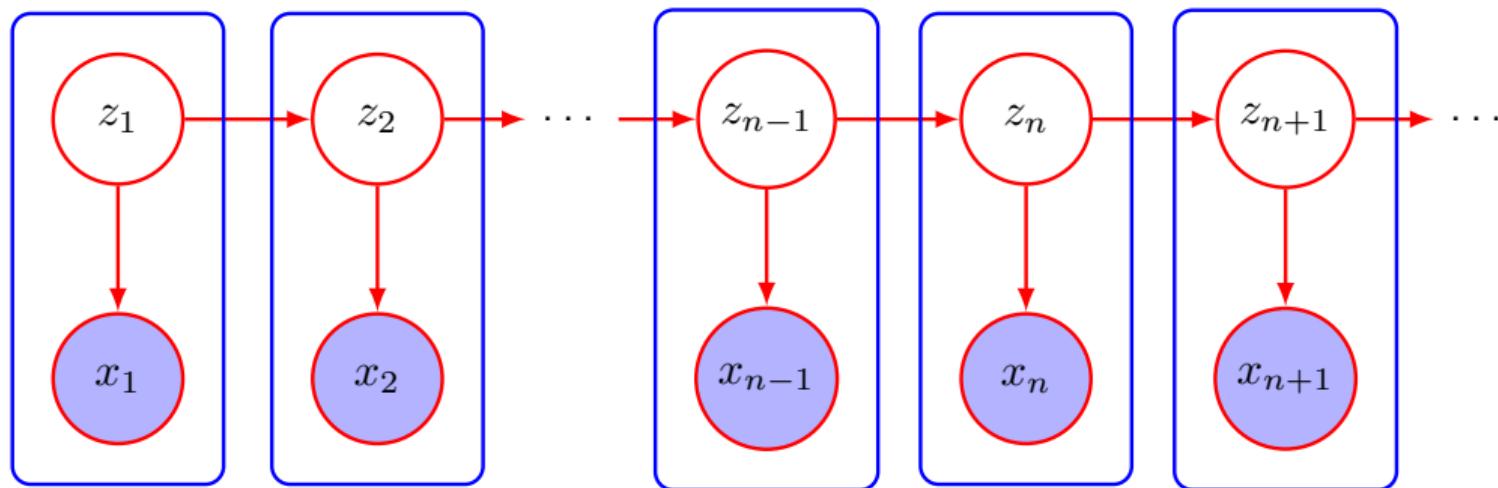


HMM Inference: Joint Distribution

$$X = \{x_1, \dots, x_N\}$$

$$Z = \{z_1, \dots, z_N\}$$

$$P(X, Z | \theta) = p(z_1 | \pi) \left[\prod_{n=2}^N p(z_n | z_{n-1}, A) \right] \prod_{m=1}^N p(x_m | z_m, \phi)$$

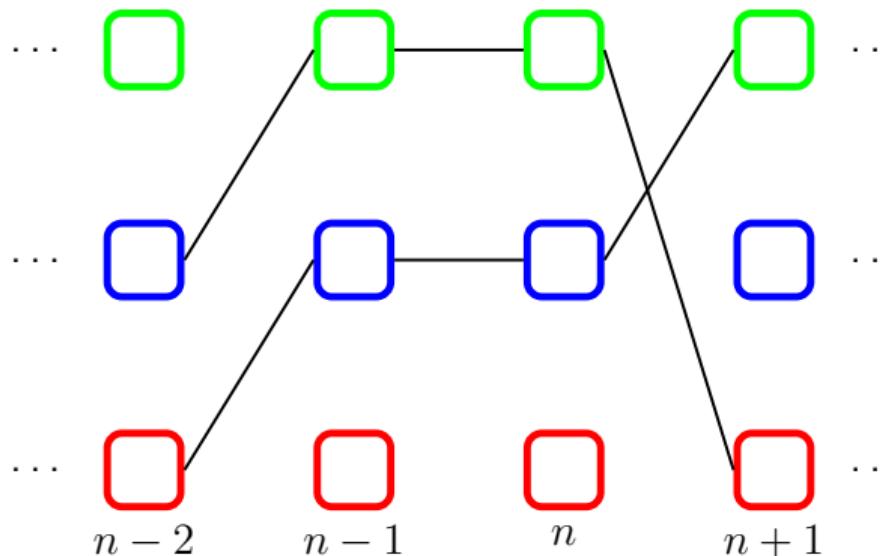


HMM Inference: Likelihood Function

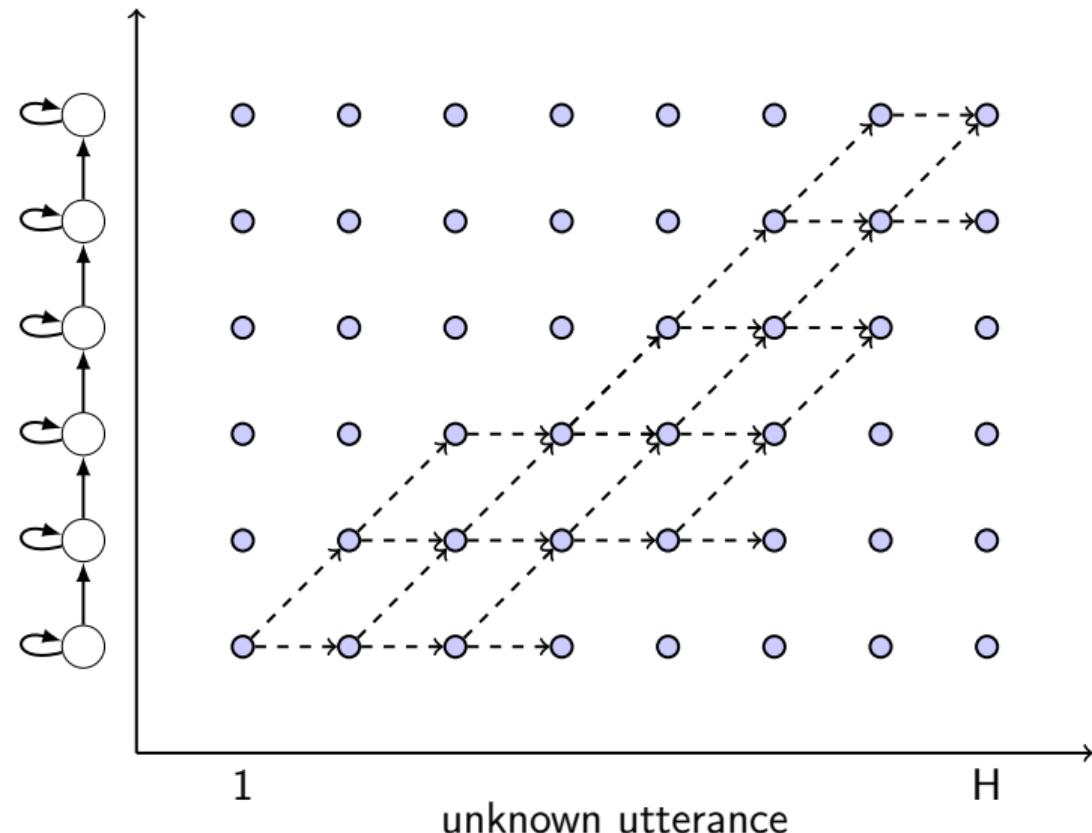
marginalise joint distribution over Z :

$$P(X|\theta) = \sum_Z p(X, Z|\theta)$$

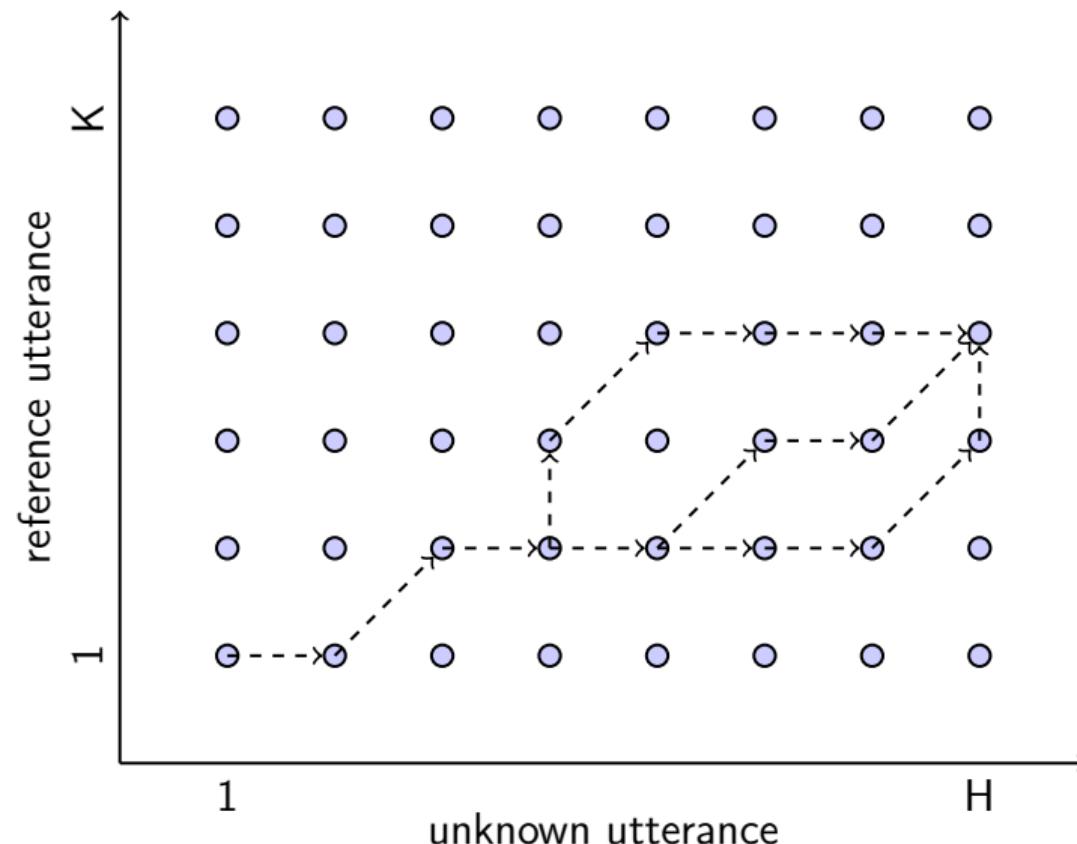
Problem: there are K^N possible sequences for Z



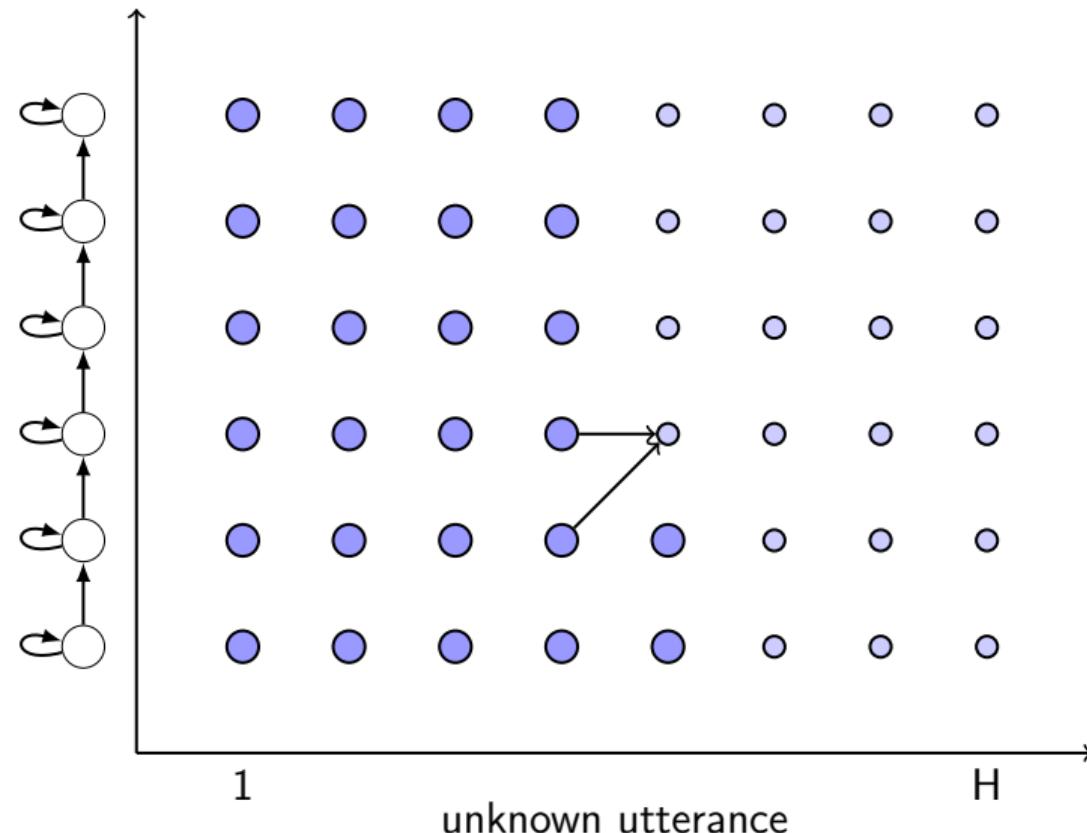
HMM Likelihood



Very Similar to Template Matching



Same Solution: Dynamic Programming



Solution: Forward algorithm

Instead of AccD[h,k] (Template Matching)

$$\alpha_n(j) \equiv p(x_1, \dots, x_n, z_n = s_j | \theta)$$

At the end, instead of AccD[H,K]:

$$P(X|\theta) = \sum_{i=1}^M \alpha_N(i)$$

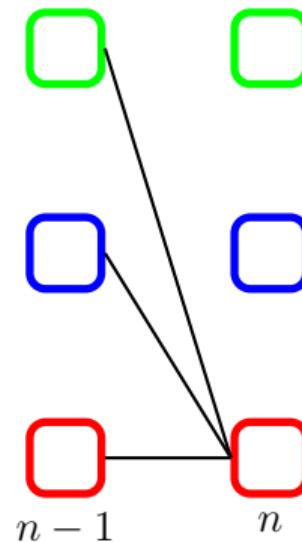
Forward Probability

Initialization:

$$\alpha_1(j) = \pi_j \phi_j(x_1)$$

Recursion:

$$\alpha_n(j) = \left[\sum_{i=1}^M \alpha_{n-1}(i) a_{ij} \right] \phi_j(x_n)$$



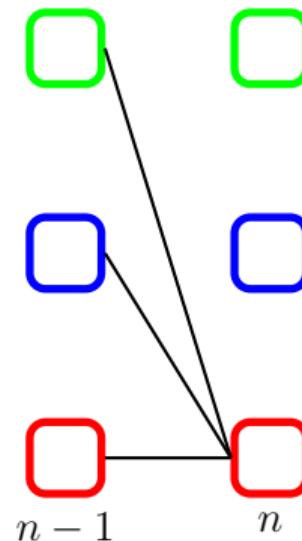
Forward Probability

Initialization:

$$\alpha_1(j) = \pi_j \phi_j(x_1)$$

Recursion:

$$\alpha_n(j) = \left[\sum_{i=1}^M \alpha_{n-1}(i) a_{ij} \right] \phi_j(x_n)$$



equivalent to **sum-product** in Bayesian Networks

Backward probability

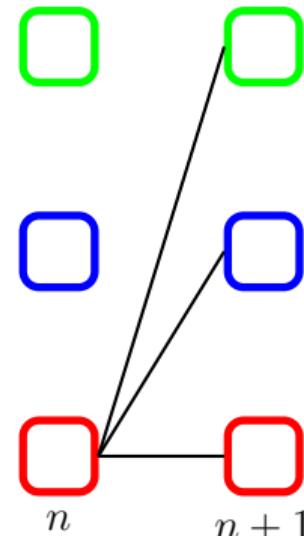
$$\beta_n(i) \equiv p(x_{n+1}, \dots, x_N | z_n = s_i)$$

Initialization:

$$\beta_N(i) \equiv p(?) | z_n = s_i) \equiv 1$$

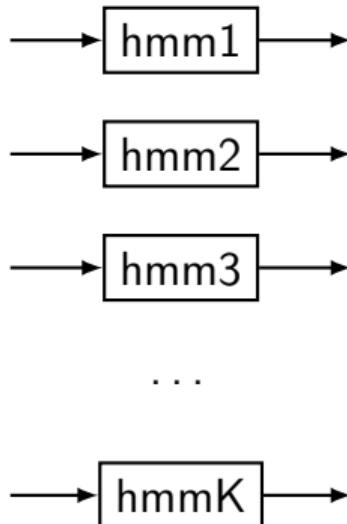
Recursion:

$$\beta_n(i) = \left[\sum_{j=1}^M a_{ij} \phi_j(x_{n+1}) \beta_{n+1}(j) \right]$$



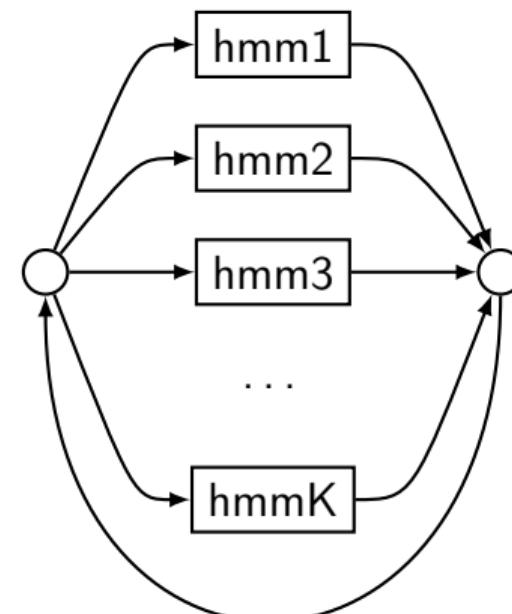
Find best sequence of states: why?

Isolated Words



(Likelihood)

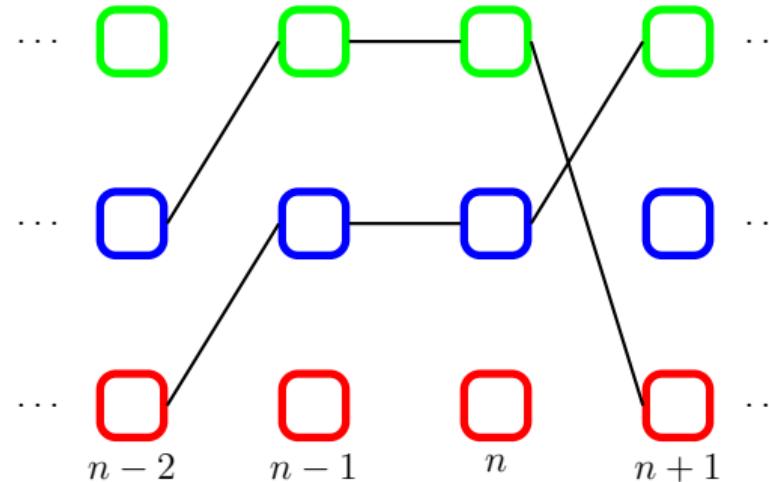
Continuous Speech



(best path)

Find best sequence of states: how?

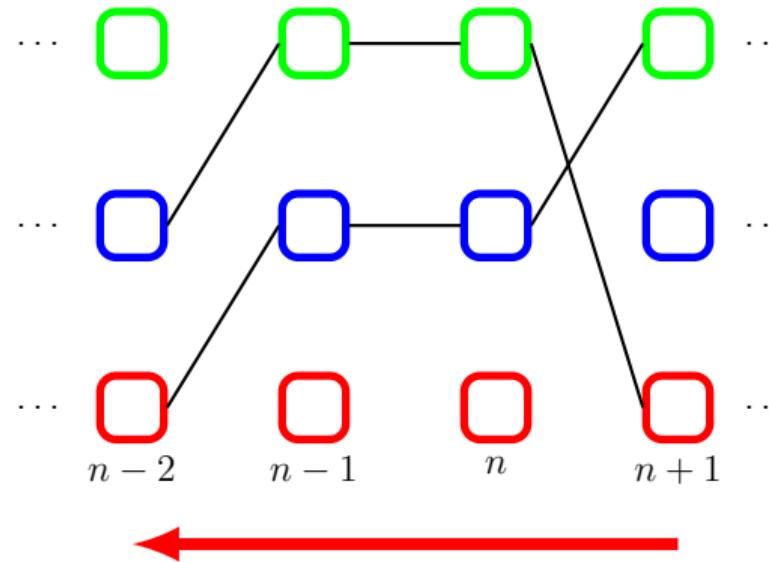
- Viterbi algorithm⁷
- equivalent to **max-sum** in Bayesian Networks



⁷A. J. Viterbi. "Error Bounds for Convolutional Codes and an Asymptotically optimum decoding algorithm". In: *IEEE Trans. Inf. Theory* IT-13 (Apr. 1967), pp. 260–269.

Find best sequence of states: how?

- Viterbi algorithm⁷
- equivalent to **max-sum** in Bayesian Networks



⁷A. J. Viterbi. "Error Bounds for Convolutional Codes and an Asymptotically optimum decoding algorithm". In: *IEEE Trans. Inf. Theory* IT-13 (Apr. 1967), pp. 260–269.

Summary: update rules

Forward algorithm (sum-product):

$$\alpha_n(j) = \left[\sum_{i=1}^M \alpha_{n-1}(i) a_{ij} \right] \phi_j(x_n)$$

Viterbi algorithm (max-sum):

$$V_n(j) = \max_{i=1}^M [V_{n-1}(i) a_{ij}] \phi_j(x_n)$$

$$B_n(j) = \arg \max_{i=1}^M [V_{n-1}(i) a_{ij}]$$

Probabilistic Modelling of Sequences: Learning

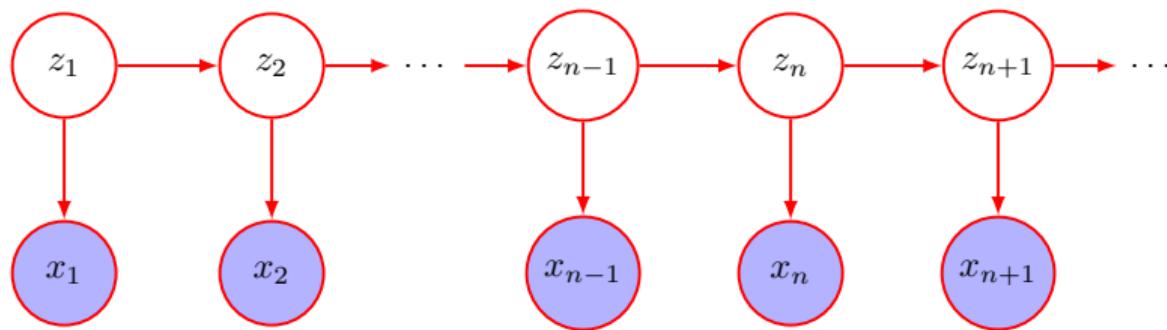
TTT4185 Machine Learning for Signal Processing

Giampiero Salvi

Department of Electronic Systems
NTNU

HT2020

HMM Inference: Learning

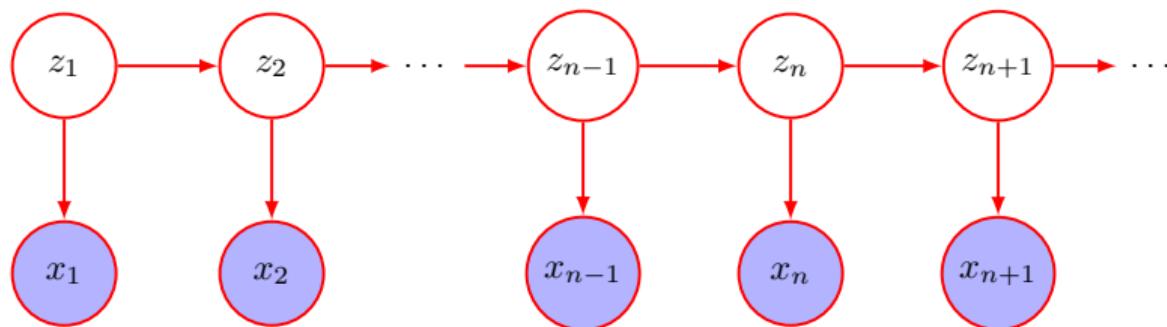


- Given observations X update model parameters

$$\theta = \{\pi, A, \phi\}$$

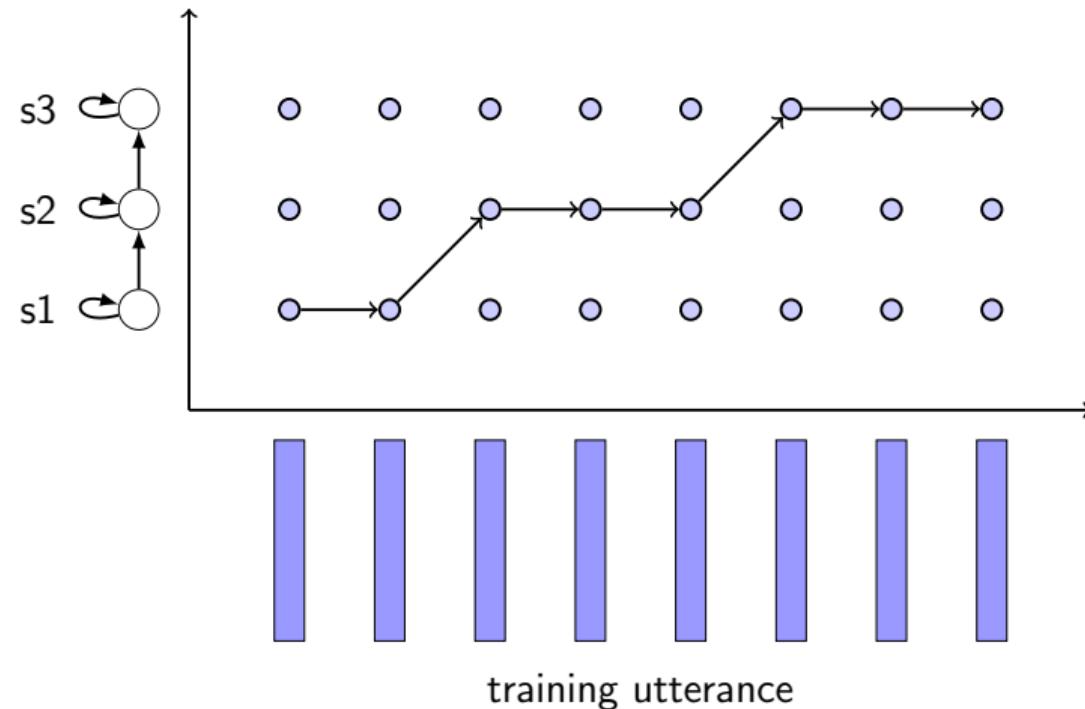
- to maximise either:
 - model fit to data (e.g. likelihood, posterior)
 - classification performance (discriminative training)

HMM Inference: Learning

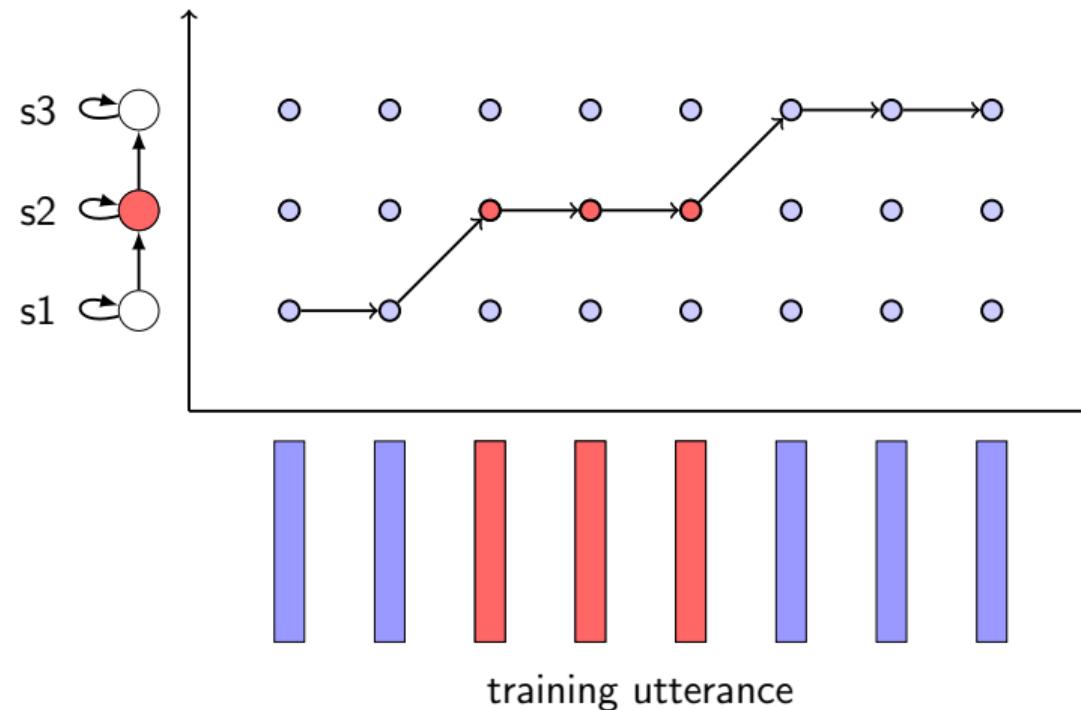


- problem: incomplete data, state sequence Z
- there is no closed-form solution
- only iterative procedures: given θ^{old} how to estimate θ^{new}

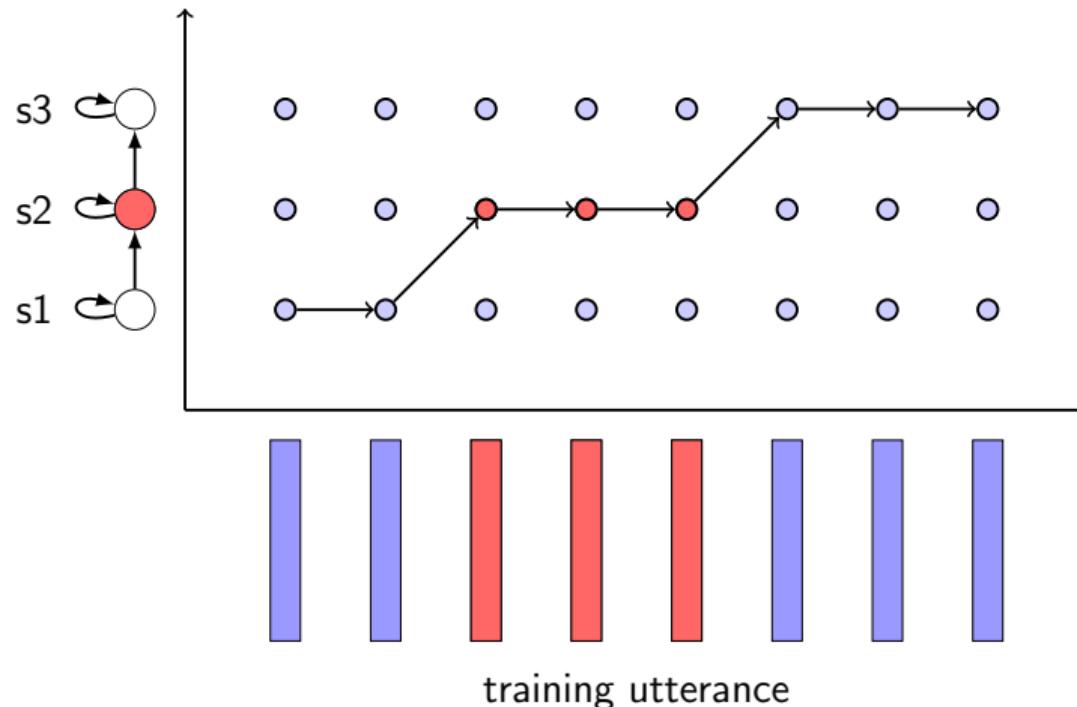
Viterbi training (simple approach)



Viterbi training (simple approach)



Viterbi training (simple approach)



problem: sensitive to misalignments
but still used for ANN/DNN training

HMM Inference: Learning

Latent variables → Expectation Maximisation

- locally maximise the likelihood of the complete data X, Z
- efficient solution with **forward-backward** or **Baum-Welch** algorithm¹
- general idea: sum over all possible paths weighted by posterior probability of the path
- also: every observation vector contributes to all parameter updates

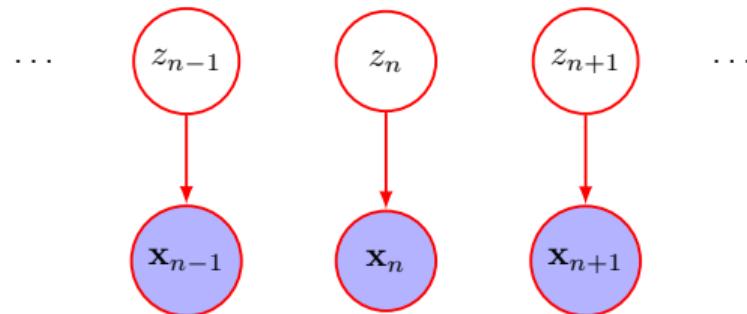
¹L. E. Baum, T. Petrie, G. Soules, and N. Weiss. "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains". In: *Ann. Math. Statist.* 41.1 (1970), pp. 164–171.

Expectation Maximization for Mixture Models

$$P(\mathbf{x}|\theta) = \sum_{k=1}^K \pi_k P(\mathbf{x}|\theta_k),$$

$$\theta = \{\pi_1, \dots, \pi_k, \theta_1, \dots, \theta_K\},$$

$$\sum_{k=1}^K \pi_k = 1$$

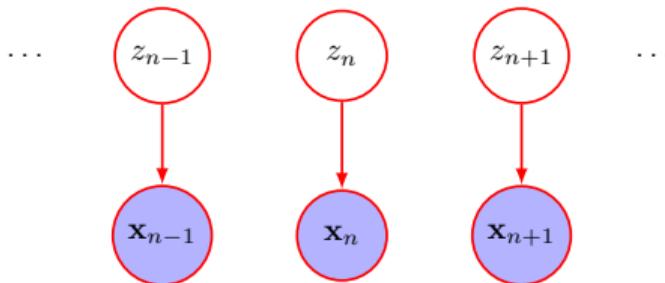


- augment the data with the latent variables:
 $z_i \in \{1, \dots, K\}$ assignment of each data point x_i to a component of the mixture
- interpret the mixture as marginal of the joint

$$P(\mathbf{x}|\theta) = \sum_z P(\mathbf{x}, \mathbf{z}|\theta)$$

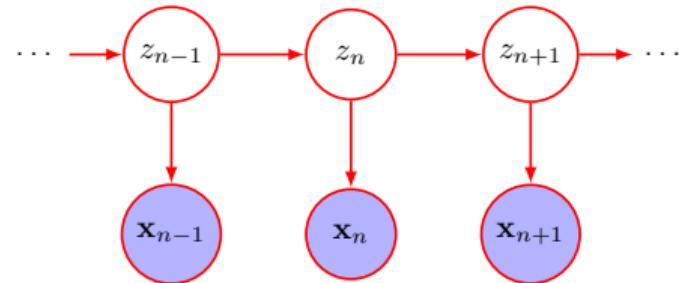
Mixture Models vs Hidden Markov Models

Mixture Model



EM Algorithm

Hidden Markov Model



Baum-Welch Algorithm (EM)

Expectation Maximization: Idea

Ideally we would like to maximize:

$$\log p(X|\theta) = \log \left\{ \sum_Z p(X, Z|\theta) \right\}$$

with $X = \{x_1, \dots, x_N\}$, $Z = \{z_1, \dots, z_N\}$

... but log of sum hard to optimize

Instead optimize likelihood of complete data:

$$\log p(X, Z|\theta)$$

Z not known, but we can compute posterior given current model $p(Z|X, \theta^{\text{old}})$

Optimize the expected value of the likelihood:

$$\mathcal{Q}(\theta, \theta^{\text{old}}) = \sum_Z p(Z|X, \theta^{\text{old}}) \log p(X, Z|\theta)$$

Expectation Maximization in Practice

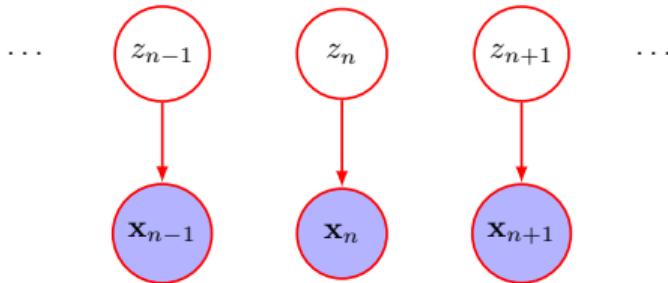
- ➊ Initialization: choose initial value of θ^{old}
- ➋ Expectation step: evaluate posterior $p(Z|X, \theta^{\text{old}})$
- ➌ Maximization step: evaluate θ^{new} with:

$$\begin{aligned}\theta^{\text{new}} &= \arg \max_{\theta} Q(\theta, \theta^{\text{old}}) \\ &= \arg \max_{\theta} \sum_Z p(Z|X, \theta^{\text{old}}) \log p(X, Z|\theta)\end{aligned}$$

- ➍ Check for convergence, otherwise $\theta^{\text{old}} \leftarrow \theta^{\text{new}}$ and go to step 2.

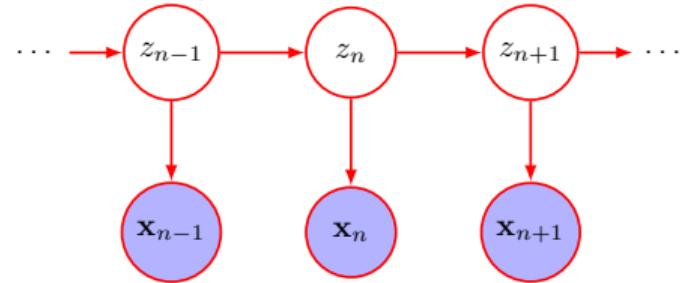
Mixture Models vs Hidden Markov Models

Mixture Model



EM Algorithm

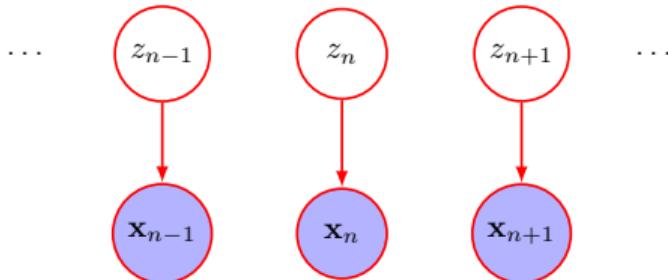
Hidden Markov Model



Baum-Welch Algorithm (EM)

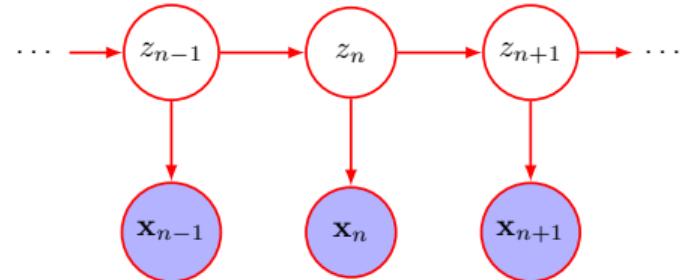
Mixture Models vs Hidden Markov Models

Mixture Model



EM Algorithm

Hidden Markov Model



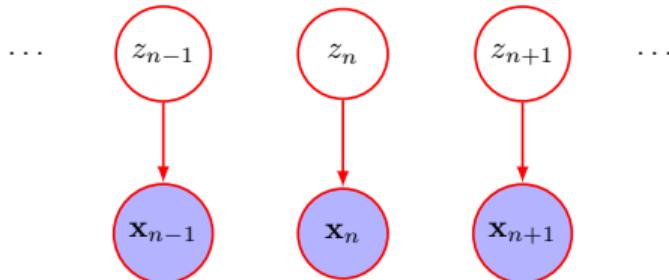
Baum-Welch Algorithm (EM)

- ① Posterior of latent variable z_n depends on full sequence X

$$\gamma_n(k) = P(z_n = k \mid X, \theta^{(t)}) \neq P(z_n = k \mid x_n, \theta^{(t)})$$

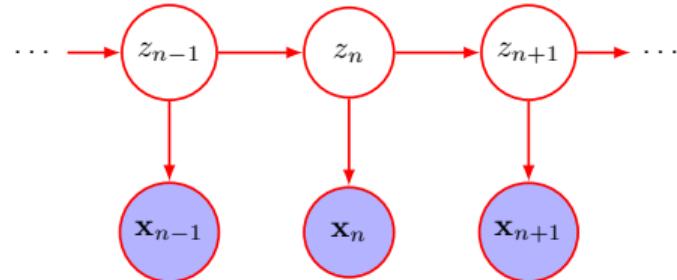
Mixture Models vs Hidden Markov Models

Mixture Model



EM Algorithm

Hidden Markov Model



Baum-Welch Algorithm (EM)

- ① Posterior of latent variable z_n depends on full sequence X

$$\gamma_n(k) = P(z_n = k \mid X, \theta^{(t)}) \neq P(z_n = k \mid x_n, \theta^{(t)})$$

- ② We also need the posterior of two subsequent latent variables

$$\xi_n(i, j) = P(z_{n-1} = s_i, z_n = s_j \mid X, \theta)$$

Baum-Welch E-Step

Estimate posterior $P(Z|X, \theta^{\text{old}})$ or, at least the sufficient statistics given:

- current model parameters
 - full sequence of observations X
- ① Posterior of latent variable z_n

$$\gamma_n(i) = P(z_n = s_i | X, \theta)$$

- ② Posterior of two subsequent latent variables

$$\xi_n(i, j) = P(z_{n-1} = s_i, z_n = s_j | X, \theta)$$

Baum-Welch E-Step

Estimate posterior $P(Z|X, \theta^{\text{old}})$ or, at least the sufficient statistics given:

- current model parameters
 - full sequence of observations X
- ① Posterior of latent variable z_n

$$\gamma_n(i) = P(z_n = s_i | X, \theta)$$

- ② Posterior of two subsequent latent variables

$$\xi_n(i, j) = P(z_{n-1} = s_i, z_n = s_j | X, \theta)$$

Note: Huang, Acero and Hon call this $\gamma_n(i, j)$

Note: posteriors conditioned to full sequence X not only x_n

Baum-Welch M-Step

Weighted Maximum Likelihood estimates:

Emission probabilities:

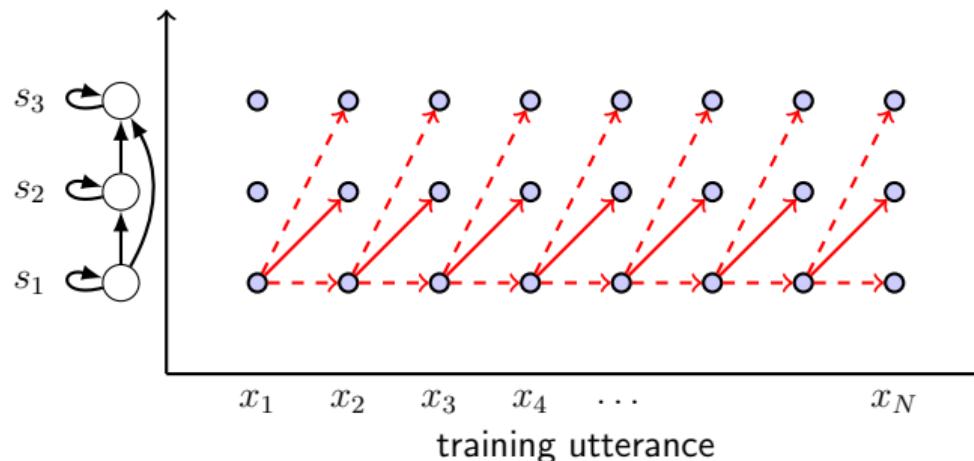
Same as mixture model given $\gamma_n(i) = P(z_n = s_j | X, \theta)$

Transition probabilities

$$\begin{aligned} a_{ij}^{\text{new}} &= \frac{E[s_i \rightarrow s_j | X, \theta^{\text{old}}]}{E[s_i \rightarrow s_{\text{any}} | X, \theta^{\text{old}}]} = \frac{\sum_{n=2}^N \xi_n(i, j)}{\sum_{n=2}^N \sum_{k=1}^M \xi_n(i, k)} \\ &\quad \text{or, equivalently,} \\ &= \frac{E[s_i \rightarrow s_j | X, \theta^{\text{old}}]}{E[s_i | X, \theta^{\text{old}}]} = \frac{\sum_{n=2}^N \xi_n(i, j)}{\sum_{n=1}^{N-1} \gamma_n(i)} \end{aligned}$$

Expectations are over the posteriors $P(Z|X, \theta^{\text{old}})$.

Example: Transition Probability



$$\begin{aligned} a_{12}^{\text{new}} &= \frac{E[s_1 \rightarrow s_2 | X, \theta^{\text{old}}]}{E[s_1 \rightarrow s_{\text{any}} | X, \theta^{\text{old}}]} = \frac{\sum_{n=2}^N \xi_n(1, 2)}{\sum_{n=2}^N \sum_{k=1}^3 \xi_n(1, k)} \\ &= \frac{E[s_1 \rightarrow s_2 | X, \theta^{\text{old}}]}{E[s_1 | X, \theta^{\text{old}}]} = \frac{\sum_{n=2}^N \xi_n(1, 2)}{\sum_{n=1}^{N-1} \gamma_n(1)} \end{aligned}$$

Example: Transition Probability

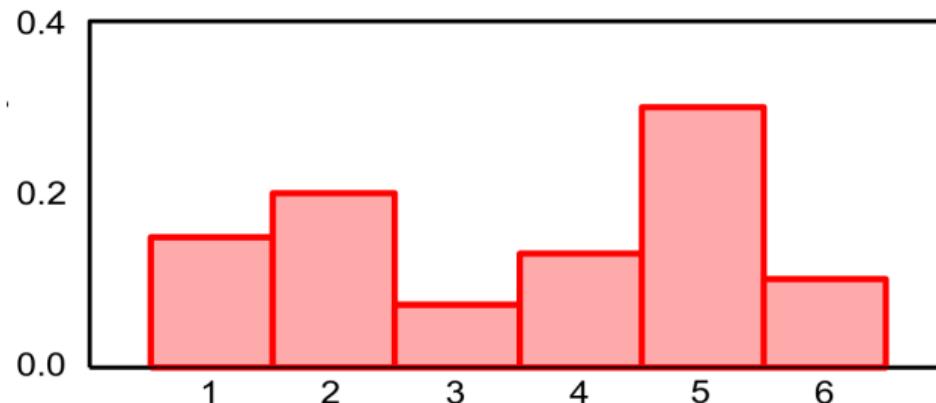
- $\sum_{n=2}^N \xi_n(i, j)$ is the expected number of transitions between state s_i and s_j (given X and θ^{old})
- $\sum_{n=1}^{N-1} \gamma_n(i)$ is the expected number of times we are in state s_i (given X and θ^{old})
- we never take a hard decision on when the transition happened

Emission probabilities

- Discrete HMMs (DHMMs)
 - vector quantisation
- Continuous HMMs
 - Single Gaussian $\phi_j(x_n) = N(x_n | \mu_j, \Sigma_j)$
 - Gaussian Mixture
- Semi-continuous HMMs (SCHMMs)

Discrete HMMs

- quantise feature vectors
- observation: sequence of discrete symbols
- $\phi_j(x_n)$ simple discrete probability distribution
- problem: quantisation error



Discrete HMMs: learn $\phi_j(x_n)$

Remember that

$$\gamma_n(j) = P(z_n = s_j | X, \theta)$$

are the posteriors of the latent variable

Update rule:

$$\phi_j(x_n = k) = p(x_n = k | z_n = s_j) = \frac{E[x_n = k, z_n = s_j]}{E[z_n = s_j]} = \frac{\sum_{\textcolor{red}{n:(x_n=k)}} \gamma_n(j)}{\sum_{\textcolor{red}{n=1}}^N \gamma_n(j)}$$

HMMs with Gaussian Emission Probability

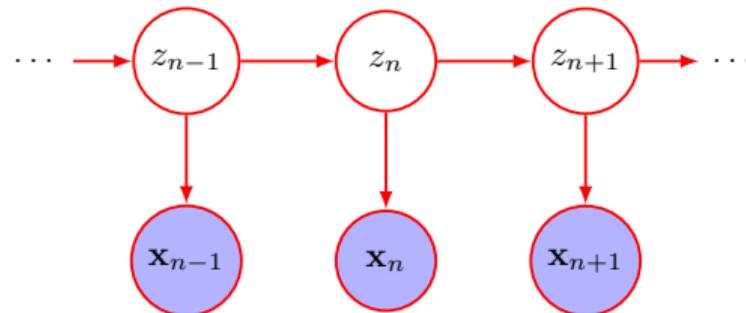
$$\phi_j(x_n) = N(x_n | \mu_j, \Sigma_j)$$

Update rules:

$$\mu_j = \frac{\sum_{n=1}^N \gamma_n(j) x_n}{\sum_{n=1}^N \gamma_n(j)}$$

$$\Sigma_j = \frac{\sum_{n=1}^N \gamma_n(j) (x_n - \mu_j) (x_n - \mu_j)^T}{\sum_{n=1}^N \gamma_n(j)}$$

Calculate sufficient statistics



$$\gamma_n(i) = P(z_n = s_i | X, \theta)$$

$$\xi_n(i, j) = P(z_{n-1} = s_i, z_n = s_j | X, \theta)$$

We can do this with the help of the forward and backward variables:

$$\alpha_n(i) = P(x_1, \dots, x_n, z_n = s_i | \theta)$$

$$\beta_n(i) = P(x_{n+1}, \dots, x_N | z_n = s_i, \theta)$$

Calculate γ (forward-backward)

$$\gamma_n(i) = P(z_n = s_i | X, \theta)$$

Calculate γ (forward-backward)

$$\begin{aligned}\gamma_n(i) &= P(z_n = s_i | X, \theta) \\ &= \frac{p(X, z_n = s_i | \theta)}{p(X | \theta)}\end{aligned}$$

Calculate γ (forward-backward)

$$\begin{aligned}\gamma_n(i) &= P(z_n = s_i | X, \theta) \\ &= \frac{p(X, z_n = s_j | \theta)}{p(X | \theta)} \\ &= \frac{p(x_1, \dots, x_n, x_{n+1}, \dots, x_N, z_n = s_j | \theta)}{p(X | \theta)}\end{aligned}$$

Calculate γ (forward-backward)

$$\begin{aligned}\gamma_n(i) &= P(z_n = s_i | X, \theta) \\ &= \frac{p(X, z_n = s_j | \theta)}{p(X | \theta)} \\ &= \frac{p(x_1, \dots, x_n, x_{n+1}, \dots, x_N, z_n = s_j | \theta)}{p(X | \theta)} \\ &= \frac{p(x_1, \dots, x_n, x_{n+1}, \dots, x_N | z_n = s_j, \theta) P(z_n = s_j | \theta)}{p(X | \theta)}\end{aligned}$$

Calculate γ (forward-backward)

$$\begin{aligned}\gamma_n(i) &= P(z_n = s_i | X, \theta) \\ &= \frac{p(X, z_n = s_i | \theta)}{p(X | \theta)} \\ &= \frac{p(x_1, \dots, x_n, x_{n+1}, \dots, x_N, z_n = s_i | \theta)}{p(X | \theta)} \\ &= \frac{p(x_1, \dots, x_n, x_{n+1}, \dots, x_N | z_n = s_i, \theta) P(z_n = s_i | \theta)}{p(X | \theta)} \\ &= \frac{p(x_1, \dots, x_n | z_n = s_i, \theta) p(x_{n+1}, \dots, x_N | z_n = s_i, \theta) P(z_n = s_i | \theta)}{p(X | \theta)}\end{aligned}$$

Calculate γ (forward-backward)

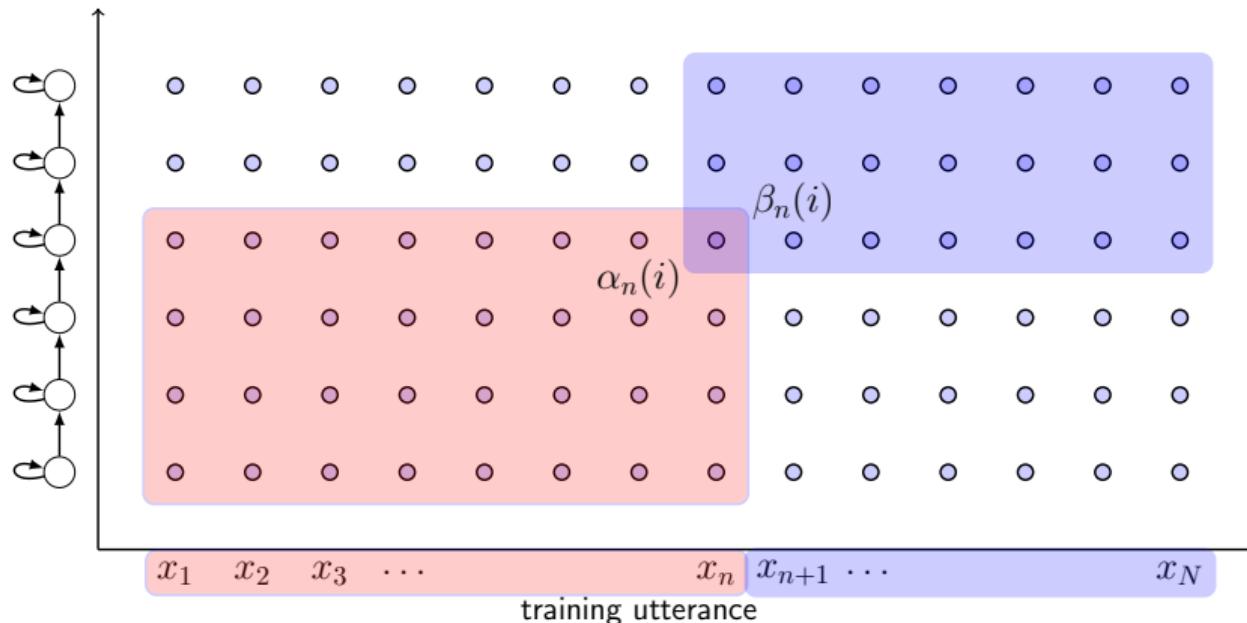
$$\begin{aligned}\gamma_n(i) &= P(z_n = s_i | X, \theta) \\ &= \frac{p(X, z_n = s_j | \theta)}{p(X | \theta)} \\ &= \frac{p(x_1, \dots, x_n, x_{n+1}, \dots, x_N, z_n = s_j | \theta)}{p(X | \theta)} \\ &= \frac{p(x_1, \dots, x_n, x_{n+1}, \dots, x_N | z_n = s_j, \theta) P(z_n = s_j | \theta)}{p(X | \theta)} \\ &= \frac{p(x_1, \dots, x_n | z_n = s_j, \theta) p(x_{n+1}, \dots, x_N | z_n = s_j, \theta) P(z_n = s_j | \theta)}{p(X | \theta)} \\ &= \frac{p(x_1, \dots, x_n, z_n = s_j | \theta) p(x_{n+1}, \dots, x_N | z_n = s_j, \theta)}{p(X | \theta)}\end{aligned}$$

Calculate γ (forward-backward)

$$\begin{aligned}\gamma_n(i) &= P(z_n = s_i | X, \theta) \\ &= \frac{p(X, z_n = s_j | \theta)}{p(X | \theta)} \\ &= \frac{p(x_1, \dots, x_n, x_{n+1}, \dots, x_N, z_n = s_j | \theta)}{p(X | \theta)} \\ &= \frac{p(x_1, \dots, x_n, x_{n+1}, \dots, x_N | z_n = s_j, \theta) P(z_n = s_j | \theta)}{p(X | \theta)} \\ &= \frac{p(x_1, \dots, x_n | z_n = s_j, \theta) p(x_{n+1}, \dots, x_N | z_n = s_j, \theta) P(z_n = s_j | \theta)}{p(X | \theta)} \\ &= \frac{p(x_1, \dots, x_n, z_n = s_j | \theta) p(x_{n+1}, \dots, x_N | z_n = s_j, \theta)}{p(X | \theta)} \\ &= \frac{\alpha_n(i) \beta_n(i)}{\sum_i \alpha_N(i)}\end{aligned}$$

Calculate γ : Illustration

$$\gamma_n(i) = P(z_n = s_i | X, \theta) = \frac{\alpha_n(i)\beta_n(i)}{\sum_i \alpha_N(i)}$$



Calculate ξ (forward-backward)

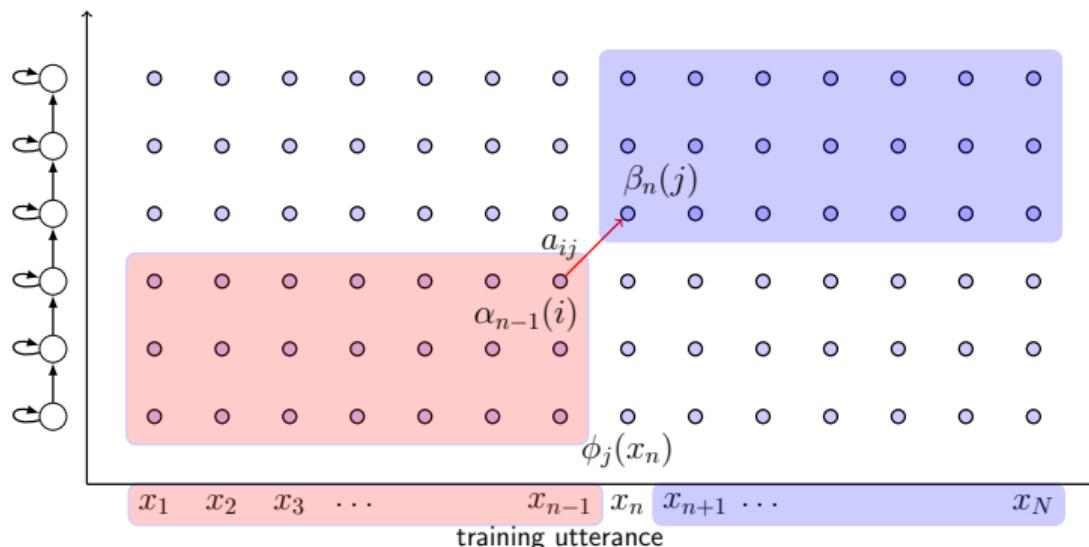
$$\xi_n(i, j) = P(z_n = s_j, z_{n-1} = s_i | X, \theta)$$

Calculate ξ (forward-backward)

$$\begin{aligned}\xi_n(i, j) &= P(z_n = s_j, z_{n-1} = s_i | X, \theta) \\ &= \frac{P(z_n = s_j, z_{n-1} = s_i, X | \theta)}{P(X | \theta)}\end{aligned}$$

Calculate ξ (forward-backward)

$$\begin{aligned}\xi_n(i, j) &= P(z_n = s_j, z_{n-1} = s_i | X, \theta) \\ &= \frac{P(z_n = s_j, z_{n-1} = s_i, X | \theta)}{P(X | \theta)} \\ \dots &= \frac{\alpha_{n-1}(i) \ a_{ij} \ \phi_j(x_n) \ \beta_n(j)}{\sum_{k=1}^M \alpha_N(k)}\end{aligned}$$



Baum-Welch: Properties

instance of Expectation Maximisation:

- iterative procedure
- guaranteed to convert to local maximum of the likelihood $P(X|\theta^{\text{new}})$
- sensitive to initialisation
- update formulae for emission probability model $\phi_j(x_n)$ same as for mixture models (with new version of posteriors)

Numerical Problems

Product of many probabilities.

Solution: work in log domain

linear domain

$$\alpha_1(j) = \pi_j \phi_j(x_1)$$

$$\alpha_n(j) = \phi_j(x_n) \sum_{i=1}^M \alpha_{n-1}(i) a_{ij}$$

$$\beta_N(i) = 1$$

$$\beta_n(i) = \sum_{j=1}^M a_{ij} \phi_j(x_{n+1}) \beta_{n+1}(j)$$

log domain

$$\alpha'_1(j) = \pi'_j + \phi'_j(x_1)$$

$$\alpha'_n(j) = \log \sum_{i=1}^M e^{(\alpha'_{n-1}(i) + a'_{ij})} + \phi'_j(x_n)$$

$$\beta'_N(i) = 0$$

$$\beta'_n(i) = \log \sum_{j=1}^M e^{(a'_{ij} + \phi'_j(x_{n+1}) + \beta'_{n+1}(j))}$$

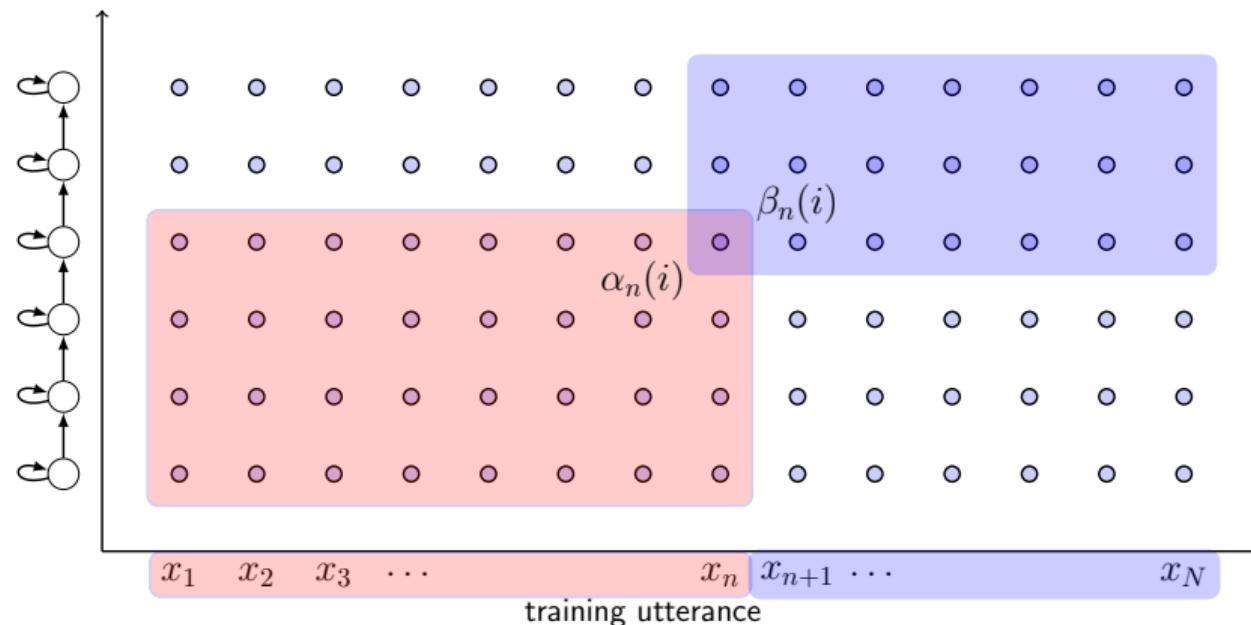
Train on several utterances

Set of utterances X^1, X^2, \dots, X^U

- cannot concatenate them: need to calculate α , β , γ and ξ each time

Each utterance corresponds to several models

- reuse model states (sentence \rightarrow words \rightarrow phonemes)



Concatenating HMMs

Utterance to words:

sil one zero one three sil

Words to phones

sil w ah n sp z iy r ow sp w ah n sp th r iy sp sil

Phones to states

sil0 sil1 sil2 w0 w1 w2 ah0 ah1 ah2 n0 n1 n2 sp0 z0 z1 z2 iy0 iy1 iy2 r0 r1 r2
ow0 ow1 ow2 sp0 w0 w1 w2 ah0 ah1 ah2 n0 n1 n2 sp0 th0 th1 th2 r0 r1 r2 iy0
iy1 iy2 sp0 sil0 sil1 sil2

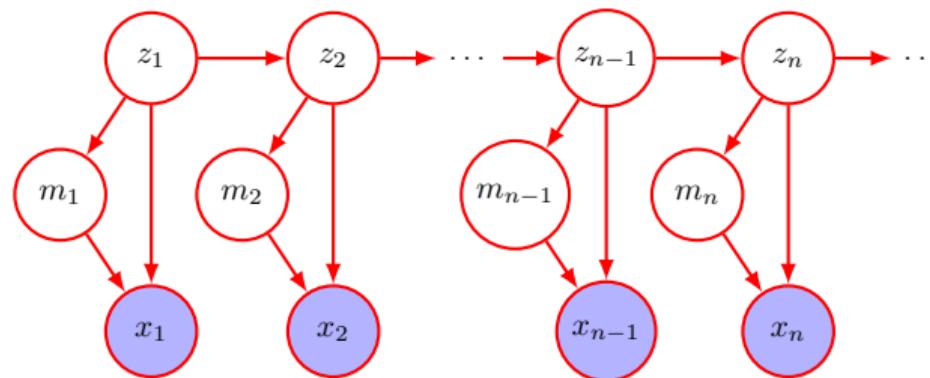
HMMs with Mixture Emission Probability

Often the Emission probability is modelled as a Mixture of Gaussians

$$\phi_j(x_n) = \sum_{k=1}^K w_{jk} N(x_n | \mu_{jk}, \Sigma_{jk})$$

$$\sum_{k=1}^M w_{jk} = 1$$

HMMs with Mixture Emission Probability



Emission:

$$\begin{aligned} p(x_n | z_n, m_n) &= \mathcal{N}(x_n; \mu_{z_n, m_n}, \Sigma_{z_n, m_n}) \\ p(m_n | z_n) &= W(m_n, z_n) \end{aligned}$$

Semi-Continuous HMMs

- All Gaussian distributions in a pool of pdfs
- each $\phi_j(x_n)$ is a discrete probability distribution over the pool of Gaussians
- similar to quantisation, but probabilistic
- used for sharing parameters

Hybrid HMM+Multi Layer Perceptron

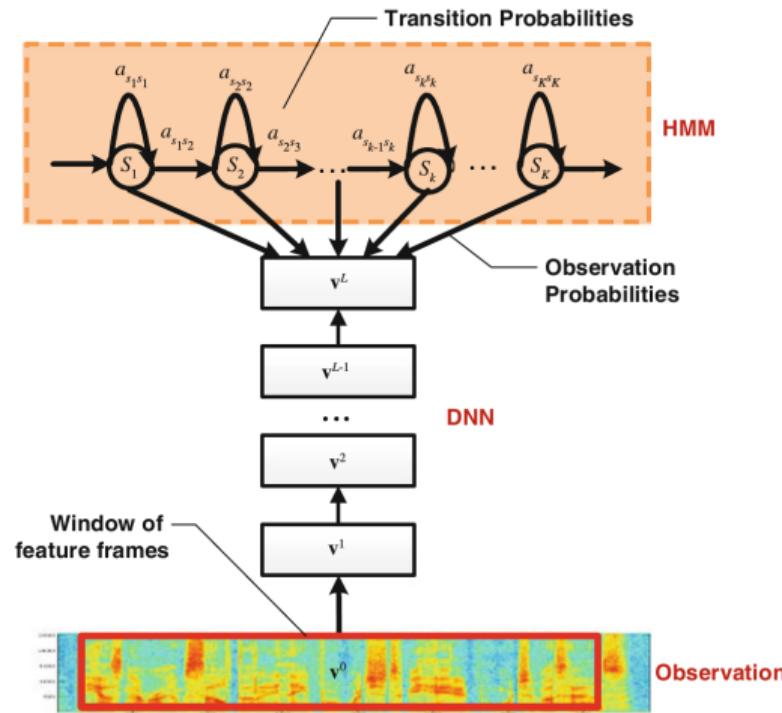


Figure from Yu and Deng

Combining probabilities

- HMMs use likelihoods $P(\text{sound}|\text{state})$
- MLPs and DNNs estimate posteriors $P(\text{state}|\text{sound})$

We can combine with Bayes:

$$P(\text{sound}|\text{state}) = \frac{P(\text{state}|\text{sound})P(\text{sound})}{P(\text{state})}$$

- $P(\text{state})$ can be estimated from the training set
- $P(\text{sound})$ is constant and can be ignored

Use **scaled likelihoods**:

$$\bar{P}(\text{sound}|\text{state}) = \frac{P(\text{state}|\text{sound})}{P(\text{state})}$$

Dimensionality Reduction

TTT4185 Machine Learning for Signal Processing

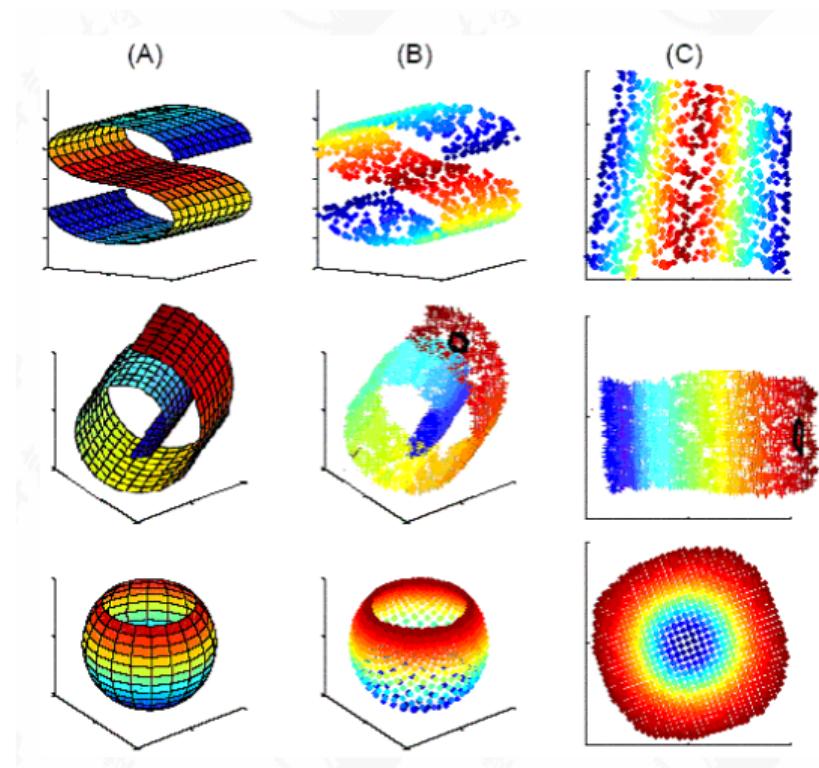
Giampiero Salvi

Department of Electronic Systems
NTNU

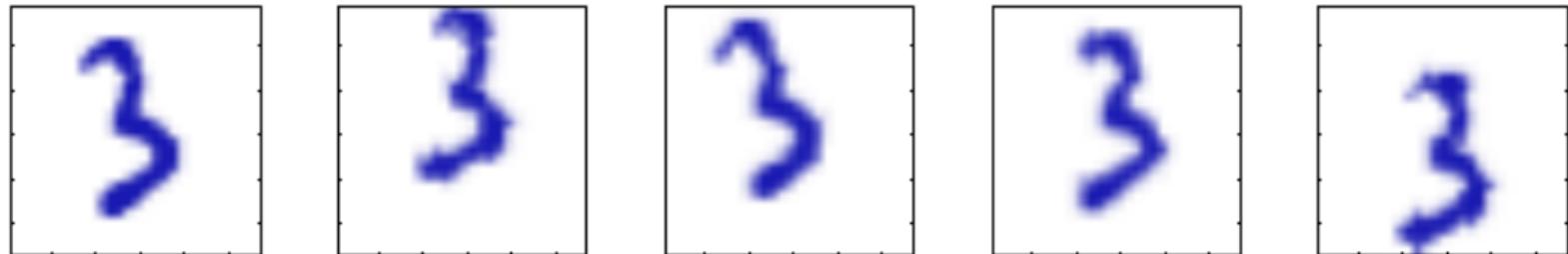
HT2020

Real Dimensionality of Data: Manifolds

- intrinsic dimension of the data
- independent of representation (features)
- manifold: low dimensional topological space embedded in feature space
- can be non-linear (but locally Euclidean)
- if linear they are subspaces

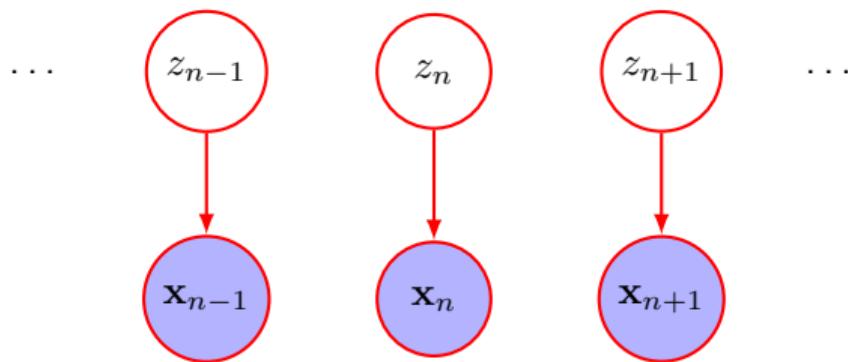


Example: Images



- one single digit example from MNIST
- translations (2 degrees of freedom)
- rotations (1 degree of freedom)
- $100 \times 100 = 10,000$ pixels (dimensions)

Continuous Latent Variables



- discrete $z \rightarrow$ mixture models
- continuous $z \rightarrow$ dimensionality reduction

Different Models

Principal Component Analysis (PCA)

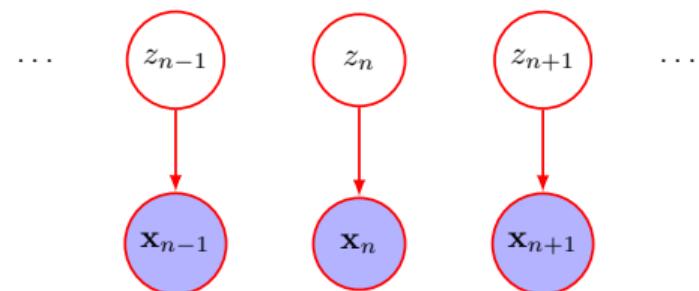
- z and x are Gaussian
- linear Gaussian dependency between x and z

Independent Component Analysis (ICA)

- non Gaussian

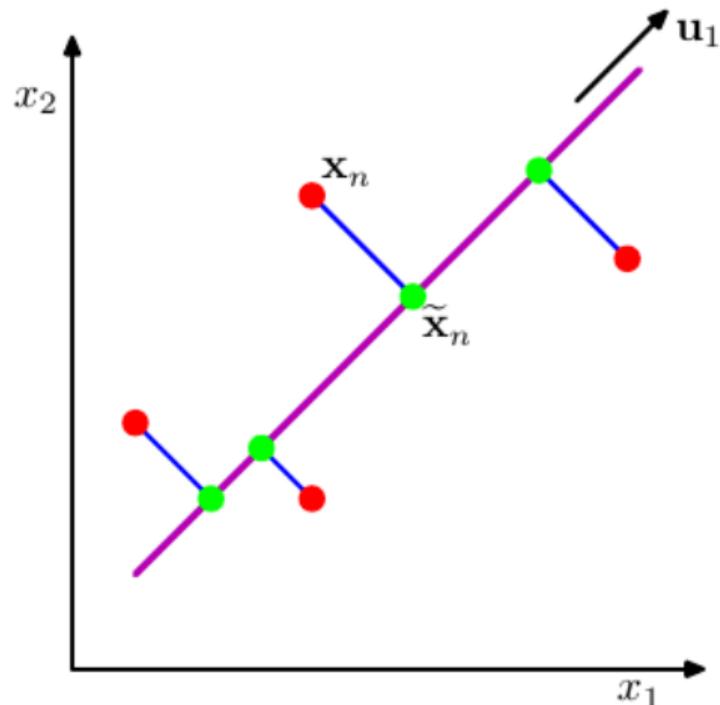
Autoencoders, Isomap, t-SNE, ...

- non-linear



Principal Component Analysis

- data in D dimensions
- sub-space with $M < D$ dimensions
- start with $M = 1$
- unit vector \mathbf{u}_1 ($\mathbf{u}_1^T \mathbf{u}_1 = 1$)
- \mathbf{x}_n is projected onto $\tilde{\mathbf{x}}_n = \mathbf{u}_1^T \mathbf{x}_n$



Principal Component Analysis

- mean: $\mathbf{u}_1^T \bar{\mathbf{x}}_n$, with

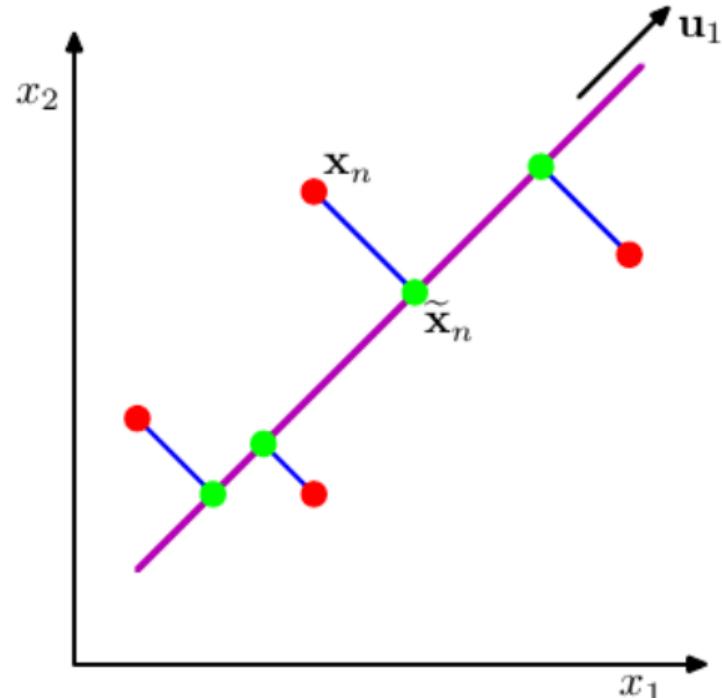
$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n$$

- projected variance:

$$\frac{1}{N} \sum_{n=1}^N \{ \mathbf{u}_1^T \mathbf{x}_n - \mathbf{u}_1^T \bar{\mathbf{x}}_n \}^2 = \mathbf{u}_1^T \mathbf{S} \mathbf{u}_1$$

- with covariance matrix \mathbf{S} :

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T$$



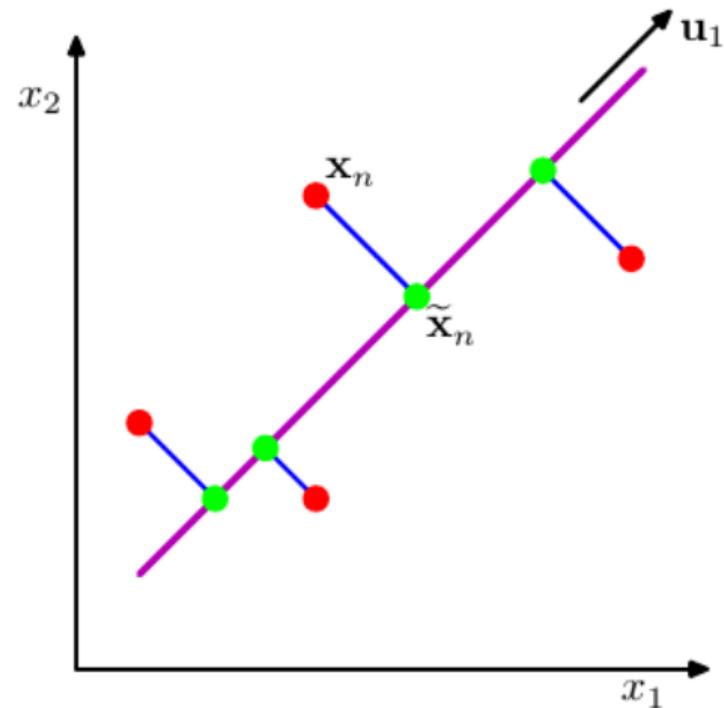
Principal Component Analysis

- maximize projected variance $\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1$
- with constraint that $\mathbf{u}_1^T \mathbf{u}_1 = 1$
- solution:

$$\mathbf{S} \mathbf{u}_1 = \lambda_1 \mathbf{u}_1$$

- \mathbf{u}_1 is eigenvector of \mathbf{S}
- left-multiply by \mathbf{u}_1^T :

$$\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 = \lambda_1 \mathbf{u}_1^T \mathbf{u}_1 = \lambda_1$$



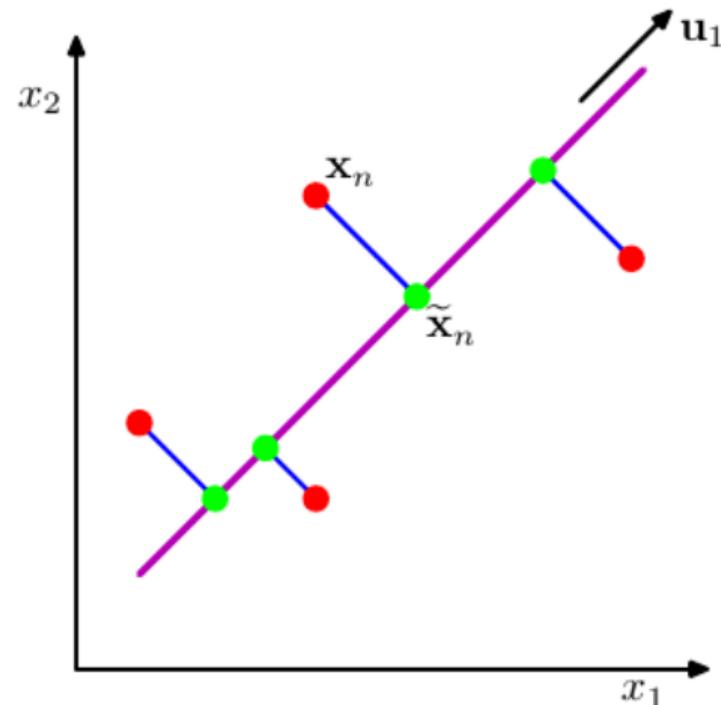
Principal Component Analysis

- find maximum eigenvalue of \mathbf{S}
- the corresponding eigenvector is the principal component
- find M principal components incrementally

$$\mathbf{u}_1, \dots, \mathbf{u}_M,$$

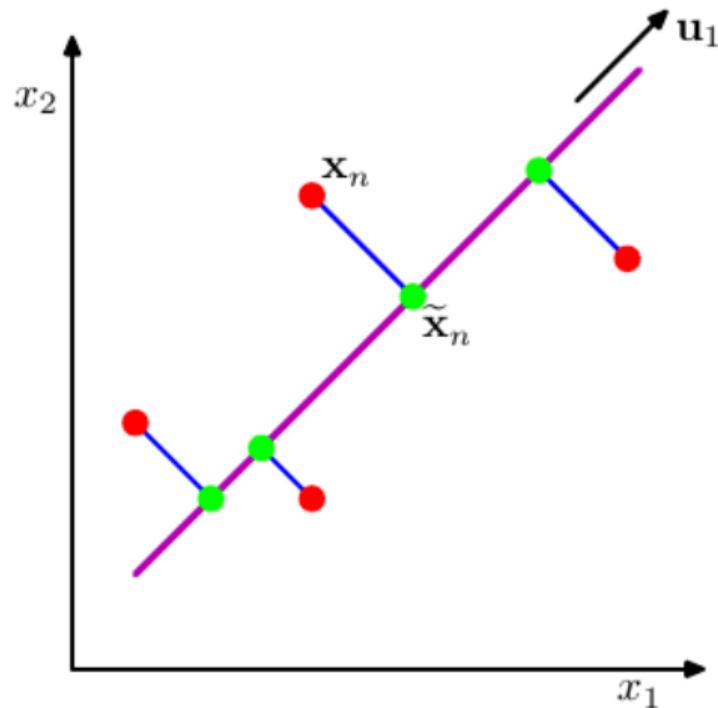
$$\mu_1, \dots, \mu_M$$

- computational cost of eigenvector decomposition $D \times D$ is $O(D^3)$
- power method $O(MD^2)$



Principal Component Analysis

- alternative view: minimize projection square error
- same solution



PCA Applications

Compression:

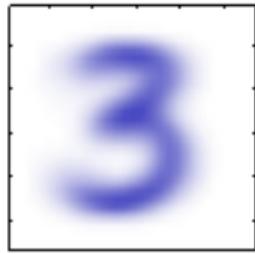
- principal components $\mathbf{u}_1, \dots, \mathbf{u}_M$ ($M \times D$ parameters)
- mixing weights: $\tilde{\mathbf{x}}_n = \sum_{i=1}^M \alpha_{ni} \mathbf{u}_i$ (M parameters)
- if N points, $M \times D + N \times M$ instead of $N \times D$

Visualization:

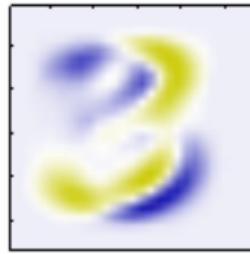
- usually $M = 2$, sometimes $M = 3$
- no big concern on reconstruction error

PCA Applications

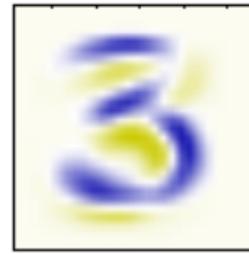
Mean



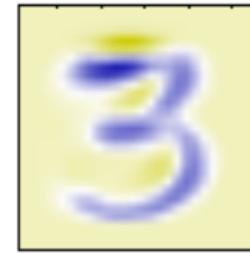
$\lambda_1 = 3.4 \cdot 10^5$



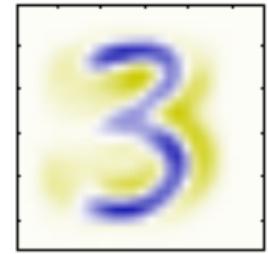
$\lambda_2 = 2.8 \cdot 10^5$



$\lambda_3 = 2.4 \cdot 10^5$

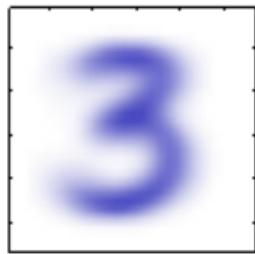


$\lambda_4 = 1.6 \cdot 10^5$

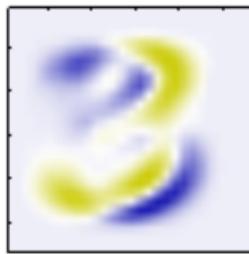


PCA Applications

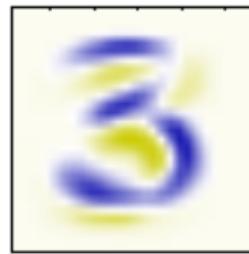
Mean



$\lambda_1 = 3.4 \cdot 10^5$



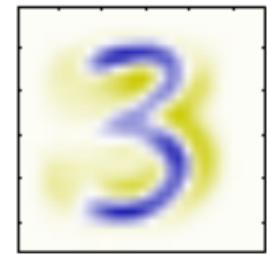
$\lambda_2 = 2.8 \cdot 10^5$



$\lambda_3 = 2.4 \cdot 10^5$

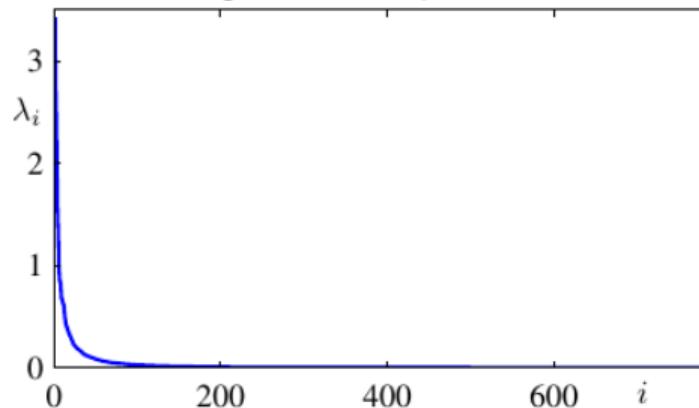


$\lambda_4 = 1.6 \cdot 10^5$



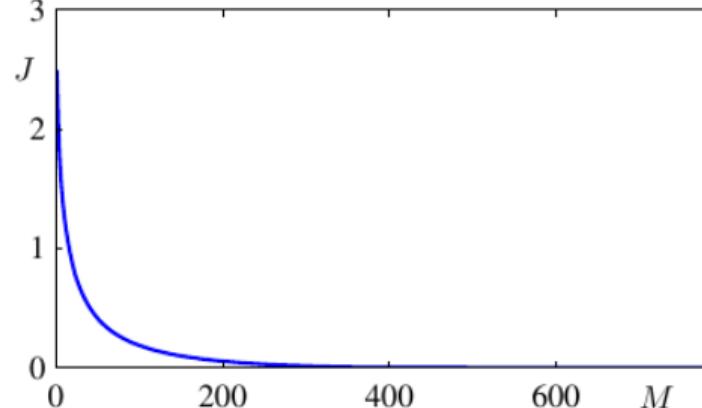
$\times 10^5$

eigenvalue spectrum

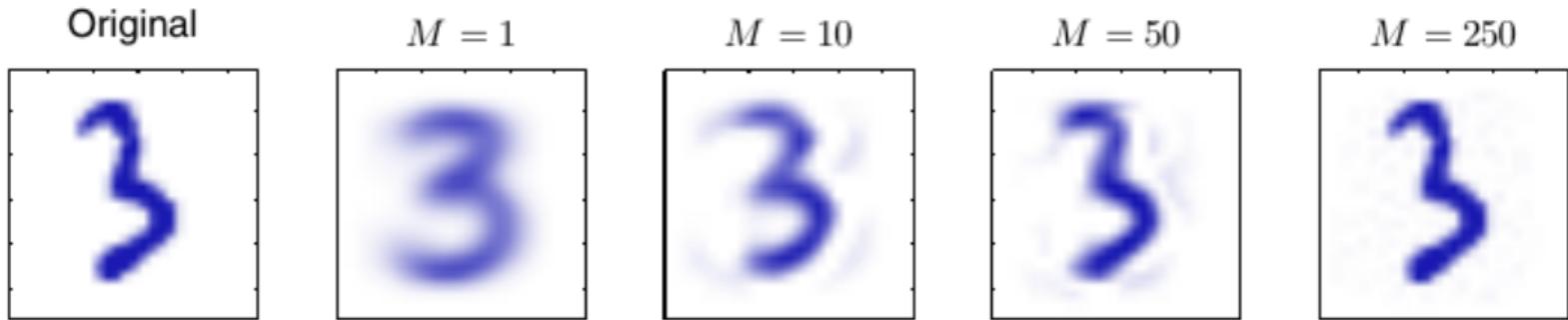


$\times 10^6$

distortion



PCA Reconstruction



- example $D = 10,000, N = 1,000,000$
- original: $N \times D = 10,000,000,000$ parameters
- PCA ($M = 10$):
 $M \times D + N \times M = 100,000 + 10,000,000 = 10,100,000$, reduction 990 times
- PCA ($M = 250$):
 $M \times D + N \times M = 2,500,000 + 250,000,000 = 252,500,000$, reduction 39 times

Eigenfaces

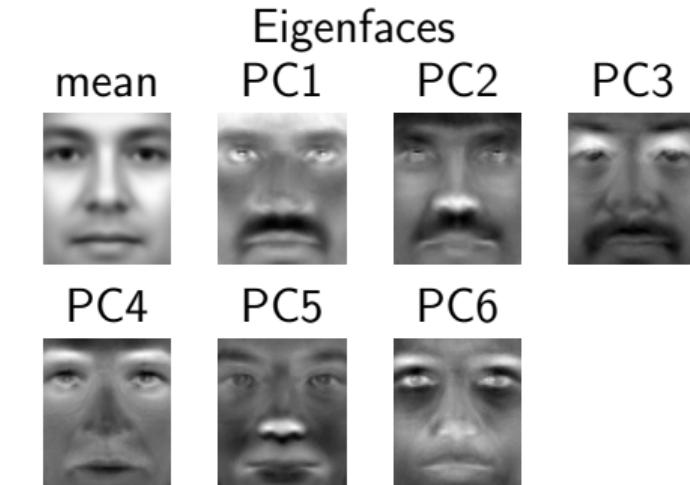


Faces from the FERET database



$64 \times 73 \text{ pixels}$
 $= 4672 \text{ dimensions!}$

Eigenfaces



from 4672 dimensions to a small basis

Faces from the FERET database

PCA for high-dimensional data

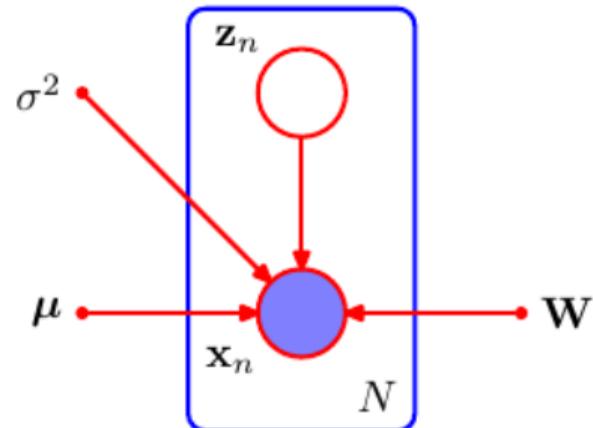
- N points in D -dimensional space, with $N < D$
- they define a subspace of at most $N - 1$ dimensions
- example: 2 points always on a line, 3 points always on a plane...
- $D - N + 1$ eigenvalues are zero!
- in the direction of the corresponding eigenvector: zero variance
- we can reformulate the eigenvector equation with a $N \times N$ matrix

Probabilistic PCA

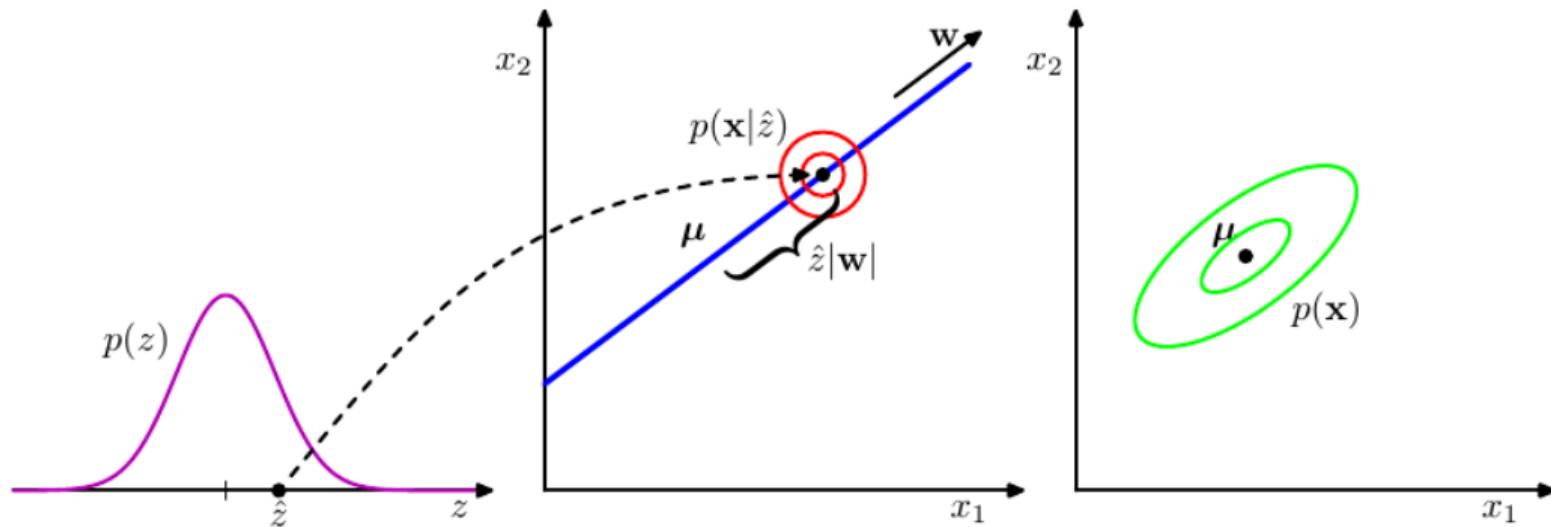
- probabilistic latent variable model
- solve with maximum likelihood

Model:

- $p(\mathbf{z}) = \mathcal{N}(\mathbf{z} | \mathbf{0}, \mathbf{I})$
- $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x} | \mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2 \mathbf{I})$
- with \mathbf{W} $D \times M$ matrix spanning the linear (principal) subspace



Probabilistic PCA: Generative View



- $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$
- $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{Wz} + \boldsymbol{\mu}, \sigma^2 \mathbf{I})$
- $\mathbf{x} = \mathbf{Wz} + \boldsymbol{\mu} + \boldsymbol{\epsilon}$

Probabilistic PCA: Advantages

- can be used to constrain the number of parameters in multivariate Gaussian
- can be solved with EM (computationally efficient)
- can deal with missing values
- we can extend it to mixture of PCA models
- Bayesian version can estimate the number of principal components
- likelihood function: points that are close to principal subspace but far from data distribution
- can create class-conditional densities (classification)
- can be used to generate (sample) data.

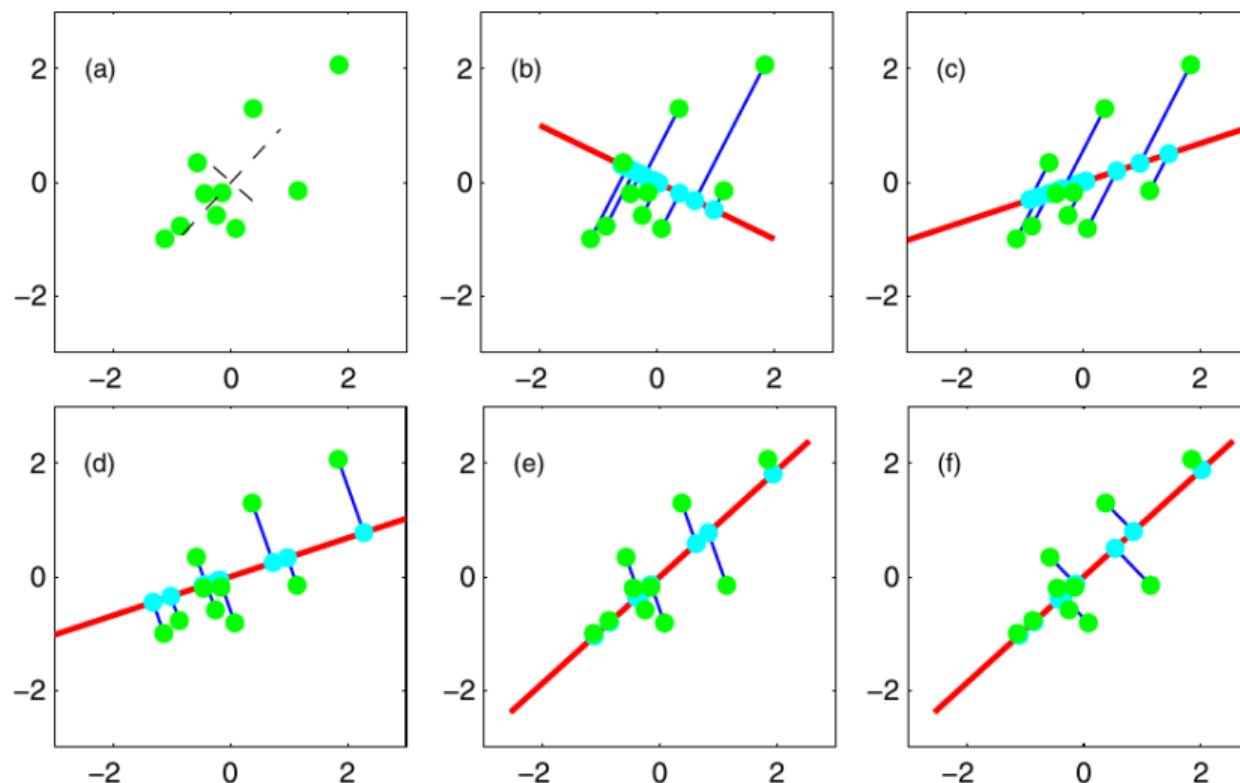
Maximum Likelihood PCA

- there exist a closed form solution to ML
- predictive distribution $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2\mathbf{I})$ is redundant:
rotations of \mathbf{W} give the same distribution
- λ_i variance in principal direction i
- σ^2 variance orthogonal to principal subspace
- statistical nonidentifiability

EM algorithm for PCA

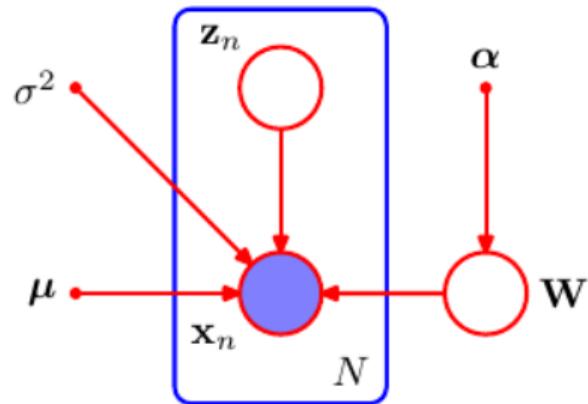
- convenient in high dimensional space (iterative instead of sample covariance matrix)
- missing values (if missing at random)
- works even for sigma square to zero (EM for standard PCA)

EM for PCA: Physical Interpretation (rod and springs)



Bayesian version can find the intrinsic dimensionality

- solution intractable
- can be approximated



Factor Analysis

- $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \boldsymbol{\Psi})$
- $\boldsymbol{\Psi}$ diagonal (in PCA it was $\boldsymbol{\Psi} = \sigma^2\mathbf{I}$)

Independent Component Analysis (ICA)

- latent distribution $p(\mathbf{z})$ is non-Gaussian
- if $p(\mathbf{z})$ factorizes into $\prod_{j=1}^M p(z_j)$ then ICA

Example: blind source separation

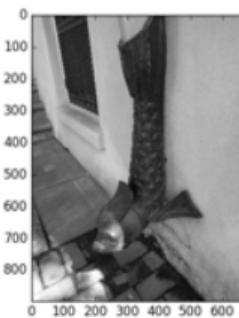
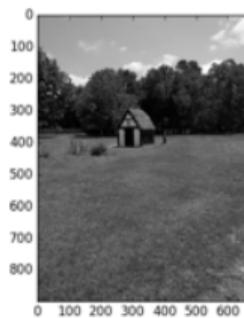
Blind Source Separation (Speech)

- N voices picked up by M microphones
- usually $M = N$
- each microphone picks up a linear combination of the two
- ignoring room acoustic and relative movements of sources and mics
- ICA can separate the voices perfectly

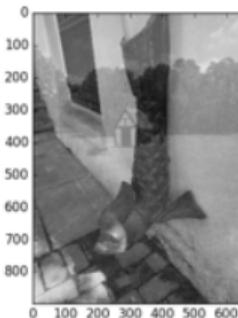
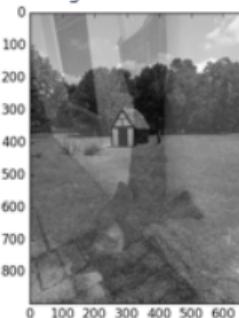
<http://www.kecl.ntt.co.jp/icl/signal/sawada/demo/bss2to4/index.html>

Blind Source Separation (Images)

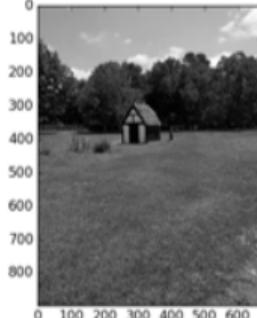
Original Signals



Mixed Signals



Separated signals

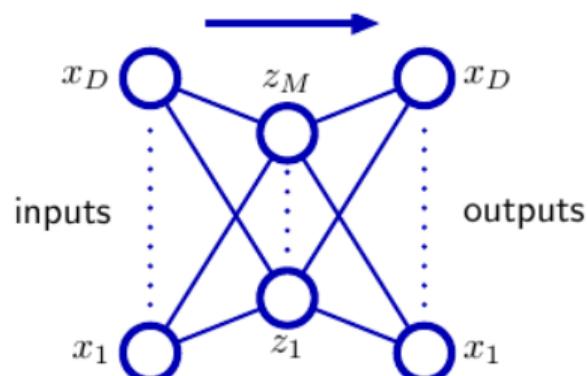


source Wikipedia

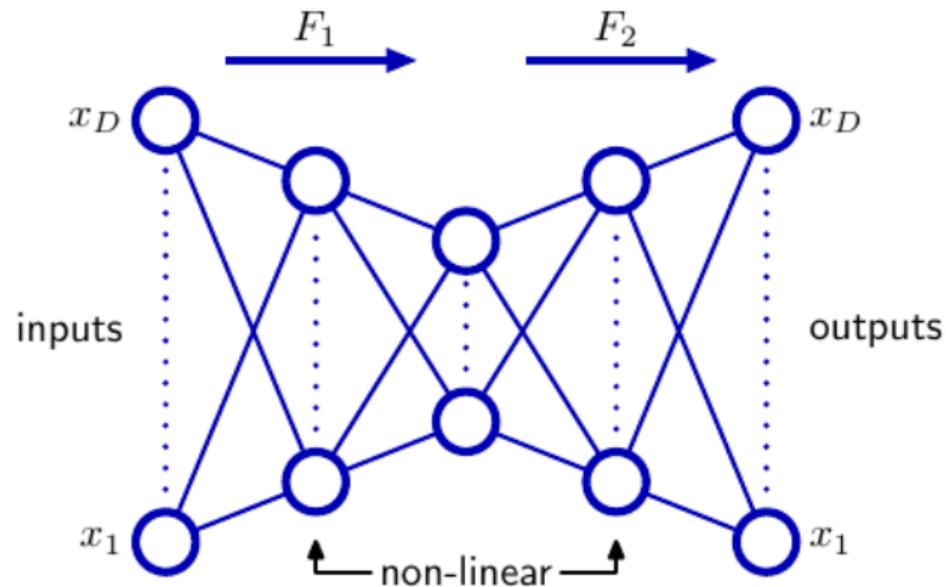
Autoencoders (linear manifolds)

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \|\mathbf{y}(\mathbf{x}_n, \mathbf{w}) - \mathbf{x}_n\|^2$$

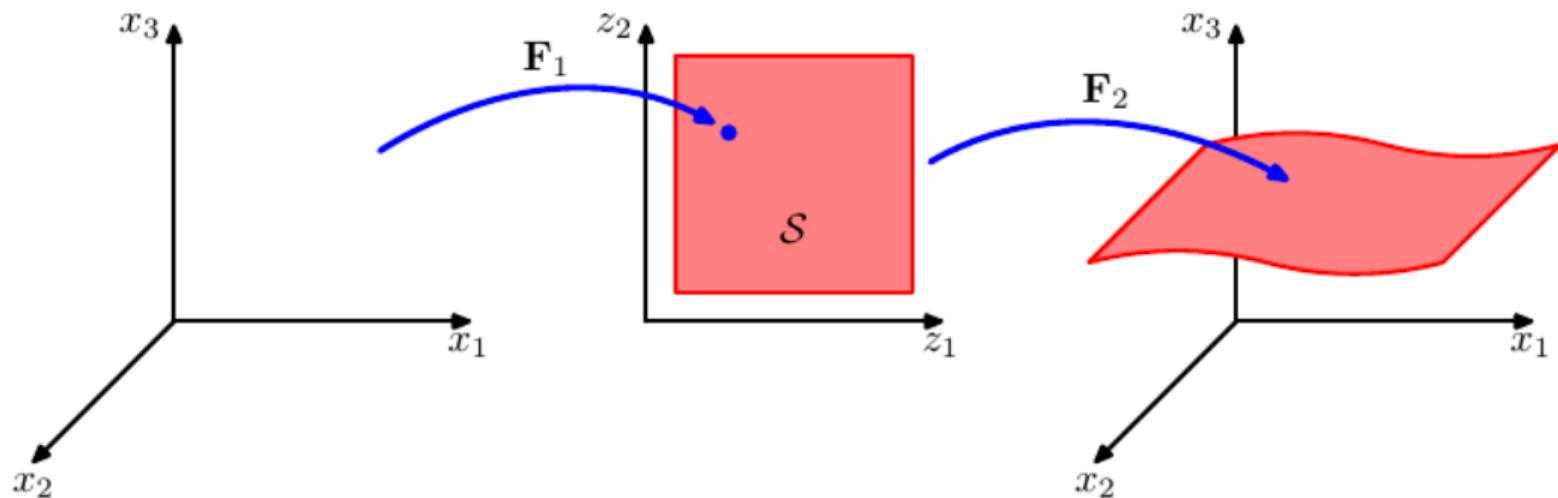
- if linear activations, then global minimum
- similar to PCA, but not orthogonal and normalized PCs
- still linear subspace even for nonlinear activations



Autoencoders (non-linear manifolds)



Autoencoders: mapping illustration

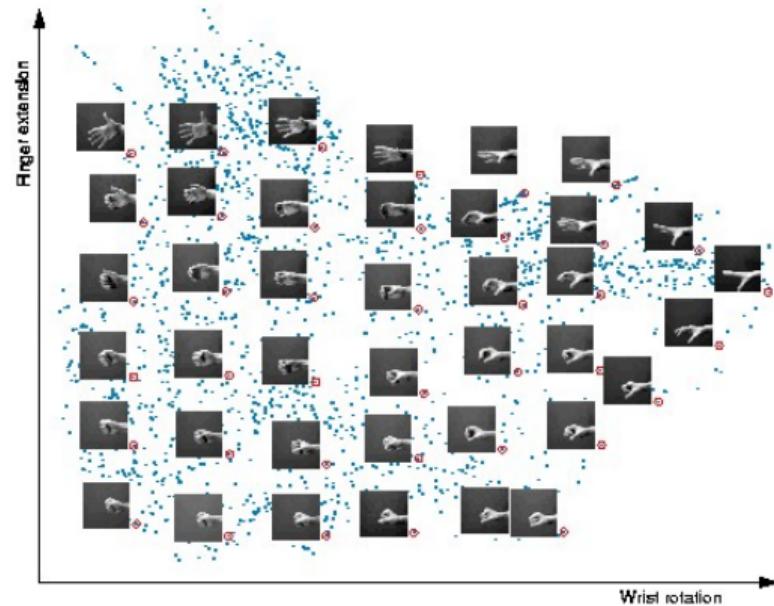
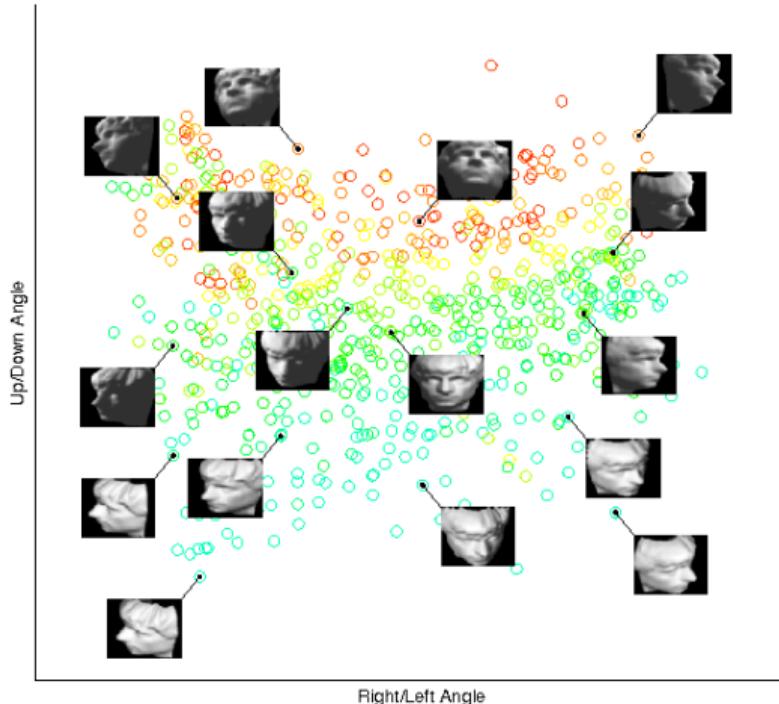


Isometric Feature Mapping (Isomap)

Using geodesic distances

<https://chart-studio.plotly.com/~empet/14345.embed>

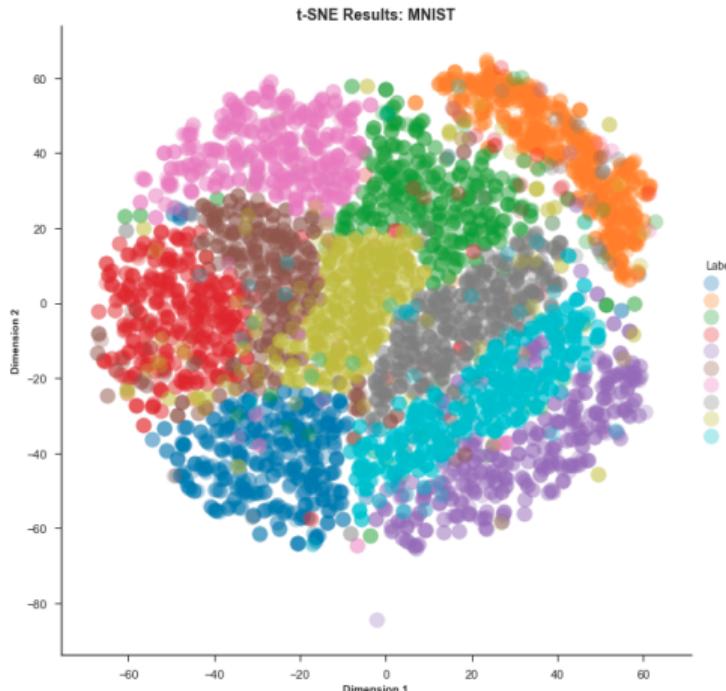
Isomap Examples



t-SNE (not in the book)

t-distributed stochastic neighbor embedding

- works best for visualization (2-3 dim)
- similar groups of points are close
- probability distribution over pairs of points in high dim (pairs of more similar points have higher probability)
- probability distribution over pairs of points in low dimensions
- minimize KL divergence



van der Maaten, L. and Hinton, G. E. (2008). Visualizing data using t-SNE. J. Machine Learning Res., 9 . 473 , 516

Summary

TTT4185 Machine Learning for Signal Processing

Giampiero Salvi

Department of Electronic Systems
NTNU

HT2021

A course on Machine Learning

Teaching and learning activities:

- 24 lectures
- 3 computer exercises

Assessment

- computer exercise submission
- computer exercise oral discussion
- written exam (2020-12-17, 9:00-13:00)
- Grade: A-F
- permitted examination aids: D (basic calculator)

You can find previous exams in Blackboard. Expect small changes (the course has evolved in the past years)

Challenges

Finding balance between

- theory vs practice
- general methods vs bleeding edge state-of-the-art
- probabilistic vs connectionist
- bayesian vs frequentist

Course content

Domain Knowledge

- speech production (source/filter model)
- speech perception (frequency and amplitude scales)
- speech analysis (linear prediction, cepstrum, MFCCs)
- for other fields you will need specific domain knowledge

Prerequisites

- probability theory
- decision theory
- information theory

Methods (next slide)

Methods

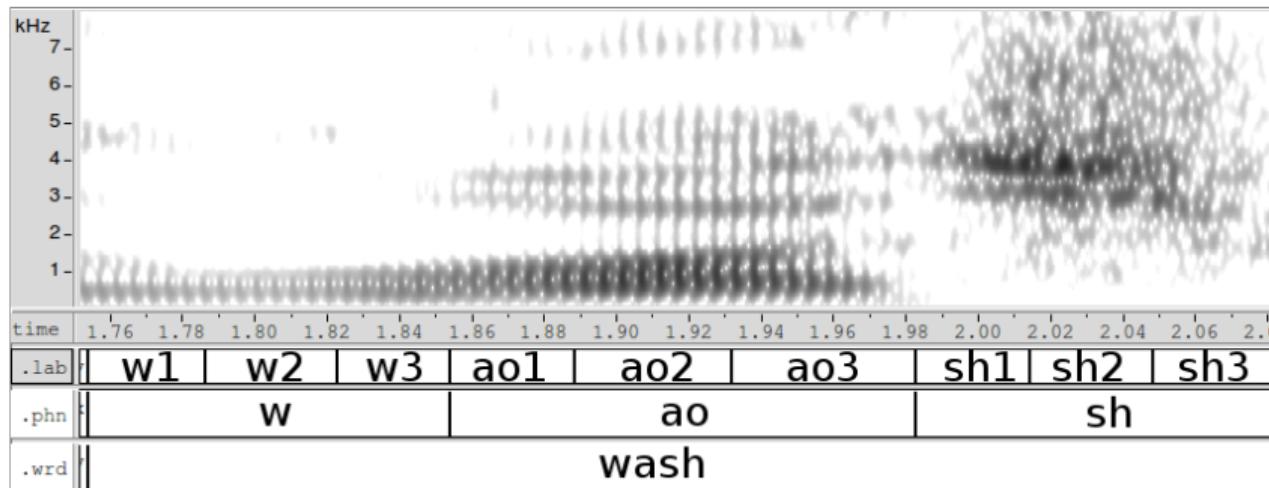
- probabilistic linear regression
- probabilistic classification (Bayes classifier)
- ML vs MAP estimates (regularization)
- least squares, fisher's discriminant, perceptron
- logistic regression
- kernel methods and support vector machines
- deep neural networks (DNNs, CNNs, RNNs, LSTMs, GANs)
- graphical models (Bayesian networks)
- k-means and mixture models (unsupervised learning)
- hidden Markov models (sequence learning)
- dimesionality reduction (PCA, FA, ICA, AEs, Isomap, t-SNE)

ML Problems

- Supervised vs unsupervised learning
- Supervised classification vs regression
- Unsupervised clustering vs distribution estimation
- Dimensionality reduction (compression or visualization)

Kind of observations

- independent and identically drawn (i.i.d.)
- sequences (order matters)



Theoretical aspects

- model complexity and overfitting
- bias vs variance trade-off
- curse of dimensionality
- generative vs discriminative methods

What we left out

- ensemble methods (boosting, decision trees, decision forests)
- probabilistic neural networks
- fully Bayesian methods
- approximated inference (sampling and variational methods)
- reinforcement learning
- active learning
- ...

Oral Presentations

- you will receive an announcement today on Blackboard
- book a time for oral presentations
- dates will be around the end of week 47 (next week)
- questions will be mostly based on understanding rather than implementation

Exam

- three exercises giving a number of points each
- motivating your answer is as important as giving the correct answer
- correct answers with no motivation give zero points
- the final grade is based on the total number of points

Previous exams in Blackboard have detailed solutions.

Pablo Ortiz (Telenor Research, Oslo)

“Spoken language understanding: from RNNs to Transformers”

This lecture will start by introducing real-world problems in the field of machine learning for language understanding, in particular machine translation and speech recognition. In the first part we will see how neural networks such as CNNs and RNNs can be applied to these problems. Towards the second part we will focus on attention mechanisms, and discuss in detail the Transformer. In particular we will cover some of the most recent transformer-based architectures that are producing groundbreaking results in the fields of natural language and computer vision, such as BERT and GPT-3. Throughout the lecture the focus will be on the concepts and we will refer to the literature for the mathematical or technical details.