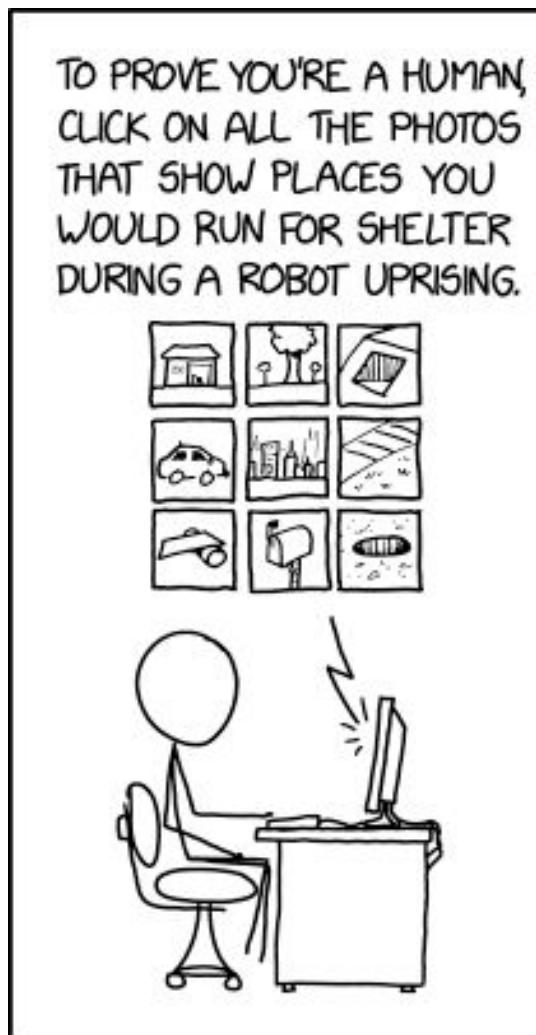


<https://xkcd.com/2228/>



<https://xkcd.com/1897/>



SO MUCH OF "AI" IS JUST FIGURING OUT WAYS TO OFFLOAD WORK ONTO RANDOM STRANGERS.

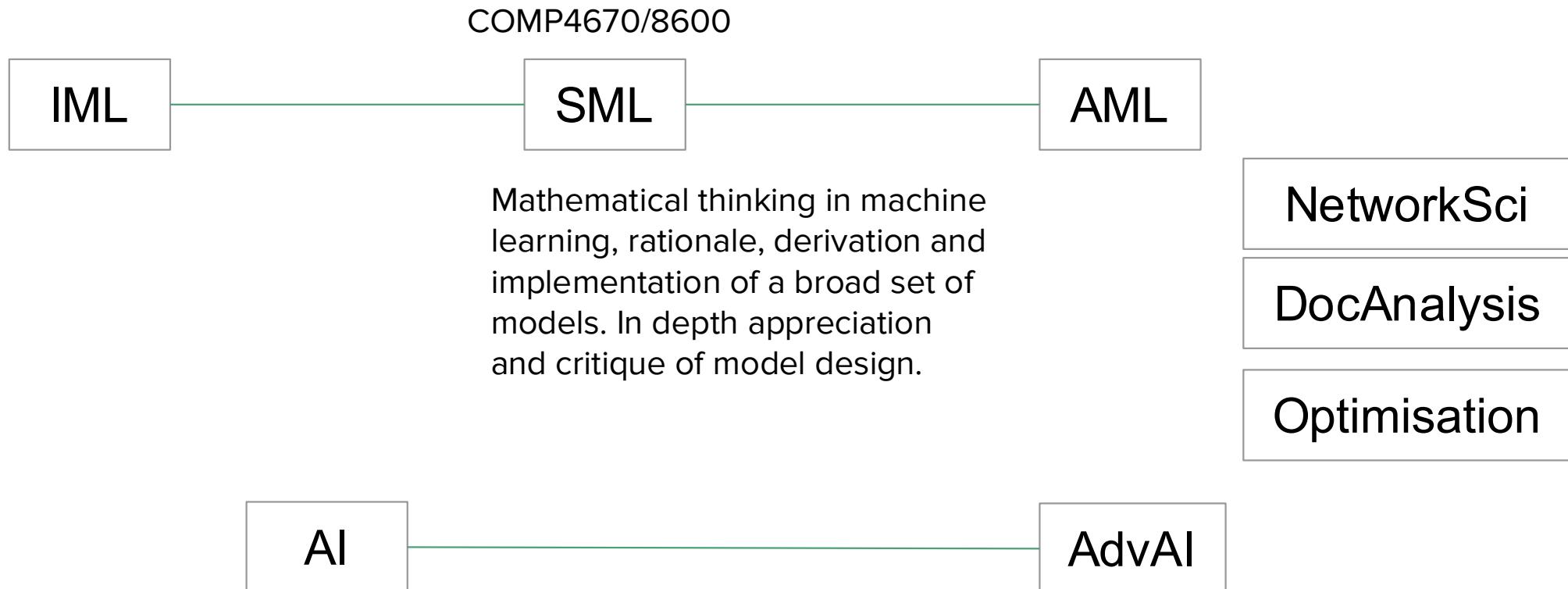
# COMP4670/8600 Statistical Machine Learning - 2022

## Agenda for today

- Course logistics and administrivia
- Helicopter view of ML + Scope of this class

# *(Introduction to) Statistical Machine Learning*

This course provides a broad but thorough introduction to the methods and practice of statistical machine learning.



# SML Team



Lexing Xie



Alexander Soen



Tianyu Wang



Ekaterina (Katya) Nikonova



Chamin Hewa Koneputugodage



Shidi Li



Josh Nguyen



Minchao Wu



Ruiqi Li



Haiqing Zhu



Belona Sonna



Dillon Chen



Rong Wang

## Past lecturers (since 2009)

Chris Weber

Cheng Soon Ong

Christian Walder

# Hello SML videos - for you to get to know the tutors

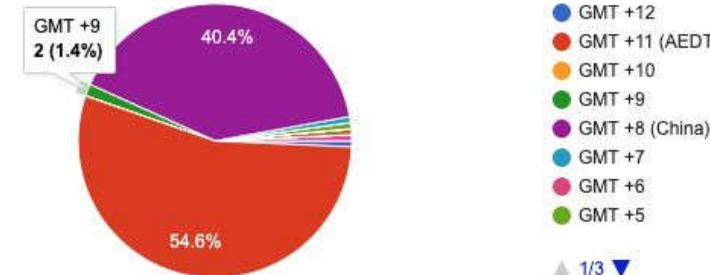
A few words about yourself, what're you working on  
What's your favorite ML algorithm, and why  
Your one advice to SML students  
One common misconception about machine learning

# Logistics + technology

- Microsoft Teams
  - live meetings of “lectures” (recorded)
  - “tutorials/labs” (not recorded)
- In person lectures and labs: TBD
  - We aim to improve the experience in 2022 S1, conditioned on the safety of all students and course staff
  - To be finalised after discussing with tutor team
- Course webpage <http://cm.cecs.anu.edu.au/sml2022/>  
infrequent, read-only updates
- Piazza: Announcements, Questions, and Discussions
- Gradescope: Assignment submission + grading
- Wattle: quiz 1+2, final exam

What timezone are you in? (Round to the nearest hour)

141 responses



▲ 1/3 ▼

# Course structure

Learning is done by the student, i.e. You :)

We are here to help, and here're a few ways.

- Online, live lecture sessions
  - **E** [lecture time] ~ 1.5 hours
  - the rest are used to cover assignments, Q&A, etc
- Weekly tutorials with hands-on exercises, starting week 2
  - Signup to an in-person session on wattle if you're on campus
- Assessment items
  - See website

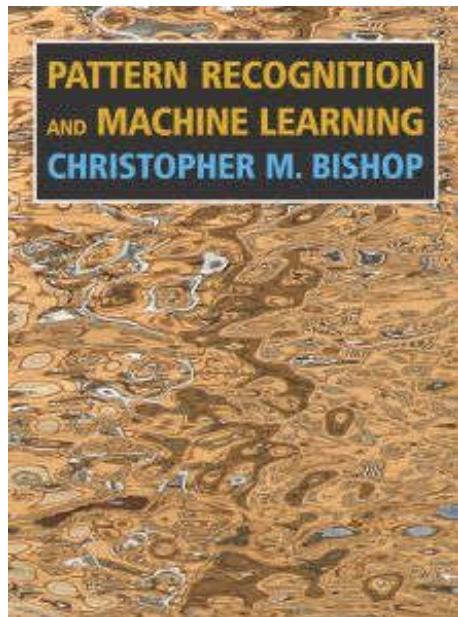
# Studying in unusual times

- Online + offline delivery --- lectures, tutorials, discussions between students
  - We strongly encourage you to study with other students
  - Technology: Zoom, Slack, ...
- The two written+programming assignment **can** be submitted **in pairs**
  - Each student can choose to submit the assignment individually or with another student.  
Students are free to change the teaming between assignment 1 and assignment 2.
  - Students submitting in a pair act as one unit:  
may share resources (such as notes) with each other and write the solutions together
  - Both of the two students should fully understand all the answers in their submission  
Each student in the pair must understand the solution well enough in order to reconstruct it by him/herself
- Video assignment – individual, specs will be available.
- Online quiz x 2
- Online exam (at home)
  - During exam period

# Pre-requisites

- Probability
    - random variables, distribution, conditional probability, expectation, variance, density
  - Linear algebra
    - matrix multiplication, eigenvector, solving linear systems, matrix inverse
  - Python, via jupyter notebook and stand alone functions / scripts.
- 
- This is a mathematically intensive course. But that's why it's exciting and rewarding! 😊
  - Practice / assess your pre-requisite in this take-home exam
    - Available on Piazza (resource section)
    - Solutions will be released by end of week 1

# Text book



Well-written, pedagogically.

A lot of timeless material.

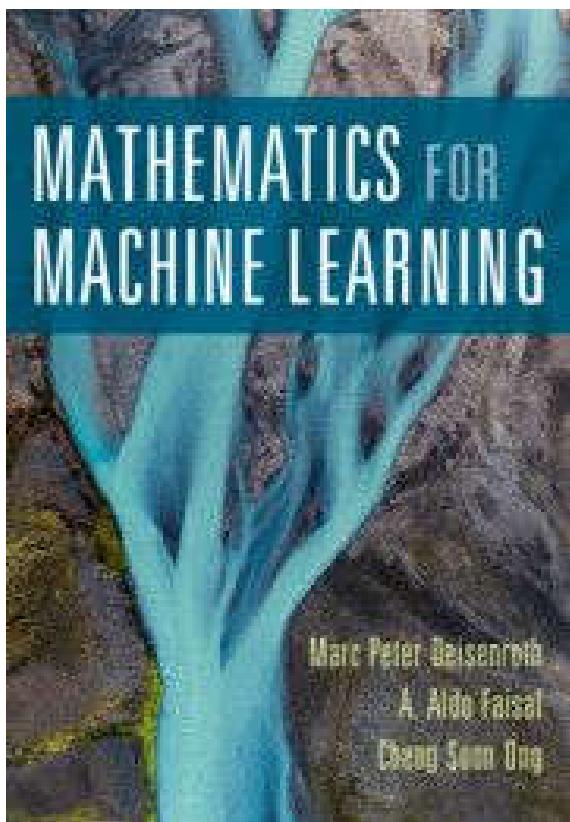
Note style difference with MML.

A note about exercises in the book and listed on tutorial sheets:  
Most of them intend to fill in the gaps or enrich the main exposition, few of them are designed to test your understanding to the presented concept or use them in relevant problems.

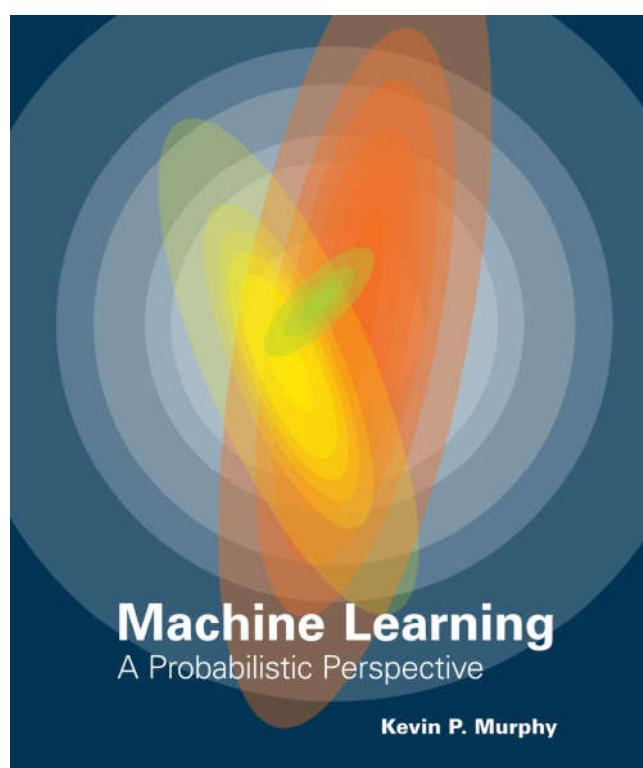
They are not similar to questions in the assignment or exam.

<https://www.microsoft.com/en-us/research/people/cmbishop/prml-book/>

# References



<https://mml-book.com>



New version <https://probml.github.io/pml-book/book1.html>

I	Introduction	1
I	Foundations	19
2	Probabilistic inference	21
3	Probabilistic models	41
4	Parameter estimation	77
5	Optimization algorithms	99
6	Information theory	145
7	Bayesian statistics	163
8	Bayesian decision theory	221
II	Linear models	247
9	Linear discriminant analysis	249
10	Logistic regression	265
11	Linear regression	305
12	Generalized linear models	353
III	Deep neural networks	369
13	Neural networks for unstructured data	371
14	Neural networks for images	407
15	Neural networks for sequences	443
IV	Nonparametric models	469
16	Exemplar-based methods	471
17	Kernel methods	491
18	Trees, forests, bagging and boosting	533
V	Beyond supervised learning	553
19	Learning with fewer labeled examples	555
20	Dimensionality reduction	591
21	Clustering	639
22	Recommender systems	663
23	Graph embeddings	675
VI	Appendix: Mathematical background	699
A	Some useful mathematics	699
B	Linear algebra	719
C	Probability	759
D	Frequentist statistics	779
E	Exercises	815

# Positioning of this class

“ML Researchers” roles, in analogy to Music

[credit: Preface in MML book]

- Astute listener: cloud-based tools, focus on extracting insight, critique the correctness, interpretation of results, reason about fairness, ethics, etc.
- Experienced artist: understands ML interfaces, their uses, benefits and limits.
- Fledgling Composer: develop and extend existing algorithms, uncover relationships between different tasks.

← **SML**

# CECS Class Representatives

Class Student Representation is an important component of the teaching and learning quality assurance and quality improvement processes within the ANU College of Engineering and Computer Science (CECS).

The role of Student Representatives is to provide ongoing constructive feedback on behalf of the student cohort to Course Conveners and to Associate Directors (Education) for continuous improvements to the course.

## **Roles and responsibilities:**

- Act as the official liaison between your peers and convener.
- Be creative, available and proactive in gathering feedback from your classmates.
- Attend regular meetings, and provide reports on course feedback to your course convener
- Close the feedback loop by reporting back to the class the outcomes of your meetings.

Any questions so far?

# Scope of this course

What is Machine Learning?

A (simplistic) taxonomy of machine learning

(Biased) samples of ML projects at the ANU

Pressing concerns in fairness, accountability and transparency

What we will not cover

Thanks: many slides are from Stanford CS229

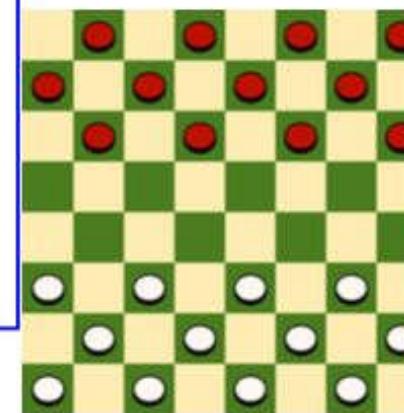
# Definition of Machine Learning

Arthur Samuel (1959): Machine Learning is the field of study that gives the computer the ability to learn without being explicitly programmed.



A. L. Samuel\*

**Some Studies in Machine Learning  
Using the Game of Checkers. II—Recent Progress**



Photos from wikipedia

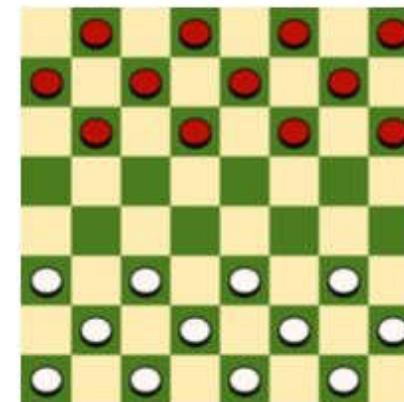
# Definition of Machine Learning

Tom Mitchell (1998): a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.



Experience (data): games played by the program (with itself)

Performance measure: winning rate



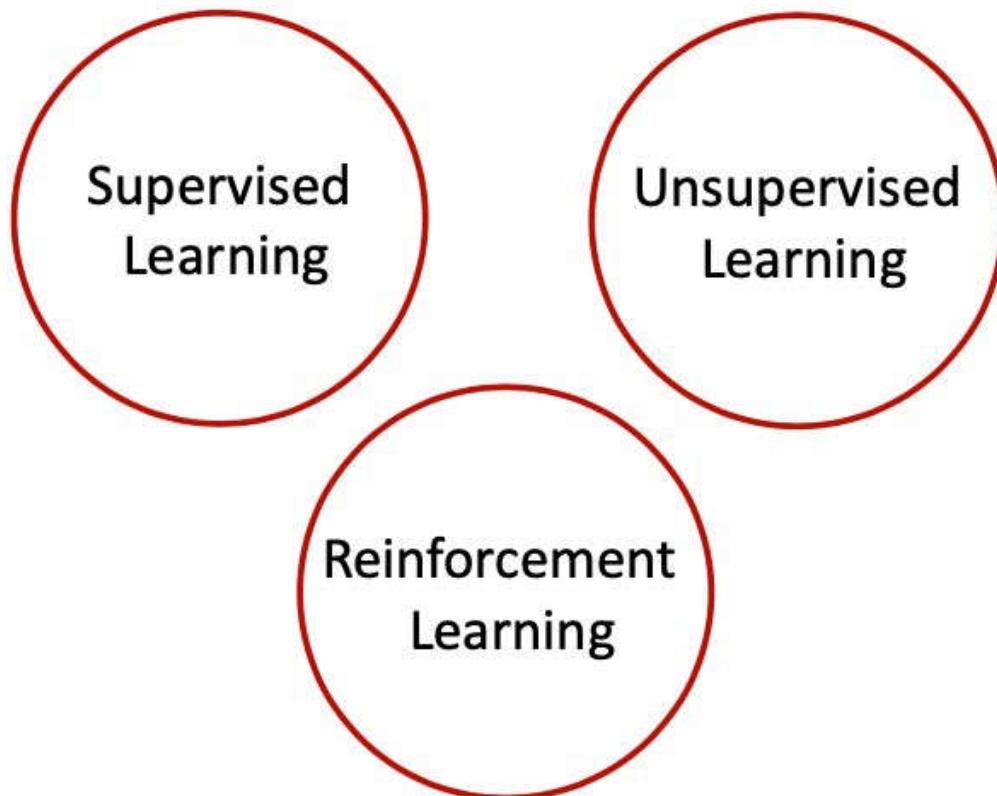
Photos from Tom Mitchell's homepage

# Statistical Machine Learning - some history

- 1960's : symbolic AI; computers learn rules from data; analysis of the underlying statistics is seldom done.
- Perceptron (Rosenblatt, 1957), "Perceptrons" (Minsky and Papert, 1969)
- 
- 1980's : artificial neural networks
- 1990's - 2000's : statistical machine learning (kernel methods, decision trees, graphical models)
- Why Statistical Machine Learning not earlier?
  - faster computers with larger memory to represent statistical models have become available
  - numerical methods on the desktop computer (BLAS, LAPACK, Optimisation)
  - found new interesting classes of algorithms (e.g. on graphs)
  - large amounts of data available which can be tapped into (flickr, social networks)
  - many data sets with partial/incomplete data (e.g. netflix)

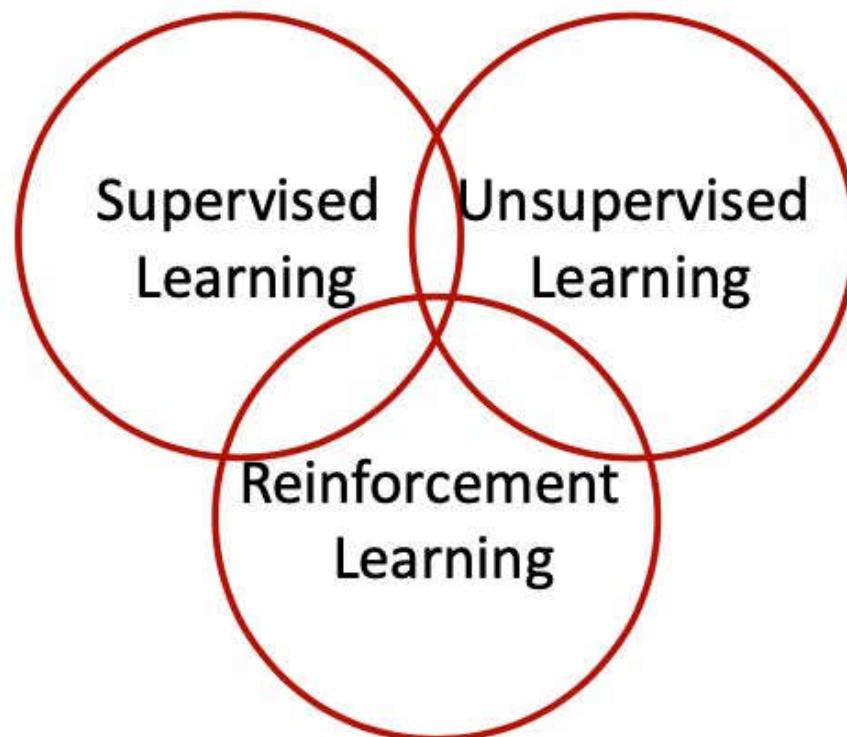
# Machine Learning Taxonomy

(A Simplistic View Based on Tasks)



# Machine Learning Taxonomy

(A Simplistic View Based on Tasks)



Can also be viewed as sets  
of tools/methods.



[← Back to Search](#) | [Home](#) > [Buy \(ACT\)](#) > [North Canberra](#) > [Lyneham](#) > 7 Glover Street

NEW

Add to watchlist

Auction 24/02/21

7 Glover Street, Lyneham ACT 2602

House • 3 1 1 1.0

Block size: 562 m<sup>2</sup> approx.

UV: \$575,000 (2020)

**BLACKSHAW**

Next Inspection: Wed 10 Feb, 5:00 PM



**Christine Shaw**  
Blackshaw Manuka

Call

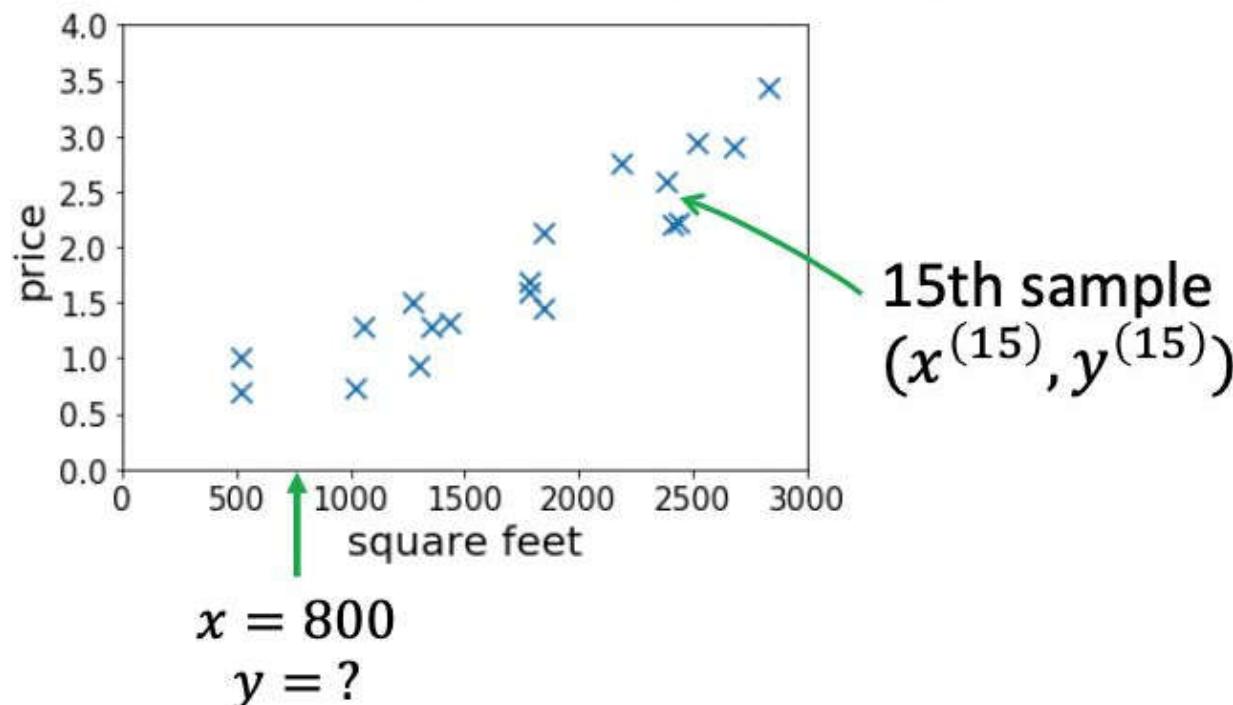
<https://www.allhomes.com.au/7-glover-street-lyneham-act-2602>

# Supervised Learning: Housing Price Prediction

- Given: a dataset that contains  $n$  samples

$$(x^{(1)}, y^{(1)}), \dots (x^{(n)}, y^{(n)})$$

- Task: if a residence has  $x$  square feet, predict its price?

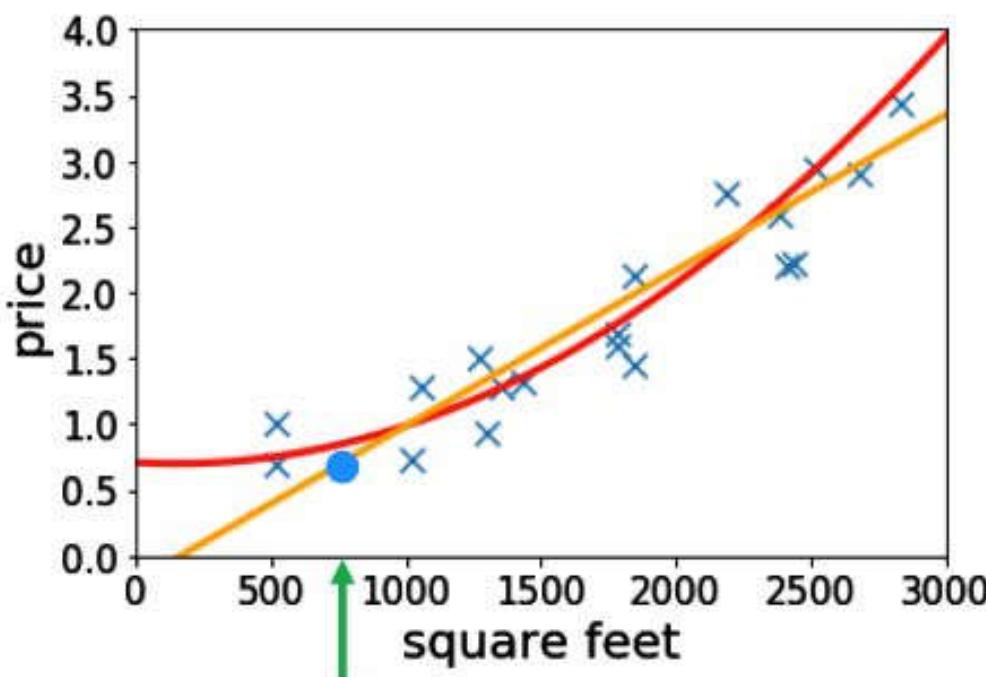


# Housing Price Prediction

- Given: a dataset that contains  $n$  samples

$$(x^{(1)}, y^{(1)}), \dots (x^{(n)}, y^{(n)})$$

- Task: if a residence has  $x$  square feet, predict its price?



Fit a line or  
quadratic curve to  
the data? Week 1  
and 2

## More features

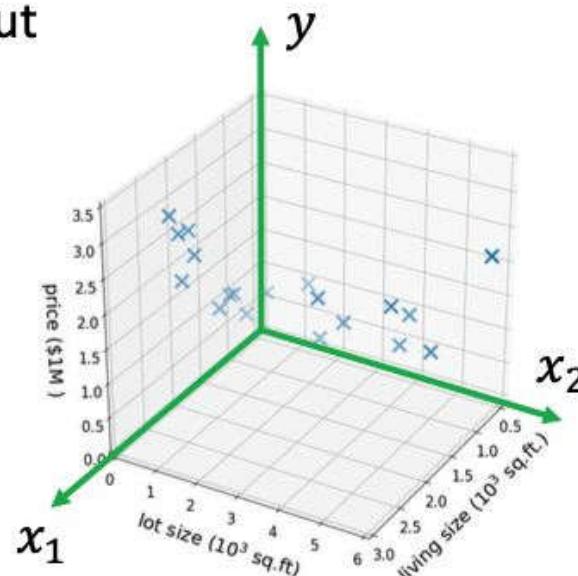
- Suppose we also know the lot size
- Task: find a function that maps

$\underbrace{(\text{size}, \text{lot size})}_{\text{features/input}} \rightarrow \underbrace{\text{price}}_{\text{label/output}}$

$$x \in \mathbb{R}^2$$
$$y \in \mathbb{R}$$

- Dataset:  $(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)})$
- where  $x^{(i)} = (x_1^{(i)}, x_2^{(i)})$
- “Supervision” refers to  $y^{(1)}, \dots, y^{(n)}$

Even more features? Number of rooms, year built, energy rating, ...



## High-dimensional features

➤  $x \in \mathbb{R}^d$  for large  $d$

➤ E.g.,

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_d \end{bmatrix} \begin{array}{l} \text{--- living size} \\ \text{--- lot size} \\ \text{--- \# floors} \\ \text{--- condition} \\ \text{--- zip code} \\ \vdots \end{array} \longrightarrow y \text{ --- price}$$

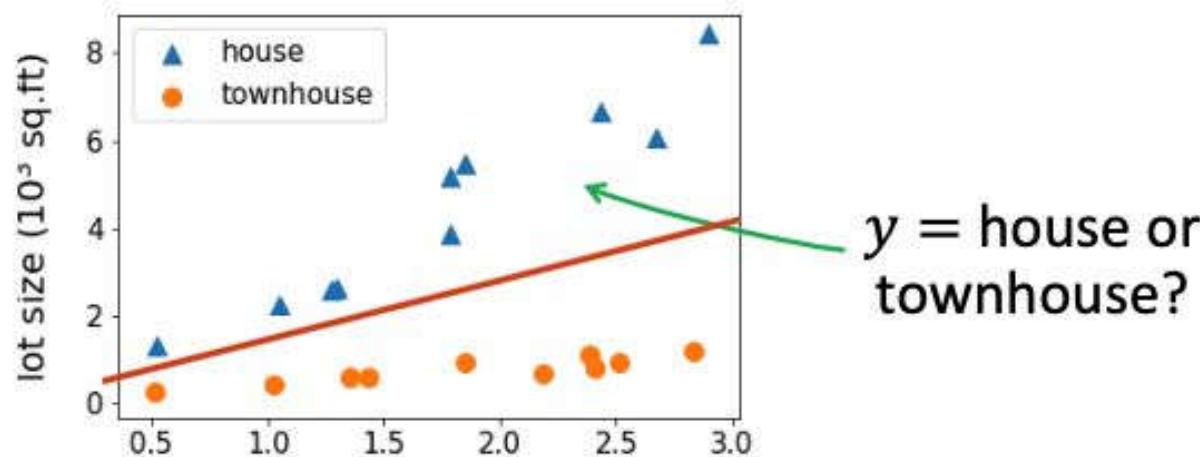
# Regression vs Classification

Week 2: (linear) regression

Week 3: (linear)  
classification

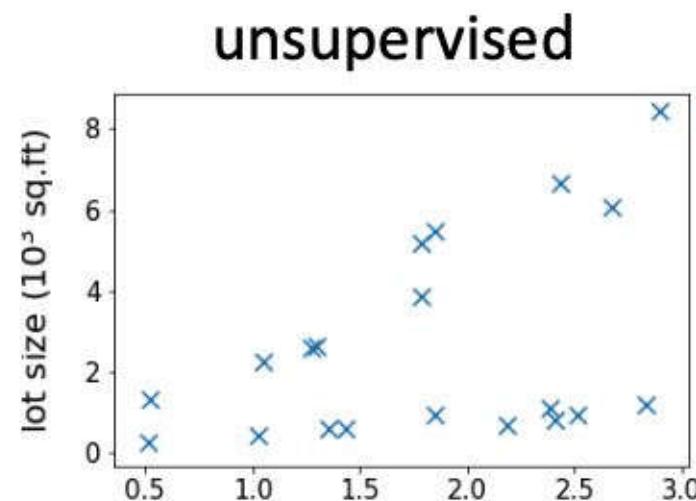
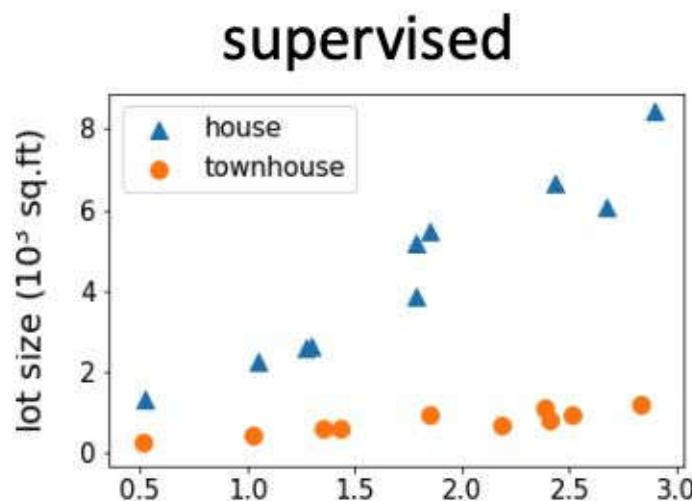
- regression: if  $y \in \mathbb{R}$  is a continuous variable
  - e.g., price prediction
- classification: the label is a discrete variable
  - e.g., the task of predicting the types of residence

(size, lot size) → house or townhouse?



# Unsupervised Learning

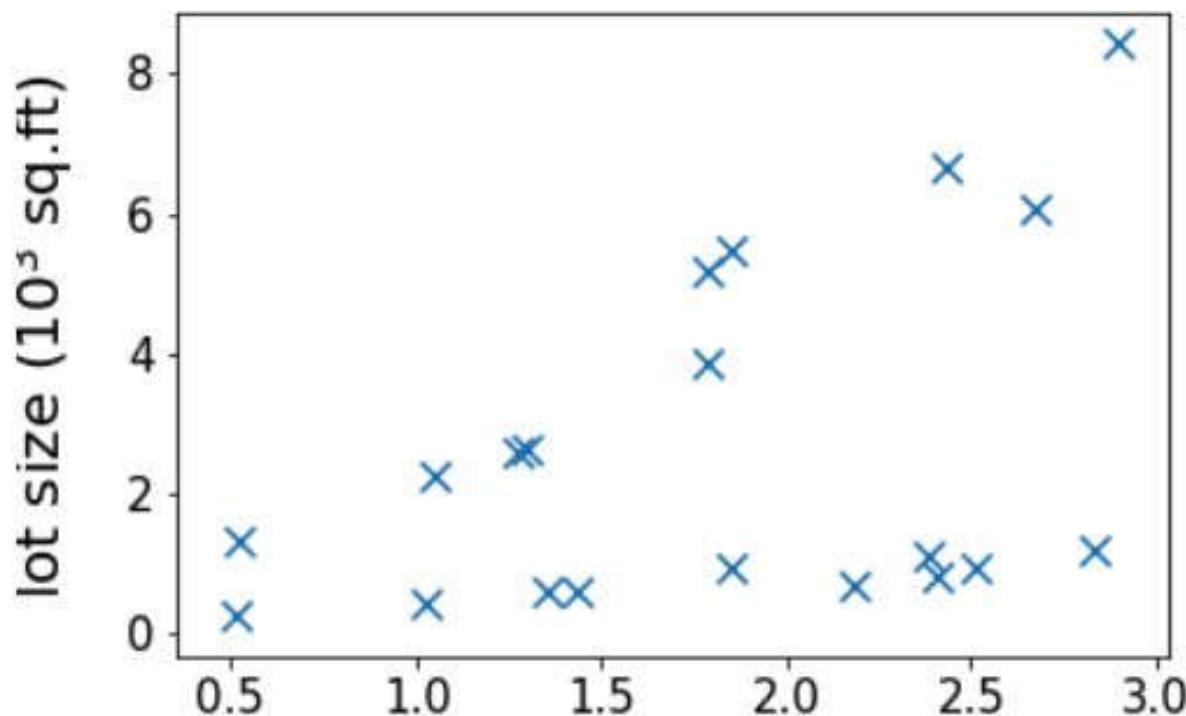
- Dataset contains **no labels**:  $x^{(1)}, \dots x^{(n)}$
- **Goal** (vaguely-posed): to find interesting structures in the data



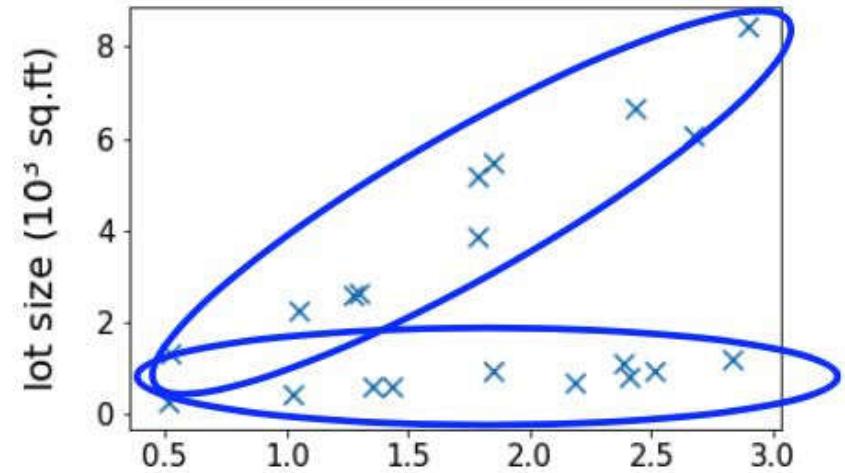
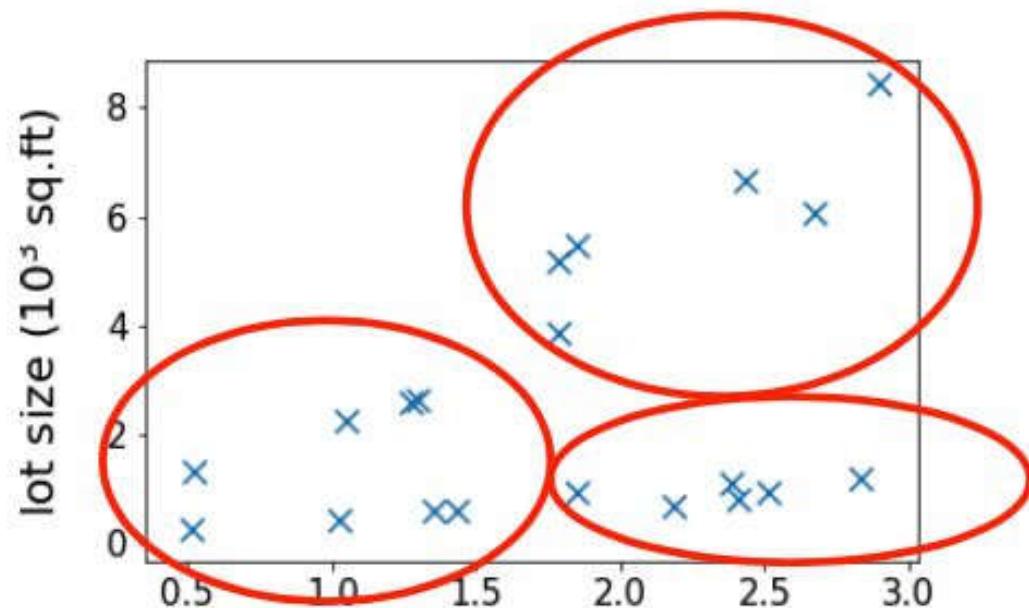
# Clustering

Week 5: Gaussian  
mixture models, k-means

Week 4: PCA



# Clustering



# Supervised Learning in Computer Vision

Covered in the computer vision course

## ➤ Image Classification

- $x$  = raw pixels of the image,  $y$  = the main object

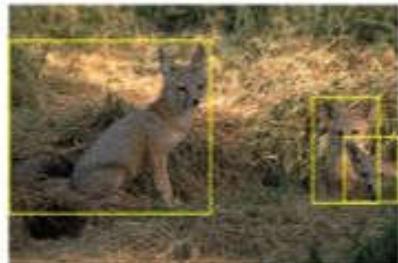


ImageNet Large Scale Visual Recognition Challenge. Russakovsky et al.'2015

# Supervised Learning in Computer Vision

Covered in the computer vision course

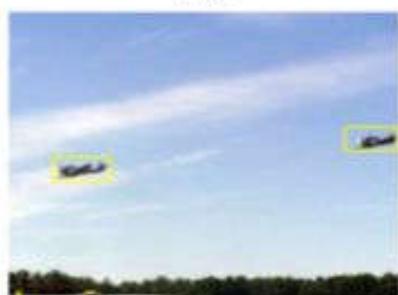
- Object localization and detection
  - $x$  = raw pixels of the image,  $y$  = the bounding boxes



kit fox



croquette



airplane



frog

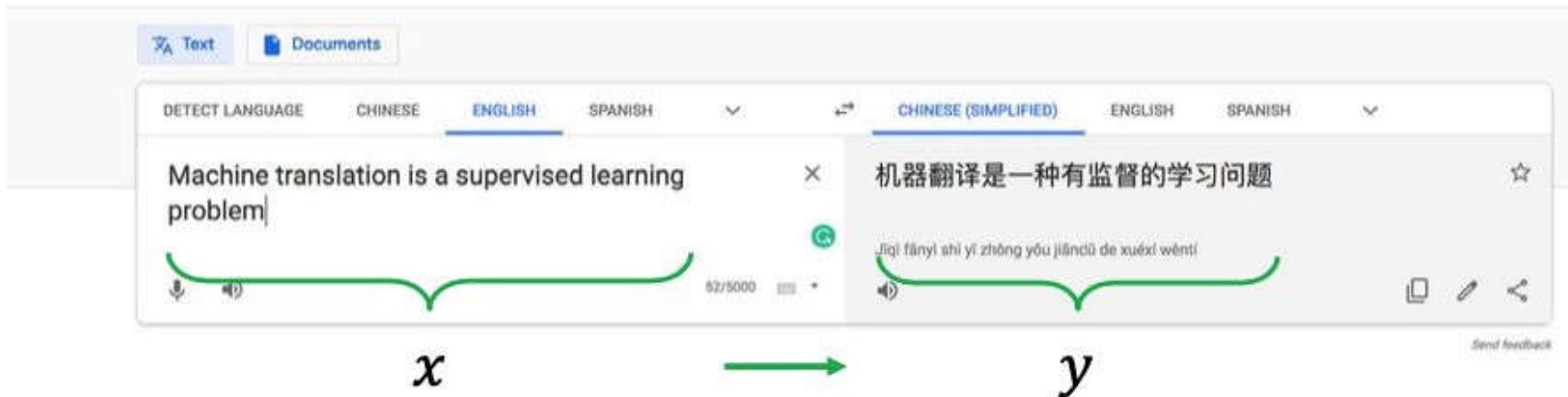
ImageNet Large Scale Visual Recognition Challenge. Russakovsky et al.'2015

# Machine Learning for NLP

Take Document Analysis  
(COMP4650) if you'd  
like to learn more.

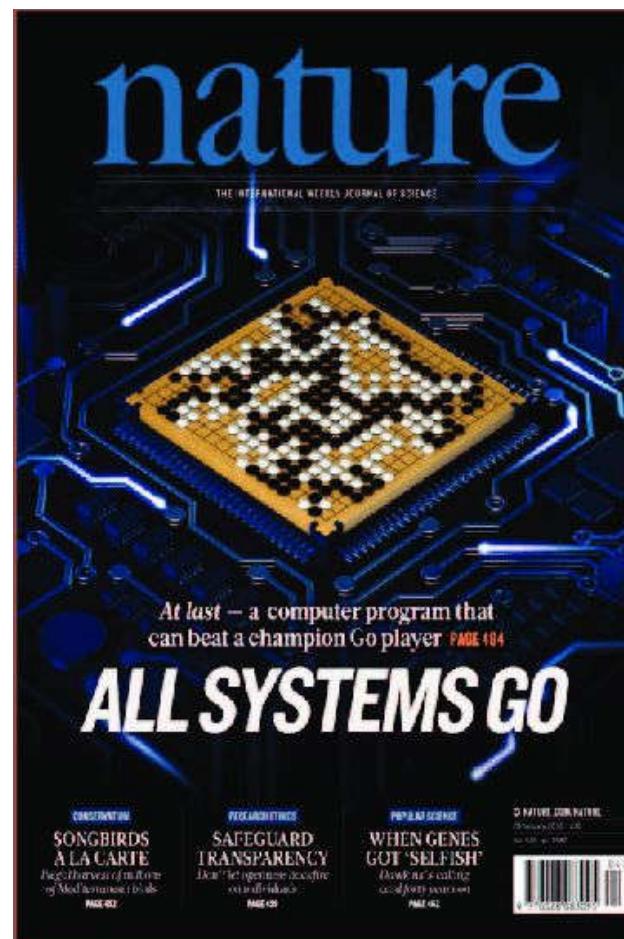
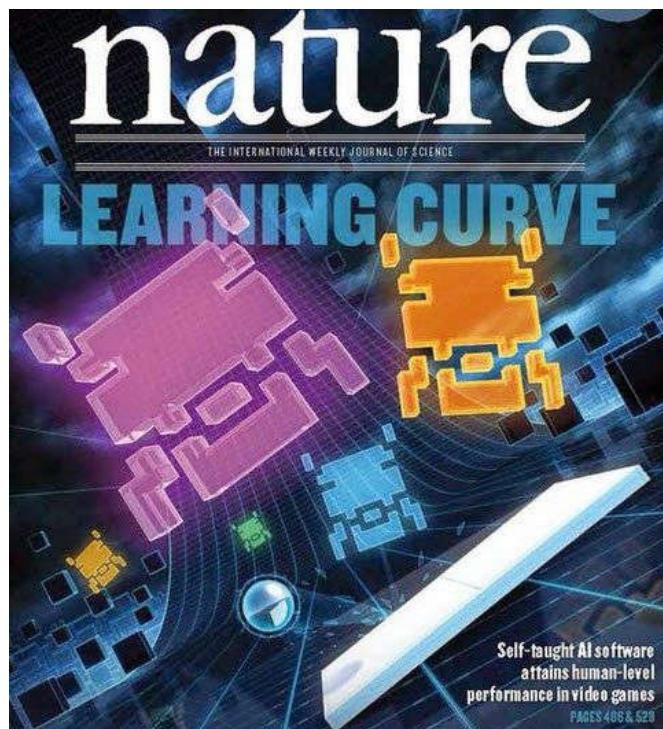
## ➤ Machine translation

Google Translate



➤ Note: this course only covers the basic and fundamental techniques of supervised learning (which are not enough for solving hard vision or NLP problems.)

# Reinforcement Learning

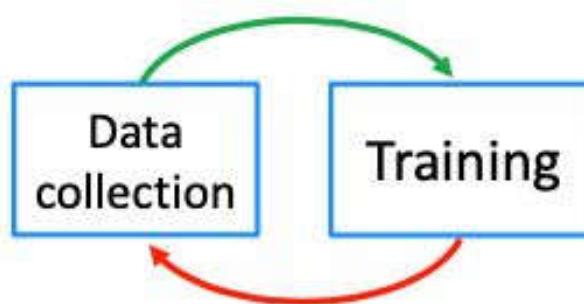


[Take Advanced AI to learn more about POMDP and reinforcement learning.](#)

# Reinforcement Learning

- The algorithm can collect data interactively

Try the strategy and collect feedbacks



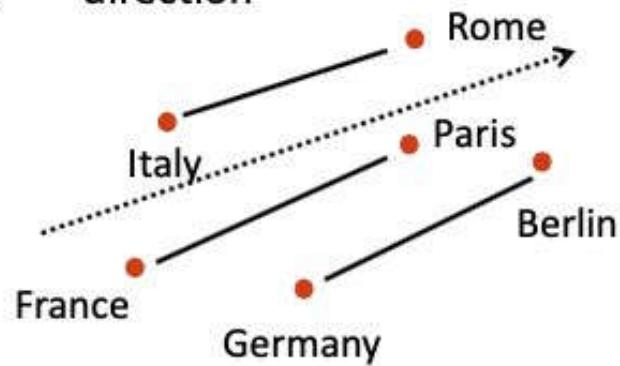
Improve the strategy based on the feedbacks

Many open challenges remain for tackling real-world problems.

# Word “Embeddings”

Represent words by vectors

- word       $\xrightarrow{\text{encode}}$       vector
- relation     $\xrightarrow{\text{encode}}$       direction



“Unlabeled” data?  
Self-supervision?

Word2vec [Mikolov et al'13]  
GloVe [Pennington et al'14]

# Senticap: Describing Images with Sentiments

	a	b	c	d
1				
	a great variety of fresh fruits and vegetables	a cuddly cat is laying on a bed	an ugly car is parked in front of an abandoned building	a lonely train pulling into a train station
2				
	a delicious piece of cake sitting on top of a white plate	a clock on the side of a beautiful building	a man in a stupid hat is riding on the back of a crazy horse	a silly cat standing in front of a dirty wall
3				
	a close up of a kite flying in a beautiful sky	a happy man flying through the air while riding a skateboard	a herd of cows grazing in a field of dead grass	a dead man doing a clever trick on a skateboard at a skate park

<http://cm.cecs.anu.edu.au/post/senticap/>

# SemStyle: Learning to Caption from Romantic Novels

success cases

(a)



[Descriptive] A woman walking with an umbrella in the rain.

[Story-like] The woman stepped underneath her umbrella and walked in the rain.

(b)



A forest that has a large tree in it.

Forest, tall, and thick trees.

failure case

(c)



A juicer is poured into a glass of juice.

I'll be in the juicer with a glass of orange juice.

Descriptive (blue) and story-like (dark red) image captions created by the SemStyle system. The story-like captions in example (a) is written as a sequence of actions, rather than a static scene description; (b) introduces a new adjective and uses a poetic sentence structure. Styled caption (c) is my favorite failure case -- it violates common sense but triggers readers' imagination.

<http://cm.cecs.anu.edu.au/post/semstyle/>  
[http://cm.cecs.anu.edu.au/post/transform\\_and\\_tell/](http://cm.cecs.anu.edu.au/post/transform_and_tell/)



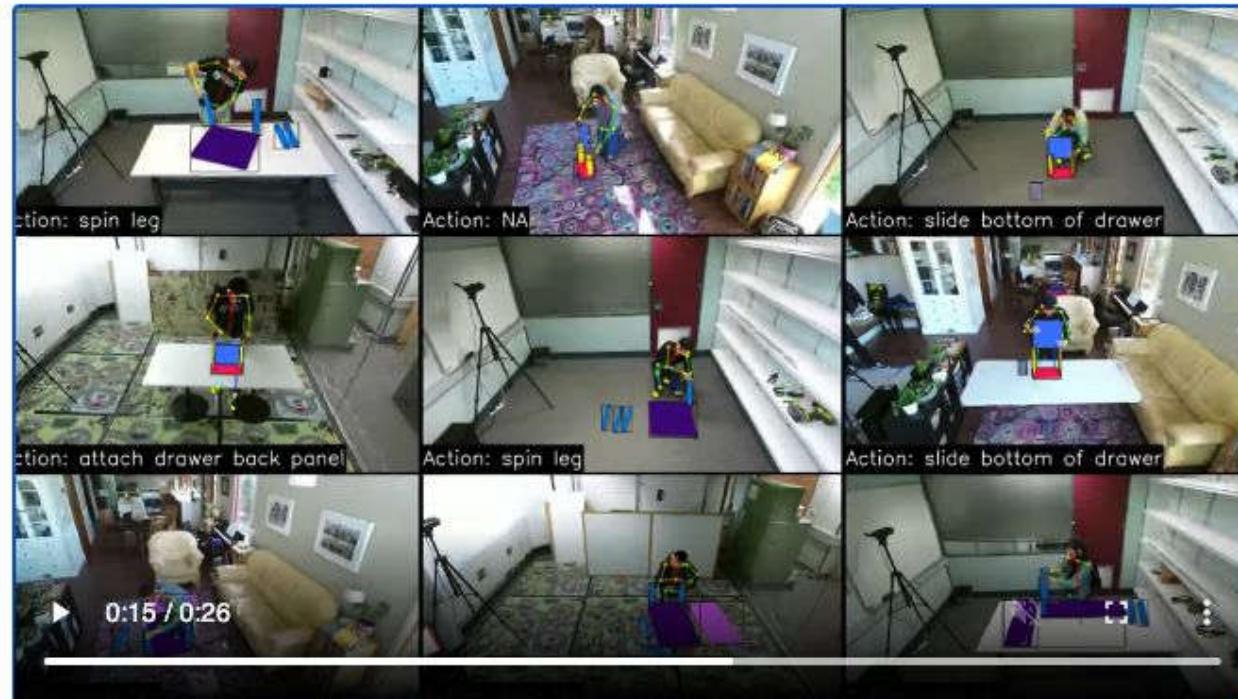
Estimating Attention Flow in Online Video Networks, Wu et al, ACM CSCW 2019

<https://attentionflow.ml/#/overview/video/rYEDA3JcQgw>

Radflow: A Recurrent, Aggregated, and Decomposable Model for Networks of Time Series, Tran et al, The Web Conference 2021

AttentionFlow: Visualising Influence in Networks of Time Series, Shin, Tran, Wu, Lyall, Wang, Mathews, Xie, WSDM Demo, 2021

The Australian National University (ANU)    Australian Centre for Robotic Vision (ACRV)

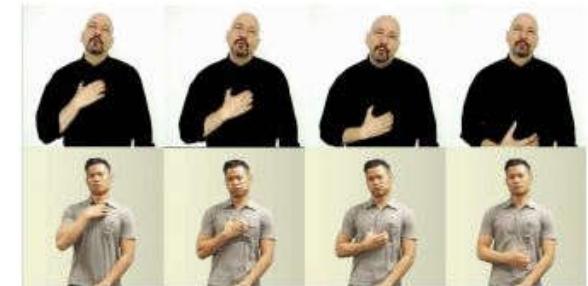


<https://ikeaasm.github.io/>

# Sign language recognition and translation

ARC Discovery Project: Hondong Li, Xin Xu

This project aims to develop an automatic two-way machine-translation system between Auslan (Australian Sign Language) and English by researching and leveraging advanced computer vision and machine learning technology. The project expects to advance research in AI technology on topics including visual recognition, language processing and deep learning. This will boost Australia's national research capacity and global competitiveness. Expected outcomes of this project will help to break the communication barriers between the Deaf and hearing population. This should provide significant benefits to Deaf communities through enhanced communication and improved quality-of-life, leading to a fair, more inclusive and resilient Australian society.



(a) The verb "**Wish**" (top) and the adjective "**hungry**" (bottom) correspond to the same sign.



(b) The same sign represents different words "**Rice**" (top) and "**soup**" (bottom).



(c) Signers perform "**Scream**" with different hand positions and amplitude of hand movements.

Figure 2: Ambiguity and variations of Signing. (a, b) shows linguistic ambiguity in ASL. (c) shows signing variations of different signers.

# Bayesian Joint Inversions for the Exploration of Earth Resources

Alistair Reid<sup>1</sup>, Simon O'Callaghan<sup>1</sup>, Edwin V. Bonilla<sup>1</sup>, Lachlan McCalman<sup>1</sup>, Tim Rawling<sup>2</sup> and Fabio Ramos<sup>3</sup>

1. NICTA, 2. University of Melbourne, 3. University of Sydney

{ alistair.reid, simon.ocallaghan, edwin.bonilla, lachlan.mccalman }@nicta.com.au  
tim.rawling@unimelb.edu.au, f.ramos@acfr.usyd.edu.au

In a geological inversion problem, properties such as temperature, conductivity, density, magnetic susceptibility and permeability are inferred from related observations such as gravity, magnetics and seismic reflexion. ... In this paper we formulate geophysical inversion as a machine learning problem, and propose an approach based on Gaussian processes regression that naturally provides both a predictive distribution over the inverted quantities and a principled method to fuse different types of observations. We apply our method to a real dataset from South Australia containing gravity and drill-hole data with the goal of characterizing rock densities for geothermal target exploration, and also to simulated validation data involving gravity, drill-hole and magnetic observations.

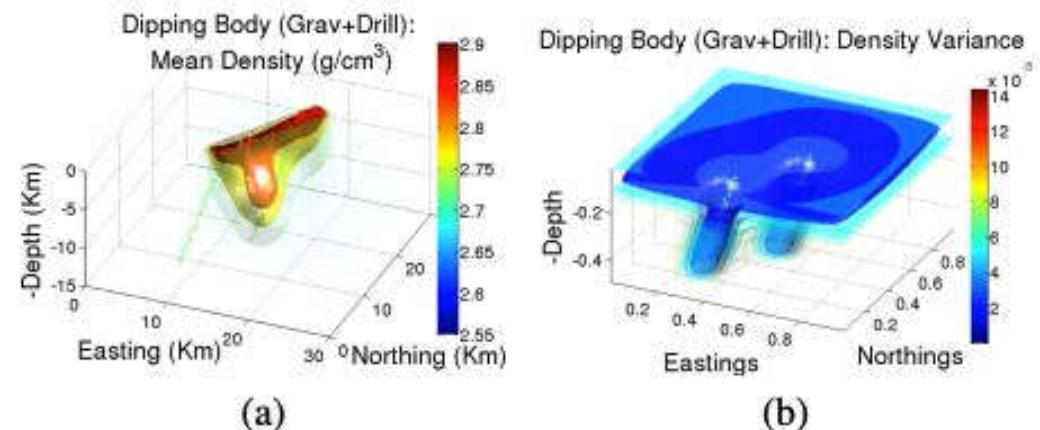


Figure 5: Outputs of the Gaussian process inversion algorithm after fusing gravity and drill observations of the dipping body.

# ML + X at ANU

- Machine learning for quantum computing and gravitational physics
- Machine learning for astronomy
- Machine learning for synthesizing materials
- Machine learning for climate change, natural resources and disasters
- Machine learning for plants and agriculture
- Machine learning for monitoring insect behavior
- Active learning for experimental design in biology
- Machine learning for enhancing in-vitro microscopy
- ... ...

# GPT-3, very large models, and all that

"an armchair in the shape of an avocado"



<https://openai.com/blog/dall-e/>

# ML algorithms can appear biased and racist



<https://archive.ieet.org/articles/sweeney.html>

## Google apologizes for mis-tagging photos of African Americans

BY AMANDA SCHUPAK  
JULY 1, 2015 / 5:04 PM / CBS NEWS



Google was quick to respond over the weekend to a user after he tweeted that the new Google Photos app had mis-categorized a photo of him and his friend in an unfortunate and offensive way.

Jacky Alciné, a Brooklyn computer programmer of Haitian descent, [tweeted a screenshot](#) of Google's new Photos app showing that it had grouped pictures of him and a black female friend under the heading "Gorillas."

"Google Photos, y'all f\*\*\*d up. My friend's not a gorilla," Alciné wrote.

**WIRED** BACKCHANNEL BUSINESS CULTURE GEAR IDEAS SCIENCE SECURITY

SIGN IN | SUBSCRIBE |

## When It Comes to Gorillas, Google Photos Remains Blind

Google promised a fix after its photo-categorization software labeled black people as gorillas in 2015. More than two years later, it hasn't found one.



# Extracting Training Data from Large Language Models

Nicholas Carlini<sup>1</sup>

Florian Tramèr<sup>2</sup>

Eric Wallace<sup>3</sup>

Matthew Jagielski<sup>4</sup>

Ariel Herbert-Voss<sup>5,6</sup>

Katherine Lee<sup>1</sup>

Adam Roberts<sup>1</sup>

Tom Brown<sup>5</sup>

Dawn Song<sup>3</sup>

Úlfar Erlingsson<sup>7</sup>

Alina Oprea<sup>4</sup>

Colin Raffel<sup>1</sup>

*Google* <sup>2</sup>*Stanford* <sup>3</sup>*UC Berkeley* <sup>4</sup>*Northeastern University* <sup>5</sup>*OpenAI* <sup>6</sup>*Harvard* <sup>7</sup>*Apple*

**Memorized Usernames.** There are BPE tokens for several usernames of individual people. For example, the Twitter handle for Donald Trump, `realDonaldTrump`, is represented by a single token in the encoding dictionary. However, this is not an instance of Eidetic memorization, as this token is contained in thousands of webpages. In contrast, through manual review of the BPEs, we identify three BPE tokens that correspond to usernames of individual users on Reddit.<sup>13</sup> These three tokens are otherwise unique on the Internet: Google searches yield 24, 29 and 34 results for each of these usernames; all results correspond to content related to these users.

Similarly, we identify one token that corresponds to the GitHub repository name of a particular user. This repository has only two “stars” on GitHub, and there are 40 results for this phrase contained on Google.

**Memorized Leaked Podesta Emails from WikiLeaks.**

We identify several memorized URLs that originated from the leaked Podesta Emails available on WikiLeaks.<sup>14</sup> There is only a single training document that contains these memorized URLs. Due to the nature of email, the text of one message is often included in subsequent replies to this email. As a result, a URL that is used (intentionally) only once can be included in the dataset tens of times due to the replies.

---

<sup>14</sup>[https://en.wikipedia.org/wiki/Podesta\\_emails](https://en.wikipedia.org/wiki/Podesta_emails)

## AUTOMATING GOVERNANCE

Data and AI are increasingly used—by states and digital platforms—to exercise power over us. What does it mean for that power to be used justly and legitimately? How can we design socio-technical systems that enable legitimate AI?

## PERSONALISATION

The most sophisticated AI systems in the world ensure that your every moment online is tailored to you: personalised media, news, ads, prices. What are the consequences for democratic societies? Can we achieve serendipitous recommendations without creating new and troubling power relations?

## ALGORITHMIC ETHICS

AI systems can increasingly make significant state changes without intervening human influence. We need to design these systems to take our values into account. But which values? And how can we translate them into algorithmic form?

## HUMAN-AI INTERACTION

We fall into predictable errors when we interact with AI; and over time, those interactions change us. What cognitive and other biases should designers of AI systems account for? And how do we avoid 'moral outsourcing' in favour of AI systems that make us better moral agents?

 PLAY ALL

### Humanising Machine Intelligence

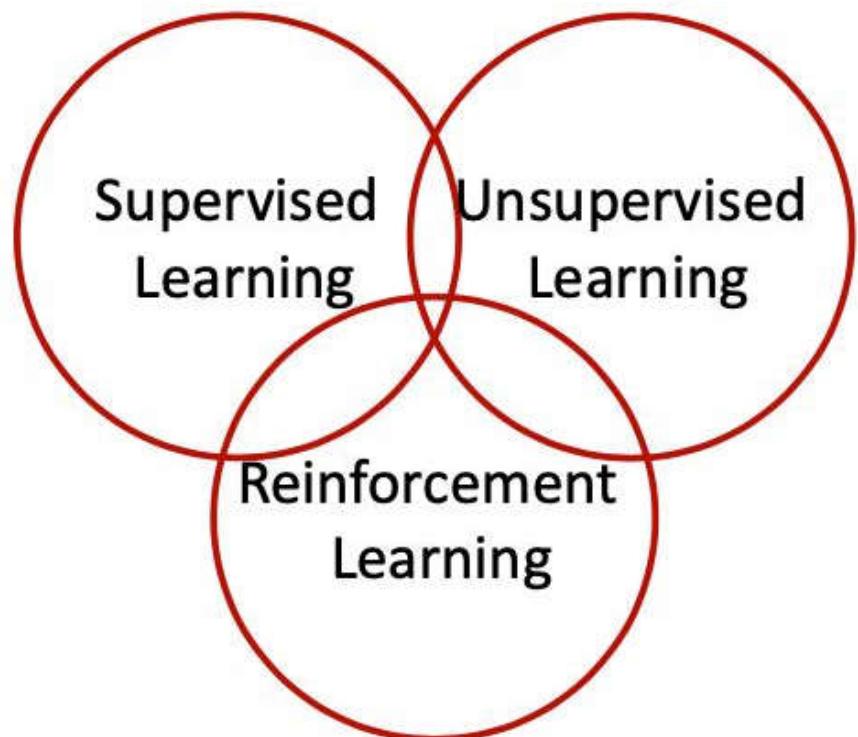
20 videos • 714 views • Last updated on Nov 30, 2020

[Data, AI and Society - The Importance of Modelling Data Missingness in Algorithmic Fairness](#)  
ANU TV

[Data, AI and Society - Resolving Algorithmic Fairness](#)  
ANU TV

[Data, AI and Society - Roles for Computing in Social Justice](#)  
ANU TV

Recall: simplistic machine learning taxonomy



## Example other machine learning tasks that this class won't cover

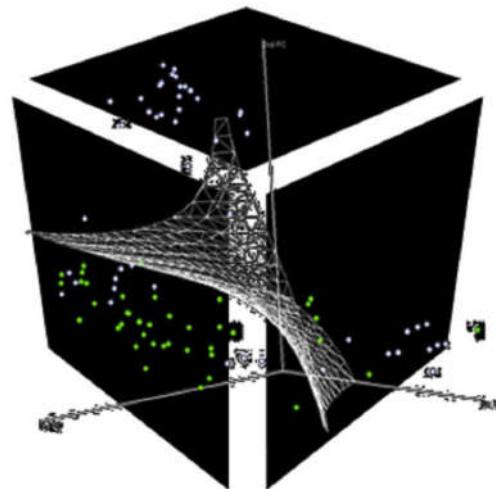
- Active Learning
  - The algorithm may choose which data  $x_i \in \mathcal{X}$  to select next when building the model.
  - The order of the data is **actively** chosen by the algorithm at run-time.
- Transduction
  - The algorithms is allowed to use the test data (but of course not labels!) when building a model.
- Estimation with missing variables.
- Co-training with two different but related data sets.
- ... and others.

## topic areas

learning theory, asymptotic analysis, classification, clustering, causality inference, game theory  
...

## models

random forest, decision trees, C4.5, logistic regression, linear regression, ridge regression, lasso, naive bayes, kmeans, spectral clustering, t-SNE, bandit algorithms, topic models, support vector machines, kernel methods, matrix and tensor factorisation, MDP, neural network, ResNet, CovNet, RNN, LSTM, learning to rank, factorisation machines, CCA, ICA, point processes, CRF ...



## methods

optimization: combinatorial, convex, non-convex, submodular; belief propagation, variational inference, stochastic gradient descent, adam, density estimation, hyperparameter selection, distributed inference, graph cut ...

## problem settings

supervised learning, unsupervised learning, semi-supervised learning, online learning, transfer learning, multitask learning, life-long learning, zero-shot learning, multi-instance learning, machine teaching, meta learning, active learning, reinforcement learning, structured prediction ...

## problem domains

computational biology, finance, computer vision, surveillance, traffic monitoring, natural language processing, geo-physics and geo-chemistry, question-answering, information retrieval, crowd-sourcing, music information retrieval, prediction markets, computational social science, knowledge extraction, neural science, astronomy, ...

## What will SML NOT cover

- Reinforcement learning
- Models for time series and graphs
- Recommender systems
- Optimisation, convex or not
- ML systems, distributed/federated learning
- 
- Make you an expert in TRENDY\_DEEPMLEARNING\_PACKAGE
- ...

## *On the Way to Learning (in Indonesia)*



A good time to learn ML

- Job prospects
- Intellectual reward
- Blend of CS and math
- In-depth understanding of inside the blackbox, so as to improve its use and social welfare

Tips (see tutor videos)

- Use different course resources
- Program it to “really understand” something
- Start early on the assessment items!



## What to do now:

- Register on Piazza and GradeScope
- Register for a tutorial slot.
- Work on Assignment 0

Questions?

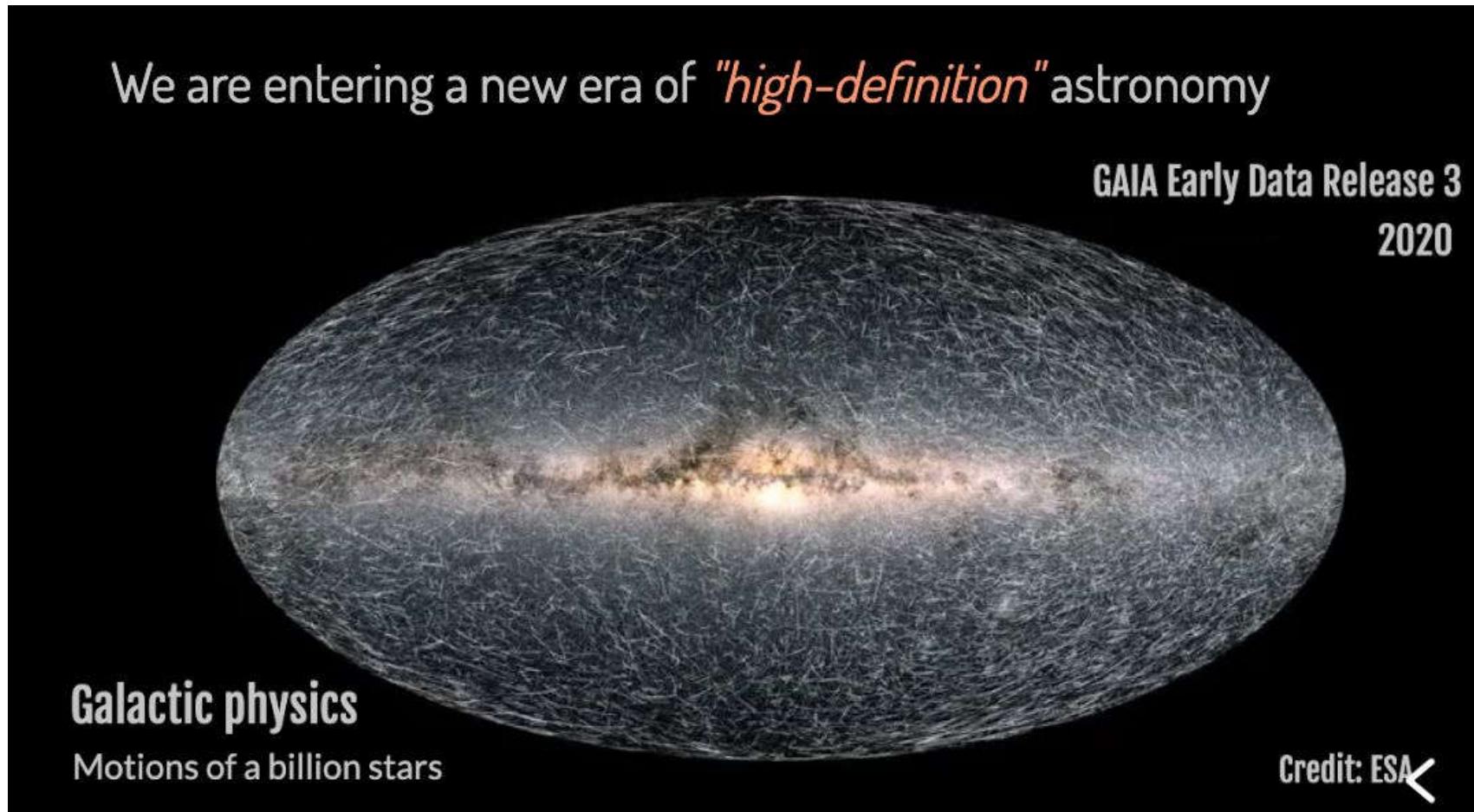
# Solar Power Prediction

- Photovoltaics now very close to grid electricity in price
- Distributed system of generators -- Energy market
- Great Machine Learning Problem: Predict the solar energy output (variability primarily due to clouds) for Australia
- Pilot project in Canberra : Use cheap cameras to take 360° sky photos in several location.
- Learn to predict 3-D model of cloud movement.
- Learn orientation and efficiency of solar panels for each house from time series of energy output.
- Predict output of each solar panel for 15 min to 1 hour from current snapshots.



Photo from wikipedia

# ML for Astronomy



Slide by Yuan-Sen Ting <http://sns.ias.edu/~ting/>

**TODO:** A bit more detail w.r.t IML and other related courses

STAT3040 Statistical Learning

STAT3017 Big Data Statistics (random matrix theory, Dale Roberts)

Hold on ... SML lecture will be starting soon.

<https://xkcd.com/605/>

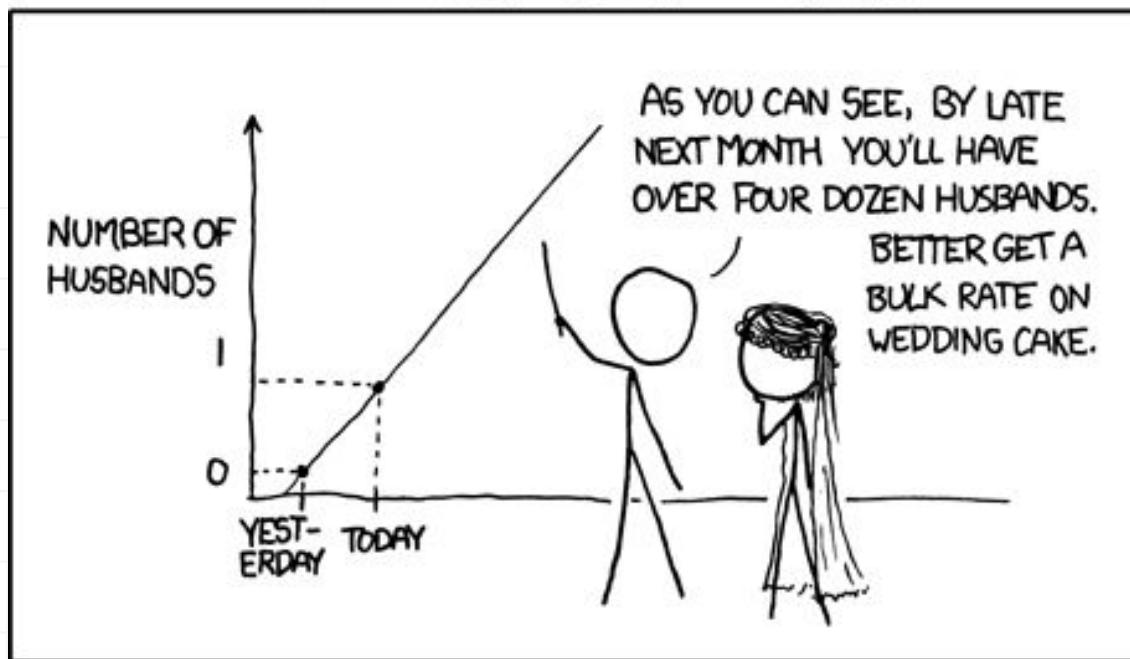
Anisvement 2022-02-23

① Mon lecture moved

12 - 2 pm

② Assignment 1

releasing SOON



On the topic of extrapolation and train-test mismatch, see

<https://www.youtube.com/watch?v=es6p6NuxOnY> and [http://ciml.info/dl/v0\\_99/ciml-v0\\_99-ch08.pdf](http://ciml.info/dl/v0_99/ciml-v0_99-ch08.pdf)

# Plan for Today

ML 101:

Polynomial curve fitting: model, loss/error function, over-fitting, regularisation

多项式拟合

Model selection

Probabilities: sum rule, product rule, Bayes theorem

Gaussians - 1D, maximum likelihood estimates (MLE), bias-variance

→ and how this helps curve-fitting

Gaussians (multidimensional)

various matrix identities, geometric intuitions

Bernoulli, Binomial, Exponential family distributions - will be in assignment 1

Review: probabilities, derivatives and finding stationary points, eigenvalues and eigenvectors

# about the book

TODAY

LATER LECTURES

## 1 Introduction

1.1	Example: Polynomial Curve Fitting . . . . .
1.2	Probability Theory . . . . .
1.2.1	Probability densities . . . . .
1.2.2	Expectations and covariances . . . . .
1.2.3	Bayesian probabilities . . . . .
1.2.4	The Gaussian distribution . . . . .
1.2.5	Curve fitting re-visited . . . . .
1.2.6	Bayesian curve fitting . . . . .
1.3	Model Selection . . . . .
1.4	The Curse of Dimensionality . . . . .
1.5	Decision Theory . . . . .
1.5.1	Minimizing the misclassification rate . . . . .
1.5.2	Minimizing the expected loss . . . . .
1.5.3	The reject option . . . . .
1.5.4	Inference and decision . . . . .
1.5.5	Loss functions for regression . . . . .
1.6	Information Theory . . . . .
1.6.1	Relative entropy and mutual information . . . . .
	Exercises . . . . .

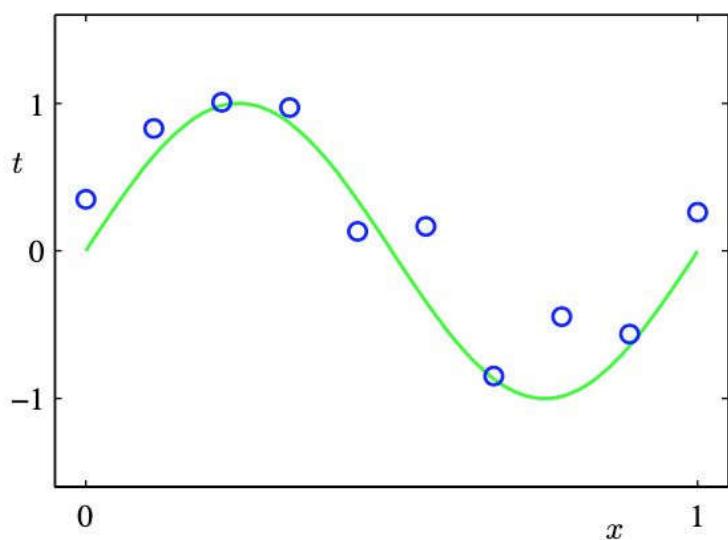
## 2 Probability Distributions

2.1	Binary Variables . . . . .
2.1.1	The beta distribution . . . . .
2.2	Multinomial Variables . . . . .
2.2.1	The Dirichlet distribution . . . . .
2.3	The Gaussian Distribution . . . . .
2.3.1	Conditional Gaussian distributions . . . . .
2.3.2	Marginal Gaussian distributions . . . . .
2.3.3	Bayes' theorem for Gaussian variables . . . . .
2.3.4	Maximum likelihood for the Gaussian . . . . .
2.3.5	Sequential estimation . . . . .
2.3.6	Bayesian inference for the Gaussian . . . . .
2.3.7	Student's t-distribution . . . . .
2.3.8	Periodic variables . . . . .
2.3.9	Mixtures of Gaussians . . . . .
2.4	The Exponential Family . . . . .
2.4.1	Maximum likelihood and sufficient statistics . . . . .
2.4.2	Conjugate priors . . . . .
2.4.3	Noninformative priors . . . . .
2.5	Nonparametric Methods . . . . .
2.5.1	Kernel density estimators . . . . .
2.5.2	Nearest-neighbour methods . . . . .

Tom Mitchell (1998): a computer program is said to learn from **experience E** with respect to some class of tasks **T** and performance measure **P**, if its performance at tasks in T, as measured by P, improves with experience E.

some artificial data created from the function

$$\sin(2\pi x) + \text{random noise} \quad x = 0, \dots, 1$$



The machine sees:

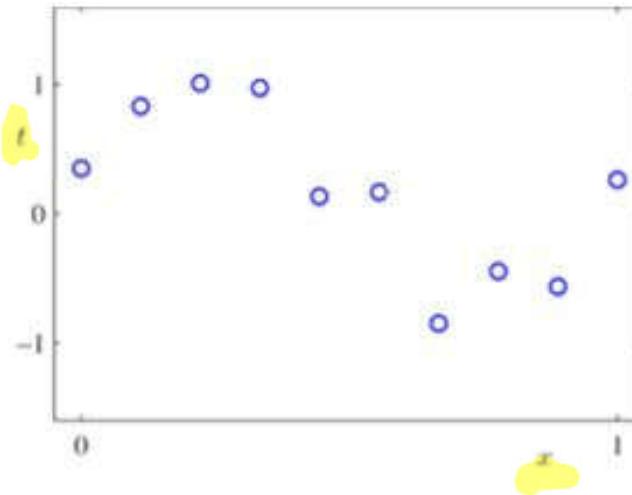
$$N = 10$$

$$\mathbf{x} \equiv (x_1, \dots, x_N)^T$$

$$\mathbf{t} \equiv (t_1, \dots, t_N)^T$$

$$x_i \in \mathbb{R} \quad i = 1, \dots, N$$

$$t_i \in \mathbb{R} \quad i = 1, \dots, N$$



Make a guess, M-th order polynomials

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \dots + w_M x^M = \sum_{j=0}^M w_j x^j \quad (1.1)$$

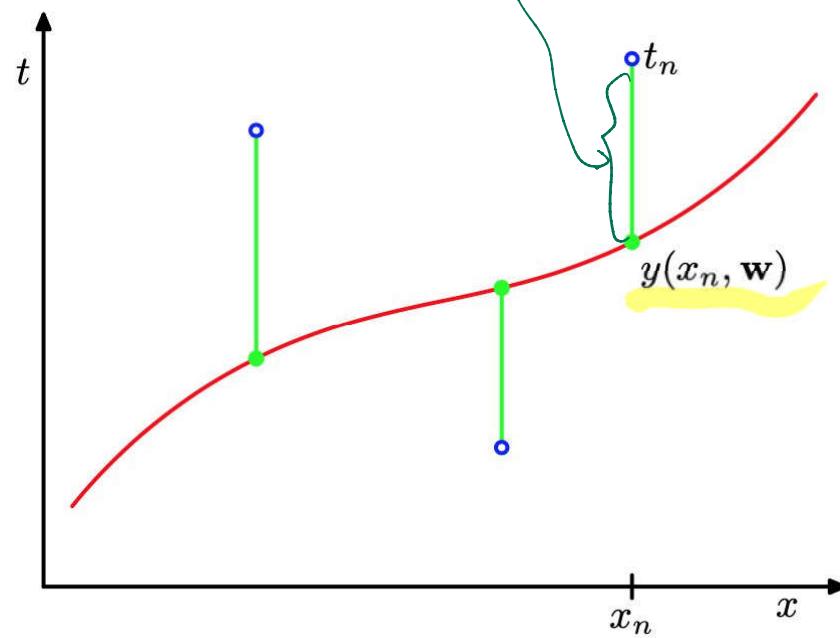
# What is a good “fit”

Performance.

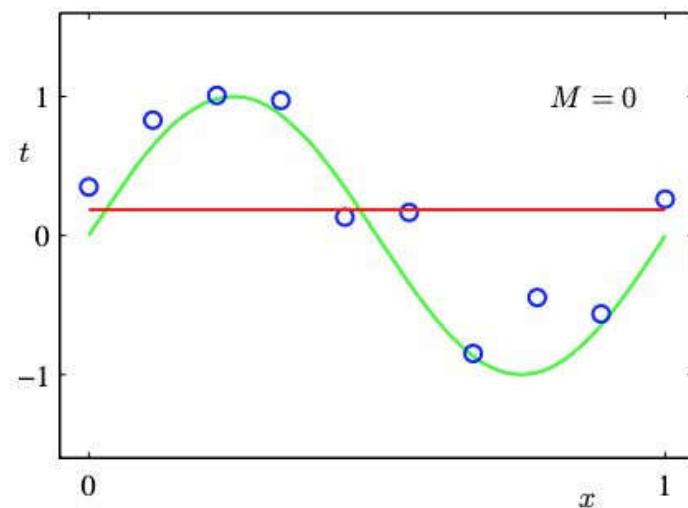
$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2 \quad (1.2)$$

平方差

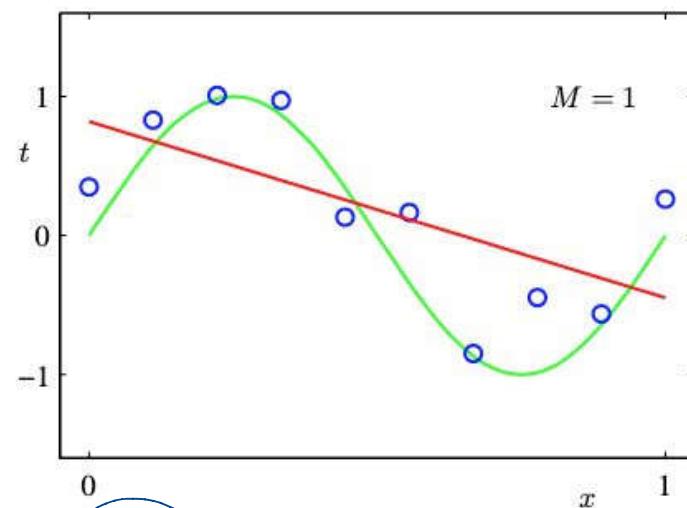
**Figure 1.3** The error function (1.2) corresponds to (one half of) the sum of the squares of the displacements (shown by the vertical green bars) of each data point from the function  $y(x, \mathbf{w})$ .



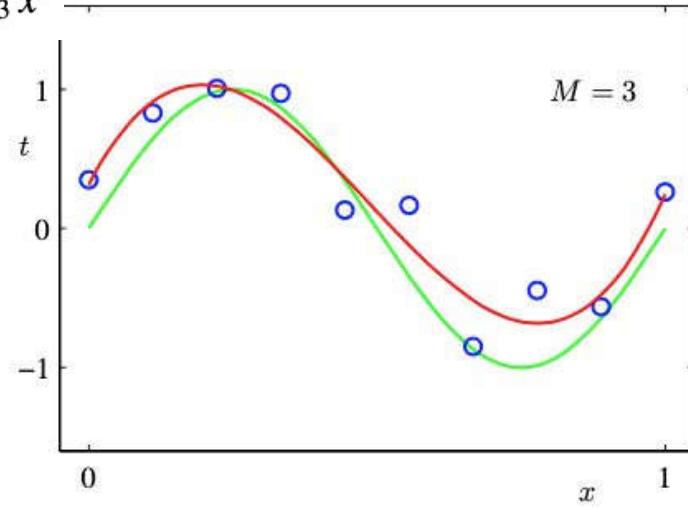
$$y(x, \mathbf{w}) = w_0$$



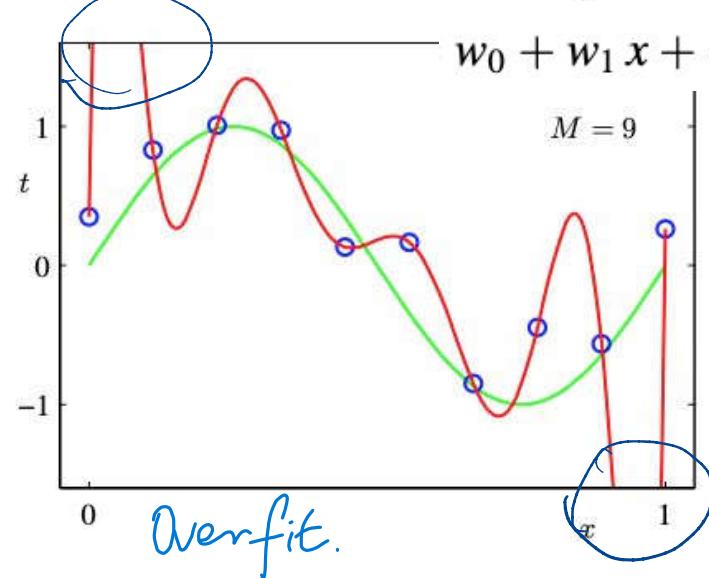
$$w_0 + w_1 x$$



$$w_0 + w_1 x + w_2 x^2 + w_3 x^3$$



$$w_0 + w_1 x + \cdots + w_8 x^8 + w_9 x^9$$



**Figure 1.4** Plots of polynomials having various orders  $M$ , shown as red curves, fitted to the data set shown in Figure 1.2.

# Test error and learning curves

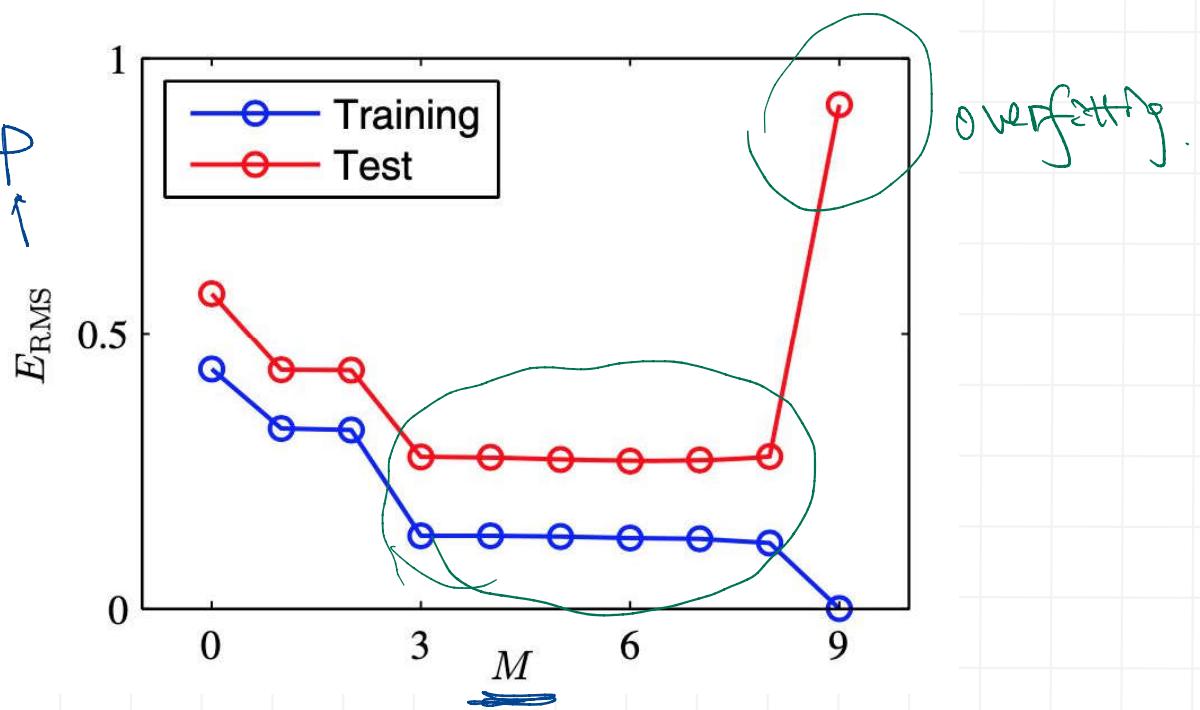
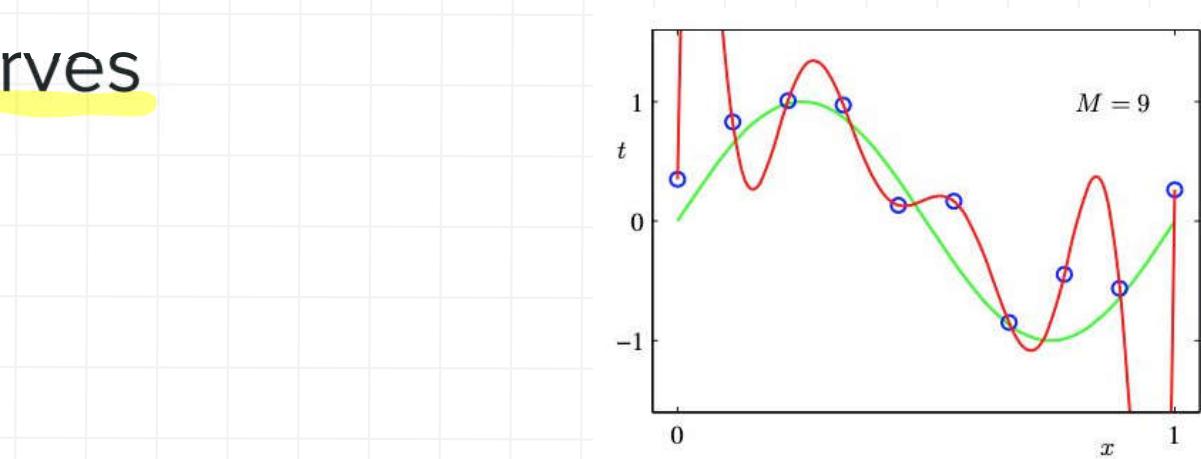
Training set: 10 points

Separate test set of 100 points

**Figure 1.5** Graphs of the root-mean-square error, defined by (1.3), evaluated on the training set and on an independent test set for various values of  $M$ .

拟合误差

$$E_{\text{RMS}} = \sqrt{2E(\mathbf{w}^*)/N}$$



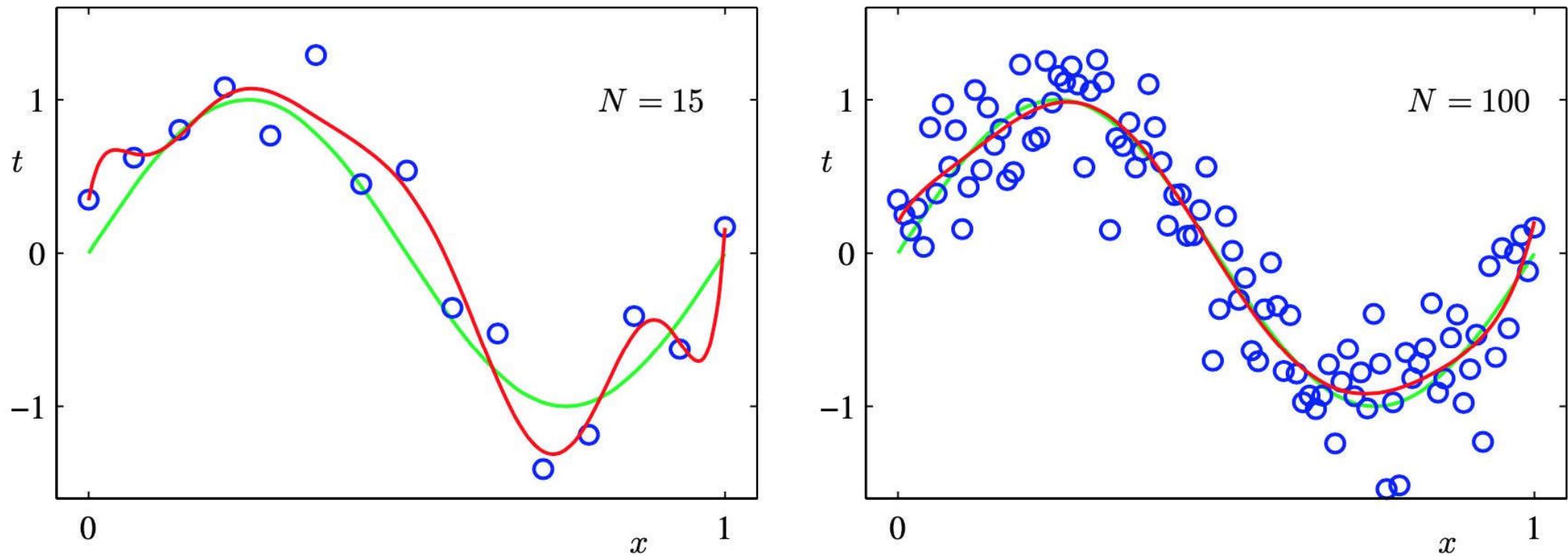
why?

Expansion of  $\sin(x)$  contains terms of all orders

**Table 1.1** Table of the coefficients  $w^*$  for polynomials of various order. Observe how the typical magnitude of the coefficients increases dramatically as the order of the polynomial increases.

	$M = 0$	$M = 1$	$M = 6$	$M = 9$
$w_0^*$	0.19	0.82	0.31	0.35
$w_1^*$		-1.27	7.99	232.37
$w_2^*$			-25.43	-5321.83
$w_3^*$				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

## Cure 1: More data :)



**Figure 1.6** Plots of the solutions obtained by minimizing the sum-of-squares error function using the  $M = 9$  polynomial for  $N = 15$  data points (left plot) and  $N = 100$  data points (right plot). We see that increasing the size of the data set reduces the over-fitting problem.

## Cure 2: regularisation

Minimize regularised error function

$$\tilde{E}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2 \quad (1.4)$$

正则化误差函数.

正则项.

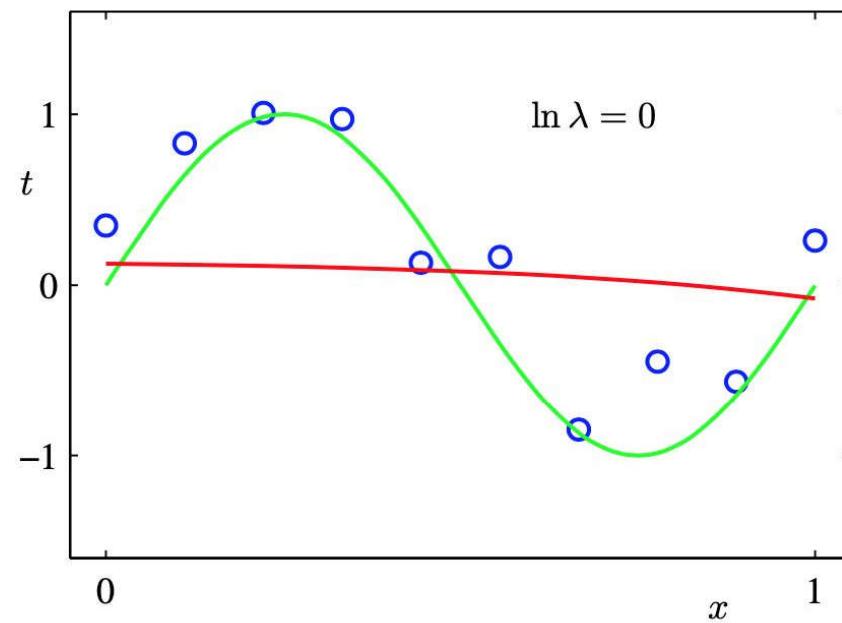
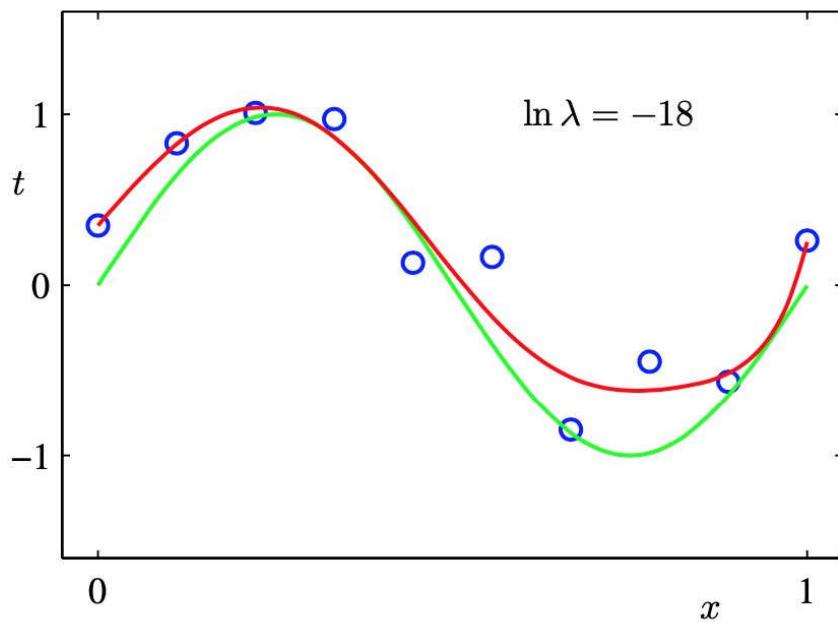
param.

$\frac{1}{2N}$

hyper-parameter.

(more in Bayesian regression next week)

$$\tilde{E}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2 \quad (1.4)$$

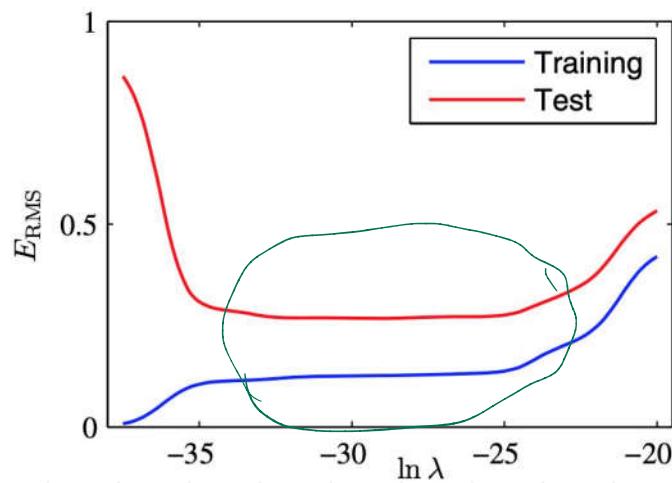


**Figure 1.7** Plots of  $M = 9$  polynomials fitted to the data set shown in Figure 1.2 using the regularized error function (1.4) for two values of the regularization parameter  $\lambda$  corresponding to  $\ln \lambda = -18$  and  $\ln \lambda = 0$ . The case of no regularizer, i.e.,  $\lambda = 0$ , corresponding to  $\ln \lambda = -\infty$ , is shown at the bottom right of Figure 1.4.

**Table 1.2** Table of the coefficients  $w^*$  for  $M = 9$  polynomials with various values for the regularization parameter  $\lambda$ . Note that  $\ln \lambda = -\infty$  corresponds to a model with no regularization, i.e., to the graph at the bottom right in Figure 1.4. We see that, as the value of  $\lambda$  increases, the typical magnitude of the coefficients gets smaller.

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^*$	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

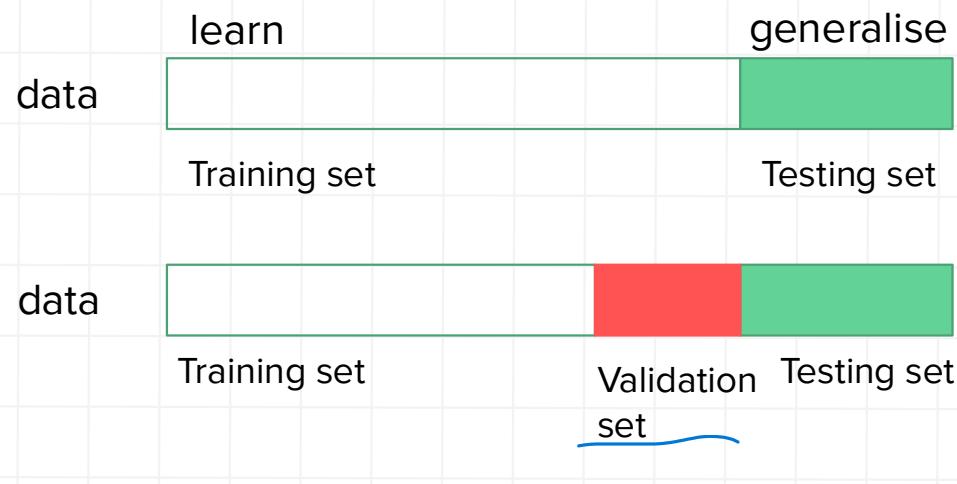
Graph of the root-mean-square error (1.3) versus  $\ln \lambda$  for the  $M = 9$  polynomial.



# Model selection (an empirical view)

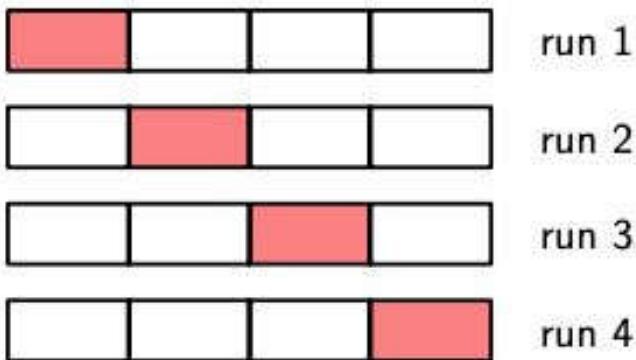
Minimizing square error / maximizing data likelihood can be a poor indication of performance on new data (generalisation) – Cause: overfitting

In the curve-fitting example: the order of the polynomial controls the number of free parameters in the model and thereby governs the model complexity.

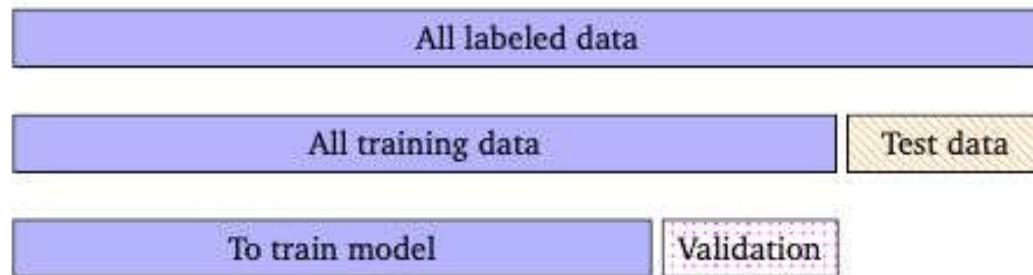


How reliable are the estimates  
for validation and generalisation  
performance?

**Figure 1.18** The technique of  $S$ -fold cross-validation, illustrated here for the case of  $S = 4$ , involves taking the available data and partitioning it into  $S$  groups (in the simplest case these are of equal size). Then  $S - 1$  of the groups are used to train a set of models that are then evaluated on the remaining group. This procedure is then repeated for all  $S$  possible choices for the held-out group, indicated here by the red blocks, and the performance scores from the  $S$  runs are then averaged.



**Figure 8.13** Nested cross-validation. We perform two levels of  $K$ -fold cross-validation.



[source: MML book]

# What we did so far

ML 101:

Polynomial curve fitting: model, loss/error function, over-fitting, regularisation

Model selection

Probabilities: sum rule, product rule, Bayes theorem

Gaussians - 1D, maximum likelihood estimates (MLE), bias-variance

→ and how this helps curve-fitting

Gaussians (multidimensional)

various matrix identities, geometric intuitions

Bernoulli, Binomial, Exponential family distributions

Review: probabilities, derivatives and finding stationary points, eigen values and eigen vectors

## The Rules of Probability

sum rule

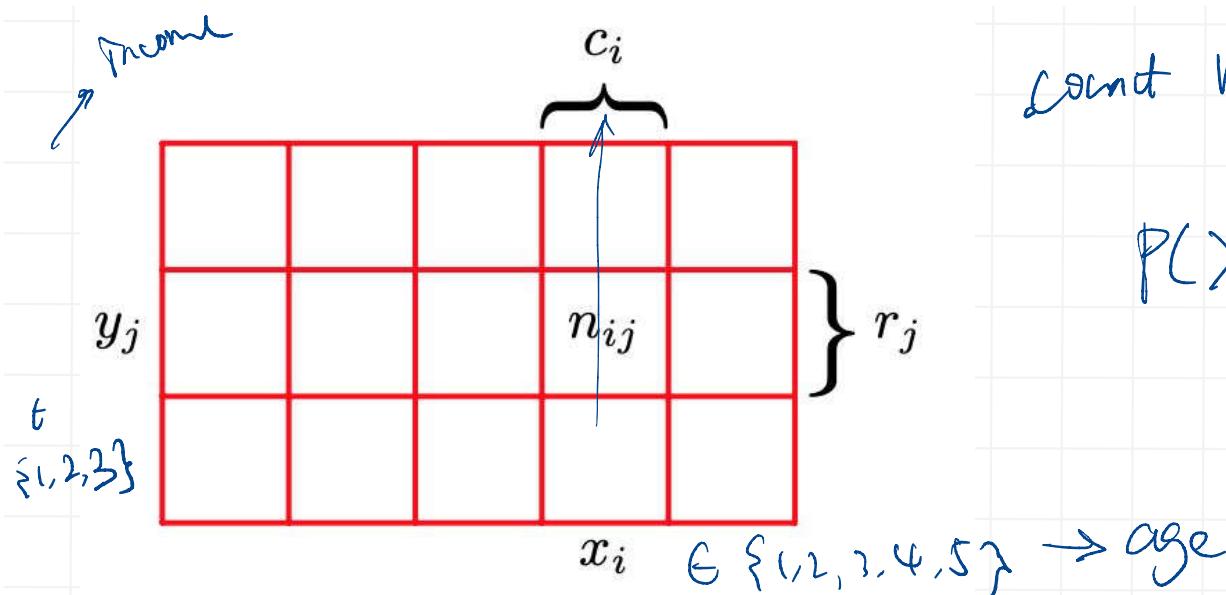
$$p(X) = \sum_Y p(X, Y)$$

(1.10)

product rule

$$p(X, Y) = p(Y|X)p(X).$$

(1.11)



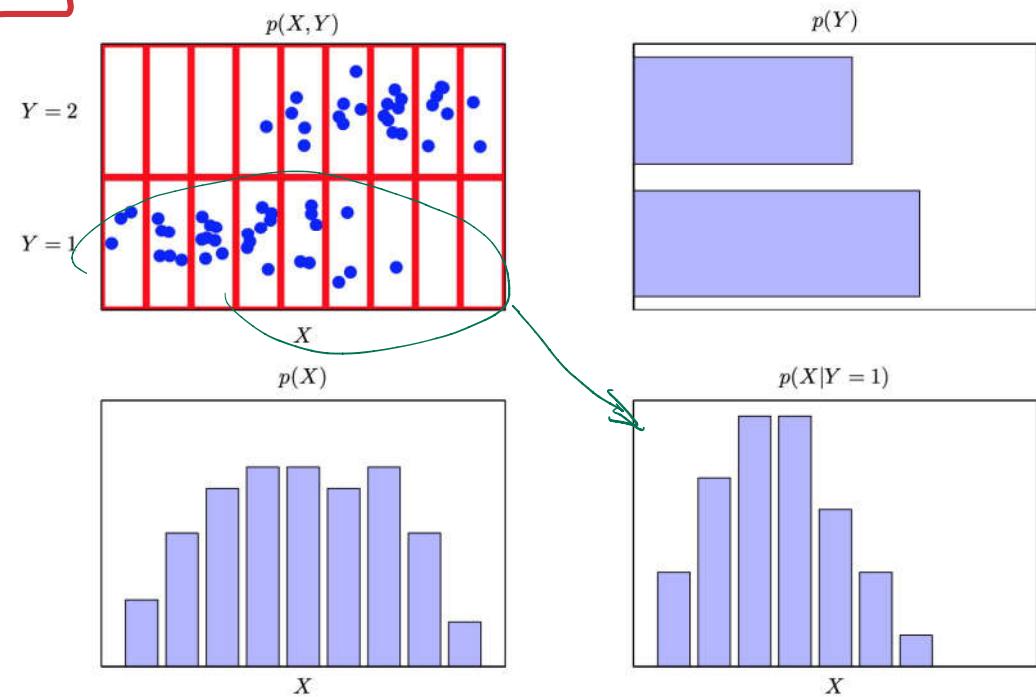
count  $n_{ij}$       N sample.

$$P(X=x_i) = \frac{c_i}{N} = \frac{1}{N} \sum_{j=1}^3 n_{ij}$$

## Bayes Theorem

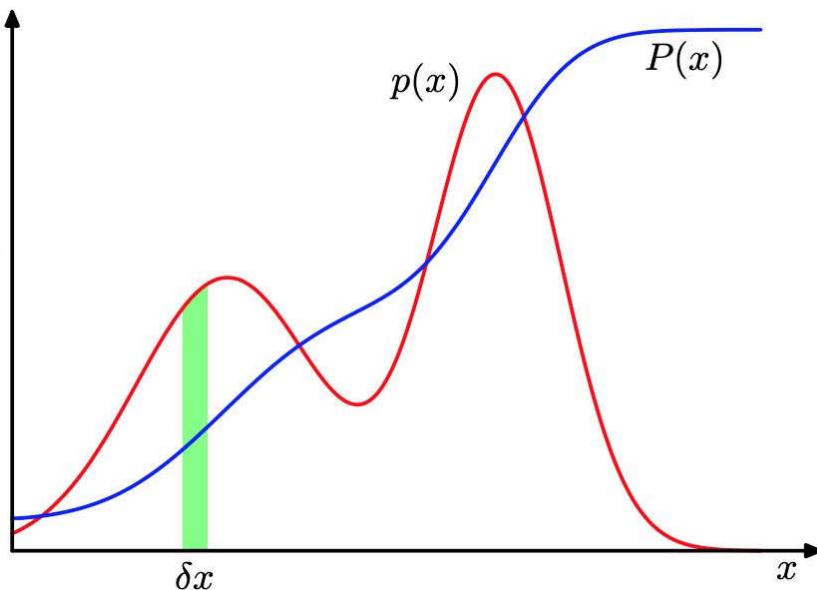
$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}$$

(1.12)



# Continuous random variables

**Figure 1.12** The concept of probability for discrete variables can be extended to that of a probability density  $p(x)$  over a continuous variable  $x$  and is such that the probability of  $x$  lying in the interval  $(x, x + \delta x)$  is given by  $p(x)\delta x$  for  $\delta x \rightarrow 0$ . The probability density can be expressed as the derivative of a cumulative distribution function  $P(x)$ .



$$p(x) = \int p(x, y) dy \quad (1.31)$$

$$p(x, y) = p(y|x)p(x). \quad (1.32)$$

## Bayes Theorem, restated (Sec 1.2.3)

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)} \quad (1.12)$$

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})} \quad (1.43)$$

Annotations for equation (1.43):

- Yellow box around  $p(\mathcal{D}|\mathbf{w})$  labeled "likelihood".
- Yellow box around  $p(\mathbf{w})$  labeled "prior".
- Yellow box around  $\mathcal{D}$  labeled "not random once data is given".
- Green arrow from "weight param" to  $\mathbf{w}$ .
- Green arrow from "data" to  $\{(x_i, t_i), i=1, \dots, N\}$ .

posterior  $\propto$  likelihood  $\times$  prior

# Expectations, variance, covariance

For review

$$\mathbb{E}[f] = \int p(x) f(x) dx. \quad (1.34)$$

Variance → show this yourself

$$\begin{aligned} \text{var}[f] &= \mathbb{E} [(f(x) - \mathbb{E}[f(x)])^2] \\ &\stackrel{x \sim p(x)}{=} \mathbb{E}[f(x)^2] - \mathbb{E}[f(x)]^2. \end{aligned} \quad (1.38) \quad (1.39)$$

to do this

$$\begin{aligned} \text{cov}[x, y] &= \mathbb{E}_{x,y} [\{x - \mathbb{E}[x]\} \{y - \mathbb{E}[y]\}] \\ x, y \sim p(x, y) &= \mathbb{E}_{x,y} [xy] - \mathbb{E}[x]\mathbb{E}[y] \end{aligned} \quad (1.41)$$

what is the expectation taken over? probability p is often implicit.

$$\mathbb{E}_x[f|y] = \sum_x p(x|y) f(x) \quad \rightarrow \text{function of } y \quad (1.37)$$

Question: for a random variable  $x \sim p(x)$ , do  $\mathbb{E}[x]$  and  $\text{var}[x]$  always exist? **No**.

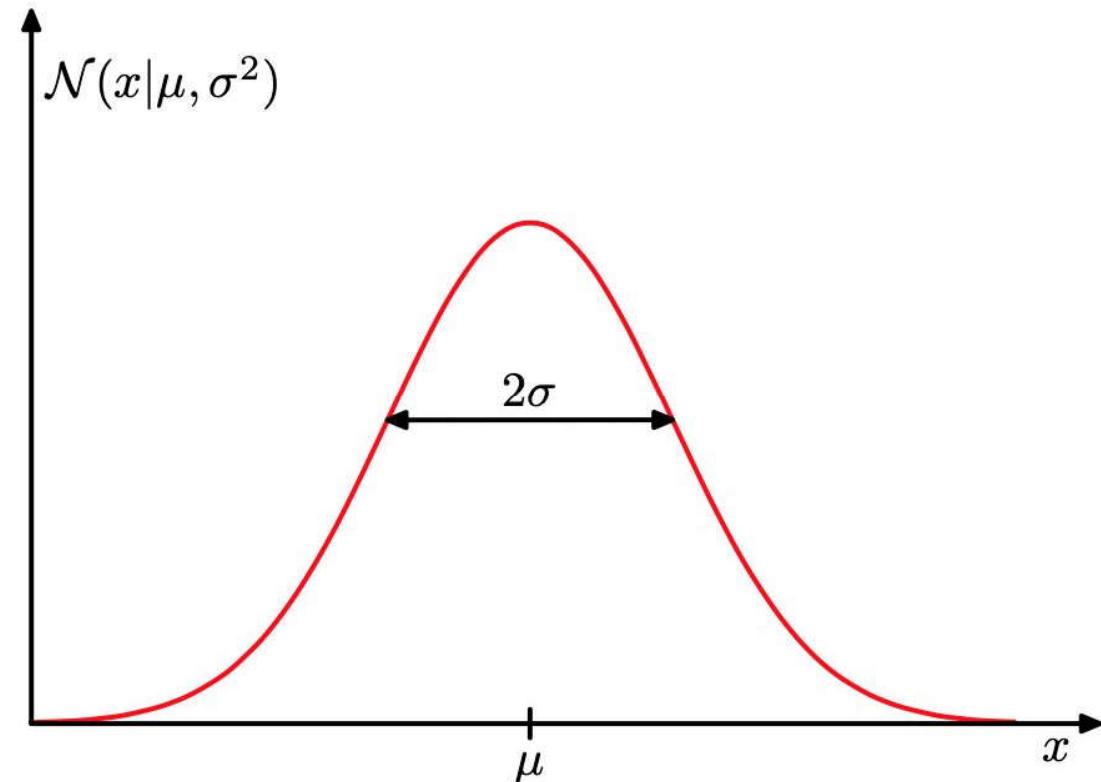
# The Gaussian Distribution

**Figure 1.13** Plot of the univariate Gaussian showing the mean  $\mu$  and the standard deviation  $\sigma$ .

高斯分布:

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp \left\{ -\frac{1}{2\sigma^2}(x - \mu)^2 \right\}$$

normalizing      quadratic term  
=  $\propto e^{-\frac{1}{2}x^2}$



For multi-gaussian:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp \left\{ -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \underline{\boldsymbol{\Sigma}^{-1}} (\mathbf{x} - \boldsymbol{\mu}) \right\}$$

单变量:

## Maximum likelihood for univariate Gaussian

$$\max_{\mu, \sigma^2} p(\mathbf{x}|\mu, \sigma^2) = \prod_{n=1}^N \mathcal{N}(x_n|\mu, \sigma^2)$$

↑ data likelihood  
variables.

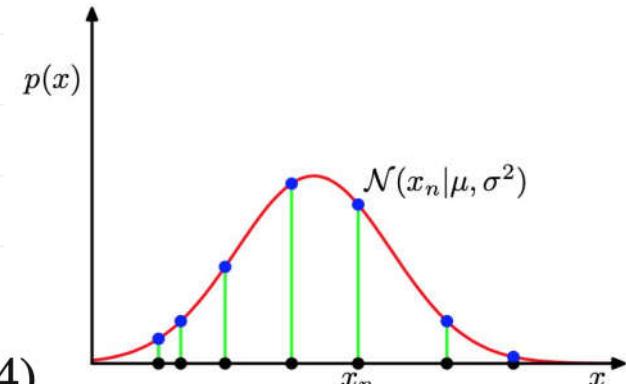
$$L(\mu, \sigma^2) = \ln p(\mathbf{x}|\mu, \sigma^2) = -\frac{1}{2\sigma^2} \sum_{n=1}^N (x_n - \mu)^2 - \frac{N}{2} \ln \sigma^2 - \frac{N}{2} \ln(2\pi). \quad (1.54)$$

$$\mu_{ML} = \frac{\partial L}{\partial \mu} = \cancel{-\frac{1}{2\sigma^2}} \cdot 2 \sum_{n=1}^N (x_n - \mu) \cancel{(-1)} = 0 \Rightarrow \sum_{n=1}^N x_n - N\mu = 0$$

$$\mu_{ML} = \frac{1}{N} \sum_{n=1}^N x_n$$

$$\sigma_{ML}^2 = \frac{1}{N} \sum_{n=1}^N (x_n - \mu_{ML})^2$$

$$\sigma_{ML}^2 = \frac{1}{N} \sum_{n=1}^N (x_n - \mu_{ML})^2$$



Maximum likelihood  $\neq$  unbiased



In statistics, the **bias** (or **bias function**) of an estimator is the difference between this estimator's **expected value** and the **true value of the parameter** being estimated. An estimator or decision rule with zero bias is called **unbiased**. In statistics, "bias" is an **objective** property of an estimator.

"Bias" is not necessarily bad!

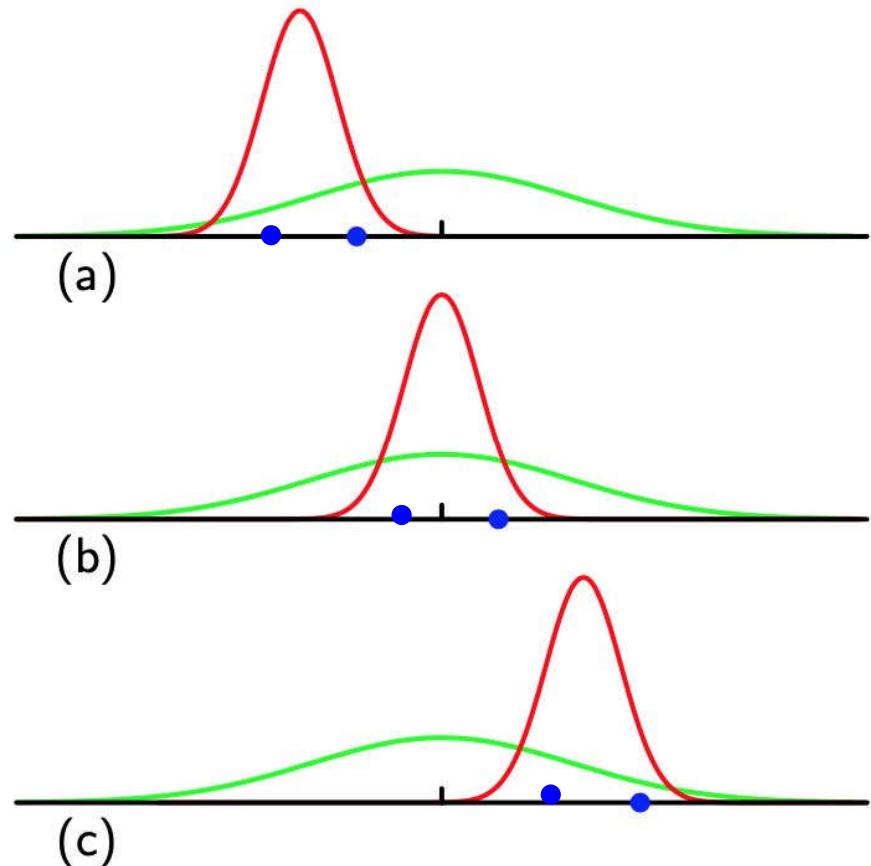
$$\mathbb{E}[\mu_{\text{ML}}] = \mu \quad (1.57)$$

$$\mathbb{E}[\sigma_{\text{ML}}^2] = \left( \frac{N-1}{N} \right) \sigma^2 \quad (1.58)$$

$$\tilde{\sigma}^2 = \frac{N}{N-1} \sigma_{\text{ML}}^2 = \frac{1}{N-1} \sum_{n=1}^N (x_n - \mu_{\text{ML}})^2. \quad (1.59)$$

**Figure 1.15**

Illustration of how bias arises in using maximum likelihood to determine the variance of a Gaussian. The green curve shows the true Gaussian distribution from which data is generated, and the three red curves show the Gaussian distributions obtained by fitting to three data sets, each consisting of two data points shown in blue, using the maximum likelihood results (1.55) and (1.56). Averaged across the three data sets, the mean is correct, but the variance is systematically under-estimated because it is measured relative to the sample mean and not relative to the true mean.



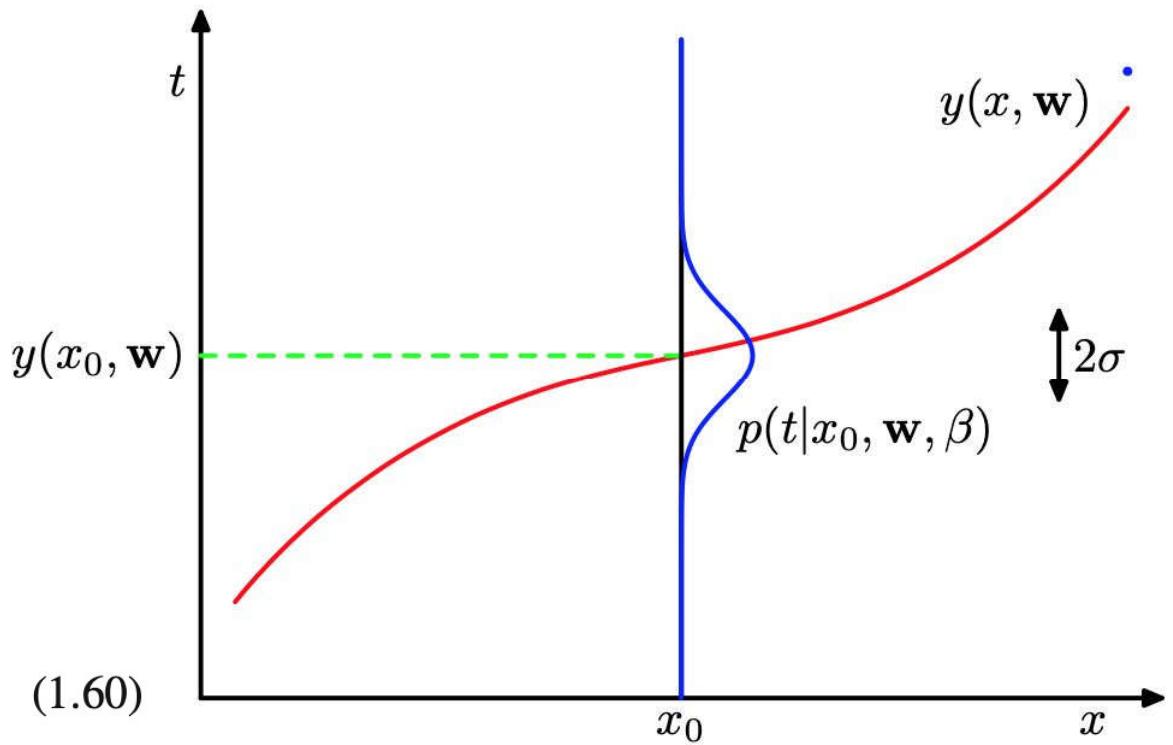
Q: does high bias/variance means that the model is overfitted, or vice versa?

## Bringing it together:

### Curve fitting with maximum likelihood

**Figure 1.16** Schematic illustration of a Gaussian conditional distribution for  $t$  given  $x$  given by (1.60), in which the mean is given by the polynomial function  $y(x, \mathbf{w})$ , and the precision is given by the parameter  $\beta$ , which is related to the variance by  $\beta^{-1} = \sigma^2$ .

$$p(t|x, \mathbf{w}, \beta) = \mathcal{N}(t|y(x, \mathbf{w}), \beta^{-1}) \quad (1.60)$$



Goal: estimate ~~\beta~~  $\beta$

$$p(t|x, \mathbf{w}, \beta) = \mathcal{N}(t|y(x, \mathbf{w}), \beta^{-1}) \quad (1.60)$$

$$p(\underline{\mathbf{t}|\mathbf{x}}, \mathbf{w}, \beta) = \prod_{n=1}^N \mathcal{N}(t_n|y(x_n, \mathbf{w}), \beta^{-1}). \quad (1.61)$$

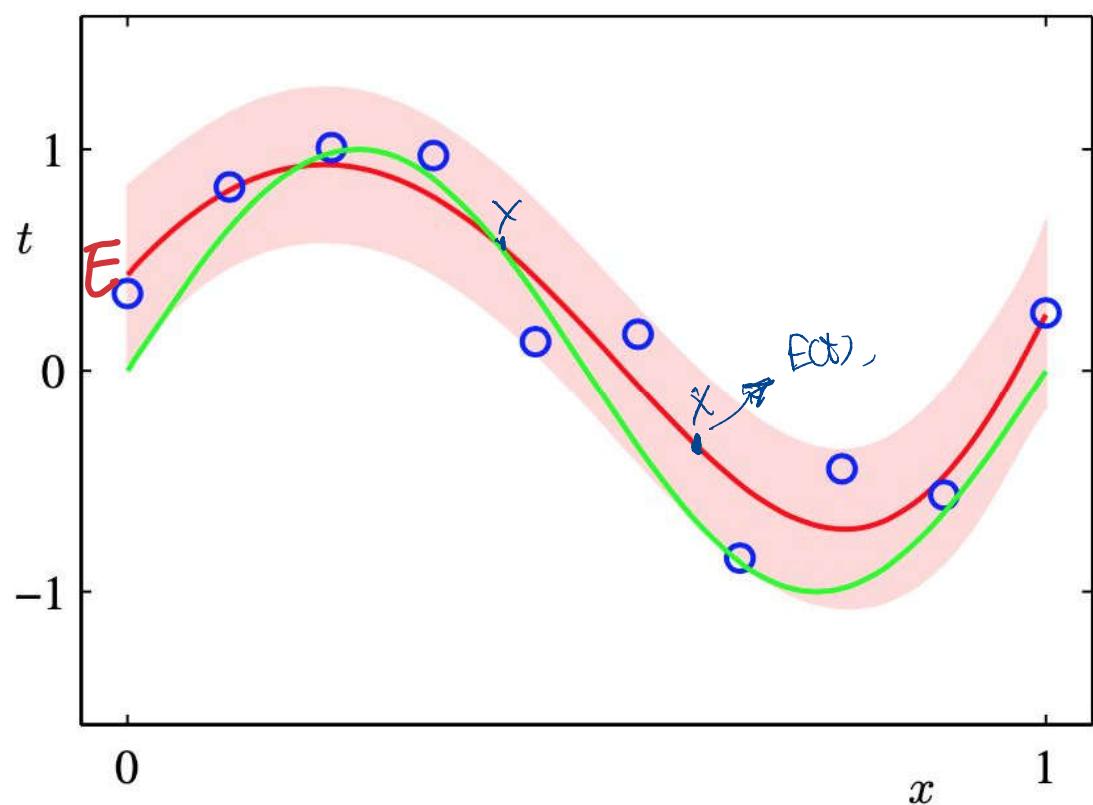
$$\ln p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = -\frac{\beta}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi). \quad (1.62)$$

$$\begin{aligned} \frac{\partial L}{\partial \beta} &= -\frac{1}{2} \sum_{n=1}^N \frac{\downarrow}{\beta} \\ \frac{1}{\beta_{ML}} &= \frac{1}{N} \sum_{n=1}^N \{y(x_n, \mathbf{w}_{ML}) - t_n\}^2. \end{aligned} \quad (1.63)$$

# curve-fitting: predictive distribution

(will cover next week in Bayesian linear regression)

**Figure 1.17** The predictive distribution resulting from a Bayesian treatment of polynomial curve fitting using an  $M = 9$  polynomial, with the fixed parameters  $\alpha = 5 \times 10^{-3}$  and  $\beta = 11.1$  (corresponding to the known noise variance), in which the red curve denotes the mean of the predictive distribution and the red region corresponds to  $\pm 1$  standard deviation around the mean.



# What we did so far

ML 101:

Polynomial curve fitting: model, loss/error function, over-fitting, regularisation

Probabilities: sum rule, product rule, Bayes theorem

Gaussians - 1D, MLE, bias-variance

→ and how this helps curve-fitting

Bernoulli, binomial

Gaussians (multidimensional)

various matrix identities, geometric intuitions

Exponential family

Review: probabilities, derivatives and finding stationary points, eigen values and eigen vectors

## From Bernoulli to Binomial

$$\text{Bern}(x|\mu) = \mu^x(1-\mu)^{1-x} \quad | \text{ coin toss}$$

即二项分布.

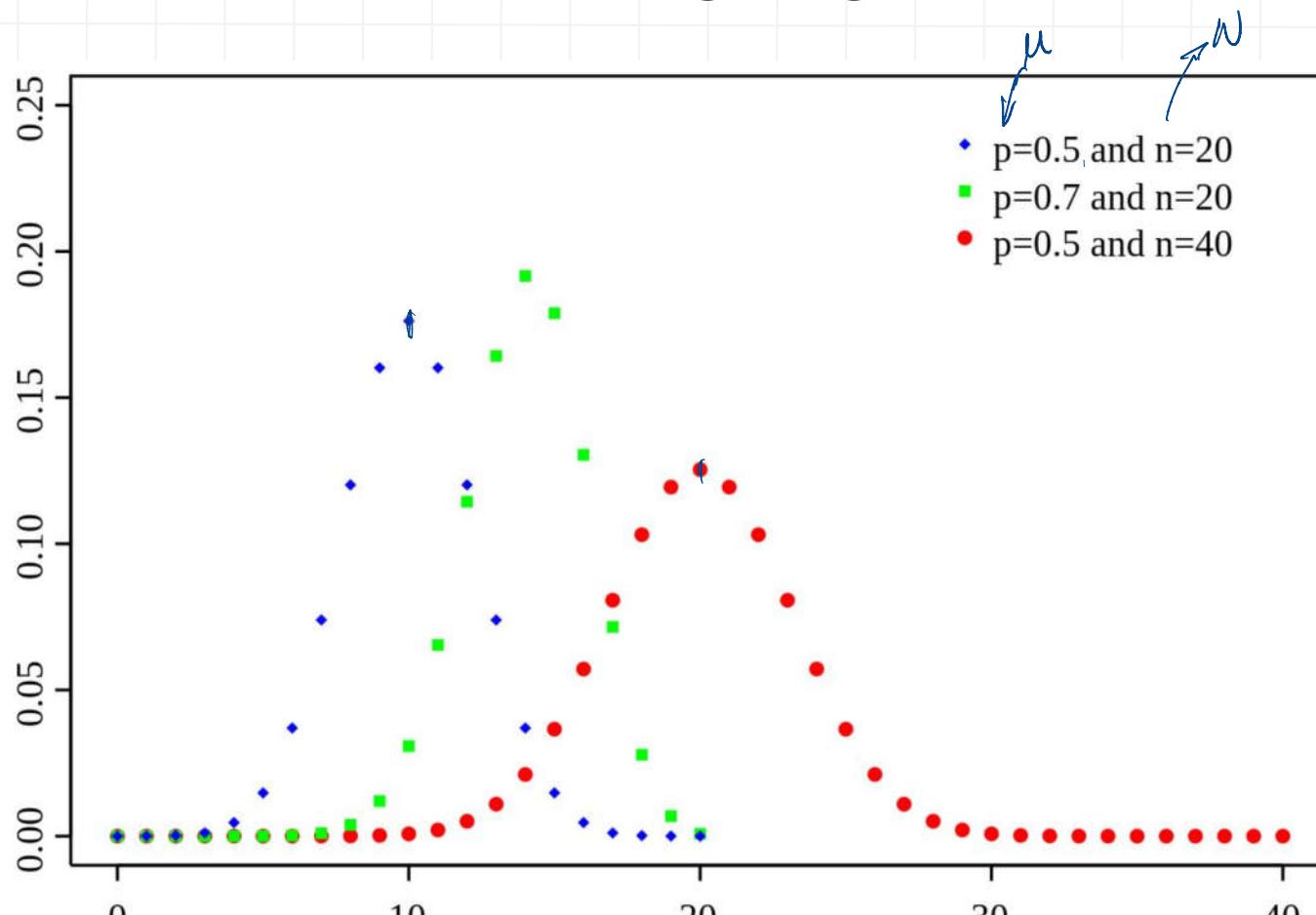
$x \in \{0, 1\}$

$N$  tosses

$$\text{Bin}(m|N, \mu) = \binom{N}{m} \mu^m (1-\mu)^{N-m} \quad | \text{P 多次重复实验 (2.9)}$$

$$\binom{N}{m} \equiv \frac{N!}{(N-m)!m!} \quad (2.10)$$

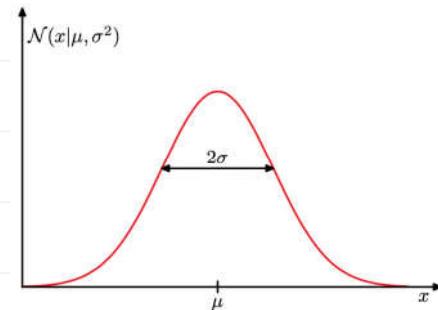
## binomial for increasing large N



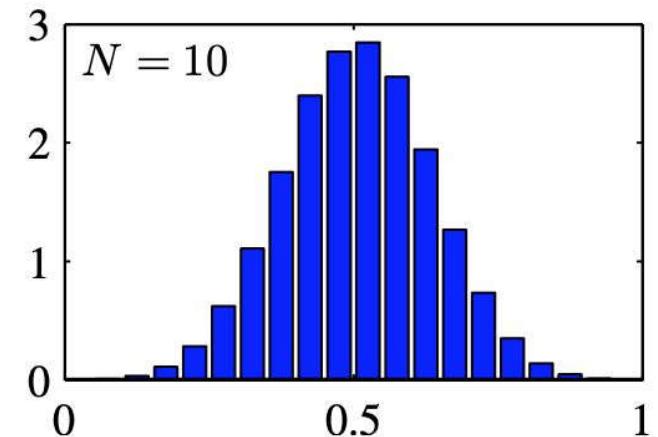
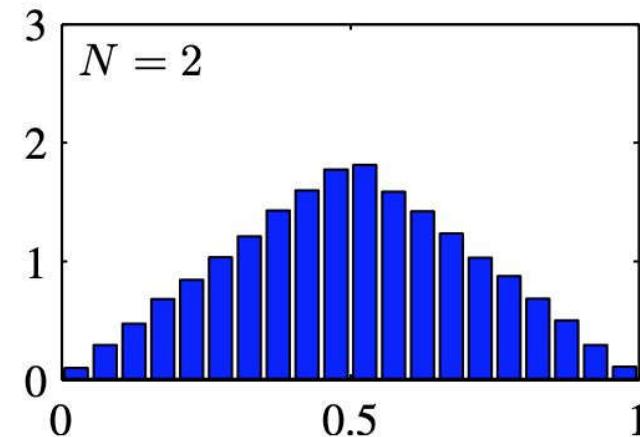
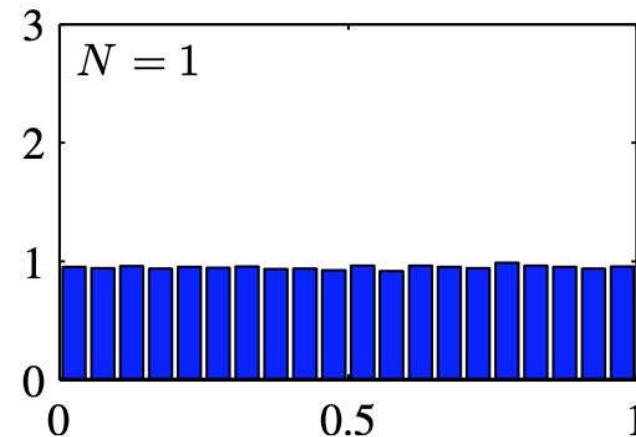
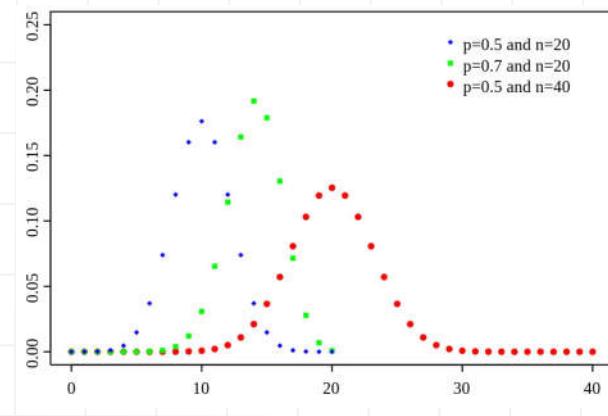
# Gaussians again: why is it cool?

CLT - central limit theorem

max entropy



n coin tosses with p



**Figure 2.6** Histogram plots of the mean of  $N$  uniformly distributed numbers for various values of  $N$ . We observe that as  $N$  increases, the distribution tends towards a Gaussian.

# Gaussians – multidimensional

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\} \quad (2.43)$$

$\boldsymbol{\Sigma}$  symmetric

$$\sigma_{12} = \sigma_{21}$$

$$\begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{bmatrix}$$

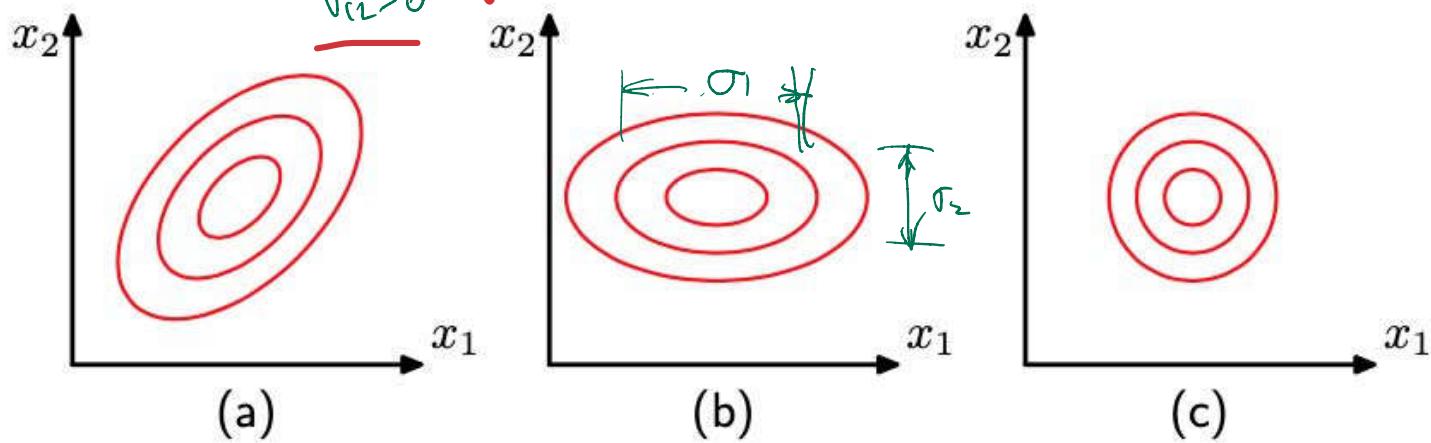
$$\begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$$

$\text{cov}(x_1, x_2) = 0$

因为不相关

**Figure 2.8** Contours of constant probability density for a Gaussian distribution in two dimensions in which the covariance matrix is (a) of general form, (b) diagonal, in which the elliptical contours are aligned with the coordinate axes, and (c) proportional to the identity matrix, in which the contours are concentric circles.



$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\} \quad (2.43)$$

*squared dist.*

Eigen decomposition of the cov matrix

$$\begin{array}{l} u_i \\ u_i^T u_i \\ u_i u_i^T \end{array} \xrightarrow{\square \rightarrow \square} \begin{array}{l} \square \\ 1 \\ \square \end{array}$$

$$\begin{array}{l} u_i^T u_i = 1 \\ u_i^T u_j = 0 \quad i \neq j \end{array}$$

$$\boldsymbol{\Sigma} \mathbf{u}_i = \lambda_i \mathbf{u}_i \quad \begin{array}{l} \lambda_i \geq 0 \\ \text{orthogonal} \end{array} \quad (2.45)$$

$$\mathbf{u}_i^T \mathbf{u}_j = I_{ij} \quad (2.46)$$

$$\boldsymbol{\Sigma} = \sum_{i=1}^D \lambda_i \mathbf{u}_i \mathbf{u}_i^T \quad \boldsymbol{\Sigma}^{-1} = \sum_{i=1}^D \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^T.$$

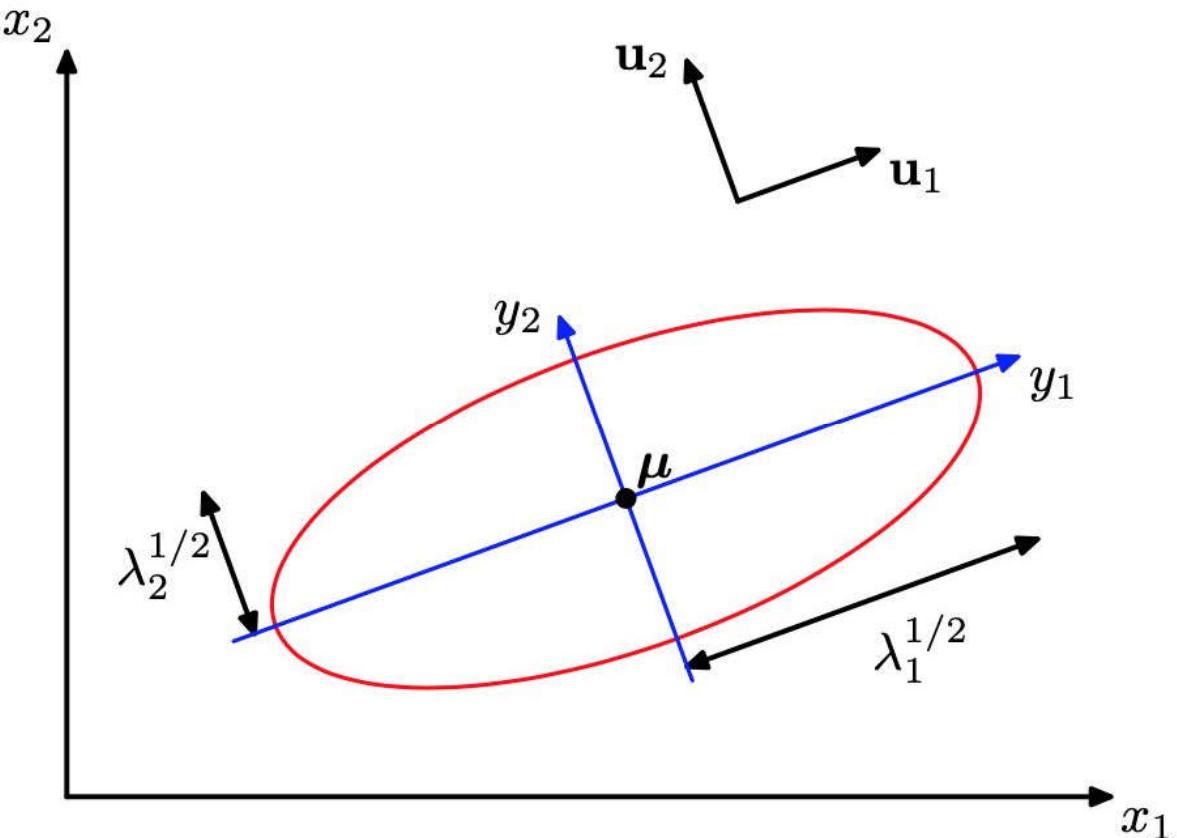
$$\Delta^2 = (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) = \sum_{i=1}^D \frac{y_i^2}{\lambda_i} \quad y_i = \mathbf{u}_i^T (\mathbf{x} - \boldsymbol{\mu})$$

# Contours of general 2-D Gaussians - a rotated ellipse

$$\mathbf{y} = \mathbf{U}(\mathbf{x} - \boldsymbol{\mu})$$

**Figure 2.7**

The red curve shows the elliptical surface of constant probability density for a Gaussian in a two-dimensional space  $\mathbf{x} = (x_1, x_2)$  on which the density is  $\exp(-1/2)$  of its value at  $\mathbf{x} = \boldsymbol{\mu}$ . The major axes of the ellipse are defined by the eigenvectors  $\mathbf{u}_i$  of the covariance matrix, with corresponding eigenvalues  $\lambda_i$ .



# The Exponential family

Beyond Gaussians: What is a class of ‘nice’ distributions for statistical machine learning?

- More expressive
- “Easy” to estimate

$$p(\mathbf{x}|\boldsymbol{\eta}) = h(\mathbf{x})g(\boldsymbol{\eta}) \exp \{ \boldsymbol{\eta}^T \mathbf{u}(\mathbf{x}) \} \quad (2.194)$$

normalisation

$$g(\boldsymbol{\eta}) \int h(\mathbf{x}) \exp \{ \boldsymbol{\eta}^T \mathbf{u}(\mathbf{x}) \} d\mathbf{x} = 1 \quad (2.195)$$

Special case: Bernoulli

$$p(x|\mu) = \text{Bern}(x|\mu) = \mu^x (1-\mu)^{1-x}. \quad (2.196)$$

$$\begin{aligned} &= \exp \{x \ln \mu + (1-x) \ln(1-\mu)\} \\ &= (1-\mu) \exp \left\{ \ln \left( \frac{\mu}{1-\mu} \right) x \right\}. \end{aligned} \quad (2.197)$$

$$\eta = \ln \left( \frac{\mu}{1-\mu} \right)$$

$$p(x|\eta) = \sigma(-\eta) \exp(\eta x) \quad (2.200)$$

## Special case: Gaussian

$$p(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp \left\{ -\frac{1}{2\sigma^2}(x - \mu)^2 \right\} \quad (2.218)$$

$$= \frac{1}{(2\pi\sigma^2)^{1/2}} \exp \left\{ -\frac{1}{2\sigma^2}x^2 + \frac{\mu}{\sigma^2}x - \frac{1}{2\sigma^2}\mu^2 \right\} \quad (2.219)$$

$$\boldsymbol{\eta} = \begin{pmatrix} \mu/\sigma^2 \\ -1/2\sigma^2 \end{pmatrix} \quad (2.220)$$

*Sufficient stat.*

$$(x_1, \dots, x_n) \xrightarrow{n \gg \infty} \mathbf{u}(x) = \begin{pmatrix} x \\ x^2 \end{pmatrix} \quad (2.221)$$

$$h(\mathbf{x}) = (2\pi)^{-1/2} \quad (2.222)$$

$$g(\boldsymbol{\eta}) = (-2\eta_2)^{1/2} \exp \left( \frac{\eta_1^2}{4\eta_2} \right). \quad (2.223)$$

normalisation

$$g(\boldsymbol{\eta}) \int h(\mathbf{x}) \exp \{ \boldsymbol{\eta}^T \mathbf{u}(\mathbf{x}) \} d\mathbf{x} = 1 \quad (2.195)$$

$$\begin{aligned} & \nabla g(\boldsymbol{\eta}) \int h(\mathbf{x}) \exp \{ \boldsymbol{\eta}^T \mathbf{u}(\mathbf{x}) \} d\mathbf{x} \\ & + g(\boldsymbol{\eta}) \int h(\mathbf{x}) \exp \{ \boldsymbol{\eta}^T \mathbf{u}(\mathbf{x}) \} \mathbf{u}(\mathbf{x}) d\mathbf{x} = 0. \end{aligned} \quad (2.224)$$

$$-\frac{1}{g(\boldsymbol{\eta})} \nabla g(\boldsymbol{\eta}) = g(\boldsymbol{\eta}) \int h(\mathbf{x}) \exp \{ \boldsymbol{\eta}^T \mathbf{u}(\mathbf{x}) \} \mathbf{u}(\mathbf{x}) d\mathbf{x} = \mathbb{E}[\mathbf{u}(\mathbf{x})] \quad (2.225)$$

MLE and sufficient stats

$$-\nabla \ln g(\boldsymbol{\eta}_{\text{ML}}) = \frac{1}{N} \sum_{n=1}^N \mathbf{u}(\mathbf{x}_n)$$

# Exponential family: a note about notations

$$p(\mathbf{x}|\boldsymbol{\eta}) = h(\mathbf{x})g(\boldsymbol{\eta}) \exp \left\{ \boldsymbol{\eta}^T \mathbf{u}(\mathbf{x}) \right\} \quad (2.194)$$

$\downarrow$

$$h(x)=1, g(\eta) = \exp(-\Psi(\eta))$$

Assignment 1 Q2

**Definition 1** (Exponential Family<sup>2</sup>). Given a function  $\mathbf{u} : \mathbb{R} \rightarrow \mathbb{R}^m$ , we denote an exponential family distribution as  $\text{EXP}(\mathbf{u}, \boldsymbol{\eta})$ , where  $\boldsymbol{\eta} \in \mathcal{P} \subset \mathbb{R}^m$  designates the  $m$ -dimensional parameters of the distribution within an exponential family<sup>3</sup>. The corresponding densities of the distributions are given by

$$q(x; \boldsymbol{\eta}) = \exp(\boldsymbol{\eta}^T \mathbf{u}(x) - \psi(\boldsymbol{\eta})), \quad (1)$$

where

$$\psi(\boldsymbol{\eta}) = \log \int \exp(\boldsymbol{\eta}^T \mathbf{u}(x)) dx. \quad (2)$$

The function  $\mathbf{u}$  is called the sufficient statistics of the exponential family and the function  $\psi$  is called the log-partition function of the exponential family.

## About these lecture notes:

- They are designed to be visual aid but not reading material (you have the book for that).
- They are generally focused on derivations + plots and less on the “story” part of the model.
- I do not aim to produce new equations nor new plots (they don’t necessarily help you learn : )

## A word about data/plots in the book:

- Reasoning about ML models on toy data is a core skill of a good ML engineer.
- Designing appropriate toy data is a core research skill in ML.

# What we covered today

ML 101:

Polynomial curve fitting: model, loss/error function, over-fitting, regularisation

Model selection

Probabilities: sum rule, product rule, Bayes theorem

Gaussians - 1D, MLE, bias-variance

→ and how this helps curve-fitting

Gaussians (multidimensional)

various matrix identities, geometric intuitions

Bernoulli, Binomial, Exponential family distributions

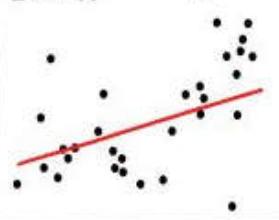
Review: probabilities, derivatives and finding stationary points, eigen values and eigen vectors



<https://xkcd.com/2048/>

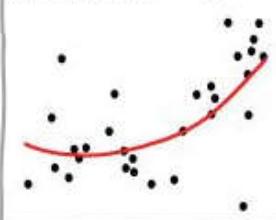
## CURVE-FITTING METHODS AND THE MESSAGES THEY SEND

LINEAR



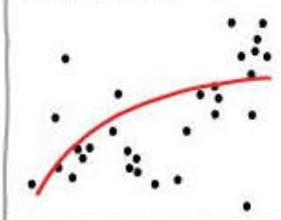
"HEY, I DID A  
REGRESSION."

QUADRATIC



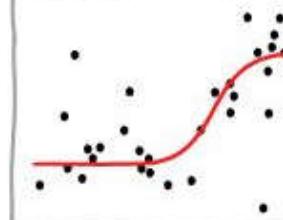
"I WANTED A CURVED  
LINE, SO I MADE ONE  
WITH MATH."

LOGARITHMIC



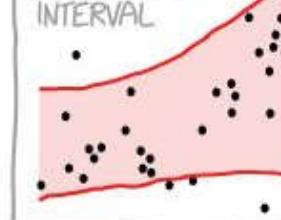
"LOOK, IT'S  
TAPERING OFF!"

LOGISTIC



"I NEED TO CONNECT THESE  
TWO LINES, BUT MY FIRST IDEA  
DIDN'T HAVE ENOUGH MATH."

CONFIDENCE  
INTERVAL



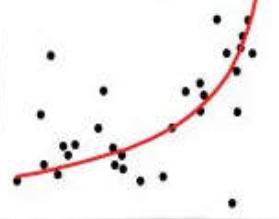
"LISTEN, SCIENCE IS HARD.  
BUT I'M A SERIOUS  
PERSON DOING MY BEST."

PIECEWISE



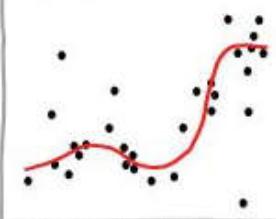
"I HAVE A THEORY,  
AND THIS IS THE ONLY  
DATA I COULD FIND."

EXPONENTIAL



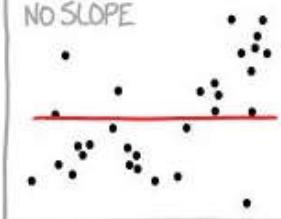
"LOOK, IT'S GROWING  
UNCONTROLLABLY!"

LOESS



"I'M SOPHISTICATED, NOT  
LIKE THOSE BUMBLING  
POLYNOMIAL PEOPLE."

LINEAR,  
NO SLOPE



"I'M MAKING A  
SCATTER PLOT BUT  
I DON'T WANT TO."

CONNECTING  
LINES



"I CLICKED 'SMOOTH  
LINES' IN EXCEL."

AD-HOC  
FILTER



"I HAD AN IDEA FOR HOW  
TO CLEAN UP THE DATA.  
WHAT DO YOU THINK?"

HOUSE OF  
CARDS



"AS YOU CAN SEE, THIS  
MODEL SMOOTHLY FITS  
THE- WAIT NO NO DON'T  
EXTEND IT AAAAAA!!"

# Announcements

Assignment 1 out, we'll talk about it in a future session.

3 new tutors joined

Tutorials start this week

Hope everyone is doing okay!

- max, learning -  
min suffering,

# Linear Regression (Linear models for regression)

Why linear models?

Input data, features, basis functions

Maximum likelihood and least squares

Geometric intuition

Regularised least squares

Multiple outputs → book

→ Bias-variance decomposition

interpretable / sparse linear models

The relation between MLE and least squares, Lagrange multipliers, multiple ways of looking at linear models.

# Why linear models? - it saves lives

The five criteria of the Apgar score:<sup>[3]</sup>

	Score of 0	Score of 1	Score of 2	Component of backronym
<b>Skin color</b>	blue or pale all over	blue at extremities, body pink (acrocyanosis)	no cyanosis body and extremities pink	Appearance
<b>Pulse rate</b>	absent	< 100 beats per minute	≥ 100 beats per minute	Pulse
<b>Reflex irritability grimace</b>	no response to stimulation	grimace on suction or aggressive stimulation	cry on stimulation	Grimace
<b>Muscle Tone</b>	none	some flexion	flexed arms and legs that resist extension	Activity
<b>Respiratory effort</b>	absent	weak, irregular, gasping	strong, robust cry	Respiration

## What do the Apgar scores mean?

A score of 7 or more is normal. A score of 6 or less at 1 minute and a score of 7 or more at 5 minutes is also normal. However, a score below 7 in the second test at 5 minutes is considered low.

If your baby's score was low in the first Apgar test and hasn't improved in the second test at 5 minutes, or there are other concerns, the doctors and nurses will closely monitor your baby and continue any necessary medical care.



Virginia Apgar, creator of the Apgar score

**Purpose** method to summarize newborn's health

Image and table from wikipedia  
<https://www.pregnancybirthbaby.org.au/apgar-score>

Everything should be made as simple as possible, but not simpler. -- Albert Einstein

**Occam's razor**, also spelled **Ockham's razor**, also called **law of economy** or **law of parsimony**, principle stated by the Scholastic philosopher [William of Ockham](#) (1285–1347/49) that *pluralitas non est ponenda sine necessitate*, “plurality should not be posited without necessity.” The principle gives precedence to simplicity: of two competing theories, the simpler [explanation](#) of an entity is to be preferred. The principle is also expressed as “Entities are not to be multiplied beyond necessity.”

<https://www.britannica.com/topic/Occams-razor>

### SNAP-II and SNAPPE II (Score for Neonatal Acute Physiology and SNAP Perinatal Extension)

Variables	Values	Points
Mean Blood Pressure	<input type="button" value="▼"/>	0
Lowest temperature	<input type="button" value="▼"/>	0
P <sub>O</sub> <sub>2</sub> (mmHg) / FIO <sub>2</sub> (%)	<input type="button" value="▼"/>	0
Lowest serum pH	<input type="button" value="▼"/>	0
Multiple seizures	<input type="button" value="▼"/>	0
Urine output (mL/kg.h)	<input type="button" value="▼"/>	0
SNAP II : 0		
Apgar score	<input type="button" value="▼"/>	
Birth weight	<input type="button" value="▼"/>	
Small for gestational age ( <a href="#">help</a> )	<input type="button" value="▼"/>	
SNAPPE II : 0		In-hospital mortality : <a href="#">see below</a> Data are collected within 11 NICU

[Clear](#)

Ref: D K. Richardson et al . SNAP-II and SNAPPE-II: Simplified newborn illness severity and mortality. 2001; 138: 92-100

<https://sfar.org/scores2/snap22.php>

<https://www.sciencedirect.com/science/article/abs/pii/S0140673603133971>

Research Letters

The Lancet, 2003

## CRIB II: an update of the clinical risk index for babies score

Dr Gareth Parry PhD <sup>a</sup>  , Janet Tucker PhD <sup>b</sup>, William Tarnow-Mordi MRCP <sup>c</sup>, for the UK Neonatal Staffing Study Collaborative Group <sup>\*</sup>

The clinical risk index for babies (CRIB) score is a risk-adjustment instrument used worldwide in [neonatal intensive care](#).<sup>1</sup> It was developed with data relating to babies born at less than 31 weeks' gestation, or 1500 g birthweight or lower, admitted for neonatal intensive care between 1988 and 1990. The appropriateness of CRIB with contemporary data has been questioned, since the score may now be poorly calibrated with mortality after neonatal intensive care, in which case it might be no better in prediction of mortality than birthweight or gestation alone. Furthermore, CRIB includes, as one of two measures of severity of illness, [fraction of inspired oxygen](#) (FiO<sub>2</sub>), which is not a true physiological measure because it is determined by the care team. CRIB also includes data up to 12 h after admission, thus potentially introducing early treatment bias.

# Linear models and likelihood

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \dots + w_D x_D$$

$$\mathbf{w}^T \vec{\mathbf{x}}$$

$$\vec{\mathbf{x}} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_D \end{bmatrix}$$

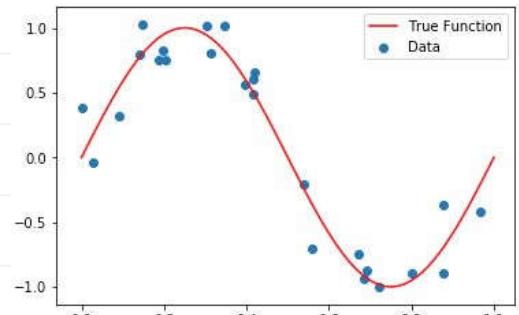
$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$$

- Linear combination of **fixed** nonlinear basis functions
- parameter  $\mathbf{w} = (w_0, \dots, w_{M-1})^T$
- basis functions  $\boldsymbol{\phi}(\mathbf{x}) = (\phi_0(\mathbf{x}), \dots, \phi_{M-1}(\mathbf{x}))^T$
- convention  $\phi_0(\mathbf{x}) = 1$
- $w_0$  is the **bias parameter**

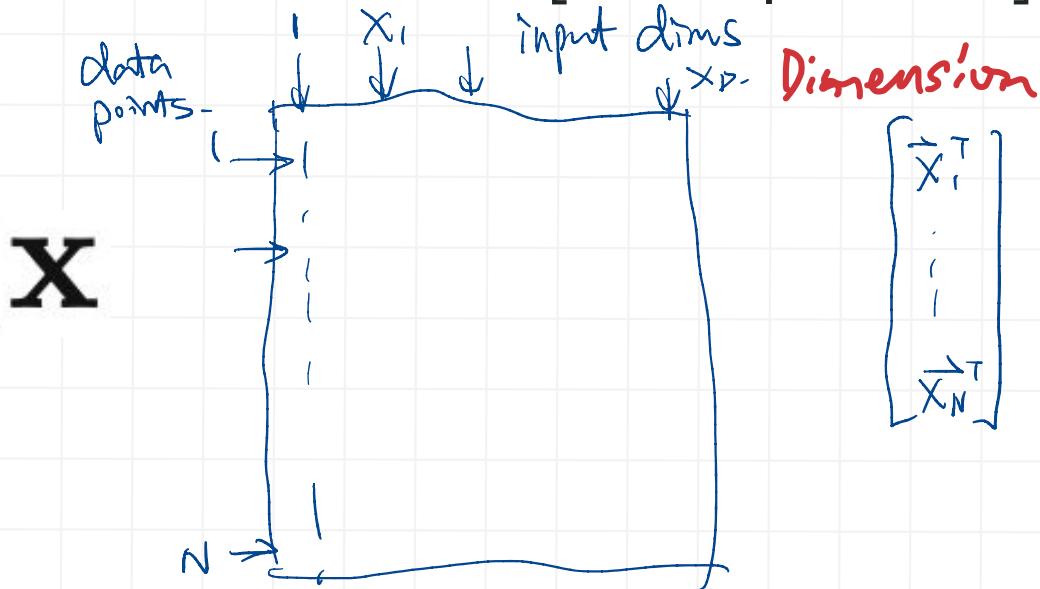
Why linear regression?

- Analytic solution when minimising sum of squared errors
- Well understood statistical behaviour
- Efficient algorithms exist for convex losses and regularizers

But what if ?



# Conventions in [Bishop 2006]



$\phi_j(\mathbf{x})$

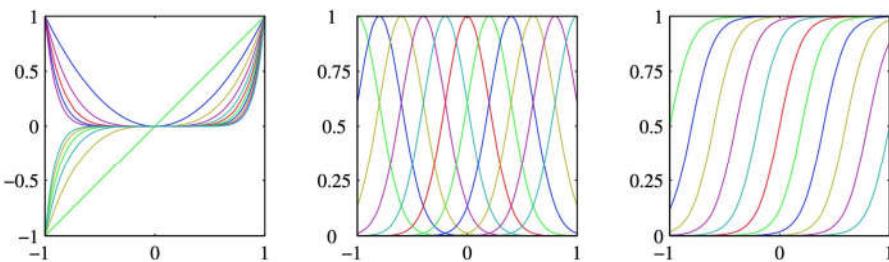
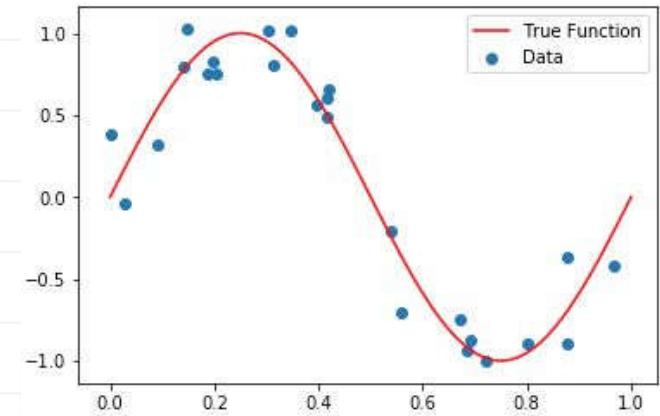
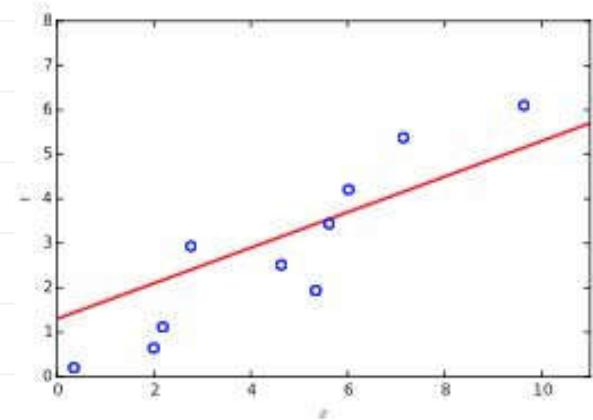


Figure 3.1 Examples of basis functions, showing polynomials on the left, Gaussians of the form (3.4) in the centre, and sigmoidal of the form (3.5) on the right.

$$\mathbf{x}_i = \begin{bmatrix} 1 \\ x_{i1} \\ \vdots \\ x_{iD} \end{bmatrix}$$

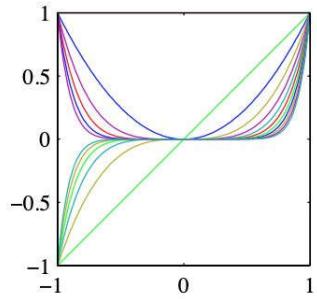


## Polynomial Basis Functions

- Scalar input variable  $x$

$$\phi_j(x) = x^j$$

- Limitation : Polynomials are global functions of the input variable  $x$  so the learned function will extrapolate poorly

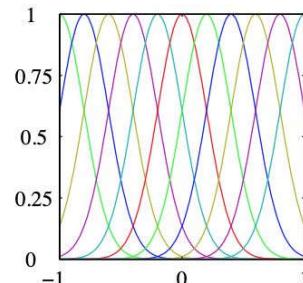


## 'Gaussian' Basis Functions

- Scalar input variable  $x$

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

- Not a probability distribution.
- No normalisation required, taken care of by the model parameters  $w$ .
- Well behaved away from the data (though pulled to zero)



## Sigmoidal Basis Functions

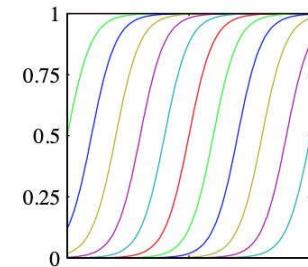
- Scalar input variable  $x$

$$\phi_j(x) = \sigma\left(\frac{x - \mu_j}{s}\right)$$

where  $\sigma(a)$  is the logistic sigmoid function defined by

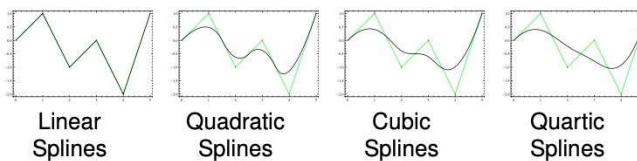
$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

- $\sigma(a)$  is related to the hyperbolic tangent  $\tanh(a)$  by  $\tanh(a) = 2\sigma(a) - 1$ .



## Other Basis Functions

- Fourier Basis : each basis function represents a specific frequency and has infinite spatial extent.
- Wavelets : localised in both space and frequency (also mutually orthogonal to simplify application).
- Splines (piecewise polynomials restricted to regions of the input space; additional constraints where pieces meet, e.g. smoothness constraints  $\rightarrow$  conditions on the derivatives).



Approximate the points  $\{(0,0), (1,1), (2,-1), (3,0), (4,-2), (5,1)\}$  by different splines.

# Linear models and likelihood

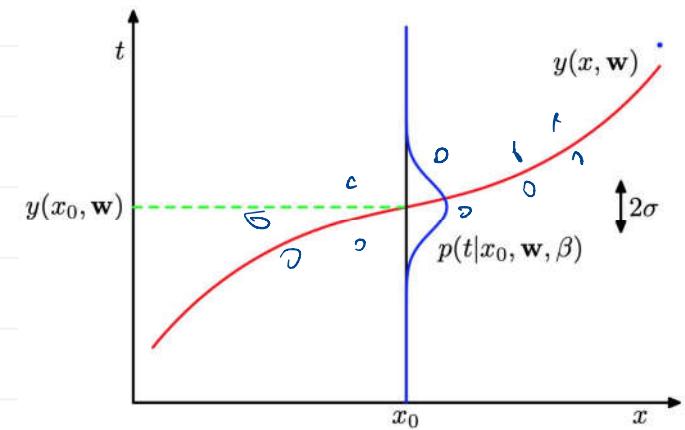
$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \dots + w_D x_D$$

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$$

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon \quad (3.7)$$

$$p(t|x, \mathbf{w}, \beta) = \mathcal{N}(t|y(x, \mathbf{w}), \beta^{-1}) \quad (1.60) \text{ also (3.8)}$$

Fig 1.16



## Computing the likelihood

$$p(t|x, \mathbf{w}, \beta) = \mathcal{N}(t|y(x, \mathbf{w}), \beta^{-1}) \quad (1.60)$$

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

$$\begin{aligned} \ln p(\mathbf{t} | \mathbf{w}, \beta) &= \sum_{n=1}^N \ln \mathcal{N}(t_n | \mathbf{w}^T \phi(\mathbf{x}_n), \beta^{-1}) \\ &= \sum_{n=1}^N \ln \left( \sqrt{\frac{\beta}{2\pi}} \exp \left\{ -\frac{\beta}{2} (t_n - \mathbf{w}^T \phi(\mathbf{x}_n))^2 \right\} \right) \\ &= \dots \text{(see next page)} \end{aligned}$$

## Computing the likelihood

$$\begin{aligned}\ln p(\mathbf{t} | \mathbf{w}, \beta) &= \sum_{n=1}^N \ln \mathcal{N}(t_n | \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}) \\ &= \sum_{n=1}^N \ln \left( \sqrt{\frac{\beta}{2\pi}} \exp \left\{ -\frac{\beta}{2} (t_n - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n))^2 \right\} \right) \\ &= \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta E_D(\mathbf{w})\end{aligned}\tag{3.11}$$

the sum-of-squares error function is defined by

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n)\}^2.\tag{3.12}$$

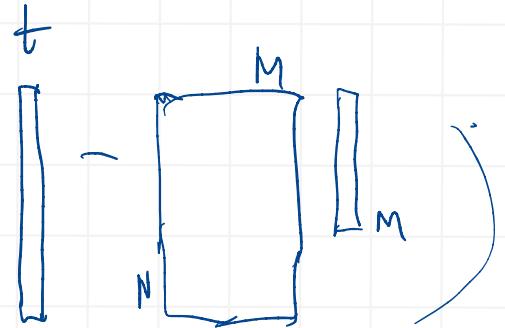
Claim:  $\arg \max_{\mathbf{w}} \ln p(\mathbf{t} | \mathbf{w}, \beta) \rightarrow \arg \min_{\mathbf{w}} E_D(\mathbf{w})$

... first rewrite the error function

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2.$$

$$= \frac{1}{2} (\mathbf{t} - \Phi \mathbf{w})^T (\mathbf{t} - \Phi \mathbf{w})$$

(3.12)



where  $\mathbf{t} = (t_1, \dots, t_N)^T$ , and

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

plug into (3.11)  $\ln p(\mathbf{t} | \mathbf{w}, \beta) = \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta E_D(\mathbf{w})$

Then find the stationary point

$$\begin{aligned}\ln p(\mathbf{t} | \mathbf{w}, \beta) &= \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta E_D(\mathbf{w}) \\ &= \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta \frac{1}{2} (\mathbf{t} - \Phi \mathbf{w})^T (\mathbf{t} - \Phi \mathbf{w})\end{aligned}$$

The gradient with respect to  $\mathbf{w}$  is

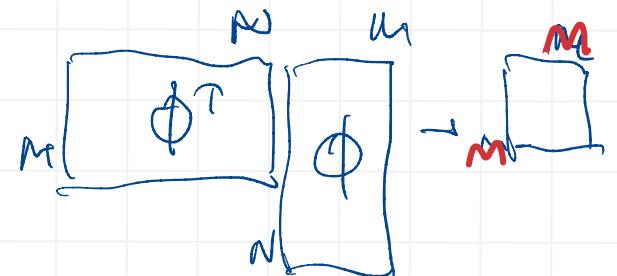
$$\nabla_{\mathbf{w}} \ln p(\mathbf{t} | \mathbf{w}, \beta) = \beta \Phi^T (\mathbf{t} - \Phi \mathbf{w}).$$

Setting the gradient to zero gives

$$0 = \Phi^T \mathbf{t} - \Phi^T \Phi \mathbf{w},$$

The normal equation:

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



(3.15)

## Obtaining and interpreting MLE results

$$w_0 = \bar{t} - \sum_{j=1}^{M-1} w_j \bar{\phi}_j$$

$$\bar{t} = \frac{1}{N} \sum_{n=1}^N t_n,$$

$$\bar{\phi}_j = \frac{1}{N} \sum_{n=1}^N \phi_j(\mathbf{x}_n).$$

Recall:  $\ln p(\mathbf{t} | \mathbf{w}, \beta) = \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta \frac{1}{2} (\mathbf{t} - \Phi \mathbf{w})^T (\mathbf{t} - \Phi \mathbf{w})$

$$\frac{1}{\beta_{\text{ML}}} = \frac{1}{N} \sum_{n=1}^N \{t_n - \mathbf{w}_{\text{ML}}^T \phi(\mathbf{x}_n)\}^2 \quad (3.21)$$



The normal equation:

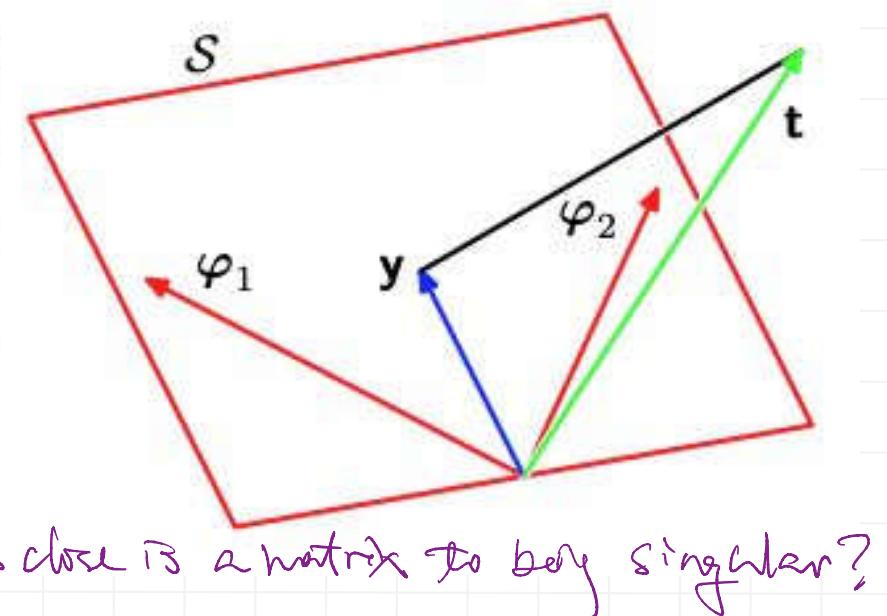
$$\mathbf{w}_{\text{ML}} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{t} \quad (3.15)$$

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$$

**Figure 3.2** Geometrical interpretation of the least-squares solution, in an  $N$ -dimensional space whose axes are the values of  $t_1, \dots, t_N$ . The least-squares regression function is obtained by finding the orthogonal projection of the data vector  $\mathbf{t}$  onto the subspace spanned by the basis functions  $\phi_j(\mathbf{x})$  in which each basis function is viewed as a vector  $\varphi_j$  of length  $N$  with elements  $\phi_j(\mathbf{x}_n)$ .

Numerical difficulties when  $\Phi^T \Phi$  is ill-conditioned.

SVD or regularisation will help.



How close is a matrix to being singular?

Cool geometric derivation of the normal equation  
<https://www.youtube.com/watch?v=PbyP3goun2Y>

## Sequential Learning - Stochastic Gradient Descent

- For **large data sets**, calculating the maximum likelihood parameters  $\mathbf{w}_{ML}$  and  $\beta_{ML}$  may be costly.
- For **online** applications, never all data in memory.
- Use a **sequential** algorithms (**online** algorithm).
- If the error function is a sum over data points  $E = \sum_n E_n$ , then
  - ➊ initialise  $\mathbf{w}^{(0)}$  to some starting value
  - ➋ update the parameter vector at iteration  $\tau + 1$  by

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n, \quad = \mathbf{w}^{(\tau)} + \eta \left( t_n - \mathbf{w}^{(\tau)T} \phi(\mathbf{x}_n) \right) \phi(\mathbf{x}_n)$$

where  $E_n$  is the error function after presenting the  $n$ th data set, and  $\eta$  is the learning rate.

(learning rate)

Sum-of-squares error

# What we did so far

Why linear models?

Input data, features, basis functions

Maximum likelihood and least squares

Geometric intuition, sequential update

Regularised least squares

Multiple outputs

Bias-variance decomposition

## Regularised least squares

$$L(\mathbf{w}) = E_D(\mathbf{w}) + \lambda E_W(\mathbf{w}) \quad (3.24)$$

$$\frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}. \quad (3.27) \quad \text{also (1.4)}$$

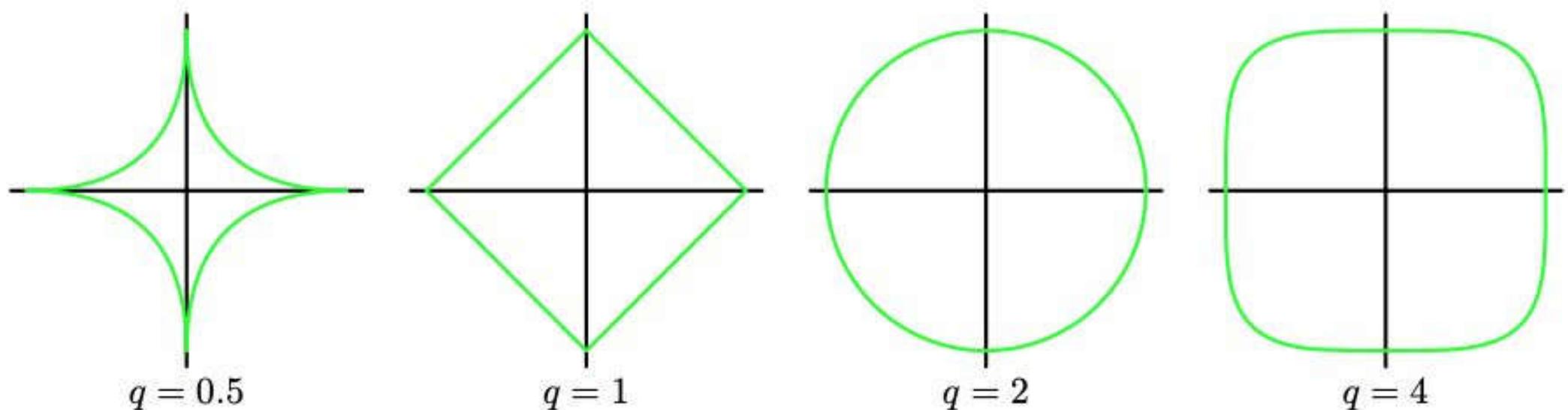
$$\mathbf{w} = (\underbrace{\lambda \mathbf{I} + \Phi^T \Phi}_{\text{better 'conditioned' eigen vals } \geq \lambda})^{-1} \Phi^T \mathbf{t}. \quad (3.28)$$

MLE, ↓

$$\frac{\partial L}{\partial \mathbf{w}} = \left( \quad \right) + \lambda \mathbf{w}$$

## Regularisation by q-norm

$$\frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2} \sum_{j=1}^M |w_j|^q \quad (3.29)$$



**Figure 3.3** Contours of the regularization term in (3.29) for various values of the parameter  $q$ .

# Lagrange multipliers (appendix E)

The first encounter in SML - we'll see it again in kernel methods.

objective function  
equality constraint

maximize  $f(x)$   
subject to  $g(x)=0$

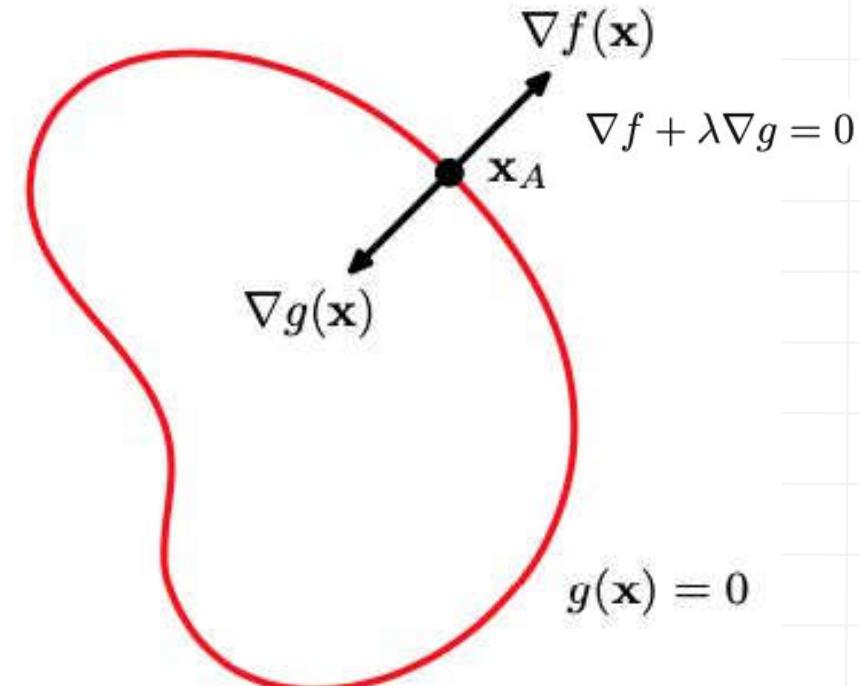


$$\mathcal{L}(x, \lambda) = f(x) + \lambda g(x)$$

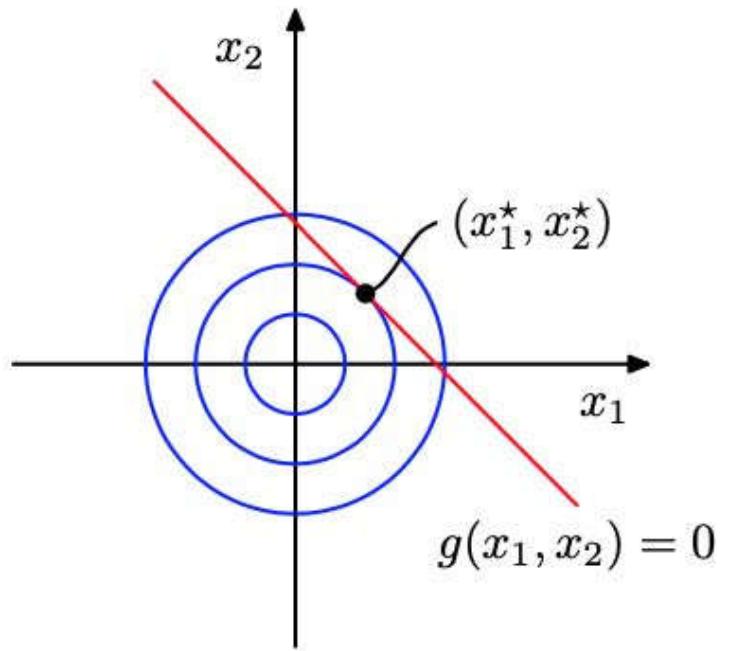
$\mathcal{L}$  = Lagrangian

$\lambda$  = Lagrange multiplier

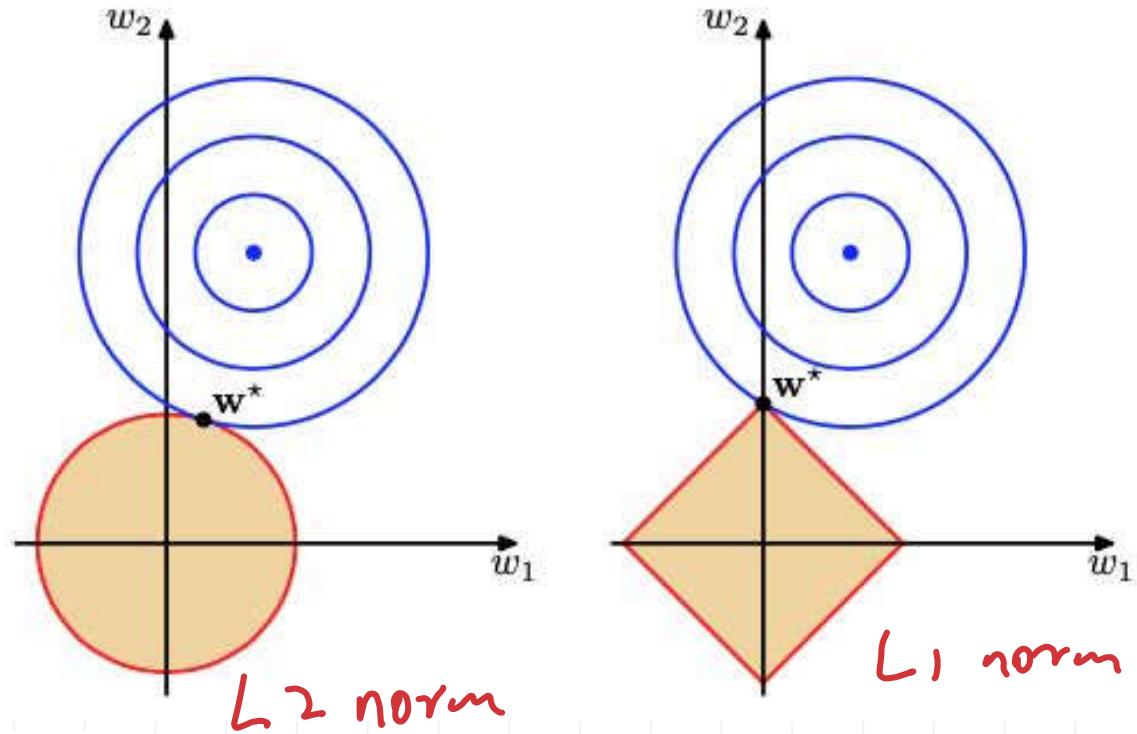
**Figure E.1** A geometrical picture of the technique of Lagrange multipliers in which we seek to maximize a function  $f(\mathbf{x})$ , subject to the constraint  $g(\mathbf{x}) = 0$ . If  $\mathbf{x}$  is  $D$  dimensional, the constraint  $g(\mathbf{x}) = 0$  corresponds to a subspace of dimensionality  $D - 1$ , indicated by the red curve. The problem can be solved by optimizing the Lagrangian function  $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$ .



**Figure E.2** A simple example of the use of Lagrange multipliers in which the aim is to maximize  $f(x_1, x_2) = 1 - x_1^2 - x_2^2$  subject to the constraint  $g(x_1, x_2) = 0$  where  $g(x_1, x_2) = x_1 + x_2 - 1$ . The circles show contours of the function  $f(x_1, x_2)$ , and the diagonal line shows the constraint surface  $g(x_1, x_2) = 0$ .



**Figure 3.4** Plot of the contours of the unregularized error function (blue) along with the constraint region (3.30) for the quadratic regularizer  $q = 2$  on the left and the lasso regularizer  $q = 1$  on the right, in which the optimum value for the parameter vector  $w$  is denoted by  $w^*$ . The lasso gives a sparse solution in which  $w_1^* = 0$ .



$$\frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2} \sum_{j=1}^M |w_j|^q$$

# Multiple regression outputs

See book

# What we did so far

Why linear models?

Input data, features, basis functions

Maximum likelihood and least squares

Geometric intuition

Regularised least squares

Multiple outputs

Bias-variance decomposition

# Bias-variance: the cartoon view

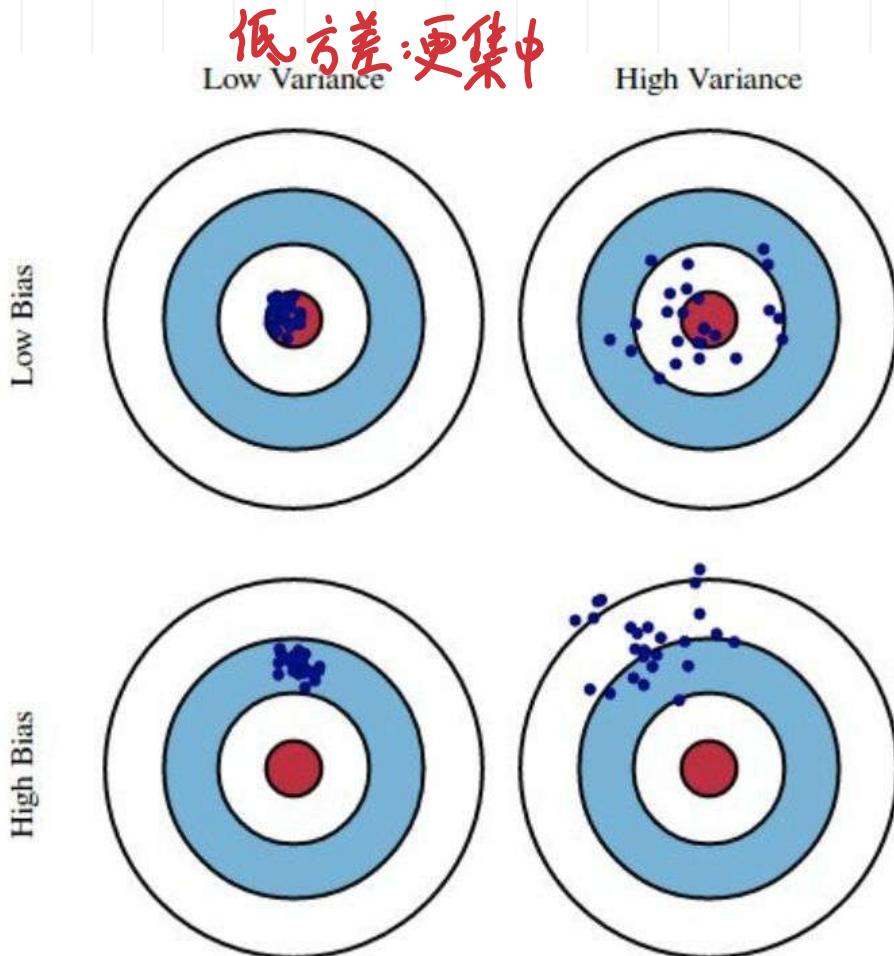


Fig. 1: Graphical illustration of bias and variance  
From Understanding the Bias-Variance Tradeoff, by Scott Fortmann-Roe.

What are the sources of different dots in this picture?

What remains constant: learning target, model specification, learning algorithm.

What changes:  
Data (subsets), randomness in learning (including but are not limited to initialisations), ...

# Expected square loss

(See Sec 1.5.5)

**Figure 1.28**

The regression function  $y(x)$ , which minimizes the expected squared loss, is given by the mean of the conditional distribution  $p(t|x)$ .

$$y(\mathbf{x}) = \mathbb{E}_t[t|\mathbf{x}]$$

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

for of  $\underline{x}$ , not  $t$

(3.36)

$$\mathbb{E}[L] = \iint \{y(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt.$$

$\uparrow$   
 $\mathbf{x}, t$

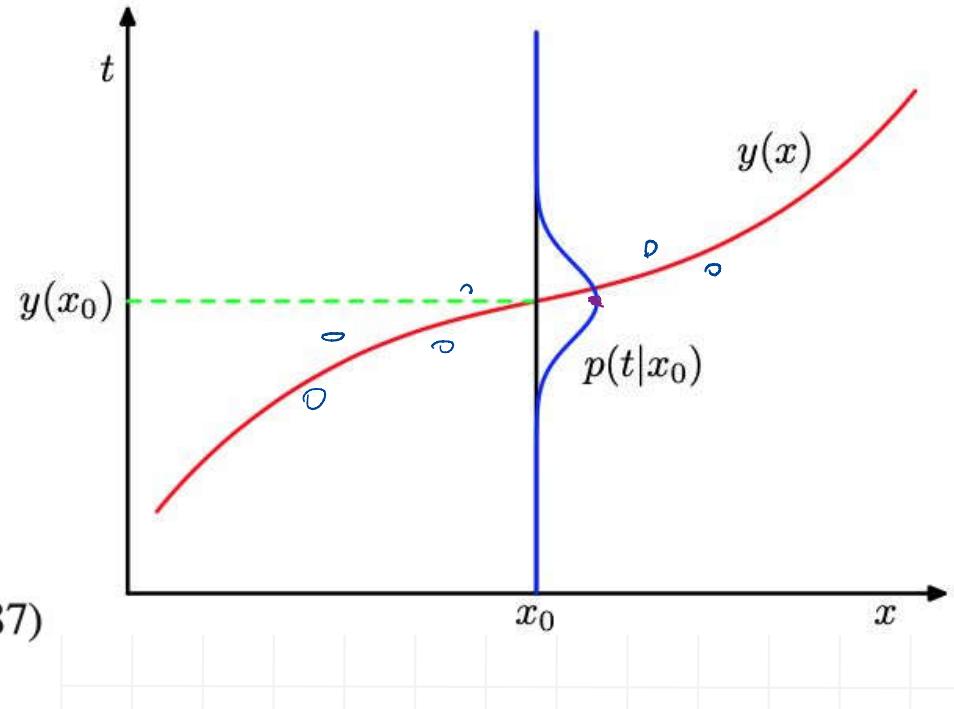
(1.87)

$$\begin{aligned} \{y(\mathbf{x}) - t\}^2 &= \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}] + \mathbb{E}[t|\mathbf{x}] - t\}^2 \\ &= \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 + 2\{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}\{\mathbb{E}[t|\mathbf{x}] - t\} + \{\mathbb{E}[t|\mathbf{x}] - t\}^2 \end{aligned}$$

$$\mathbb{E}[L] = \int \{y(\mathbf{x}) - h(\mathbf{x})\}^2 p(\mathbf{x}) d\mathbf{x} + \int \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt.$$

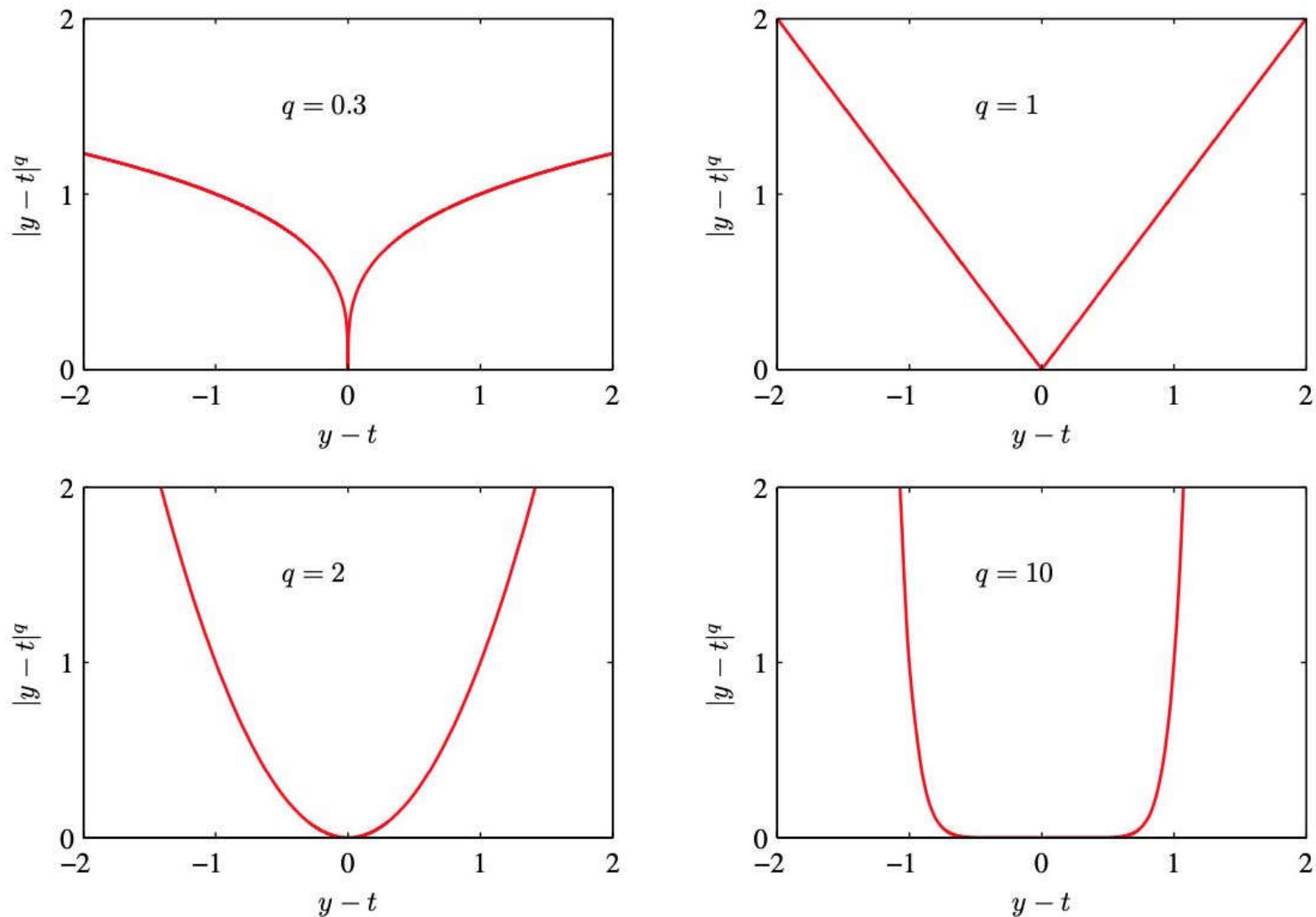
*noise*

(3.37)



$$\mathbb{E}[L] = \int \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 p(\mathbf{x}) d\mathbf{x} + \int \{\mathbb{E}[t|\mathbf{x}] - t\}^2 p(\mathbf{x}) d\mathbf{x}. \quad (1.90)$$

## Generalising squared loss to Mikowski distances



**Figure 1.29** Plots of the quantity  $L_q = |y - t|^q$  for various values of  $q$ .

$$\mathbb{E}[L] = \int \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 p(\mathbf{x}) d\mathbf{x} + \int \{\mathbb{E}[t|\mathbf{x}] - t\}^2 p(\mathbf{x}) d\mathbf{x}. \quad (1.90)$$

$$h(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}] = \int tp(t|\mathbf{x}) dt. \quad (3.36)$$

$$\mathbb{E}[L] = \int \{y(\mathbf{x}) - h(\mathbf{x})\}^2 p(\mathbf{x}) d\mathbf{x} + \int \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt. \quad (3.37)$$

$$\mathbb{E}[L] = \int_{\mathbb{R}^d} \{y(\mathbf{x}) - h(\mathbf{x})\}^2 p(\mathbf{x}) d\mathbf{x} + \int \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt. \quad (3.37)$$

↑  
Int. over both  $x$  &  $t$

Now introduce data  $D$

$$\{y(\mathbf{x}; D) - h(\mathbf{x})\}^2. \quad (3.38)$$

$$\begin{aligned} &= \{y(\mathbf{x}; D) - \mathbb{E}_D[y(\mathbf{x}; D)] + \mathbb{E}_D[y(\mathbf{x}; D)] - h(\mathbf{x})\}^2 \\ &= \{y(\mathbf{x}; D) - \mathbb{E}_D[y(\mathbf{x}; D)]\}^2 + \{\mathbb{E}_D[y(\mathbf{x}; D)] - h(\mathbf{x})\}^2 \\ &\quad + 2\{y(\mathbf{x}; D) - \mathbb{E}_D[y(\mathbf{x}; D)]\}\{\mathbb{E}_D[y(\mathbf{x}; D)] - h(\mathbf{x})\}. \end{aligned} \quad (3.39)$$

Int. over  $\mathbf{x}$  //

Q: how does one know to add+subtract  $\mathbb{E}_D[y(\mathbf{x}; D)]$ ?

$$\begin{aligned} &\mathbb{E}_D [\{y(\mathbf{x}; D) - h(\mathbf{x})\}^2] \\ &= \underbrace{\{\mathbb{E}_D[y(\mathbf{x}; D)] - h(\mathbf{x})\}^2}_{\text{(bias)}^2} + \underbrace{\mathbb{E}_D [\{y(\mathbf{x}; D) - \mathbb{E}_D[y(\mathbf{x}; D)]\}^2]}_{\text{variance}}. \end{aligned} \quad (3.40)$$

$$\mathbb{E}[L] = \int \{y(\mathbf{x}) - h(\mathbf{x})\}^2 p(\mathbf{x}) d\mathbf{x} + \int \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt. \quad (3.37)$$



$$\mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2]$$

$$= \underbrace{\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2}_{\text{(bias)}^2} + \underbrace{\mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2]}_{\text{variance}}. \quad (3.40)$$

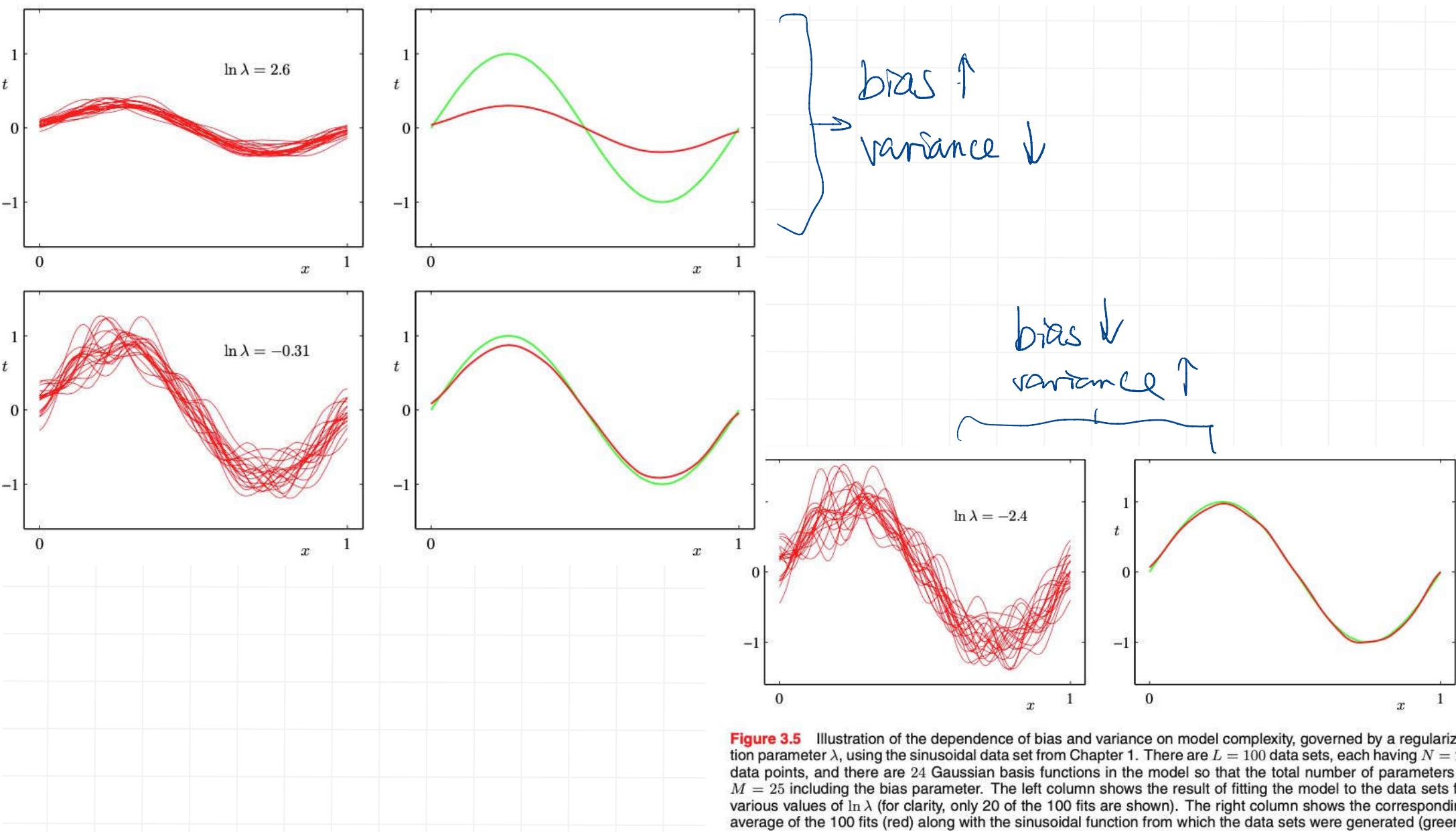
$$\text{expected loss} = \text{(bias)}^2 + \text{variance} + \text{noise} \quad (3.41)$$

where

$$(\text{bias})^2 = \int \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 p(\mathbf{x}) d\mathbf{x} \quad (3.42)$$

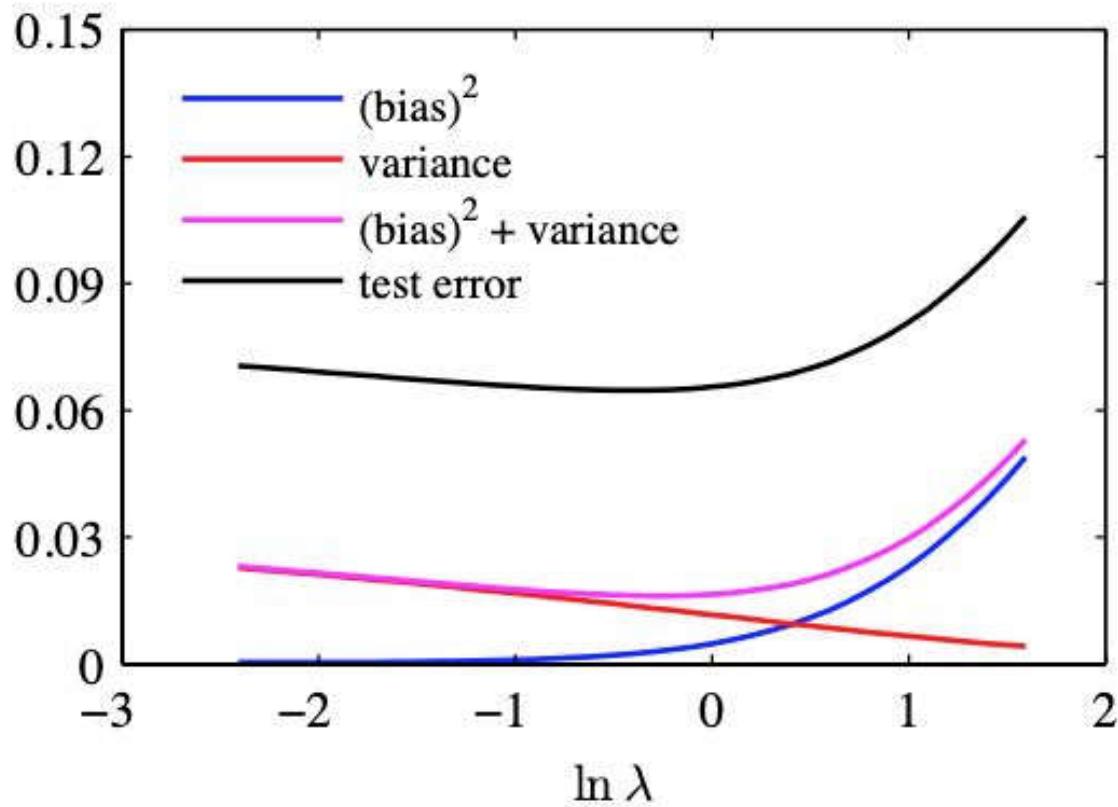
$$\text{variance} = \int \mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2] p(\mathbf{x}) d\mathbf{x} \quad (3.43)$$

$$\text{noise} = \int \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt \quad (3.44)$$



- Dependence of bias and variance on the model complexity
- Squared bias, variance, their sum, and test data
- The minimum for  $(\text{bias})^2 + \text{variance}$  occurs close to the value that gives the minimum error

Fig 3.6



# Unbiased estimators

- You may have encountered *unbiased estimators*
- Why guarantee zero bias? To quote the pioneer of Bayesian inference, Edwin Jaynes, from his book *Probability Theory: The Logic of Science* (2003):

Why do they do this? Why do orthodoxians put such exaggerated emphasis on bias? We suspect that the main reason is simply that they are caught in a **psycho-semantic trap** of their own making. When we call the quantity  $(\langle \beta \rangle - \alpha)$  the “bias”, that makes it sound like something awfully reprehensible, which we must get rid of at all costs. If it had been called instead the **“component of error orthogonal to the variance”**, as suggested by the Pythagorean form of (17–2), it would have been clear to all that these two contributions to the error are on an equal footing; it is folly to decrease one at the expense of increasing the other. This is just the price one pays for choosing a technical terminology that carries an emotional load, implying value judgments; orthodoxy falls constantly into this tactical error.

# The bias-variance decomposition

- Tradeoff between bias and variance
  - simple models have low variance and high bias
  - complex models have high variance and low bias
- The sum of bias and variance has a minimum at a certain model complexity.
- Expected loss  $\mathbb{E}_{\mathcal{D}} [L]$  over all data sets  $\mathcal{D}$

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

- The noise comes from the data, and can not be removed from the expected loss.
- To analyse the bias-variance decomposition : many data sets needed, which are not always available.

# “explainable models”? Interpretable models

Dawes, Robyn M.. “The robust beauty of improper linear models in decision making.” *American Psychologist* 34 (1979): 571-582.

Ustun, Berk and Cynthia Rudin. “Supersparse linear integer models for optimized medical scoring systems.” *Machine Learning* 102 (2015): 349-391.

Rudin, Cynthia. “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead.” *Nature Machine Intelligence* 1 (2019): 206-215.



Cynthia Rudin, Duke University

2022 Squirrel Prize in AI for Humanity

<https://aaai.org/Awards/squirrel-ai-award.php>

For pioneering scientific work in the area of interpretable and transparent AI systems in real-world deployments, the advocacy for these features in highly sensitive areas such as social justice and medical diagnosis, and serving as a role model for researchers and practitioners.

Video: <https://www.youtube.com/watch?v=PwLN5irdMT8>

- An interpretable machine learning model obeys a domain-specific set of constraints that makes its computations easier to understand.
- My technical definition: An interpretable machine learning model is constrained in model form so that it is either useful to someone, or obeys structural knowledge of the domain, such as monotonicity, causality, structural (generative) constraints, additivity, or physical constraints that come from domain knowledge.
- There's a spectrum.

# Example: super-sparse linear models

Slide by Cynthia Rudin

- 2HELP2B was not created by doctors
- It is a ML model
- It is just as accurate as black box models.
- Doctors can decide themselves whether to trust it
- Doctors can calibrate the score with information not in the database

## 2HELP2B

1.	Any cEEG Pattern with Frequency <b>2 Hz</b>	1 point	...
2.	Epileptiform Discharges	1 point	+
3.	Patterns include [LPD, LRDA, BIPD]	1 point	+
4.	Patterns Superimposed with Fast or Sharp Activity	1 point	+
5.	Prior Seizure	1 point	+
6.	Brief Rhythmic Discharges	2 points	+
<b>SCORE</b>			...

SCORE	0	1	2	3	4	5	6+
RISK	<5%	11.9%	26.9%	50.0%	73.1%	88.1%	95.3%

## Risk-Calibrated Supersparse Linear Integer Models (Risk-SLIM)

(Ustun, R, 2019)

$$\min_{\lambda \in L} \sum_{i=1}^n \log(1 + e^{-y_i \mathbf{x}_i^\top \lambda}) + C \|\lambda\|_0$$

$\lambda \in L$  means that  $\forall j, \lambda_j \in \{-10, -9, \dots, 0, \dots, 9, 10\}$

MINLP – really hard...

Small Integer Coefficients

# So far...

- 2HELP2B validated on independent multicenter cohort (N=2111)
- Implemented: University of Wisconsin, Massachusetts General Hospital/Harvard Medical School
- Ongoing implementation: Emory University, Duke University, Medical University of South Carolina, Free University of Brussels (Belgium)
- Resulted in **63.6%** reduction in duration of EEG monitoring per patient
  - \$1,134.831 saving per patient<sup>1</sup>
- **2.82 X** More Patients Monitored
- **\$6.1M** estimated savings in FY 2018 at MGH,UW

<sup>1</sup>2016 Medicare Reimbursement Most Common Professional Code

# Linear models for regression 1

Why linear models?

Input data, features, basis functions

Maximum likelihood and least squares

Geometric intuition

Regularised least squares

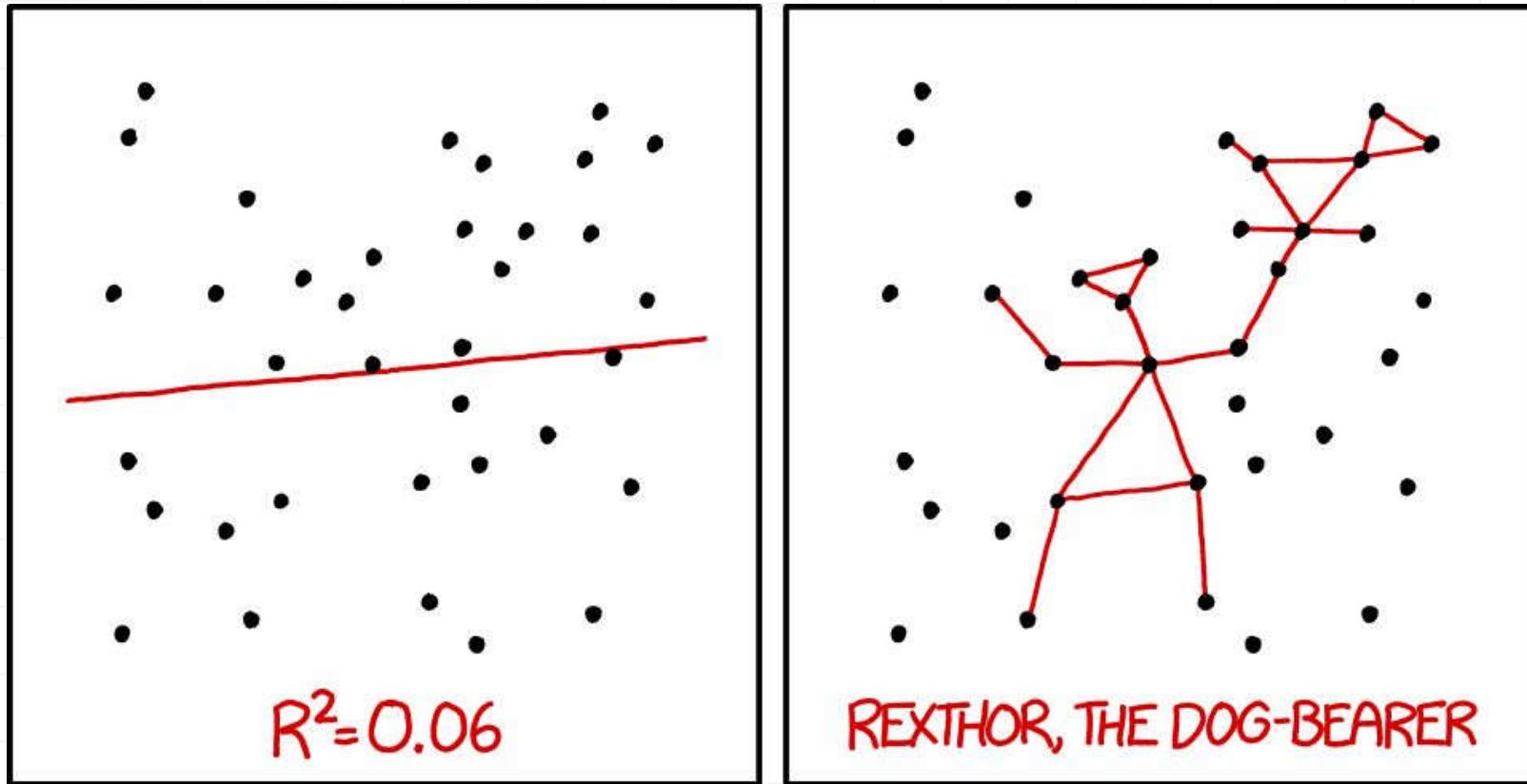
Multiple outputs

Bias-variance decomposition

The relation between MLE and least squares, Lagrange multipliers, multiple ways of looking at linear models.



<https://xkcd.com/1725/>



I DON'T TRUST LINEAR REGRESSIONS WHEN IT'S HARDER  
TO GUESS THE DIRECTION OF THE CORRELATION FROM THE  
SCATTER PLOT THAN TO FIND NEW CONSTELLATIONS ON IT.

# Announcements

Mon 14 Mar week 4 - no lecture, Canberra day!

Assignment 1, submit pdf + code – gradescope entry will be up.

More about this 1630-1700 today

- You're encouraged to typeset the solution with Latex, since making your computer "speak math" (and manage bibliography) is one of the important skills for AI/ML. We liken this to using source control in COMP1100/1110.
  - 2 bonus marks for solutions made with Latex
- Grading expectations for COMP4670 students – max{COMP4670 scheme, COMP8600 scheme}

# Linear models for regression 2

We covered:

- Basis functions
- Maximum Likelihood with Gaussian Noise
- Regularisation
- Bias variance decomposition

Today:

Conditional gaussians

Bayesian regression

Predictive distributions

Curse of dimensionality

Info theory 101

Mon

$$\mathbf{w} = (\lambda \mathbf{I} + \Phi^T \Phi)^{-1} \Phi^T \mathbf{t}. \quad (3.28)$$

We'll get there via 3 stepping stones - conjugate prior, conditional Gaussian, Bayes theorem for Gaussian variables

## *Definition ( Conjugate Prior)*

A class of prior probability distributions  $p(w)$  is conjugate to a class of likelihood functions  $p(x | w)$  if the resulting posterior distributions  $p(w | x)$  are in the same family as  $p(w)$ .

*Table:* Discrete likelihood distributions

Likelihood	Conjugate Prior
Bernoulli	Beta
Binomial	Beta
Poisson	Gamma
Multinomial	Dirichlet

$$\begin{aligned} p(w) \\ p(w|x) \end{aligned}$$

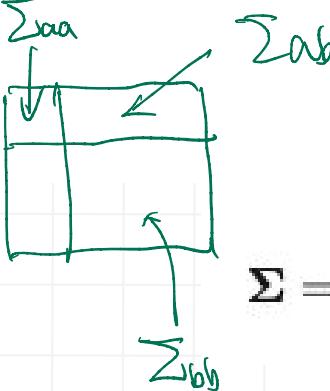
*Table:* Continuous likelihood distributions

Likelihood	Conjugate Prior
Uniform	Pareto
Exponential	Gamma
Normal	Normal (mean parameter)
Multivariate normal	Multivariate normal (mean parameter)

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\} \quad (2.43)$$

## Partitioned Gaussians

Given a joint Gaussian distribution  $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$  with  $\boldsymbol{\Lambda} \equiv \boldsymbol{\Sigma}^{-1}$  and



The diagram shows a 2x2 matrix with four entries labeled  $\Sigma_{aa}$ ,  $\Sigma_{ab}$ ,  $\Sigma_{ba}$ , and  $\Sigma_{bb}$ . A green bracket on the left side spans both rows and is labeled  $\Sigma_{aa}$ . A green bracket at the top spans both columns and is labeled  $\Sigma_{ab}$ . A green bracket on the bottom spans both columns and is labeled  $\Sigma_{bb}$ . A green bracket on the right side spans both rows and is labeled  $\Sigma_{ba}$ .

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{pmatrix}, \quad \boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{pmatrix} \quad (2.94)$$

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{pmatrix}, \quad \boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{pmatrix}. \quad (2.95)$$

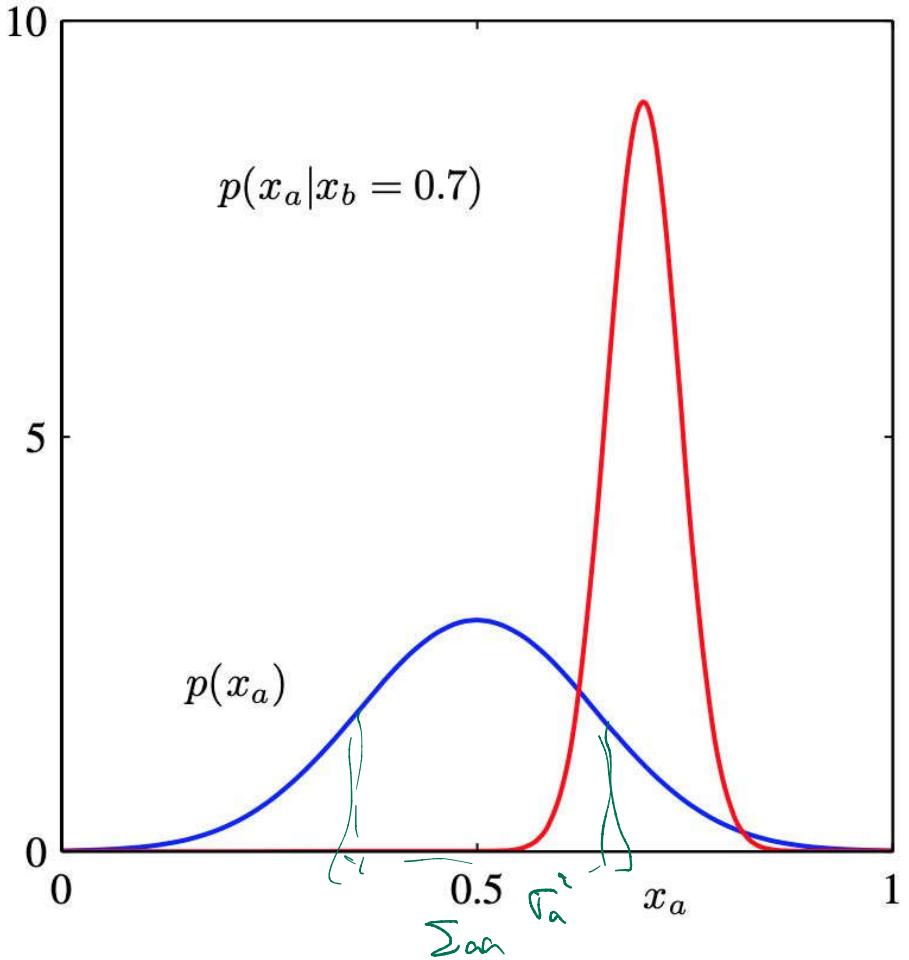
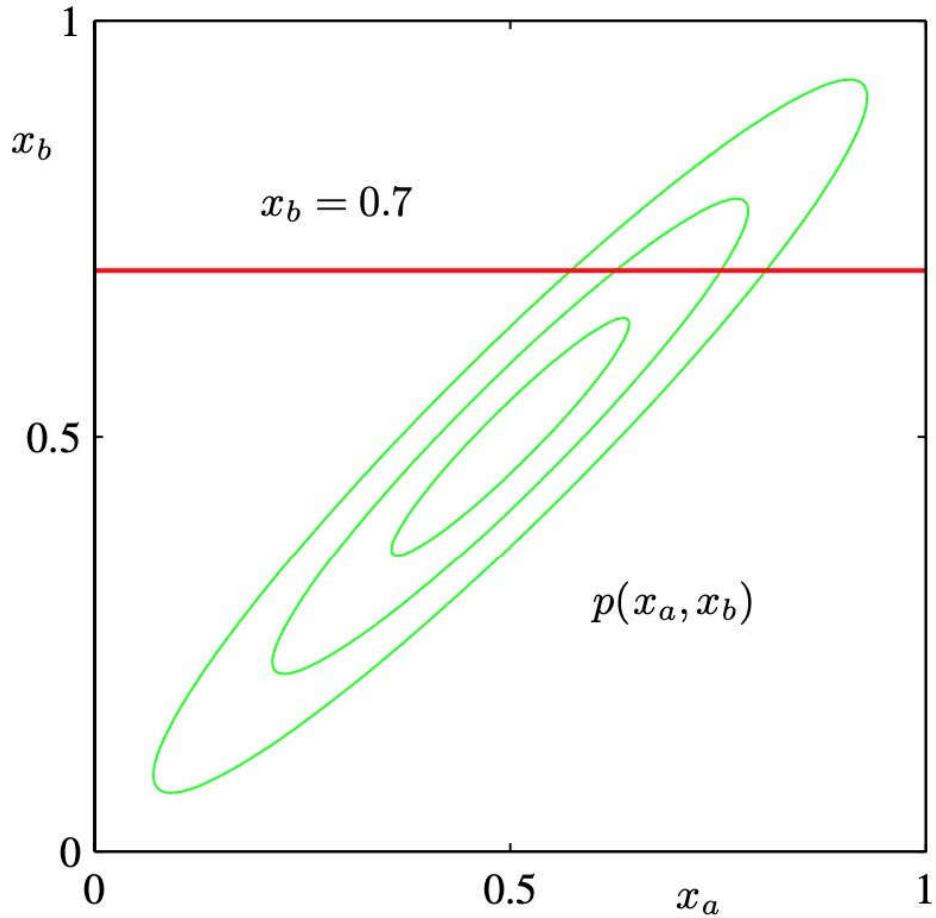
Marginal distribution:  $p(\mathbf{x}_a) = \int p(\mathbf{x}_a, \mathbf{x}_b) d\mathbf{x}_b$

$$p(\mathbf{x}_a) = \mathcal{N}(\mathbf{x}_a | \boldsymbol{\mu}_a, \boldsymbol{\Sigma}_{aa}). \quad (2.98)$$

What about the conditional distribution?

$$p(\mathbf{x}_a | \mathbf{x}_b)$$

## Intuitions for the conditional and marginal of jointly Gaussian variables



**Figure 2.9** The plot on the left shows the contours of a Gaussian distribution  $p(x_a, x_b)$  over two variables, and the plot on the right shows the marginal distribution  $p(x_a)$  (blue curve) and the conditional distribution  $p(x_a|x_b)$  for  $x_b = 0.7$  (red curve).

## Covariance and precision matrices

$$\begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}^{-1} = \begin{pmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{pmatrix} \quad (2.78)$$

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{M} & -\mathbf{MBD}^{-1} \\ -\mathbf{D}^{-1}\mathbf{CM} & \mathbf{D}^{-1} + \mathbf{D}^{-1}\mathbf{CMBD}^{-1} \end{pmatrix} \quad (2.76)$$

where we have defined

$$\mathbf{M} = \underbrace{(\mathbf{A} - \mathbf{BD}^{-1}\mathbf{C})^{-1}}_{(2.77)}.$$

The quantity  $\mathbf{M}^{-1}$  is known as the *Schur complement* of the matrix on the left-hand side of (2.76) with respect to the submatrix  $\mathbf{D}$ .

$$\Lambda_{aa} = (\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba})^{-1} \quad (2.79)$$

$$\Lambda_{ab} = -(\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba})^{-1}\Sigma_{ab}\Sigma_{bb}^{-1}. \quad (2.80)$$

## “Completing the square”

A useful view for manipulating Gaussian distributions

$$\log ( N(x | \mu, \Sigma) ) = \text{const} + \log ( \exp ( \text{quadratic expression of } x ) )$$

$$-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu) = -\frac{1}{2}x^T \Sigma^{-1} x + x^T \Sigma^{-1} \mu + \text{const} \quad (2.71)$$

$$\begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}^{-1} = \begin{pmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{pmatrix} \quad (2.78)$$

## “Completing the square”

$$\Lambda_{aa} = (\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba})^{-1} \quad (2.79)$$

$$\begin{aligned} -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}) &= \text{quadratic in } \mathbf{x}_a \\ -\frac{1}{2}(\mathbf{x}_a - \boldsymbol{\mu}_a)^T \Lambda_{aa}(\mathbf{x}_a - \boldsymbol{\mu}_a) - \frac{1}{2}(\mathbf{x}_a - \boldsymbol{\mu}_a)^T \Lambda_{ab}(\mathbf{x}_b - \boldsymbol{\mu}_b) &\rightarrow \text{mean in } \mathbf{x}_a \\ -\frac{1}{2}(\mathbf{x}_b - \boldsymbol{\mu}_b)^T \Lambda_{ba}(\mathbf{x}_a - \boldsymbol{\mu}_a) - \frac{1}{2}(\mathbf{x}_b - \boldsymbol{\mu}_b)^T \Lambda_{bb}(\mathbf{x}_b - \boldsymbol{\mu}_b). &\quad (2.70) \\ &\text{mean in } \mathbf{x}_a \\ &\text{const.} \end{aligned}$$

Consider

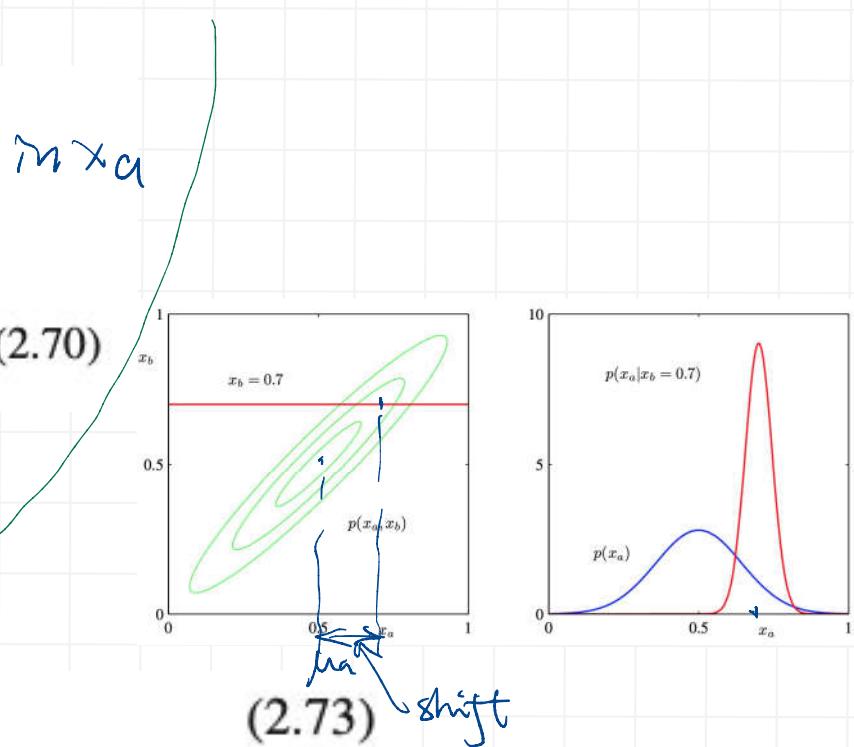
$$p(\mathbf{x}_a | \mathbf{x}_b) \quad \Lambda_{a|b} = \Lambda_{aa} \xleftarrow{\text{const. } X_b}$$

From the quadratic term:

$$\Sigma_{a|b} = \Lambda_{aa}^{-1}.$$

From the linear term:

$$\begin{aligned} \boldsymbol{\mu}_{a|b} &= \Sigma_{a|b} \{ \Lambda_{aa} \boldsymbol{\mu}_a - \Lambda_{ab} (\mathbf{x}_b - \boldsymbol{\mu}_b) \} \\ &= \boldsymbol{\mu}_a - \underbrace{\Lambda_{aa}^{-1} \Lambda_{ab} (\mathbf{x}_b - \boldsymbol{\mu}_b)}_{\propto \text{Var } a} \xrightarrow{\text{shift in mean}} \text{how far } \mathbf{x}_b \text{ is from its mean} \quad (2.75) \end{aligned}$$



Conditional distribution:

$$(\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba})$$

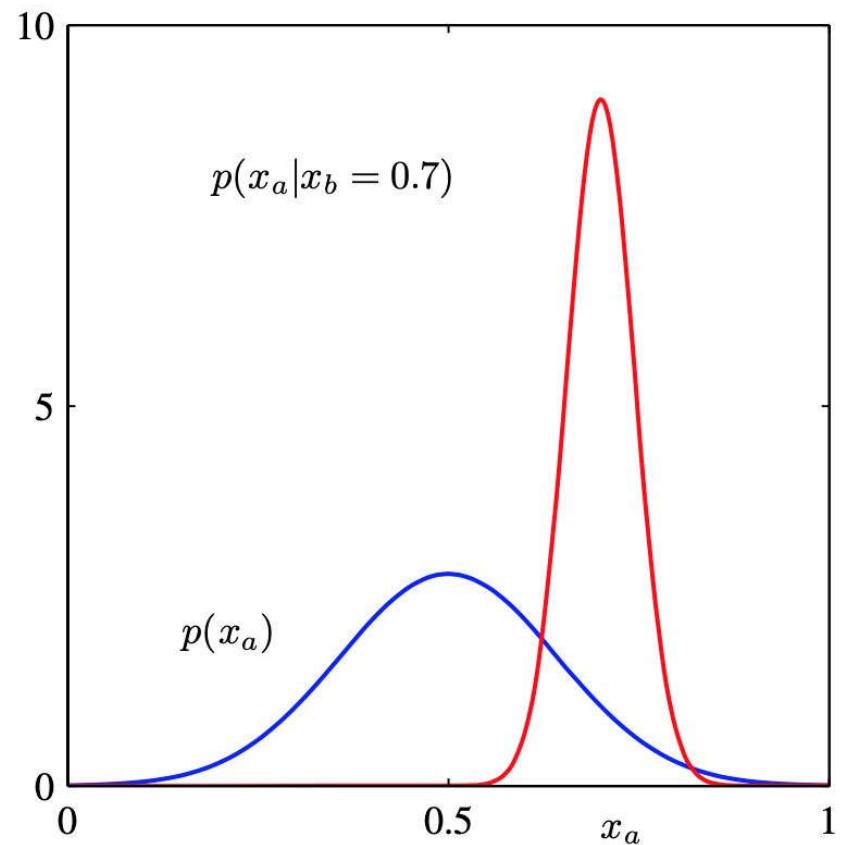
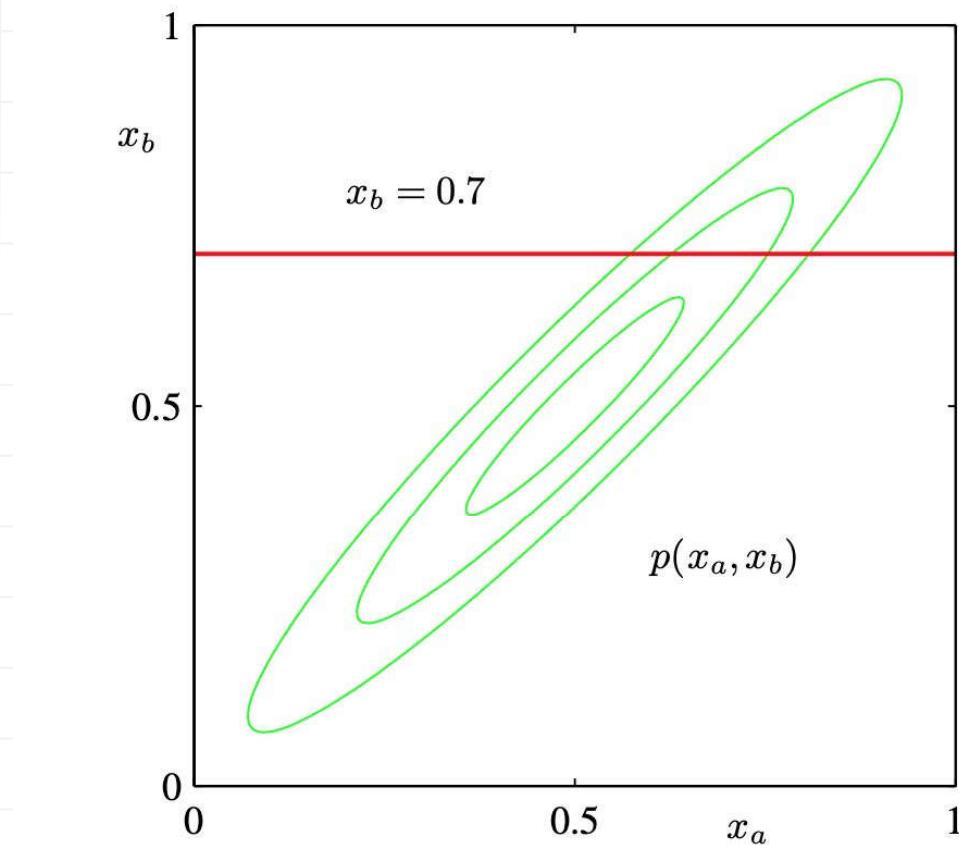
$$p(\mathbf{x}_a|\mathbf{x}_b) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{a|b}, \boldsymbol{\Lambda}_{aa}^{-1}) \quad (2.96)$$

$$\boldsymbol{\mu}_{a|b} = \boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{aa}^{-1}\boldsymbol{\Lambda}_{ab}(\mathbf{x}_b - \boldsymbol{\mu}_b). \quad (2.97)$$

Marginal distribution:

$$p(\mathbf{x}_a) = \mathcal{N}(\mathbf{x}_a|\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_{aa}). \quad (2.98)$$

$$\begin{aligned}
 p(\mathbf{x}_a | \mathbf{x}_b) &= \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_{a|b}, (\boldsymbol{\Sigma}_{aa} - \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}\boldsymbol{\Sigma}_{ba})^{-1}) \\
 \boldsymbol{\mu}_{a|b} &= \boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{aa}^{-1}\boldsymbol{\Lambda}_{ab}(\mathbf{x}_b - \boldsymbol{\mu}_b).
 \end{aligned}$$



**Figure 2.9** The plot on the left shows the contours of a Gaussian distribution  $p(x_a, x_b)$  over two variables, and the plot on the right shows the marginal distribution  $p(x_a)$  (blue curve) and the conditional distribution  $p(x_a | x_b)$  for  $x_b = 0.7$  (red curve).

# Bayes theorem for Gaussian variables

Given a marginal Gaussian distribution for  $\mathbf{x}$  and a conditional Gaussian distribution for  $\mathbf{y}$  given  $\mathbf{x}$  in the form

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}) \quad (2.113)$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\underbrace{\mathbf{Ax} + \mathbf{b}}_{\text{circled}}, \mathbf{L}^{-1}) \quad (2.114)$$

the marginal distribution of  $\mathbf{y}$  and the conditional distribution of  $\mathbf{x}$  given  $\mathbf{y}$  are given by

Hunch :  $\mathbf{My} = \mathbf{A}\boldsymbol{\mu} + \mathbf{b}$        $\sigma_y^2$  : involve  $\boldsymbol{\Lambda}^{-1}, \mathbf{L}^{-1}$

$$p(\mathbf{y}) =$$
$$p(\mathbf{x}|\mathbf{y}) =$$
$$\frac{\mathcal{N}_{\mathbf{y}|\mathbf{x}}}{\sigma_y^2}$$
$$\mathcal{N}_{\mathbf{x}|\mathbf{y}} \quad \mathcal{N}_{\mathbf{A}, \mathbf{b}}$$
$$\boldsymbol{\Lambda}^{-1}, \mathbf{L}^{-1}$$

General direction: find the joint of  $p(\mathbf{x}, \mathbf{y})$ , and then use results from conditional+marginal Gaussians.

$$\mathbf{z} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}$$

$p(\mathbf{x})$	$=$	$\mathcal{N}(\mathbf{x} \boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$
$p(\mathbf{y} \mathbf{x})$	$=$	$\mathcal{N}(\mathbf{y} \mathbf{Ax} + \mathbf{b}, \mathbf{L}^{-1})$

$$\begin{aligned}
 \ln p(\mathbf{z}) &= \ln p(\mathbf{x}) + \ln p(\mathbf{y}|\mathbf{x}) \\
 &= -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Lambda} (\mathbf{x} - \boldsymbol{\mu}) \\
 &\quad -\frac{1}{2}(\mathbf{y} - \mathbf{Ax} - \mathbf{b})^T \mathbf{L} (\mathbf{y} - \mathbf{Ax} - \mathbf{b}) + \underline{\text{const}}
 \end{aligned} \tag{2.102}$$

$$\begin{aligned}
 &-\frac{1}{2}\mathbf{x}^T(\boldsymbol{\Lambda} + \mathbf{A}^T \mathbf{L} \mathbf{A})\mathbf{x} - \frac{1}{2}\mathbf{y}^T \mathbf{L} \mathbf{y} + \frac{1}{2}\mathbf{y}^T \mathbf{L} \mathbf{A} \mathbf{x} + \frac{1}{2}\mathbf{x}^T \mathbf{A}^T \mathbf{L} \mathbf{y} \\
 &= -\frac{1}{2} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}^T \begin{pmatrix} \boldsymbol{\Lambda} + \mathbf{A}^T \mathbf{L} \mathbf{A} & -\mathbf{A}^T \mathbf{L} \\ -\mathbf{L} \mathbf{A} & \mathbf{L} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = -\frac{1}{2} \mathbf{z}^T \mathbf{R} \mathbf{z}
 \end{aligned} \tag{2.103}$$

$$\mathbf{R} = \begin{pmatrix} \boldsymbol{\Lambda} + \mathbf{A}^T \mathbf{L} \mathbf{A} & -\mathbf{A}^T \mathbf{L} \\ -\mathbf{L} \mathbf{A} & \mathbf{L} \end{pmatrix}. \tag{2.104}$$

$$\mathbf{R} = \begin{pmatrix} \mathbf{\Lambda} + \mathbf{A}^T \mathbf{L} \mathbf{\Lambda} & -\mathbf{A}^T \mathbf{L} \\ -\mathbf{L} \mathbf{\Lambda} & \mathbf{L} \end{pmatrix}. \quad (2.104)$$

$$\mathbb{E}[\mathbf{z}] = \mathbf{R}^{-1} \begin{pmatrix} \mathbf{\Lambda}\boldsymbol{\mu} - \mathbf{A}^T \mathbf{L} \mathbf{b} \\ \mathbf{L} \mathbf{b} \end{pmatrix}. \quad \mathbb{E}[\mathbf{z}] = \begin{pmatrix} \boldsymbol{\mu} \\ \mathbf{A}\boldsymbol{\mu} + \mathbf{b} \end{pmatrix}. \quad (2.108)$$

recall

$$\boldsymbol{\Sigma}_{a|b} = \mathbf{\Lambda}_{aa}^{-1}. \quad (2.73)$$

$$\begin{aligned} \boldsymbol{\mu}_{a|b} &= \boldsymbol{\Sigma}_{a|b} \{ \mathbf{\Lambda}_{aa} \boldsymbol{\mu}_a - \mathbf{\Lambda}_{ab} (\mathbf{x}_b - \boldsymbol{\mu}_b) \} \\ &= \boldsymbol{\mu}_a - \mathbf{\Lambda}_{aa}^{-1} \mathbf{\Lambda}_{ab} (\mathbf{x}_b - \boldsymbol{\mu}_b) \end{aligned} \quad (2.75)$$

$$\mathbb{E}[\mathbf{x}|\mathbf{y}] = (\mathbf{\Lambda} + \mathbf{A}^T \mathbf{L} \mathbf{\Lambda})^{-1} \{ \mathbf{A}^T \mathbf{L} (\mathbf{y} - \mathbf{b}) + \mathbf{\Lambda} \boldsymbol{\mu} \} \quad (2.111)$$

$$\text{cov}[\mathbf{x}|\mathbf{y}] = (\mathbf{\Lambda} + \mathbf{A}^T \mathbf{L} \mathbf{\Lambda})^{-1}. \quad (2.112)$$

# Bayes theorem for Gaussian variables

## Marginal and Conditional Gaussians

Given a marginal Gaussian distribution for  $\mathbf{x}$  and a conditional Gaussian distribution for  $\mathbf{y}$  given  $\mathbf{x}$  in the form

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}) \quad (2.113)$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{Ax} + \mathbf{b}, \mathbf{L}^{-1}) \quad (2.114)$$

the marginal distribution of  $\mathbf{y}$  and the conditional distribution of  $\mathbf{x}$  given  $\mathbf{y}$  are given by

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^T) \quad (2.115)$$

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma}\{\mathbf{A}^T\mathbf{L}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Lambda}\boldsymbol{\mu}\}, \boldsymbol{\Sigma}) \quad (2.116)$$

where

$$\boldsymbol{\Sigma} = (\boldsymbol{\Lambda} + \mathbf{A}^T\mathbf{L}\mathbf{A})^{-1}. \quad (2.117)$$

# What we have covered

Prelude: Conjugate prior, conditional of Gaussian variables, Bayes Theorem of Gaussian distributions

**Bayesian regression**

Predictive distributions

Curse of dimensionality

Info theory 101

Goal: estimate the posterior of  $w$ , not just  $w_{ML}$

if

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^N \mathcal{N}(t_n | \mathbf{w}^T \phi(\mathbf{x}_n), \beta^{-1}) \quad (3.10)$$

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \mathbf{m}_0, \mathbf{S}_0) \quad (3.48)$$

then

*posterior after seeing N data points  $\{(x_i, t_i), i=1, \dots, n\}$*

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w} | \underbrace{\mathbf{m}_N}_{\mathbf{m}_0 + \beta \Phi^T \mathbf{t}}, \underbrace{\mathbf{S}_N}_{\mathbf{S}_0^{-1} + \beta \Phi^T \Phi}) \quad (3.49)$$

$$\mathbf{m}_N = \mathbf{S}_N (\mathbf{S}_0^{-1} \mathbf{m}_0 + \beta \Phi^T \mathbf{t}) \quad (3.50)$$

$$\mathbf{S}_N^{-1} = \mathbf{S}_0^{-1} + \beta \Phi^T \Phi. \quad (3.51)$$

## Marginal and Conditional Gaussians

for linear reg,

Given a marginal Gaussian distribution for  $\mathbf{x}$  and a conditional Gaussian distribution for  $\mathbf{y}$  given  $\mathbf{x}$  in the form

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}) \quad (2.113)$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}) \quad (2.114)$$

the marginal distribution of  $\mathbf{y}$  and the conditional distribution of  $\mathbf{x}$  given  $\mathbf{y}$  are given by

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^T) \quad (2.115)$$

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma}\{\mathbf{A}^T\mathbf{L}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Lambda}\boldsymbol{\mu}\}, \boldsymbol{\Sigma}) \quad (2.116)$$

where

$$\boldsymbol{\Sigma} = (\boldsymbol{\Lambda} + \mathbf{A}^T\mathbf{L}\boldsymbol{\Lambda})^{-1}. \quad (2.117)$$

assume

then

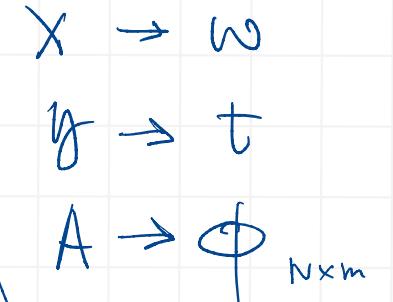
$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^N \mathcal{N}(t_n | \mathbf{w}^T \phi(\mathbf{x}_n), \underline{\beta^{-1}}) \quad (3.10)$$

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \mathbf{m}_0, \underline{\mathbf{S}_0}) \quad (3.48)$$

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w} | \mathbf{m}_N, \mathbf{S}_N) \quad (3.49)$$

$$\mathbf{m}_N = \mathbf{S}_N (\mathbf{S}_0^{-1} \mathbf{m}_0 + \beta \boldsymbol{\Phi}^T \mathbf{t}) \quad (3.50)$$

$$\mathbf{S}_N^{-1} = \mathbf{S}_0^{-1} + \beta \boldsymbol{\Phi}^T \boldsymbol{\Phi}. \quad (3.51)$$



## Isotropic Gaussian prior

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N) \quad (3.49)$$

general prior  
 $p(\omega) \sim N(m_0, S_0)$

$$\mathbf{m}_N = \underline{\mathbf{S}_N} (\underline{\mathbf{S}_0^{-1}} \mathbf{m}_0 + \beta \Phi^T \mathbf{t}) \quad (3.50)$$

$$\underline{\mathbf{S}_N^{-1}} = \underline{\mathbf{S}_0^{-1}} + \beta \Phi^T \Phi \quad (3.51)$$

$\uparrow$  "data covariance"  
 fixed, doesn't change after seeing data.

$\leftarrow$  prior  $\uparrow$  if  $N$  small  
 data  $\uparrow$  if  $N$  large.

$$\leftarrow p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1} \mathbf{I}) \quad (3.52)$$

$$\mathbf{m}_N = \beta \mathbf{S}_N \Phi^T \mathbf{t} \quad (3.53)$$

$$\mathbf{S}_N^{-1} = (\alpha \mathbf{I} + \beta \Phi^T \Phi)^{-1} \quad (3.54)$$

$$\mathbf{w}_{ML} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{t} \quad (3.15)$$

## Bayesian Regression

- Log of posterior is sum of log likelihood and log of prior

$$\ln p(\mathbf{w} | \mathbf{t}) = -\beta \underbrace{\frac{1}{2} \|\mathbf{t} - \Phi \mathbf{w}\|^2}_{\text{sum-of-squares-error}} - \frac{\alpha}{2} \underbrace{\|\mathbf{w}\|^2}_{\text{regulariser}} + \text{const.}$$

- The *maximum a posteriori* estimator

$$\mathbf{w}_{\text{m.a.p.}} = \arg \max_{\mathbf{w}} p(\mathbf{w} | \mathbf{t})$$

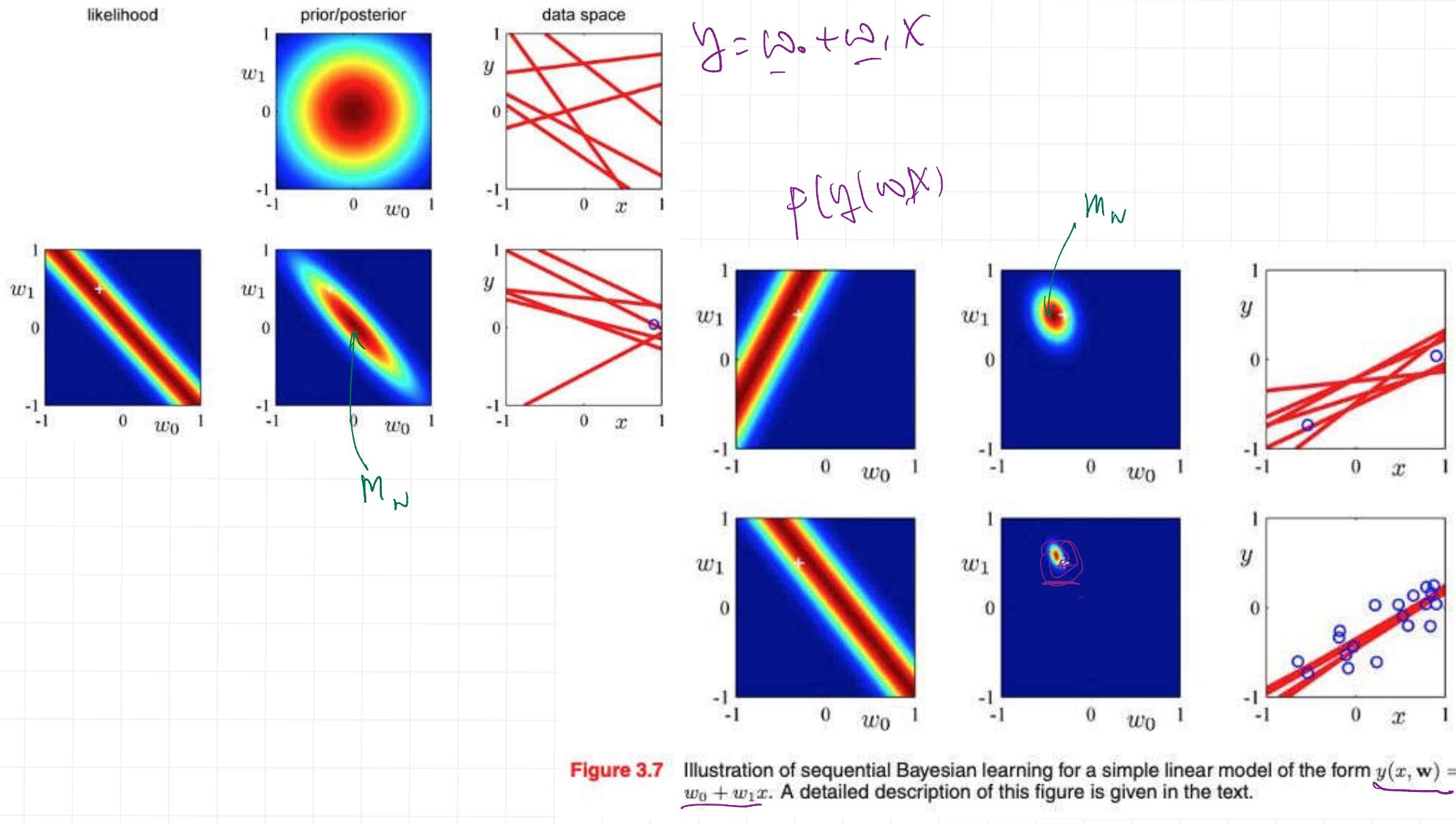
corresponds to minimising the sum-of-squares error function with quadratic regularisation coefficient  $\lambda = \alpha/\beta$ .

- The posterior is Gaussian so mode = mean:  $\mathbf{w}_{\text{m.a.p.}} = \mathbf{m}_N$ .
- For  $\alpha \ll \beta$  we recover unregularised least squares (equivalently m.a.p. approaches maximum likelihood), for example in case of
  - an infinitely broad prior with  $\alpha \rightarrow 0$
  - an infinitely precise likelihood with  $\beta \rightarrow \infty$

$$\mathbf{m}_N = \underbrace{\beta \mathbf{S}_N \Phi^T \mathbf{t}}_{\text{unregularised least squares}}$$
$$\mathbf{S}_N^{-1} = \underbrace{\alpha \mathbf{I} + \beta \Phi^T \Phi}_{\text{regularisation matrix}}$$



$$\mathbf{w} = \underbrace{(\lambda \mathbf{I} + \Phi^T \Phi)^{-1}}_{\text{posterior covariance}} \underbrace{\Phi^T \mathbf{t}}_{\text{posterior mean}}$$



**Figure 3.7** Illustration of sequential Bayesian learning for a simple linear model of the form  $y(x, w) = \underline{w_0} + \underline{w_1}x$ . A detailed description of this figure is given in the text.

# What we have covered

Prelude: Conjugate prior, conditional of Gaussian variables, Bayes Theorem of Gaussian distributions

Bayesian regression

Predictive distributions

Curse of dimensionality

Info theory 101

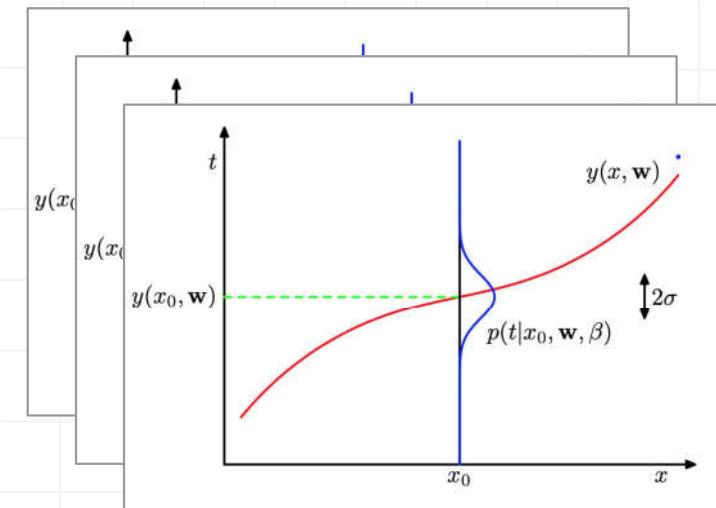
# Predictive distribution

Goal of regression: estimate  $y(x; w)$  at unobserved values of  $x$

$$p(t|\mathbf{t}, \alpha, \beta) = \int p(t|\mathbf{w}, \beta)p(\mathbf{w}|\mathbf{t}, \alpha, \beta) d\mathbf{w} \quad (3.57)$$

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}). \quad (3.8)$$

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N) \quad (3.49)$$

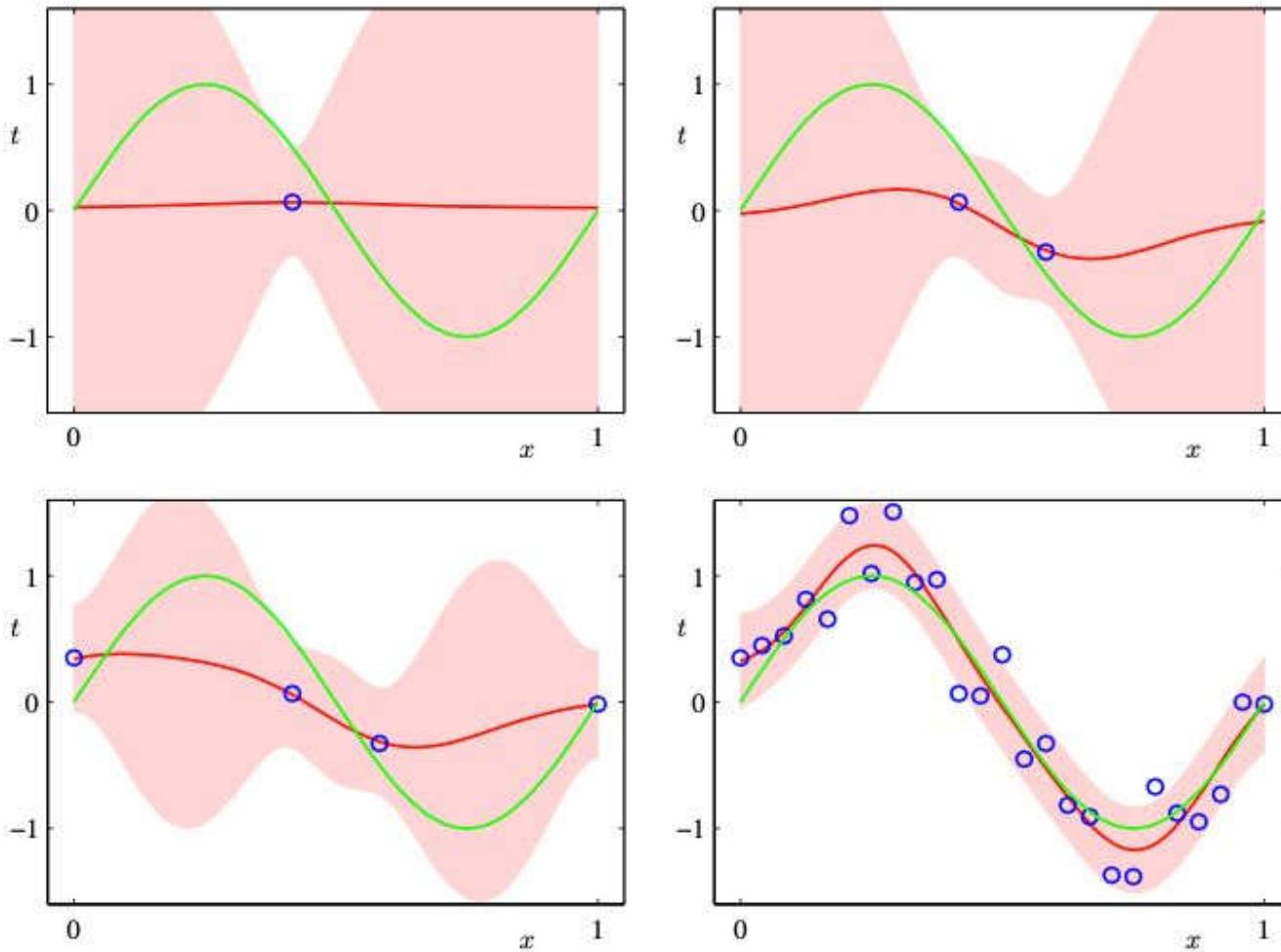


$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\Lambda^{-1}\mathbf{A}^T) \quad (2.115)$$

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) \quad (3.52)$$

$$p(t|\mathbf{x}, \mathbf{t}, \alpha, \beta) = \mathcal{N}(t|\mathbf{m}_N^T \phi(\mathbf{x}), \sigma_N^2(\mathbf{x})) \quad (3.58)$$

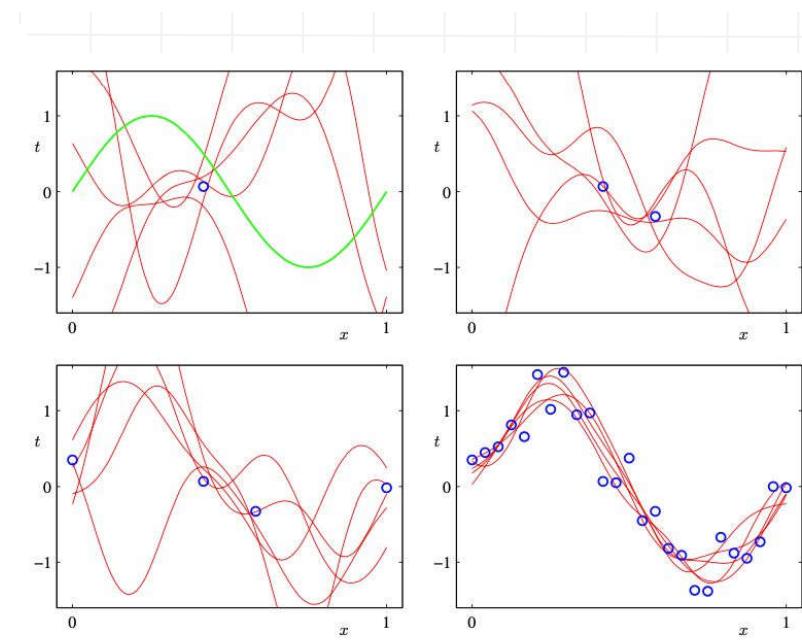
$$\sigma_N^2(\mathbf{x}) = \frac{1}{\beta} + \phi(\mathbf{x})^T \mathbf{S}_N \phi(\mathbf{x}). \quad (3.59)$$



**Figure 3.8** Examples of the predictive distribution (3.58) for a model consisting of 9 Gaussian basis functions of the form (3.4) using the synthetic sinusoidal data set of Section 1.1. See the text for a detailed discussion.

$$p(t|\mathbf{x}, \mathbf{t}, \alpha, \beta) = \mathcal{N}(t | \mathbf{m}_N^T \boldsymbol{\phi}(\mathbf{x}), \sigma_N^2(\mathbf{x}))$$

$$\sigma_N^2(\mathbf{x}) = \frac{1}{\beta} + \boldsymbol{\phi}(\mathbf{x})^T \mathbf{S}_N \boldsymbol{\phi}(\mathbf{x}).$$



**Figure 3.9** Plots of the function  $y(x, \mathbf{w})$  using samples from the posterior distributions over  $\mathbf{w}$  corresponding to the plots in Figure 3.8.

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w} | \mathbf{m}_N, \mathbf{S}_N)$$

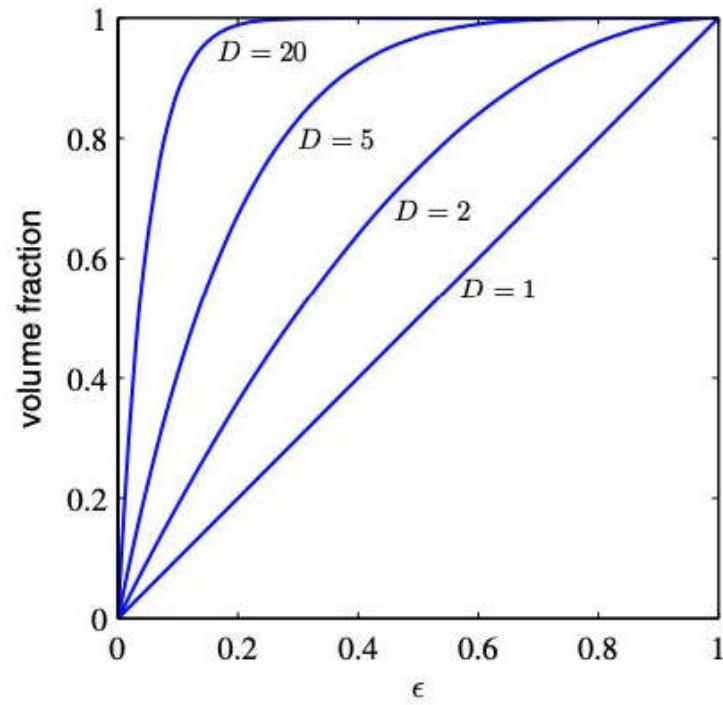
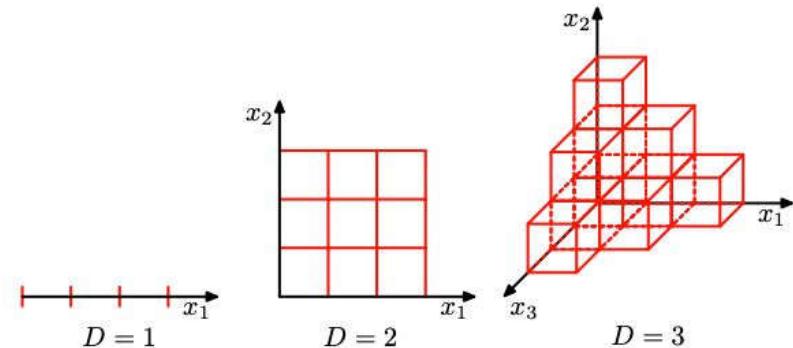
# Curse of dimensionality

- Linear Algebra allows us to operate in  $n$ -dimensional vector spaces using the intuition from our 3-dimensional world as a vector space. No surprises as long as  $n$  is finite.
- If we add more structure to a vector space (e.g. inner product, metric), our intuition gained from the 3-dimensional world around us may be wrong.
- Example: Sphere of radius  $r = 1$ . What is the fraction of the volume of the sphere in a  $D$ -dimensional space which lies between radius  $r = 1$  and  $r = 1 - \epsilon$  ?
- Volume scales like  $r^D$ , therefore the formula for the volume of a sphere is  $V_D(r) = K_D r^D$ .

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

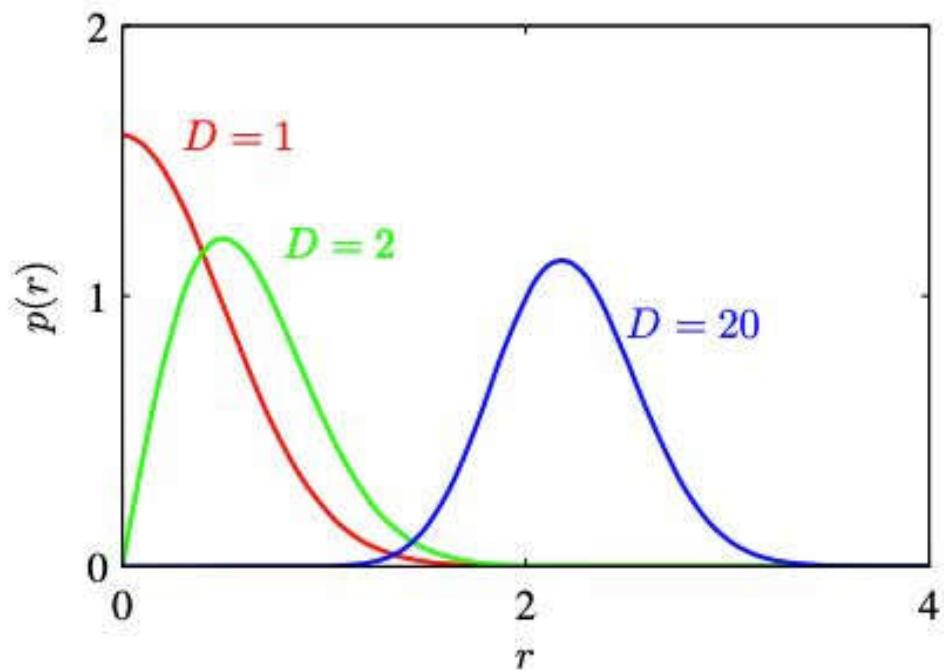
Fig 1.22  
Eq (1.76)

**Figure 1.21** Illustration of the curse of dimensionality, showing how the number of regions of a regular grid grows exponentially with the dimensionality  $D$  of the space. For clarity, only a subset of the cubical regions are shown for  $D = 3$ .



**Figure 1.23**

Plot of the probability density with respect to radius  $r$  of a Gaussian distribution for various values of the dimensionality  $D$ . In a high-dimensional space, most of the probability mass of a Gaussian is located within a thin shell at a specific radius.



$$x_1 = r \cos(\phi) \quad x_2 = r \sin(\phi)$$

$$p(r, \phi | 0, I) = \frac{1}{2\pi} r \exp \left\{ -\frac{1}{2} r^2 \right\}$$

## Discussions on fixed basis functions

**Pros:** assumption of linearity in the parameters (1) led to a range of useful properties including closed-form solutions to the least-squares problem, as well as a tractable Bayesian treatment. (2) for a suitable choice of basis functions, we can model arbitrary nonlinearities in the mapping from input variables to targets.

**Cons:** fixed basis functions before training data set is observed, the number of basis functions needs to grow rapidly, often exponentially, with the dimensionality  $D$  of the input space. Fixes: (1) localized basis functions that scatter only in regions containing data --> RBF networks, NN: adaptive basis functions. (2) target variables may have significant dependence on only a small number of possible directions. NN: choose response directions.

# What we have covered

Prelude: Conjugate prior, conditional of Gaussian variables, Bayes Theorem of Gaussian distributions

**Bayesian regression**

Predictive distributions

Curse of dimensionality

Info theory 101

# Information theory 101

How much information is a random variable  $x$ ?

the amount of information : ‘degree of surprise’ on learning the value of  $x$ ., say,  
expressed in function  $h(x)$

Assumptions:

- $h(x)$  is monotonic in the probability  $p(x)$
- For unrelated  $x \sim p(x)$ , and  $y \sim p(y)$ .  $h(x, y) = h(x) + h(y)$

(Information) Entropy [Shannon, 1948]

$$H[x] = - \sum_x p(x) \log_2 p(x). \quad (1.93)$$

# What is the unit of information entropy? - the origin of bits

Consider random variable x, with 8 equally likely states.

$$H[x] = -8 \times \frac{1}{8} \log_2 \frac{1}{8} = 3 \text{ bits.}$$

How to encode and transmit the value of x in binary digits? {000, 001, 010, 011, 100, 101, 110, 111}

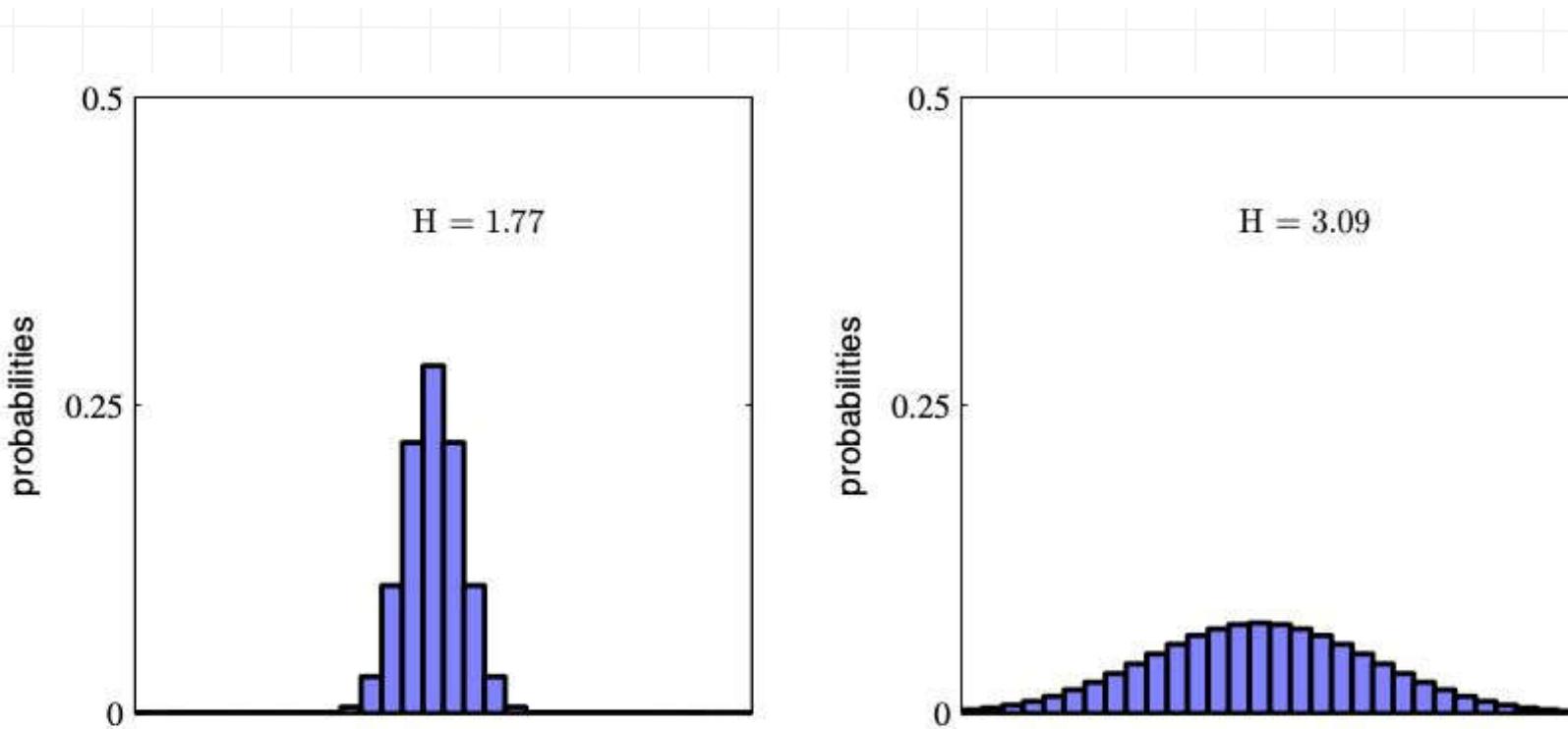
Now consider an example (Cover and Thomas, 1991) of a variable having 8 possible states  $\{a, b, c, d, e, f, g, h\}$  for which the respective probabilities are given by  $(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64})$ . The entropy in this case is given by

$$H[x] = -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{4} \log_2 \frac{1}{4} - \frac{1}{8} \log_2 \frac{1}{8} - \frac{1}{16} \log_2 \frac{1}{16} - \frac{4}{64} \log_2 \frac{1}{64} = 2 \text{ bits.}$$

How to encode these? 0, 10, 110, 1110, 111100, 111101, 111110, 111111.

$$\text{average code length} = \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{16} \times 4 + 4 \times \frac{1}{64} \times 6 = 2 \text{ bits}$$

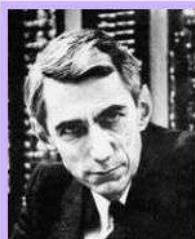
**Noiseless coding theorem** (Shannon 1948) the entropy is a lower bound on the number of bits needed to transmit the state of a random variable.



**Figure 1.30** Histograms of two probability distributions over 30 bins illustrating the higher value of the entropy  $H$  for the broader distribution. The largest entropy would arise from a uniform distribution that would give  $H = -\ln(1/30) = 3.40$ .

Reading: connection to entropy in physics

Recommended: “The Bit Player” (2018 documentary)



Claude Shannon  
1916–2001

After graduating from Michigan and MIT, Shannon joined the AT&T Bell Telephone laboratories in 1941. His paper ‘A Mathematical Theory of Communication’ published in the *Bell System Technical Journal* in 1948 laid the foundations for modern information the-

ory. This paper introduced the word ‘bit’, and his concept that information could be sent as a stream of 1s and 0s paved the way for the communications revolution. It is said that von Neumann recommended to Shannon that he use the term entropy, not only because of its similarity to the quantity used in physics, but also because “nobody knows what entropy really is, so in any discussion you will always have an advantage”.

Reading: deriving differential entropy

## Entropy for continuous variables

$$H[x] = - \int p(x) \ln p(x) dx. \quad (1.104)$$

What is a distribution  $p(x)$  that maximises  $H[x]$ , provided that  $p(x)$  is well defined, has given mean and variance?

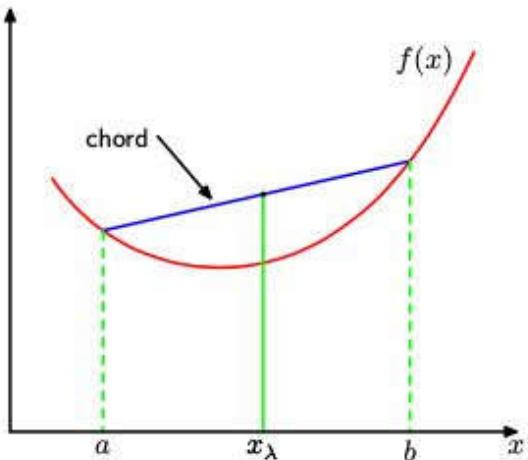
Lagrange multipliers!

$$\begin{aligned} & - \int_{-\infty}^{\infty} p(x) \ln p(x) dx + \lambda_1 \left( \int_{-\infty}^{\infty} p(x) dx - 1 \right) & p(x) = \exp \left\{ -1 + \lambda_1 + \lambda_2 x + \lambda_3 (x - \mu)^2 \right\}. \\ & + \lambda_2 \left( \int_{-\infty}^{\infty} x p(x) dx - \mu \right) + \lambda_3 \left( \int_{-\infty}^{\infty} (x - \mu)^2 p(x) dx - \sigma^2 \right) \end{aligned} \quad (1.108)$$

$$p(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp \left\{ -\frac{(x - \mu)^2}{2\sigma^2} \right\} \quad (1.109)$$

# Convex functions

**Figure 1.31** A convex function  $f(x)$  is one for which every chord (shown in blue) lies on or above the function (shown in red).



$$f(\lambda a + (1 - \lambda)b) \leq \lambda f(a) + (1 - \lambda)f(b). \quad (1.114)$$

Jesen's inequality:  
 $f()$  - a convex function

$$f\left(\sum_{i=1}^M \lambda_i x_i\right) \leq \sum_{i=1}^M \lambda_i f(x_i)$$

where  $\lambda_i \geq 0$  and  $\sum_i \lambda_i = 1$ ,

$$f(\mathbb{E}[x]) \leq \mathbb{E}[f(x)]$$

$$f\left(\int \mathbf{x} p(\mathbf{x}) d\mathbf{x}\right) \leq \int f(\mathbf{x}) p(\mathbf{x}) d\mathbf{x}.$$

# KL divergence

$$\begin{aligned} \text{KL}(p\|q) &= - \int p(\mathbf{x}) \ln q(\mathbf{x}) d\mathbf{x} - \left( - \int p(\mathbf{x}) \ln p(\mathbf{x}) d\mathbf{x} \right) \\ &= - \int p(\mathbf{x}) \ln \left\{ \frac{q(\mathbf{x})}{p(\mathbf{x})} \right\} d\mathbf{x}. \end{aligned} \tag{1.113}$$

If we have “incorrectly” represented  $p(x)$  with  $q(x)$ , how much more *information* do we need to recover  $p(x)$ ?

Apply Jensen’s inequality,  $-\ln()$  is convex

$$f \left( \int \mathbf{x} p(\mathbf{x}) d\mathbf{x} \right) \leq \int f(\mathbf{x}) p(\mathbf{x}) d\mathbf{x}. \tag{1.117}$$

Not symmetric, not a distance measure!

But, has interesting algebraic and geometric properties, see Assignment 1

$$\text{KL}(p\|q) = - \int p(\mathbf{x}) \ln \left\{ \frac{q(\mathbf{x})}{p(\mathbf{x})} \right\} d\mathbf{x} \geq - \ln \int q(\mathbf{x}) d\mathbf{x} = 0 \tag{1.118}$$

Equality will only hold iff.  $p(\mathbf{x}) = q(\mathbf{x})$  for all  $\mathbf{x}$

# “The Venn diagram of entropy and information”

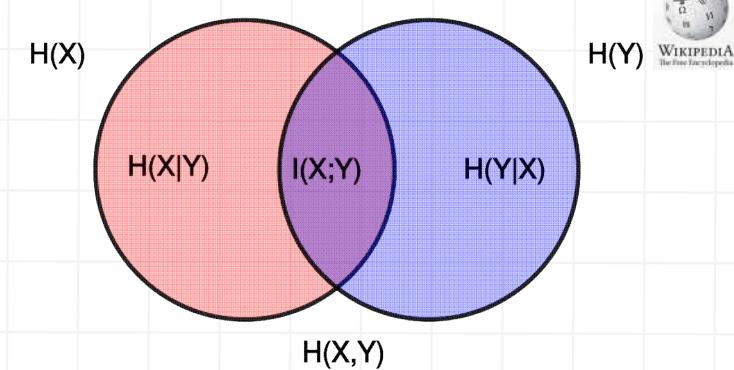
How much *information* is carried in  $x$  about  $y$ ? Two different metrics:  
conditional entropy and mutual information.

$$H[y|x] = - \iint p(y,x) \ln p(y|x) dy dx \quad (1.111)$$

$$H[x,y] = H[y|x] + H[x] \quad (1.112)$$

$$\begin{aligned} I[x,y] &\equiv \text{KL}(p(x,y)\|p(x)p(y)) \\ &= - \iint p(x,y) \ln \left( \frac{p(x)p(y)}{p(x,y)} \right) dx dy \end{aligned} \quad (1.120)$$

$$I[x,y] = H[x] - H[x|y] = H[y] - H[y|x]. \quad (1.121)$$



WIKIPEDIA  
The Free Encyclopedia

# Recap: Linear models for regression

- Basis functions
- Maximum Likelihood with Gaussian Noise
- Regularisation
- Bias variance decomposition

Prelude: Conjugate prior, conditional of Gaussian variables, Bayes Theorem of Gaussian distributions

Bayesian regression

Predictive distributions

Curse of dimensionality + discussions

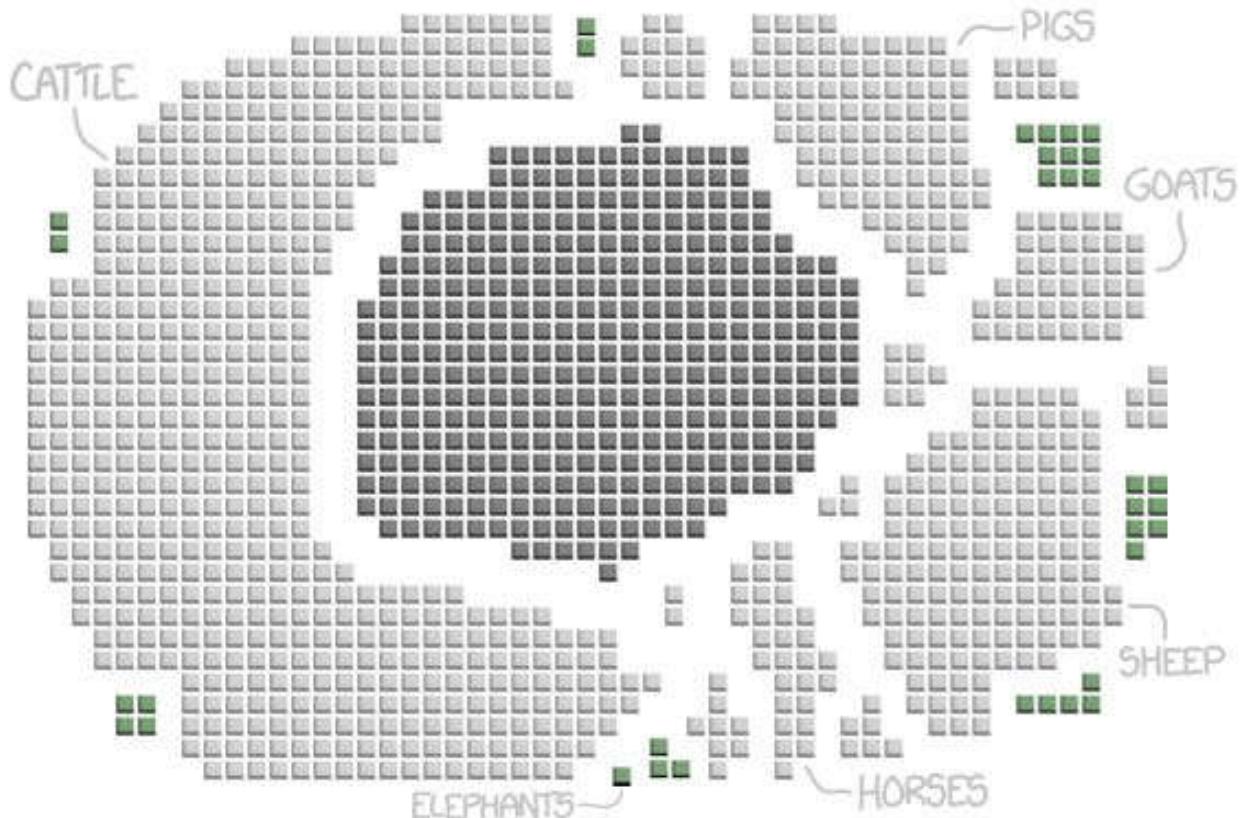
Info theory 101

<https://xkcd.com/1338/>

# EARTH'S LAND MAMMALS BY WEIGHT

■ = 1,000,000 TONS

■ HUMANS ■ OUR PETS AND LIVESTOCK ■ WILD ANIMALS



DATA FROM VACUNA SMITH'S THE EARTH'S BIOSPHERE: DIVERSITY, DYNAMICS, AND CHANGE, PLUS A FEW OTHER SOURCES.

# Mixture Models and Expectation Maximisation

Clustering and density approximation

K-means

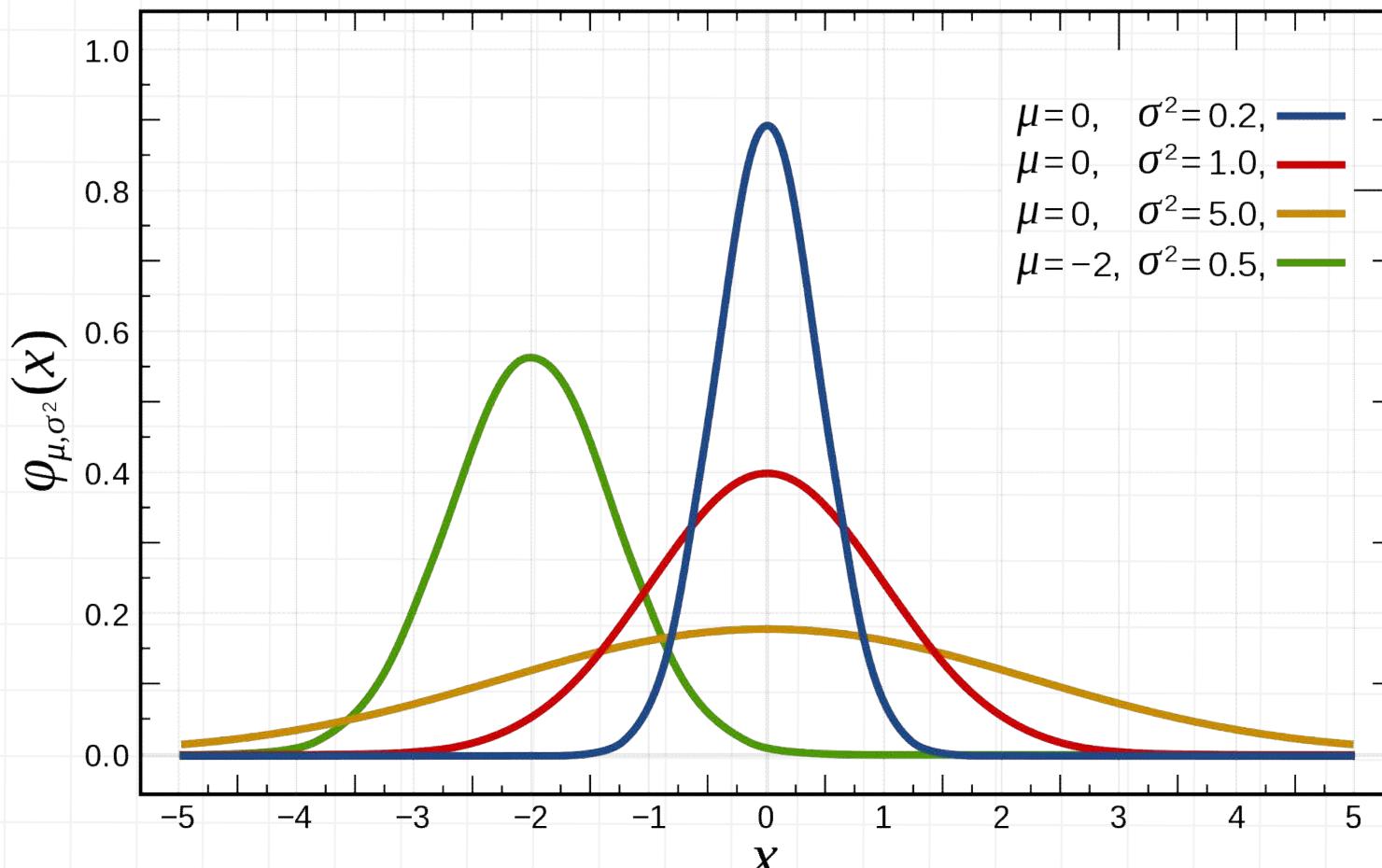
Gaussian mixture models

Discrete latent variables

Expectation maximization, a general technique for finding maximum likelihood estimators in latent variable models -- or, how to solve hard-looking problems by solving several easy-looking pieces.

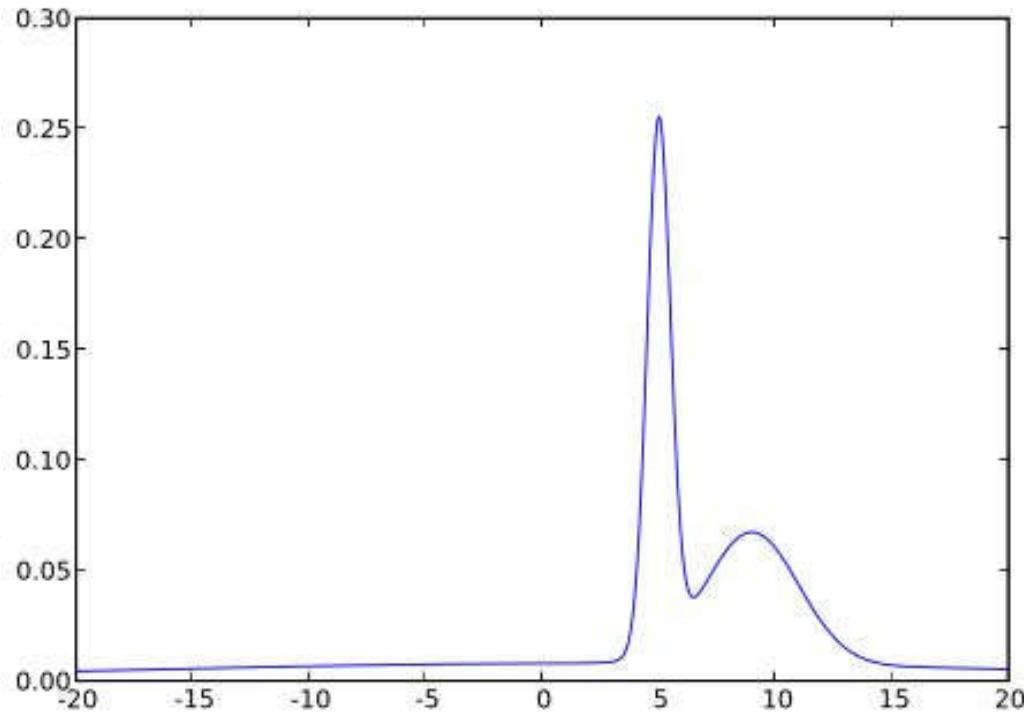
- no workshop Fri clustering / EM #2 after break
- Assignment 1 due today.

# Is there something more in the familiar bell curves?



[https://en.wikipedia.org/wiki/Normal\\_distribution](https://en.wikipedia.org/wiki/Normal_distribution)

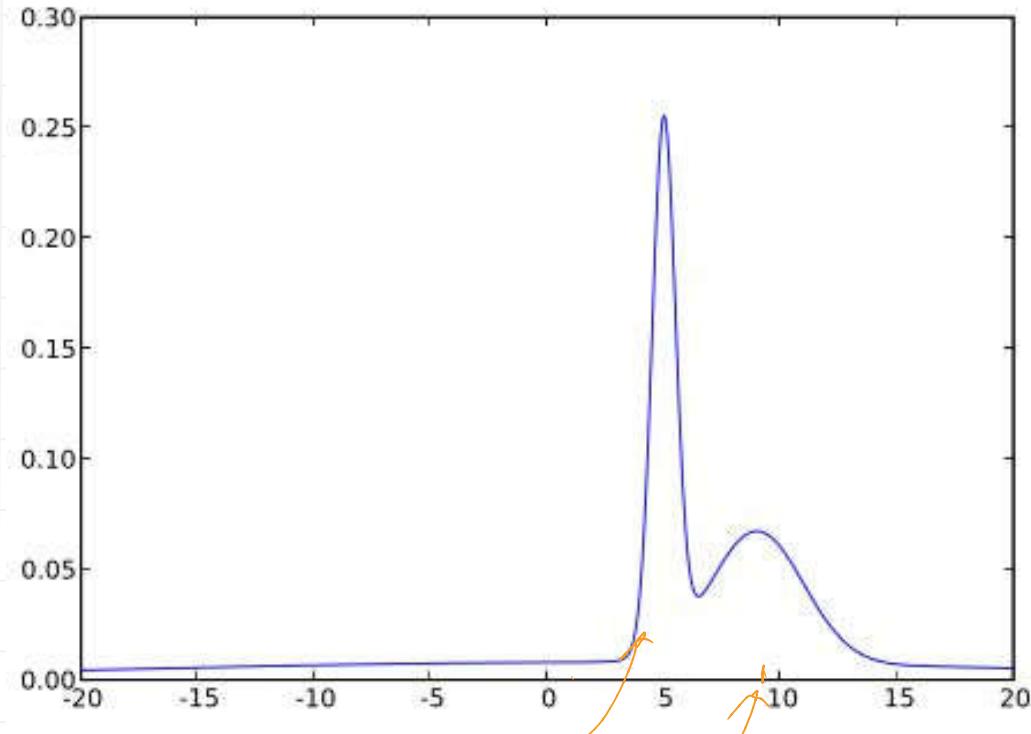
# A “Wallaby” distribution



<https://animalscomparison.com/wallaby-vs-kangaroo-fight-comparison-who-will-win/>

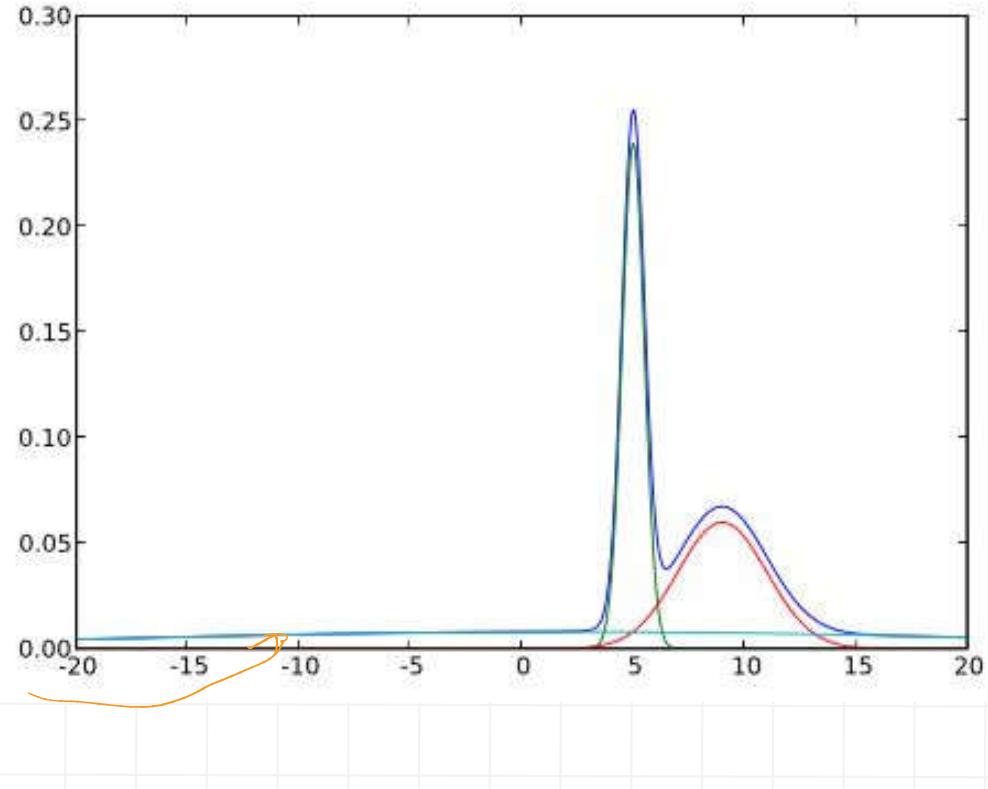


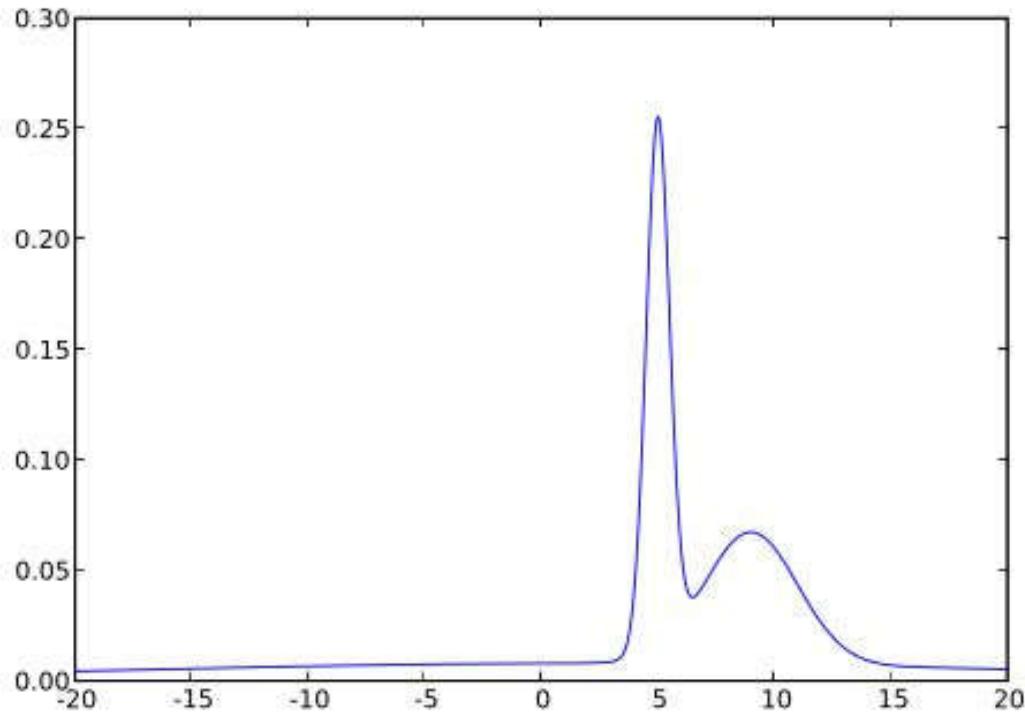
# A “Wallaby” distribution



$$p(x) = \frac{3}{10} \mathcal{N}(x | 5, 0.5) + \frac{3}{10} \mathcal{N}(x | 9, 2) + \frac{4}{10} \mathcal{N}(x | 2, 20)$$

<https://animalscomparison.com/wallaby-vs-kangaroo-fight-comparison-who-will-win/>





$$p(x) = \frac{3}{10} \mathcal{N}(x | 5, 0.5) + \frac{3}{10} \mathcal{N}(x | 9, 2) + \frac{4}{10} \mathcal{N}(x | 2, 20)$$

any smooth density can be approximated to arbitrary precision by a Gaussian mixture model with enough components.

Park, J. and Sandberg, I.W., 1991. Universal approximation using radial-basis-function networks. *Neural computation*, 3(2), pp.246-257.

Geoffrey McLachlan and David Peel. *Finite mixture models*. John Wiley & Sons, 2004.

# Clustering 101: K-Means

- Given a set of data  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  where  $\mathbf{x}_n \in \mathbb{R}^D$ ,  $n = 1, \dots, N$ .

- Goal: Partition the data into  $K$  clusters.

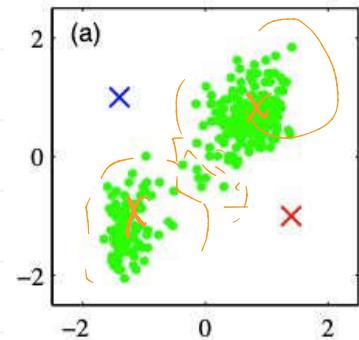
*given*

- Each cluster contains points close to each other.

- Introduce a prototype  $\mu_k \in \mathbb{R}^D$  for each cluster.

- Goal: Find

- a set prototypes  $\mu_k$ ,  $k = 1, \dots, K$ , each representing a different cluster.
- an assignment of each data point to exactly one cluster.



$K=2$

A.K.A.

Vector Quantization

# K-means clustering: representation + loss

- Start with arbitrary chosen prototypes  $\mu_k, k = 1, \dots, K$ .
  - ① Assign each data point to the closest prototype.
  - ② Calculate new prototypes as the mean of all data points assigned to each of them.
- Binary indicator variables

$$r_{nk} = \begin{cases} 1, & \text{if data point } \mathbf{x}_n \text{ belongs to cluster } k \\ 0, & \text{otherwise} \end{cases}$$

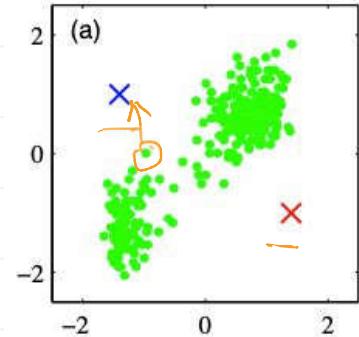
using the 1-of- $K$  coding scheme.

- Define a **distortion measure**

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2 \quad (9.1)$$

*only 1 of  $k$  term  $> 0$*

- Find the values for  $\{r_{nk}\}$  and  $\{\boldsymbol{\mu}_k\}$  so as to minimise  $J$ .

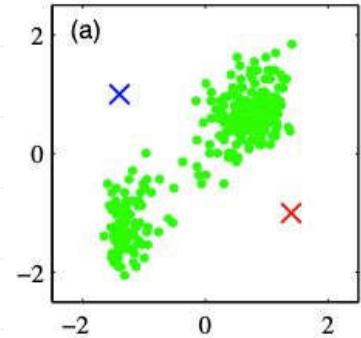


data point  $\mathbf{x}_n$

$$\mathbf{r}_n = [0, 1, \dots, 0]$$

## K-means clustering: algorithm

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2 \quad (9.1)$$



But  $\{r_{nk}\}$  depends on  $\{\boldsymbol{\mu}_k\}$ , and  $\{\boldsymbol{\mu}_k\}$  depends on  $\{r_{nk}\}$ .

If  $\{\boldsymbol{\mu}_k\}$  are given, we only need to determine  $r_{nk}$

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg \min_j \|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2 \\ 0 & \text{otherwise.} \end{cases} \quad (9.2)$$

$\{\boldsymbol{\mu}_k\}$  fixed

take the min

$$\left. \begin{array}{l} \|\mathbf{x}_n - \boldsymbol{\mu}_1\|^2 \\ \|\mathbf{x}_n - \boldsymbol{\mu}_2\|^2 \\ \vdots \\ \|\mathbf{x}_n - \boldsymbol{\mu}_K\|^2 \end{array} \right\}$$

## K-means clustering: algorithm

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2 \quad (9.1)$$

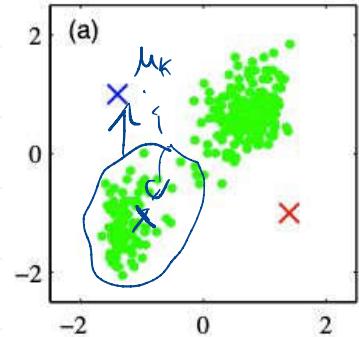
But  $\{r_{nk}\}$  depends on  $\{\boldsymbol{\mu}_k\}$ , and  $\{\boldsymbol{\mu}_k\}$  depends on  $\{r_{nk}\}$ .

Assume  $r_{nk}$  is known, what would  $\{\boldsymbol{\mu}_k\}$  be?

$$2 \sum_{n=1}^N r_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k) = 0 \quad (9.3)$$

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk} \mathbf{x}_n}{\sum_n r_{nk}}. \quad (9.4)$$

↖  $\#\{\mathbf{x}_n\}$  assigned to cluster  $k$



# K-means clustering: recap

Expectation step

Re-assign data points to clusters, determine  $r_{nk}$

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg \min_j \|\mathbf{x}_n - \mu_j\|^2 \\ 0 & \text{otherwise.} \end{cases} \quad (9.2)$$

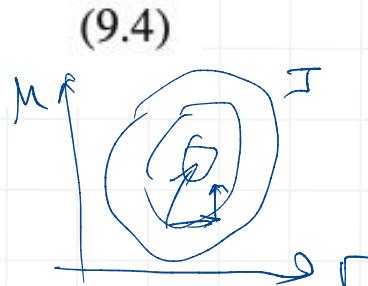
Maximisation step

Re-compute the cluster means - update  $\{\mu_k\}$

$$\mu_k = \frac{\sum_n r_{nk} \mathbf{x}_n}{\sum_n r_{nk}}.$$

fix  $\mu_k$   
min. ]

fix  $r_{nk}$   
min. ]



Where to start?  
When to stop?  
Why does this work?

{randomly pick  
k out of  $X^n$ }  
- Kmean++

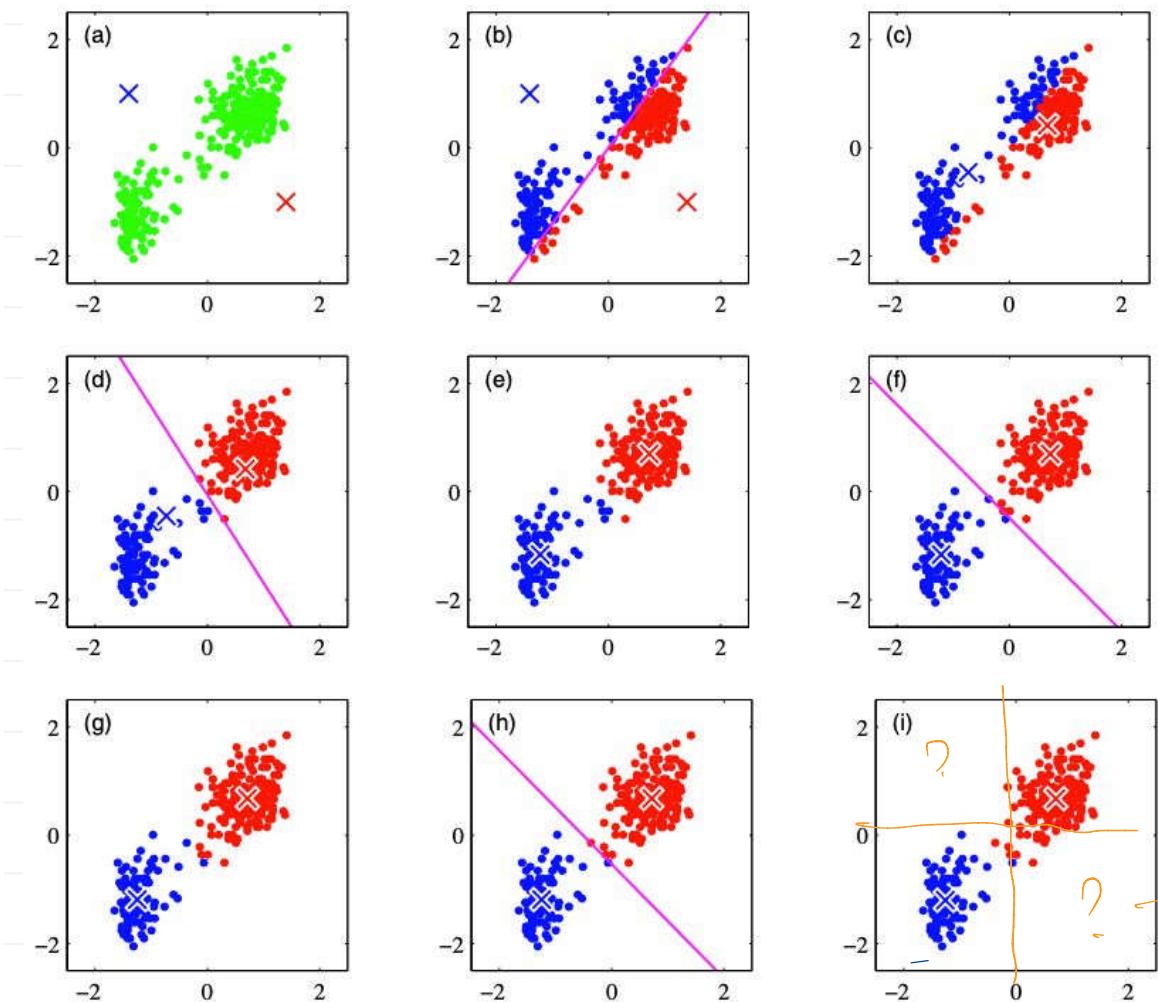
E-step:

Stop if  $\{r_{nk}\}$  does not change.

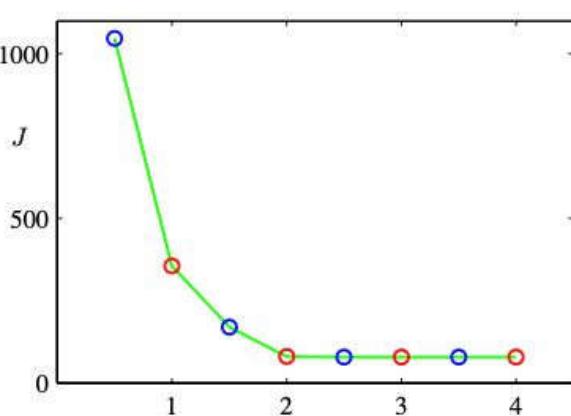
M-step

Stop if  $\{\mu_k\}$  do not change  
or  $\sum_k (\mu_k^{(t+1)} - \mu_k^{(t)})^2 \leq \epsilon$

OR maximum iteration reached.



Plot of the cost function  $J$  given by (9.1) after each E step (blue points) and M step (red points) of the  $K$ -means algorithm for the example shown in Figure 9.1. The algorithm has converged after the third M step, and the final EM cycle produces no changes in either the assignments or the prototype vectors.



**Figure 9.1** Illustration of the  $K$ -means algorithm using the re-scaled Old Faithful data set. (a) Green points denote the data set in a two-dimensional Euclidean space. The initial choices for centres  $\mu_1$  and  $\mu_2$  are shown by the red and blue crosses, respectively. (b) In the initial E step, each data point is assigned either to the red cluster or to the blue cluster, according to which cluster centre is nearer. This is equivalent to classifying the points according to which side of the perpendicular bisector of the two cluster centres, shown by the magenta line, they lie on. (c) In the subsequent M step, each cluster centre is re-computed to be the mean of the points assigned to the corresponding cluster. (d)–(i) show successive E and M steps through to final convergence of the algorithm.

u, Q  
empty

- Initial condition crucial for convergence.
- What happens, if at least one cluster centre is too far from all data points?
- Complex step: Finding the nearest neighbour. (Use triangle inequality; build K-D trees, ...)
- Generalise to non-Euclidean dissimilarity measures  $\mathcal{V}(\mathbf{x}_n, \boldsymbol{\mu}_k)$  (called **K-medoids** algorithm),

$$\tilde{J} = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \mathcal{V}(\mathbf{x}_n, \boldsymbol{\mu}_k).$$

- Online stochastic algorithm
  - Draw data point  $\mathbf{x}_n$  and locate nearest prototype  $\boldsymbol{\mu}_k$ .
  - Update only  $\boldsymbol{\mu}_k$  using decreasing learning rate  $\eta_n$

$$\boldsymbol{\mu}_k^{\text{new}} = \boldsymbol{\mu}_k^{\text{old}} + \eta_n (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{old}}).$$

• finds a local min.  
of  $J$

• what to do if

$$J_K, \{r_{nk}, \eta_n\} = 0$$

can't update  $\boldsymbol{\mu}_k$ ,

re-initialise  $\boldsymbol{\mu}_k$  to  
another data point  
data dim?  
 $O(NKD)$

$K = 2$



4.2%

(6k colors)

$K = 10$



16.7%

quant(x|10)

$K = 3$



8.3%

Original image



100 %

24 bits

Pixel (R, G, B) tuple, 8-bit int for each channel  
 $[0, 255]$

K-means for illustrating image segmentation and data compression

- Segment an image into regions of reasonable homogeneous appearance.
- Each pixel is a point in  $\mathbb{R}^3$  (red, blue, green). (Note that the pixel intensities are bounded in the range  $[0, 1]$  and therefore this space is strictly speaking not Euclidean).
- Run  $K$ -means on all points of the image until convergence. Replace all pixels with the corresponding mean  $\mu_k$ .
- Results in an image with a palette only  $K$  different colours.
- There are much better approaches to image segmentation (but it is an active research topic), this here serves only to illustrate  $K$ -means.
- Store the **code-book vectors**  $\mu_k$ .
- Store the data in the form of references (labels) to the code-book. Each data point has a label in the range  $[1, \dots, K]$ .
- New data points are also compressed by finding the closest code-book vector and then storing only the label.
- This technique is also called **vector quantisation**.

# Mixture Models and Expectation Maximisation

Clustering and density approximation

K-means

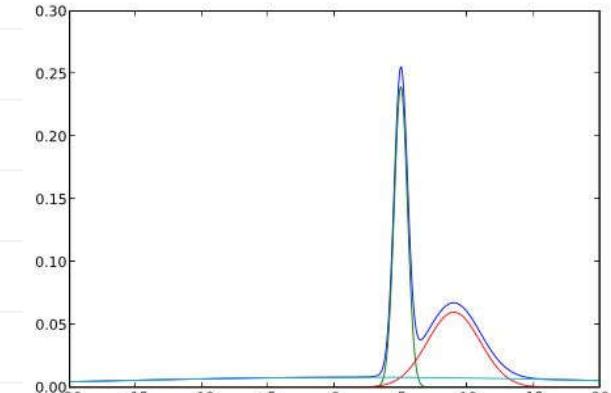
Gaussian mixture models

## Gaussian Mixture Models (GMM)

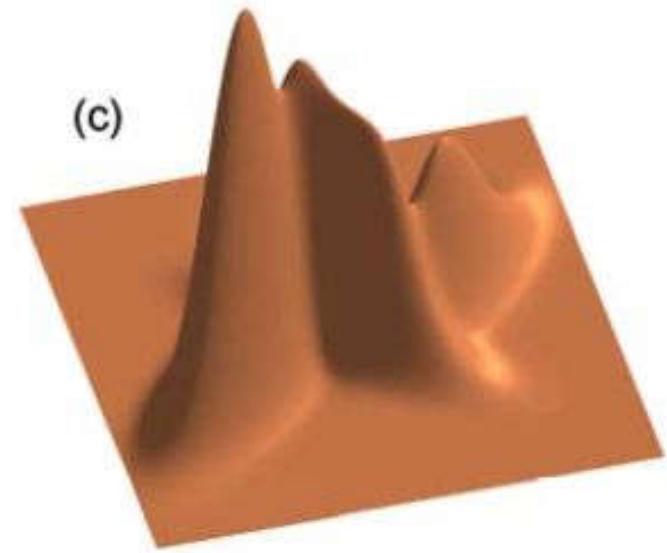
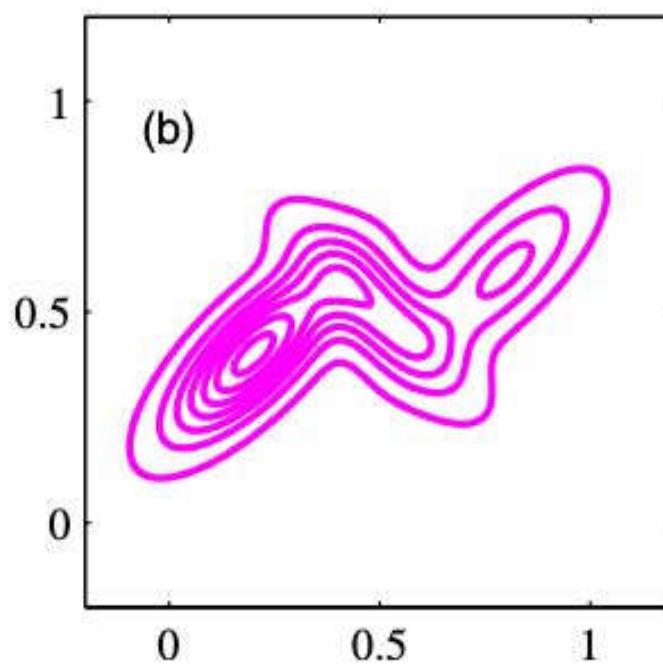
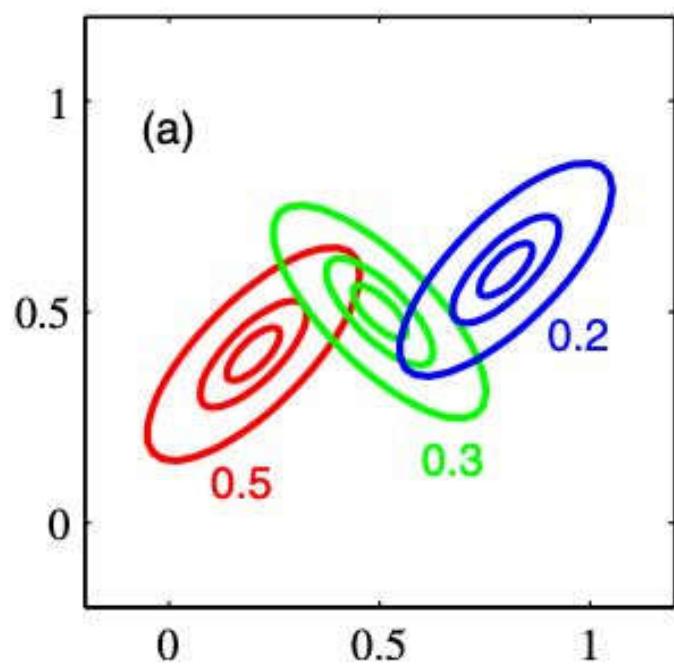
Mixture distributions are formed by taking linear combinations of more basic distributions.

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad (2.188) \text{ also (9.7)}$$

$$0 \leq \pi_k \leq 1. \quad \sum_{k=1}^K \pi_k = 1.$$



complex densities. By using a sufficient number of Gaussians, and by adjusting their means and covariances as well as the coefficients in the linear combination, almost any continuous density can be approximated to arbitrary accuracy.



**Figure 2.23** Illustration of a mixture of 3 Gaussians in a two-dimensional space. (a) Contours of constant density for each of the mixture components, in which the 3 components are denoted red, blue and green, and the values of the mixing coefficients are shown below each component. (b) Contours of the marginal probability density  $p(x)$  of the mixture distribution. (c) A surface plot of the distribution  $p(x)$ .

## GMM as a latent variable model

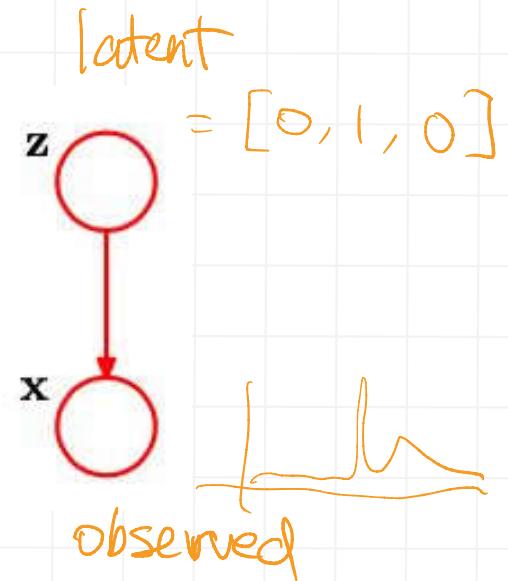
**Figure 9.4** Graphical representation of a mixture model, in which the joint distribution is expressed in the form  $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$ .

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad (2.188) \text{ also (9.7)}$$

$$0 \leq \pi_k \leq 1. \quad \sum_{k=1}^K \pi_k = 1.$$

$$p(z_k = 1) = \pi_k$$

$$p(\mathbf{z}) = \prod_{k=1}^K \pi_k^{z_k}. \quad (9.10)$$



$$p(z_k=1) = \pi_k$$

## Conditional and marginal distributions

$$p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^K \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}. \quad (9.11)$$

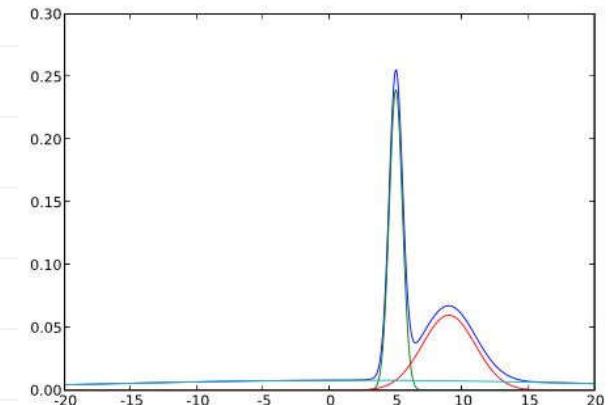
only one of the  $z_k$   
is active

$\mathbf{x}$  is gaussian  
conditioned on  $\mathbf{z}$



$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad (9.12)$$

$$\sum_z \prod_{k=1}^K \pi_k^{z_k} N(x|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}$$



## Posterior

Conditional probability of  $\mathbf{z}$  given  $\mathbf{x}$  by Bayes' theorem

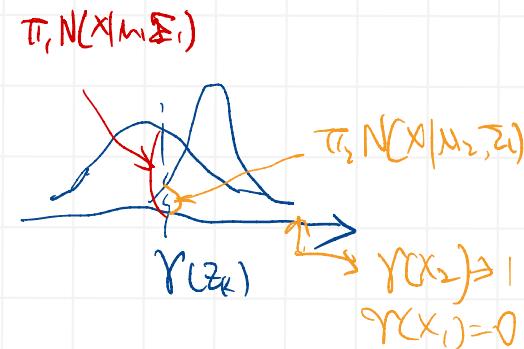
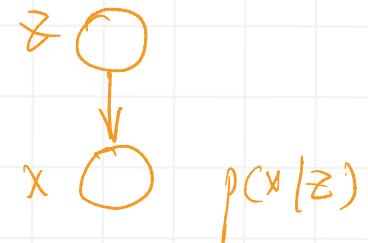
$$\gamma(z_k) \equiv p(z_k = 1 | \mathbf{x}) = \frac{p(z_k = 1)p(\mathbf{x} | z_k = 1)}{\sum_{j=1}^K p(z_j = 1)p(\mathbf{x} | z_j = 1)}$$

$$= \frac{\pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

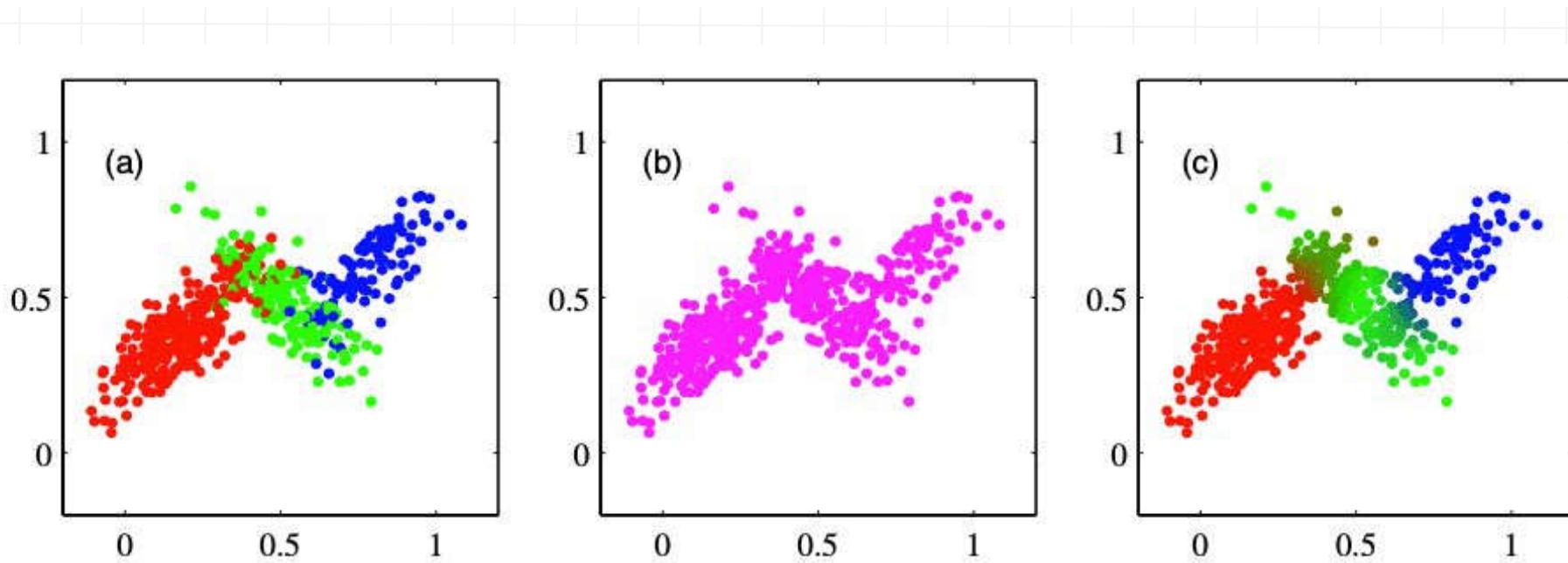
function of  $\mathbf{x}$

$$\frac{P(\mathbf{x}, z_k = 1)}{P(\mathbf{x})}$$

(9.13)



$\gamma(z_k)$  is the **responsibility** of component  $k$  to 'explain' the observation  $\mathbf{x}$ .



**Figure 9.5** Example of 500 points drawn from the mixture of 3 Gaussians shown in Figure 2.23. (a) Samples from the joint distribution  $p(z)p(x|z)$  in which the three states of  $z$ , corresponding to the three components of the mixture, are depicted in red, green, and blue, and (b) the corresponding samples from the marginal distribution  $p(x)$ , which is obtained by simply ignoring the values of  $z$  and just plotting the  $x$  values. The data set in (a) is said to be *complete*, whereas that in (b) is *incomplete*. (c) The same samples in which the colours represent the value of the responsibilities  $\gamma(z_{nk})$  associated with data point  $x_n$ , obtained by plotting the corresponding point using proportions of red, blue, and green ink given by  $\gamma(z_{nk})$  for  $k = 1, 2, 3$ , respectively

# Maximum likelihood estimation

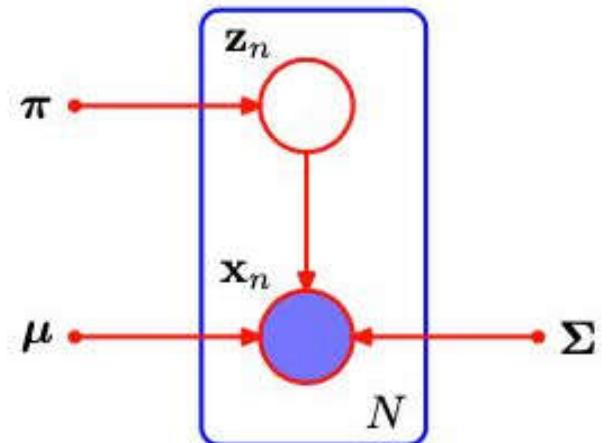
Observed:  $\{\mathbf{x}_n\}$ ; Unobserved/latent:  $\{\mathbf{z}_n\}$

Need to estimate:  $\pi, \mu, \Sigma$

**Figure 9.6** Graphical representation of a Gaussian mixture model for a set of  $N$  i.i.d. data points  $\{\mathbf{x}_n\}$ , with corresponding latent points  $\{\mathbf{z}_n\}$ , where  $n = 1, \dots, N$ .

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}. \quad (9.14)$$

*P(x<sub>n</sub>)*



# Issues in maximum likelihood

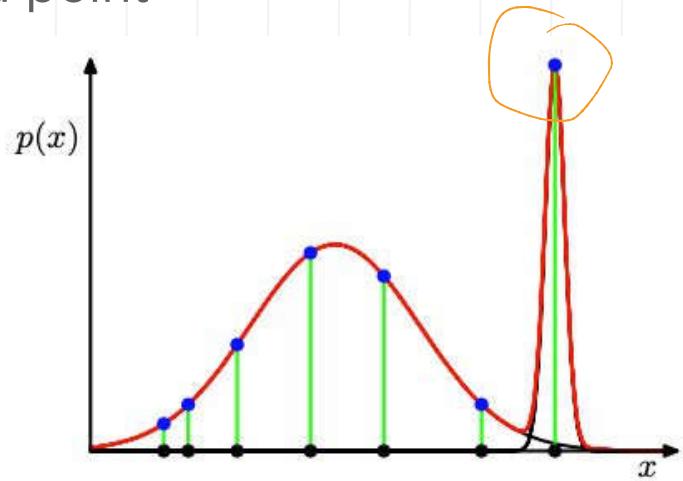
Parameter space symmetries, or identifiability problem

- A  $K$  component mixture has a total of  $K!$  equivalent solutions corresponding to the  $K!$  ways of assigning  $K$  sets of parameters to  $K$  solutions.
- Also called **identifiability problem**. Needs to be considered when the parameters discovered by a model are interpreted.

at least  $K!$ ,  
global minima  
that are equivalent to  
each other

Singularity due to component collapsing onto one data point

**Figure 9.7** Illustration of how singularities in the likelihood function arise with mixtures of Gaussians. This should be compared with the case of a single Gaussian shown in Figure 1.14 for which no singularities arise.



## Challenges in maximising the likelihood function

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}. \quad (9.14)$$

- Maximising the log likelihood of a Gaussian mixture is more complex than for a single Gaussian. Summation over all  $K$  components inside of the logarithm make it harder.
- Setting the derivatives of the log likelihood to zero does not longer result in a closed form.
- May use gradient-based optimisation.
- Or EM algorithm. Stay tuned.

Whole problem: solve for  $\pi, \mu, \Sigma$

$\text{cost} \propto \exp(-\frac{\|\mu\|^2}{2})$

Divide-and-conquer: solve for  $\mu_k$

only one term depend on  $\mu_k$

$$L = \ln p(\mathbf{X} | \pi, \mu, \Sigma) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \mu_k, \Sigma_k) \right\}. \quad (9.14)$$

Differentiate wrt  $\mu_k$

$$\frac{\partial L}{\partial \mu_k} = \sum_{n=1}^N \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \mu_k, \Sigma_k) \quad (9.15)$$

$\pi_k$  - const.,  $\exp[-\frac{\|\mu_k\|^2}{2}] \dots$

$$0 = - \sum_{n=1}^N \left( \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \mu_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n | \mu_j, \Sigma_j)} \right) \Sigma_k (\mathbf{x}_n - \mu_k) \quad (9.16)$$

$\gamma(z_{nk})$

$$0 = - \sum_{n=1}^N \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}}_{\gamma(z_{nk})} \boldsymbol{\Sigma}_k (\mathbf{x}_n - \boldsymbol{\mu}_k) \quad (9.16)$$

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n \quad (9.17)$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T \quad (9.19)$$

$N_k = \sum_{n=1}^N \gamma(z_{nk}).$

# of  
"effective" points  
in cluster k

Solve for  $\pi_k$

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \lambda \left( \sum_{k=1}^K \pi_k - 1 \right) \quad (9.20)$$

$$0 = \sum_{n=1}^N \frac{\mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} + \lambda \quad (9.21)$$

$$\lambda = -\mathbb{N}$$

$$\pi_k = \frac{N_k}{N} \quad (9.22)$$

$\sum_n \gamma(z_{nk})$



## So, what happened?

Given  $\gamma_{nk}$ , solve for  $\pi_k \boldsymbol{\mu}_k \boldsymbol{\Sigma}_k$

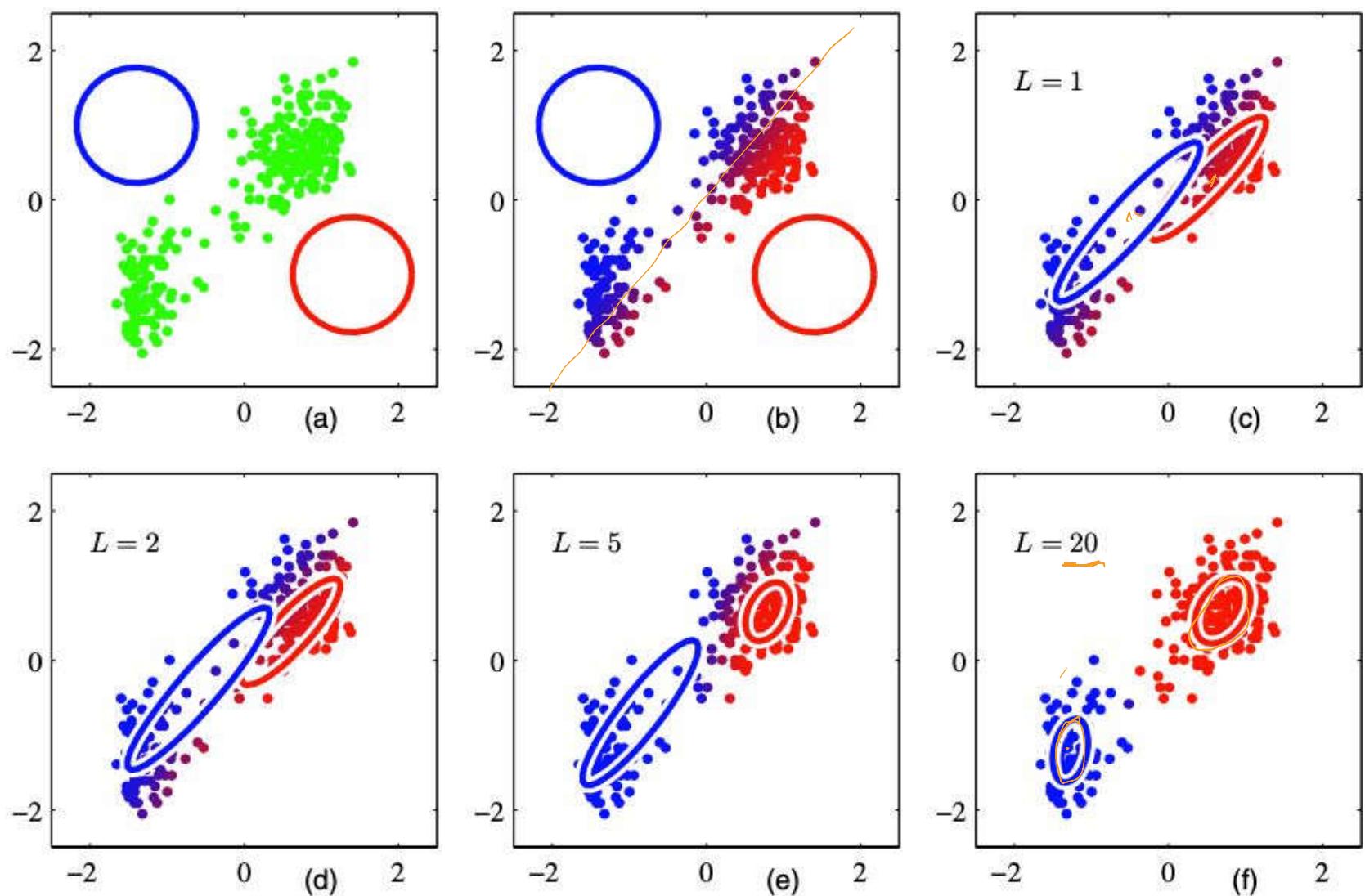
But  $\gamma_{nk}$  is computed from  $\{x_n\}, \pi_k \boldsymbol{\mu}_k \boldsymbol{\Sigma}_k$

...

In the expectation step, or E step, we use the current values for the parameters to evaluate the posterior probabilities, or responsibilities, given by (9.13).

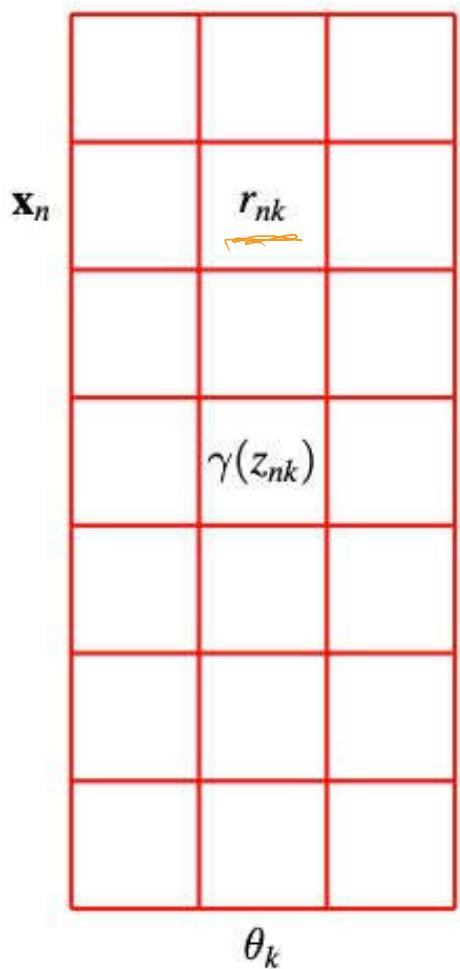
$$\gamma_k = \frac{\pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \quad (9.13)$$

We then use these probabilities in the maximization step, or M step, to re-estimate the means, covariances, and mixing coefficients using the results (9.17), (9.19), and (9.22).



**Figure 9.8** Illustration of the EM algorithm using the Old Faithful set as used for the illustration of the  $K$ -means algorithm in Figure 9.1. See the text for details.

# Responsibilities -- one connection to K-means



For  $k$ -means clustering,  
we have hard assignments

For GMM,  
we have soft assignments

$$r_{nk} = \begin{cases} 1 & \text{if } x_n \text{ belongs to cluster } k \\ 0 & \text{o.w.} \end{cases}$$

$$\gamma(z_{nk}) = P(z_{nk}=1 | x_n)$$

## EM for Gaussian Mixtures

Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters (comprising the means and covariances of the components and the mixing coefficients).

1. Initialize the means  $\mu_k$ , covariances  $\Sigma_k$  and mixing coefficients  $\pi_k$ , and evaluate the initial value of the log likelihood.
2. **E step.** Evaluate the responsibilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

using k-means to init  $\mu$

3. **M step.** Re-estimate the parameters using the current responsibilities

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n \quad (9.24)$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^T \quad (9.25)$$

$$\pi_k^{\text{new}} = \frac{N_k}{N} \quad (9.26)$$

where

$$N_k = \sum_{n=1}^N \gamma(z_{nk}). \quad (9.27)$$

4. Evaluate the log likelihood

$$\ln p(\mathbf{X} | \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\} \quad (9.28)$$

and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.

# Mixture Models and Expectation Maximisation

Clustering and density approximation

K-means

Gaussian mixture models

Discrete latent variables

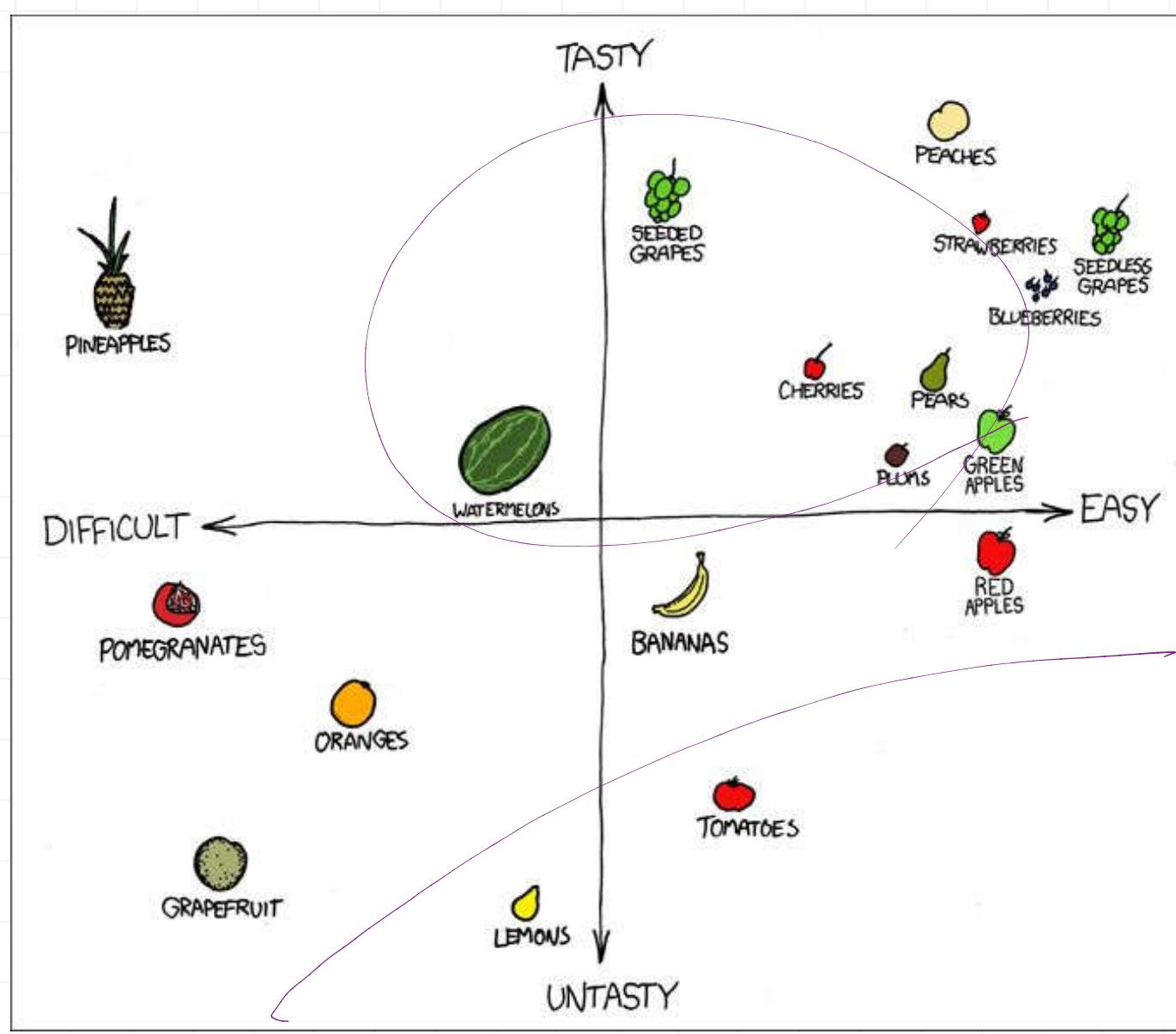
Expectation maximization, a general technique for finding maximum likelihood estimators in latent variable models -- or, how to solve hard-looking problems by solving several easy-looking pieces.

NEXT TIME:

Why "expectation"  
& "max." ?

Why does EM work?

<https://xkcd.com/388/>



# Announcements

In person lecture: Wed 16 March, PHYS T (will try *simulcast* in Teams, with us luck!)

<https://studentvip.com.au/anu/main/maps/142757>

Quiz 1 next week, due Thu (releasing by Mon)

# Linear models for classification

Classification problems

A glimpse of decision theory (Sec 1.5)

Discriminant functions - why least squares doesn't work here

The perceptron algorithm

Probabilistic generative models - origin of the logistic function

Probabilistic discriminative models

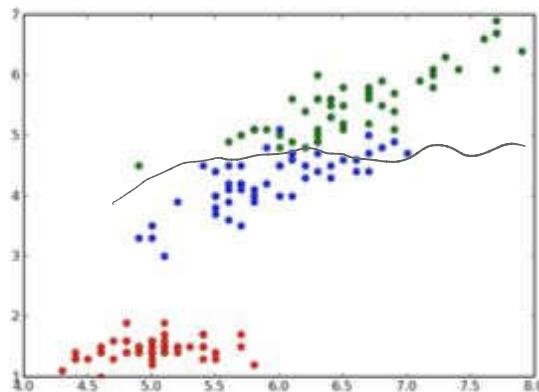
Logistic regression

Laplace approximation (Bayesian logistic regression) - later

Origin of the logistic function, logistic regression and how it connects to perceptron

# Classification

- Goal : Given input data  $\mathbf{x}$ , assign it to one of  $K$  discrete classes  $\mathcal{C}_k$  where  $k = 1, \dots, K$ .
- Divide the input space into different regions.
- Equivalently: map each point to a categorical label.



Length of petal [in cm] vs sepal [cm] for three types of flowers  
(Iris Setosa, Iris Versicolor, Iris Virginica).

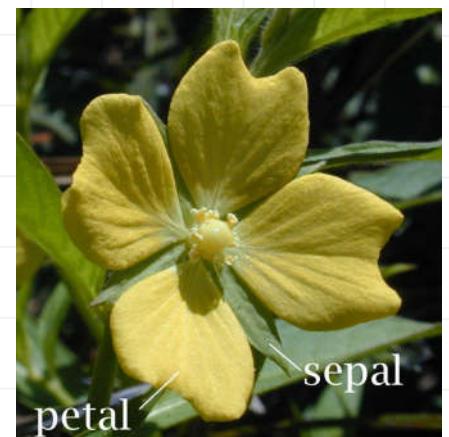


Image from wikipedia

# Representing classes

binary  $t \in \{0, 1\}$

Meeting in person/remote, pass/fail, benign/malignant, should be given credit (Y/N), ...

one-hot  $\mathbf{t} = (0, 1, 0, 0, 0)^T.$  (4.1)

$$\sum_k t_k = 1$$

Also denote  $t_k \longrightarrow \mathcal{C}_k$

Iris flowers, object classes in photos, natural language, ...

Other representations abound, e.g. structured (time series, sequences – PRML Chapter 13, graphs),  
Sec 4.1.5 (an output representation that connects Fisher Discriminant to least squares),  
Sec 4.1.7 perceptrons, SVMs  $\{-1, +1\}$

## A primer on decision theory (Sec 1.5)

Inference: compute  $p(\mathbf{x}, \mathbf{t})$ , e.g. from a set of training data.

Decision theory: take a specific action based on our understanding of the values  $\mathbf{t}$  is likely to take.

$$\underbrace{p(\mathcal{C}_k | \mathbf{x})}_{\text{---}} = \frac{p(\mathbf{x} | \mathcal{C}_k) p(\mathcal{C}_k)}{p(\mathbf{x})}. \quad (1.77)$$



(end of page 38) "We shall see that the decision stage is generally very simple, even trivial, once we have solved the inference problem."

- many counter-examples! e.g. multiple medical tests, test + quarantine for covid
- frontiers of theoretical and practical ML, e.g. active/online learning, ML and economics

## Minimise error

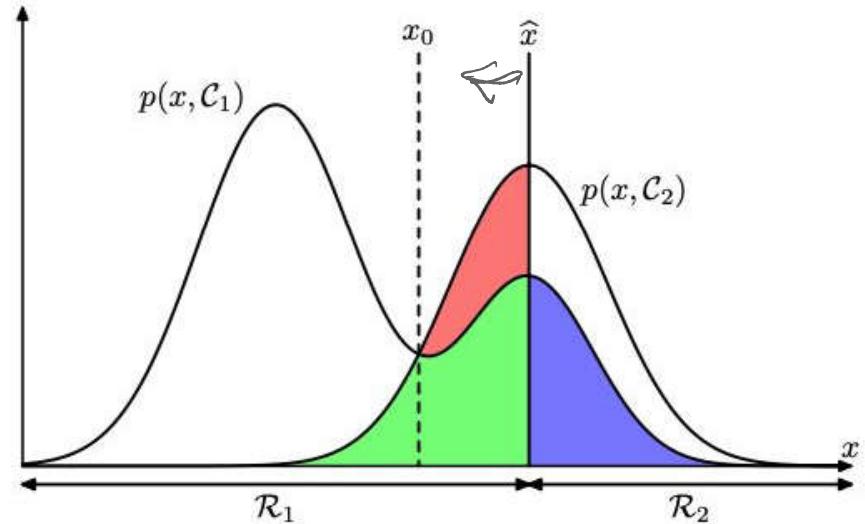
$$p(\text{correct}) = \sum_{k=1}^K p(\mathbf{x} \in \mathcal{R}_k, \mathcal{C}_k)$$

$$= \sum_{k=1}^K \int_{\mathcal{R}_k} p(\mathbf{x}, \mathcal{C}_k) d\mathbf{x}$$

*p red+green*

$$p(\text{mistake}) = p(\mathbf{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\mathbf{x} \in \mathcal{R}_2, \mathcal{C}_1)$$

$$= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x}. \quad (1.78)$$



**Figure 1.24** Schematic illustration of the joint probabilities  $p(x, \mathcal{C}_k)$  for each of two classes plotted against  $x$ , together with the decision boundary  $x = \hat{x}$ . Values of  $x \geq \hat{x}$  are classified as class  $\mathcal{C}_2$  and hence belong to decision region  $\mathcal{R}_2$ , whereas points  $x < \hat{x}$  are classified as  $\mathcal{C}_1$  and belong to  $\mathcal{R}_1$ . Errors arise from the blue, green, and red regions, so that for  $x < \hat{x}$  the errors are due to points from class  $\mathcal{C}_2$  being misclassified as  $\mathcal{C}_1$  (represented by the sum of the red and green regions), and conversely for points in the region  $x \geq \hat{x}$  the errors are due to points from class  $\mathcal{C}_1$  being misclassified as  $\mathcal{C}_2$  (represented by the blue region). As we vary the location  $\hat{x}$  of the decision boundary, the combined areas of the blue and green regions remains constant, whereas the size of the red region varies. The optimal choice for  $\hat{x}$  is where the curves for  $p(x, \mathcal{C}_1)$  and  $p(x, \mathcal{C}_2)$  cross, corresponding to  $\hat{x} = x_0$ , because in this case the red region disappears. This is equivalent to the minimum misclassification rate decision rule, which assigns each value of  $x$  to the class having the higher posterior probability  $p(\mathcal{C}_k|x)$ .

## What are the potential problems for minimising classification error?

$$\begin{aligned} p(\text{mistake}) &= p(\mathbf{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\mathbf{x} \in \mathcal{R}_2, \mathcal{C}_1) \\ &= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x}. \end{aligned} \quad (1.78)$$

Example scenario: medical diagnosis

# Minimise loss, balanced metric, having a reject option

**Figure 1.25** An example of a loss matrix with elements  $L_{kj}$  for the cancer treatment problem. The rows correspond to the true class, whereas the columns correspond to the assignment of class made by our decision criterion.

$$\mathbb{E}[L] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj} p(\mathbf{x}, \mathcal{C}_k) d\mathbf{x}. \quad (1.80)$$

$$\text{Decision: } \operatorname{argmin}_j \sum_k L_{kj} p(\mathcal{C}_k | \mathbf{x}) \quad (1.81)$$

True  $\rightarrow$

	cancer	normal
cancer	0	1000
normal	1	0

		Predicted 1 - Cancer	Predicted 0 - No Cancer	
Actual 1 - Cancer	90 [TP]	4 [FN]		
	1 [FP]	5 [TN]		

$$\frac{TP}{P} = \frac{90}{94}$$

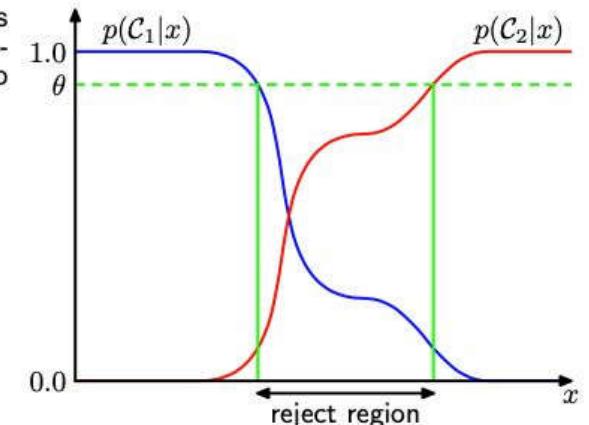
— Cancer Records [90+4=94]

— Non Cancer Records [5+1=6]

<https://towardsdatascience.com/confusion-matrix-clearly-explained-fee63614dc7>

$$\frac{TN}{N} = \frac{5}{6}$$

**Figure 1.26** Illustration of the reject option. Inputs  $x$  such that the larger of the two posterior probabilities is less than or equal to some threshold  $\theta$  will be rejected.



Brodersen, K. H.; Ong, C. S.; Stephan, K. E.; Buhmann, J. M., 2010.  
**The balanced accuracy and its posterior distribution.**  
 International Conference on Pattern Recognition.

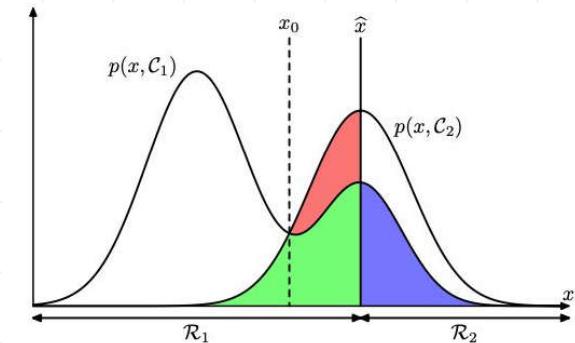
$$\frac{1}{2} (TP/P + TN/N)$$

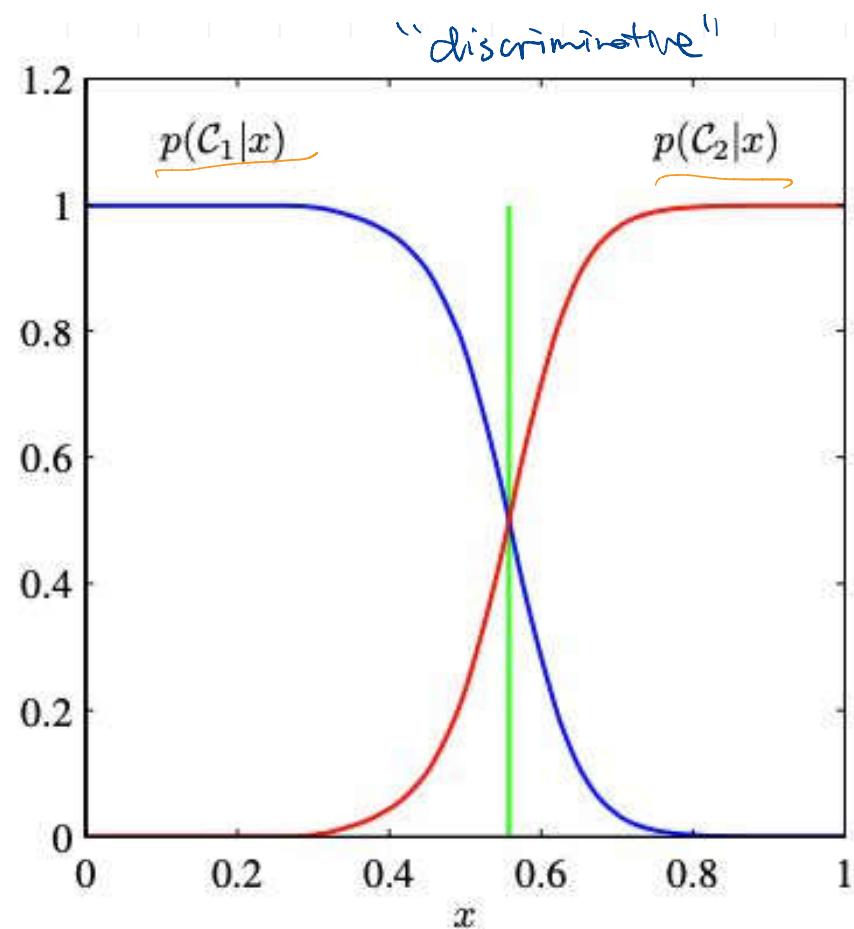
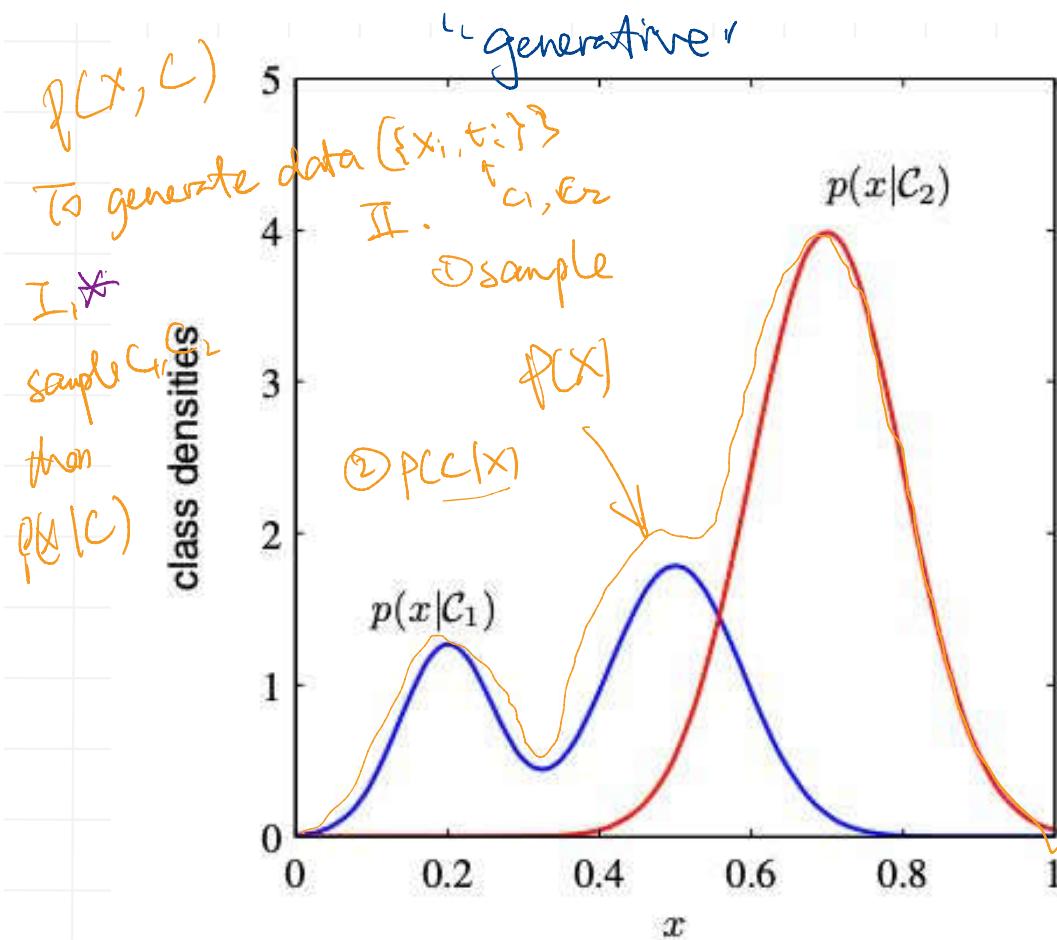
(many other metrics also address this problem)

# Three types of models for decision problems

①  
②  
③

- Find a **discriminant function**  $f(\mathbf{x})$  which maps each input directly onto a class label.
- Discriminative Models
  - ① Solve the inference problem of determining the posterior class probabilities  $p(C_k | \mathbf{x})$ .
  - ② Use decision theory to assign each new  $\mathbf{x}$  to one of the classes.
- Generative Models
  - ① Solve the inference problem of determining the class-conditional probabilities  $p(\mathbf{x} | C_k)$ .
  - ② Also, infer the prior class probabilities  $p(C_k)$ .
  - ③ Use Bayes' theorem to find the posterior  $p(C_k | \mathbf{x})$ .
  - ④ Alternatively, model the joint distribution  $p(\mathbf{x}, C_k)$  directly.
  - ⑤ Use decision theory to assign each new  $\mathbf{x}$  to one of the classes.

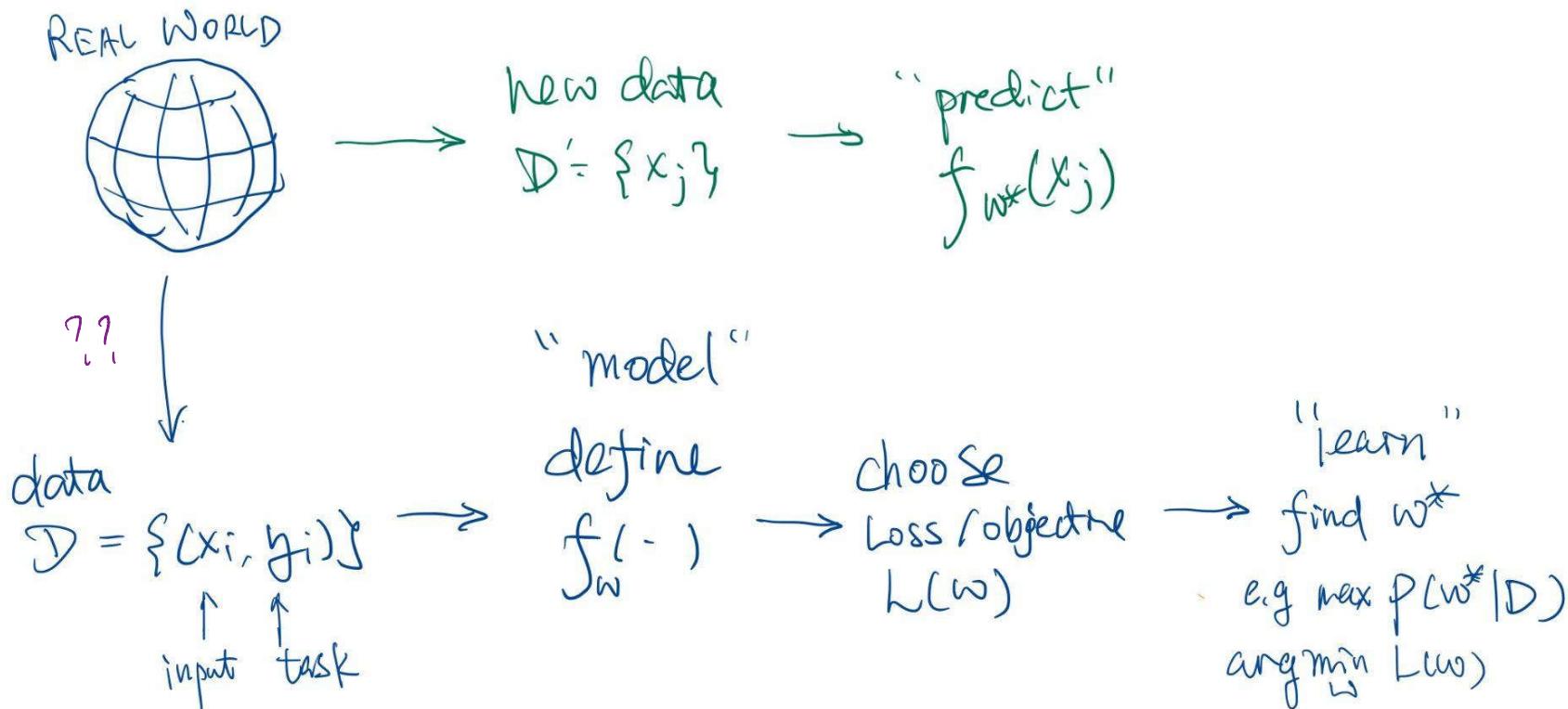




**Figure 1.27** Example of the class-conditional densities for two classes having a single input variable  $x$  (left plot) together with the corresponding posterior probabilities (right plot). Note that the left-hand mode of the class-conditional density  $p(x|C_1)$ , shown in blue on the left plot, has no effect on the posterior probabilities. The vertical green line in the right plot shows the decision boundary in  $x$  that gives the minimum misclassification rate.

high level goal of "learning": the non-trivial dependency between  $x, C$

# A modular view of (supervised) ML



# What we covered so far

Classification problems

A glimpse of decision theory (Sec 1.5)

Discriminant functions - why least squares doesn't work here

The perceptron algorithm

Probabilistic generative models - origin of the logistic function

Probabilistic discriminative models

Logistic regression

(Laplace approximation, Bayesian logistic regression)

# Discriminant function

## *Definition*

A **discriminant** is a function that maps from an input vector  $\mathbf{x}$  to one of  $K$  classes, denoted by  $\mathcal{C}_k$ .

- Consider first two classes ( $K = 2$ ).
- Construct a linear function of the inputs  $\mathbf{x}$

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 \quad (4.4)$$

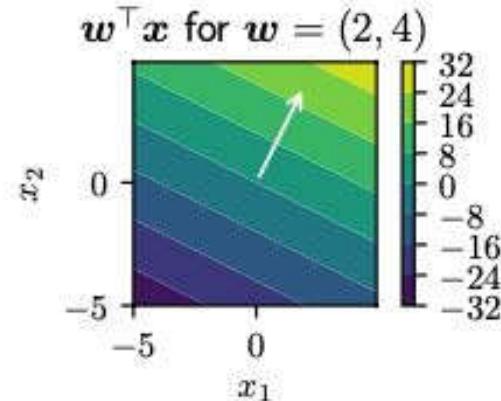
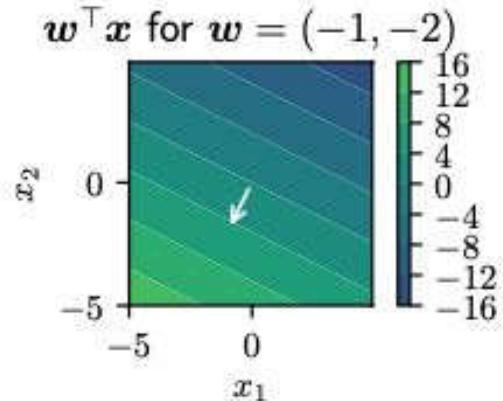
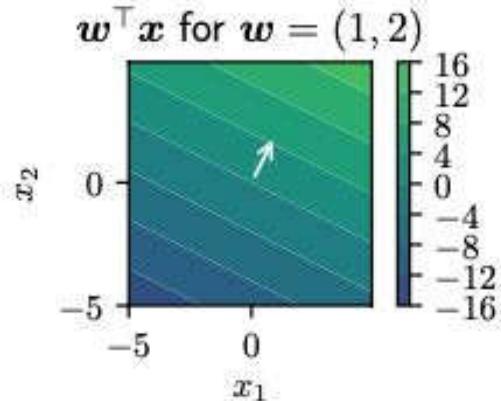
such that  $\mathbf{x}$  being assigned to class  $\mathcal{C}_1$  if  $y(\mathbf{x}) \geq 0$ , and to class  $\mathcal{C}_2$  otherwise.

- **weight vector  $\mathbf{w}$**
- **bias  $w_0$**  ( sometimes  $-w_0$  called **threshold** )

# Linear discriminant functions

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 \quad (4.4)$$

- Decision boundary  $y(\mathbf{x})=0$  -- is a hyperplane of D-1 dimensions (in a D-dimensional input space).
- $\mathbf{w}$  is orthogonal to any vector lying in the decision surface.
- Contour lines of  $y(\mathbf{x})$  are hyperplanes too
- $y(\mathbf{x})$  is the signed distance of point  $\mathbf{x}$  from the hyper-plane (see next page)



# Discriminant for two classes

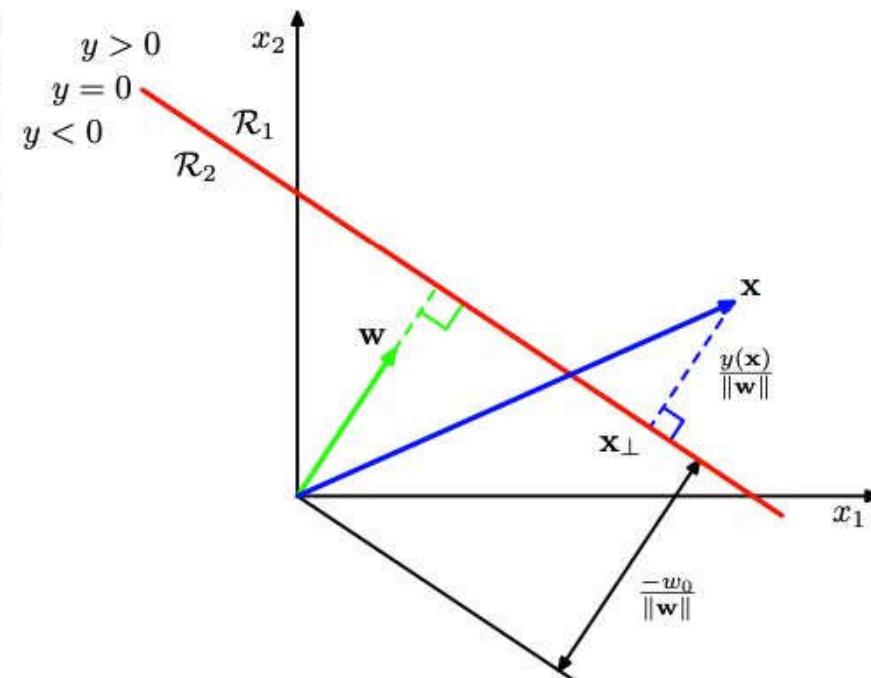
$y(\mathbf{x})$  gives a **signed** measure of the perpendicular distance  $r$  from the decision surface to  $\mathbf{x}$ , that is  $r = y(\mathbf{x})/\|\mathbf{w}\|$ .

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

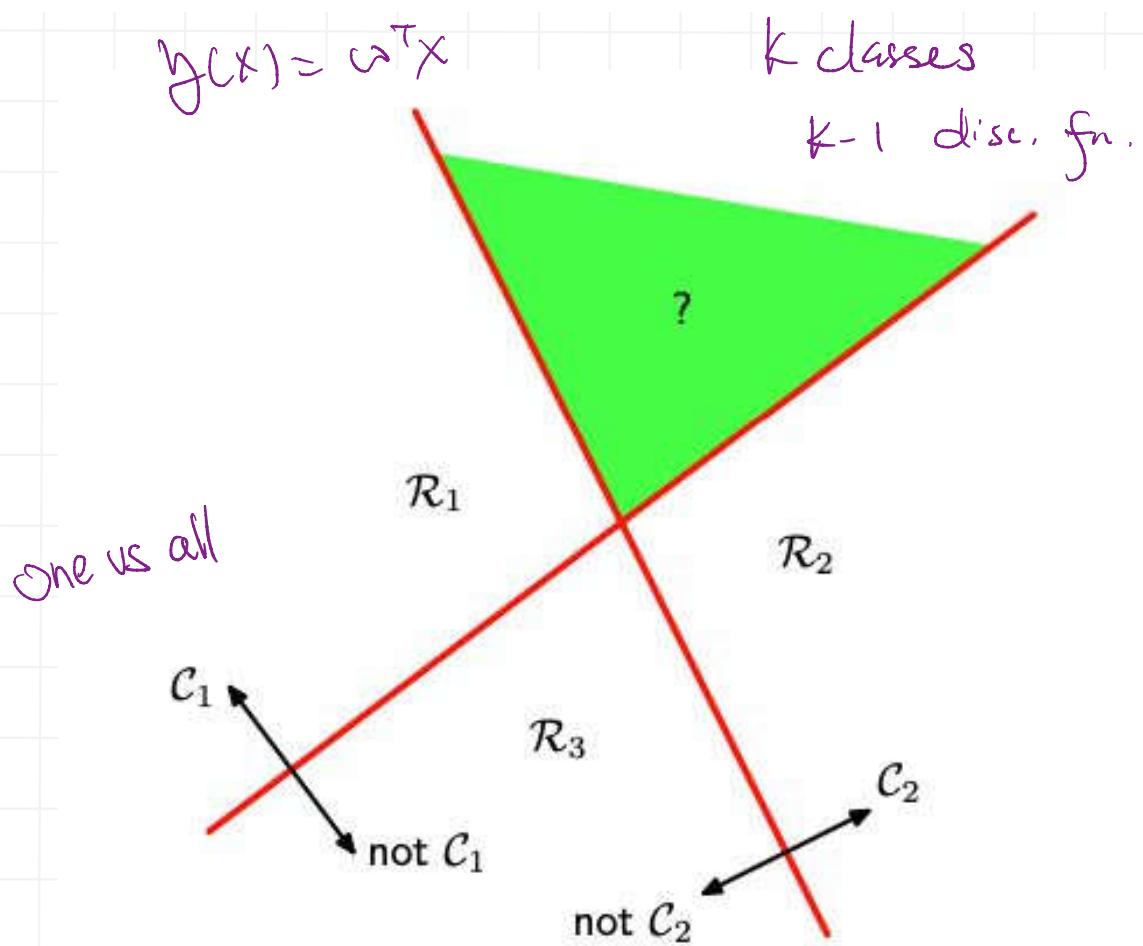
(4.4)

$$y(\mathbf{x}) = \mathbf{w}^T \left( \overbrace{\mathbf{x}_\perp + r \frac{\mathbf{w}}{\|\mathbf{w}\|}}^{\mathbf{x}} \right) + w_0 = r \frac{\mathbf{w}^T \mathbf{w}}{\|\mathbf{w}\|} + \overbrace{\mathbf{w}^T \mathbf{x}_\perp + w_0}^0 = r \|\mathbf{w}\|$$

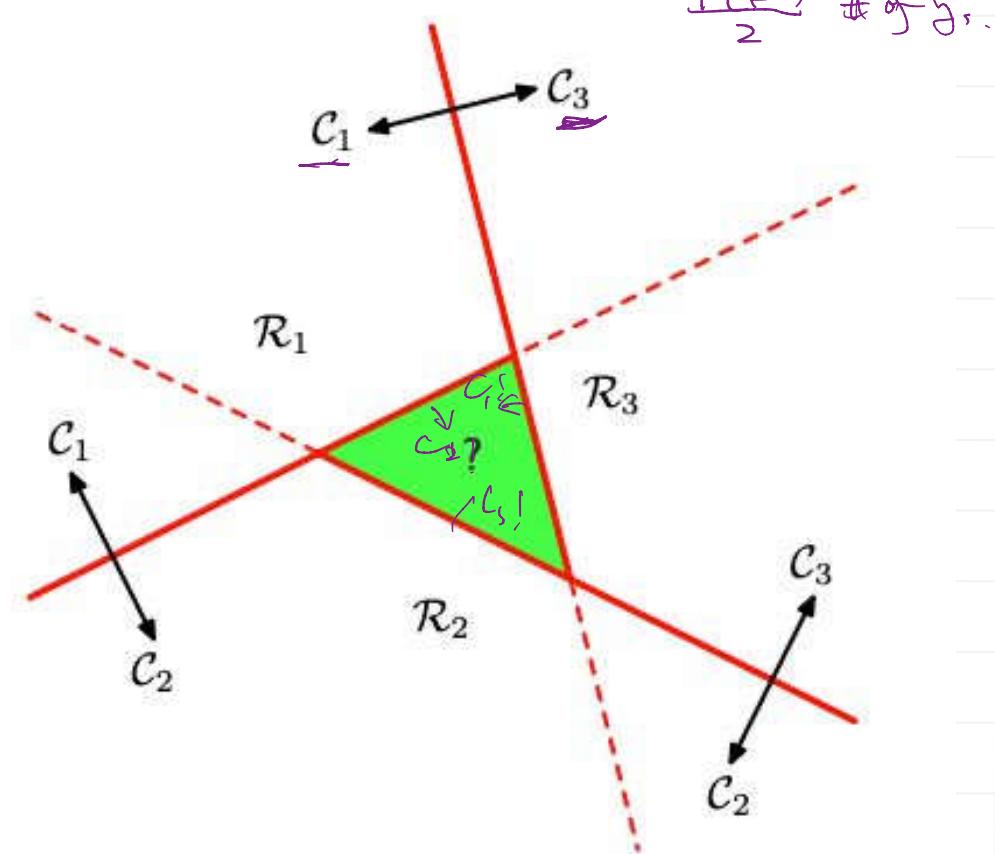
**Figure 4.1** Illustration of the geometry of a linear discriminant function in two dimensions. The decision surface, shown in red, is perpendicular to  $\mathbf{w}$ , and its displacement from the origin is controlled by the bias parameter  $w_0$ . Also, the signed orthogonal distance of a general point  $\mathbf{x}$  from the decision surface is given by  $y(\mathbf{x})/\|\mathbf{w}\|$ .



$$y(x) = w^T x$$



One vs one



**Figure 4.2** Attempting to construct a  $K$  class discriminant from a set of two class discriminants leads to ambiguous regions, shown in green. On the left is an example involving the use of two discriminants designed to distinguish points in class  $C_k$  from points not in class  $C_k$ . On the right is an example involving three discriminant functions each of which is used to separate a pair of classes  $C_k$  and  $C_j$ .

# Discriminant function for $K \geq 2$ classes

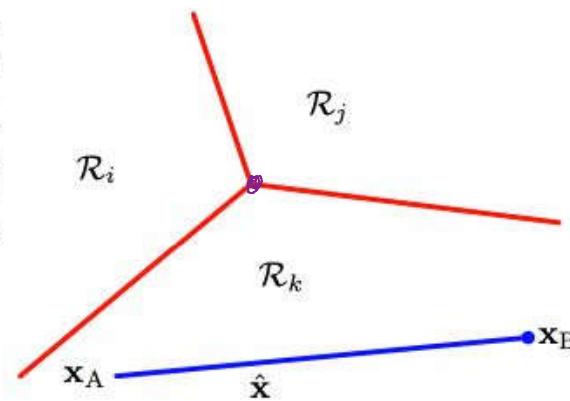
Learn a series of functions  $k=1, \dots, K$ , assigning a point  $\mathbf{x}$  to class  $\mathcal{C}_k$  if  $y_k(\mathbf{x}) > y_j(\mathbf{x})$  for all  $j \neq k$ .

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0} \quad (4.9)$$

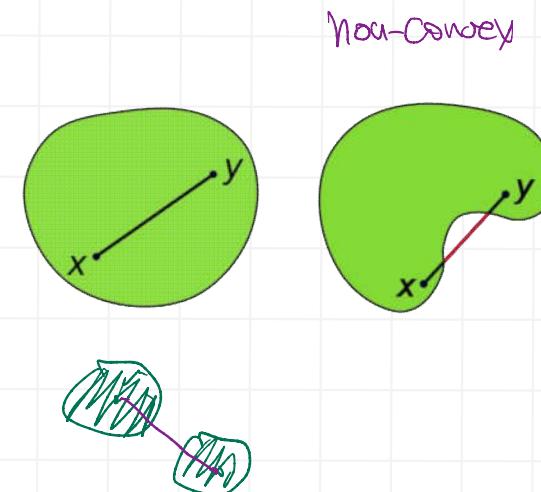
$$k^* \leftarrow \operatorname{argmax}_k y_k(\mathbf{x})$$

Claim: The decision regions of such a discriminant are always singly connected and convex.

**Figure 4.3** Illustration of the decision regions for a multiclass linear discriminant, with the decision boundaries shown in red. If two points  $\mathbf{x}_A$  and  $\mathbf{x}_B$  both lie inside the same decision region  $\mathcal{R}_k$ , then any point  $\hat{\mathbf{x}}$  that lies on the line connecting these two points must also lie in  $\mathcal{R}_k$ , and hence the decision region must be singly connected and convex.



$$y_k(\hat{\mathbf{x}}) = \lambda y_k(\mathbf{x}_A) + (1 - \lambda) y_k(\mathbf{x}_B).$$



Can we use least squares?

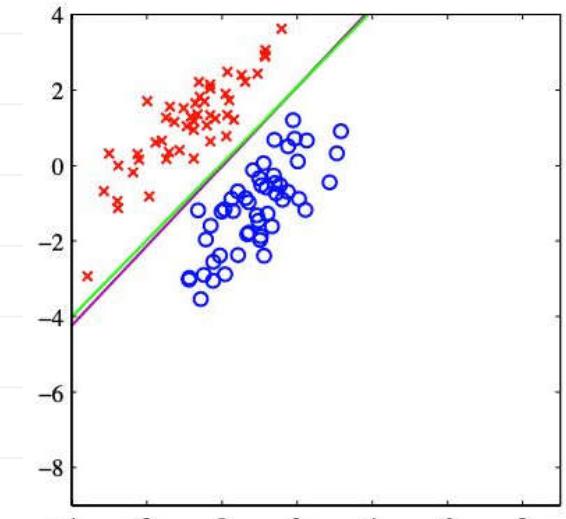
$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0} \quad (4.13)$$

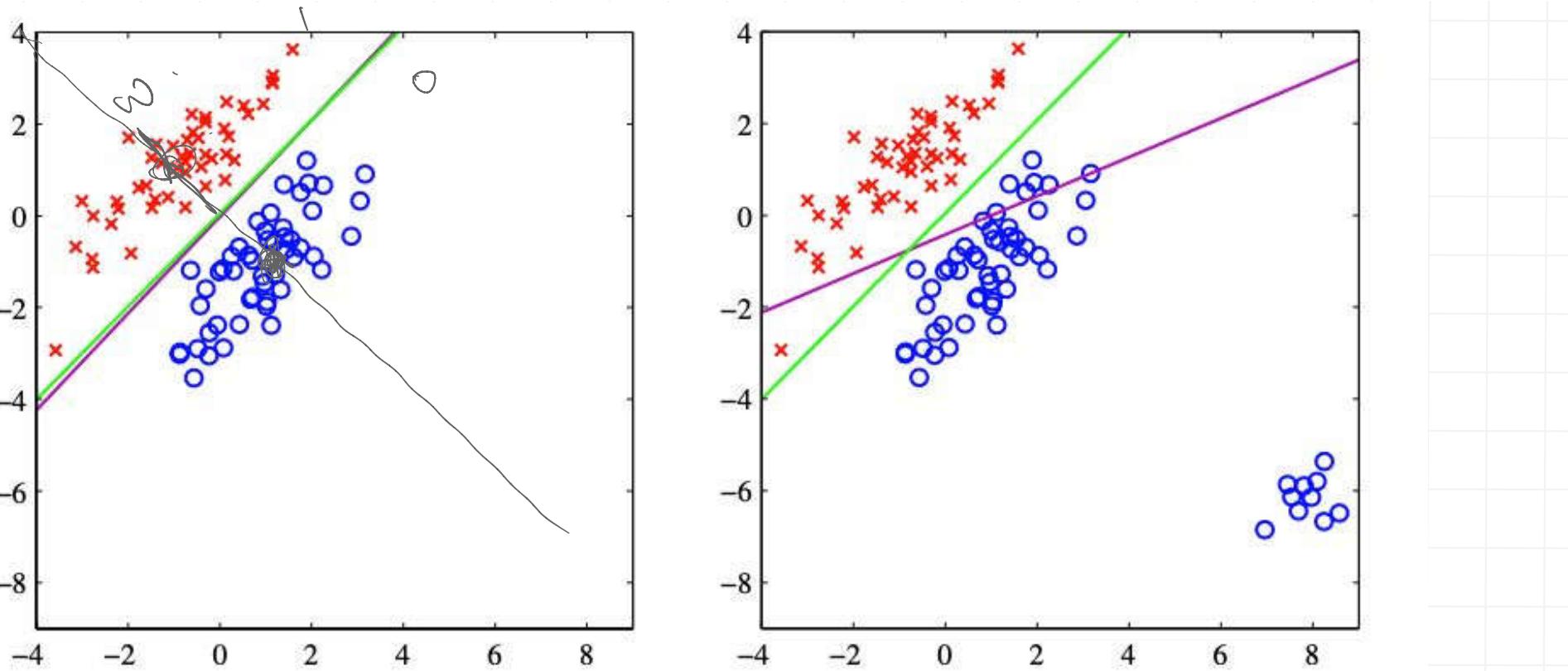
$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^T \widetilde{\mathbf{x}} \quad (4.14)$$

$$E_D(\widetilde{\mathbf{W}}) = \frac{1}{2} \text{Tr} \left\{ (\widetilde{\mathbf{X}} \widetilde{\mathbf{W}} - \mathbf{T})^T (\widetilde{\mathbf{X}} \widetilde{\mathbf{W}} - \mathbf{T}) \right\}. \quad (4.15)$$

*t<sub>u</sub> ∈ {0, 1}*

$$\widetilde{\mathbf{W}} = (\widetilde{\mathbf{X}}^T \widetilde{\mathbf{X}})^{-1} \widetilde{\mathbf{X}}^T \mathbf{T} = \widetilde{\mathbf{X}}^\dagger \mathbf{T} \quad (4.16)$$





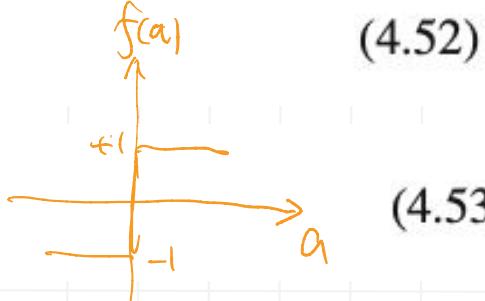
**Figure 4.4** The left plot shows data from two classes, denoted by red crosses and blue circles, together with the decision boundary found by least squares (magenta curve) and also by the logistic regression model (green curve), which is discussed later in Section 4.3.2. The right-hand plot shows the corresponding results obtained when extra data points are added at the bottom left of the diagram, showing that least squares is highly sensitive to outliers, unlike logistic regression.

Mismatch between model requirement and model specification.

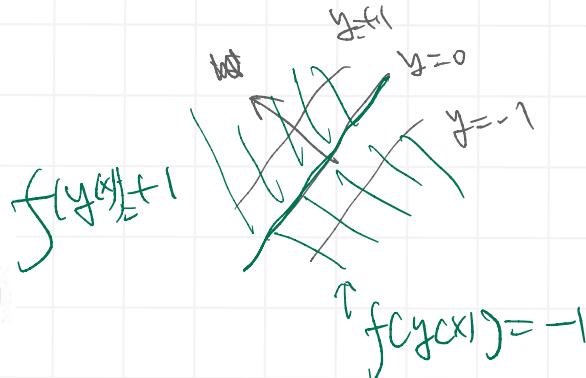
# The (historically significant) perceptron [Rosenblatt 1962]

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

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0. \end{cases}$$



$$(4.53)$$



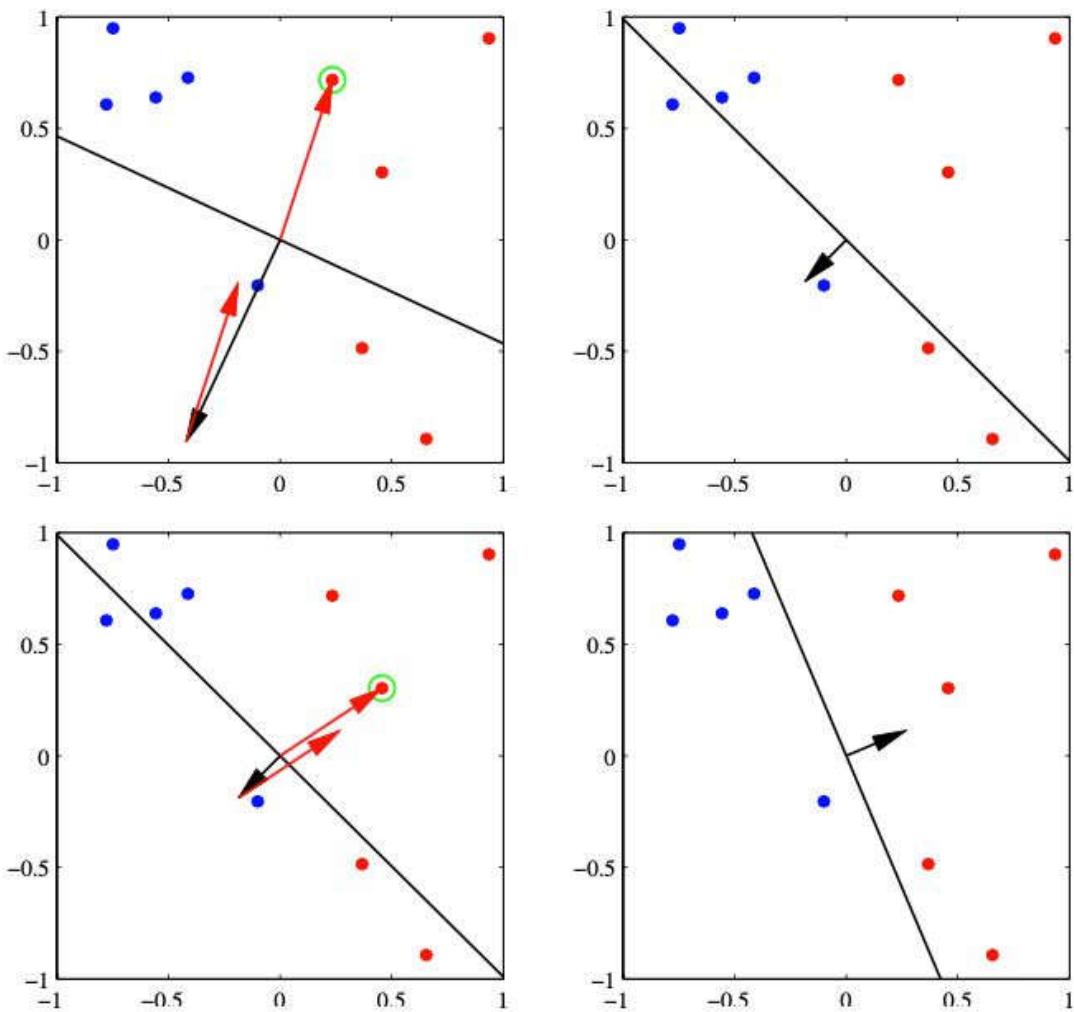
$$t \in \{-1, +1\}$$

$$\min_{\mathbf{w}} E_P(\mathbf{w}) = - \sum_{n \in \mathcal{M}} \mathbf{w}^T \underline{\phi_n t_n} \quad (4.54)$$

Stochastic gradient descent

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_P(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta \underline{\phi_n t_n} \quad (4.55)$$

"The **perceptron convergence theorem** states that if there exists an exact solution (in other words, if the training data set is linearly separable), then the perceptron learning algorithm is guaranteed to find an exact solution in a finite number of steps... however, until convergence is achieved, we will not be able to distinguish between a nonseparable problem and one that is simply slow to converge."



**Figure 4.7** Illustration of the convergence of the perceptron learning algorithm, showing data points from two classes (red and blue) in a two-dimensional feature space ( $\phi_1, \phi_2$ ). The top left plot shows the initial parameter vector  $w$  shown as a black arrow together with the corresponding decision boundary (black line), in which the arrow points towards the decision region which classified as belonging to the red class. The data point circled in green is misclassified and so its feature vector is added to the current weight vector, giving the new decision boundary shown in the top right plot. The bottom left plot shows the next misclassified point to be considered, indicated by the green circle, and its feature vector is again added to the weight vector giving the decision boundary shown in the bottom right plot for which all data points are correctly classified.



**Figure 4.8** Illustration of the Mark 1 perceptron hardware. The photograph on the left shows how the inputs were obtained using a simple camera system in which an input scene, in this case a printed character, was illuminated by powerful lights, and an image focussed onto a  $20 \times 20$  array of cadmium sulphide photocells, giving a primitive 400 pixel image. The perceptron also had a patch board, shown in the middle photograph, which allowed different configurations of input features to be tried. Often these were wired up at random to demonstrate the ability of the perceptron to learn without the need for precise wiring, in contrast to a modern digital computer. The photograph on the right shows one of the racks of adaptive weights. Each weight was implemented using a rotary variable resistor, also called a potentiometer, driven by an electric motor thereby allowing the value of the weight to be adjusted automatically by the learning algorithm.

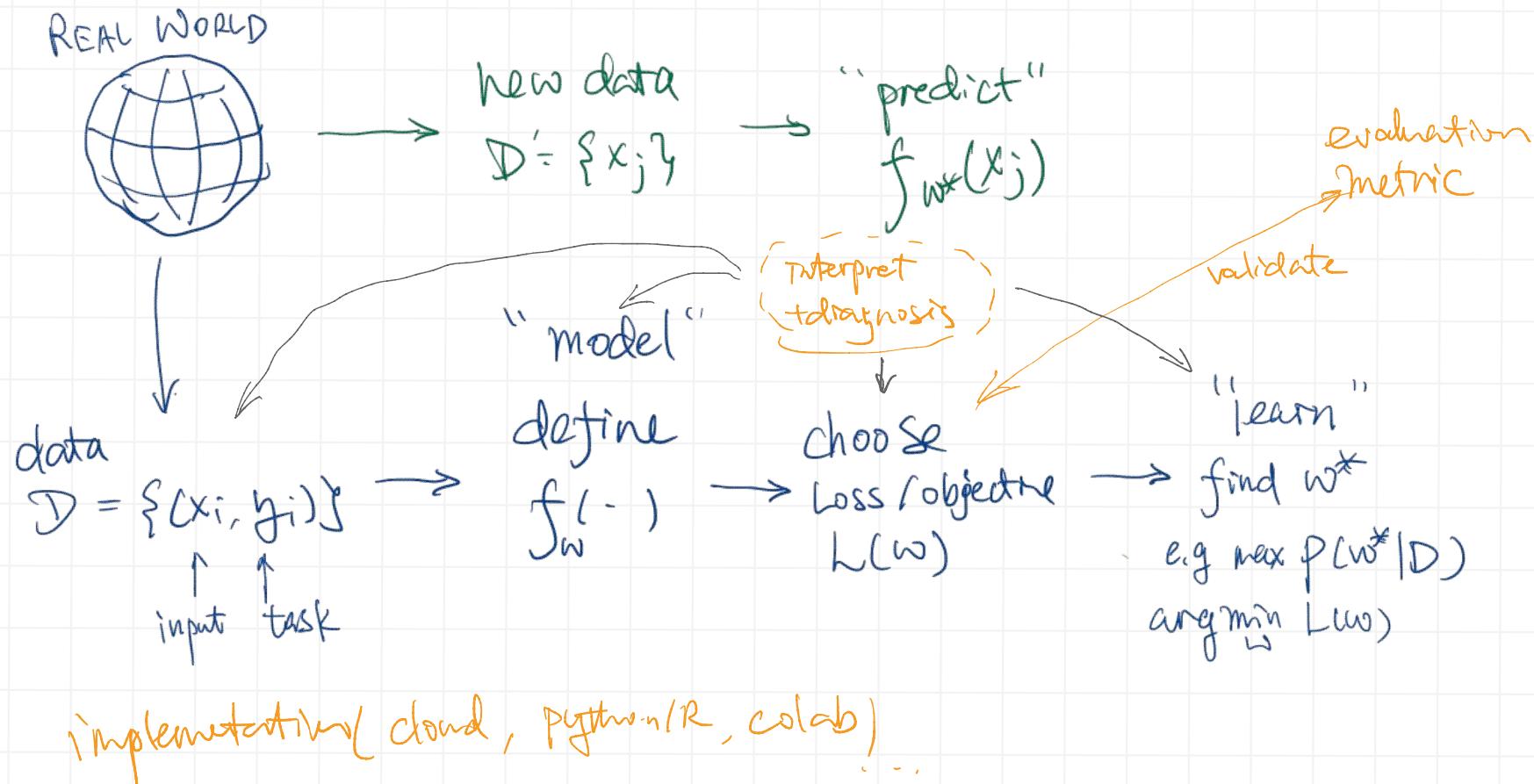
## Frank Rosenblatt 1928–1969



Rosenblatt's perceptron played an important role in the history of machine learning. Initially, Rosenblatt simulated the perceptron on an IBM 704 computer at Cornell in 1957, but by the early 1960s he had built special-purpose hardware that provided a direct, parallel implementation of perceptron learning. Many of his ideas were encapsulated in "Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms" published in 1962. Rosenblatt's work was criticized by Marvin Minsky, whose objections were published in the book "Perceptrons", co-authored with

Papert. This book was widely misinterpreted at the time as showing that neural networks were fatally flawed and could only learn solutions for linearly separable problems. In fact, it only proved such limitations in the case of single-layer networks such as the perceptron and merely conjectured (incorrectly) that they applied to more general network models. Unfortunately, however, this book contributed to the substantial decline in research funding for neural computing, a situation that was not reversed until the mid-1980s. Today, there are many hundreds, if not thousands, of applications of neural networks in widespread use, with examples in areas such as handwriting recognition and information retrieval being used routinely by millions of people.

# Another look at the modular view of supervised ML



# What we covered so far

Classification problems

A glimpse of decision theory (Sec 1.5)

Discriminant functions - why least squares doesn't work here

The perceptron algorithm

Probabilistic generative models - [origin of the logistic function](#)

Probabilistic discriminative models

Logistic regression

(Laplace approximation, Bayesian logistic regression)

## Bayes theorem, again

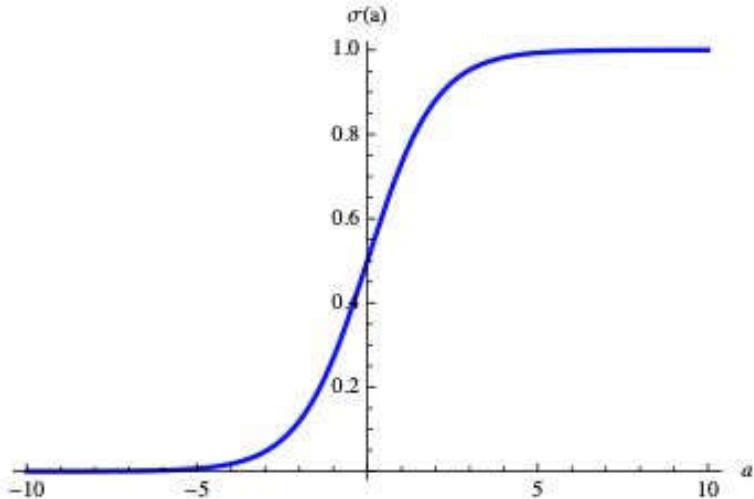
$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \leftarrow \rho(\mathbf{x})$$

$$a(\mathbf{x}) = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} = \ln \frac{p(\mathbf{x}, \mathcal{C}_1)}{p(\mathbf{x}, \mathcal{C}_2)}$$

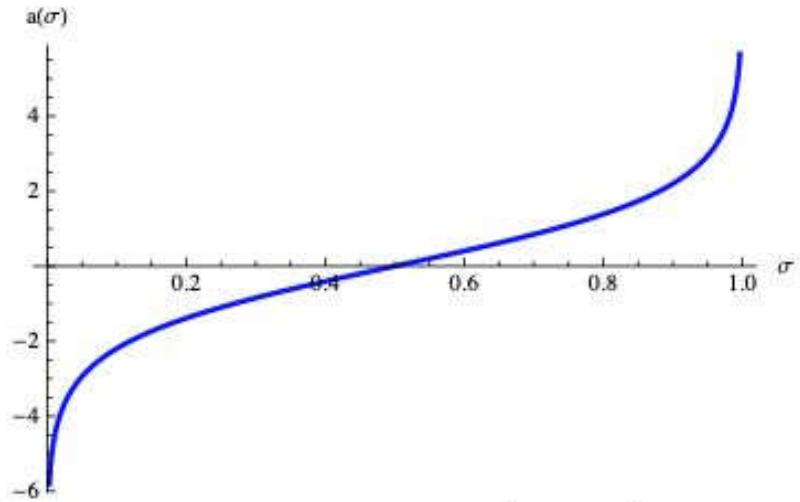
$$\sigma(a) = \frac{1}{1 + \exp(-a)}.$$

logistic sigmoid function

$$= \frac{1}{1 + \exp(-a)} = \sigma(a) \quad (4.57)$$



$$\text{Sigmoid } \sigma(a) = \frac{1}{1+\exp(-a)}$$



$$\text{Logit } a(\sigma) = \ln \left( \frac{\sigma}{1-\sigma} \right)$$

$\ln [p(\mathcal{C}_1|\mathbf{x})/p(\mathcal{C}_2|\mathbf{x})]$   
a.k.a log odds

**Symmetry:**  $\sigma(-a) = 1 - \sigma(a)$

**Derivative:**  $\frac{d}{da}\sigma(a) = \sigma(a)\sigma(-a) = \sigma(a)(1 - \sigma(a))$

# Probabilistic Generative Models - Multiclass

- The normalised exponential is given by

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

where

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

- Usually called the softmax function as it is a smoothed version of the arg max function, in particular:

$$a_k \gg a_j \quad \forall j \neq k \Rightarrow \left( p(\mathcal{C}_k | \mathbf{x}) \approx 1 \wedge p(\mathcal{C}_j | \mathbf{x}) \approx 0 \right)$$

- So, softargmax is a more descriptive though less common name.

## Class-conditional distribution being Gaussians

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_k)^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) \right\}. \quad (4.64)$$

cancel    cancel

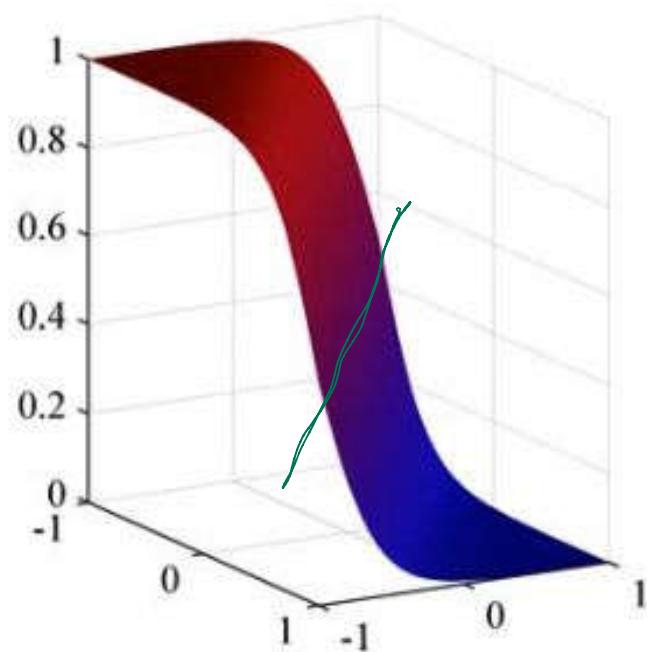
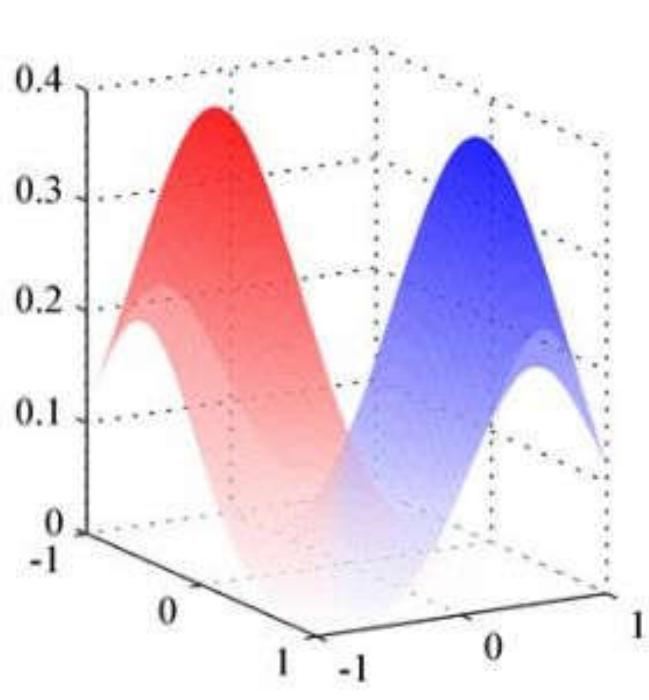
$$a(\mathbf{x}) = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_1)^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}_1) + \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_2)^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}_2) \quad (4.65)$$

two classes  $\mathcal{C}_1, \mathcal{C}_2$   
class-means  $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$   
share cov.  $\Sigma$

$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x} + w_0)$$

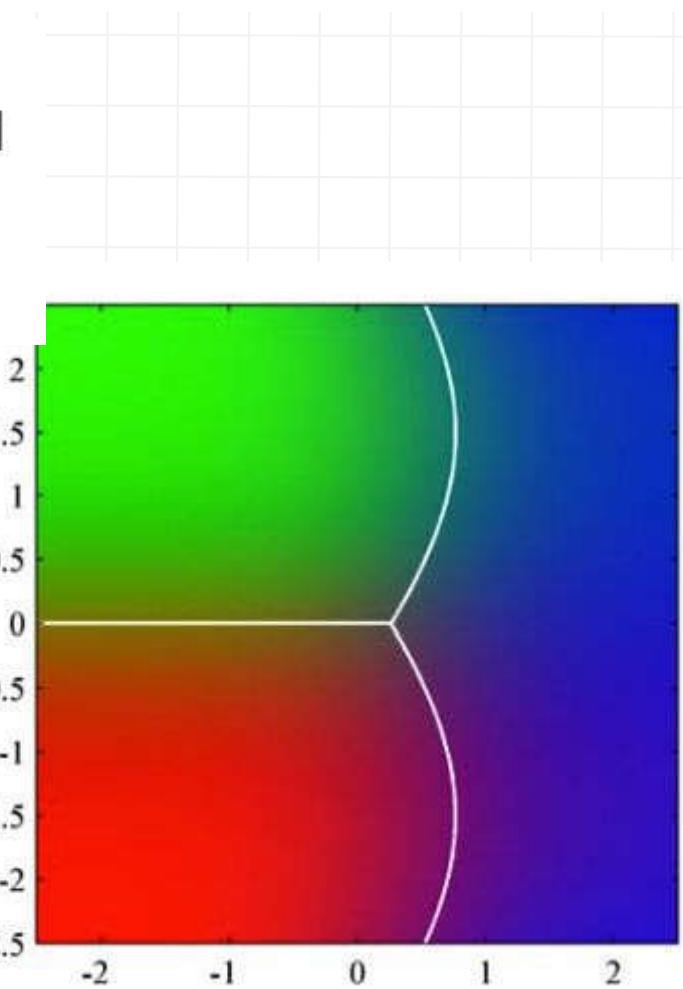
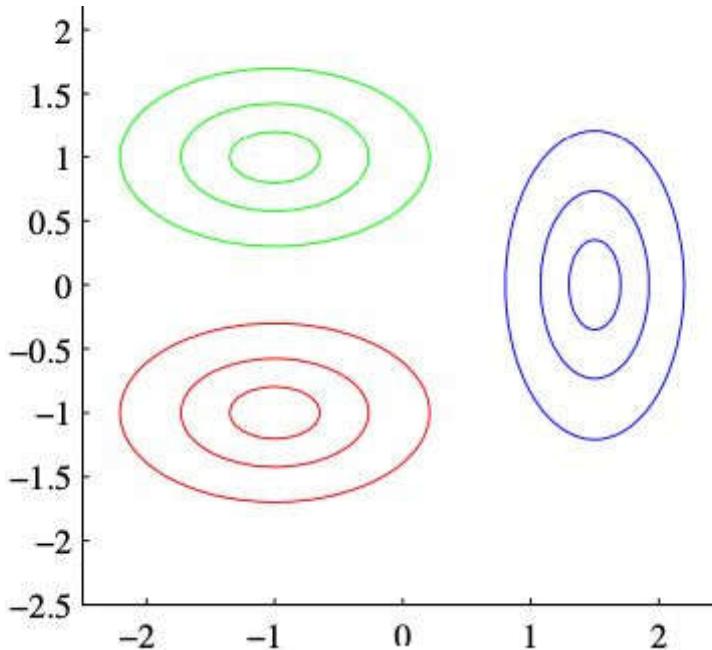
$$\mathbf{w} = \underline{\Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)} \quad (4.66)$$

$$w_0 = -\frac{1}{2} \boldsymbol{\mu}_1^T \Sigma^{-1} \boldsymbol{\mu}_1 + \frac{1}{2} \boldsymbol{\mu}_2^T \Sigma^{-1} \boldsymbol{\mu}_2 + \ln \frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}. \quad (4.67)$$



**Figure 4.10** The left-hand plot shows the class-conditional densities for two classes, denoted red and blue. On the right is the corresponding posterior probability  $p(\mathcal{C}_1|\mathbf{x})$ , which is given by a logistic sigmoid of a linear function of  $\mathbf{x}$ . The surface in the right-hand plot is coloured using a proportion of red ink given by  $p(\mathcal{C}_1|\mathbf{x})$  and a proportion of blue ink given by  $p(\mathcal{C}_2|\mathbf{x}) = 1 - p(\mathcal{C}_1|\mathbf{x})$ .

- If the class-conditional distributions have different covariances, the quadratic terms  $-\frac{1}{2}\mathbf{x}^T \boldsymbol{\Sigma}^{-1} \mathbf{x}$  do not cancel out.
- We get a quadratic discriminant.



**Figure 4.11** The left-hand plot shows the class-conditional densities for three classes each having a Gaussian distribution, coloured red, green, and blue, in which the red and green classes have the same covariance matrix. The right-hand plot shows the corresponding posterior probabilities, in which the RGB colour vector represents the posterior probabilities for the respective three classes. The decision boundaries are also shown. Notice that the boundary between the red and green classes, which have the same covariance matrix, is linear, whereas those between the other pairs of classes are quadratic.

## Maximum likelihood estimates

$$\begin{array}{l} \pi^{P(C_1)} \quad P(C_2) = 1 - \pi \\ \{\pi, \mu_1, \mu_2, \Sigma\} \end{array}$$

Given the functional form of the class-conditional densities  $p(\mathbf{x} | C_k)$ , how can we determine the parameters  $\mu$  and  $\Sigma$  and the class prior?

Q9

$$\underbrace{p(\mathbf{x}_n, C_1)}_{p(\mathbf{x}_n, C_2)} = p(C_1)p(\mathbf{x}_n | C_1) = \pi \mathcal{N}(\mathbf{x}_n | \mu_1, \Sigma)$$

$$p(\mathbf{x}_n, C_2) = p(C_2)p(\mathbf{x}_n | C_2) = (1 - \pi) \mathcal{N}(\mathbf{x}_n | \mu_2, \Sigma)$$

$\{C_1, C_2\} \{0, 1\}$

$$p(\mathbf{t}, \mathbf{X} | \underbrace{\pi, \mu_1, \mu_2, \Sigma}_{\{C_1, C_2\}})$$

$$= \prod_{n=1}^N [\pi \mathcal{N}(\mathbf{x}_n | \mu_1, \Sigma)]^{t_n} \times [(1 - \pi) \mathcal{N}(\mathbf{x}_n | \mu_2, \Sigma)]^{1-t_n}$$

$t_n = C_1 \text{ or } t_n = C_2$

$$L = \log \left\{ \prod_{n=1}^N [\pi \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_1, \boldsymbol{\Sigma})]^{t_n} \times [(1 - \pi) \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_2, \boldsymbol{\Sigma})]^{1-t_n} \right\}$$

$\sum_{n=1}^N t_n [\log \pi + \log \mathcal{N}(\cdot)] + (1-t_n) [\log(1-\pi) + \log \mathcal{N}(\cdot)]$

The term depending on  $\pi$  is

$$\frac{\partial L}{\partial \pi} = \frac{\partial}{\partial \pi} \sum_{n=1}^N (t_n \ln \pi + (1-t_n) \ln(1-\pi))$$

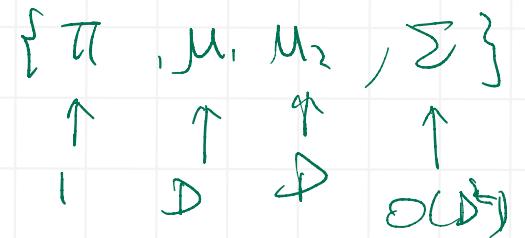
$$\rightarrow \sum_{n=1}^N t_n \frac{1}{\pi} + (1-t_n) \frac{-1}{1-\pi}$$

The maximum is at:  $\pi = \frac{1}{N} \sum_{n=1}^N t_n = \frac{N_1}{N} = \frac{N_1}{N_1 + N_2}$

## MLE solution for means and covariance

$$\boldsymbol{\mu}_1 = \frac{1}{N_1} \sum_{n=1}^N t_n \mathbf{x}_n \quad (4.75)$$

$$\boldsymbol{\mu}_2 = \frac{1}{N_2} \sum_{n=1}^N (1 - t_n) \mathbf{x}_n \quad (4.76)$$



$$\mathbf{S} = \frac{N_1}{N} \mathbf{S}_1 + \frac{N_2}{N} \mathbf{S}_2 \quad (4.78)$$

$$\mathbf{S}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^T \quad (4.79)$$

$$\mathbf{S}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T. \quad (4.80) \quad O(D^2) \text{ parameters!}$$

Left as reading:  
Sec 4.2.3 naive bayes  
Sec 4.2.4 exponential family

## Discrete features - Naive Bayes

- Assume the input space consists of discrete features, in the simplest case  $x_i \in \{0, 1\}$ .
- For a  $D$ -dimensional input space, a general distribution would be represented by a table with  $2^D$  entries.
- Together with the normalisation constraint, this are  $2^D - 1$  independent variables.
- Grows exponentially with the number of features.
- The **Naïve Bayes** assumption is that, given the class  $\mathcal{C}_k$ , the features are independent of each other:

$$\begin{aligned} p(\mathbf{x} | \mathcal{C}_k) &= \prod_{i=1}^D p(x_i | \mathcal{C}) \\ &= \prod_{i=1}^D \mu_{ki}^{x_i} (1 - \mu_{ki})^{1-x_i} \end{aligned}$$

## Naive Bayes classifier

$$p(\mathbf{x}|\mathcal{C}_k) = \prod_{i=1}^D \mu_{ki}^{x_i} (1 - \mu_{ki})^{1-x_i} \quad (4.81)$$

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

$$a_k(\mathbf{x}) = \sum_{i=1}^D \{x_i \ln \mu_{ki} + (1 - x_i) \ln(1 - \mu_{ki})\} + \ln p(\mathcal{C}_k) \quad (4.82)$$

Linear functions of input  $\mathbf{x}$

# What we covered so far

Classification problems

A glimpse of decision theory (Sec 1.5)

Discriminant functions - why least squares doesn't work here  
(Fisher discriminant)

The perceptron algorithm

Probabilistic generative models – origin of the logistic function

Probabilistic discriminative models

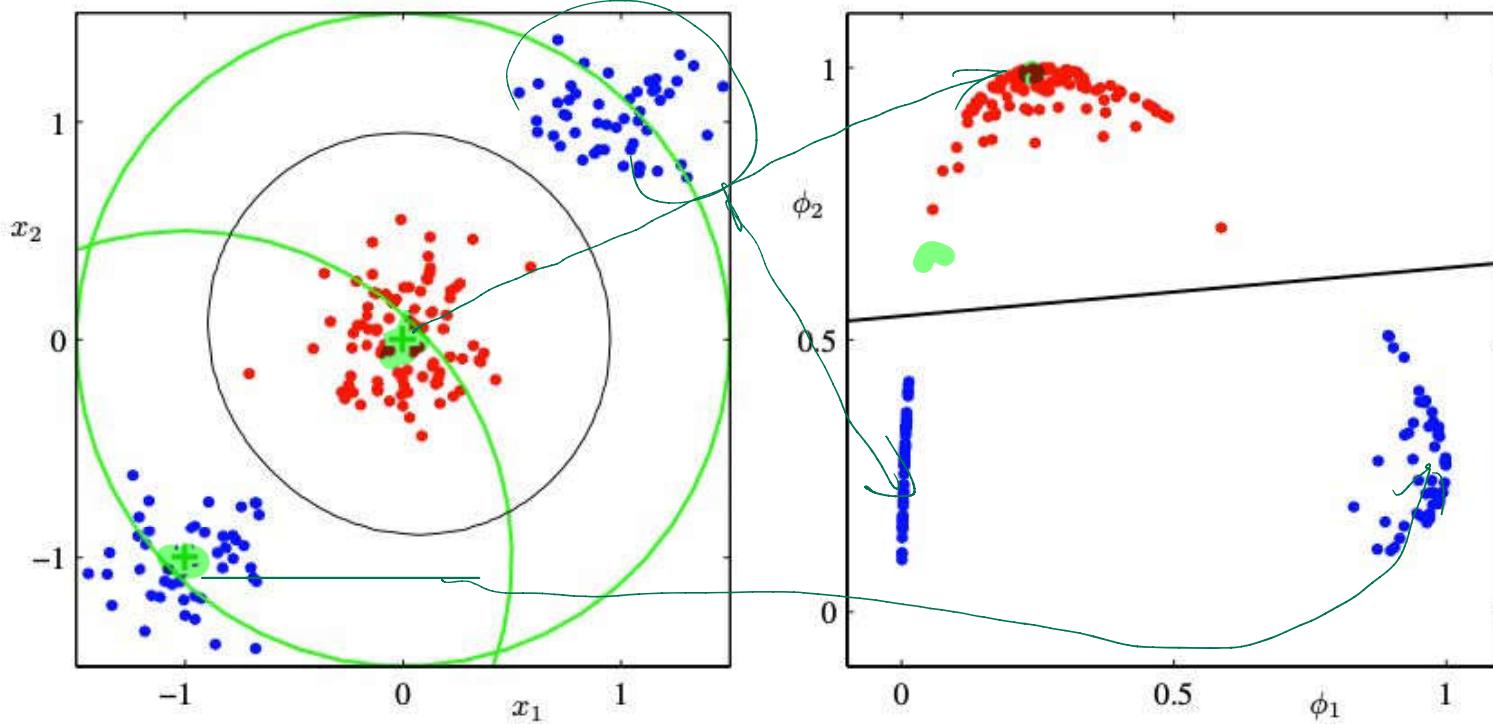
Logistic regression

(Laplace approximation, Bayesian logistic regression)

# Probabilistic Discriminative Models

- **Discriminative** training: learn only to discriminate between the classes.
- Maximise **a likelihood function** defined through the conditional distribution  $p(C_k | \mathbf{x})$  directly.
- Typically fewer parameters to be determined.
- As we learn the posterior  $p(C_k | \mathbf{x})$  directly, prediction may be better than with a generative model where the class-conditional density assumptions  $p(\mathbf{x} | C_k)$  poorly approximate the true distributions.
- But: **discriminative** models can not **create synthetic data**, as  $p(\mathbf{x})$  is not modelled.
- As an aside: *certain theoretical analyses show that generative models converge faster to their — albeit worse — asymptotic classification performance and are superior in some regimes.*

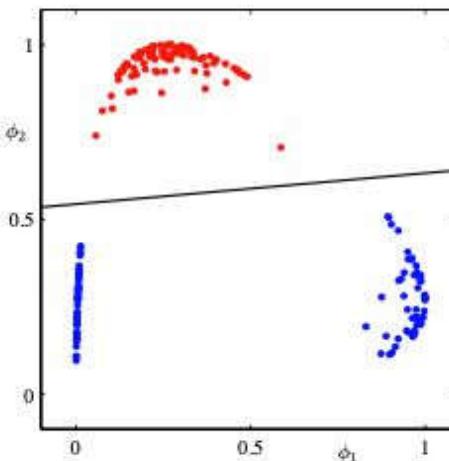
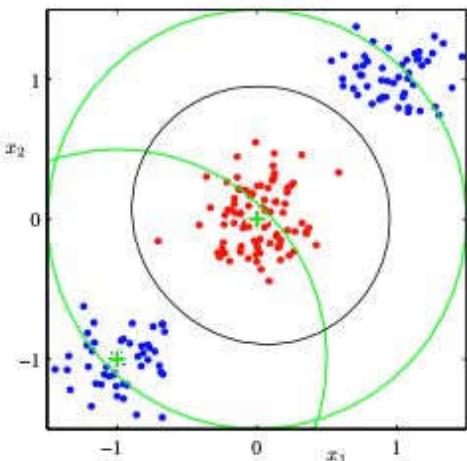
## Fixed basis functions



**Figure 4.12** Illustration of the role of nonlinear basis functions in linear classification models. The left plot shows the original input space  $(x_1, x_2)$  together with data points from two classes labelled red and blue. Two 'Gaussian' basis functions  $\phi_1(x)$  and  $\phi_2(x)$  are defined in this space with centres shown by the green crosses and with contours shown by the green circles. The right-hand plot shows the corresponding feature space  $(\phi_1, \phi_2)$  together with the linear decision boundary obtained given by a logistic regression model of the form discussed in Section 4.3.2. This corresponds to a nonlinear decision boundary in the original input space, shown by the black curve in the left-hand plot.

## Original input vs feature space

- Linear decision boundaries in the feature space generally correspond to nonlinear boundaries in the input space.
- Classes which are NOT linearly separable in the input space may become linearly separable in the feature space:



- If classes overlap in input space, they will also overlap in feature space — nonlinear features  $\phi(x)$  can not remove the overlap; but they may increase it.

# Logistic regression

$$p(C_1|\phi) = y(\phi) = \sigma(w^T \phi) \quad (4.87)$$

→ ignore  $P(C_i)$   
→ ignore shape of  
 $P(x|C_i)$

The name came from statistics. It's a model for classification rather than regression!

Compare to generative model of one Gaussian per class with shared variance

$$p(C_1|x) = \sigma(w^T x + w_0) \quad (4.65)$$

model	$p(x C_1)$	$\sim$ Gaussian
	$p(C_1)$	$\sim$ number -

Same expression for posterior estimates, but linear number of parameters rather than quadratic.

↑  
 $D$  in millions / billions . sparse .

# Maximum likelihood for logistic regression

- Determine the parameter via maximum likelihood for data  $(\phi_n, t_n)$ ,  $n = 1, \dots, N$ , where  $\phi_n = \phi(\mathbf{x}_n)$ . The class membership is coded as  $t_n \in \{0, 1\}$ .
- Likelihood function

$$p(\mathbf{t} | \mathbf{w}) = \prod_{n=1}^N y_n^{t_n} (1 - y_n)^{1-t_n}$$

$y_n = p(C_1 | \phi_n)$

- Error function : negative log likelihood resulting in the **cross-entropy** error function

$$y_n = p(C_1 | \phi_n) = \sigma(\mathbf{w}^T \phi_n)$$

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

↳ assumes every data point counts as the same.

Gradient of Logistic regression

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

$$= -\sum_{n=1}^N \{ t_n \ln \sigma(\omega^T \phi_n) + (1-t_n) \ln(1-\sigma(\omega^T \phi_n)) \}$$

$$\nabla \ln \sigma(\omega^T \phi_n) =$$

$$\nabla \ln(1-\sigma(\omega^T \phi_n)) =$$

$$\nabla E(\omega) =$$

$$y_n = p(C_1 | \phi_n) = \sigma(\omega^T \phi_n)$$

$$\frac{d\sigma}{d\alpha} = \sigma(1-\sigma)$$

Gradient of Logistic regression

$$E(\omega) = -\ln p(\tilde{\omega}) = - \sum_{n=1}^N \{ t_n \ln \underline{y_n} + (1-t_n) \ln (1-\underline{y_n}) \}$$

$$= - \sum_{n=1}^N \left\{ t_n \ln \sigma(\omega^\top \phi_n) + (1-t_n) \ln (1-\sigma(\omega^\top \phi_n)) \right\}$$

$$\nabla_\omega \ln \sigma(\omega^\top \phi_n) = \frac{1}{\sigma(\omega^\top \phi_n)} \sigma(\omega^\top \phi_n) (1-\sigma(\omega^\top \phi_n)) \phi_n$$

$$\rightarrow \frac{d\sigma}{d\alpha} = \sigma(1-\sigma)$$

$$\nabla_\omega E(\omega) = 0$$

→ maybe a local maximum

$$\omega^{(t+1)} = \omega^{(t-1)} + \eta \nabla E(\omega)$$

$$\begin{aligned} \nabla \ln (1-\sigma(\omega^\top \phi_n)) &= \frac{1}{1-\sigma(\omega^\top \phi_n)} [-\sigma(\omega^\top \phi_n) (1-\sigma(\omega^\top \phi_n)) \phi_n] \\ &= -\sigma(\omega^\top \phi_n) \phi_n \end{aligned}$$

$$\begin{aligned} \nabla E(\omega) &= - \sum_{n=1}^N \left\{ \underbrace{t_n (1-\sigma(\omega^\top \phi_n)) \phi_n}_{\text{scalar}} + (1-t_n) (-\sigma(\omega^\top \phi_n) \phi_n) \right\} \\ &= - \sum_{n=1}^N \left\{ t_n \phi_n + t_n \sigma(\omega^\top \phi_n) \phi_n - \sigma(\omega^\top \phi_n) \phi_n + t_n \sigma(\omega^\top \phi_n) \phi_n \right\} \\ \boxed{\nabla E(\omega)} &= \sum_{n=1}^N (\underline{y_n} - t_n) \phi_n \end{aligned}$$

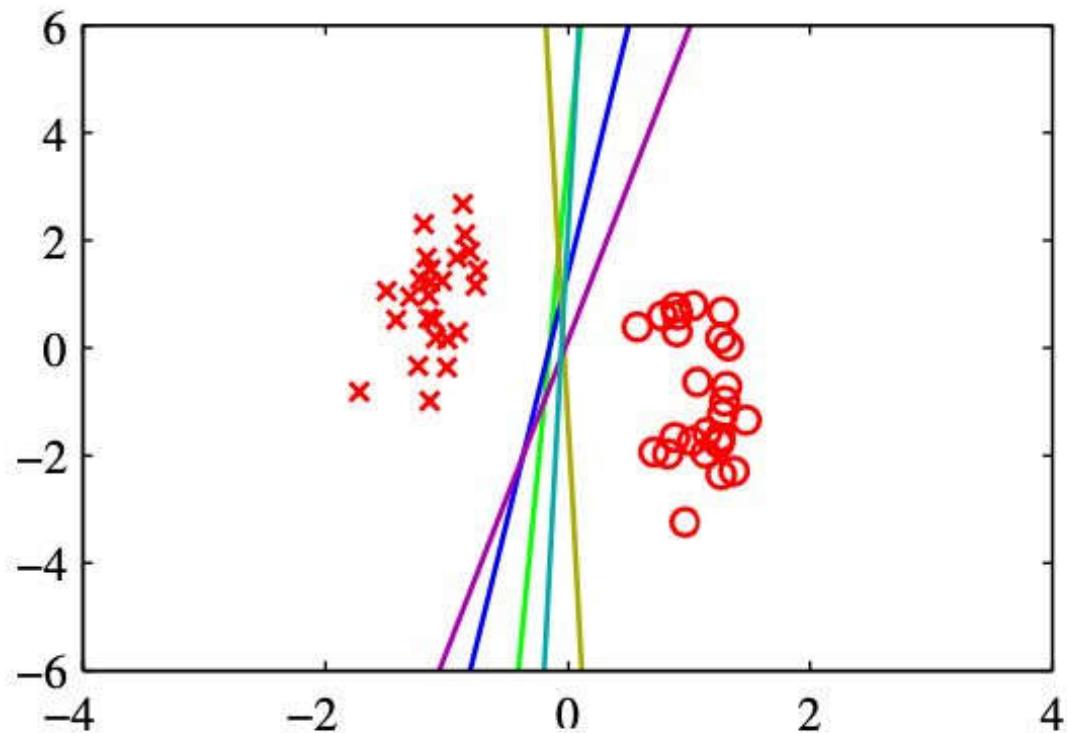
stochastic gradient

$$\nabla E_n(\omega) = \underbrace{(y_n - t_n) \phi_n}_{\text{error}} -$$

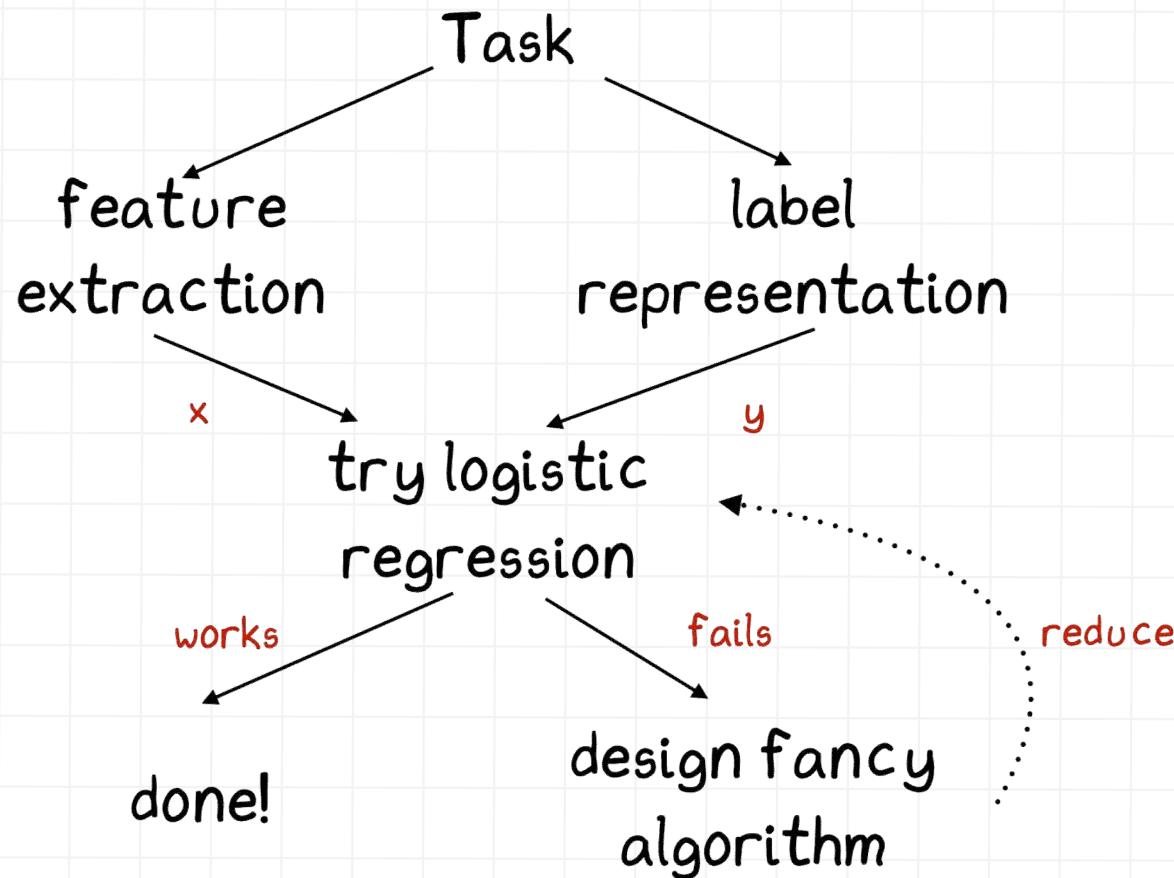
# A critique of logistic regression

$$P(C_1(x) = 1 | \underline{w}^T x)$$

PRML Fig 10.13



# the machine learning “master algorithm\*”



\*Not included: optimisation, inference, asymptotic bounds, evaluation setting

Courtesy of Aditya Menon  
(Google Research)

# Summary of Linear Models for classification

Classification problems

A glimpse of decision theory (Sec 1.5)

Discriminant functions - why least squares doesn't work here

The perceptron algorithm

Probabilistic generative models - origin of the logistic function

Probabilistic discriminative models

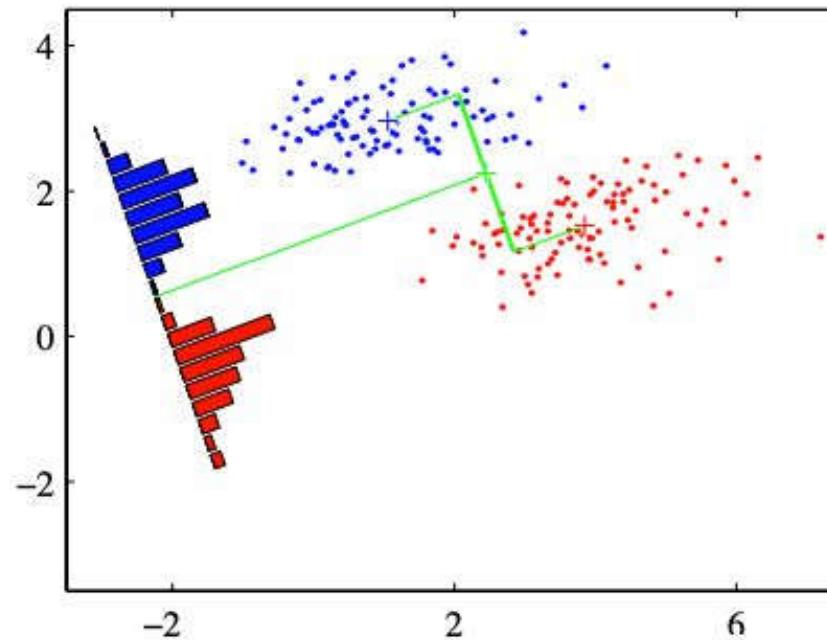
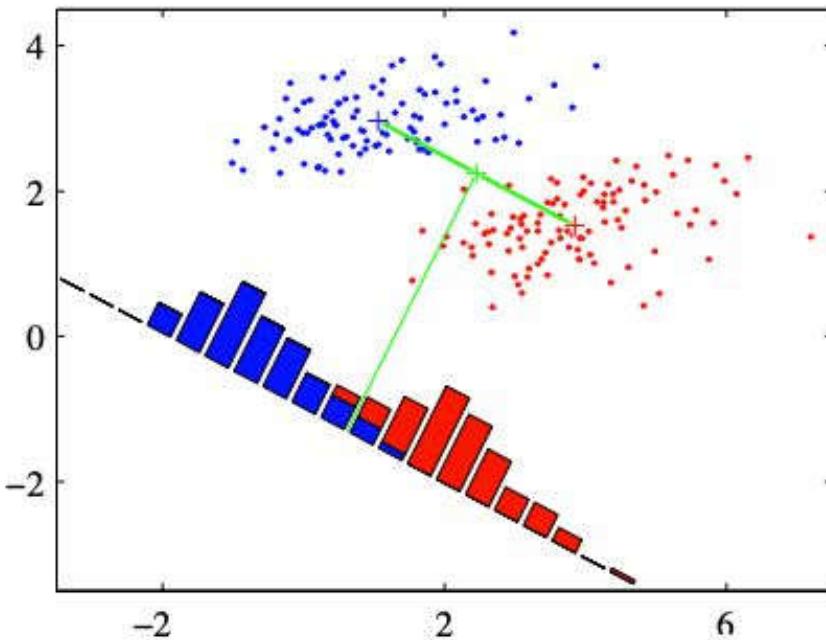
Logistic regression

(Laplace approximation, Bayesian logistic regression) – later in the semester

Origin of the logistic function, logistic regression and how it connects to perceptron

# Laplace approximation

## Other linear discriminants (e.g. Fisher)

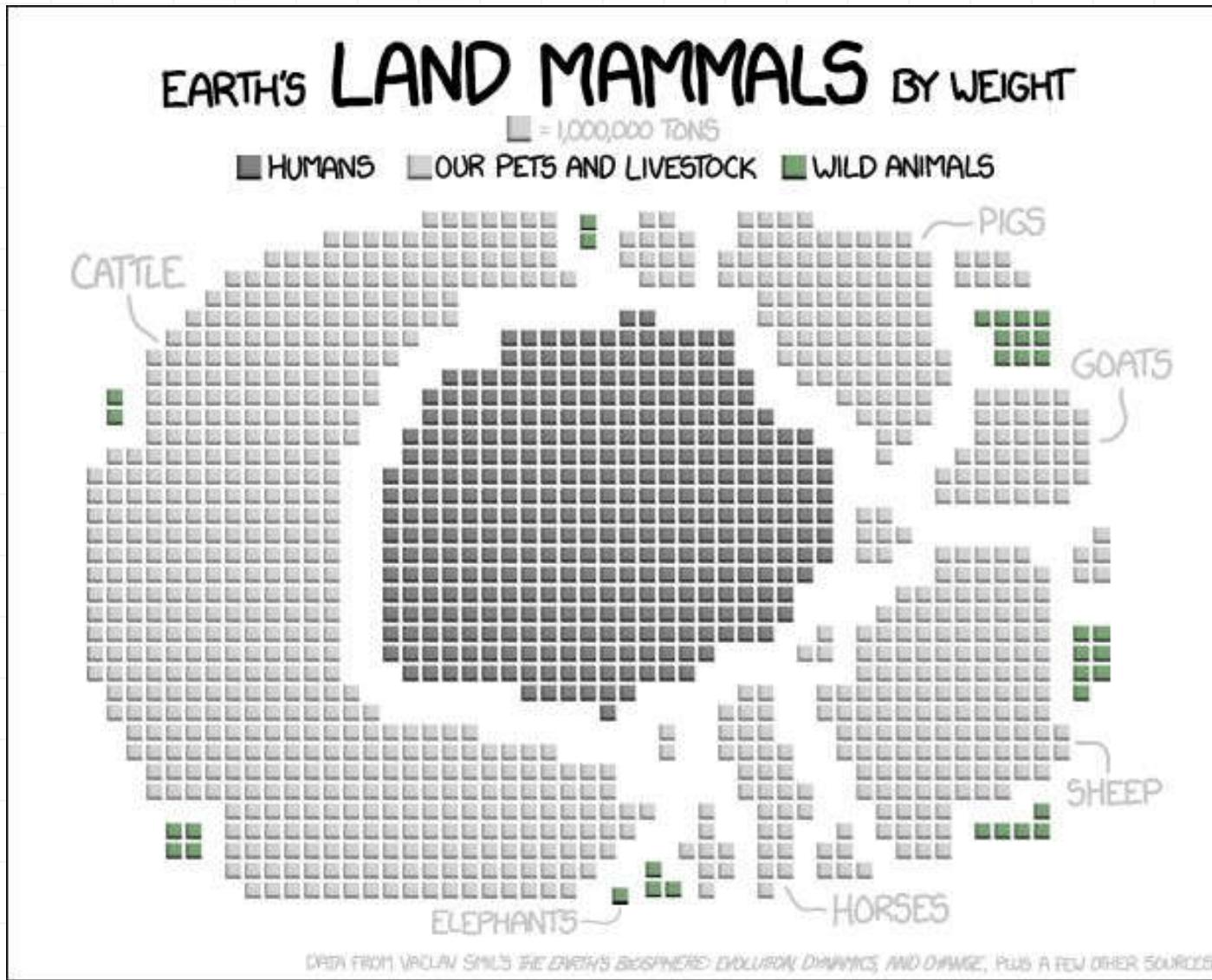


**Figure 4.6** The left plot shows samples from two classes (depicted in red and blue) along with the histograms resulting from projection onto the line joining the class means. Note that there is considerable class overlap in the projected space. The right plot shows the corresponding projection based on the Fisher linear discriminant, showing the greatly improved class separation.

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \quad (4.26)$$

$$\mathbf{w} \propto \mathbf{S}_W^{-1} (\mathbf{m}_2 - \mathbf{m}_1). \quad (4.30)$$

<https://xkcd.com/1338/>



# Mixture Models and Expectation Maximisation

Pre-read/watch: [K-Means algorithm \(PRML 9.1\)](#) and update equations of Gaussian Mixture Models ([MML book Ch 11](#))

EM revisited

- An alternative view of EM
- Connections between GMM and K-means
- Bernoulli mixture

EM in general - does it really maximise likelihood, and why?

Practical considerations and other topics

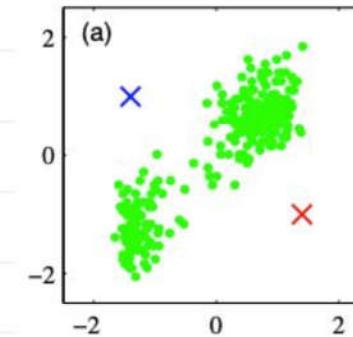
- impossibility of clustering [Kleinberg 2003]
- Kmeans++ [Vassilvitskii and Arthur, 2006]

## Recap: what is clustering

- Given a set of data  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  where  $\mathbf{x}_n \in \mathbb{R}^D$ ,  $n = 1, \dots, N$ .
- Goal: Partition the data into  $K$  clusters.

### K-means

- Each cluster contains points close to each other.
- Introduce a prototype  $\mu_k \in \mathbb{R}^D$  for each cluster.
- Goal: Find
  - a set prototypes  $\mu_k$ ,  $k = 1, \dots, K$ , each representing a different cluster.
  - an assignment of each data point to exactly one cluster.



Luxburg, U.V., Williamson, R.C., & Guyon, I. (2012). Clustering: Science or Art? *ICML Unsupervised and Transfer Learning*.



These ambiguities, redundancies, and deficiencies recall those attributed by Dr. Franz Kuhn to a certain Chinese encyclopedia called the [Heavenly Emporium of Benevolent Knowledge](#). In its distant pages it is written that animals are divided into (a) those that belong to the emperor; (b) embalmed ones; (c) those that are trained; (d) suckling pigs; (e) mermaids; (f) fabulous ones; (g) stray dogs; (h) those that are included in this classification; (i) those that tremble as if they were mad; (j) innumerable ones; (k) those drawn with a very fine camel's-hair brush; (l) etcetera; (m) those that have just broken the flower vase; (n) those that at a distance resemble flies. (Borges, 1999)

(initialise a set of cluster centers / component means)

Expectation step

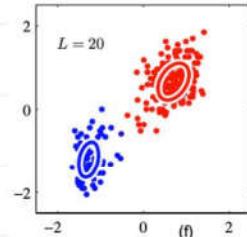
K-means

Re-assign data points to clusters,  
determine  $r_{nk}$

$$\{0, 1\}$$

Gaussian Mixture Models

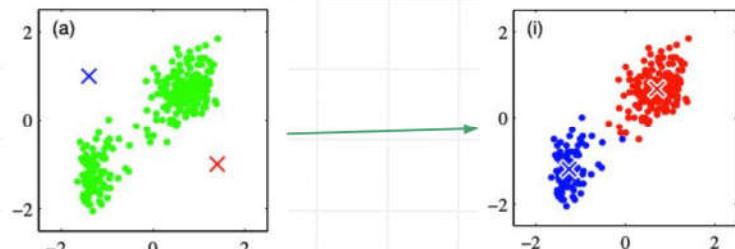
Fig 9.8



Maximisation step

Re-compute the cluster means -  
update  $\{\mu_k\}$

Fig 9.1



2. **E step.** Evaluate the responsibilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \quad (9.23)$$

3. **M step.** Re-estimate the parameters using the current responsibilities

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n \quad (9.24)$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^T \quad (9.25)$$

$$\pi_k^{\text{new}} = \frac{N_k}{N} \quad (9.26)$$

where

$$N_k = \sum_{n=1}^N \gamma(z_{nk}). \quad (9.27)$$

# K-means and GMM - hard vs soft assignments

$\mathbf{x}_n$	$r_{nk}$	
$\theta_k$	$\gamma(z_{nk})$	

For  $k$ -means clustering,  
we have hard assignments

For GMM,  
we have soft assignments

Assume a Gaussian mixture model.

Covariance matrices given by  $\epsilon \mathbf{I}$ , where  $\epsilon$  is shared by all components.

Then

$$p(\mathbf{x} | \mu_k, \Sigma_k) = \frac{1}{(2\pi\epsilon)^{D/2}} \exp \left\{ -\frac{1}{2\epsilon} \|\mathbf{x} - \mu_k\|^2 \right\}.$$

$$\gamma(z_{nk}) = \frac{\pi_k \exp \left\{ -\frac{\|\mathbf{x}_n - \mu_k\|^2}{2\epsilon} \right\}}{\sum_j \pi_j \exp \left\{ -\frac{\|\mathbf{x}_n - \mu_j\|^2}{2\epsilon} \right\}}$$

$\epsilon \downarrow 0$

Taking the limit  $\underbrace{\epsilon \rightarrow 0}$

$$\gamma(z_{nk}) = \begin{cases} 1 & \text{if } \|\mathbf{x}_n - \mu_k\| < \|\mathbf{x}_n - \mu_j\| \quad \forall j \neq k \\ 0 & \text{otherwise} \end{cases}$$

# Wait ... what is being maximised in EM?

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\} \quad (9.28)$$

also (9.14)

$$0 = - \sum_{n=1}^N \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}} \underbrace{\boldsymbol{\Sigma}_k (\mathbf{x}_n - \boldsymbol{\mu}_k)}_{\gamma(z_{nk})} \quad (9.16)$$

cheating :-)

In each maximisation step?

Overall?

**M step.** Re-estimate the parameters using the current responsibilities

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n \quad (9.24)$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^T \quad (9.25)$$

$$\pi_k^{\text{new}} = \frac{N_k}{N} \quad (9.26)$$

# Expectation-maximization revisited

$\mathbf{X}$  observed,  $\mathbf{Z}$  “latent”

$$\ln p(\mathbf{X}|\boldsymbol{\theta}) = \ln \left\{ \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) \right\}. \quad (9.29)$$

$$z_{nk} \in \{0, 1\}.$$

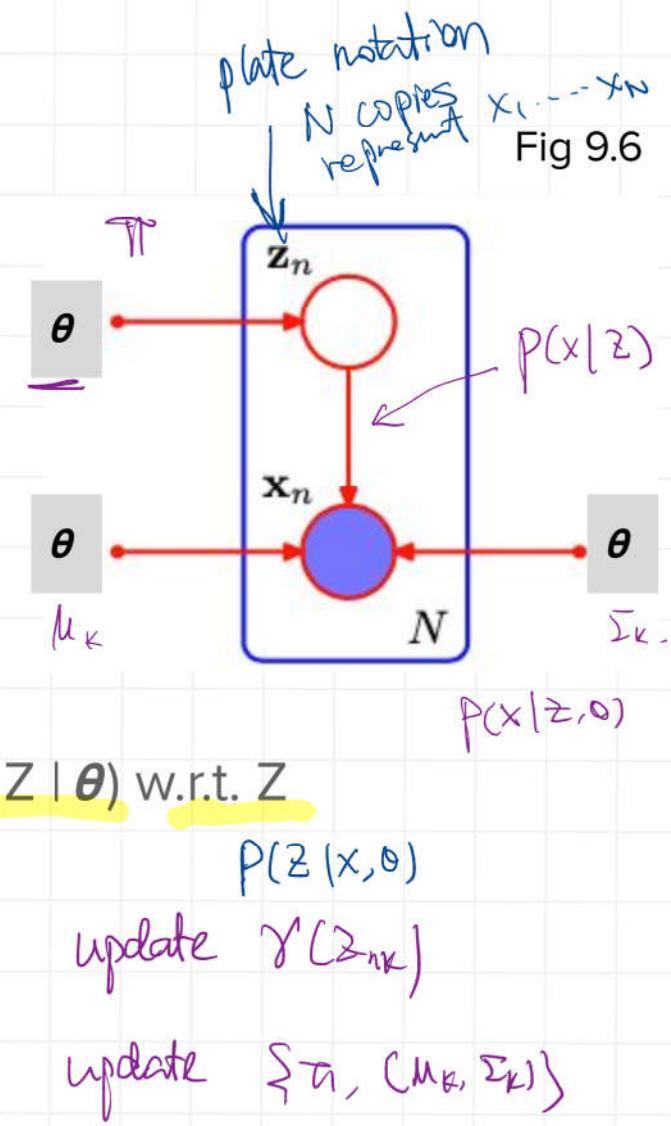
compute posterior  $P(Z|X, \boldsymbol{\theta}) \rightarrow \gamma(z_{nk})$

Take expectation of the complete data log-likelihood  $P(X, Z|\boldsymbol{\theta})$  w.r.t.  $Z$

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}). \quad (9.30)$$

Maximise this expectation w.r.t.  $\boldsymbol{\theta}$

$$\boldsymbol{\theta}^{\text{new}} = \arg \max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}). \quad (9.31)$$

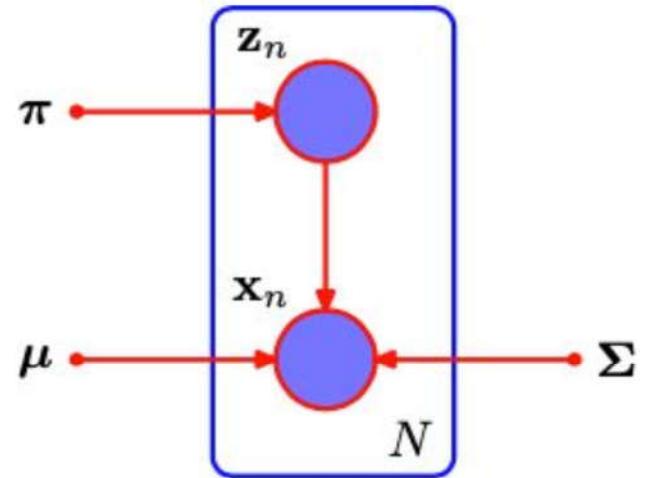


# Convince ourselves that this is true for GMM M-step

**Figure 9.9** This shows the same graph as in Figure 9.6 except that we now suppose that the discrete variables  $z_n$  are observed, as well as the data variables  $x_n$ .

$$\boldsymbol{\theta}^{\text{new}} = \arg \max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}). \quad (9.31)$$

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(x_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\} \quad (9.28)$$



$$p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \prod_{n=1}^N \prod_{k=1}^K \pi_k^{z_{nk}} \mathcal{N}(x_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_{nk}} \quad (9.35)$$

only  $n$ th term is active

$$\ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^N \sum_{k=1}^K z_{nk} \{ \ln \pi_k + \ln \mathcal{N}(x_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \}. \quad (9.36)$$

$$p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) \propto \prod_{n=1}^N \prod_{k=1}^K [\pi_k \mathcal{N}(x_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]^{z_{nk}}. \quad (9.38)$$

$$\mathbb{E}_{\mathbf{Z}} [\ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})] = \sum_{n=1}^N \sum_{k=1}^K \gamma(z_{nk}) \{ \underbrace{\ln \pi_k + \ln \mathcal{N}(x_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}_{\theta^{\text{old}}} \} \quad (9.40)$$

## The General EM Algorithm

Given a joint distribution  $p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$  over observed variables  $\mathbf{X}$  and latent variables  $\mathbf{Z}$ , governed by parameters  $\boldsymbol{\theta}$ , the goal is to maximize the likelihood function  $p(\mathbf{X}|\boldsymbol{\theta})$  with respect to  $\boldsymbol{\theta}$ .

1. Choose an initial setting for the parameters  $\boldsymbol{\theta}^{\text{old}}$ .
2. **E step** Evaluate  $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}})$ .
3. **M step** Evaluate  $\boldsymbol{\theta}^{\text{new}}$  given by MAP objective  
$$\boldsymbol{\theta}^{\text{new}} = \arg \max_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) \quad (9.32) \quad \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) + \ln p(\boldsymbol{\theta})$$

where

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}). \quad (9.33)$$

4. Check for convergence of either the log likelihood or the parameter values.  
If the convergence criterion is not satisfied, then let

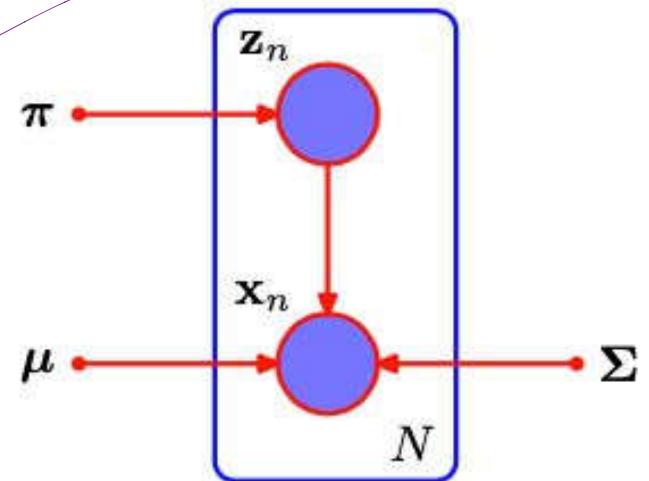
$$\boldsymbol{\theta}^{\text{old}} \leftarrow \boldsymbol{\theta}^{\text{new}} \quad (9.34)$$

and return to step 2.

# EM for GMM, revisited

**Figure 9.9** This shows the same graph as in Figure 9.6 except that we now suppose that the discrete variables  $z_n$  are observed, as well as the data variables  $x_n$ .

$$\ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^N \sum_{k=1}^K z_{nk} \{ \ln \pi_k + \ln \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \}. \quad (9.36)$$



$$\mathbb{E}_{\mathbf{Z}}[\ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})] = \sum_{n=1}^N \sum_{k=1}^K \gamma(z_{nk}) \{ \ln \pi_k + \ln \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \}. \quad (9.40)$$

$$\frac{\partial \mathbb{E}_{\mathbf{Z}}[ \cdot ]}{\partial \boldsymbol{\mu}_k, \pi_k, \dots}$$

## K-means and GMM - differences & connections

$\downarrow \text{MLE}$

$\dots \exp \left\{ (\mathbf{x}_n - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_k) \right\}$

$\frac{q}{\infty} \quad | \quad (9.14)$

$\uparrow \quad \sum_k$

$\vdots$   
 $/$   
 $/$

$$\ln p(\mathbf{X}|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}.$$

$\nearrow \text{only one term active}$

$(9.1)$

$\nearrow$   
 $\nwarrow$

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

$\underbrace{\text{data}}_{\text{of}} \quad \underbrace{\boldsymbol{\mu}_k}_{\text{cluster centers}}$

# K-means and GMM - differences & connections

$$\rightarrow (x - \mu_k)^\top \Sigma_k^{-1} (x - \mu_k)$$

	$\mu_k$	$\Sigma_k$	$\pi_k$	$N_k$
GMM	$\frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n$	$\frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \mu_k) (\mathbf{x}_n - \mu_k)^\top$	$\frac{N_k}{N}$	$\sum_{n=1}^N \gamma(z_{nk}).$

K-Means  $\frac{\sum_n r_{nk} \mathbf{x}_n}{\sum_n r_{nk}}.$

$$\epsilon I$$

Does not matter if  $> 0$

$$\sum_n r_{nk}$$

$$x_n \in \mathbb{R}^D \quad \text{GMM: } k + kD + k \cdot \frac{D(D+1)}{2}$$

$$\text{kmean } k \cdot D$$

Complexity: how many parameters for each?

Implementation:

Can a cluster center "die" in each model, how to handle?  $\rightarrow$  Initialise another center,

What if some Gaussian components are singular?



## D separate Bernoulli Distributions

a set of D binary variables  $x_i$ , each governed by Bernoulli distribution with mean  $\mu_i$

$$\mathbf{x} = \underbrace{(x_1, \dots, x_D)^T}_{D \text{ components}} \text{ and } \boldsymbol{\mu} = (\mu_1, \dots, \mu_D)^T$$

~~$p(\mathbf{x}|\boldsymbol{\mu}) = \prod_{i=1}^D \mu_i^{x_i} (1 - \mu_i)^{1-x_i}$~~

*$x_i$  from the  $i$ -th component on/off.*

$$p(\mathbf{x}|\boldsymbol{\mu}) = \prod_{i=1}^D \mu_i^{x_i} (1 - \mu_i)^{1-x_i} \quad (9.44)$$

$$\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu} \quad (9.45)$$

$$\text{cov}[\mathbf{x}] = \text{diag}\{\mu_i(1 - \mu_i)\}. \quad (9.46)$$

# Mixture of Bernoulli Distributions

$$\mathbf{x} = (x_1, \dots, x_D)^T$$

$$\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K\}, \boldsymbol{\pi} = \{\pi_1, \dots, \pi_K\}$$

$$p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\pi}) = \sum_{k=1}^K \pi_k p(\mathbf{x}|\boldsymbol{\mu}_k) \quad (9.47)$$

$$p(\mathbf{x}|\boldsymbol{\mu}_k) = \prod_{i=1}^D \mu_{ki}^{x_i} (1 - \mu_{ki})^{(1-x_i)}. \quad (9.48)$$

$$\mathbb{E}[\mathbf{x}] = \sum_{k=1}^K \pi_k \boldsymbol{\mu}_k \quad (9.49)$$

$$\text{cov}[\mathbf{x}] = \sum_{k=1}^K \pi_k \left\{ \boldsymbol{\Sigma}_k + \boldsymbol{\mu}_k \boldsymbol{\mu}_k^T \right\} - \mathbb{E}[\mathbf{x}] \mathbb{E}[\mathbf{x}]^T \quad (9.50) \quad \boldsymbol{\Sigma}_k = \text{diag} \{ \mu_{ki}(1 - \mu_{ki}) \}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_D \end{bmatrix}$$

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 & \dots & \boldsymbol{\mu}_K \\ \vdots & \ddots & \vdots \\ \boldsymbol{\mu}_{D1} & \dots & \boldsymbol{\mu}_{DK} \end{bmatrix}$$

$$\mathbb{E}[\mathbf{x}] = \underbrace{(\mathbb{E}[x])^T}_{\mathbf{1}}$$

Data likelihood : (9.51)

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\pi}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k p(\mathbf{x}_n|\boldsymbol{\mu}_k) \right\}.$$

$$p(\mathbf{x}|\mathbf{z}, \boldsymbol{\mu}) = \prod_{k=1}^K p(\mathbf{x}|\boldsymbol{\mu}_k)^{z_k} \quad p(\mathbf{z}|\boldsymbol{\pi}) = \prod_{k=1}^K \pi_k^{z_k}.$$

Complete data likelihood

$$\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\mu}, \boldsymbol{\pi}) = \sum_{n=1}^N \sum_{k=1}^K z_{nk} \left\{ \ln \pi_k + \sum_{i=1}^D [x_{ni} \ln \mu_{ki} + (1 - x_{ni}) \ln(1 - \mu_{ki})] \right\} \quad (9.54)$$

Expectations of complete data likelihood

$$\mathbb{E}_{\mathbf{Z}}[\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\mu}, \boldsymbol{\pi})] = \sum_{n=1}^N \sum_{k=1}^K \gamma(z_{nk}) \left\{ \ln \pi_k + \sum_{i=1}^D [x_{ni} \ln \mu_{ki} + (1 - x_{ni}) \ln(1 - \mu_{ki})] \right\} \quad (9.55)$$

where

$$\gamma(z_{nk}) = \mathbb{E}[z_{nk}] = \frac{\pi_k p(\mathbf{x}_n|\boldsymbol{\mu}_k)}{\sum_{j=1}^K \pi_j p(\mathbf{x}_n|\boldsymbol{\mu}_j)}. \quad (9.56)$$

Similar calculation as with mixture of Gaussian

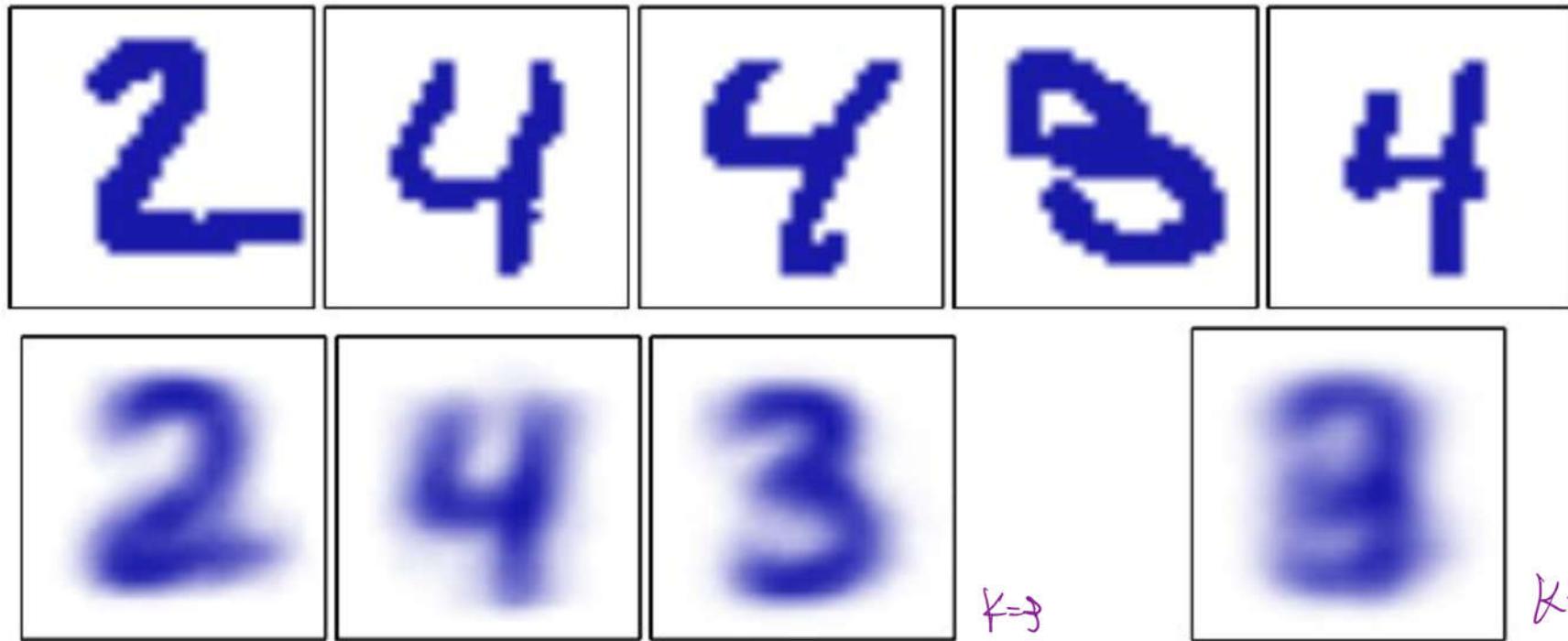
$$\gamma(z_{nk}) = \frac{\pi_k p(\mathbf{x}_n | \boldsymbol{\mu}_k)}{\sum_{j=1}^K \pi_j p(\mathbf{x}_n | \boldsymbol{\mu}_j)}$$

$$N_k = \sum_{n=1}^N \gamma(z_{nk})$$

$$\bar{\mathbf{x}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n \quad \mu_k = \bar{\mathbf{x}}$$

$$\pi_k = \frac{N_k}{N}$$

$16 \times 16$   
 $D=256$



**Figure 9.10** Illustration of the Bernoulli mixture model in which the top row shows examples from the digits data set after converting the pixel values from grey scale to binary using a threshold of 0.5. On the bottom row the first three images show the parameters  $\mu_{ki}$  for each of the **three components** in the mixture model. As a comparison, we also fit the same data set using a single multivariate Bernoulli distribution, again using maximum likelihood. This amounts to simply averaging the counts in each pixel and is shown by the right-most image on the bottom row.

# Outline

## EM revisited

- Connections between GMM and K-means
- Bernoulli mixture

EM in general - does it really maximise likelihood, and why?

## Practical considerations and other topics

- impossibility of clustering [Kleinberg 2003]
- Kmeans++ [Vassilvitskii and Arthur, 2006]

## KL divergence (reminder)

$$\begin{aligned} \text{KL}(p\|q) &= - \int p(\mathbf{x}) \ln q(\mathbf{x}) d\mathbf{x} - \left( - \int p(\mathbf{x}) \ln p(\mathbf{x}) d\mathbf{x} \right) \\ &= - \int p(\mathbf{x}) \ln \left\{ \frac{q(\mathbf{x})}{p(\mathbf{x})} \right\} d\mathbf{x}. \end{aligned} \quad (1.113)$$

If we have “incorrectly” represented  $p(x)$  with  $q(x)$ , how much more *information* do we need to recover  $p(x)$ ?

Apply Jensen’s inequality,  $-\ln()$  is convex

$$f \left( \int \mathbf{x} p(\mathbf{x}) d\mathbf{x} \right) \leq \int f(\mathbf{x}) p(\mathbf{x}) d\mathbf{x}. \quad (1.117)$$

$$\text{KL}(p\|q) = - \int p(\mathbf{x}) \ln \left\{ \frac{q(\mathbf{x})}{p(\mathbf{x})} \right\} d\mathbf{x} \geq - \ln \int q(\mathbf{x}) d\mathbf{x} = 0 \quad (1.118)$$

Equality will only hold iff.  $p(x) = q(x)$  for all  $x$

# The EM algorithm in general

Goal: show that the EM algorithm maximises the likelihood function.

$$p(\mathbf{X}|\boldsymbol{\theta}) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}). \quad (9.69)$$

*lower bound*  $\downarrow$   $\nearrow \geq 0$

$$\ln p(\mathbf{X}|\boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + \text{KL}(q||p) \quad (9.70)$$

where we have defined

$$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})}{q(\mathbf{Z})} \right\} \quad (9.71)$$

$$\text{KL}(q||p) = - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}. \quad (9.72)$$

Derive (9.70)

Likelihood

$$P(X|\theta) = \sum_z P(X, Z|\theta)$$

complete data likelihood

sum rule

$$\ln P(X|\theta) + \ln P(Z|X, \theta) = \ln P(X, Z|\theta)$$

$$\ln P(X|\theta) = \ln P(X, Z|\theta) - \ln P(Z|X, \theta)$$

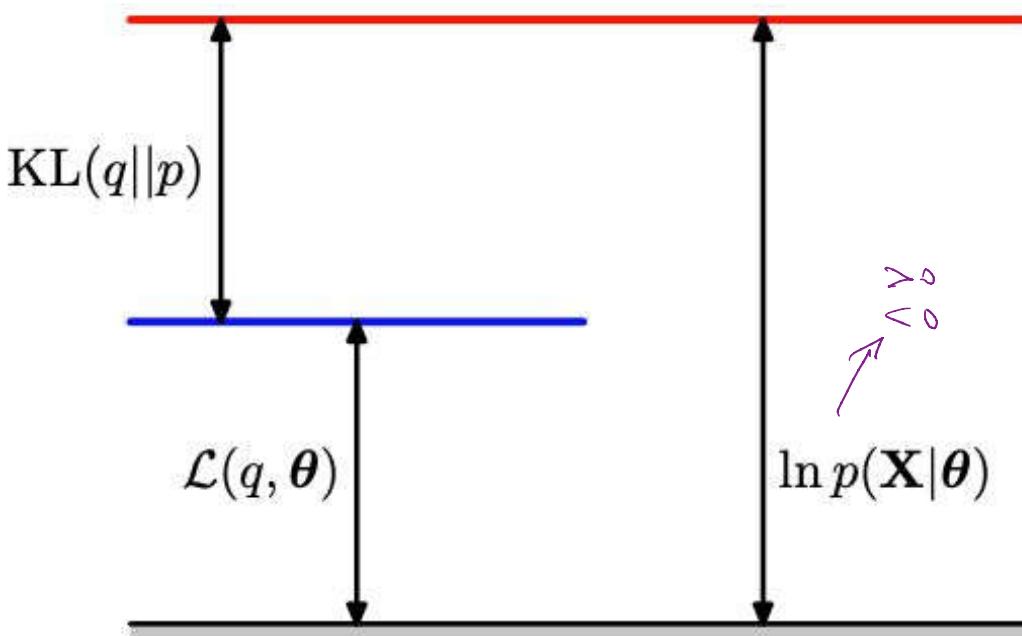
introduce  $q(z)$

$$\begin{aligned} \sum_z q(z) &= 1 \\ &= \sum_z q(z) \ln P(X|\theta) \\ &= \sum_z q(z) [\ln P(X, Z|\theta) - \ln P(Z|X, \theta) + \ln q(z) - \ln q(z)] \\ &= \sum_z q(z) [\ln P(X, Z|\theta) - \ln q(z)] - \sum_z q(z) \underbrace{\ln \frac{P(Z|X, \theta)}{q(z)}}_{\text{KL}(q(z) \parallel P(Z|X, \theta))} \end{aligned}$$

## Illustrating the decomposition

$$\ln p(\mathbf{X}|\boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + \text{KL}(q\|p) \quad (9.70)$$

**Figure 9.11** Illustration of the decomposition given by (9.70), which holds for any choice of distribution  $q(\mathbf{Z})$ . Because the Kullback-Leibler divergence satisfies  $\text{KL}(q\|p) \geq 0$ , we see that the quantity  $\mathcal{L}(q, \boldsymbol{\theta})$  is a lower bound on the log likelihood function  $\ln p(\mathbf{X}|\boldsymbol{\theta})$ .

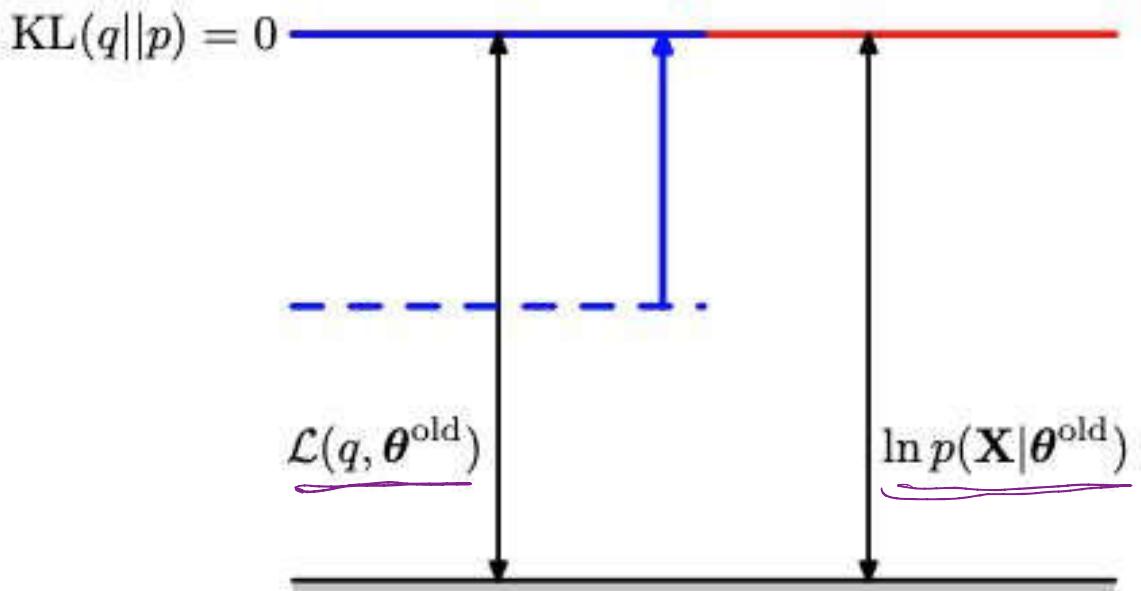


## E step

set  $q(Z) = p(Z | X, \theta)$

$$\text{KL}(q||p) = - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X}, \theta)}{q(\mathbf{Z})} \right\}. \quad (9.72)$$

**Figure 9.12** Illustration of the E step of the EM algorithm. The  $q$  distribution is set equal to the posterior distribution for the current parameter values  $\theta^{\text{old}}$ , causing the lower bound to move up to the same value as the log likelihood function, with the KL divergence vanishing.



3. **M step** Evaluate  $\theta^{\text{new}}$  given by

$$\theta^{\text{new}} = \arg \max_{\theta} Q(\theta, \theta^{\text{old}}) \quad (9.32)$$

where

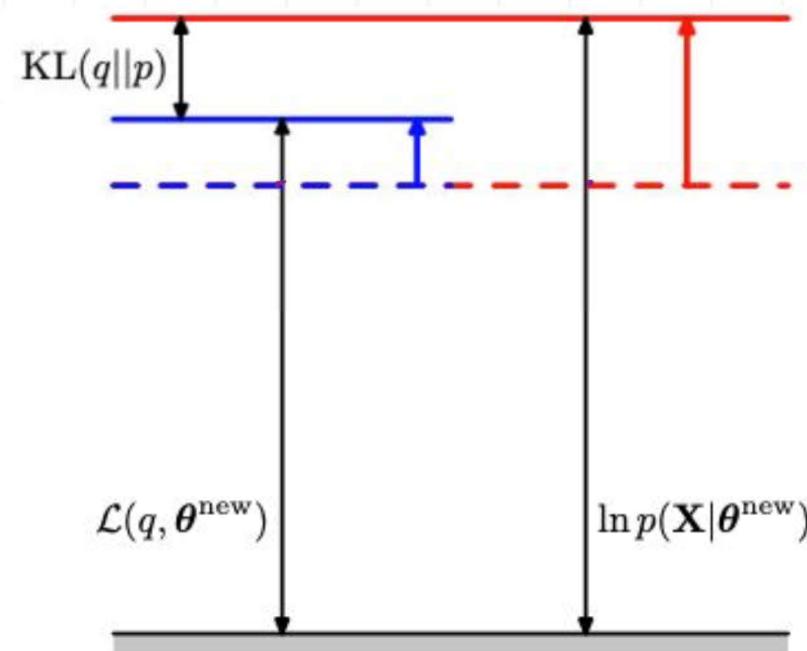
$$Q(\theta, \theta^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\theta). \quad (9.33)$$

$$\mathcal{L}(q, \theta) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z}|\theta)}{q(\mathbf{Z})} \right\} \quad (9.71)$$

*updated.*  
 *$q(\mathbf{Z}) \rightarrow \text{fixed}$ .*

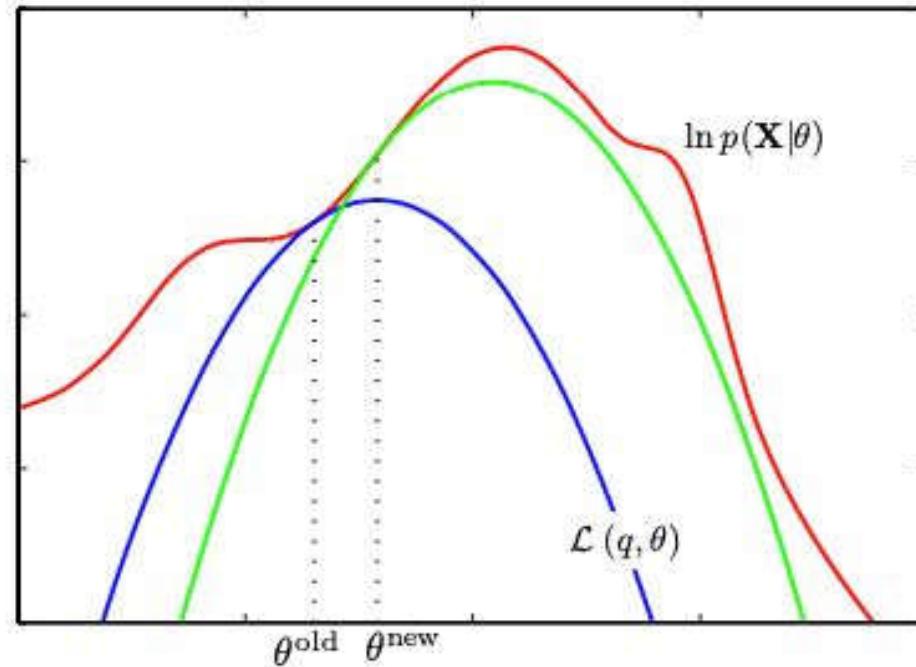
$p(\mathbf{Z}|\mathbf{X}, \theta^{\text{old}})$

**Figure 9.13** Illustration of the M step of the EM algorithm. The distribution  $q(\mathbf{Z})$  is held fixed and the lower bound  $\mathcal{L}(q, \theta)$  is maximized with respect to the parameter vector  $\theta$  to give a revised value  $\theta^{\text{new}}$ . Because the KL divergence is nonnegative, this causes the log likelihood  $\ln p(\mathbf{X}|\theta)$  to increase by at least as much as the lower bound does.



# EM as alternating maximization

**Figure 9.14** The EM algorithm involves alternately computing a lower bound on the log likelihood for the current parameter values and then maximizing this bound to obtain the new parameter values. See the text for a full discussion.



Lower bound  $L(q, \theta)$  is a convex function having a unique maximum (for mixture components from the exponential family).

Extensions: Generalised EM seeks to improve rather than maximise  $L(q, \theta)$ ; expectation conditional maximisation seeks to maximise  $L(q, \theta)$  for a subset of the parameters; Incremental algorithms also exist.

# Outline

## EM revisited

- Connections between GMM and K-means
- Bernoulli mixture

EM in general - does it really maximise likelihood, and why?

## Practical considerations and other topics

- impossibility of clustering [Kleinberg 2003]
- Kmeans++ [Vassilvitskii and Arthur, 2006]

# An Impossibility Theorem for Clustering

[NeuRIPS 2003]



Jon Kleinberg

Professor of Computer Science, Cornell University  
Verified email at cs.cornell.edu - [Homepage](#)

algorithms data mining information networks

**Theorem 2.1** For each  $n \geq 2$ , there is no clustering function  $f$  that satisfies Scale-Invariance, Richness, and Consistency.

Single-linkage operates by initializing each point as its own cluster, and then repeatedly merging the pair of clusters whose distance to one another (as measured from their closest points of approach) is minimum.  
*Stopping conditions:*

K-clusters	Distance- <u>r</u>	Scale- $\alpha$	K-means
✓		✓	✓
	✓	✓	✓
✓	✓	✓	✗

SCALE-INVARIANCE. For any distance function  $d$  and any  $\alpha > 0$ , we have  $f(d) = f(\alpha \cdot d)$ .

RICHNESS. Range( $f$ ) is equal to the set of all partitions of  $S$ .

CONSISTENCY. Let  $d$  and  $d'$  be two distance functions. If  $f(d) = \Gamma$ , and  $d'$  is a  $\Gamma$ -transformation of  $d$ , then  $f(d') = \Gamma$ .

*shrink d' not bet. points in the same cluster*

# K-means ++

Vassilvitskii, Sergei, and David Arthur. "k-means++: The advantages of careful seeding." In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027-1035. 2006.



**Sergei Vassilvitskii**

I am a Research Scientist at [Google](#) New York. Previously I was a Research Scientist at [Yahoo! Research](#) and an Adjunct Assistant Professor at [Columbia University](#). I completed my PhD at [Stanford University](#) under the supervision of [Rajeev Motwani](#). Prior to that I was an undergraduate at [Cornell University](#).

[sergei at cs.stanford.edu](mailto:sergei@cs.stanford.edu)

Problem with K-means

Finds a local optimum that *could be arbitrarily worse than the global optimum*.

Algorithm summary:

1. Choose one center uniformly at random among the data points.
2. For each data point  $x$  not chosen yet, compute  $D(x)$ , the distance between  $x$  and the nearest center that has already been chosen.
3. Choose one new data point at random as a new center, using a weighted probability distribution where a point  $x$  is chosen with probability proportional to  $D(x)^2$ .
4. Repeat Steps 2 and 3 until  $k$  centers have been chosen.
5. Now that the initial centers have been chosen, proceed using standard [k-means clustering](#).

**Theorem:** k-means++ is  $\Theta(\log k)$  approximate in expectation.

**What do you do in practice?** Normalise data. Use Kmeans++ to initialise Kmeans (`sklearn.cluster.KMeans_plusplus`). Use Kmeans (and spherical covariances) to initialise GMM. Try a number of different initialisations `sklearn.cluster.KMeans(n_clusters=8, *, init='k-means++', n_init=10, ...)`

# Summary for today

## EM revisited

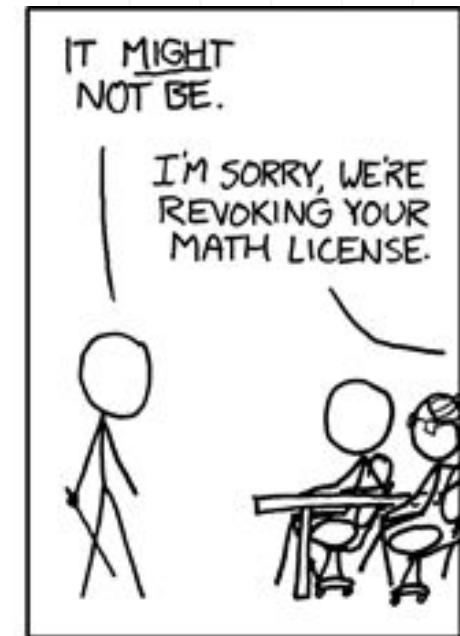
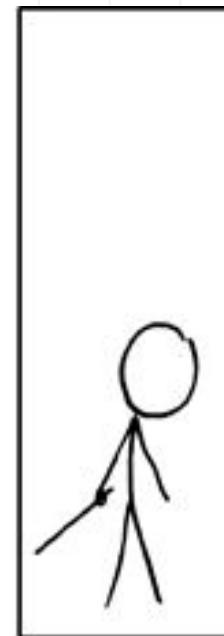
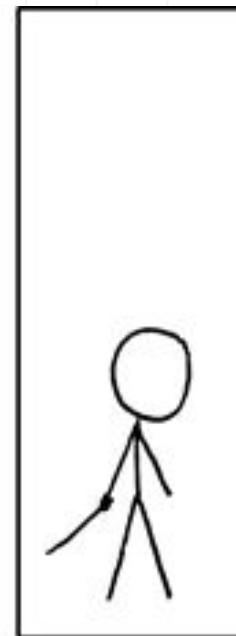
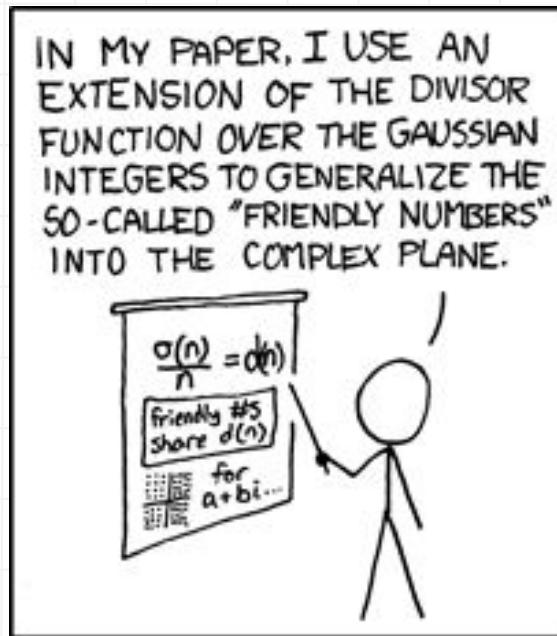
- Connections between GMM and K-means
- Bernoulli mixture

EM in general - does it really maximise likelihood, and why?

## Practical considerations and other topics

- impossibility of clustering [Kleinberg 2003]
- Kmeans++ [Vassilvitskii and Arthur, 2006]

<https://xkcd.com/410/>



## announcements

Quiz 1 – open until Fri 10am

Assignment 1 – due in < 2 weeks (Mon noon week 6)

# (a bite-sized intro to) Generalisation

The MLStory book

<https://mlstory.org/generalization.html>

Chapter 5

High level questions for today:

Why learning works? Why over-parameterisation works?

The notion of generalization gap

Overparameterization: empirical phenomena

Prelude: three inequalities

Theories of generalization

- Algorithmic stability
- Model complexity and uniform convergence
- Generalization from algorithms



Moritz Hardt

Director  
Social Foundations of Computation  
Max Planck Institute for Intelligent Systems, Tübingen  
  
Associate Professor, on leave  
Electrical Engineering and Computer Sciences  
University of California, Berkeley



Benjamin Recht  
Professor, UC  
Berkeley



Peter Bartlett



Bob Williamson

# Notations - Loss function and risks

$$R[f] = \mathbb{E} [\text{loss}(f(X), Y)]$$

predictor  
↓  
Risk ↑  
E - over true data distribution (X, Y)  
– we don't have access to this

target ↑

Stretched notation, loss on one data point (x,y)

$$\text{loss}(f, (x, y))$$

$$\text{loss}(w, (x, y))$$

$$S = ((x_1, y_1), \dots, (x_n, y_n)) \in (\mathcal{X} \times \mathcal{Y})^n.$$

A sample as ordered tuples

$$R_S[f] = \frac{1}{n} \sum_{i=1}^n \text{loss}(f(x_i), y_i).$$

Empirical risk for this sample

## Empirical risk minimisation (ERM)

seeks to find a predictor  $f^*$  in a specified class  $\mathcal{F}$  that minimizes the empirical risk

$$R_S[f^*] = \min_{f \in \mathcal{F}} R_S[f]$$

min. training error, training loss

Ideally

$$R_S[f] \approx R[f].$$

↑  
loss on seen  
examples

↑  
loss on unseen  
(and seen)  
examples

We expect this to be worse (larger loss/risk)

# Generalisation gap

**Definition 1.** Define the generalization gap of a predictor  $f$  with respect to a dataset  $S$  as

$$\Delta_{\text{gen}}(f) = R[f] - R_S[f].$$

Aka generalisation error, or excess risk

$$R[f] = R_S[f] + \Delta_{\text{gen}}(f)$$

“If we manage to make the empirical risk small through optimization (most of this class and other ML classes), then all that remains to worry about is generalization gap.”

But how?

# Evidence from ML practice: overparameterization

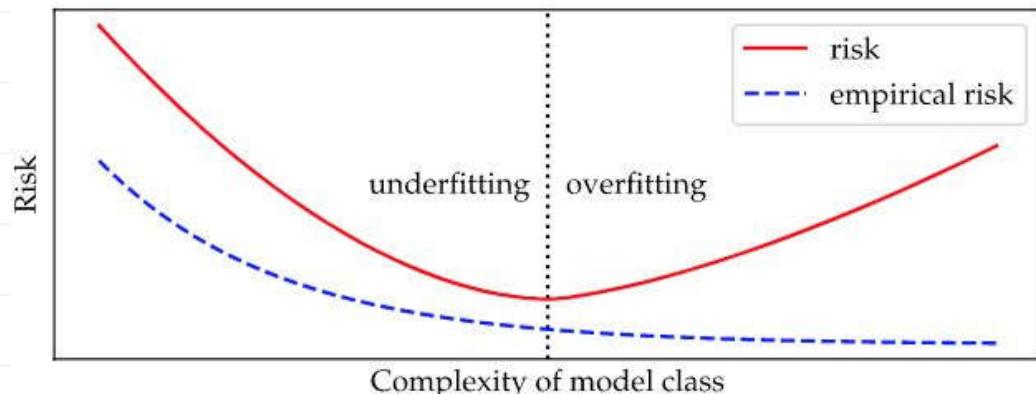
Model size / complexity (informally): number of trainable parameters, for a given model family.

old theory: over-parameterisation is bad

$$\ln p(\mathcal{D}|\mathbf{w}_{\text{ML}}) - M \quad (1.73)$$

$$\ln p(\mathcal{D}) \simeq \ln p(\mathcal{D}|\boldsymbol{\theta}_{\text{MAP}}) - \frac{1}{2}M \ln N \quad (4.139)$$

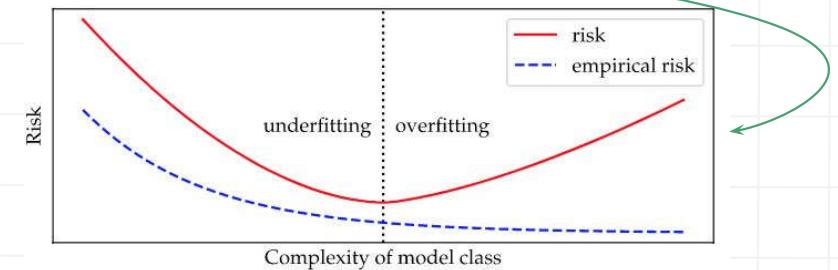
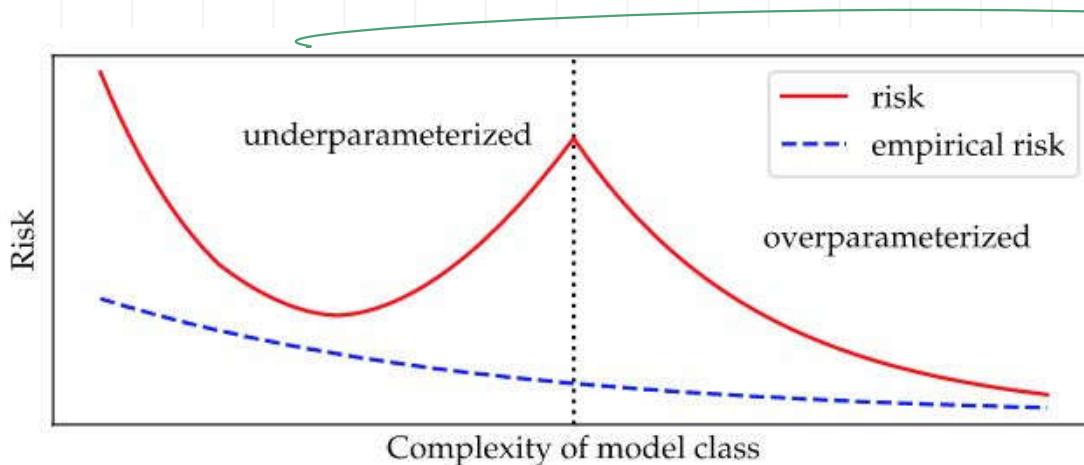
Traditional view of generalization



M - number of parameters; N - number of points

# Evidence from ML practice: double descent

- Complex models also can simultaneously achieve close to zero training loss and still generalize well
- Risk continues to decreases as model complexity grows and training data are interpolated exactly down to (nearly) zero training loss=
- Empirical relationship between overparameterization and risk appears to be robust and manifests in numerous model classes, including overparameterized linear models, ensemble methods, and neural networks.
- Increasing model complexity in the overparameterized regime continues to decrease risk indefinitely, albeit at decreasing marginal returns, toward some convergence point.

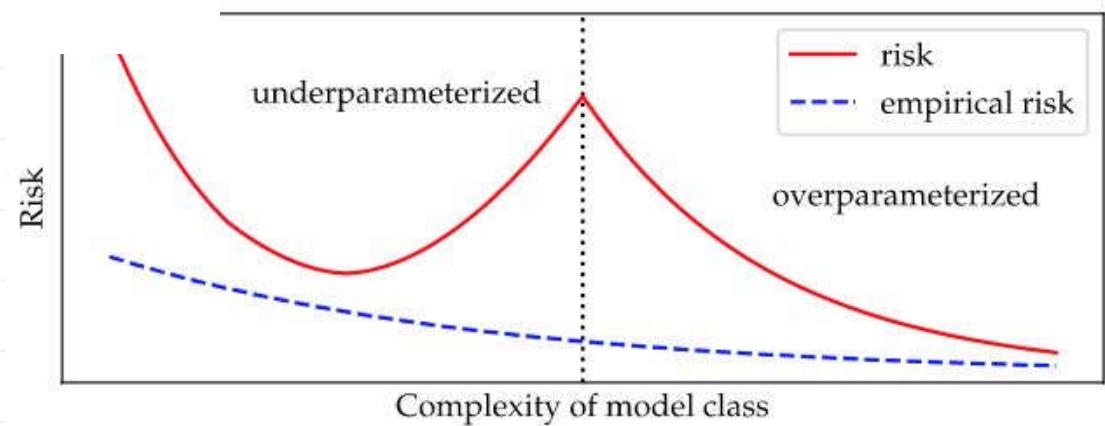
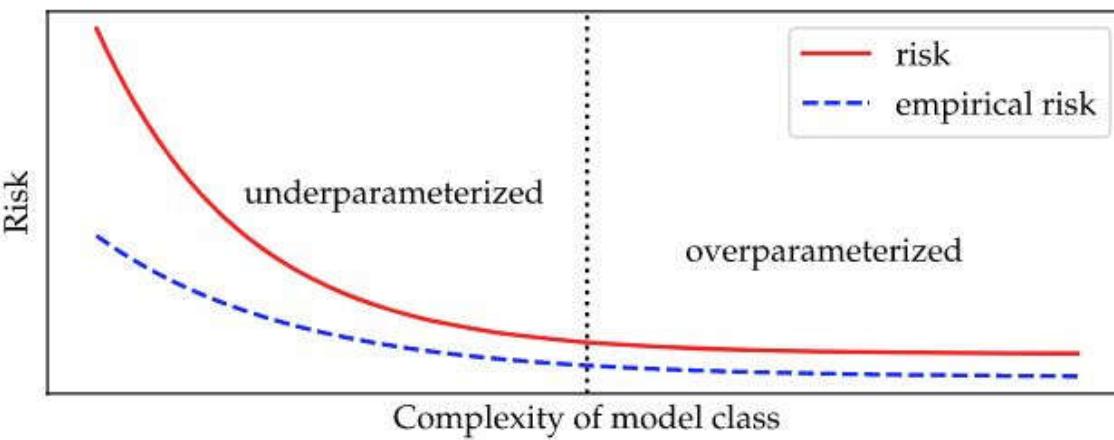


Loog, M., Viering, T.J., Mey, A., Krijthe, J.H., & Tax, D.M. (2020). A brief prehistory of double descent. *Proceedings of the National Academy of Sciences*, 117, 10625 - 10626.

Dar, Y., Muthukumar, V., & Baraniuk, R. (2021). A Farewell to the Bias-Variance Tradeoff? An Overview of the Theory of Overparameterized Machine Learning. *ArXiv*, *abs/2109.02355*.

# Single descent: larger models work better ...

e.g. ResNet [He et al 2016] in computer vision



\*Sometimes we see multiple bumps too ...

# Optimisation versus generalisation

$$R[f] = R_S[f] + \Delta_{\text{gen}}(f)$$

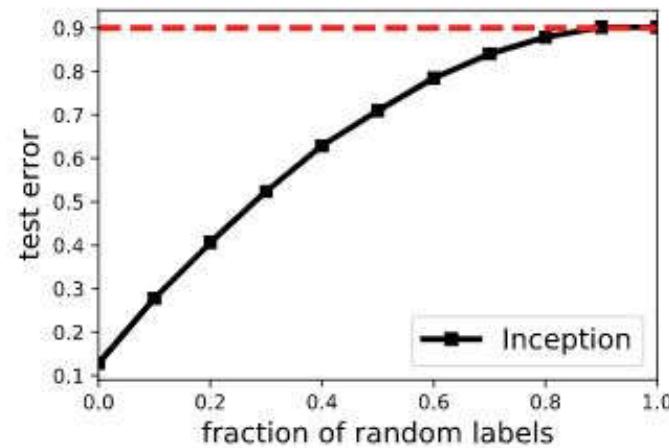
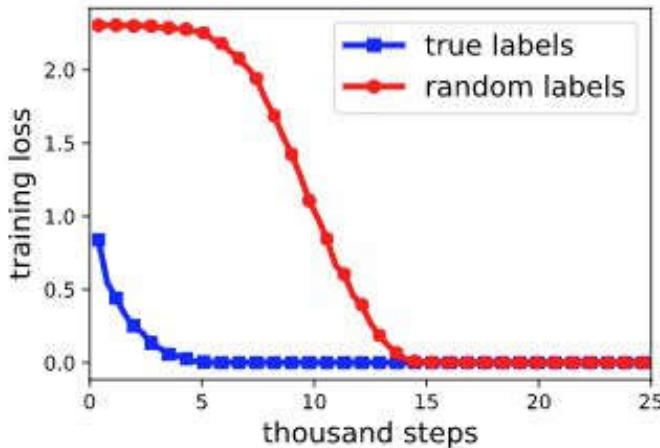
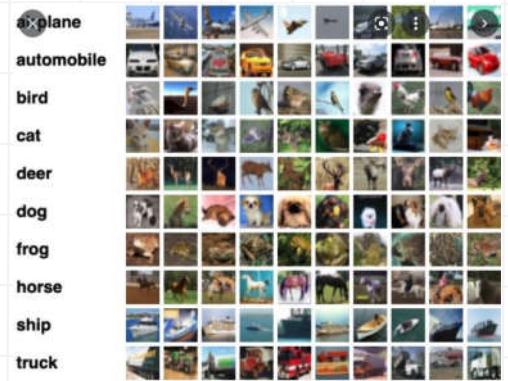
Training/optimisation is “easy” i.e. when # parameters > data points

BUT, overparameterization puts burden on generalisation.

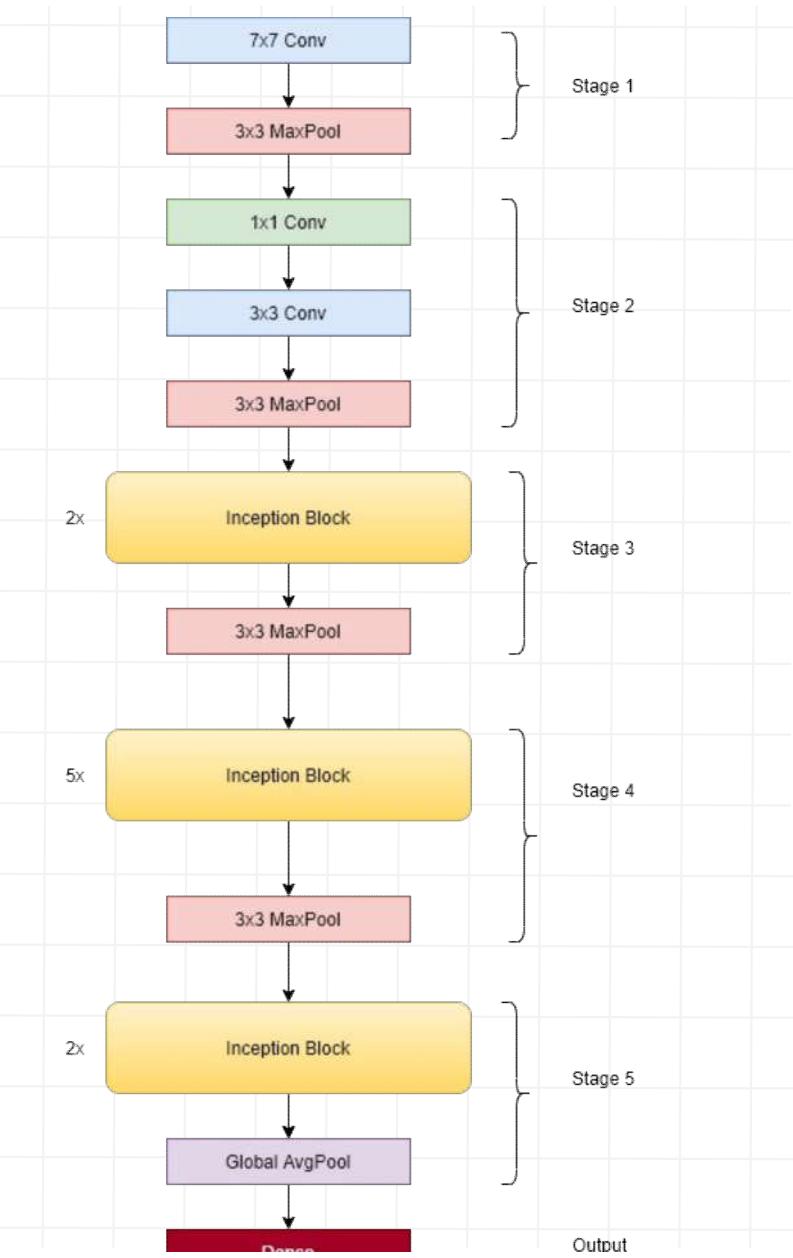
Experiment: Training with **random (!)** labels on CIFAR-10 (10 classes)

**R[f] is known ... test accuracy should be 1/10**

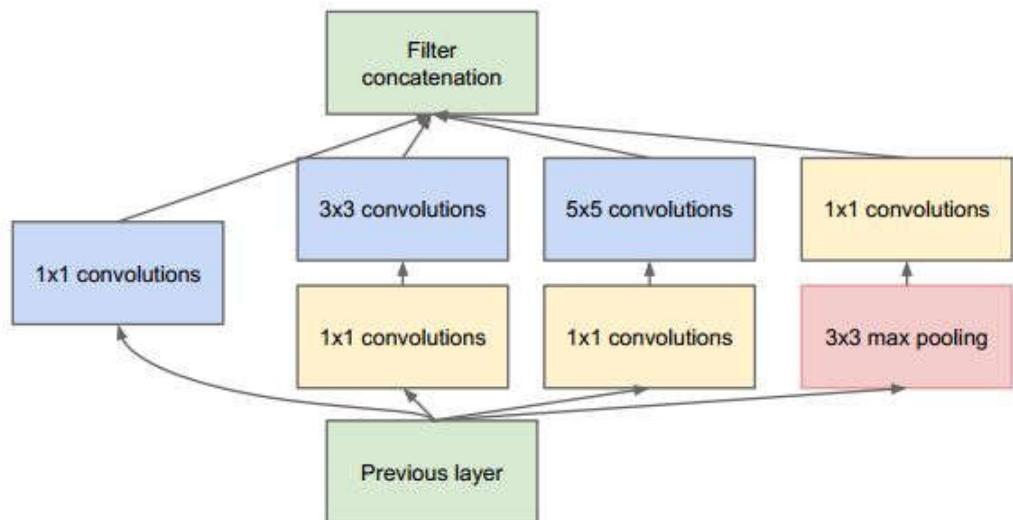
Training error is driven to zero by the optimisation algorithm → overfits  
(similar observations hold for many overparameterized architectures in literature)



→ proof of convergence in optimisation may not reveal insights into the nature of generalisation.



# Inception illustrated



# What about regularization?

- L2 regularisation (regression, large neural nets)
- Data augmentation (e.g. random cropping and rotation of training images)

Experiment on CIFAR-10 (50K training examples) + Inception (1.5M param)

The training and test accuracy (in percentage) with and without data augmentation and  $\ell_2$ -regularization.

params	random crop	$\ell_2$ -regularization	train accuracy	test accuracy
1,649,402	yes	yes	100.0	89.05
	yes	no	100.0	89.31
	no	yes	100.0	86.03
	no	no	100.0	85.75

→ yes regularizations help, but is by no means necessary for strong generalisation.

# Why learning works?

?  $\Delta_{\text{gen}}$

Four views presented in this chapter

- *Algorithmic stability*: generalization arises when *models are insensitive to perturbations in the data* on which they are trained.
- *VC dimension and Rademacher complexity*: how small generalization gaps can arise when we *restrict the complexity of models* we wish to fit to data.
- *Margin bounds*: whenever the *data is easily separable*, good generalization will occur.
- *Optimization*: how *choice of an algorithmic scheme* itself can yield models with desired generalization properties

The four different views of generalization can all arrive at similar results – the UPPER bound on  $\Delta_{\text{gen}}(f)$  depends on  $n$  (decreases as  $n$  increase) and the complexity of the ideal predictor (increases as complexity increase)

Generalization is multifaceted and multiple perspectives are useful when designing data-driven predictive systems.

# How do we expect the gap to scale?

$$R[f] = R_S[f] + \Delta_{\text{gen}}(f)$$

For a fixed prediction function  $f$ , with infinite amount of data.

$E_S[\text{empirical risk}] = \text{population risk } R[f]$ .

Recall CLT (Central Limit Theorem)

If  $Z$  is a random variable with bounded variance then, then its sample mean converges in distribution to a Gaussian random variable with mean zero and variance on the order of  $1/n$ .

Goal: Upper bound on  $\Delta_{\text{gen}}(f)$ , we want it to be small with high probability

$$P[|R[f] - R_S[f]| \geq \epsilon] \leq \delta \quad \text{OR} \quad P[|R[f] - R_S[f]| \leq \epsilon] \geq 1 - \delta$$

$\Delta_{\text{gen}}$

$(\epsilon, \delta)$

$$\frac{1}{\sqrt{n}} \quad \frac{1}{\log n}$$

How fast does  $\Delta_{\text{gen}}(f)$  shrink w.r.t. number of data points  $n$ ?

E.g.  $k^n, n^k, O(n), \log(n)$

$$\Delta_{\text{gen}} \sim O\left(\frac{1}{n}\right) \quad \frac{1}{n^k}$$

$$\frac{1}{K^n}$$

# (a bite-sized intro to) Generalisation

The notion of generalization gap

Overparameterization: empirical phenomena

Prelude: three inequalities – that we'll need later

Theories of generalization

- Algorithmic stability
- Model complexity and uniform convergence
- Margin bounds
- Generalization from algorithms

# Concentration inequalities

- Markov's inequality: Let  $Z$  be a nonnegative random variable. Then,

$$\mathbb{P}[Z \geq t] \leq \frac{\mathbb{E}[Z]}{t}.$$

Let's say that  $X$  can take values  $x_1 < x_2 < \dots < x_j = t < \dots < x_n$ .

[thanks: UW CS312]

$$\mathbb{E}[X] = \sum_{i=1}^n x_i * \Pr[X = x_i] \geq \sum_{i=j}^n x_i * \Pr[X = x_i] \geq \sum_{i=j}^n t * \Pr[X = x_i]$$

Second form    let  $t = s\mathbb{E}[X]$  ,  $s > 0$      $\longrightarrow$      $\Pr[X \geq s \cdot \mathbb{E}[X]] \leq \frac{1}{s}$ .

*Proof of Markov's Inequality.* Below is the proof when  $X$  is continuous. The proof for discrete RVs is similar (just change all the integrals into summations).

$$\begin{aligned}
 \mathbb{E}[X] &= \int_0^\infty xf_X(x)dx && [\text{because } X \geq 0] \\
 &= \int_0^k xf_X(x)dx + \int_k^\infty xf_X(x)dx && [\text{split integral at some } 0 \leq k \leq \infty] \\
 &\geq \int_k^\infty xf_X(x)dx && \left[ \int_0^k xf_X(x)dx \geq 0 \text{ because } k \geq 0, x \geq 0 \text{ and } f_X(x) \geq 0 \right] \\
 &\geq \int_k^\infty kf_X(x)dx && [\text{because } x \geq k \text{ in the integral}] \\
 &= k \int_k^\infty f_X(x)dx \\
 &= k\mathbb{P}(X \geq k)
 \end{aligned}$$

## Example: a weighted coin

$$\frac{E(X)}{t}$$

A coin is weighted so that its probability of landing on heads is 20%, independently of other flips.

Suppose the coin is flipped 20 times.

Use Markov's inequality to bound the probability it lands on heads at least 16 times.

Also use Chebyshev's inequality to upper bound the same probability.

$X \sim \text{Bin}(n = 20, p = 0.2)$ :

$$E[X] = np = 20 \cdot 0.2 = 4$$

$$\mathbb{P}(X \geq 16) \leq \frac{E[X]}{16} = \frac{4}{16} = \frac{1}{4}$$

Let's compare this to the actual probability that this happens:

$$\mathbb{P}(X \geq 16) = \sum_{k=16}^{20} \binom{20}{k} 0.2^k \cdot 0.8^{20-k} \approx 1.38 \cdot 10^{-8}$$

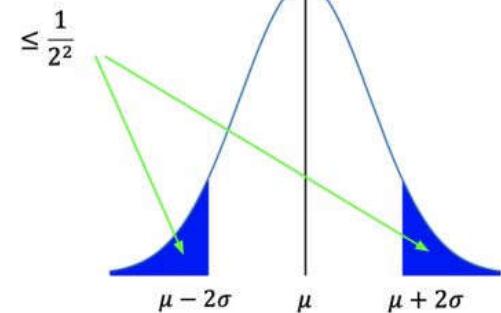
This is not a good bound, since we only assume to know the expected value. Again, we knew the exact distribution, but chose not to use any of that information (the variance, the PMF, etc.).  $\square$

[thanks: UW CS312]

- Chebyshev’s inequality: Suppose  $Z$  is a random variable with mean  $\mu_Z$  and variance  $\sigma_Z^2$ . Then,

$$\mathbb{P}[Z \geq t + \mu_Z] \leq \frac{\sigma_Z^2}{t^2}$$

$$P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$$



Explains why sample averages are good estimates of the mean.

$$\mathbb{P}[\hat{\mu} \geq t + \mu_X] \leq \frac{\sigma_X^2}{nt^2}, \quad \mathbb{P}[\hat{\mu} \geq 2\mu_X] \leq \frac{\sigma_X^2}{n\mu_X^2}.$$

[thanks: UW CS312]

- Chebyshev’s inequality: Suppose  $Z$  is a random variable with mean  $\mu_Z$  and variance  $\sigma_Z^2$ . Then,

$$\mathbb{P}[Z \geq t + \mu_Z] \leq \frac{\sigma_Z^2}{t^2}$$

Proof sketch

$$\begin{aligned}
 \mathbb{P}(|X - \mathbb{E}[X]| \geq \alpha) &= \mathbb{P}\left((X - \mathbb{E}[X])^2 \geq \alpha^2\right) && [\text{square both sides}] \\
 &\leq \frac{\mathbb{E}[(X - \mathbb{E}[X])^2]}{\alpha^2} && [\text{Markov's inequality}] \\
 &= \frac{\text{Var}(X)}{\alpha^2} && [\text{def of variance}]
 \end{aligned}$$

$$\boxed{\mathbb{P}[Z \geq t] \leq \frac{\mathbb{E}[Z]}{t}}$$

[thanks: UW CS312]

## Example: a weighted coin (continued)

A coin is weighted so that its probability of landing on heads is 20%, independently of other flips. Suppose the coin is flipped 20 times.

Use Markov's inequality to bound the probability it lands on heads at least 16 times.  
Also use Chebyshev's inequality to upper bound the same probability.

$$X \sim \text{Bin}(n = 20, p = 0.2)$$

$$\mathbb{E}[X] = np = 20 \cdot 0.2 = 4$$

$$\text{Var}(X) = np(1 - p) = 20 \cdot 0.2 \cdot (1 - 0.2) = 3.2$$

Chebyshev's inequality is symmetric about the mean (difference of 12;  $4 \pm 12$  gives the interval  $[-8, 16]$ ):

$$\begin{aligned} \mathbb{P}(X \geq 16) &\leq \mathbb{P}(X \geq 16 \cup X \leq -8) && [\text{adding another event can only increase probability}] \\ &= \mathbb{P}(|X - 4| \geq 12) && [\text{def of abs value}] \\ &= \mathbb{P}(|X - \mathbb{E}[X]| \geq 12) && [\mathbb{E}[X] = 4] \\ &\leq \frac{\text{Var}(X)}{12^2} && [\text{Chebyshev's inequality}] \\ &= \frac{3.2}{12^2} = \frac{1}{45} \end{aligned}$$

[thanks: UW CS312]

- Hoeffding’s inequality: Let  $Z_1, Z_2, \dots, Z_n$  be independent random variables, each taking values in the interval  $[a_i, b_i]$ . Let  $\hat{\mu}$  denote the sample mean  $\frac{1}{n} \sum_{i=1}^n Z_i$ . Then

$$\mathbb{P}[\hat{\mu} \geq \mu_Z + t] \leq \exp\left(-\frac{2n^2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

An important special case is when the  $Z_i$  are identically distributed copies of  $Z$  and take values in  $[0, 1]$ . Then we have

$$\mathbb{P}[\hat{\mu} \geq \mu_Z + t] \leq \exp(-2nt^2).$$

when random variables are bounded, sample averages concentrate around their mean value exponentially quickly.

With probability at least  $1-\delta$ ,

$$\hat{\mu} - \mu_Z \leq \epsilon$$

two-sided:

$$\mathbb{P}[|\hat{\mu} - \mu_z| \leq \epsilon] \geq 1 - 2 \exp(-2n\epsilon^2)$$

*Use this!*

one-sided:

$$\mathbb{P}[\hat{\mu} - \mu_z \leq \epsilon] \geq 1 - \exp(-2n\epsilon^2)$$

# Example application of a concentration inequality

A person's height  $(0, 9]$  (unit: feet, 1 feet = 12 inches  $\sim 30$  cm)

Sample 30,000 individuals (randomly!)  $\{h_1, \dots, h_{30,000}\}$

Hoeffding's inequality →

With 83% probability, sample mean  $\mu'_h$  is within one inch of true mean  $\mu_h$

When random variables have:

- Low variance or are tightly bounded, small experiments quickly reveal insights about the population.
- Large variances or effectively unbounded, the number of samples required for high precision estimates might be impractical and our estimators and algorithms and predictions may need to be rethought.

## Two scaling regimes

$R_s[f]$  large - generalisation gap decreasing at  $1/\sqrt{n}$

$R_s[f]$  small - generalisation gap decreasing at  $1/n$

Why?

Consider a *single* prediction function  $f$ , chosen *independently* of the sample  $S$

Note: this is not a random predictor

Hoeffding's inequality  $\rightarrow$

With probability  $1-\delta$ ,

$$\mathbb{P}[|\hat{\mu} - \mu_z| \leq \epsilon] \geq 1 - 2 \exp(-2n\epsilon^2)$$

$$\mathbb{P}[R[f] - R_s[f] \geq \epsilon] \leq \exp(-2n\epsilon^2) .$$

$$|\Delta_{gen}(f)| \leq \sqrt{\frac{\log(1/\delta)}{2n}}.$$

$(\log \frac{1}{\delta}) + (\log \frac{1}{\delta})$

$$\frac{\log \frac{1}{\delta} + \log \frac{1}{\delta}}{2n}$$

Looser bound

$$|\Delta_{gen}| \leq \sqrt{\frac{\log(2/\delta)}{2n}}$$

Looser bound  
 $2 \leq 1/\delta$

$$|\Delta_{gen}| \leq \sqrt{\frac{\log(1/\delta)}{n}}$$

## Two scaling regimes

$R_s[f]$  large - generalisation gap decreasing at  $1/\sqrt{n}$

$R_s[f]$  small - generalisation gap decreasing at  $1/n$

In the regime where we observe no empirical mistakes, a more refined analysis can be applied. Suppose that  $R[f] > \epsilon$ . Then the probability that we observe  $R_s[f] = 0$  cannot exceed

$$\begin{aligned}\mathbb{P}[\forall i: \text{sign}(f(x_i)) = y_i] &= \prod_{i=1}^n \mathbb{P}[\text{sign}(f(x_i)) = y_i] \\ &\leq (1 - \epsilon)^n \leq e^{-\epsilon n}.\end{aligned}$$

Hence, with probability  $1 - \delta$ ,

$$|\Delta_{\text{gen}}(f)| \leq \frac{\log(1/\delta)}{n},$$

# (a bite-sized intro to) Generalisation

The notion of generalization gap

Overparameterization: empirical phenomena

Prelude: three inequalities – that we'll need later

Theories of generalization

- Algorithmic stability
- Model complexity and uniform convergence
- Margin bounds
- Generalization from algorithms

# Algorithmic stability

generalization arises when **models are insensitive to perturbations in the data** on which they are trained.

Specifically: how sensitive an algorithm is to changes in **a single training example**.

Three ways of data perturbation, all yielding similar generalisation bounds.

- **Resample a single data point**
- Leave one data point out
- A single data point is arbitrarily corrupted (adversarial scenario)

# Some notations

A sample of n data points  $S = ((x_1, y_1), \dots, (x_n, y_n)) \in (\mathcal{X} \times \mathcal{Y})^n$ .

A labeled example

$$z = (x, y)$$

$$\text{loss}(f, z) = \text{loss}(f(x), y)$$

$Z \sim$  distribution of  $(X, Y)$

n hybrid samples

$$S^{(i)} = (Z_1, \dots, Z_{i-1}, Z'_i, Z_{i+1}, \dots, Z_n)$$

$$S = (Z_1, \dots, Z_n)$$

$$S' = (Z'_1, \dots, Z'_n)$$

A learning algorithm

$$A: (\mathcal{X} \times \mathcal{Y})^n \rightarrow \Omega$$

$$\mathcal{F}$$

Space of data samples

space of function f

A is assumed to minimize  $R_s$  –  
optimisation bounds will be  
mentioned later

# Average stability and its link to $\Delta_{\text{gen}}$

**Definition.** The average stability of an algorithm  $A: (\mathcal{X} \times \mathcal{Y})^n \rightarrow \Omega$  is

$$\Delta(A) = \mathbb{E}_{S,S'} \left[ \frac{1}{n} \sum_{i=1}^n \left( loss(A(S), Z'_i) - loss(A(S^{(i)}), Z'_i) \right) \right].$$

↑  
"test"  
↑  
"train"

**Proposition.** *The expected generalization gap equals average stability:*

$$\mathbb{E}[\Delta_{\text{gen}}(A(S))] = \Delta(A)$$

[proof omitted, see book]

$$A(S) \rightarrow f$$

## Two interpretations

change w.r.t. one example;

change w.r.t seen vs unseen  
examples

# Uniform stability

Average stability

$$\Delta(A) = \mathbb{E}_{S, S'} \left[ \frac{1}{n} \sum_{i=1}^n \left( loss(A(S), Z'_i) - loss(A(S^{(i)}), Z'_i) \right) \right]$$

Definition. *The uniform stability of an algorithm A is defined as*

$$\Delta_{\text{sup}}(A) = \sup_{\substack{S, S' \in (\mathcal{X} \times \mathcal{Y})^n \\ d_H(S, S')=1}} \sup_{z \in \mathcal{X} \times \mathcal{Y}} |loss(A(S), z) - loss(A(S'), z)|,$$

where  $d_H(S, S')$  is the Hamming distance between tuples S and S' .

The two *sups* here – worst case scenario

Why is this called uniform? – will hold for every A, (X, Y)

z has nothing to do with S or S', but is sampled from the same distribution (X, Y)

The worst-case difference in the predictions of the learning algorithm run on two arbitrary datasets that differ in exactly one point.

Uniform stability upper bounds generalization gap (in expectation)  
See book for proof.

$$\mathbb{E}[\Delta_{\text{gen}}(A(S))] = \Delta(A) \leq \Delta_{\text{sup}}(A)$$

# Strongly convex functions

[thanks: EPFL OptML,  
eq numbers therein]

Convex functions: lower-bounded by tangent lines.

$$f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) \leq \lambda f(\mathbf{x}) + (1 - \lambda) f(\mathbf{y}).$$

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x})$$

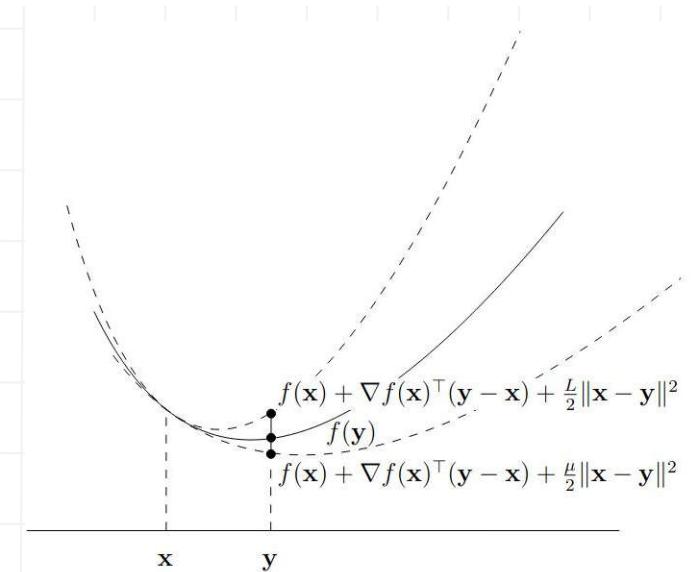
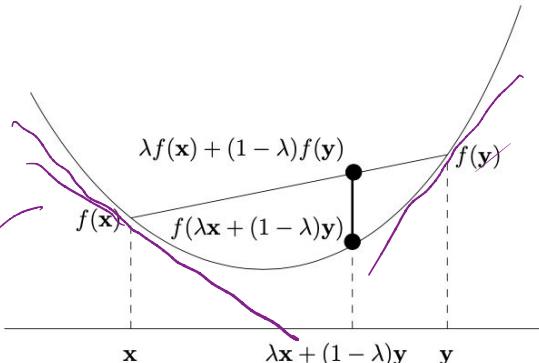


Figure 2.3: A smooth and strongly convex function

Strictly convex functions  $f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) < \lambda f(\mathbf{x}) + (1 - \lambda) f(\mathbf{y}).$  (1.8)

Strongly convex functions,  $\mu > 0$   $f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{\mu}{2} \|\mathbf{x} - \mathbf{y}\|^2, \quad \forall \mathbf{x}, \mathbf{y} \in X.$  (2.19)

Nice properties:

- Lower-bounded by another quadratic function.
- There is a unique global minimum
- It's “fast” to find

$x$        $x^2$        $x^4$   
↑  
convex  
↑ strictly convex.  
not strongly convex

# Strongly convex and L-Lipschitz loss functions

Goal: show that strong convexity of the loss function is sufficient for the uniform stability of empirical risk minimization.

Two assumptions needed:

Loss function differentiable and strongly convex

$$\text{loss}(w', z) \geq \text{loss}(w, z) + \langle \nabla \text{loss}(w, z), w' - w \rangle + \frac{\mu}{2} \|w - w'\|^2$$

If  $\Phi: \mathbb{R}^d \rightarrow \mathbb{R}$  is  $\mu$ -strongly convex and  $w^*$  is a stationary point (and hence global minimum)

$$\Phi(w) - \Phi(w^*) \geq \frac{\mu}{2} \|w - w^*\|^2$$

$\text{loss}(w, z)$  is  $L$ -Lipschitz in  $w$  for every  $z$

$$\|\nabla \text{loss}(w, z)\| \leq L \quad |\text{loss}(w, z) - \text{loss}(w', z)| \leq L \|w - w'\|.$$

# Stability of empirical risk minimisation

See proof in MLstory

**Theorem 1.** Assume that for every  $z$ ,  $\text{loss}(w, z)$  is  $\mu$ -strongly convex in  $w$  over the domain  $\Omega$ , i.e., Further assume that, that the loss function  $\text{loss}(w, z)$  is  $L$ -Lipschitz in  $w$  for every  $z$ . Then, empirical risk minimization (ERM) satisfies

$$\Delta_{\sup}(\text{ERM}) \leq \frac{4L^2}{\mu n}.$$

There is no explicit reference to model class. But what is implied here?

<https://math.stackexchange.com/questions/1106154/any-example-of-strongly-convex-functions-whose-gradients-are-lipschitz-continuous>

What about regularisation?

$$r(w, z) = \text{loss}(w, z) + \frac{\mu}{2} \|w\|^2$$

$L_2$  regularisation turns convex loss into a  $\mu$ -strongly convex one

assume  $\|w\| \leq B$       set  $\mu = \frac{L}{B\sqrt{n}}$        $\frac{\mu}{2} \|w\|^2$  at most  $O\left(\frac{LB}{\sqrt{n}}\right)$

$$\boxed{\Delta_{\sup}(\text{ERM}) \leq \frac{4L^2}{\mu n}}$$

the generalization gap will also be  $O\left(\frac{LB}{\sqrt{n}}\right)$

$$f \in \mathcal{F}$$

$$|\mathcal{F}|$$

# Model complexity and uniform convergence

Uniform convergence: bounding the generalization gap from above for all functions in a function class

Model complexity: counting the number of different functions that can be described with the given model parameters.

Assume loss function bounded in  $[0, 1]$ , apply Hoeffding's

$$\mathbb{P}[R_S[f] > R[f] + t] \leq \exp(-2nt^2)$$

For data-independent prediction function  $f$

$$\text{With probability } 1-\delta, |\Delta_{\text{gen}}(f)| \leq \sqrt{\frac{\log(1/\delta)}{2n}}.$$

With probability  $1 - \delta$ ,  $\forall f \in \mathcal{F}$

$$\Delta_{\text{gen}}(f) \leq \sqrt{\frac{\ln |\mathcal{F}| + \ln(1/\delta)}{n}}. \quad (1)$$

$\ln |\mathcal{F}|$   $\xrightarrow{\text{union}}$

$$P(\exists f \in \mathcal{F}, \Delta_{\text{gen}}(f) \geq \varepsilon) \leq \sum_{f \in \mathcal{F}} P(\Delta_{\text{gen}}(f) \geq \varepsilon)$$

$$P(\Delta_{\text{gen}}(f_i) \geq \varepsilon)$$

The cardinality bound  $|\mathcal{F}|$  is a basic measure of the complexity of the model family  $\mathcal{F}$ .

We can think of the term  $\ln(\mathcal{F})$  as a measure of complexity of the function class  $\mathcal{F}$ .

The gestalt of the generalization bound as " $\sqrt{\text{complexity}/n}$ " routinely appears with varying measures of complexity.

# VC Dimension (Vapnik-Chervonenkis)

Uniform convergence:= bounding the generalization gap from above for all functions in a function class.  
What happens if  $|F|$  infinite?

$\text{VC}(\mathcal{F}) :=$  the size of the largest set  $Q \subseteq X$  such that for any Boolean function  $h: Q \rightarrow \{-1, 1\}$ , there is a predictor  $f \in \mathcal{F}$  such that  $f(x) = h(x)$  for all  $x \in Q$ .

there is a size- $d$  sample  $Q$  such that the functions of  $\mathcal{F}$  induce all  $2^d$  possible binary labelings of  $Q$ , then the VC-dimension of  $\mathcal{F}$  is at least  $d$ .

The VC-dimension measures the ability of the model class to conform to an arbitrary labeling of a set of points.

Example: linear models over  $\mathbb{R}^d$  has a VC dimension of  $d$  – same as number of model parameters.

# VC inequalities

$$\Delta_{\text{gen}}(f) \leq \sqrt{\frac{\text{VC}(\mathcal{F}) \ln n + \ln(1/\delta)}{n}}. \quad (2)$$

Slower rate here!

Consider all hyperplanes in  $\mathbb{R}^d$  with norm at most  $\gamma^1$ , data has bounded norm  $\|x\| \leq D$   
The VC dimension of these hyperplanes is  $D^2/\gamma^2$

$$\Delta_{\text{gen}}(f) \leq \sqrt{\frac{D^2 \ln n + \gamma^2 \ln(1/\delta)}{\gamma^2 n}}.$$

$d$  does not appear, ‘free’ from curse of dimensionality

$$\text{Hoeffding's : } P(|\hat{\mu} - \mu_2| \geq \varepsilon) \leq \exp(-2n\varepsilon^2)$$

## Summary of risk and bounds

*Algorithmic stability:* generalization arises when **models are insensitive to perturbations in the data** on which they are trained.

*VC dimension and Rademacher complexity:* how small generalization gaps can arise when we **restrict the complexity of models** we wish to fit to data.

*Margin bounds:* whenever the **data is easily separable**, good generalization will occur.

*Optimization:* how **choice of an algorithmic scheme** itself can yield models with desired generalization properties

$$\mathbb{E}[\Delta_{\text{gen}}(A(S))] = \Delta(A) \leq \Delta_{\text{sup}}(A)$$

$$\Delta_{\text{sup}}(\text{ERM}) \leq \frac{4L^2}{\mu n}.$$

$$\Delta_{\text{gen}}(f) \leq \sqrt{\frac{\ln |\mathcal{F}| + \ln(1/\delta)}{n}}.$$

$$R[f] - R_S^\theta[f] \leq 4 \frac{\mathfrak{R}(\mathcal{W}_B)}{\theta} + O\left(\frac{\log(1/\delta)}{\sqrt{n}}\right)$$

$$\Delta_{\text{sup}}(\text{SGM}) \leq \frac{2L^2}{n} \sum_{t=1}^T \eta_t.$$

# Summary - (a bite-sized intro to) Generalisation

The notion of generalization gap

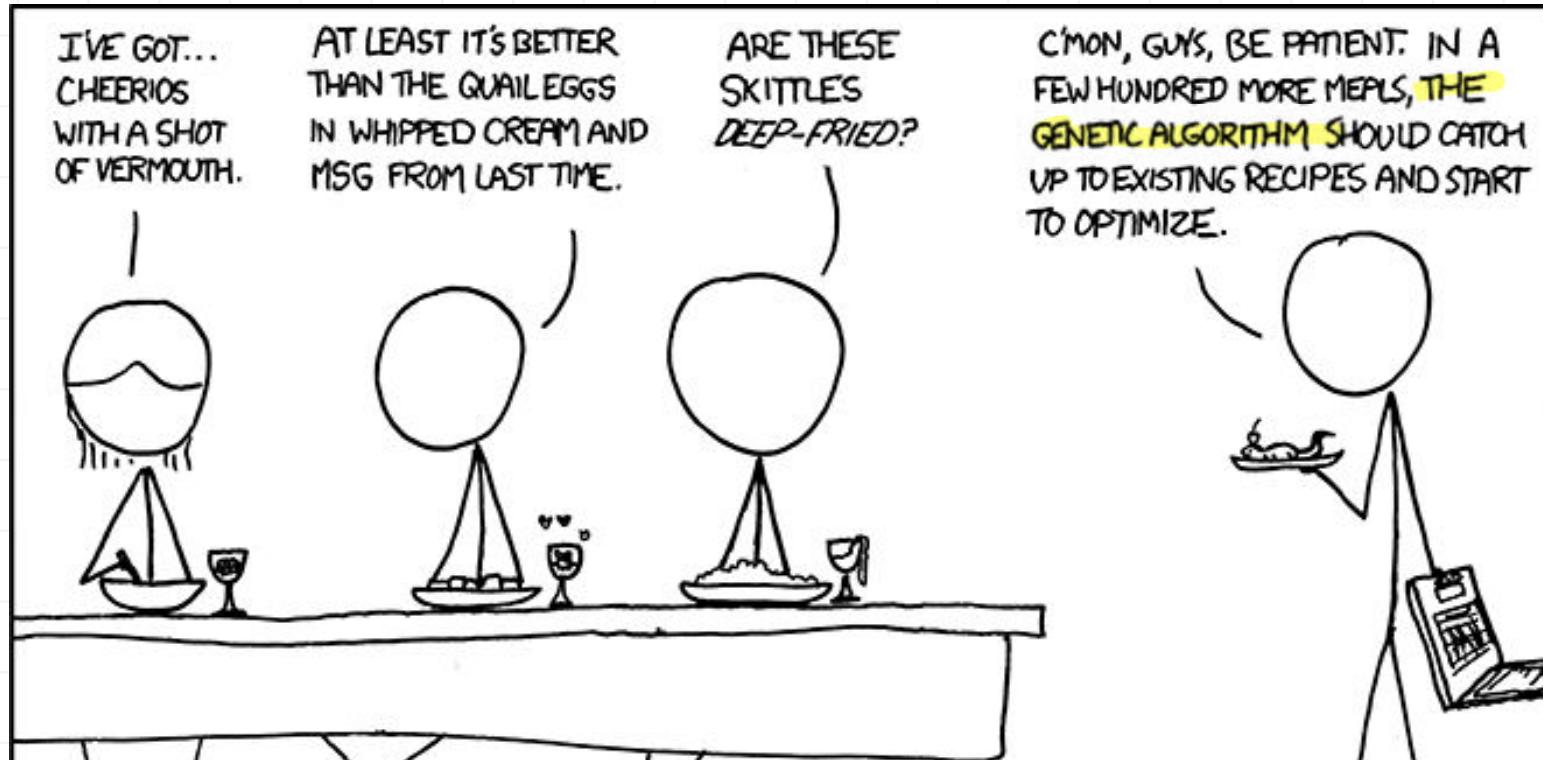
Overparameterization: empirical phenomena

Prelude: three inequalities – that we'll need

Theories of generalization

- Algorithmic stability
- Model complexity and uniform convergence
- Margin bounds
- Generalization from algorithms

<https://xkcd.com/720/>



# Announcements

Quiz 1 recap – today after the main lecture

287 submit mean ~ 73

Assignment 1 due next Mon

(Extra support sessions this week will be announced soon)

# Neural networks

Chap 5 PRML

Neural network as adaptive basis functions

- Weight-space symmetries

Network training

- parameter optimisation
- gradient descent

Error backpropagation and automatic differentiation

Regularisation

# Feed-forward network functions

$$y(\mathbf{x}, \mathbf{w}) = f \left( \sum_{j=1}^M w_j \phi_j(\mathbf{x}) \right) \quad (5.1)$$

logistic regression

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma \left( \sum_{j=0}^M w_{kj}^{(2)} h \left( \sum_{i=0}^D w_{ji}^{(1)} x_i \right) \right). \quad (5.9)$$

activation 2      activation

$\downarrow$                    $\downarrow$

$z_j$

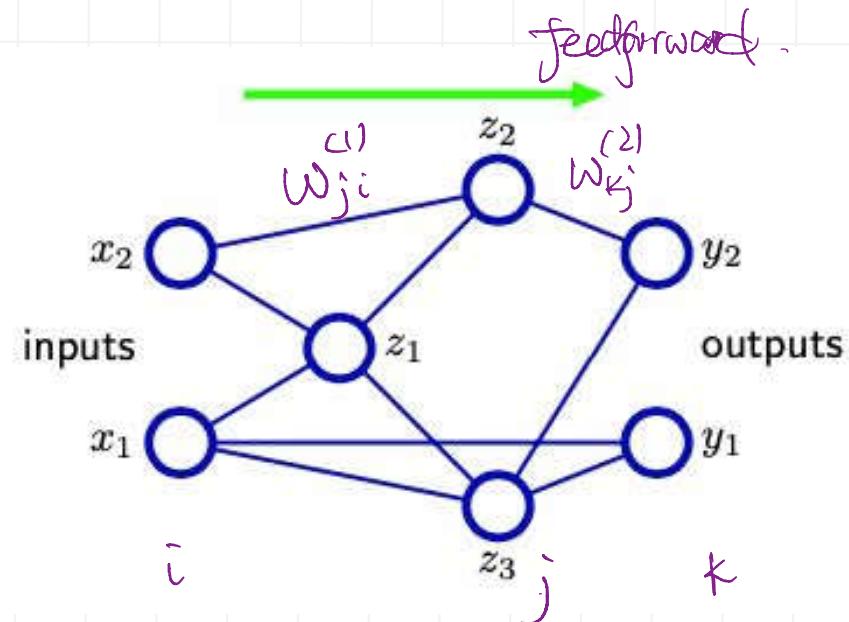
$\uparrow$   
feedforward  
 $x \rightarrow (\cdot) \rightarrow y$

**Figure 5.2** Example of a neural network having a general feed-forward topology. Note that each hidden and output unit has an associated bias parameter (omitted for clarity).

- Can have skip connections;
- Connections can be sparse;
- Require **feed-forward structure**, no directed cycles
- Convention: number of layers refers to the number of weights (rather than nodes)

Two key ideas:

- 1) Generalised linear model with activation function
- 2) Recursive composition of these



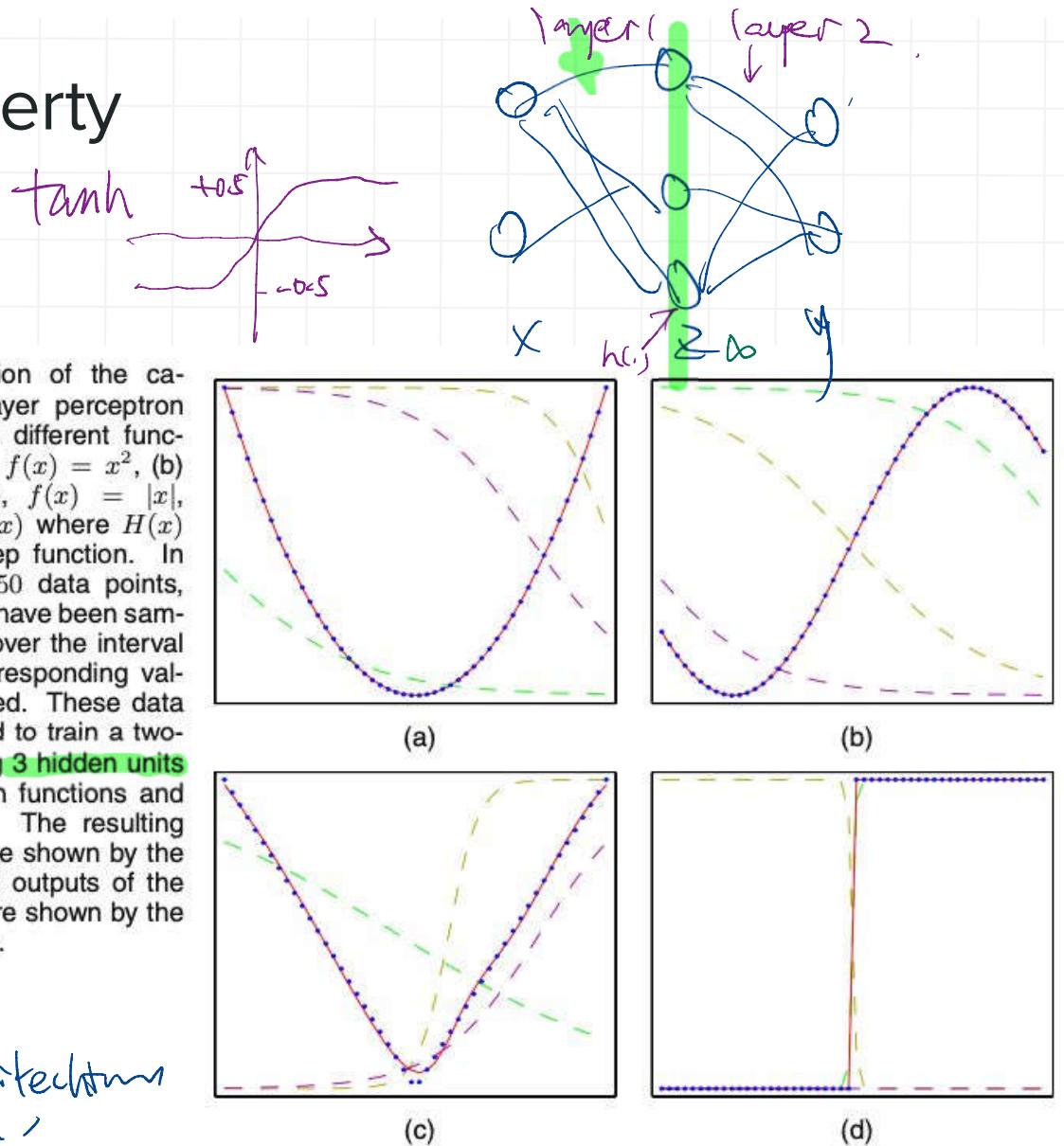
# Universal approximation property

(regression) a two-layer network with linear outputs can uniformly approximate any continuous function on a compact input domain to arbitrary accuracy provided the network has a sufficiently large number of hidden units.

(classification) a two-layer network can uniformly approximate any discriminative function on a compact input domain to arbitrary accuracy provided the network has a sufficiently large number of hidden units.

Select # of layers, neurons, etc.

Bayesian Opt. / AutoML / NN architecture search



## Comparison to Perceptron

- A neural network looks like a **multilayer perceptron**.
- But perceptron's nonlinear activation function was a step function — neither smooth nor differentiable.

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$

Sigmoid  
tanh  
relu  
...

- The activation functions  $h(\cdot)$  and  $g(\cdot)$  of a neural network are **smooth and differentiable**.

having gradients 'are nice' ;  
everywhere .

What if:

All activation functions are linear?  $\rightarrow$  linear layer

Number of hidden units smaller than the number of input dimensions?

$\rightarrow$  No UA

Linear fn  $h(\cdot)$   
 $h(\alpha \cdot x) = \alpha h(x)$ ,

$$f(x) = \frac{1}{1 + e^{-x}}$$

$f(\alpha x) \neq \alpha f(x)$

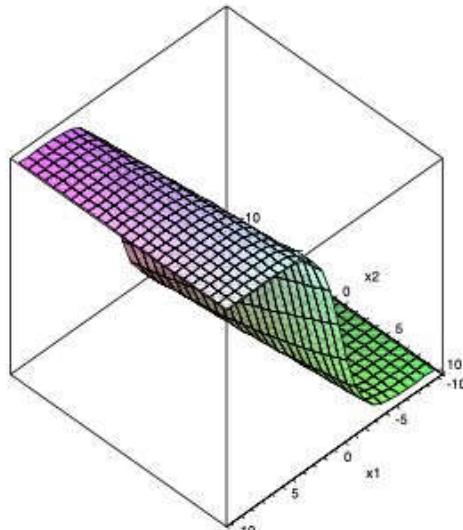


don't need  $\phi(x)$

## Neural network as variable basis functions

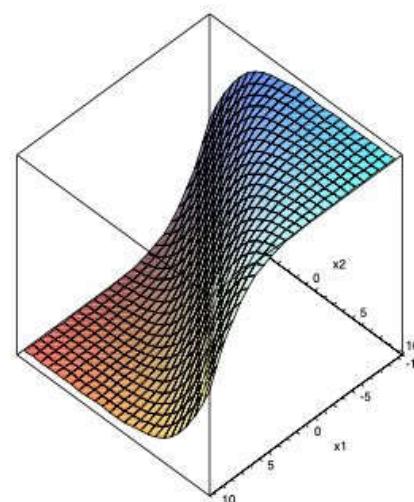
"tuned to training data"

$$z = \sigma(w_0 + w_1x_1 + w_2x_2)$$

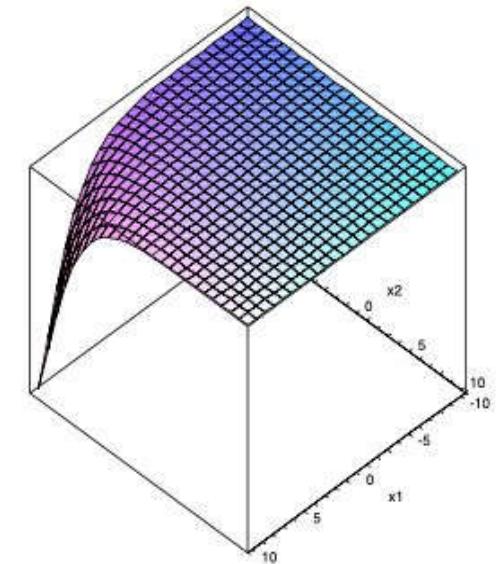


$$(w_0, w_1, w_2) = (0.0, 1.0, 0.1)$$

$$(w_0, w_1, w_2) = (0.0, 0.1, 1.0)$$



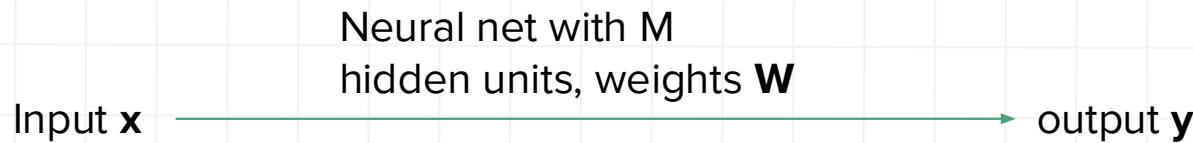
$$(w_0, w_1, w_2) = (0.0, -0.5, 0.5)$$



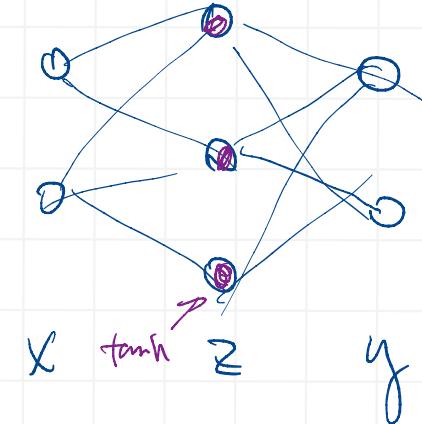
$$(w_0, w_1, w_2) = (10.0, -0.5, 0.5)$$

(the source of these figures is lost -- let me know if you encounter them, with apologies to the author)

# Weight-space symmetries



$$\mathbf{W} = [w^{(1)}, w^{(2)}]^M$$



Q: is there another set of weights  $\mathbf{W}'$  that always map the same input to the same output as  $\mathbf{W}$  .

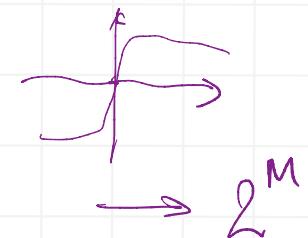
$$y = f(x)$$

$\underbrace{[w^{(1)}, w^{(2)}]}_{\mathbf{W}} \rightarrow w$

① Sign flips

e.g.  $\tanh(\cdot)$

$$-w^{(1)} \quad -w^{(2)}$$



$$y = f(x)$$

$w'$

②  $z = [z_1, z_2, z_3] \rightarrow$  "permutation invariant"

$$M!$$

$$\text{combined } 2^M \cdot M!$$

# What we covered so far

Neural network as adaptive basis functions

- Weight-space symmetries

Network training

- parameter optimisation
- gradient descent

Error backpropagation and automatic differentiation

Regularisation

# Neural network training -- objectives

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma \left( \sum_{j=0}^M w_{kj}^{(2)} h \left( \sum_{i=0}^D w_{ji}^{(1)} x_i \right) \right). \quad (5.9)$$

Regression (linear activation at output)

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(\mathbf{x}_n, \mathbf{w}) - t_n\}^2 \quad (5.14)$$

*over fitting.*

Binary classification (logistic output activation)

$$t_n \in \{0, 1\}$$

$$E(\mathbf{w}) = - \sum_{n=1}^N \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\} \quad (5.21) \quad \text{also (4.90)}$$

*either term is active  
not both.*

# Different versions of classification

Binary classification (logistic output activation)

$$E(\mathbf{w}) = - \sum_{n=1}^N \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\} \quad (5.21)$$

Multiple/independent binary classification

←  
cat? Y/N  
car? Y/N

$$E(\mathbf{w}) = - \sum_{n=1}^N \sum_{k=1}^K \{t_{nk} \ln y_{nk} + (1 - t_{nk}) \ln(1 - y_{nk})\} \quad (5.23)$$

Multi-class classification

{truck, crane, digger, car, ... }

1-hot encoding [0, 0, 1, ..., 0]

$$E(\mathbf{w}) = - \sum_{n=1}^N \sum_{k=1}^K t_{kn} \ln y_k(\mathbf{x}_n, \mathbf{w}). \quad (5.24)$$

only 1 out of K terms active,

# Recap: Convex functions

**Figure 1.31** A convex function  $f(x)$  is one for which every chord (shown in blue) lies on or above the function (shown in red).

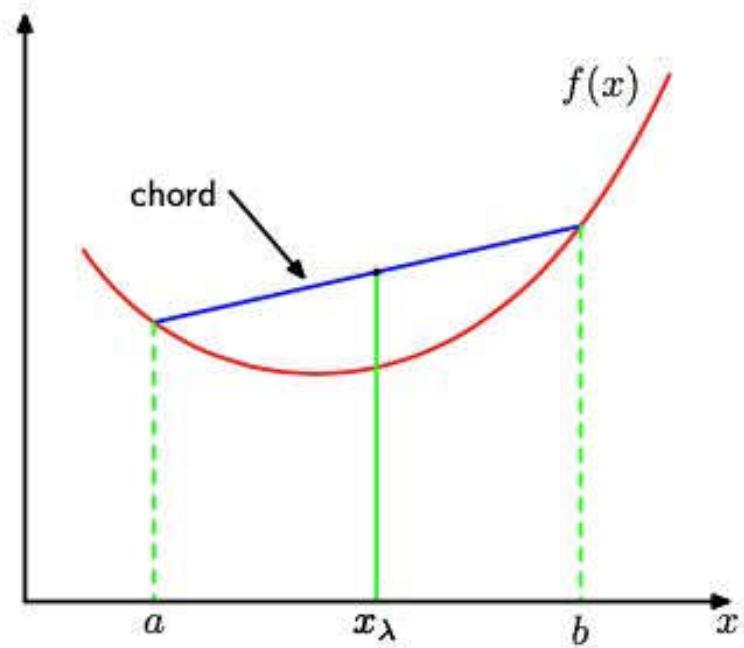
$$f(\lambda a + (1 - \lambda)b) \leq \lambda f(a) + (1 - \lambda)f(b). \quad (1.114)$$

$$f'' \geq 0$$

Examples:  $x^2$ ,  $x \ln x$  ( $x > 0$ ), ...  $\mathbf{x}^\top S \mathbf{x}$  ( $S$  positive semi definite)

Jesen's inequality

$$f(\mathbb{E}[x]) \leq \mathbb{E}[f(x)] \quad (1.116)$$



$$\frac{\partial^2 f}{\partial x^2} \geq 0$$

positive semi-definite,

# Non-linear optimisation

$\mathcal{O}(\cdot) / \mathcal{h}(\cdot)$  non-linear  
many options.

Task: find weight vector  $\mathbf{w}$ , that minimizes the function  $E(\mathbf{w})$

$$\nabla E(\mathbf{w}) = 0 \quad (5.26)$$

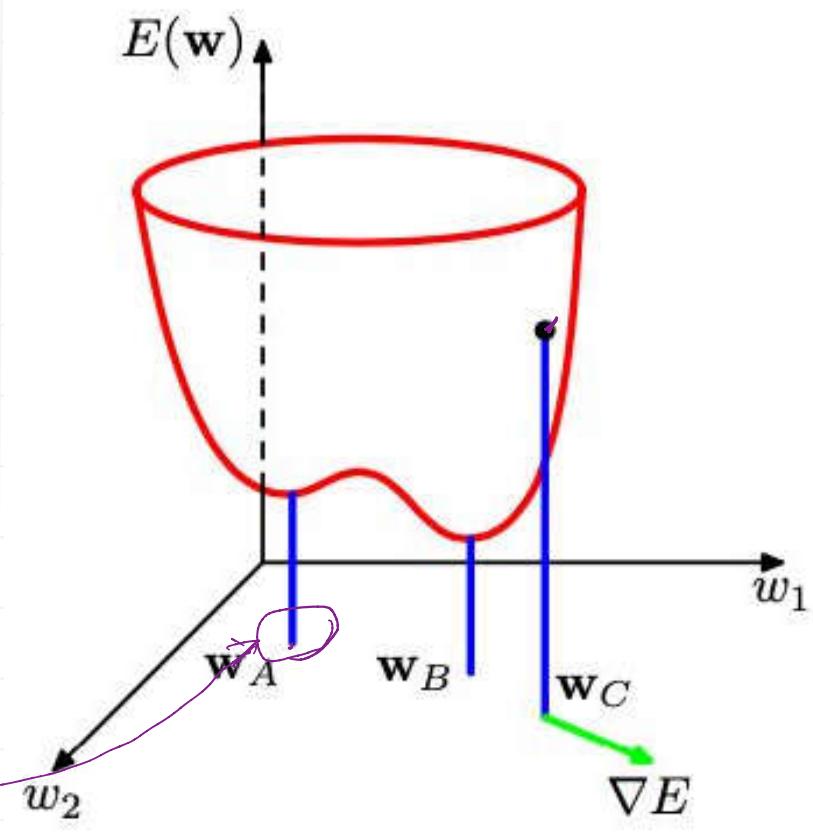
stationary point

**Definition (Global Minimum)**

A point  $\mathbf{w}^*$  for which the error  $E(\mathbf{w}^*)$  is smaller than any other error  $E(\mathbf{w})$ .

**Definition (Local Minimum)**

A point  $\mathbf{w}^*$  for which the error  $E(\mathbf{w}^*)$  is smaller than any other error  $E(\mathbf{w})$  in some neighbourhood of  $\mathbf{w}^*$ .

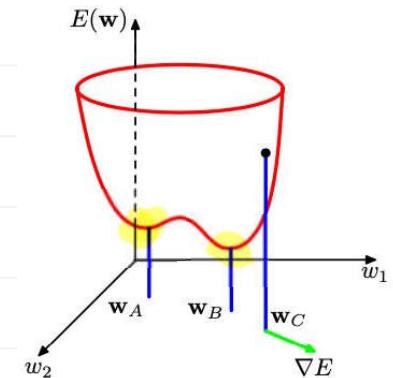


# Non-linear optimisation (a local view)

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \underbrace{\Delta\mathbf{w}^{(\tau)}}_{(5.27)}$$

Goal: figure out a good direction  $\Delta\mathbf{w}$  (and how far)

$$f(x) \quad f(x_0) \quad (x - x_0) \quad f'(x)$$



Taylor expansion

at  $\hat{\mathbf{w}}$

$$E(\mathbf{w}) \simeq E(\hat{\mathbf{w}}) + (\mathbf{w} - \hat{\mathbf{w}})^T \mathbf{b} + \frac{1}{2}(\mathbf{w} - \hat{\mathbf{w}})^T \mathbf{H}(\mathbf{w} - \hat{\mathbf{w}}) \quad (5.28)$$

$10^6$  millions  
gradient/Jacobian

$$\mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_D \end{bmatrix} \rightarrow \nabla E = \begin{bmatrix} \frac{\partial E}{\partial w_1} \\ \vdots \\ \frac{\partial E}{\partial w_D} \end{bmatrix}$$

$$\underline{\mathbf{b}} \equiv \underline{\nabla E}|_{\mathbf{w}=\hat{\mathbf{w}}} \quad (5.29)$$

Hessian

$\mathcal{O}(10^2)$

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1^2} & \frac{\partial^2 E}{\partial w_1 \partial w_2} & \dots & \frac{\partial^2 E}{\partial w_1 \partial w_D} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial w_D \partial w_1} & \dots & \ddots & \frac{\partial^2 E}{\partial w_D^2} \end{bmatrix}$$

$$(\mathbf{H})_{ij} \equiv \left. \frac{\partial E}{\partial w_i \partial w_j} \right|_{\mathbf{w}=\hat{\mathbf{w}}} . \quad (5.30)$$

$\mathbf{H}$  is p.s.d.  $\rightarrow$  local minima. Many other geometric structures possible, maxima, saddle points, ring structure ...

$$\nabla E = 0$$

# Gradient descent: the computational argument

- Hessian is symmetric and contains  $W(W + 1)/2$  independent entries where  $W$  is the total number of weights in the network.
- If we use function evaluations only:
  - Need to gather this  $O(W^2)$  pieces of information by doing  $O(W^2)$  number of function evaluations each of which cost  $O(W)$  time, for an overall cost of order  $O(W^3)$ .
- If we use gradients of the function:
  - Surprisingly the gradient  $\nabla E$  also costs only  $O(W)$  time, although it provides  $W$  pieces of information.
  - We now need only  $O(W)$  steps, so the order of time complexity is reduced to  $O(W^2)$ .

$$\frac{\partial^2 E}{\partial w_i \partial w_j} = \frac{\partial^2 E}{\partial w_j \partial w_i}$$

$O(W)$

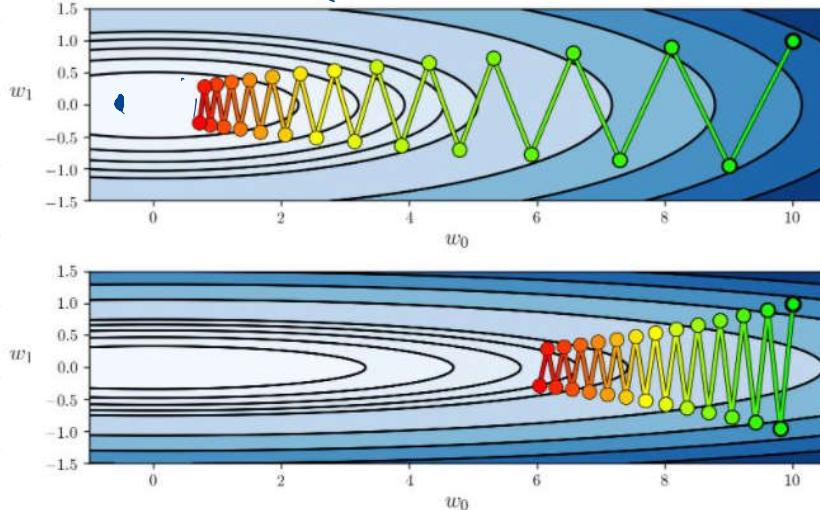
each  $w_i$ ,  $i = 1, \dots, W$   
↓ goes through one multiply

$$y_k \sim \Gamma\left(\sum_i w_{ji} h\left(\sum_i w_i x_i\right)\right)$$

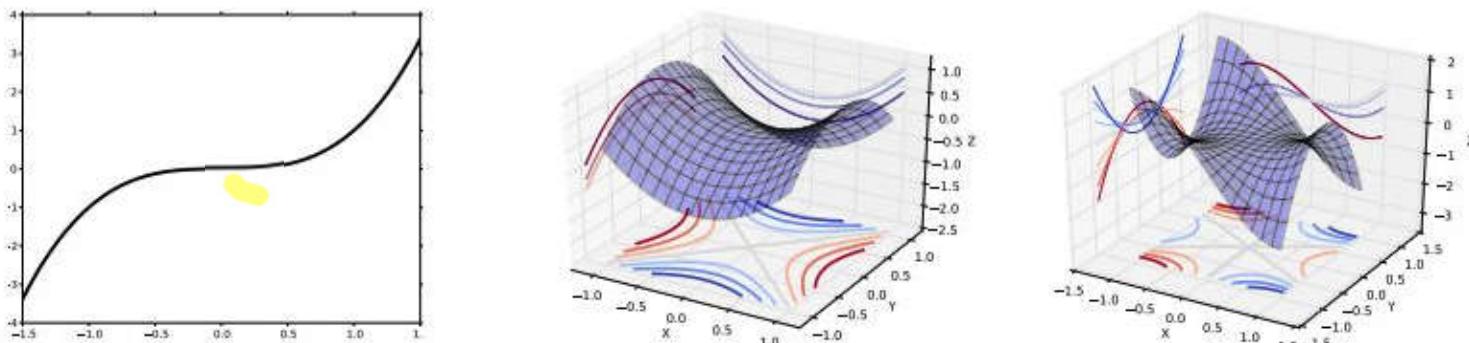
The **Cheap Gradient Principle** (Griewank 2008) --- the computational cost of computing the gradient of a scalar-valued function is nearly the same (often within a factor of 5) as that of simply computing the function itself --- is of central importance in optimization; it allows us to quickly obtain (high dimensional) gradients of scalar loss functions which are subsequently used in black box gradient-based optimization procedures.

# Limitations of gradient-based methods

Long valleys



Saddle points



Follow-up topics:

- \* Newton methods
- \* Quasi-Newton methods, e.g. L-BFGS
- \* conjugate gradient descent
- \* momentum
- \*
- ...

[https://jermwatt.github.io/machine\\_learning\\_refined/notes/3\\_First\\_order\\_methods/3\\_7\\_Problems.html](https://jermwatt.github.io/machine_learning_refined/notes/3_First_order_methods/3_7_Problems.html)

“On the saddle point problem for non-convex optimization”, Pascanu et al, <https://arxiv.org/abs/1405.4604>

# What we covered so far

Neural network as adaptive basis functions

- Weight-space symmetries

Network training

- parameter optimisation
- gradient descent

Error backpropagation and automatic differentiation

Regularisation

# Gradient descent optimisation

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)}) \quad (5.41)$$

Online version, stochastic gradient

$$E(\mathbf{w}) = \sum_{n=1}^N E_n(\mathbf{w}). \quad (5.42)$$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n(\mathbf{w}^{(\tau)}). \quad (5.43)$$

## Error backpropagation: linear version

$$y_k = \sum_i w_{ki} x_i \quad (5.45)$$

$$y_{nk} = y_k(\mathbf{x}_n, \mathbf{w})$$

$$E_n = \frac{1}{2} \sum_k (y_{nk} - t_{nk})^2 \quad (5.46)$$

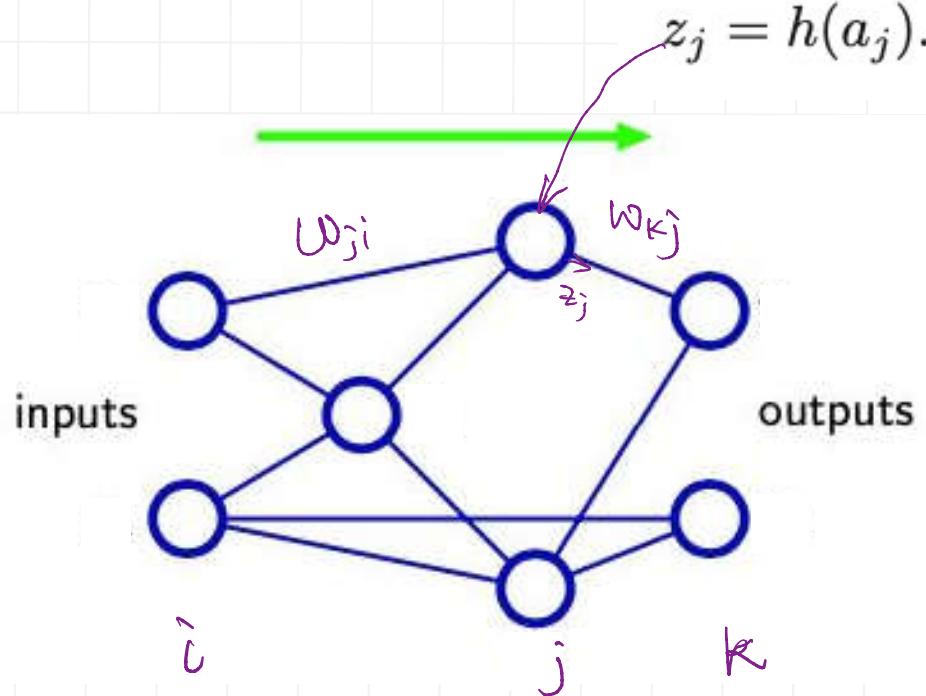
$$E_{n_k} = \frac{1}{2} (y_{nk} - t_{nk})^2$$

$$\frac{\partial E_n}{\partial w_{ji}} = \underbrace{(y_{nj} - t_{nj})}_{\text{error for output } n\text{th data } j\text{th component}} x_{ni} \quad (5.47)$$

## Forward propagation in general

$$\underline{a_j} = \sum_i w_{ji} z_i \quad (5.48)$$

$$z_j = h(a_j). \quad (5.49)$$



## Backpropagation for 1 layer

$$a_j = \sum_i w_{ji} z_i$$

$z_j = h(a_j).$

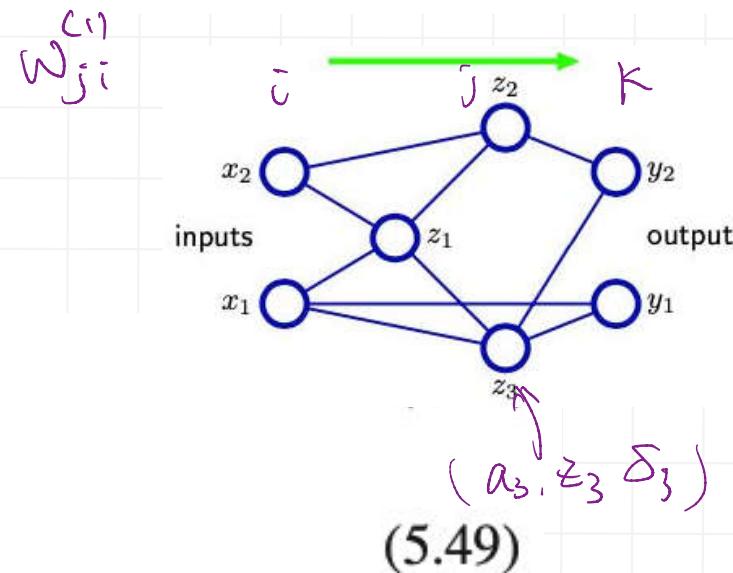
$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}$ .

(5.50)

$$\delta_j \equiv \frac{\partial E_n}{\partial a_j} \quad \frac{\partial a_j}{\partial w_{ji}} = z_i.$$

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i.$$

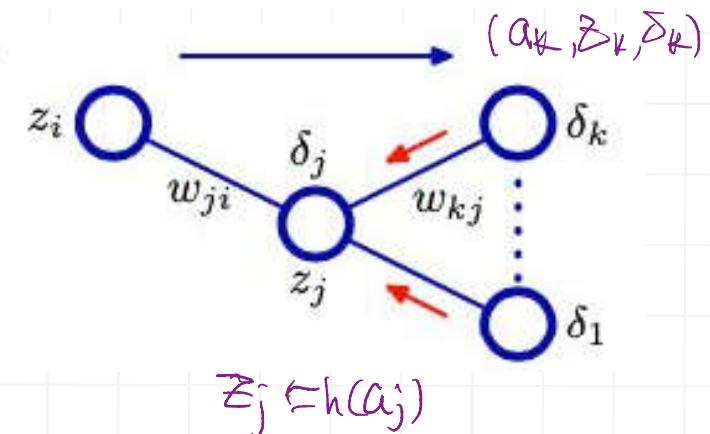
(5.53)



**Figure 5.7** Illustration of the calculation of  $\delta_j$  for hidden unit  $j$  by backpropagation of the  $\delta$ 's from those units  $k$  to which unit  $j$  sends connections. The blue arrow denotes the direction of information flow during forward propagation, and the red arrows indicate the backward propagation of error information.

$$\delta_j \equiv \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial a_j} \quad (5.55)$$

over all output units  $\sum_k$ ,  $k=1, \dots, k$

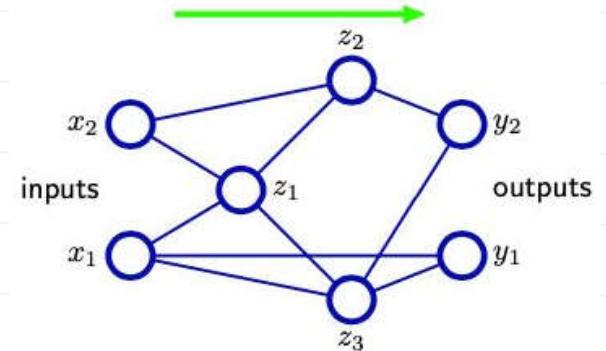


$$\delta_j = \underline{h'(a_j)} \sum_k \underline{w_{kj}} \underline{\delta_k} \quad (5.56)$$

$$\underline{E(\mathbf{w})} = - \sum_{n=1}^N \sum_{k=1}^K \{ t_{nk} \ln y_{nk} + (1 - t_{nk}) \ln(1 - y_{nk}) \} \quad (5.23)$$

$$E_n(\omega) = \frac{1}{\pi} \sum_{k=1}^{\infty} \frac{1}{\omega_k - \omega}$$

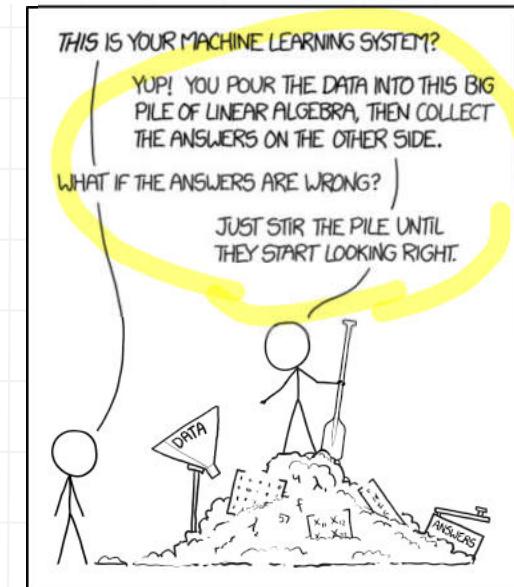
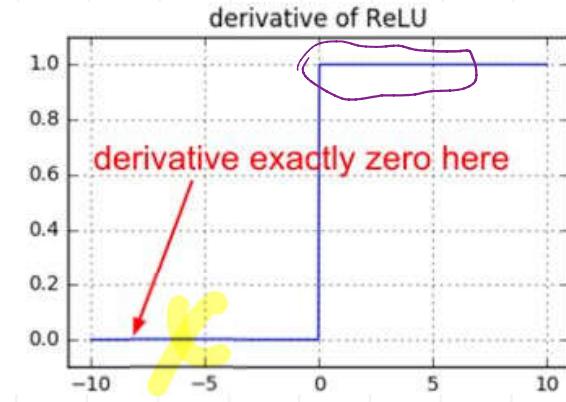
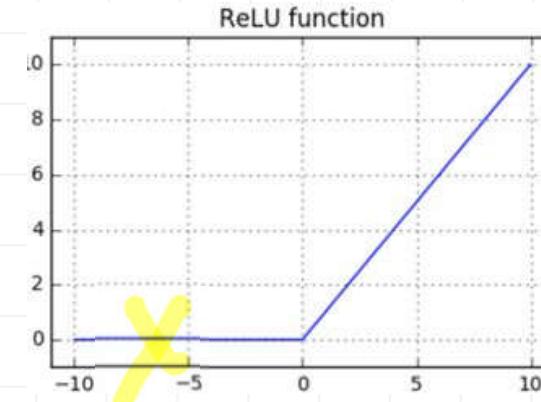
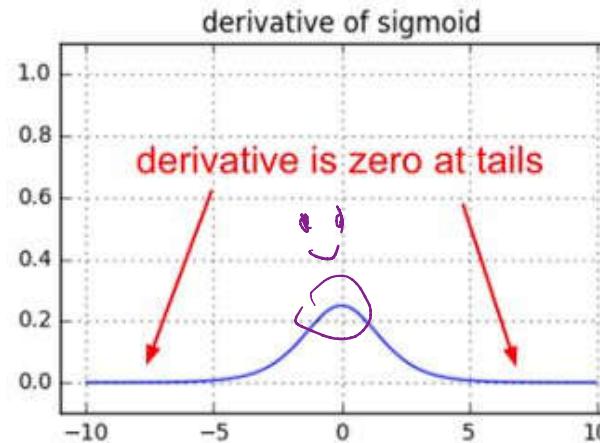
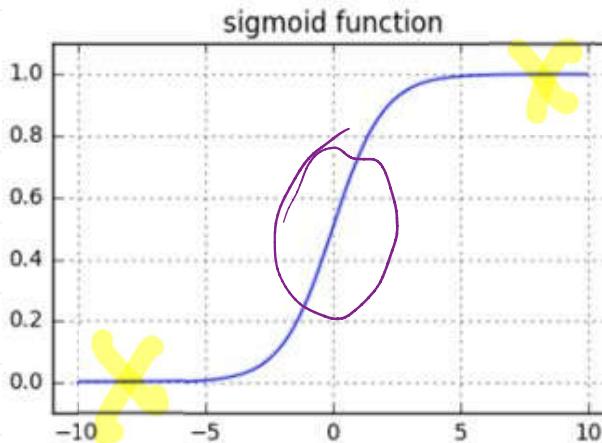
$$y_k(\mathbf{x}, \mathbf{w}) = \sigma \left( \sum_{j=0}^M w_{kj}^{(2)} h \left( \sum_{i=0}^D w_{ji}^{(1)} x_i \right) \right). \quad (5.9)$$



# “Yes you should understand backprop”

<https://medium.com/@karpathy/yes-you-should-understand-backprop-e2f06eab496b>

It's a *leaky abstraction* - it is easy to fall into the trap of abstracting away the learning process — believing that you can simply stack arbitrary layers together and backprop will “magically make them work” on your data

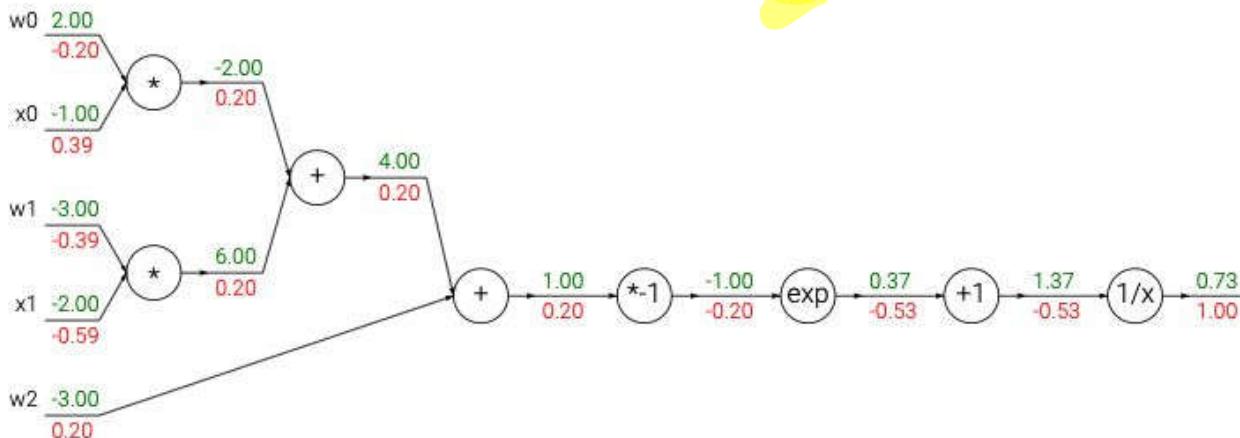


<https://xkcd.com/1838/>

<https://cs231n.github.io/optimization-2/>, also see MML Sec 5.6, esp 5.6.2

# Implementing Backprop using Automatic Differentiation

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = \frac{1}{x}$$

$$f_c(x) = c + x$$

$$f(x) = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = -1/x^2$$

$$\frac{df}{dx} = 1$$

$$\frac{df}{dx} = e^x$$

$$\frac{df}{dx} = a$$

→

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

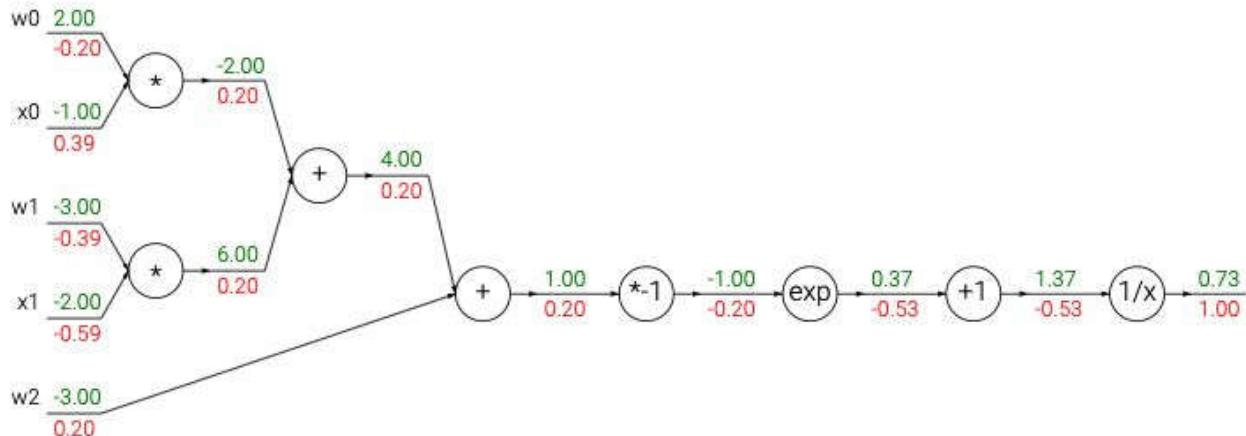
100%

100%

<https://cs231n.github.io/optimization-2/>, also see MML Sec 5.6, esp 5.6.2

# Implementing Backprop using Automatic Differentiation

- Have differentials of elementary function
- Use chain rule to compose the stage-wise gradient
- Caching - store the differentials and function values as computer variables



Example circuit for a 2D neuron with a sigmoid activation function. The inputs are  $[x_0, x_1]$  and the (learnable) weights of the neuron are  $[w_0, w_1, w_2]$ . As we will see later, the neuron computes a dot product with the input and then its activation is softly squashed by the sigmoid function to be in range from 0 to 1.

# What we covered so far

Neural network as adaptive basis functions

- Weight-space symmetries

Network training

- parameter optimisation
- gradient descent

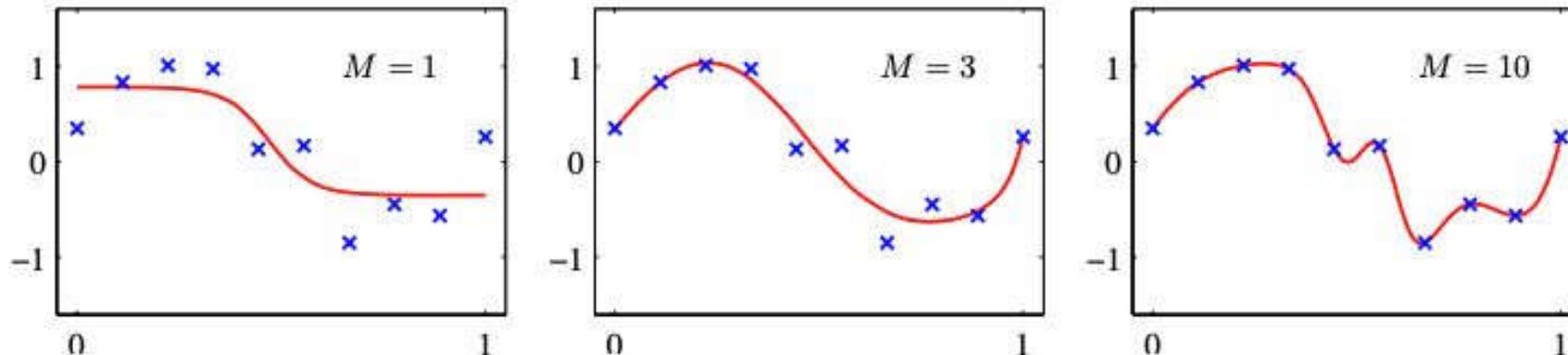
Error backpropagation and automatic differentiation

Regularisation

# Regularising neural networks

The number of input and output nodes determined by the application.

The number of hidden nodes is a free parameter.

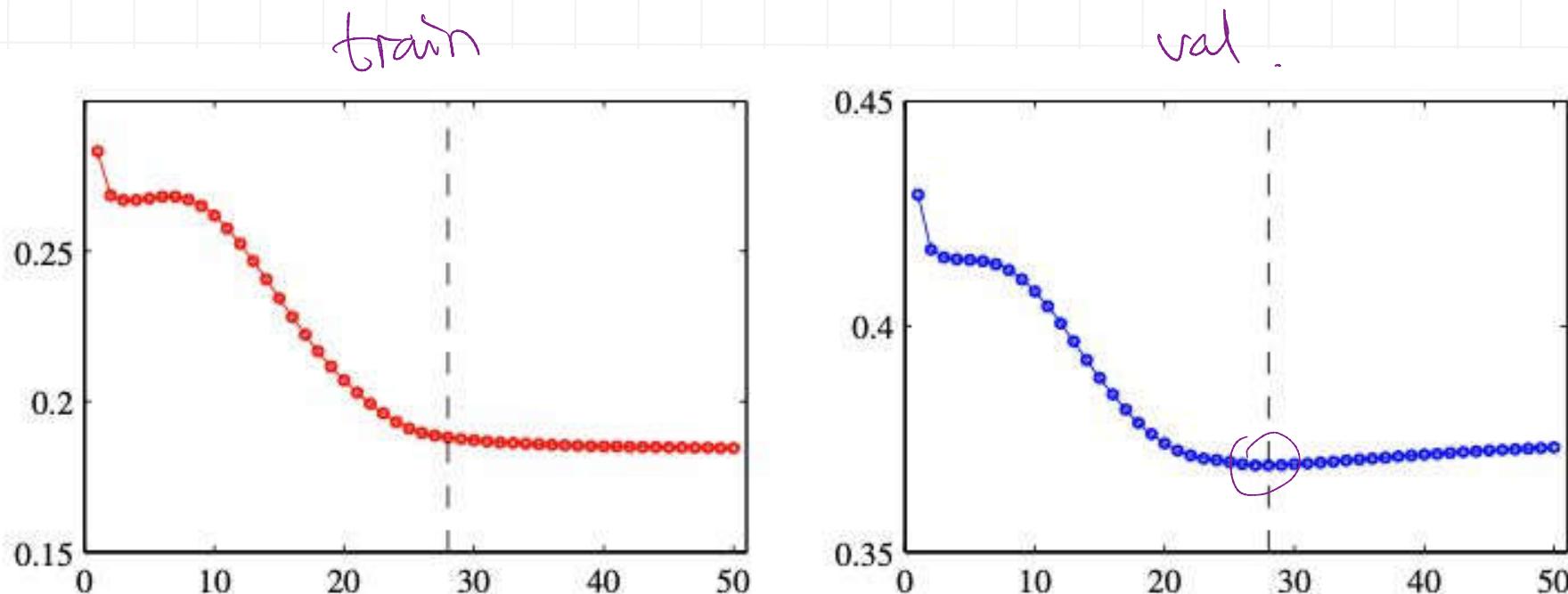


**Figure 5.9** Examples of two-layer networks trained on 10 data points drawn from the sinusoidal data set. The graphs show the result of fitting networks having  $M = 1, 3$  and 10 hidden units, respectively, by minimizing a sum-of-squares error function using a scaled conjugate-gradient algorithm.

Recipe: use the L2 regularisation that we know

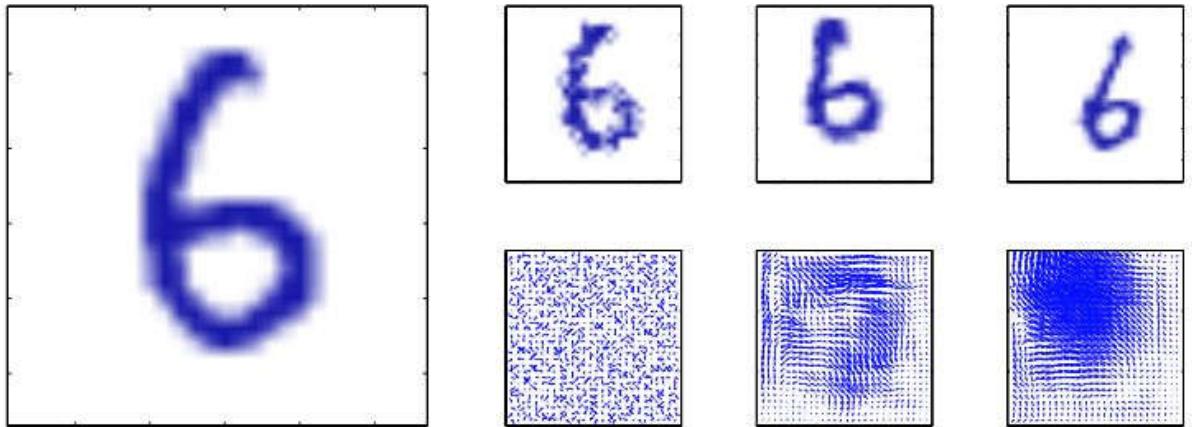
$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}. \quad (5.112)$$

## Regularisation by early-stopping



**Figure 5.12** An illustration of the behaviour of training set error (left) and validation set error (right) during a typical training session, as a function of the iteration step, for the sinusoidal data set. The goal of achieving the best generalization performance suggests that training should be stopped at the point shown by the vertical dashed lines, corresponding to the minimum of the validation set error.

# Invariances



**Figure 5.14** Illustration of the synthetic warping of a handwritten digit. The original image is shown on the left. On the right, the top row shows three examples of warped digits, with the corresponding displacement fields shown on the bottom row. These displacement fields are generated by sampling random displacements  $\Delta x, \Delta y \in (0, 1)$  at each pixel and then smoothing by convolution with Gaussians of width 0.01, 30 and 60 respectively.

- + Augment training set with perturbed/transformed training patterns (Fig 5.14)
- + Preprocess input by normalising against transformations (e.g. rectifying faces)
- + Build important invariance into network structure -- e.g. Convolutional Nets
- Explicitly allow invariances using improper priors (as regulariser) - 5.5.1 , tangent propagation (5.5.4)

## Bayesian Neural Networks

- Predict a single target  $t$  from a vector of inputs  $\mathbf{x}$
- Assume conditional distribution to be Gaussian with precision  $\beta$

$$p(t | \mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t | y(\mathbf{x}, \mathbf{w}), \beta^{-1})$$

- Prior distribution over weights  $\mathbf{w}$  is also assumed to be Gaussian

$$p(\mathbf{w} | \alpha) = \mathcal{N}(\mathbf{w} | \mathbf{0}, \alpha^{-1} \mathbf{I})$$

- For an i.i.d training data set  $\{\mathbf{x}_n, t_n\}_{n=1}^N$ , the likelihood of the targets  $\mathcal{D} = \{t_1, \dots, t_N\}$  is

$$p(\mathcal{D} | \mathbf{w}, \beta) = \prod_{n=1}^N \mathcal{N}(t_n | y(\mathbf{x}_n, \mathbf{w}), \beta^{-1})$$

- Posterior distribution

$$p(\mathbf{w} | \mathcal{D}, \alpha, \beta) \propto p(\mathbf{w} | \alpha) p(\mathcal{D} | \mathbf{w}, \beta)$$

# Troubling Trends in Machine Learning Scholarship

Zachary C. Lipton\* & Jacob Steinhardt\*

Carnegie Mellon University, Stanford University

[zlipton@cmu.edu](mailto:zlipton@cmu.edu), [jsteinhardt@cs.stanford.edu](mailto:jsteinhardt@cs.stanford.edu)

July 27, 2018

## 3.4 Misuse of Language

We identify three common avenues of language misuse in machine learning: *suggestive definitions*, *overloaded terminology*, and *suitcase words*.

### 3.4.1 Suggestive Definitions

In the first avenue, a *new technical term is coined* that has a *suggestive colloquial meaning*, thus sneaking in connotations without the need to argue for them. This often manifests in anthropomorphic characterizations of tasks (*reading comprehension* [31] and *music composition* [59]) and techniques (*curiosity* [66] and *fear* [48]). A number of papers name components of proposed models in a manner suggestive of human cognition, e.g. “*thought vectors*” [36] and the “consciousness prior” [4]. Our goal is not to rid the academic literature of all such language; when properly qualified, these connections might communicate a fruitful source of inspiration. However, when a suggestive term is assigned technical meaning, each subsequent paper has no choice but to confuse its readers, either by embracing the term or by replacing it.

Describing empirical results with loose claims of “*human-level*” *performance* can also portray a false sense of current capabilities. Take, for example, the “*dermatologist-level classification of skin cancer*” reported in [21]. The comparison to dermatologists conceals the fact that classifiers and dermatologists perform fundamentally different tasks. Real dermatologists encounter a wide variety of circumstances and must perform their jobs despite unpredictable changes. The machine

### 3.4.2 Overloading Technical Terminology

A second avenue of misuse consists of taking a term that holds precise technical meaning and using it in an imprecise or contradictory way. Consider the case of *deconvolution*, which formally describes the process of reversing a convolution, but is now used in the deep learning literature to refer to transpose convolutions (also called up-convolutions) as commonly found in auto-encoders and generative adversarial networks. This term first took root in deep learning in [79], which does address deconvolution, but was later over-generalized to refer to any neural architectures using upconvolutions [78, 50]. Such overloading of terminology can create lasting confusion. New machine learning papers referring to deconvolution might be (i) invoking its original meaning, (ii) describing

As another example, *generative models* are traditionally models of either the input distribution  $p(x)$  or the joint distribution  $p(x, y)$ . In contrast, discriminative models address the conditional distribution  $p(y | x)$  of the label given the inputs. However, in recent works, “generative model” imprecisely refers to any model that produces realistic-looking structured data. On the surface, this may seem consistent with the  $p(x)$  definition, but it obscures several shortcomings—for instance, the inability of GANs or VAEs to perform conditional inference (e.g. sampling from  $p(x_2 | x_1)$  where  $x_1$  and  $x_2$  are two distinct input features). Bending the term further, some discriminative models are now referred to as generative models on account of producing structured outputs [76], a mistake that we (ZL) make in [47]. Seeking to resolve the confusion and provide historical context, [58] distinguishes between *prescribed* and *implicit* generative models.

Among the consequences of mis-using language is that (as with generative models) we might conceal lack of progress by redefining an unsolved task to refer to something easier. This often combines with suggestive definitions via anthropomorphic naming. *Language understanding* and *reading comprehension*, once grand challenges of AI, now refer to making accurate predictions on specific datasets [31].

### 3.4.3 Suitcase Words

Finally, we discuss the overuse of *suitcase words* in ML papers. Coined by Minsky in the 2007 book *The Emotion Machine* [56], suitcase words pack together a variety of meanings. Minsky describes mental processes such as *consciousness*, *thinking*, *attention*, *emotion*, and *feeling* that may not share “a single cause or origin”. Many terms in ML fall into this category. For example, [46] notes that *interpretability* holds no universally agreed-upon meaning, and often references disjoint methods and desiderata. As a consequence, even papers that appear to be in dialogue with each other may have different concepts in mind.

As another example, *generalization* has both a specific technical meaning (generalizing from train to test) and a more colloquial meaning that is closer to the notion of *transfer* (generalizing from one population to another) or of external validity (*generalizing from an experimental setting to the real world*) [67]. Conflating these notions leads to overestimating the capabilities of current systems.

Suggestive definitions and overloaded terminology can contribute to the creation of new suitcase words. In the fairness literature, where legal, philosophical, and statistical language are often overloaded, terms like *bias* become suitcase words that must be subsequently unpacked [17].

In common speech and as aspirational terms, suitcase words can serve a useful purpose. Perhaps the suitcase word reflects an overarching concept that unites the various meanings. For example, *artificial intelligence* might be well-suited as an aspirational name to organize an academic department. On the other hand, using suitcase words in technical arguments can lead to confusion. For example, [6] writes an equation (Box 4) involving the terms *intelligence* and *optimization power*, implicitly assuming that these suitcase words can be quantified with a one-dimensional scalar.

# Neural networks

Neural network as adaptive basis functions

- Weight-space symmetries

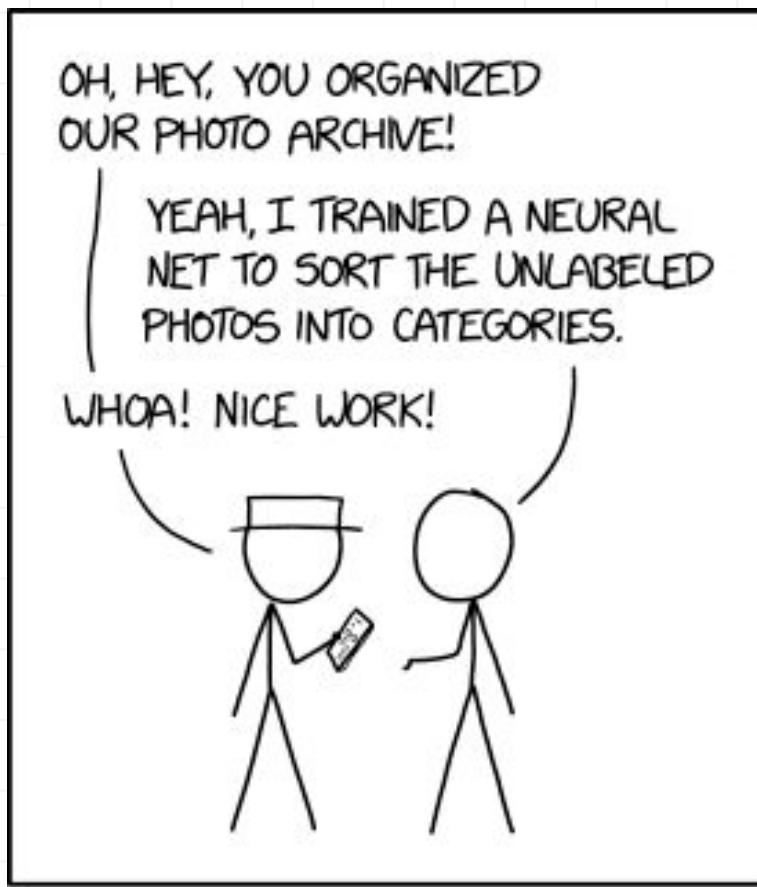
Network training

- parameter optimisation
- gradient descent

Error backpropagation and automatic differentiation

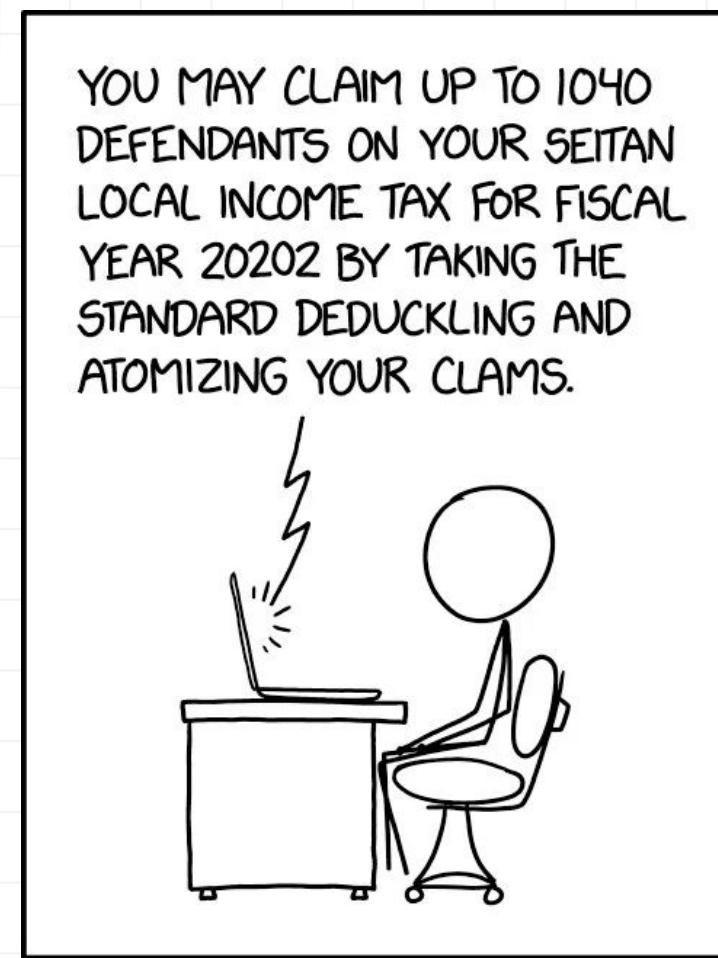
Regularisation

<https://xkcd.com/2173/>



ENGINEERING TIP:  
WHEN YOU DO A TASK BY HAND,  
YOU CAN TECHNICALLY SAY YOU  
TRAINED A NEURAL NET TO DO IT.

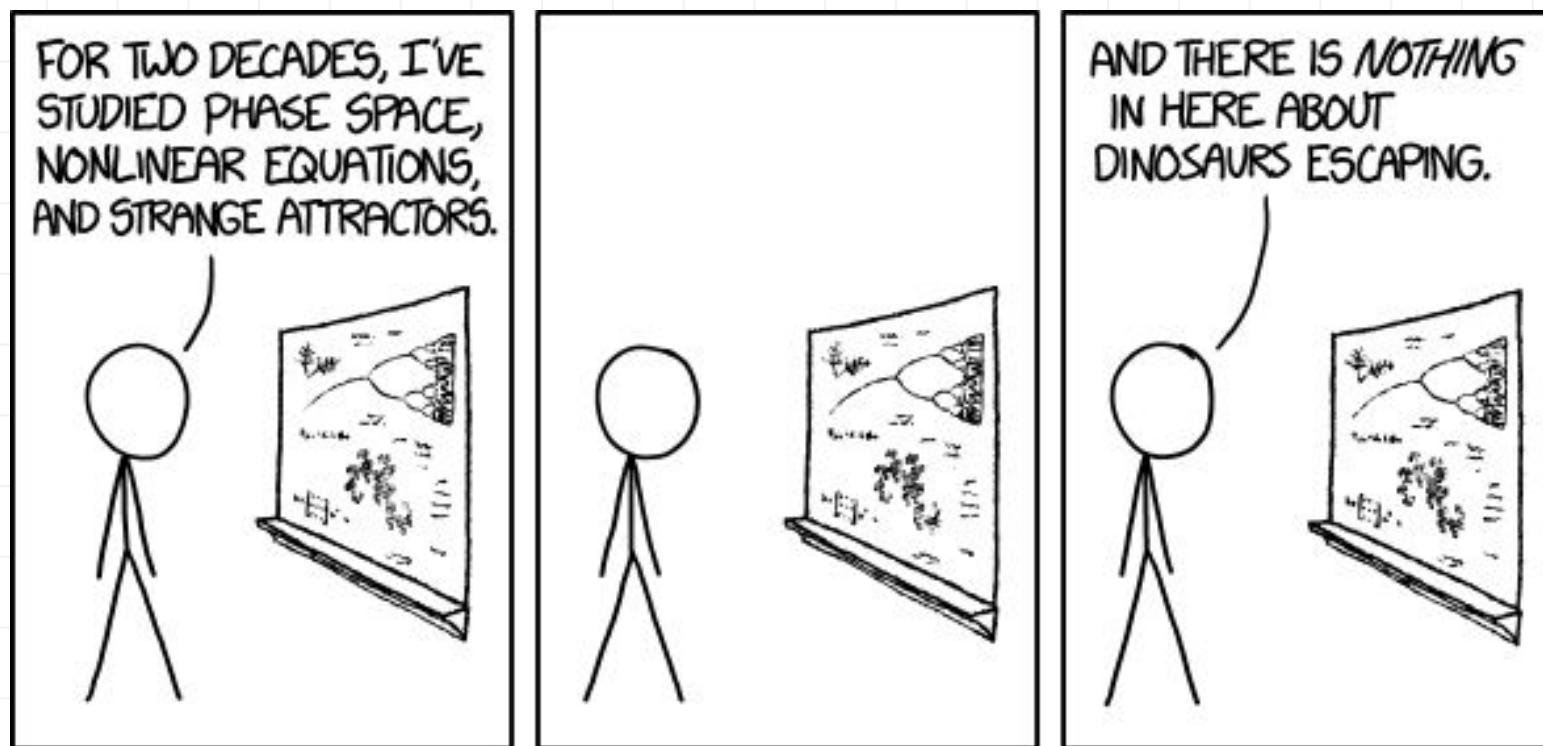
<https://xkcd.com/2265/>



I USED A NEURAL NET TO PREPARE  
MY TAX RETURNS, BUT I THINK I  
CUT OFF ITS TRAINING TOO EARLY.



<https://xkcd.com/1399/>



# Linear and Nonlinear Component Analysis

Probabilistic PCA

[PRML book]

Pre-read PCA - 12.1

This lecture:

12.2, 12.2.1, 12.4.2

Autoencoders

Kernel PCA 12.3 (read after  
week6 lectures)

Autoencoders for image processing

Recsys:  
22.1 of Murphy's PML book

Recommender systems

Assignment 1 Q&A Thu 1-3 pm

Fri 3-5 pm

## PCA: recap

data  $\{x_1, \dots, x_n\}$   
 $x_i \in \mathbb{R}^D$   
 $D \rightarrow M$

Maximum variance projection

$$x_n = \sum_{i=1}^D \underbrace{(x_n^T u_i)}_{\text{assignment to diff components.}} u_i. \quad (12.9)$$

Covariance matrix of  
 (centered) data

$$\underline{\mathbf{S}} = \frac{1}{N} \sum_{n=1}^N (x_n - \bar{x})(x_n - \bar{x})^T$$

$$\mathbf{S}u_i = \lambda_i u_i$$

$$u_i^T u_i = 1$$

$$u_i^T u_j = 0$$

Minimum error formulation

$$\min J = \frac{1}{N} \sum_{n=1}^N \sum_{i=M+1}^D (x_n^T u_i - \bar{x}^T u_i)^2 = \sum_{i=M+1}^D u_i^T \mathbf{S} u_i. \quad (12.15) \quad = \sum_{i=M+1}^D \lambda_i$$

# Probabilistic PCA

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z} | \mathbf{0}, \mathbf{I}).$$

$$\Theta = \{W, \mu, \Sigma\}$$

(12.31)

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x} | \mathbf{Wz} + \mu, \sigma^2 \mathbf{I})$$

(12.32)

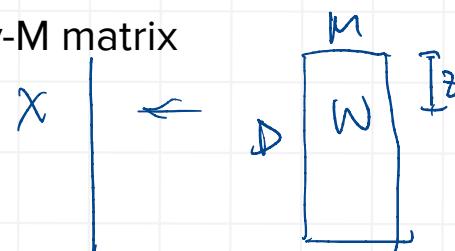
*(linear trans - center x)*

$\mathbf{x}$ : D-dimensional vector

$\mathbf{z}$ : M-dimensional Gaussian latent variable

$D \geq M$

$\mathbf{W}$ : D-by-M matrix



## Marginal and Conditional Gaussians

Given a marginal Gaussian distribution for  $\mathbf{x}$  and a conditional Gaussian distribution for  $\mathbf{y}$  given  $\mathbf{x}$  in the form

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}) \quad (2.113)$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y} | \mathbf{Ax} + \mathbf{b}, \mathbf{L}^{-1}) \quad (2.114)$$

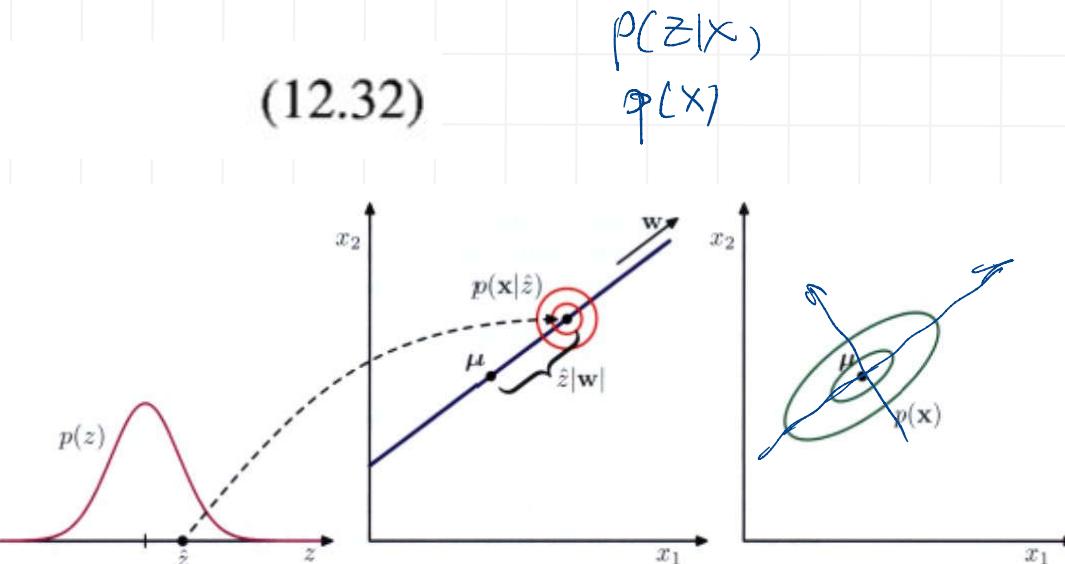
the marginal distribution of  $\mathbf{y}$  and the conditional distribution of  $\mathbf{x}$  given  $\mathbf{y}$  are given by

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y} | \mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^T) \quad (2.115)$$

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x} | \boldsymbol{\Sigma}\{\mathbf{A}^T\mathbf{L}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Lambda}\boldsymbol{\mu}\}, \boldsymbol{\Sigma}) \quad (2.116)$$

where

$$\boldsymbol{\Sigma} = (\boldsymbol{\Lambda} + \mathbf{A}^T\mathbf{L}\mathbf{A})^{-1}. \quad (2.117)$$



**Figure 12.9** An illustration of the generative view of the probabilistic PCA model for a two-dimensional data space and a one-dimensional latent space. An observed data point  $\mathbf{x}$  is generated by first drawing a value  $\hat{z}$  for the latent variable from its prior distribution  $p(z)$  and then drawing a value for  $\mathbf{x}$  from an isotropic Gaussian distribution (illustrated by the red circles) having mean  $\mathbf{w}\hat{z} + \boldsymbol{\mu}$  and covariance  $\sigma^2 \mathbf{I}$ . The green ellipses show the density contours for the marginal distribution  $p(\mathbf{x})$ .

$C^{-1}$  needs  $O(D^3)$  computation,

also (2.113)-(2.117)

## Likelihood and posterior for $x$ and $z$

$$p(z) = \mathcal{N}(z|\mathbf{0}, \mathbf{I})$$

$$p(x|z) = \mathcal{N}(x|Wz + \mu, \sigma^2 \mathbf{I})$$

$$p(x) = \int p(x|z)p(z) dz. \quad p(x) = \mathcal{N}(x|\mu, C) \quad (12.35)$$

$$C = WW^T + \sigma^2 \mathbf{I}. \quad D\text{-by-}D \quad (12.36)$$

$$x = Wz + \mu + \epsilon$$

$$\mathbb{E}[x] = \mathbb{E}[Wz + \mu + \epsilon] = \mu \quad (12.37)$$

$$\begin{aligned} \text{cov}[x] &= \mathbb{E}[(Wz + \epsilon)(Wz + \epsilon)^T] \\ &= \mathbb{E}[Wzz^TW^T] + \mathbb{E}[\epsilon\epsilon^T] = WW^T + \sigma^2 \mathbf{I} \end{aligned} \quad (12.38)$$

Woodbury / matrix inversion identity

$$(A + BD^{-1}C)^{-1} = A^{-1} - A^{-1}B(D + CA^{-1}B)^{-1}CA^{-1} \quad (C.7)$$

$$C^{-1} = \sigma^{-1} \mathbf{I} - \sigma^{-2} WM^{-1}W^T$$

$$M = W^TW + \sigma^2 \mathbf{I}.$$

M-by-M

$O(M^3)$

$$p(z|x) = \mathcal{N}(z|M^{-1}W^T(x - \mu), \sigma^{-2}M). \quad (12.42)$$

rotation matrix  $\mathbf{U}$

$$\mathbf{C} = \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}$$

Maximum likelihood for  $\mathbf{W}, \boldsymbol{\mu}, \sigma^2 \rightarrow$  noise,

$$p(\mathbf{x}) = N(\boldsymbol{\mu}, \mathbf{C})$$

$$\begin{aligned} \ln p(\mathbf{X}|\boldsymbol{\mu}, \mathbf{W}, \sigma^2) &= \sum_{n=1}^N \ln p(\mathbf{x}_n|\mathbf{W}, \boldsymbol{\mu}, \sigma^2) \\ &= -\frac{ND}{2} \ln(2\pi) - \frac{N}{2} \ln |\mathbf{C}| - \frac{1}{2} \sum_{n=1}^N (\mathbf{x}_n - \boldsymbol{\mu})^T \mathbf{C}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}). \quad (12.43) \end{aligned}$$

↳ linear Gaussian latent var.

$$\underline{\mathbf{W}_{ML}} = \underline{\mathbf{U}_M} (\underline{\mathbf{L}_M} - \sigma^2 \mathbf{I})^{1/2} \underline{\mathbf{R}} \quad (12.45)$$

$$\sigma_{ML}^2 = \frac{1}{D-M} \sum_{i=M+1}^D \lambda_i \quad (12.46)$$

$\mathbf{L}_M$ : diag  
with first  $M$  eigen  
vectors.

$\mathbf{U}_M$ : first  $M$  eigen  
vectors  
 $\mathbf{R}$ : arbitrary rotation

, 6

# EM for probabilistic PCA

$$\mathbb{E}_{\mathbf{p}(\mathbf{x}|\mathbf{z})}[\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\mu}, \mathbf{W}, \sigma^2)] = \dots + \dots + \dots$$

E-step -

$$\mathbb{E}[\mathbf{z}_n] = \mathbf{M}^{-1}\mathbf{W}^T(\mathbf{x}_n - \bar{\mathbf{x}}) \quad (12.54)$$

$$\mathbb{E}[\mathbf{z}_n \mathbf{z}_n^T] = \sigma^2 \mathbf{M}^{-1} + \mathbb{E}[\mathbf{z}_n] \mathbb{E}[\mathbf{z}_n]^T \quad (12.55)$$

M-step -

$$\mathbf{W}_{\text{new}} = \left[ \sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}}) \mathbb{E}[\mathbf{z}_n]^T \right] \left[ \sum_{n=1}^N \mathbb{E}[\mathbf{z}_n \mathbf{z}_n^T] \right]^{-1} \quad (12.56)$$

$$\begin{aligned} \sigma_{\text{new}}^2 &= \frac{1}{ND} \sum_{n=1}^N \left\{ \|\mathbf{x}_n - \bar{\mathbf{x}}\|^2 - 2\mathbb{E}[\mathbf{z}_n]^T \mathbf{W}_{\text{new}}^T (\mathbf{x}_n - \bar{\mathbf{x}}) \right. \\ &\quad \left. + \text{Tr}(\mathbb{E}[\mathbf{z}_n \mathbf{z}_n^T] \mathbf{W}_{\text{new}}^T \mathbf{W}_{\text{new}}) \right\}. \end{aligned} \quad (12.57)$$

why EM?

MLE for  $\mathbf{W}, \sigma^2$  requires  $S$

EM: only need  $M$

# Linear and Nonlinear Component Analysis

Probabilistic PCA

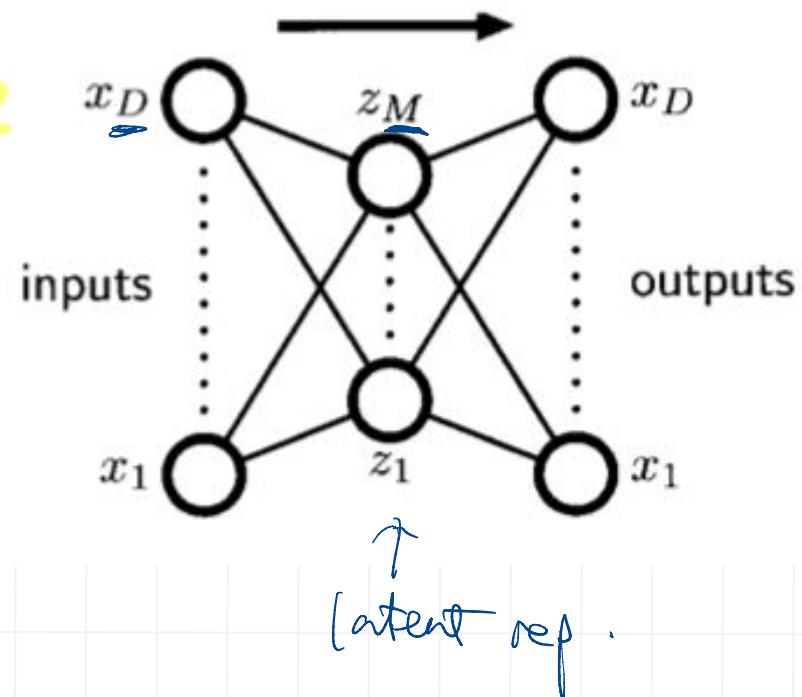
Autoencoders

Autoencoders for image processing

Recommender systems

## Two-layer associative neural nets

**Figure 12.18** An autoassociative multilayer perceptron having two layers of weights. Such a network is trained to map input vectors onto themselves by minimization of a sum-of-squares error. Even with non-linear units in the hidden layer, such a network is equivalent to linear principal component analysis. Links representing bias parameters have been omitted for clarity.



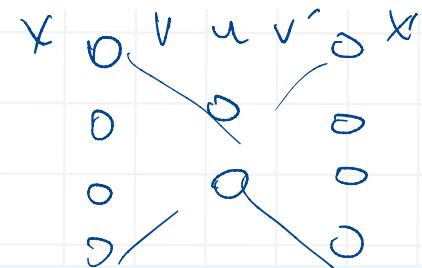
## Linear autoencoder → PCA solution

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \|\mathbf{y}(\mathbf{x}_n, \mathbf{w}) - \mathbf{x}_n\|^2. \quad (12.91)$$

If the hidden units have linear activations functions, then it can be shown that the error function has a unique global minimum, and that at this minimum the network performs a projection onto the  $M$ -dimensional subspace which is spanned by the first  $M$  principal components of the data (Bourlard and Kamp, 1988; Baldi and Hornik, 1989). Thus, the vectors of weights which lead into the hidden units in Figure 12.18 form a basis set which spans the principal subspace. Note, however, that these vectors need not be orthogonal or normalized. This result is unsurprising, since both principal component analysis and the neural network are using linear dimensionality reduction and are minimizing the same sum-of-squares error function.

# Linear autoencoder $\rightarrow$ PCA solution

$S \in \mathbb{R}^{4 \times 4}$

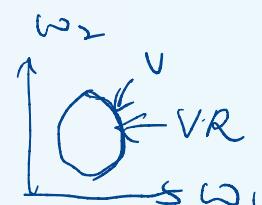


Consider an autoencoder with four inputs  $\mathbf{x} = [x_1, x_2, x_3, x_4]$ , two hidden units  $\mathbf{u} = [u_1, u_2]$ , and four outputs  $\mathbf{x}' = [x'_1, x'_2, x'_3, x'_4]$ . Both the input and output layers have linear activation. Given a training data set  $\mathbf{X} \in \mathbb{R}^{10000 \times 4}$  with zero mean, denote the covariance matrix of  $\mathbf{X}$  as  $\mathbf{S}$ , with corresponding eigen vectors  $\mathbf{V} = [v_1, v_2, v_3, v_4]$  satisfying  $\mathbf{S}v_i = \lambda_i v_i$ ,  $i = 1, 2, 3, 4$ , and  $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \lambda_4$ . We also denote the row vectors of  $\mathbf{V}$  as  $\mathbf{V} = [v_{:,1}; v_{:,2}; v_{:,3}; v_{:,4}]$ , where ";" denote vertical concatenation of row vectors.

Now we set the weights of the input layer to  $\mathbf{V}_1$  and  $\mathbf{V}_2$  for each hidden unit, respectively. We also set the weight for the output layer to be  $\mathbf{v}_{:,1}$  and  $\mathbf{v}_{:,2}$ , respectively. Which of the following statements are correct:

Select one or more:

- A. If we aim to minimize the square error between  $X$  and  $X'$  with a commonly used L2 regulariser, the weights will remain unchanged.
- B. None of the other options.
- C.  $X'$  is the minimum error reconstruction of  $X$  given this network structure.
- D. If we perform gradient descent on the square difference between  $\mathbf{x}$  and  $\mathbf{x}'$ , the weight will remain unchanged.
- E. The variance of  $X'$  is maximised given this network structure.



V.A.-of-NN

"any" function  $z = f(x)$

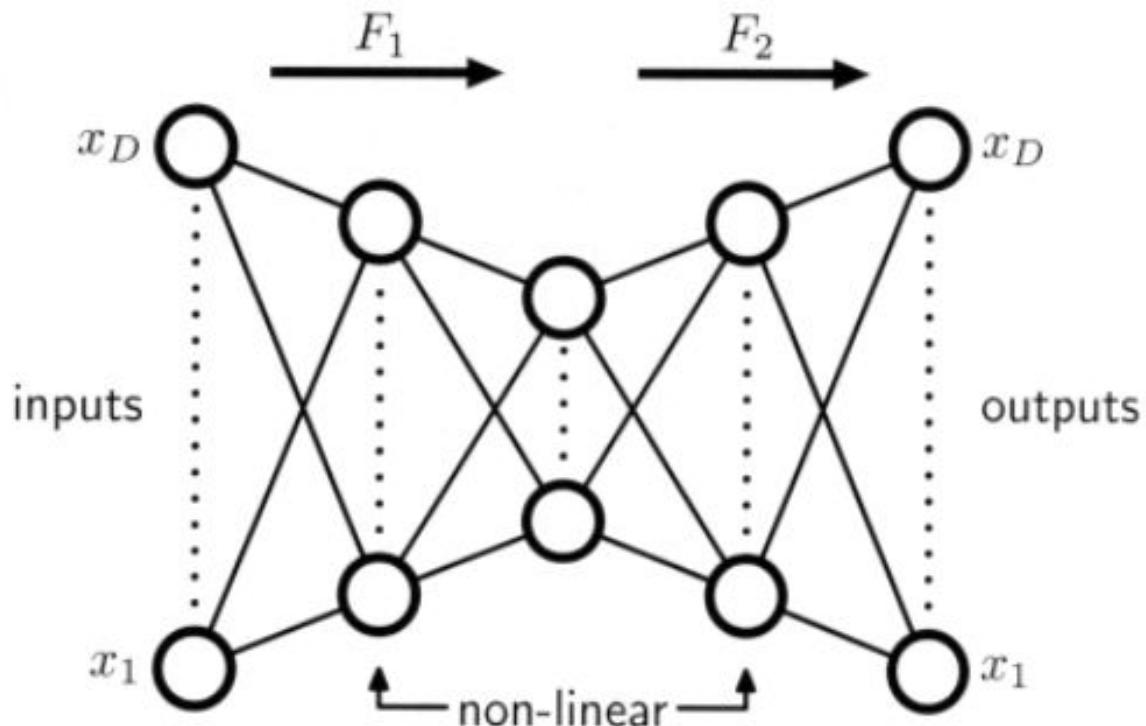
$x$   
two  
layer

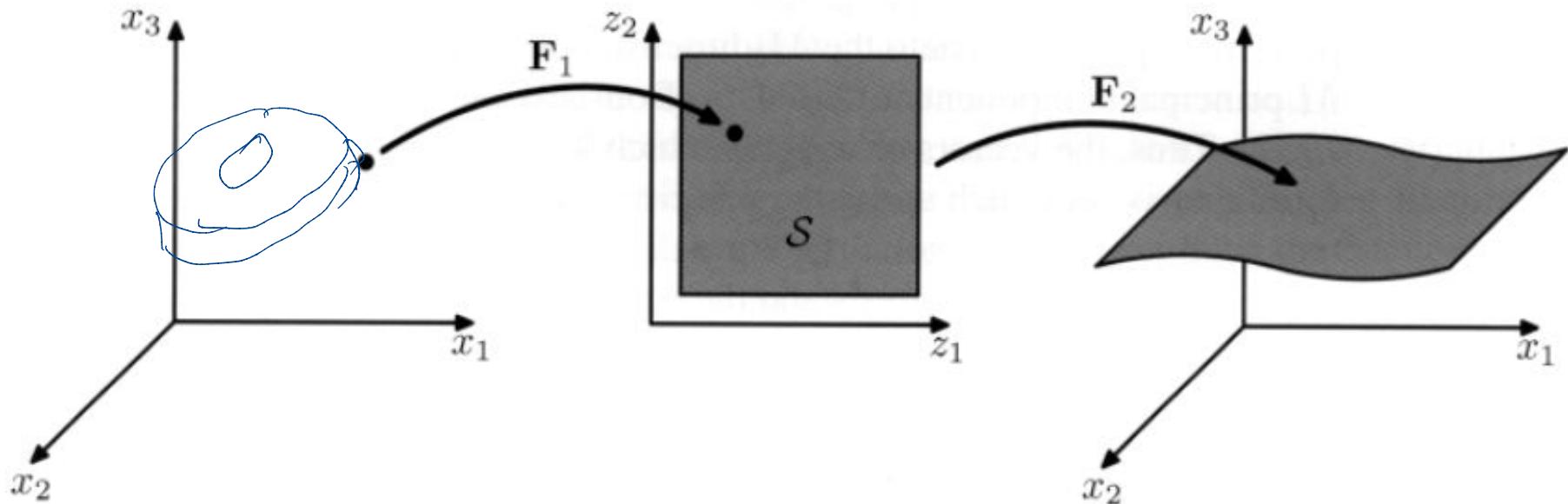
$z$

$\rightarrow$

$x'$

**Figure 12.19** Addition of extra hidden layers of **nonlinear units** gives an autoassociative network which can perform a nonlinear dimensionality reduction.





**Figure 12.20** Geometrical interpretation of the mappings performed by the network in Figure 12.19 for the case of  $D = 3$  inputs and  $M = 2$  units in the middle hidden layer. The function  $F_2$  maps from an  $M$ -dimensional space  $S$  into a  $D$ -dimensional space and therefore defines the way in which the space  $S$  is embedded within the original  $x$ -space. Since the mapping  $F_2$  can be nonlinear, the embedding of  $S$  can be nonplanar, as indicated in the figure. The mapping  $F_1$  then defines a projection of points in the original  $D$ -dimensional space into the  $M$ -dimensional subspace  $S$ .

# Easier to represent with more layers

An old result:

- functions that can be compactly represented by a depth  $k$  architecture might require an exponential number of computational elements to be represented by a depth  $k - 1$  architecture
- Example, the  $d$  bit parity function

$$\text{parity} : (b_1, \dots, b_d) \in \{0, 1\}^d \mapsto \begin{cases} 1 & \text{if } \sum_{i=1}^d b_i \text{ is even} \\ 0 & \text{otherwise} \end{cases}$$

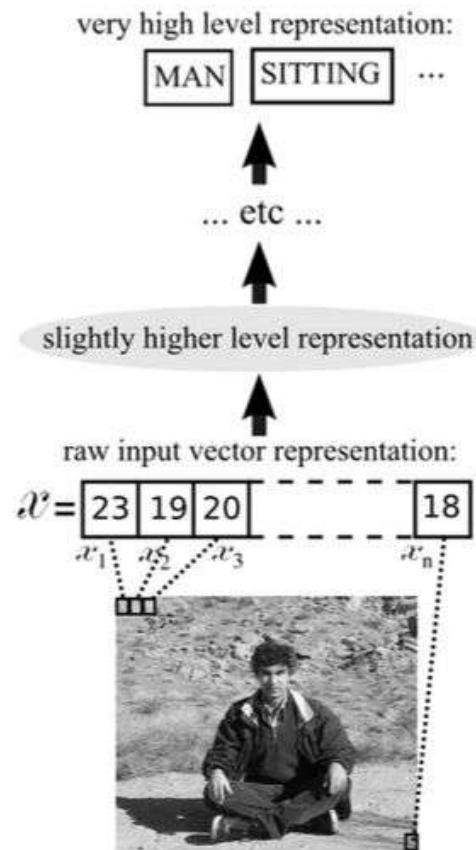
- **Theorem:**  $d$ -bit parity circuits of depth 2 have exponential size

“On the Expressive Power of Deep Architectures”,  
Bengio and Delalleau, 2011

Yao, A. (1985). *Separating the polynomial-time hierarchy by oracles*. In Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science, pages 1–10. *FOCS*

Håstad, J. (1986). *Almost optimal lower bounds for small depth circuits*. In Proceedings of the 18th annual ACM Symposium on Theory of Computing, pages 6–20, Berkeley, California.

## *Deep representation - intuition*

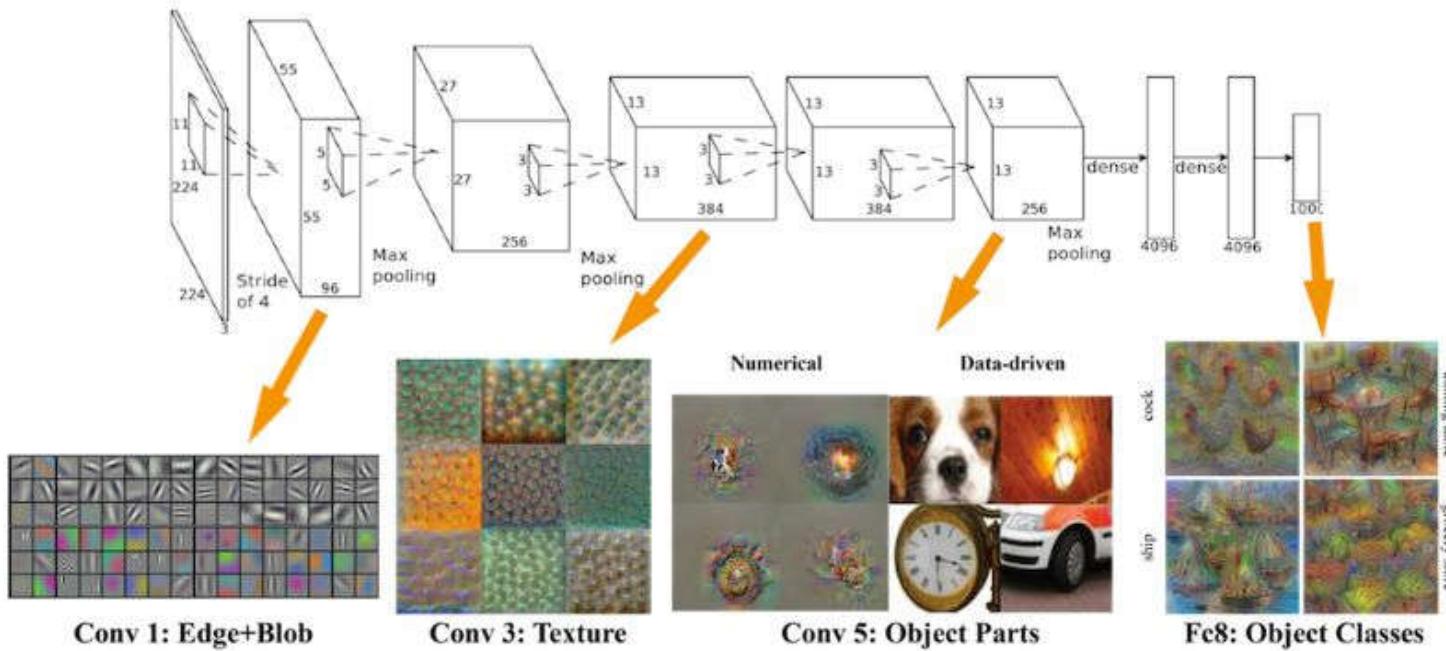


Bengio, "Learning Deep Architectures for AI", 2009

## Challenges of “just add more layers”

- Deep architectures get stuck in local minima or plateaus
- As architecture gets deeper, more difficult to obtain good generalisation
- Hard to initialise random weights well
- 1 or 2 hidden layers seem to perform better
- 2006: Unsupervised pre-training, find distributed representation

## Deep representation - practice

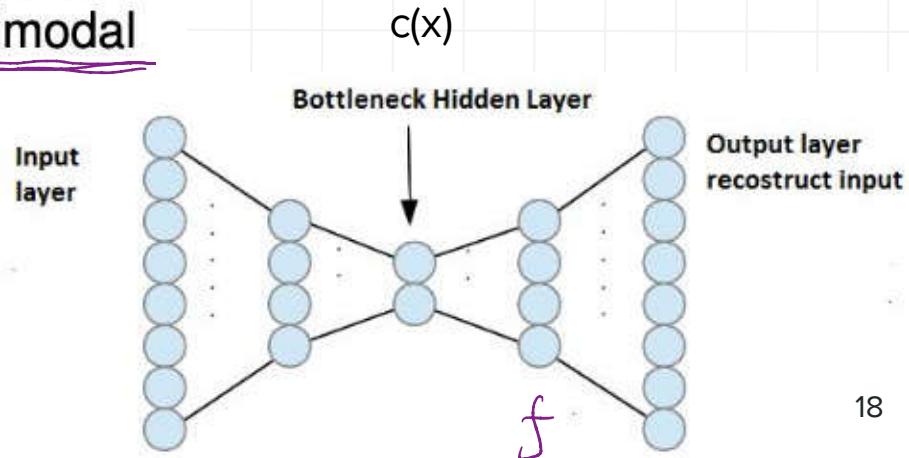


AlexNet / VGG-F network visualized by mNeuron.

## Autoencoder

- An autoencoder is trained to encode the input  $x$  into some representation  $c(x)$  so that the input can be reconstructed from that representation
- the target output of the autoencoder is the autoencoder input itself
- With one linear hidden layer and the mean squared error criterion, the  $k$  hidden units learn to project the input in the span of the first  $k$  principal components of the data
- If the hidden layer is nonlinear, the autoencoder behaves differently from PCA, with the ability to capture multimodal aspects of the input distribution
- Let  $f$  be the decoder. We want to minimise the reconstruction error

$$\sum_{n=1}^N \ell(x_n, f(c(x_n)))$$



# Cost function of an autoencoder

- Recall:  $f(c(x))$  is the reconstruction produced by the network
- Minimisation of the negative log likelihood of the reconstruction, given the encoding  $c(x)$

$$\text{RE} = -\log P(x|c(x))$$

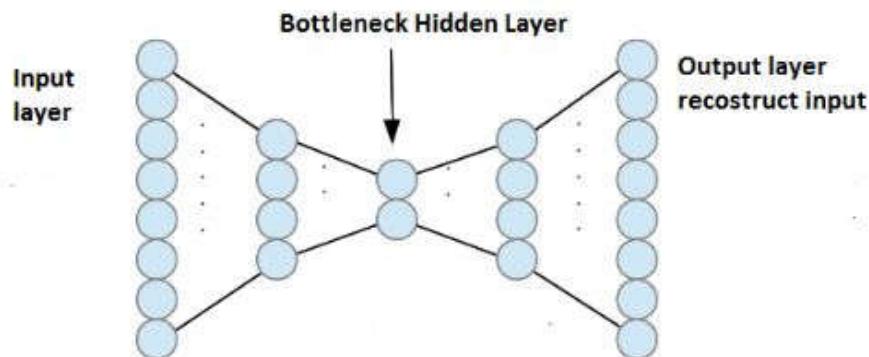
- If  $x|c(x)$  is Gaussian, we recover the familiar squared error
- If the inputs  $x_i$  are either binary or considered to be binomial probabilities, then the loss function would be the cross entropy

$$-\log P(x|c(x)) = -x_i \log f_i(c(x)) + (1 - x_i) \log(1 - f_i(c(x)))$$

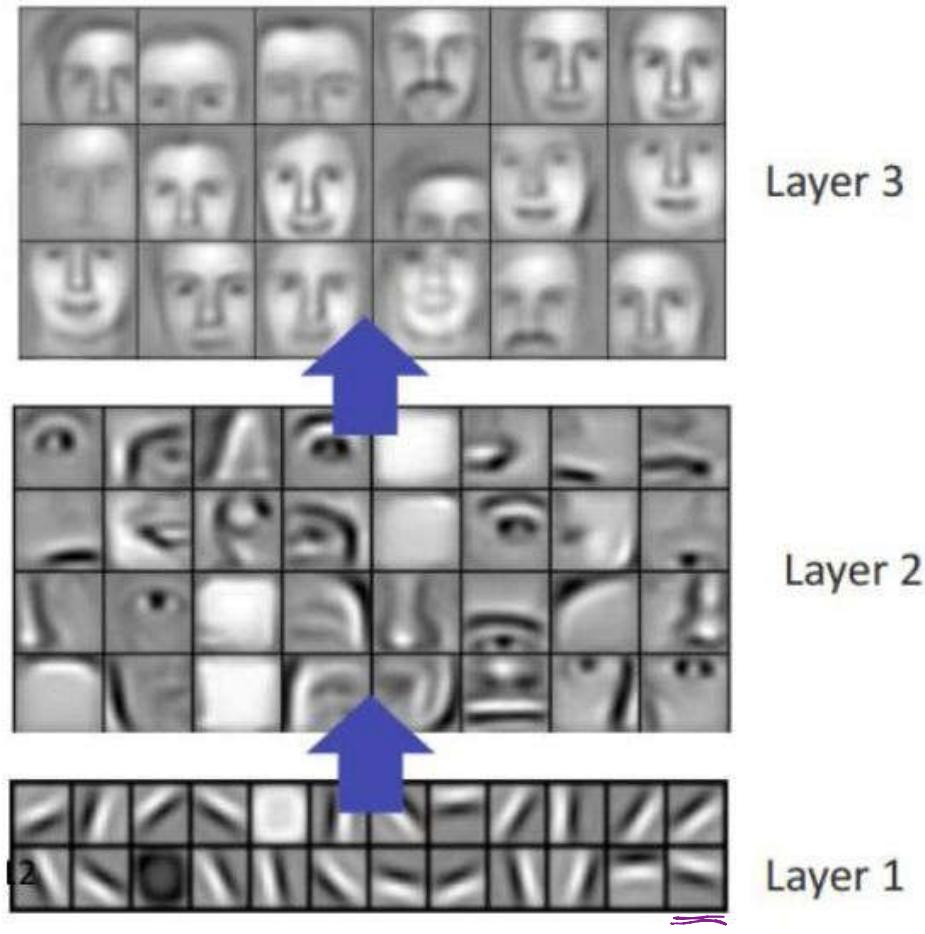
where  $f_i(\cdot)$  is the  $i^{\text{th}}$  component of the decoder

## *Stacking autoencoders*

- Let  $c_j$  and  $f_j$  be the encoder and corresponding decoder of the  $j^{\text{th}}$  layer
- Let  $z_j$  be the representation after the encoder  $c_j$
- We can define multiple layers of autoencoders recursively.
- For example, let  $z_1 = c_1(x)$ , and  $z_2 = c_2(z_1)$ ,  
the corresponding decoding is given by  $\underline{f_1(f_2(z_2))}$
- Because of non-linear activation functions, the latent feature  $z_2$  can capture more complex patterns than  $z_1$ .



## *Higher level image features - faces*



## *Pre-training supervised neural networks*

- Latent features  $z_j$  in layer  $j$  can capture high level patterns

$$z_j = c_j(c_{j-1}(\cdots c_2(c_1(x)) \cdots))$$

- These features may also be useful for supervised learning tasks.
- In contrast to the feed forward network, the features  $z_j$  are constructed in an unsupervised fashion.
- Discard the decoding layers, and directly use  $z_j$  with a supervised training method, such as logistic regression.
- Various such pre-trained networks are available on-line, e.g **VGG-19**.

• Task transfer.

CV  
parup  
...

• self-supervision,

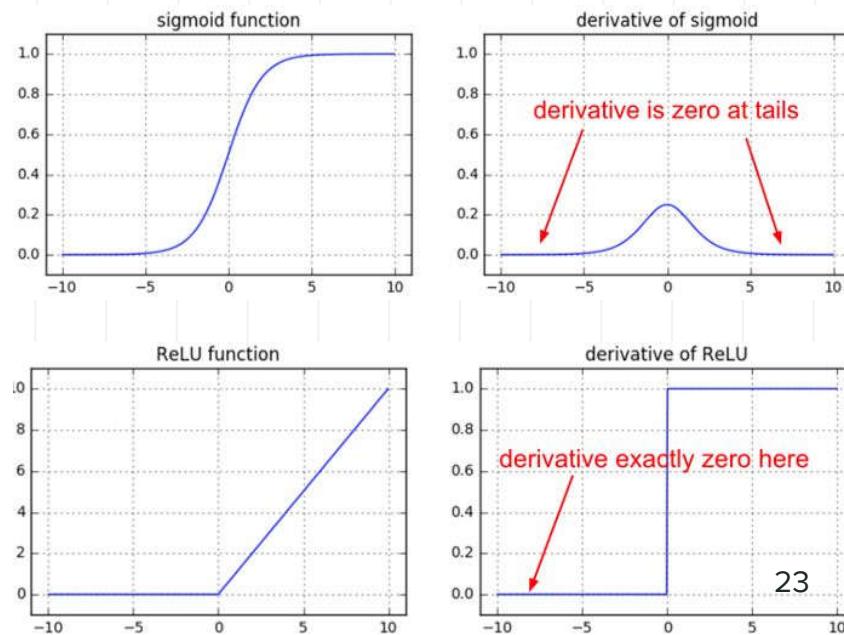
## Xavier Initialisation / ReLU

- Layer-wise unsupervised pre-training helps by extracting useful features for subsequent supervised backprop.
- Pre-training also avoids **saturation** (large magnitude arguments to, say, sigmoidal units).
- Simpler **Xavier initialization** can also avoid saturation.
- Let the inputs  $x_i \sim \mathcal{N}(0, 1)$ , weights  $w_i \sim \mathcal{N}(0, \sigma^2)$  and activation  $z = \sum_{i=1}^m x_i w_i$ . Then:

$$\begin{aligned}\text{VAR}[z] &= \mathbb{E}[(z - \mathbb{E}[z])^2] = \mathbb{E}[z^2] = \mathbb{E}\left[\left(\sum_{i=1}^m x_i w_i\right)^2\right] \\ &= \sum_{i=1}^m \mathbb{E}[(x_i w_i)^2] = \sum_{i=1}^m \mathbb{E}[x_i^2] \mathbb{E}[w_i^2] = m\sigma^2.\end{aligned}$$

- So we set  $\sigma = 1/\sqrt{m}$  to have “nice” activations.
- **Glorot initialization** takes care to have nice back-propagated signals — see the auto-encoder lab.
- **ReLU** activations  $h(x) = \max(x, 0)$  also help in practice.

Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." In AISTATS 2010 pp. 249-256.



# Linear and Nonlinear Component Analysis

Probabilistic PCA

Autoencoders

Autoencoders for image processing

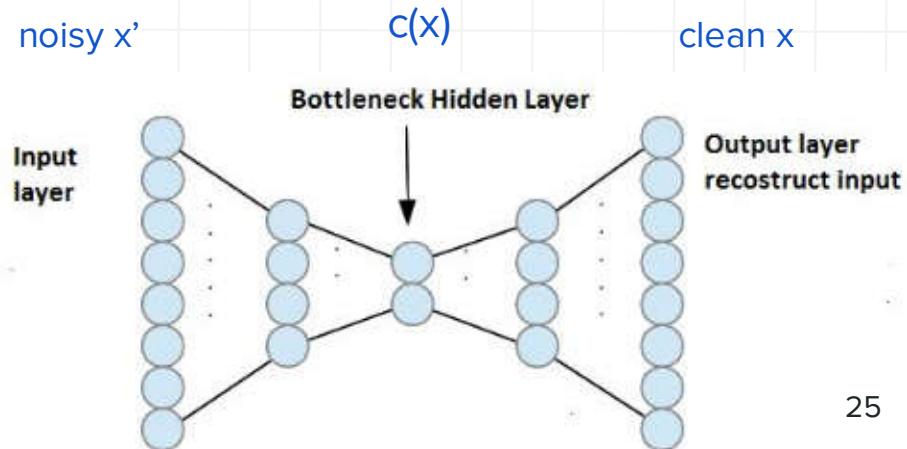
Recommender systems

## Denoising autoencoder

- Add noise to input, keeping perfect example as output
- Autoencoder tries to:
  - ➊ preserve information of input
  - ➋ undo stochastic corruption process
- Reconstruction log likelihood

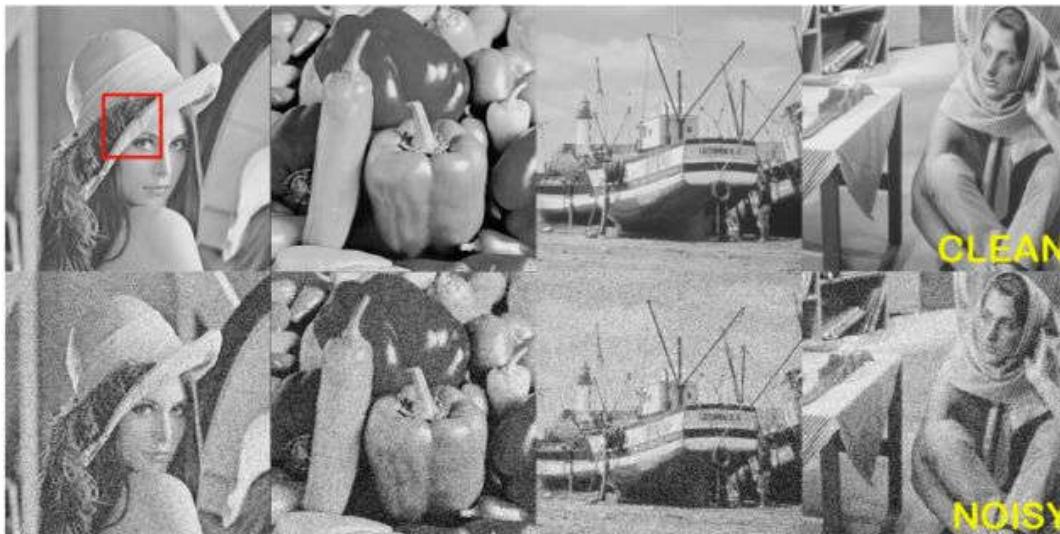
$$-\log P(x|c(\hat{x}))$$

where  $x$  noise free,  $\hat{x}$  corrupted



## Image denoising

- Images with Gaussian noise added.



- Denoised using Stacked Sparse Denoising Autoencoder



Xie, Junyuan, Linli Xu, and Enhong Chen. "Image denoising and inpainting with deep neural networks." *Advances in neural information processing systems* 25 (2012): 341-349.

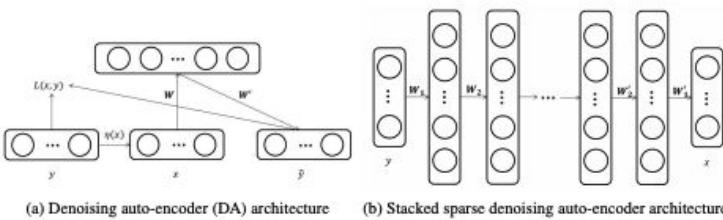


Figure 1: Model architectures.

## 2.1 Problem Formulation

Assuming  $\mathbf{x}$  is the observed noisy image and  $\mathbf{y}$  is the original noise free image, we can formulate the image corruption process as:

$$\mathbf{x} = \eta(\mathbf{y}). \quad (1)$$

where  $\eta : \mathbb{R}^n \rightarrow \mathbb{R}^n$  is an arbitrary stochastic corrupting process that corrupts the input. Then, the denoising task's learning objective becomes:

$$f = \operatorname{argmin}_f \mathbf{E}_{\mathbf{y}} \|f(\mathbf{x}) - \mathbf{y}\|_2^2 \quad (2)$$

From this formulation, we can see that the task here is to find a function  $f$  that best approximates  $\eta^{-1}$ . We can now treat the image denoising and inpainting problems in a unified framework by choosing appropriate  $\eta$  in different situations.

## 2.2 Denoising Auto-encoder

Let  $\mathbf{y}_i$  be the original data for  $i = 1, 2, \dots, N$  and  $\mathbf{x}_i$  be the corrupted version of corresponding  $\mathbf{y}_i$ . DA is defined as shown in Fig.1a:

$$h(\mathbf{x}_i) = \sigma(\mathbf{W}\mathbf{x}_i + \mathbf{b}) \quad (3)$$

$$\hat{y}(\mathbf{x}_i) = \sigma(\mathbf{W}'h(\mathbf{x}_i) + \mathbf{b}') \quad (4)$$

where  $\sigma(x) = (1 + \exp(-x))^{-1}$  is the sigmoid activation function which is applied element-wise to vectors,  $h_i$  is the hidden layer activation,  $\hat{y}(\mathbf{x}_i)$  is an approximation of  $\mathbf{y}_i$  and  $\Theta = \{\mathbf{W}, \mathbf{b}, \mathbf{W}', \mathbf{b}'\}$  represents the weights and biases. DA can be trained with various optimization methods to minimize the reconstruction loss:

$$\theta = \operatorname{argmin}_{\theta} \sum_{i=1}^N \|\mathbf{y}_i - \hat{y}(\mathbf{x}_i)\|. \quad (5)$$

After finish training a DA, we can move on to training the next layer by using the hidden layer activation of the first layer as the input of the next layer. This is called Stacked denoising auto-encoder (SDA) [21].

## 2.3 Stacked Sparse Denoising Auto-encoders

In this section, we will describe the structure and optimization objective of the proposed model Stacked Sparse Denoising Auto-encoders (SSDA). Due to the fact that directly processing the entire image is intractable, we instead draw overlapping patches from the image as our data objects. In the training phase, the model is supplied with both the corrupted noisy image patches  $\mathbf{x}_i$ , for  $i = 1, 2, \dots, N$ , and the original patches  $\mathbf{y}_i$ . After training, SSDA will be able to reconstruct the corresponding clean image given any noisy observation.

To combine the virtues of sparse coding and neural networks and avoid over-fitting, we train a DA to minimize the reconstruction loss regularized by a sparsity-inducing term:

$$L_1(\mathbf{X}, \mathbf{Y}; \theta) = \frac{1}{N} \sum_{i=1}^N \frac{1}{2} \|\mathbf{y}_i - \hat{y}(\mathbf{x}_i)\|_2^2 + \beta \operatorname{KL}(\rho \parallel \rho) + \frac{\lambda}{2} (\|\mathbf{W}\|_F^2 + \|\mathbf{W}'\|_F^2) \quad (6)$$

Method	Standard deviation $\sigma$		
	25/PSNR=20.17	50/PSNR=14.16	100/PSNR=8.13
SSDA	$30.52 \pm 1.02$	$27.37 \pm 1.10$	$24.18 \pm 1.39$
BLS-GSM	$30.49 \pm 1.17$	$27.28 \pm 1.44$	$24.37 \pm 1.36$
KSVD	$30.96 \pm 0.77$	$27.34 \pm 1.11$	$23.50 \pm 1.15$

Table 1: Comparison of the denoising performance. Performance is measured by Peak Signal to Noise Ratio (PSNR). Results are averaged over testing set.

where

$$\operatorname{KL}(\hat{\rho} \parallel \rho) = \sum_{j=1}^{|\hat{\rho}|} \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{(1 - \rho)}{1 - \hat{\rho}_j}, \quad \hat{\rho} = \frac{1}{N} \sum_i^N h(\mathbf{x}_i).$$

and  $h(\cdot)$  and  $\hat{y}(\cdot)$  are defined in (3), (4) respectively. Here  $\hat{\rho}$  is the average activation of the hidden layer. We regularize the hidden layer representation to be sparse by choosing small  $\rho$  so that the KL-divergence term will encourage the mean activation of hidden units to be small. Hence the hidden units will be zero most of the time and achieve sparsity.

After training of the first DA, we use  $h(\mathbf{y}_i)$  and  $h(\mathbf{x}_i)$  as the clean and noisy input respectively for the second DA. This is different from the approach described in [21], where  $\mathbf{x}_i$  is discarded and  $\eta(h(\mathbf{y}_i))$  is used as the noisy input. We point out that our method is more natural in that, since  $h(\mathbf{y}_i)$  lies in a different space from  $\mathbf{y}_i$ , the meaning of applying  $\eta(\cdot)$  to  $h(\mathbf{y}_i)$  is not clear.

We then initialize a deep network with the weights obtained from  $K$  stacked DAs. The network has one input layer, one output and  $2K - 1$  hidden layers as shown in Fig.1b. The entire network is then trained using the standard back-propagation algorithm to minimize the following objective:

$$L_2(\mathbf{X}, \mathbf{Y}; \theta) = \frac{1}{N} \sum_{i=1}^N \frac{1}{2} \|\mathbf{y}_i - \hat{y}(\mathbf{x}_i)\|_2^2 + \frac{\lambda}{2} \sum_{j=1}^{2K} (\|\mathbf{W}_j\|_F^2). \quad (7)$$

Here we removed the sparsity regularization because the pre-trained weights will serve as regularization to the network [18].

In both of the pre-training and fine-tuning stages, the loss functions are optimized with L-BFGS algorithm (a Quasi-Newton method) which, according to [22], can achieve fastest convergence in our settings.

## 3 Experiments

We narrow our focus down to denoising and inpainting of grey-scale images, but there is no difficulty in generalizing to colored images. We use a set of natural images collected from the web<sup>1</sup> as our training set and standard testing images<sup>2</sup> as the testing set. We create noisy images from clean training and testing images by applying the function (1) to them. Image patches are then extracted from both clean and noisy images to train SSDAs. We employ Peak Signal to Noise Ratio (PSNR) to quantify denoising results:  $10 \log_{10}(255^2 / \sigma_n^2)$ , where  $\sigma_n^2$  is the mean squared error. PSNR is one of the standard indicators used for evaluating image denoising results.

### 3.1 Denoising White Gaussian Noise

We first corrupt images with additive white Gaussian noise of various standard deviations. For the proposed method, one SSDA model is trained for each noise level. We evaluate different hyper-parameter combinations and report the best result. We set  $K$  to 2 for all cases because adding more layers may slightly improve the performance but require much more training time. In the meantime, we try different patch sizes and find that higher noise level generally requires larger patch size. The

<sup>1</sup><http://decsai.ugr.es/cvg/dbimagenes/>

<sup>2</sup>Widely used images commonly referred to as Lena, Barbara, Boat, Pepper, etc. in the image processing community.

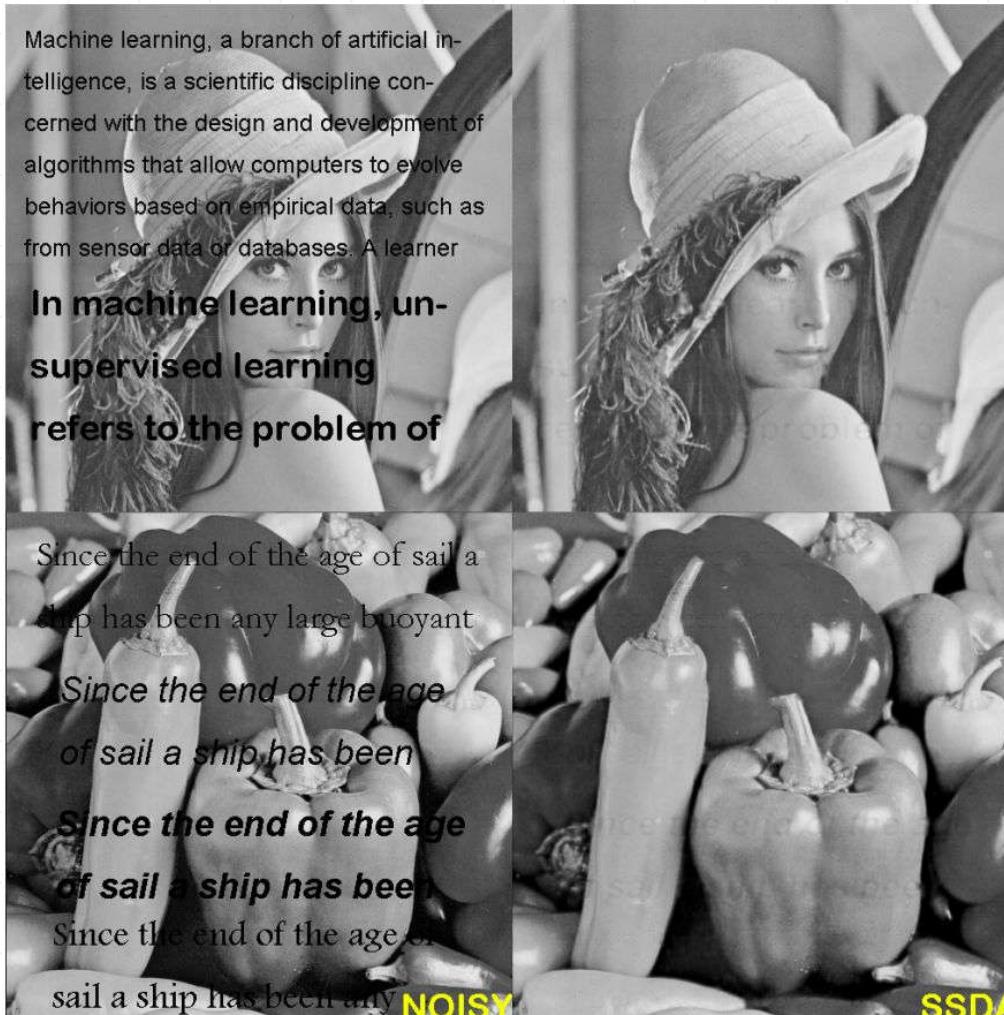
Xie, Junyuan, Linli Xu, and Enhong Chen. "Image denoising and inpainting with deep neural networks." *Advances in neural information processing systems 25* (2012): 341-349.

# Resizing an image

<https://cimg.eu/greycstoration/demonstration.shtml>



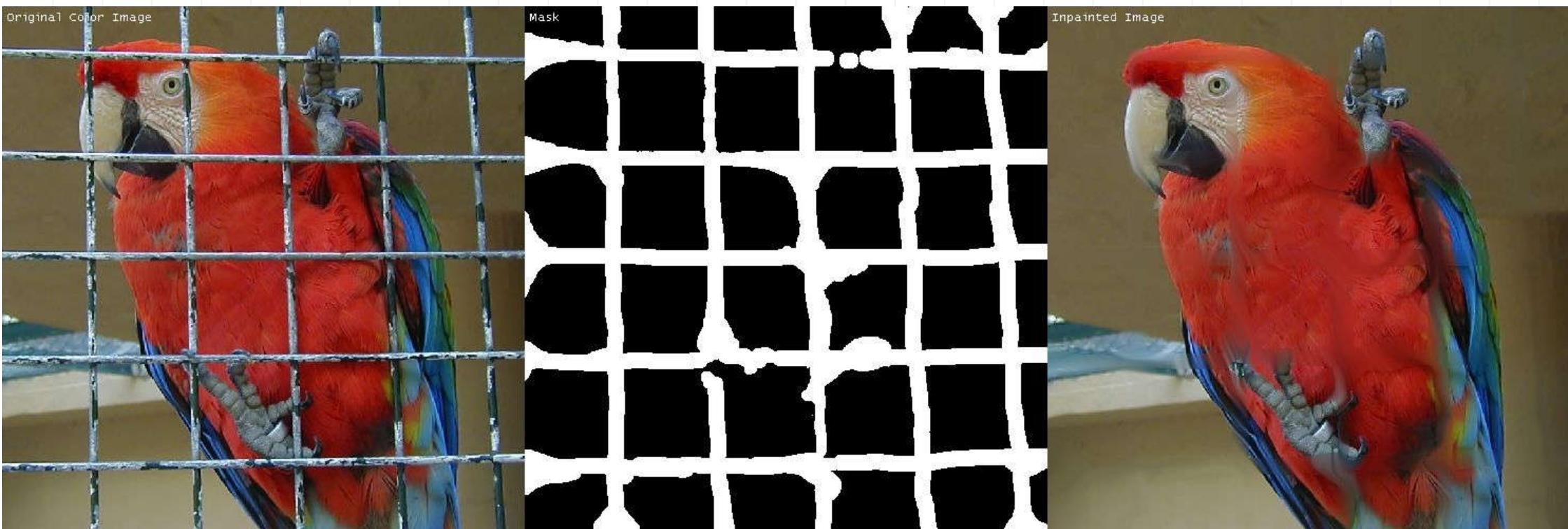
# Image inpainting - undo text over image



Xie, Junyuan, Linli Xu, and Enhong Chen. "Image denoising and inpainting with deep neural networks." *Advances in neural information processing systems* 25 (2012): 341-349.

# Image inpainting - free a bird

<https://cimg.eu/greycstoration/demonstration.shtml>



# Linear and nonlinear component analysis

Probabilistic PCA

Autoencoders

Applications in Image processing -- denoising, upscaling, inpainting

Recommender systems

# Relational data and recommender systems

items

users	1	2
1		
3		
5	1	4
4	1	5

users

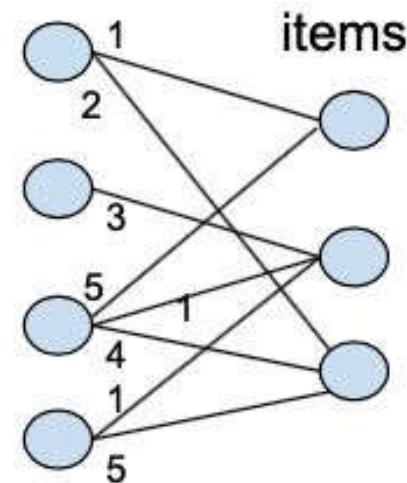


Figure 22.1: Example of a relational dataset represented as a sparse matrix (left) or a sparse bipartite graph (right). Values corresponding to empty cells (missing edges) are unknown. Rows 3 and 4 are similar to each other, indicating that users 3 and 4 might have similar preferences, so we can use the data from user 3 to predict user 4's preferences. However, user 1 seems quite different in their preferences, and seems to give low ratings to all items. For user 2, we have very little observed data, so it is hard to make reliable predictions.

	MovieID	Title	Genres
36	858	Godfather, The (1972)	Action Crime Drama
35	1387	Jaws (1975)	Action Horror
65	2028	Saving Private Ryan (1998)	Action Drama War
63	1221	Godfather: Part II, The (1974)	Action Crime Drama
11	913	Maltese Falcon, The (1941)	Film-Noir Mystery
20	3417	Crimson Pirate, The (1952)	Adventure Comedy Sci-Fi
34	2186	Strangers on a Train (1951)	Film-Noir Thriller
55	2791	Airplane! (1980)	Comedy
31	1188	Strictly Ballroom (1992)	Comedy Romance
28	1304	Butch Cassidy and the Sundance Kid (1969)	Action Comedy Western

(a)

	MovieID	Title	Genres
516	527	Schindler's List (1993)	Drama War
1848	1953	French Connection, The (1971)	Action Crime Drama Thriller
596	608	Fargo (1996)	Crime Drama Thriller
1235	1284	Big Sleep, The (1946)	Film-Noir Mystery
2085	2194	Untouchables, The (1987)	Action Crime Drama
1188	1230	Annie Hall (1977)	Comedy Romance
1198	1242	Glory (1989)	Action Drama War
897	922	Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)	Film-Noir
1849	1954	Rocky (1976)	Action Drama
581	593	Silence of the Lambs, The (1991)	Drama Thriller

(b)

Figure 22.4: (a) Top 10 movies (from a list of 69) that user “837” has already highly rated. (b) Top 10 predictions (from a list of 3637) from the algorithm. Generated by code at [figures.probml.ai/book1/22.4](http://figures.probml.ai/book1/22.4).

# Collaborative filtering

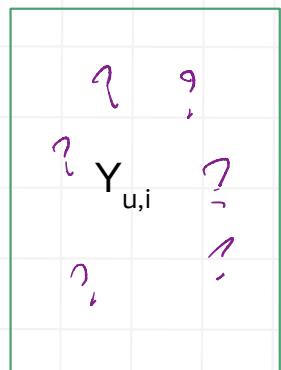
Netflix prize dataset: 480K+ users, rating 17.8K movies, 100M ratings total

Newer public datasets:

MovieLens 1 M - 6040 users, 3760 movies, 1M ratings

MovieLens 10 M - ~70k users, 10K movies, ~10M ratings

User  $u$       observed rating  $y_{u,i}$   
item  $i$       estimate unknown rating  $\hat{y}_{u,i}$



$$\hat{Y}_{ui} = \sum_{u': Y_{u',i} \neq ?} \underbrace{\text{sim}(u, u')}_{\downarrow} \underbrace{Y_{u',i}}_{\text{Can come from user profile info.}} \quad (22.1)$$

Can come from user profile info.  
Can come from other user profile info.

## Matrix completion / matrix factorisation

Loss function

$$\mathcal{L}(\mathbf{Z}) = \sum_{ij: Y_{ij} \neq ?} (Z_{ij} - Y_{ij})^2 = \|\mathbf{Z} - \mathbf{Y}\|_F^2$$

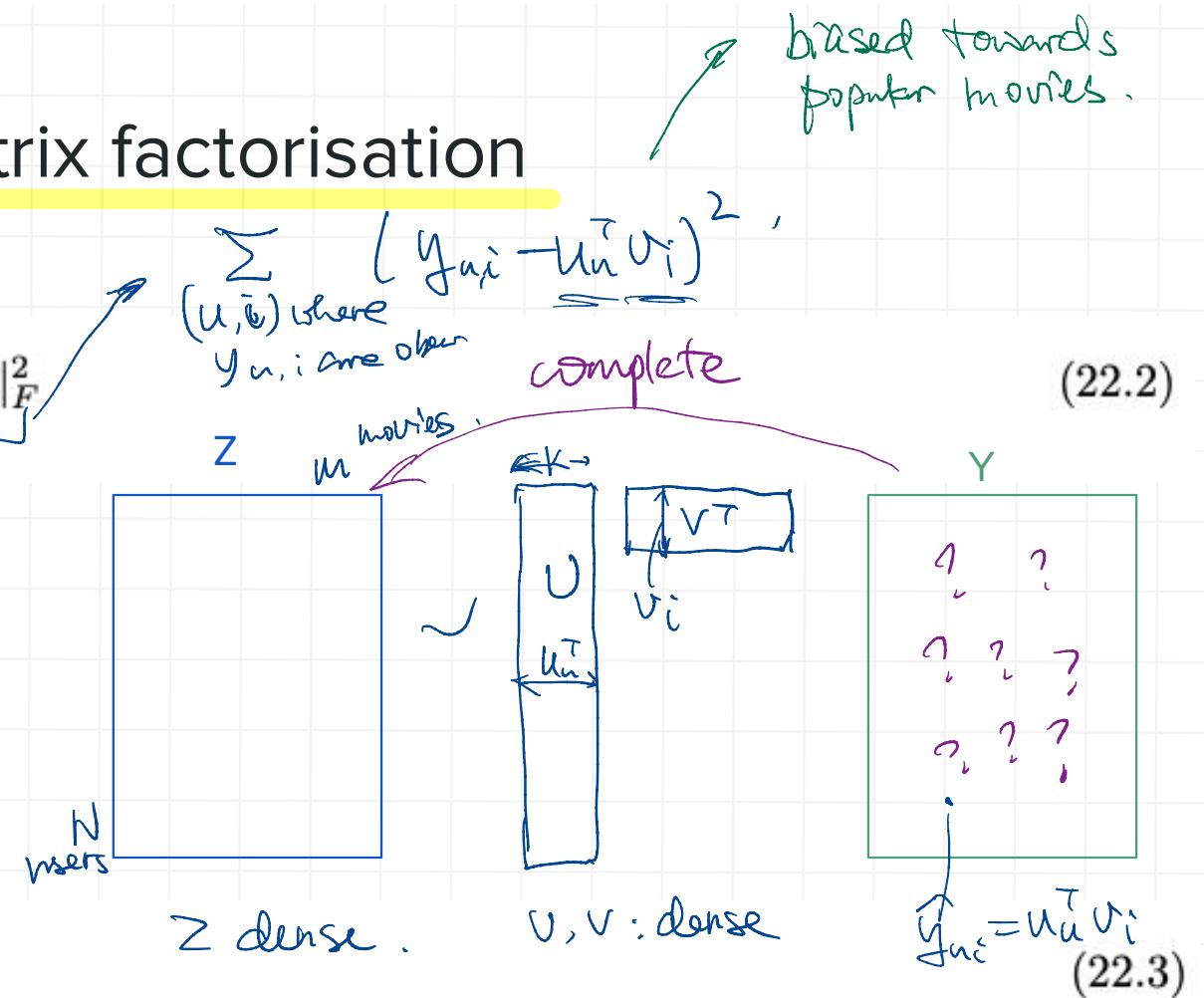
(22.2)

“model”

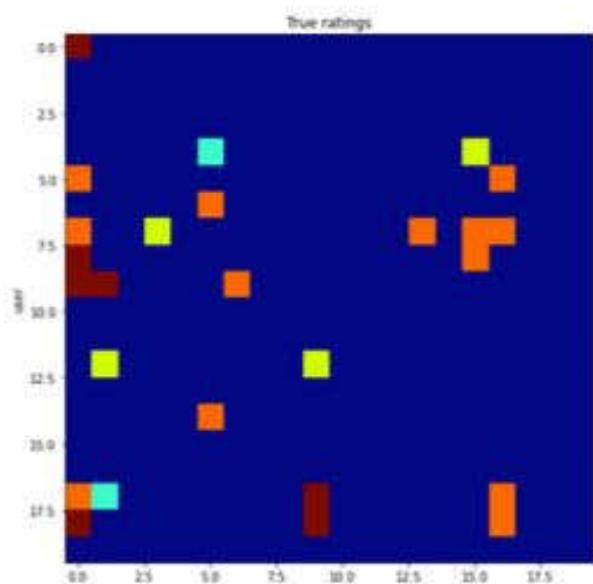
$$\mathbf{Z} = \mathbf{U}\mathbf{V}^\top \approx \mathbf{Y}$$

prediction

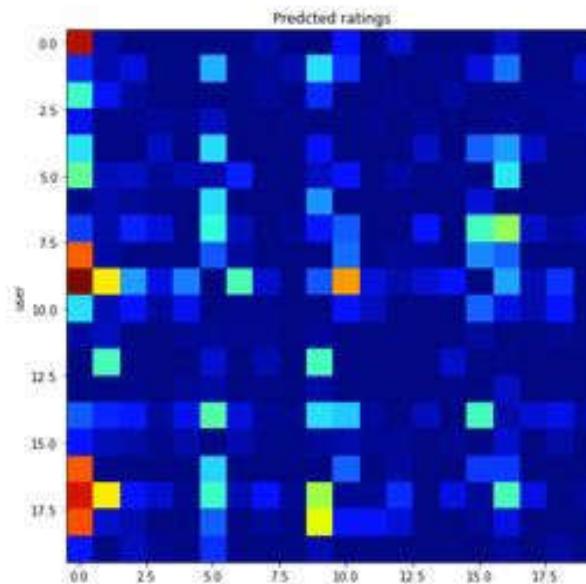
$$\hat{y}_{ui} = \mathbf{u}_u^\top \mathbf{v}_i$$



Optimisation: alternating least squares, or stochastic gradient descent (SGD)



(a)



(b)

Figure 22.3: (a) A fragment of the observed ratings matrix from the MovieLens-1M dataset. (b) Predictions using SVD with 50 latent components. Generated by code at [figures.probml.ai/book1/22.3](http://figures.probml.ai/book1/22.3).

$$\begin{array}{c} \text{fact' 1} \\ \text{factor} \rightarrow \boxed{\sqrt{\gamma}} \end{array}$$

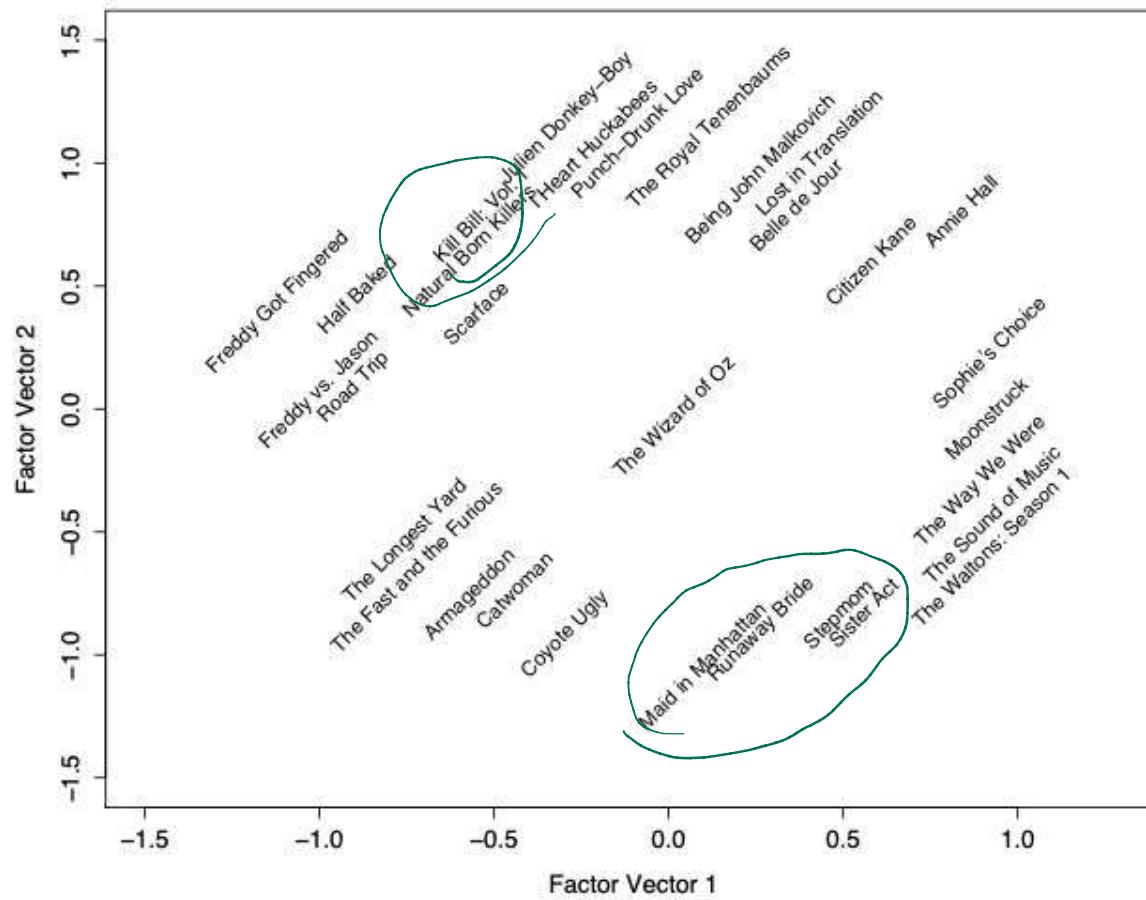


Figure 22.2: Visualization of the first two latent movie factors estimated from the Netflix challenge data. Each movie  $j$  is plotted at the location specified by  $\mathbf{v}_j$ . See text for details. From Figure 3 of [KBV09]. Used with kind permission of Yehuda Koren.

# Autoencoders for recommendation

“model”

prediction

$$f(\mathbf{y}_{:,i}; \boldsymbol{\theta}) = \mathbf{W}^\top \varphi(\mathbf{V}\mathbf{y}_{:,i} + \boldsymbol{\mu}) + \mathbf{b}$$



Loss function

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^N \sum_{u:y_{ui} \neq ?} (y_{ui} - f(\mathbf{y}_{:,i}; \boldsymbol{\theta})_u)^2 + \frac{\lambda}{2} (\|\mathbf{W}\|_F^2 + \|\mathbf{V}\|_F^2)$$

square loss

L2 regularizer

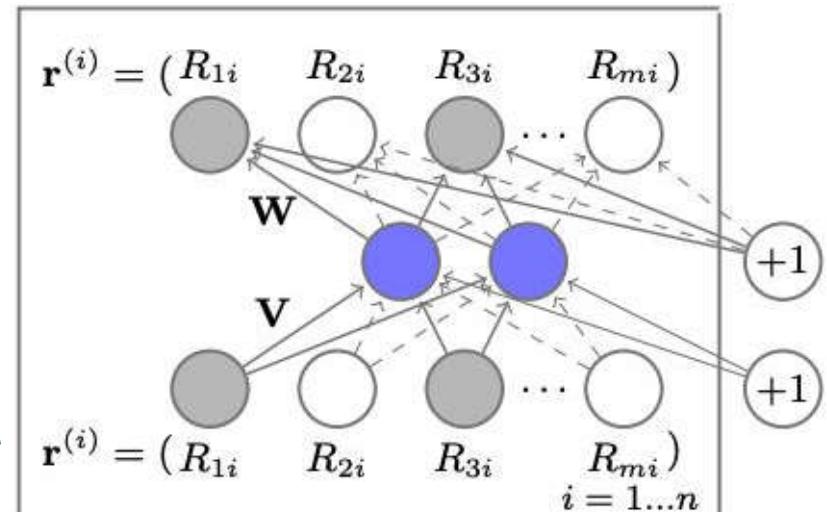


Figure 1: Item-based AutoRec model. We use plate notation to indicate that there are  $n$  copies of the neural network (one for each item), where  $\mathbf{W}$  and  $\mathbf{V}$  are tied across all copies.

# AutoRec: Autoencoders Meet Collaborative Filtering

Suvash Sedhain<sup>†\*</sup>, Aditya Krishna Menon<sup>†\*</sup>, Scott Sanner<sup>†\*</sup>, Lexing Xie<sup>‡\*</sup>

<sup>†</sup> NICTA, <sup>‡</sup> Australian National University

suvash.sedhain@anu.edu.au, {aditya.menon, scott.sanner}@nicta.com.au,  
lexing.xie@anu.edu.au

## ABSTRACT

This paper proposes AutoRec, a novel autoencoder framework for collaborative filtering (CF). Empirically, AutoRec's compact and efficiently trainable model outperforms state-of-the-art CF techniques (biased matrix factorization, RBM-CF and LLORMA) on the MovieLens and Netflix datasets.

**Categories and Subject Descriptors** D.2.8 [Information Storage and Retrieval]Information Filtering

**Keywords** Recommender Systems; Collaborative Filtering; Autoencoders

## 1. INTRODUCTION

Collaborative filtering (CF) models aim to exploit information about users' preferences for items (e.g. star ratings) to provide personalised recommendations. Owing to the Netflix challenge, a panoply of different CF models have been proposed, with popular choices being matrix factorisation [1, 2] and neighbourhood models [5]. This paper proposes *AutoRec*, a new CF model based on the autoencoder paradigm; our interest in this paradigm stems from the recent successes of (deep) neural network models for vision and speech tasks. We argue that AutoRec has representational and computational advantages over existing neural approaches to CF [4], and demonstrate empirically that it outperforms the current state-of-the-art methods.

## 2. THE AUTOREC MODEL

In rating-based collaborative filtering, we have  $m$  users,  $n$  items, and a partially observed user-item rating matrix  $R \in \mathbb{R}^{m \times n}$ . Each user  $u \in U = \{1 \dots m\}$  can be represented by a partially observed vector  $\mathbf{r}^{(u)} = (R_{u1}, \dots, R_{un}) \in \mathbb{R}^n$ . Similarly, each item  $i \in I = \{1 \dots n\}$  can be represented by a partially observed vector  $\mathbf{r}^{(i)} = (R_{1i}, \dots, R_{ni}) \in \mathbb{R}^m$ . Our aim in this work is to design an item-based (user-based) autoencoder which can take as input each partially observed  $\mathbf{r}^{(u)}$  ( $\mathbf{r}^{(i)}$ ), project it into a low-dimensional latent (hidden) space, and then reconstruct  $\mathbf{r}^{(i)}$  ( $\mathbf{r}^{(u)}$ ) in the output space to predict missing ratings for purposes of recommendation.

Formally, given a set  $S$  of vectors in  $\mathbb{R}^d$ , and some  $k \in \mathbb{N}_+$ , an autoencoder solves

$$\min_{\theta} \sum_{r \in S} \|\mathbf{r} - h(\mathbf{r}; \theta)\|_2^2, \quad (1)$$

Copyright is held by the author/owner(s).  
WWW 2015 Companion, May 18–22, 2015, Florence, Italy.  
ACM 978-1-4503-3473-0/15/05.  
<http://dx.doi.org/10.1145/2740908.2742726>.

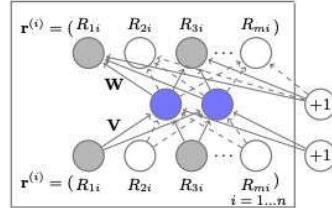


Figure 1: Item-based AutoRec model. We use plate notation to indicate that there are  $n$  copies of the neural network (one for each item), where  $\mathbf{W}$  and  $\mathbf{V}$  are tied across all copies.

where  $h(\mathbf{r}; \theta)$  is the *reconstruction* of input  $\mathbf{r} \in \mathbb{R}^d$ ,

$$h(\mathbf{r}; \theta) = f(\mathbf{W} \cdot g(\mathbf{V}\mathbf{r} + \boldsymbol{\mu}) + \mathbf{b})$$

for activation functions  $f(\cdot), g(\cdot)$ . Here,  $\theta = \{\mathbf{W}, \mathbf{V}, \boldsymbol{\mu}, \mathbf{b}\}$  for transformations  $\mathbf{W} \in \mathbb{R}^{d \times k}, \mathbf{V} \in \mathbb{R}^{k \times d}$ , and biases  $\boldsymbol{\mu} \in \mathbb{R}^k, \mathbf{b} \in \mathbb{R}^d$ . This objective corresponds to an auto-associative neural network with a single,  $k$ -dimensional hidden layer. The parameters  $\theta$  are learned using backpropagation.

The item-based AutoRec model, shown in Figure 1, applies an autoencoder as per Equation 1 to the set of vectors  $\{\mathbf{r}^{(i)}\}_{i=1}^n$ , with two important changes. First, we account for the fact that each  $\mathbf{r}^{(i)}$  is partially observed by only updating during backpropagation those weights that are associated with observed inputs, as is common in matrix factorisation and RBM approaches. Second, we regularise the learned parameters so as to prevent overfitting on the observed ratings. Formally, the objective function for the Item-based AutoRec (I-AutoRec) model is, for regularisation strength  $\lambda > 0$ ,

$$\min_{\theta} \sum_{i=1}^n \|\mathbf{r}^{(i)} - h(\mathbf{r}^{(i)}; \theta)\|_2^2 + \frac{\lambda}{2} \cdot (\|\mathbf{W}\|_F^2 + \|\mathbf{V}\|_F^2), \quad (2)$$

where  $\|\cdot\|_2^2$  means that we only consider the contribution of observed ratings. User-based AutoRec (U-AutoRec) is derived by working with  $\{\mathbf{r}^{(u)}\}_{u=1}^m$ . In total, I-AutoRec requires the estimation of  $2mk + m + k$  parameters. Given learned parameters  $\hat{\theta}$ , I-AutoRec's predicted rating for user  $u$  and item  $i$  is

$$\hat{R}_{ui} = (h(\mathbf{r}^{(i)}; \hat{\theta}))_u. \quad (3)$$

Figure 1 illustrates the model, with shaded nodes corresponding to observed ratings, and solid connections corresponding to weights that are updated for the input  $\mathbf{r}^{(i)}$ .

	ML-1M	ML-10M	$f(\cdot)$	$g(\cdot)$	RMSE	ML-1M	ML-10M	Netflix	
U-RBM	0.881	0.823	Identity	Identity	0.872	BiasedMF	0.845	0.803	0.844
I-RBM	0.854	0.825	Sigmoid	Identity	0.852	I-RBM	0.854	0.825	-
U-AutoRec	0.874	0.867	Identity	Sigmoid	<b>0.831</b>	U-RBM	0.881	0.823	0.845
I-AutoRec	<b>0.831</b>	<b>0.782</b>	Sigmoid	Sigmoid	0.836	LLORMA	0.833	<b>0.782</b>	0.834
			(a)		(b)	I-AutoRec	<b>0.831</b>	<b>0.782</b>	<b>0.823</b>

Table 1: (a) Comparison of the RMSE of I/U-AutoRec and RBM models. (b) RMSE for I-AutoRec with choices of linear and nonlinear activation functions, MovieLens 1M dataset. (c) Comparison of I-AutoRec with baselines on MovieLens and Netflix datasets. We remark that I-RBM did not converge after one week of training. LLORMA's performance is taken from [2].

AutoRec is distinct to existing CF approaches. Compared to the RBM-based CF model (RBM-CF) [4], there are several differences. First, RBM-CF proposes a generative, probabilistic model based on restricted Boltzmann machines, while AutoRec is a discriminative model based on autoencoders. Second, RBM-CF estimates parameters by maximising log likelihood, while AutoRec directly minimises RMSE, the canonical performance in rating prediction tasks. Third, training RBM-CF requires the use of contrastive divergence, whereas training AutoRec requires the comparatively faster gradient-based backpropagation. Finally, RBM-CF is only applicable for discrete ratings, and estimates a separate set of parameters for each rating value. For  $r$  possible ratings, this implies  $nkr$  or  $(nk)r$  parameters for user- (item-) based RBM. AutoRec is agnostic to  $r$  and hence requires fewer parameters. Fewer parameters enable AutoRec to have less memory footprint and less prone to overfitting. Compared to matrix factorisation (MF) approaches, which embed both users and items into a shared latent space, the item-based AutoRec model only embeds items into latent space. Further, while MF learns a linear latent representation, AutoRec can learn a *nonlinear* latent representation through activation function  $g(\cdot)$ .

## 3. EXPERIMENTAL EVALUATION

In this section, we evaluate and compare AutoRec with RBM-CF [4], Biased Matrix Factorisation [1] (BiasedMF), and Local Low-Rank Matrix Factorisation (LLORMA) [2] on the MovieLens 1M, 10M and Netflix datasets. Following [2], we use a default rating of 3 for test users or items without training observations. We split the data into random 90%-10% train-test sets, and hold out 10% of the training set for hyperparameter tuning. We repeat this splitting procedure 5 times and report average RMSE. 95% confidence intervals on RMSE were  $\pm 0.003$  or less in each experiment. For all baselines, we tuned the regularisation strength  $\lambda \in \{0.001, 0.01, 0.1, 1, 100, 1000\}$  and the appropriate latent dimension  $k \in \{10, 20, 40, 80, 100, 200, 300, 400, 500\}$ .

A challenge training autoencoders is non-convexity of the objective. We found resilient propagation (RProp) [3] to give comparable performance to L-BFGS, while being much faster. Thus, we use RProp for all subsequent experiments:

**Which is better, item- or user-based autoencoding with RBMs or AutoRec?** Table 1a shows item-based (I-) methods for RBM and AutoRec generally perform better; this is likely since the average number of ratings per item is much more than those per user; high variance in the number of user ratings leads to less reliable prediction for user-based methods. I-AutoRec outperforms all RBM variants.

**How does AutoRec performance vary with linear and nonlinear activation functions  $f(\cdot), g(\cdot)$ ?** Table 1b indicates that nonlinearity in the hidden layer (via  $g(\cdot)$ ) is critical for good performance of I-AutoRec, indicating its

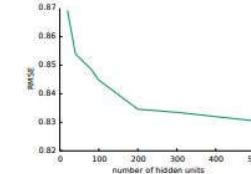


Figure 2: RMSE of I-AutoRec on MovieLens 1M as the number of hidden units  $k$  varies.

potential advantage over MF methods. Replacing sigmoids with Rectified Linear Units (ReLU) performed worse. All other AutoRec experiments use identity  $f(\cdot)$  and sigmoid  $g(\cdot)$  functions.

**How does performance of AutoRec vary with the number of hidden units?** In Figure 2, we evaluate the performance of AutoRec model as the number of hidden units varies. We note that performance steadily increases with the number of hidden units, but with diminishing returns. All other AutoRec experiments use  $k = 500$ .

**How does AutoRec perform against all baselines?** Table 1c shows that AutoRec consistently outperforms all baselines, except for comparable results with LLORMA on MovieLens 10M. Competitive performance with LLORMA is of interest, as the latter involves *weighting 50 different local matrix factorization models*, whereas AutoRec only uses a single latent representation via a neural net autoencoder.

**Do deep extensions of AutoRec help?** We developed a deep version of I-AutoRec with three hidden layers of (500, 250, 500) units, each with a sigmoid activation. We used greedy pretraining and then fine-tuned by gradient descent. On MovieLens 1M, RMSE reduces from 0.831 to 0.827 indicating potential for further improvement via deep AutoRec.

**Acknowledgments** NICTA is funded by the Australian Government as represented by the Dept. of Communications and the ARC through the ICT Centre of Excellence program. This research was supported in part by ARC DP140102185.

## References

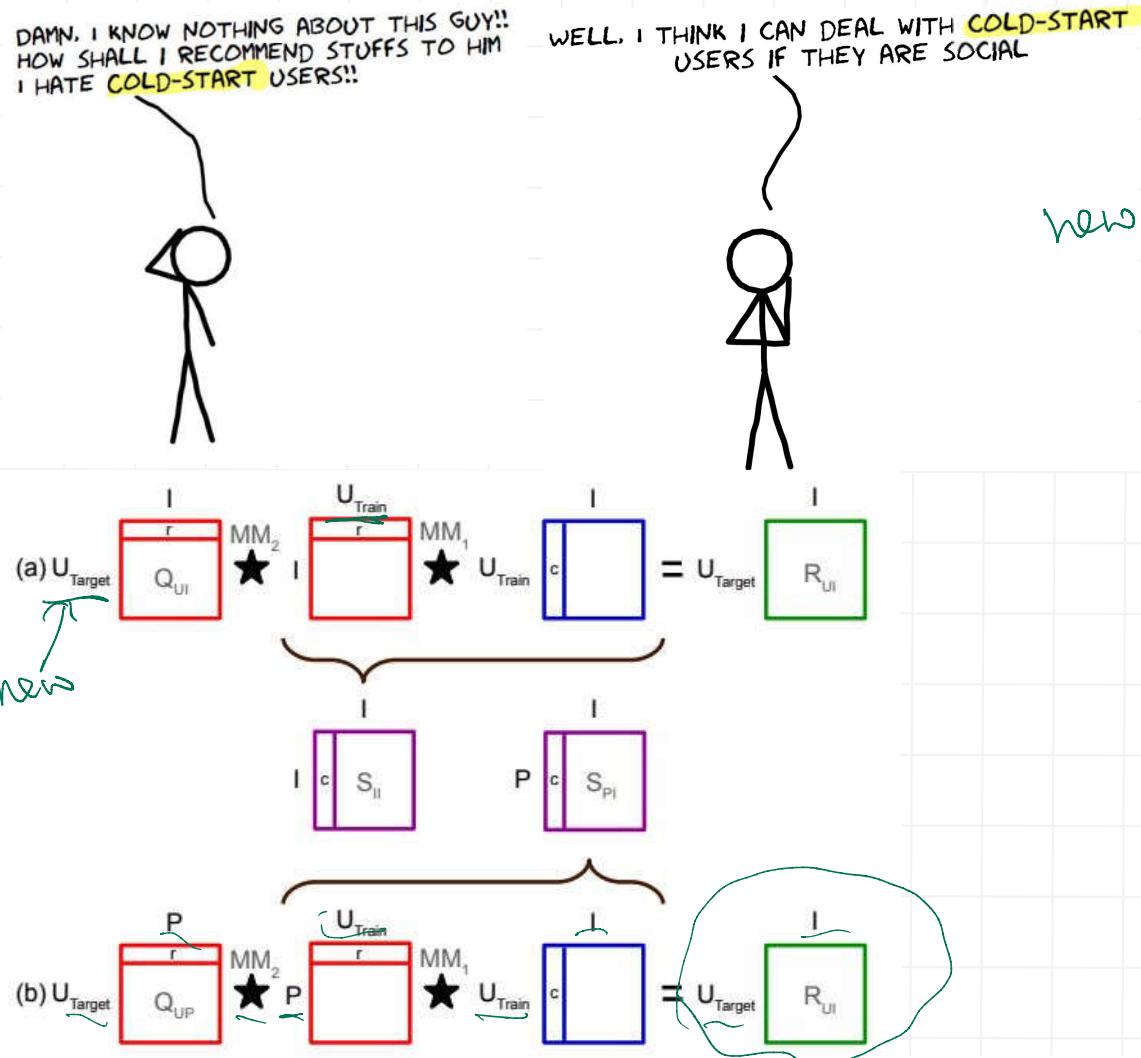
- [1] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42, 2009.
- [2] J. Lee, S. Kim, G. Lebanon, and Y. Singer. Local low-rank matrix approximation. In *ICML*, 2013.
- [3] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: the rprop algorithm. In *IEEE International Conference on Neural Networks*, 1993.
- [4] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted Boltzmann machines for collaborative filtering. In *ICML*, 2007.
- [5] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.

A fireside chat with Dr Suvash Sedhain and Dr Xi Yang from Twitter Engineering



<https://cecs.anu.edu.au/events/event-series/anu-computing-leadership-seminar-series>

# Social recommendation



Social Collaborative Filtering for Cold-start Recommendations. Sedhain, Suvash, Sanner, Scott, Braziunas, Darius, and Xie, Lexing, Recsys 2014

[credit: Suvash Sedhain for poster comic]

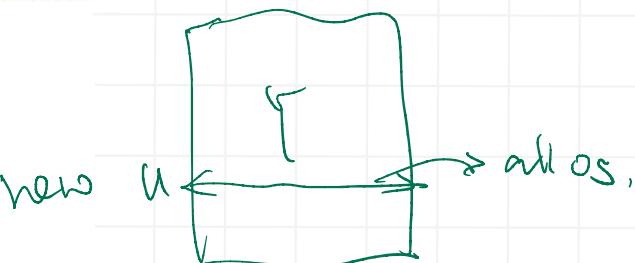


Figure 1: (a) A matrix algebra view of standard item-item neighborhood-based recommendation and (b) a variation for social cold-start recommendation that instead uses page-item similarity and does not require item purchase information for target users.  $U$  represents the user dimension (subdivided into train and test users),  $I$  the item dimension, and  $P$  the user's personal information dimension (demographics, friends, or page likes). The  $\star$  operators annotated by  $MM_1$  and  $MM_2$  denote *generalized matrix multiplication*, permitting *any* similarity metric over two vectors in place of the usual inner product.

## References

- Yoshua Bengio, "Learning Deep Architectures for AI", Foundations and Trends in Machine Learning, 2009
- <http://deeplearning.net/tutorial/>
- <http://www.deeplearningbook.org/contents/autoencoders.html>
- Fuchs, "On Sparse Representations in Arbitrary Redundant Bases", IEEE Trans. Info. Theory, 2004
- Xavier Glorot and Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks", 2010.

# Linear and Nonlinear Component Analysis

Probabilistic PCA

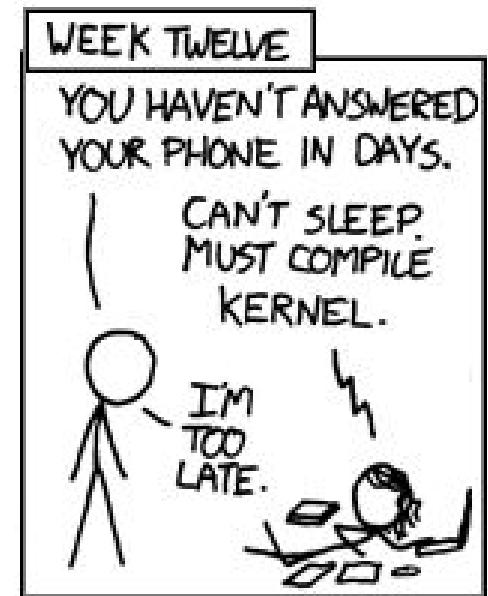
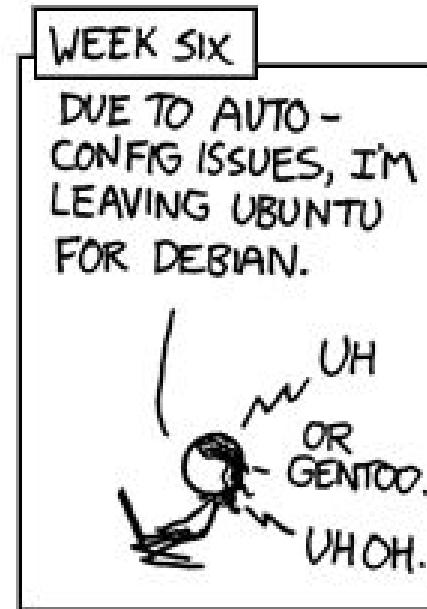
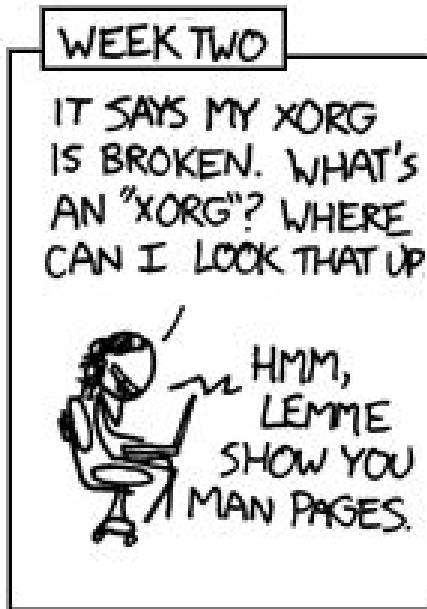
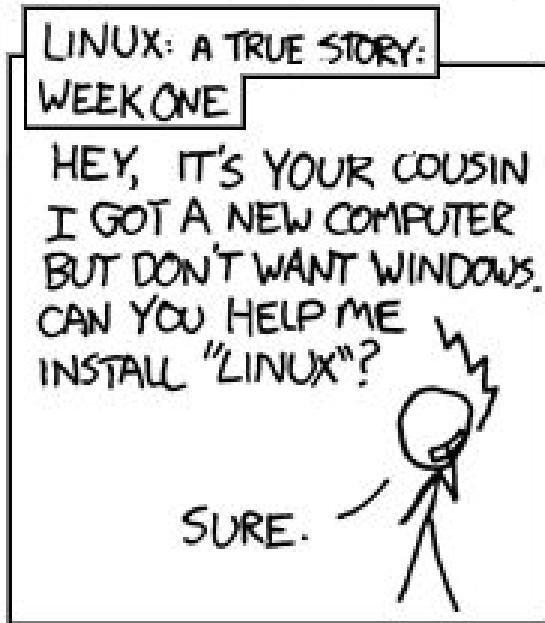
Autoencoders

Autoencoders for image processing

Recommender systems

<https://xkcd.com/456/>

Not the same kind of kernels ...



PARENTS: TALK TO YOUR  
KIDS ABOUT LINUX...  
BEFORE SOMEBODY ELSE DOES.

# Announcements

From the course reps – Taylor Qin and  
Yuchen Fang    also see piazza @229

Education info session: FAQ on admin  
your degree

After the break:

Week 7 Monday - Easter; Week 8 Monday - ANZAC day: no lecture or tutorial, join another online tutorial the same week.

**Topic: How to Admin Your Degree**

**Date:** Friday 1 April 10am

**Venue:** Rm 1.36 Birch Building (35) or

<https://anu.zoom.us/j/88641270633?pwd=ZU9paE1nUzJ0SU90c1krRnBJMWpGUT09>

**Facilitator:** Paul Dowden, Manager, Student Services, Employability & Experience

## About this event

There are a number of important elements CECS coursework students need to be aware of when managing their degree. The session will include:

1. Terminology (program/degree, major/minor; maximum/minimum etc.)
2. Building a degree:
  - Programs and Courses (P&C)
  - units of credit
  - Compulsory courses
  - Program List electives
  - 'Free' electives
  - Minors - Majors/Specialisations what are they and how and when to register
3. Program planning using P&C
4. What to do if it goes wrong!
  - Repeating a course
  - Reduced Study load
  - Program transfers incl. Credit & Exemption

# Kernels

Basis functions recap

Non-parametric methods (Chap 2.5)

Dual representation (of linear models, Chap 6.1)

Constructing kernels (Chap 6.2)

Next time: maximum-margin classifiers (Chap 7.1)

# Kernel

From Wikipedia, the free encyclopedia

*Not to be confused with Colonel.*

Kernel may refer to:

Look up *kernel* in  
Wiktionary, the free  
dictionary.

## Computing [edit]

- Kernel (operating system), the central component of most operating systems
- Kernel (image processing), a matrix used for image convolution
- Compute kernel, in GPGPU programming
- Kernel method, in machine learning
- In numerical analysis, a subroutine that performs a common numerical operation
  - In particular, a routine that is executed in a vectorized loop; see, e.g., General-purpose computing on graphics processing units § Kernels
- Kernelization, a technique for designing efficient algorithms

## Contents [hide]

- 1 Computing
- 2 Mathematics
  - 2.1 Objects
  - 2.2 Functions
- 3 Natural sciences
- 4 Other uses
- 5 People
- 6 See also

## Mathematics [edit]

### Objects [edit]

- Kernel (algebra), a general concept that includes:
  - Kernel (linear algebra) or null space, a set of vectors mapped to the zero vector
  - Kernel (category theory), a generalization of the kernel of a homomorphism
  - Kernel (set theory), an equivalence relation: partition by image under a function
  - Difference kernel, a binary equalizer: the kernel of the difference of two functions

### Functions [edit]

- Kernel (geometry), the set of points within a polygon from which the whole polygon boundary is visible
- Kernel (statistics), a weighting function used in kernel density estimation to estimate the probability density function of a random variable
- Integral kernel or kernel function, a function of two variables that defines an integral transform
- Heat kernel, the fundamental solution to the heat equation on a specified domain
- Convolution kernel
- Stochastic kernel, the transition function of a stochastic process
- Transition kernel, a generalization of a stochastic kernel
- Pricing kernel, the stochastic discount factor used in mathematical finance
- Positive-definite kernel, a generalization of a positive-definite matrix
- Kernel trick, in statistics
- Reproducing kernel Hilbert space

## Natural sciences [edit]

- Seed, inside the nut of most plants, especially:

# Recap: basis functions in linear models for classification

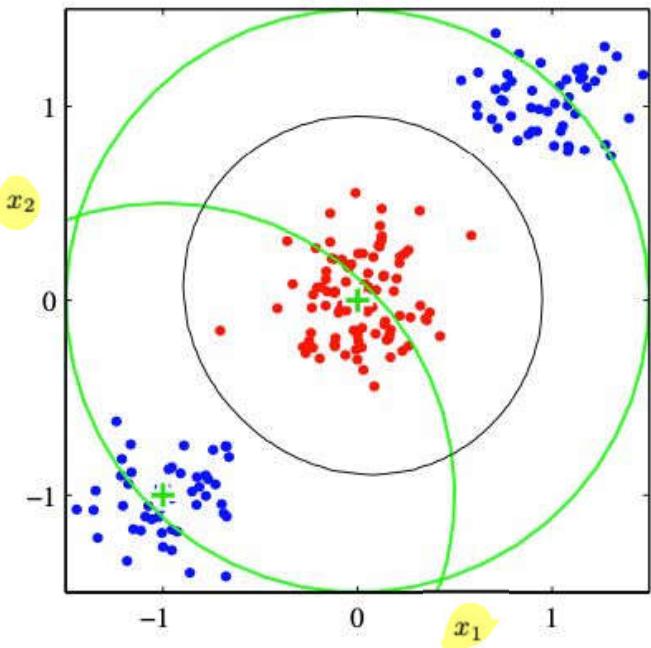
Linear w.r.t. Input  $x$

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 \quad (4.4)$$

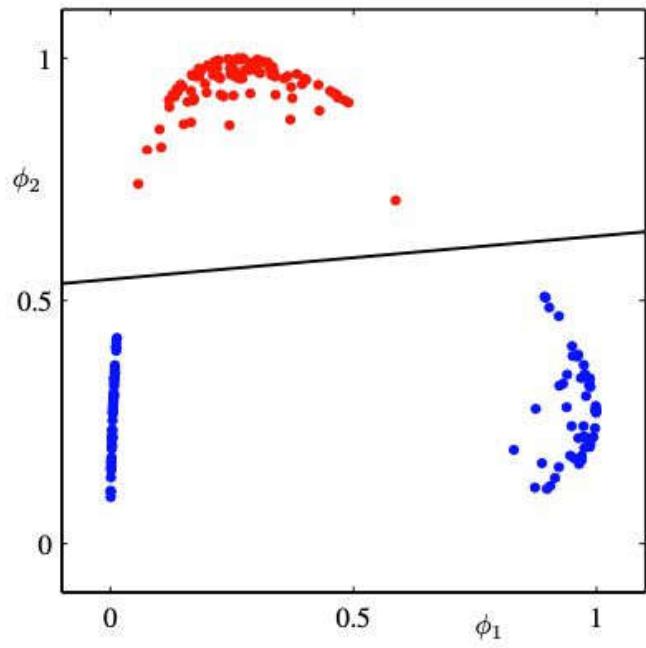
Linear w.r.t. basis  $\phi(x)$

$$p(\mathcal{C}_1|\phi) = y(\phi) = \sigma(\mathbf{w}^T \phi) \quad (4.87)$$

Fig 4.12



$$(x_1, x_2) \rightarrow (\phi_1(x_1, x_2), \phi_2(x_1, x_2))$$



## Recap: basis in linear models for regression

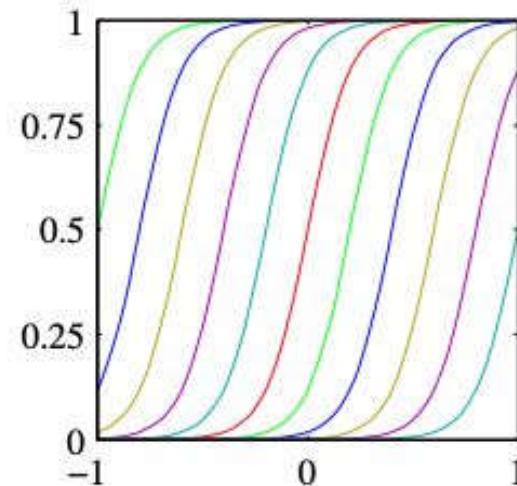
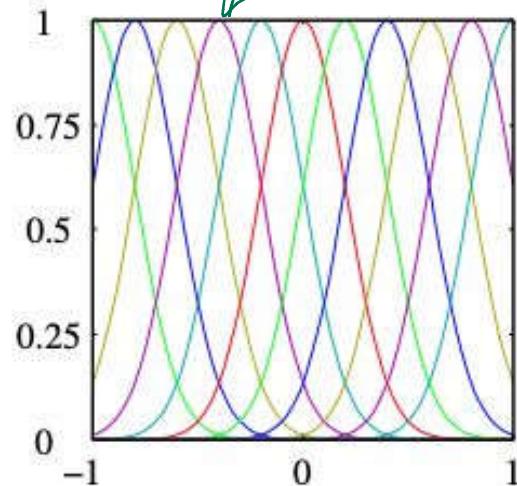
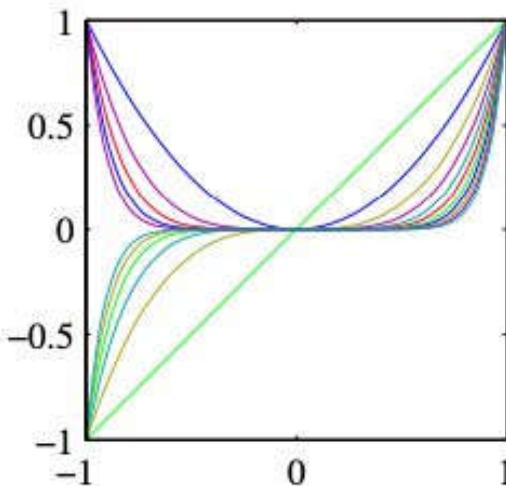
Linear w.r.t. input  $x$

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \dots + w_D x_D$$

Linear w.r.t. basis  $\phi(x)$

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$$

fixed width



**Figure 3.1** Examples of basis functions, showing polynomials on the left, Gaussians of the form (3.4) in the centre, and sigmoidal of the form (3.5) on the right.

# So far

Basis functions CAN:

- Represent non-linear decision boundaries in the input space.
- Separate classes linearly (in the feature space) for classes that are not linearly separable in the input space.

Basis functions CAN NOT:

- Remove class overlap that already exist in input space. ← 
- Adapt its own shape to data (but neural nets can, “flexible basis functions”)
- Adapt the number of features to data (this lecture and next lecture ... )

$$x \in \mathbb{R}^d$$

$$w \in \mathbb{R}^m$$

$$y \sim w^T \phi(x)$$

"parametric"

## Where are we going? Two key ideas

- Instead of "summarising" the training data into a set of weights (with fixed length), why not ask the features to adapt to data?
  - Continuity : Mostly targets don't change abruptly.
  - Similarity : Each training pair (input, target) tells us something about the possible targets in the neighbourhood of the input.

Kernels formalise these ideas

- Nonparametric methods: do not rely on a fixed number of parameters, but rather usually on storing the entire training set (various loose definitions are used here).

"no parameters"  
OR . unbounded # of "parameters"

## How to get there?

- 0) **Histograms**
- 1) Simple density estimation kernels (must only be non-negative integrable)
- 2) Implicit feature mapping kernels (must be positive definite)

### Warning:

- The term **kernel** is highly overloaded.
- Even today we consider two different types of kernel:
  - ➊ Smoothing kernel / density estimation kernel / Parzen window estimator kernel / Nadaraya-Watson kernel
  - ➋ Positive semidefinite kernel / reproducing kernel hilbert space kernel / implicit feature map kernel / support vector machine kernel /  $\approx$  Gaussian process covariance function or kernel

## Density estimation

Observe data points  $\{x_n\}_{n=1}^N$ , draw i.i.d. from  $p(x)$

Goal: estimate  $p(x)$  from data

**Figure 2.24** An illustration of the histogram approach to density estimation, in which a data set of 50 data points is generated from the distribution shown by the green curve. Histogram density estimates, based on (2.241), with a common bin width  $\Delta$  are shown for various values of  $\Delta$ .

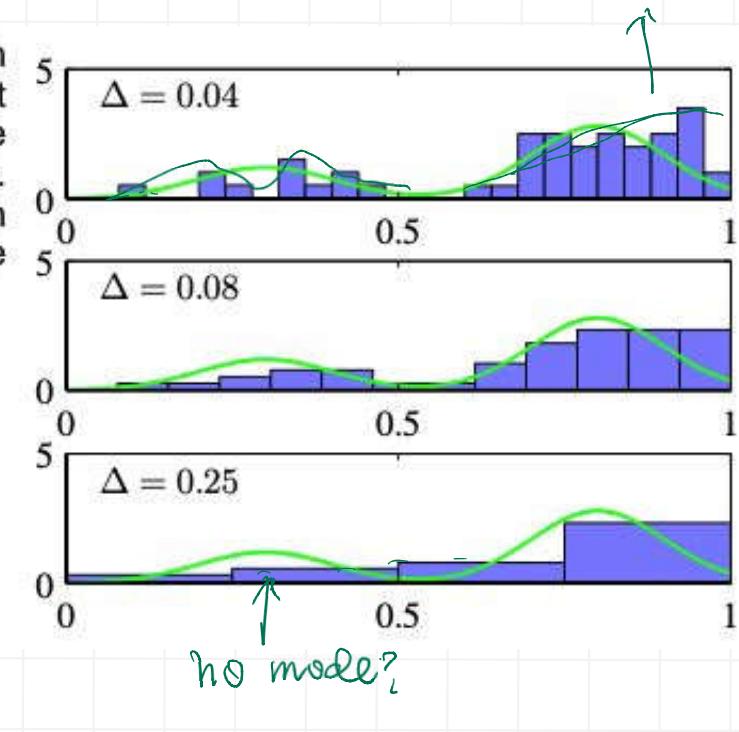
Histograms:

- Partition space into bins of width  $\Delta$
- Count the number of points in each bin
- Normalise

$$p_i = \frac{n_i}{N\Delta_i}$$

(2.241)

"Supervised ML"  $\{(X, y)\}_{n=1}^{\infty}$   
 $\{X_i\}_{i=1}^{\infty}$   
 $\hookrightarrow p(x)$



# Histograms as density estimators

## Pros

- Each data point is used once. Applicable to sequentially arriving data.
- Data points can be thrown away once the histogram is computed -- fixed storage cost given bin width.
- Good for quickly visualising densities in 1 or 2 dimensions

## Cons

- Depend on bin width  $\Delta$
- Has discontinuities due to bin edges
- Does not work well in  $>2$  dimensions - curse of dimensionality

what does a 2-dm histogram look like?

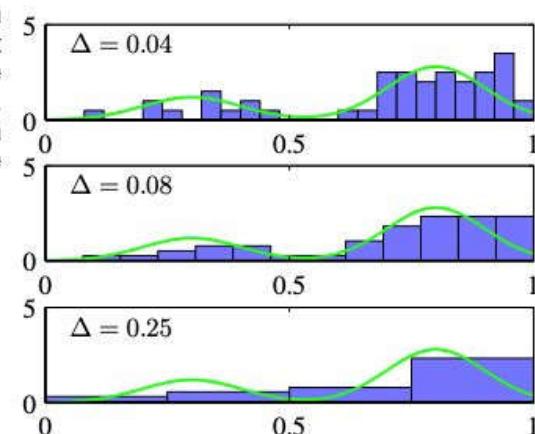
Implicit assumption: a distance measure in some local neighbourhood



Kernel density estimators

Nearest-neighbour methods

**Figure 2.24** An illustration of the histogram approach to density estimation, in which a data set of 50 data points is generated from the distribution shown by the green curve. Histogram density estimates, based on (2.241), with a common bin width  $\Delta$  are shown for various values of  $\Delta$ .



→ very few ~0 parameters

## Non-parametric density estimation

N data points drawn i.i.d. from unknown distribution  $p(x)$  in  $R^D$

Probability mass in a small region  $\mathcal{R}$

$$P = \int_{\mathcal{R}} p(\mathbf{x}) d\mathbf{x}. \quad (2.242)$$

Prob that K points fall within  $\mathcal{R}$

$$\text{Bin}(K|N, P) = \frac{N!}{K!(N-K)!} P^K (1-P)^{N-K}. \quad (2.243)$$

For large N

expected:  $K \simeq NP$  (2.244)

$P \simeq \underline{p(\mathbf{x})V}$  volume of  $\mathcal{R}$  (2.245)

$$p(\mathbf{x}) = \frac{K}{NV}. \quad (2.246)$$

→ Fix K - K-nearest neighbours

Fix V - kernel density estimation

density est  
classification

≠ K-means clustering

It can be shown that both the K-nearest-neighbour density estimator and the kernel density estimator converge to the true probability density in the limit  $N \rightarrow \infty$  provided  $V$  shrinks suitably with  $N$ , and  $K$  grows with  $N$  (Duda and Hart, 1973).

# Kernel density estimation (KDE), aka Parzen windows

$$p(\mathbf{x}) = \frac{K}{NV}. \quad (2.246)$$

Hypercubes

$$k(\mathbf{u}) = \begin{cases} 1, & |u_i| \leq 1/2, \\ 0, & \text{otherwise} \end{cases} \quad i = 1, \dots, D, \quad (2.247)$$

$$K = \sum_{n=1}^N k\left(\frac{\mathbf{x} - \mathbf{x}_n}{h}\right). \quad \xrightarrow{\text{how many points are } \pm \frac{1}{2}h \text{ within } x_n} \quad (2.248)$$

$$p(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^N \frac{1}{h^D} k\left(\frac{\mathbf{x} - \mathbf{x}_n}{h}\right) \quad (2.249)$$

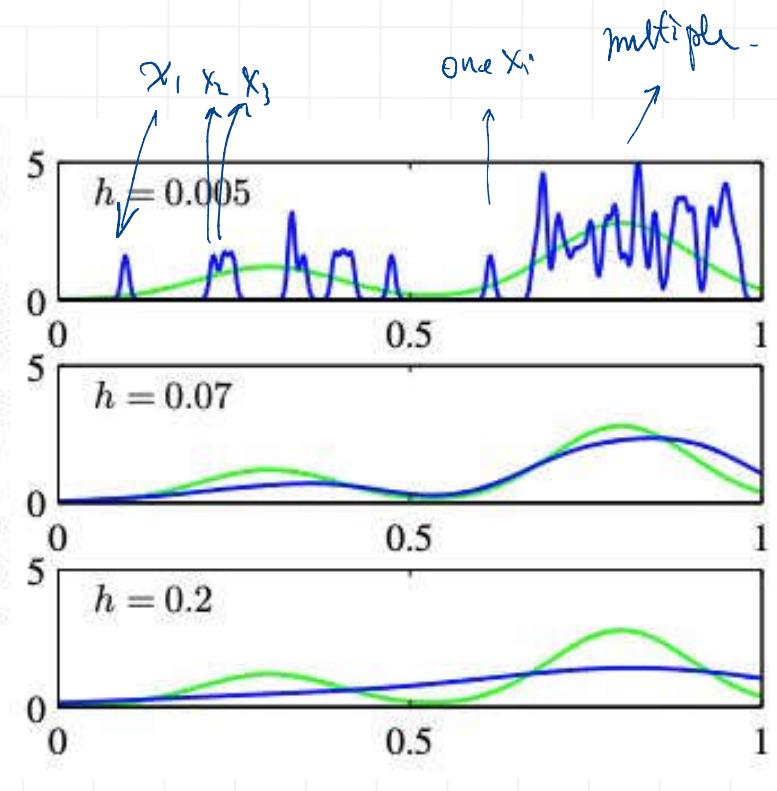
! We haven't improved from histograms (yet)

# Kernel density estimation with Gaussians

$$p(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^N \frac{1}{(2\pi h^2)^{1/2}} \exp \left\{ -\frac{\|\mathbf{x} - \mathbf{x}_n\|^2}{2h^2} \right\} \quad (2.250)$$

**Figure 2.25**

Illustration of the kernel density model (2.250) applied to the same data set used to demonstrate the histogram approach in Figure 2.24. We see that  $h$  acts as a smoothing parameter and that if it is set too small (top panel), the result is a very noisy density model, whereas if it is set too large (bottom panel), then the bimodal nature of the underlying distribution from which the data is generated (shown by the green curve) is washed out. The best density model is obtained for some intermediate value of  $h$  (middle panel).



## Kernel density estimation in general

Choose any  $k(\mathbf{u})$  s.t.

$$k(\mathbf{u}) \geq 0, \quad (2.251)$$

$$\int k(\mathbf{u}) d\mathbf{u} = 1 \quad (2.252)$$

Pro: training data simply stored, no computation needed for “training”

Con: training data simply stored, expensive to evaluate the density

$$p(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^N \frac{1}{h^D} k\left(\frac{\mathbf{x} - \mathbf{x}_n}{h}\right) \quad (2.249)$$

# Nearest neighbour methods

$$p(\mathbf{x}) = \frac{K}{NV}. \quad (2.246)$$

Drawback of KDE: fixed  $V$  (not adapting to data), fixed  $k(\mathbf{u})$ , having to choose  $h$  (1-d kernel width parameter)

Alternative: fix  $K$ , use data to find  $V$

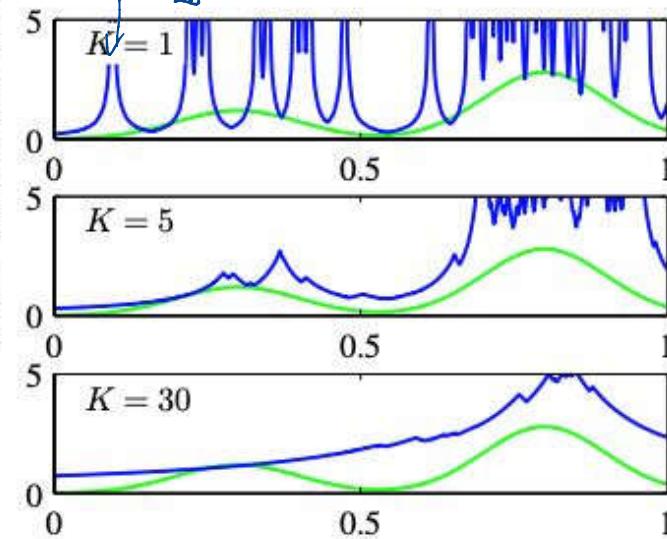
Consider a small sphere around  $\mathbf{x}$  and then allow the radius to increase until it contains exactly  $K$  data points. Set  $V$  to the volume of the resulting sphere.

! Do not produce true density estimates.

**Figure 2.26**

Illustration of  $K$ -nearest-neighbour density estimation using the same data set as in Figures 2.25 and 2.24. We see that the parameter  $K$  governs the degree of smoothing, so that a small value of  $K$  leads to a very noisy density model (top panel), whereas a large value (bottom panel) smoothes out the bimodal nature of the true distribution (shown by the green curve) from which the data set was generated.

$V \rightarrow 0$  When  $X \rightarrow X_n, n=1, \dots, N$   $p(x) \rightarrow \infty$



# Nearest neighbour for classification

$$p(\mathbf{x}) = \frac{K}{NV}. \quad (2.246)$$

draw a sphere centred on  $\mathbf{x}$  containing precisely  $K$  points irrespective of their class.

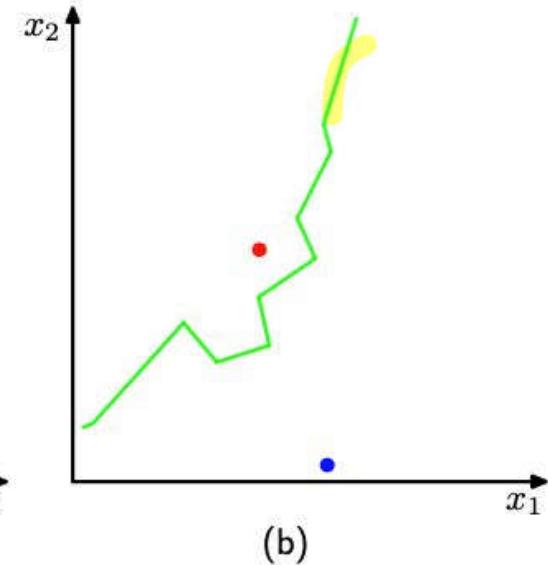
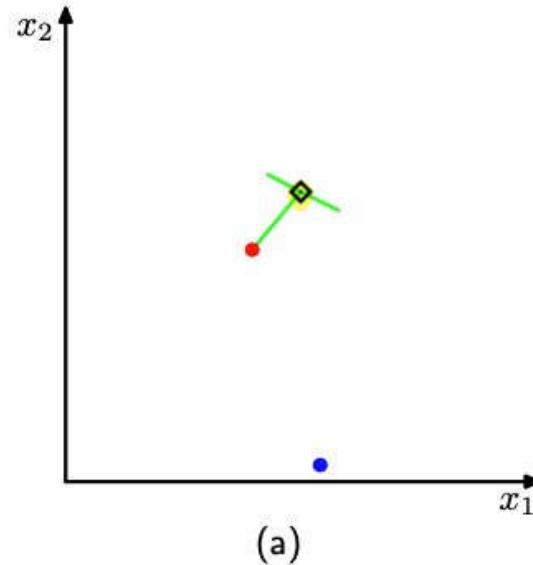
$$p(\mathbf{x}|\mathcal{C}_k) = \frac{K_k}{N_k V}. \quad (2.253)$$

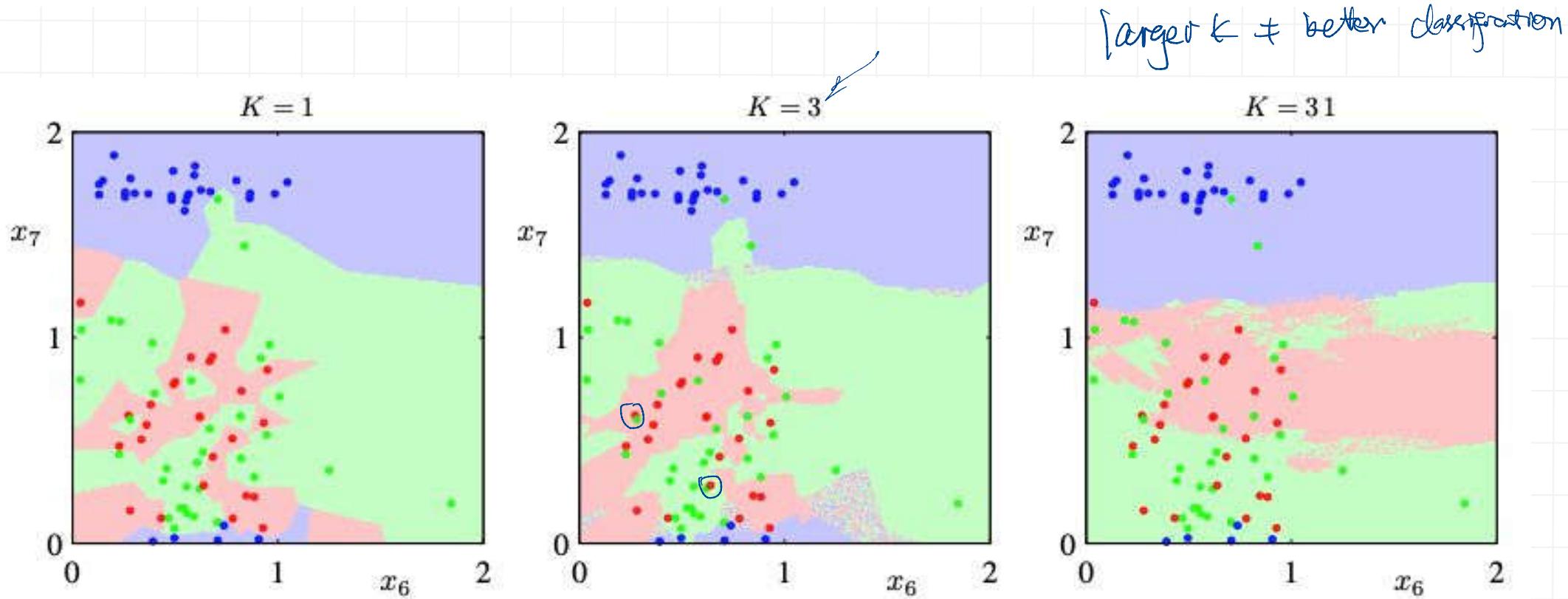
$K = 1$  : the *nearest-neighbour* rule

in the limit  $N \rightarrow \infty$ , the error rate of

1-NN is never more than twice the minimum achievable error rate of an optimal classifier, i.e., one that uses the true class distributions (Cover and Hart, 1967).

**Figure 2.27** (a) In the  $K$ -nearest-neighbour classifier, a new point, shown by the black diamond, is classified according to the majority class membership of the  $K$  closest training data points, in this case  $K = 3$ . (b) In the nearest-neighbour ( $K = 1$ ) approach to classification, the resulting decision boundary is composed of hyperplanes that form perpendicular bisectors of pairs of points from different classes.





**Figure 2.28** Plot of 200 data points from the oil data set showing values of  $x_6$  plotted against  $x_7$ , where the red, green, and blue points correspond to the 'laminar', 'annular', and 'homogeneous' classes, respectively. Also shown are the classifications of the input space given by the  $K$ -nearest-neighbour algorithm for various values of  $K$ .

# Kernels

Basis functions recap

Non-parametric methods (Chap 2.5)

Dual representation (of linear models, Chap 6.1)

Constructing kernels (Chap 6.2)

# The role of training data

- **Parametric methods**

$$w^T x, w^T \phi(x)$$

- Learn the model parameter  $w$  from the training data  $t$ .
- Discard the training data  $t$ .

- **Nonparametric methods**

circa 2005 ~ 2015  
"non-parametric Bayesian"  
 $\# \text{param} \rightarrow \infty$ ,

- Use training data directly for prediction
- $k$ -nearest neighbours : use  $k$ -closest data from the 'training' set for classification

- **Kernel methods**

- Base prediction on linear combination of kernel functions evaluated at the training data.

# Input vs Features

- A **feature** is a **measurable property of a phenomenon** being observed or any derived property thereof
  - raw features: the original data → **Input**
  - derived features: mappings of the original features to some other space (possibly high- or infinite dimensional, e.g., basis functions)
- **Feature selection**: which features matter for the problem at hand?
  - **redundant features** → w.r.t each other
  - problem dependent → are they informative of the target ?
- **Feature extraction**: can we combine the important features to a smaller set of new features?
  - compact representation versus ability to explain to a human

$$I(X_i; t) = 0$$

# Kernel methods in one slide

- Consider a labelled training set  $\{\mathbf{x}_i, t_i\}_{i=1}^N$  drawn i.i.d.,
- On a new point  $\mathbf{x}$ , we will predict

$$y(\mathbf{x}) = \sum_{i=1}^N a_i \cdot K(\mathbf{x}, \mathbf{x}_i)$$

*over all training points.*  
*"centered" on one training point  $\mathbf{x}_i$*

where  $\{a_i\}_{i=1}^N$  are weights to be determined based on our training set, and  $K(\cdot, \cdot)$  is a kernel function

- This is a major departure from the linear models considered previously!
- The kernel function measures the similarity between any two examples
  - Prediction is a weighted average of the training targets
  - Weights depend on the similarity of  $\mathbf{x}$  to each training example

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

loosely  
 $y(x) \sim \sum_i \alpha_i \text{sim}(\mathbf{x}, \mathbf{x}_i)$

$$\text{KDE } p(x) = \frac{1}{N} \sum_{i=1}^N K(\mathbf{x}, \mathbf{x}_i)$$

## From feature function to kernels

MSE + L<sub>2</sub> regularization -

$$J(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{ \mathbf{w}^T \underline{\phi(\mathbf{x}_n)} - t_n \}^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w} \quad (6.2) \quad \text{also (3.27)}$$

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

*↙ not what we need*

Optimal regularised w       $\mathbf{w}^* = (\lambda \mathbf{I} + \underline{\Phi^T \Phi})^{-1} \underline{\Phi^T \mathbf{t}}$       ← from chap 3

Prediction function for new x       $y(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w}^* = \phi(\mathbf{x})^T (\lambda \mathbf{I} + \Phi^T \Phi)^{-1} \Phi^T \mathbf{t}$

Goal

$$y(\mathbf{x}) = \sum_{i=1}^N a_i \cdot K(\mathbf{x}, \mathbf{x}_i)$$

$$\boxed{N \times N \text{ Gram matrix } \mathbf{K} = \Phi \Phi^T}$$

$$K \sim \text{Sim}(x_i, x_j)$$

$$\Phi \Phi^T \in \mathbb{R}^{N \times N}$$

$$\Phi^T \Phi \in \mathbb{R}^{M \times M}$$

# Dual representation

$$J(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{\mathbf{w}^\top \phi(\mathbf{x}_n) - t_n\}^2 + \frac{\lambda}{2} \mathbf{w}^\top \mathbf{w} \quad (3.27)$$

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

$$\frac{\partial J}{\partial \mathbf{w}} = 0$$

$$(\Phi^\top \Phi + \lambda \mathbf{I}) \mathbf{w} = \Phi^\top \mathbf{t}$$

$$\lambda \mathbf{w} = \Phi^\top (\mathbf{t} - \Phi \mathbf{w})$$

$$\mathbf{w} = \Phi^\top \mathbf{a}$$

$$= \sum_{n=1}^N \phi(\mathbf{x}_n) a_n$$

$$J(\mathbf{a}) = \frac{1}{2} \mathbf{a}^\top \underbrace{\Phi \Phi^\top}_{\mathbf{K}} \underbrace{\Phi \Phi^\top}_{\mathbf{K}} \mathbf{a} - \mathbf{a}^\top \underbrace{\Phi \Phi^\top}_{\mathbf{K}} \mathbf{t} + \frac{1}{2} \mathbf{t}^\top \mathbf{t} + \frac{\lambda}{2} \mathbf{a}^\top \underbrace{\Phi \Phi^\top}_{\mathbf{K}} \mathbf{a}$$

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

$N \times N$  Gram matrix  $\mathbf{K} = \Phi \Phi^\top$

not  $\Phi^\top \Phi$

$$\mathbf{w}^* = (\lambda \mathbf{I} + \Phi^\top \Phi)^{-1} \Phi^\top \mathbf{t} \quad K^\top = K$$

$$\vec{a} \in \mathbb{R}^N \quad \vec{w} \in \mathbb{R}^M$$

$$a_n = -\frac{1}{\lambda} \underbrace{\{\mathbf{w}^\top \phi(\mathbf{x}_n) - t_n\}}_{\text{pred. error on } x_n}$$

## Dual representation (contd)

$$J(\mathbf{a}) = \frac{1}{2} \mathbf{a}^T \Phi \Phi^T \Phi \Phi^T \mathbf{a} - \mathbf{a}^T \Phi \Phi^T \mathbf{t} + \frac{1}{2} \mathbf{t}^T \mathbf{t} + \frac{\lambda}{2} \mathbf{a}^T \Phi \Phi^T \mathbf{a} \quad (6.5)$$

Define the  $N \times N$  Gram matrix  $\mathbf{K} = \underline{\Phi \Phi^T}$  with elements

$$\underline{K_{nm}} = \underline{\phi(\mathbf{x}_n)^T} \underline{\phi(\mathbf{x}_m)} = \underline{k(\mathbf{x}_n, \mathbf{x}_m)} \quad (6.6)$$

for  $x$  not in  $\{\mathbf{x}_i\}_{i=1}^N$

$$\underline{k(x, x')} = \underline{\phi(x)^T} \underbrace{\Phi^T}_{\substack{\text{fact} \\ \text{of training}}} \sum_{i=1}^M \phi_i(x) \phi_i(x') \quad (6.10)$$

$$\underline{J(\mathbf{a})} = \frac{1}{2} \underline{\mathbf{a}^T K K \mathbf{a}} - \underline{\mathbf{a}^T K \mathbf{t}} + \frac{1}{2} \underline{\mathbf{t}^T \mathbf{t}} + \frac{\lambda}{2} \underline{\mathbf{a}^T K \mathbf{a}} \quad (6.7)$$

linear in  $\alpha$       linear in  $K$       const      quadratic in  $\alpha$

## Solving for $\mathbf{a}$ , prediction function

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^T \mathbf{K} \mathbf{K} \mathbf{a} - \mathbf{a}^T \mathbf{K} \mathbf{t} + \frac{1}{2}\mathbf{t}^T \mathbf{t} + \frac{\lambda}{2}\mathbf{a}^T \mathbf{K} \mathbf{a}. \quad (6.7)$$

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

(6.8)

$$\mathbf{w} = \Phi^T \mathbf{a}$$

GOAL

$$y(\mathbf{x}) = \sum_{i=1}^N a_i \cdot K(\mathbf{x}, \mathbf{x}_i),$$

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

(6.9)

*X w. all  $\{x_n\}_{n=1}^N$*

*pair-wise sim of training  $\{x_n\}_{n=1}^N$*

*inverse of  $R^{N \times N}$*

*pre-compute during training  $R^N$*

*$k_n(\mathbf{x}) = k(\mathbf{x}_n, \mathbf{x})$*

Prediction function for new  $\mathbf{x}$

$$y(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w}^* = \phi(\mathbf{x})^T (\lambda \mathbf{I} + \Phi^T \Phi)^{-1} \Phi^T \mathbf{t}$$

*training points summarized in  $(\lambda \mathbf{I} + \Phi^T \Phi)^{-1} \Phi^T \mathbf{t}$*

# Kernels

Basis functions recap

Non-parametric methods (Chap 2.5)

Dual representation (of linear models, Chap 6.1)

Constructing kernels (Chap 6.2)

Loosely,  $k(x, x')$  : similarity

symmetric

RHS.

$\forall \{x_n\}_{n=1}^N$

$K$  positive semi definite.

## The kernel function

- The **kernel function** is defined over two points,  $x$  and  $x'$ , of the **input space**

$$k(x, x') = \phi(x)^T \phi(x') = \sum_{i=1}^M \phi_i(x) \phi_i(x') \quad (6.10)$$

- $k(x, x')$  is **symmetric**.
- It is an **inner product of two vectors of basis functions**

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle.$$

- For prediction, the **kernel function** will be evaluated at the **training data points**. (See next slides.)

## An example

$$k(x, x') = \phi(x)^T \phi(x') = \sum_{i=1}^M \phi_i(x) \phi_i(x') \quad (6.10)$$

*inner prod.*

$$k(\mathbf{x}, \mathbf{z}) = (\underbrace{\mathbf{x}^T \mathbf{z}}_2)^2. \quad (x_1, x_2) \quad (z_1, z_2) \quad (6.11)$$

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

*only involve  
(x<sub>1</sub>, x<sub>2</sub>)*

*only involve  
(z<sub>1</sub>, z<sub>2</sub>)*

$$\phi''(\mathbf{x}) = (x_1, x_2, x_1 x_2, x_1^2, x_2^2, \dots)$$

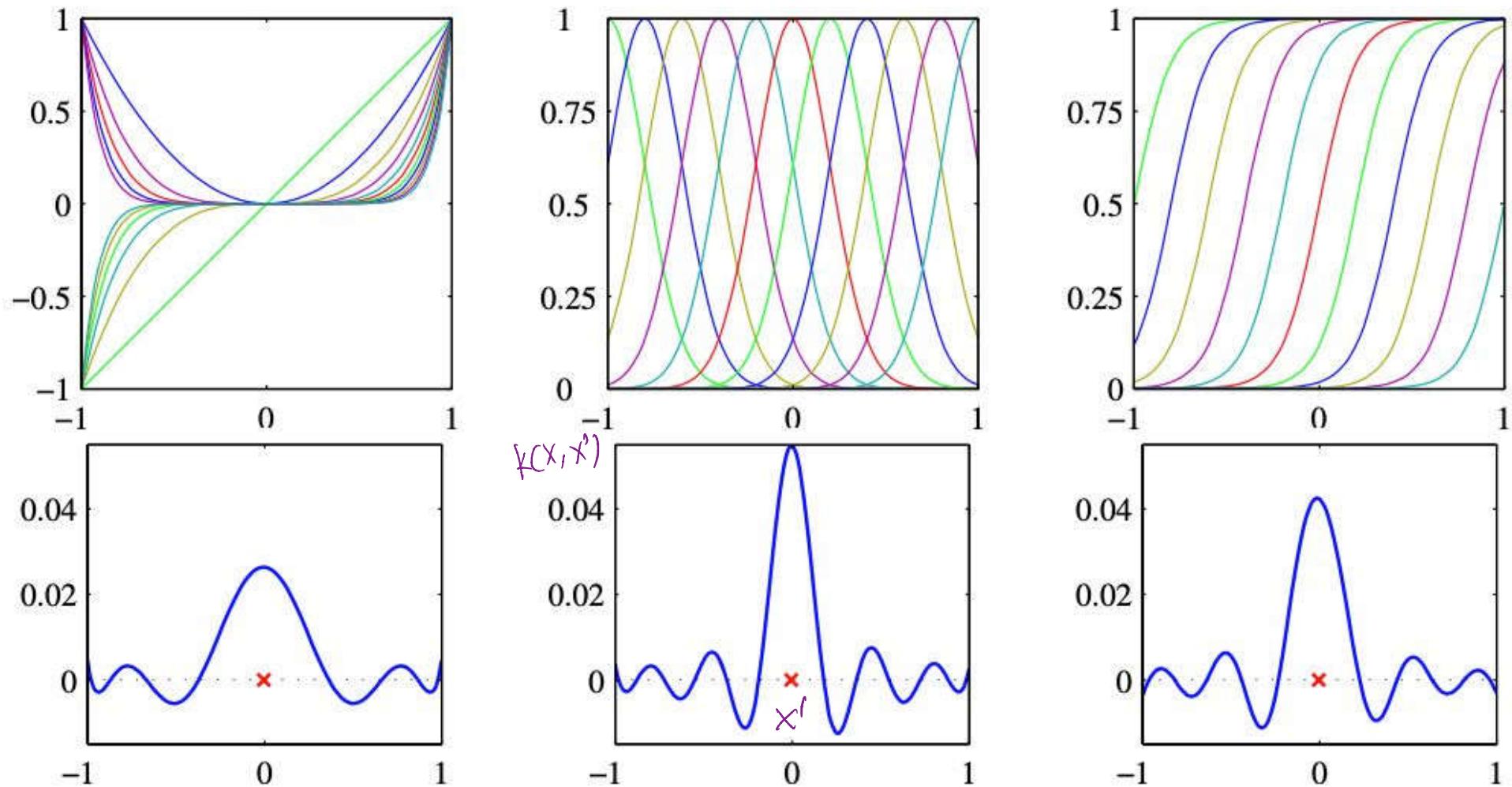
$$\phi(\mathbf{x}) = (x_1^2, x_1 x_2, x_2^2)^T$$

$$\underline{\phi(\mathbf{x})} = (x_1^2, \sqrt{2}x_1 x_2, x_2^2)^T$$

*features of polynomial  
order 2*

More generally, however, we need a simple way to test whether a function constitutes a valid kernel without having to construct the function  $\phi(\mathbf{x})$  explicitly. A **necessary and sufficient condition for a function  $k(\mathbf{x}, \mathbf{x}')$  to be a valid kernel** (Shawe-Taylor and Cristianini, 2004) is that the **Gram matrix  $\mathbf{K}$** , whose elements are given by  $k(\mathbf{x}_n, \mathbf{x}_m)$ , should be **positive semidefinite for all possible choices of the set  $\{\mathbf{x}_n\}$** . Note that a **positive semidefinite matrix** is not the same thing as a matrix whose elements are nonnegative.

*all eigenvalues  $\geq 0$*



**Figure 6.1** Illustration of the construction of kernel functions starting from a corresponding set of basis functions. In each column the lower plot shows the kernel function  $k(x, x')$  defined by (6.10) plotted as a function of  $x$  for  $x' = 0$ , while the upper plot shows the corresponding basis functions given by polynomials (left column), ‘Gaussians’ (centre column), and logistic sigmoids (right column).

## Techniques for Constructing New Kernels.

Given valid kernels  $k_1(\mathbf{x}, \mathbf{x}')$  and  $k_2(\mathbf{x}, \mathbf{x}')$ , the following new kernels will also be valid:

$$k(\mathbf{x}, \mathbf{x}') = ck_1(\mathbf{x}, \mathbf{x}') \quad (6.13)$$

$$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}') \quad (6.14)$$

$$k(\mathbf{x}, \mathbf{x}') = q(k_1(\mathbf{x}, \mathbf{x}')) \quad (6.15)$$

$$k(\mathbf{x}, \mathbf{x}') = \exp(k_1(\mathbf{x}, \mathbf{x}')) \quad (6.16)$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \quad (6.17)$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}') \quad (6.18)$$

$$k(\mathbf{x}, \mathbf{x}') = k_3(\phi(\mathbf{x}), \phi(\mathbf{x}')) \quad (6.19)$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{A} \mathbf{x}' \quad (6.20)$$

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b) \quad (6.21)$$

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a)k_b(\mathbf{x}_b, \mathbf{x}'_b) \quad (6.22)$$

where  $c > 0$  is a constant,  $f(\cdot)$  is any function,  $q(\cdot)$  is a polynomial with nonnegative coefficients,  $\phi(\mathbf{x})$  is a function from  $\mathbf{x}$  to  $\mathbb{R}^M$ ,  $k_3(\cdot, \cdot)$  is a valid kernel in  $\mathbb{R}^M$ ,  $\mathbf{A}$  is a symmetric positive semidefinite matrix,  $\mathbf{x}_a$  and  $\mathbf{x}_b$  are variables (not necessarily disjoint) with  $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$ , and  $k_a$  and  $k_b$  are valid kernel functions over their respective spaces.

$$\mathbf{x}^T \mathbf{x}'$$

$$(\mathbf{x}^T \mathbf{x}')^2$$

## Gaussian / Radial Basis Function (RBF) kernel

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2/2\sigma^2) \quad (6.23) \quad \rightarrow \text{What's } \phi(\mathbf{x})?$$

$$\|\mathbf{x} - \mathbf{x}'\|^2 = \mathbf{x}^T \mathbf{x} + (\mathbf{x}')^T \mathbf{x}' - 2\mathbf{x}^T \mathbf{x}' \quad (6.24)$$

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\mathbf{x}^T \mathbf{x}/2\sigma^2) \exp(\mathbf{x}^T \mathbf{x}'/\sigma^2) \exp(-(\mathbf{x}')^T \mathbf{x}'/2\sigma^2) \quad (6.25)$$

$$\begin{array}{ccc} f(\mathbf{x}) & \exp\left(\frac{1}{\sigma^2} k(\mathbf{x}, \mathbf{x}')\right) & f(\mathbf{x}') \\ & \downarrow & \\ & \mathbf{x}^T \mathbf{x}' & \end{array}$$

$$\phi(\mathbf{x}) = \mathbf{x}$$

## Further examples of kernels

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}')^M$$

only terms of degree  $M$

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + c)^M$$

all terms up to degree  $M$

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

Gaussian kernel

$$k(\mathbf{x}, \mathbf{x}') = \tanh(a \mathbf{x}^\top \mathbf{x}' + b)$$

Sigmoidal kernel (invalid)

easier to compute  
that  $\phi(\mathbf{x})^\top \phi(\mathbf{x}')$

Generally, we call

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$$

linear kernel

$$k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}')$$

only depend  
vec. diff

stationary kernel

$$k(\mathbf{x}, \mathbf{x}') = k(\|\mathbf{x} - \mathbf{x}'\|)$$

homogeneous kernel

only depend on distance

## Kernels over graphs, strings, sets

- We 'only' need an appropriate similarity measure  $k(\mathbf{x}, \mathbf{x}')$  which is a kernel.
- Example: Given a set  $\mathcal{A}$  and the set of all subsets of  $\mathcal{A}$ , called the **power set**  $\mathcal{P}(\mathcal{A})$ .
- For two subsets  $\mathcal{A}_1, \mathcal{A}_2 \in \mathcal{P}(\mathcal{A})$ , denote the number of elements of the intersection of  $\mathcal{A}_1$  and  $\mathcal{A}_2$  by  $|\mathcal{A}_1 \cap \mathcal{A}_2|$ .
- Then it can be shown that

$$k(\mathcal{A}_1, \mathcal{A}_2) = 2^{|\mathcal{A}_1 \cap \mathcal{A}_2|}$$

corresponds to an inner product in a feature space.  
Therefore,  $k(\mathcal{A}_1, \mathcal{A}_2)$  is a valid kernel function.

# Kernels from probabilistic generative models

- Given  $p(\mathbf{x})$ , we can define a kernel

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

which means two inputs  $\mathbf{x}$  and  $\mathbf{x}'$  are similar if they both have high probabilities.

- Include a weighting function  $p(i)$  and extend the kernel to

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

- For a continuous variable  $\mathbf{z}$

$$k(\mathbf{x}, \mathbf{x}') = \int p(\mathbf{x} | \mathbf{z}) p(\mathbf{x}' | \mathbf{z}) p(\mathbf{z}) d\mathbf{z}.$$

- Hidden Markov Model with sequences of length  $L$ .

# Kernels for regression and classification: summary

- Pick a suitable kernel function  $k(\mathbf{x}, \mathbf{x}')$ 
  - e.g. by computing inner product of some basis functions
- Make predictions by suitably combining  $k(\mathbf{x}, \mathbf{x}_n)$  for each training example  $\mathbf{x}_n$ 
  - implicitly, a linear model in some high-dimensional space
- For linear regression, we go from

$$y(\mathbf{x}) = \phi(\mathbf{x})^\top (\lambda \mathbf{I} + \Phi^\top \Phi)^{-1} \Phi^\top \mathbf{t}$$

to

$$y(\mathbf{x}) = \mathbf{k}(\mathbf{x})^\top (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t}$$

- can plug in suitable kernel function to **implicitly** perform **nonlinear transformation**

no need to write out  $\phi(\mathbf{x})$

Cons of  $k(\mathbf{x}, \mathbf{x}')$   
 $\mathcal{O}(N^2)$   
 $(\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathcal{O}(N^3)$

benefits of using  $k(\mathbf{x}, \mathbf{x}')$

- $\phi(\mathbf{x})$  can be implicitly  
“infinite” dimensions
- Compose new  $\mathbf{k}$  from  
existing  $\mathbf{k}$
- “sparse”  $\mathbf{x}$
- RBF SVMs → “sparse”  
weighted nearest neighbour
- New toolset

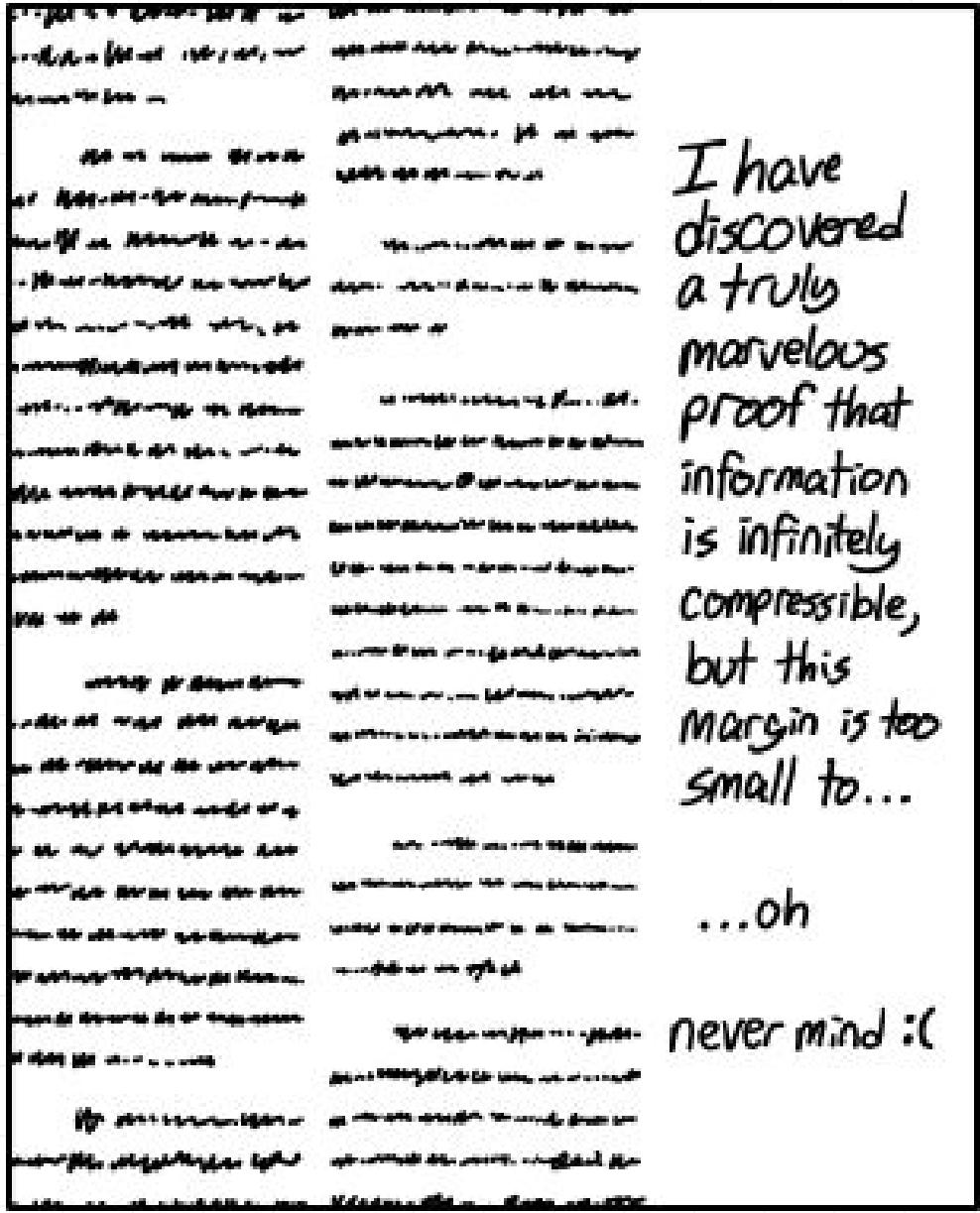
# Kernels

Basis functions recap

Non-parametric methods (Chap 2.5)

Dual representation (of linear models, Chap 6.1)

Constructing kernels (Chap 6.2)



I have  
discovered  
a truly  
marvelous  
proof that  
information  
is infinitely  
compressible,  
but this  
margin is too  
small to...

...oh

never mind :(

<https://xkcd.com/1381/>

Not the same kind of margin either ... :)

## Announcements

BREAK COMING UP :)

Watch out for survey

Week 7 Mon - Easter

Week 8 Mon - ANZAC Day

# Kernel machines

Lagrange multipliers, take 2

Maximum-margin classifiers (Chap 7.1)

- The intuition of margins
- Constructing the max-margin classifier
- The dual representation (cultural exposure)
- Support vectors and their geometric intuitions

SVM for overlapping class distributions

Relations to logistic regression

SVM for regression

The kernel trick.

More exposure to optimisation (lagrangian multipliers, KKT conditions, transforming a problem ... )

on Mon:

re-expressing regression  
w. kernels.  
 $O(n^3)$  ↑  
sparse?

# From kernels to sparse kernel machines

- Nonlinear kernels extended our toolbox of methods considerably
  - Wherever an inner product was used in an algorithm, we can replace it with a kernel.
  - Kernels act as a kind of 'similarity' measure and can be defined over graphs, sets, strings, and documents.
- But the kernel matrix is a square matrix with dimensions equal to the number of data points  $N$   $\xrightarrow{O(N^2)}$ 
  - In order to calculate it, the kernel function must be evaluated for all pairs of training inputs.
- **Sparse Kernel Machines** implement learning algorithms where, for prediction, the kernels are only evaluated at a subset of the training data.
- Today we introduce the famous **Support Vector Machine** — a non-probabilistic, non-parametric classifier, related to Kernel Logistic Regression  $\xrightarrow{\# \text{ of parameters is NOT fixed a priori}}$ 
  - partially alleviates the time complexity problems, but in general approximations are needed for a scalable algorithm.

For linear regression, we go from

$$y(\mathbf{x}) = \phi(\mathbf{x})^\top (\lambda \mathbf{I} + \Phi^\top \Phi)^{-1} \Phi^\top \mathbf{t}$$

to

$$y(\mathbf{x}) = \mathbf{k}(\mathbf{x})^\top (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t}$$

$K \in \mathbb{R}^{N \times N} \rightarrow \binom{N}{2} \text{ elements}$

$\xrightarrow{f: O(N^3)} \times$

don't do this,

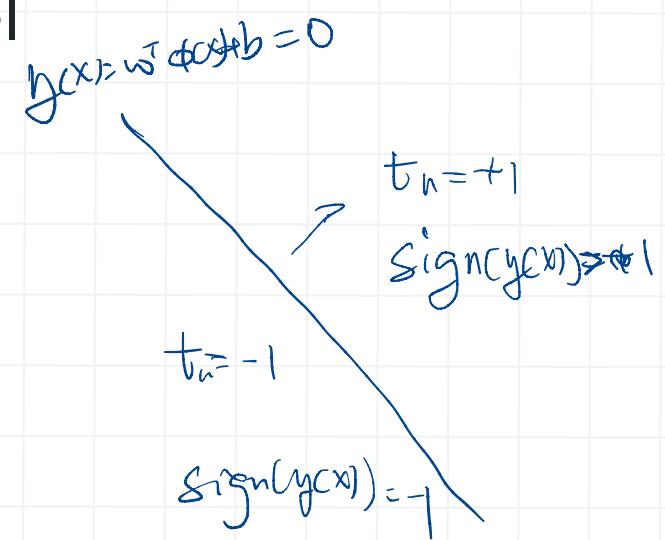
# Separating two classes with a linear model

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b \quad (7.1)$$

- Training data are  $N$  input vectors  $\mathbf{x}_1, \dots, \mathbf{x}_N$  with corresponding targets  $t_1, \dots, t_N$  where  $t_n \in \{-1, +1\}$ .
- The class of a new point is predicted as  $\text{sign}(y(\mathbf{x}))$ .

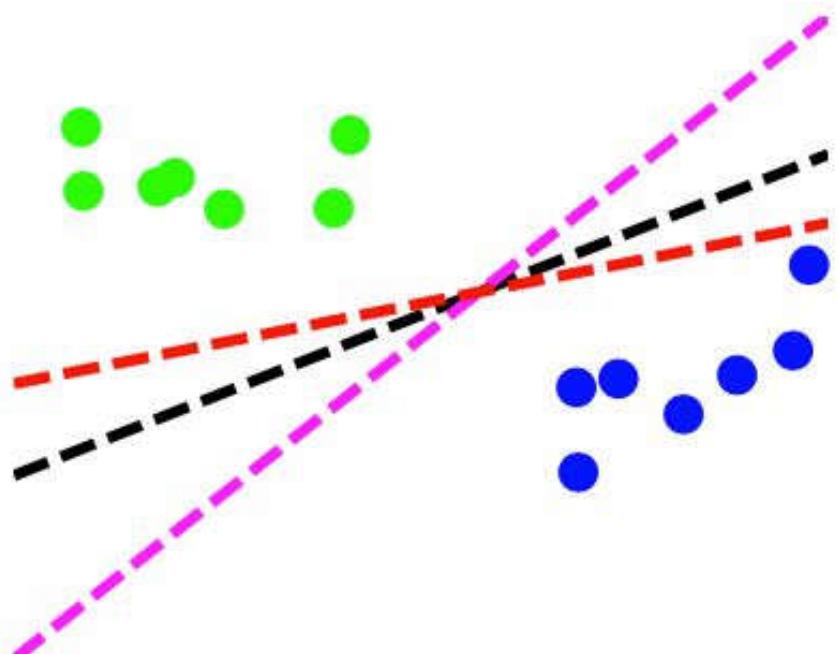
Assume: training dataset linearly separable, i.e. there exist  $\mathbf{w}$  and  $b$  such that:

$$t_n \text{ sign}(y(\mathbf{x}_n)) > 0 \quad n = 1, \dots, N.$$



# The multiple separator problem

There may exist many solutions  $w$  and  $b$  for which the linear classifier perfectly separates the two classes!

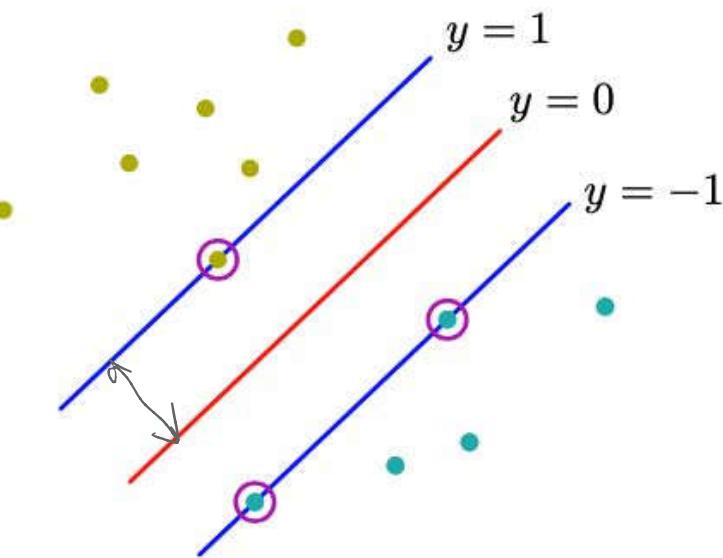
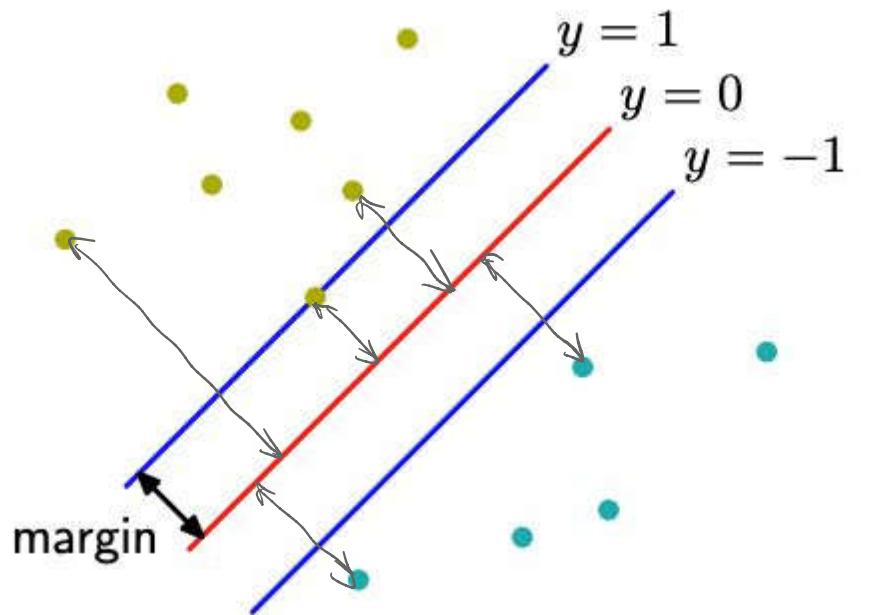


'far-away' from either class seems better

The perceptron algorithm can find a solution, but this depends on the initial choice of parameters.

What is the decision boundary which results in the best generalisation (smallest generalisation error)?

# Maximum margin classifier



**Figure 7.1** The margin is defined as the perpendicular distance between the decision boundary and the closest of the data points, as shown on the left figure. Maximizing the margin leads to a particular choice of decision boundary, as shown on the right. The location of this boundary is determined by a subset of the data points, known as support vectors, which are indicated by the circles.

Figure incomplete in pdf version of the book.

# Flashback: Discriminant for two classes

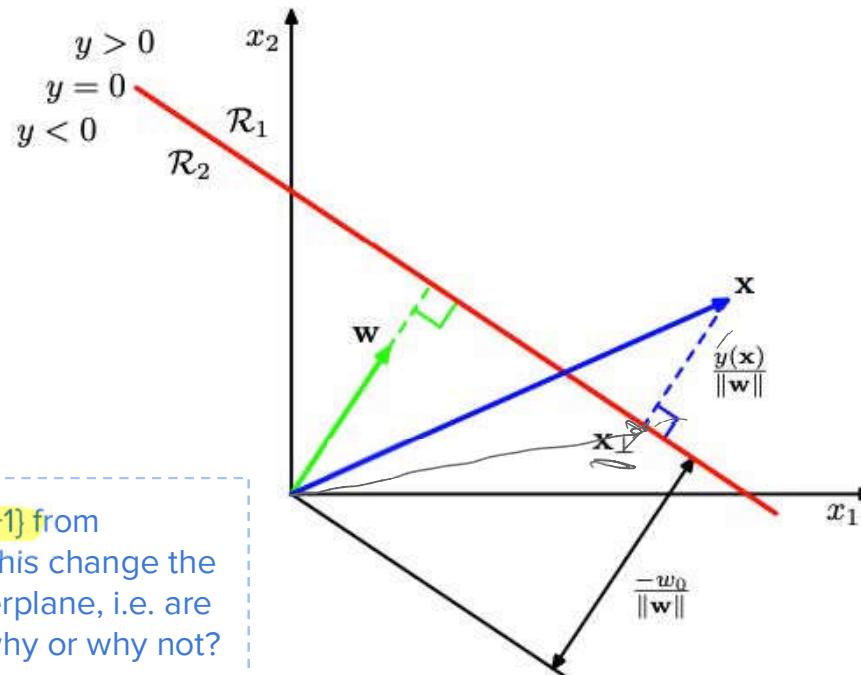
$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

(4.4)

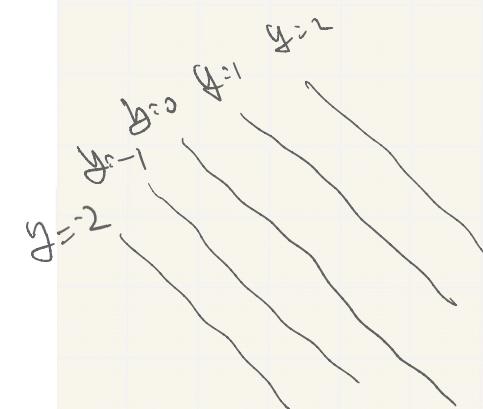
$y(\mathbf{x})$  gives a **signed** measure of the perpendicular distance  $r$  from the decision surface to  $\mathbf{x}$ , that is  $r = y(\mathbf{x})/\|\mathbf{w}\|$ .

$$y(\mathbf{x}) = \mathbf{w}^T \underbrace{\left( \mathbf{x}_\perp + r \frac{\mathbf{w}}{\|\mathbf{w}\|} \right)}_{\mathbf{x}} + w_0 = r \frac{\mathbf{w}^\top \mathbf{w}}{\|\mathbf{w}\|} + \underbrace{\mathbf{w}^\top \mathbf{x}_\perp + w_0}_{0} = r \|\mathbf{w}\|$$

**Figure 4.1** Illustration of the geometry of a linear discriminant function in two dimensions. The decision surface, shown in red, is perpendicular to  $\mathbf{w}$ , and its displacement from the origin is controlled by the bias parameter  $w_0$ . Also, the signed orthogonal distance of a general point  $\mathbf{x}$  from the decision surface is given by  $y(\mathbf{x})/\|\mathbf{w}\|$ .



Q: The target encoding is changed to  $\{-1, +1\}$  from one-of-K encoding from Chapter 4. Does this change the meaning and value of the separating hyperplane, i.e. are the two classes still separated by  $y(\mathbf{x})=0$ , why or why not?

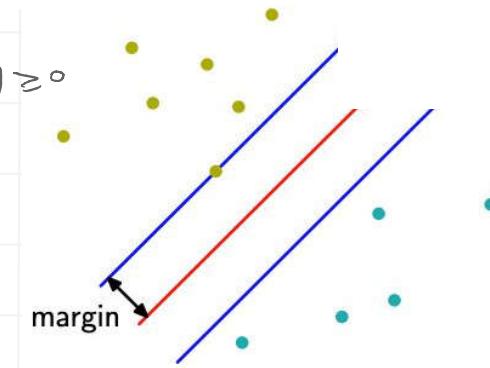


Opt. problem

$$\begin{aligned} \max \quad & f(x) \\ \text{s.t.} \quad & g(x) \geq 0 \end{aligned}$$

## Maximum margin solution: objective function

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b \quad (7.1)$$



Separating hyperplane:  $y(\mathbf{x}) = 0$

Distance of  $y(\mathbf{x})$  from they separating hyperplane  $y(\mathbf{x})=0$ :  $|y(\mathbf{x})|/\|\mathbf{w}\|$ .

$$\frac{t_n y(\mathbf{x}_n)}{\|\mathbf{w}\|} = \frac{t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b)}{\|\mathbf{w}\|}. \quad (7.2)$$

unsigned distance ,

$$t_n \operatorname{sgn}(y(\mathbf{x})) > 0$$

Solve the following to obtain maximum margin solution:

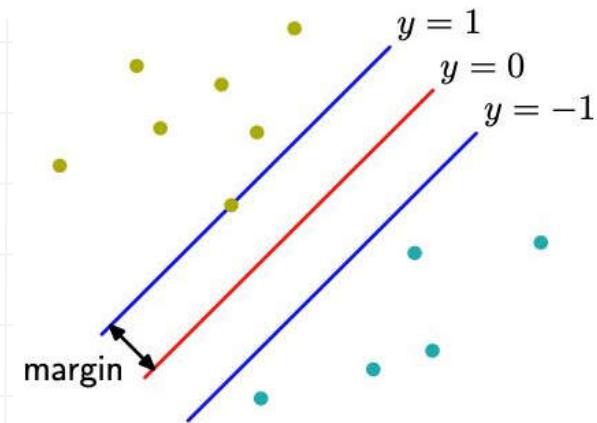
$$\arg \max_{\mathbf{w}, b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n [t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b)] \right\} \quad (7.3)$$

$\mathbf{w}$  unconstrained  $\in \mathbb{R}^d$ .

## Normalising the margin

$$\arg \max_{\mathbf{w}, b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n [t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b)] \right\} \quad (7.3)$$

non-linear obj  
in  $\mathbf{w}$



Rescaling doesn't change the margin of each data point  $\{\mathbf{x}_n, t_n\}$ :

If  $\mathbf{w} \rightarrow \kappa \mathbf{w}$  and  $b \rightarrow \kappa b$ ,  $t_n y(\mathbf{x}_n)/\|\mathbf{w}\|$  unchanged.

For points closest to decision boundary, set:

$$t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) = 1 \quad (7.4)$$

This implies that all data points lie “outside the margin”

$$t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1, \quad n = 1, \dots, N. \quad (7.5)$$

$\xrightarrow{\text{linear constraints}} \text{in } \mathbf{w}$ .

## Equivalent objective function

from last page

$$\arg \max_{\mathbf{w}, b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n [t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b)] \right\} \quad (7.3)$$

"the point"

For points closest to decision boundary, set:

$$\underbrace{t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b)}_{=1} \quad (7.4)$$

This implies that all data points lie "on or outside the margin"

$$t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1, \quad n = 1, \dots, N. \quad (7.5)$$

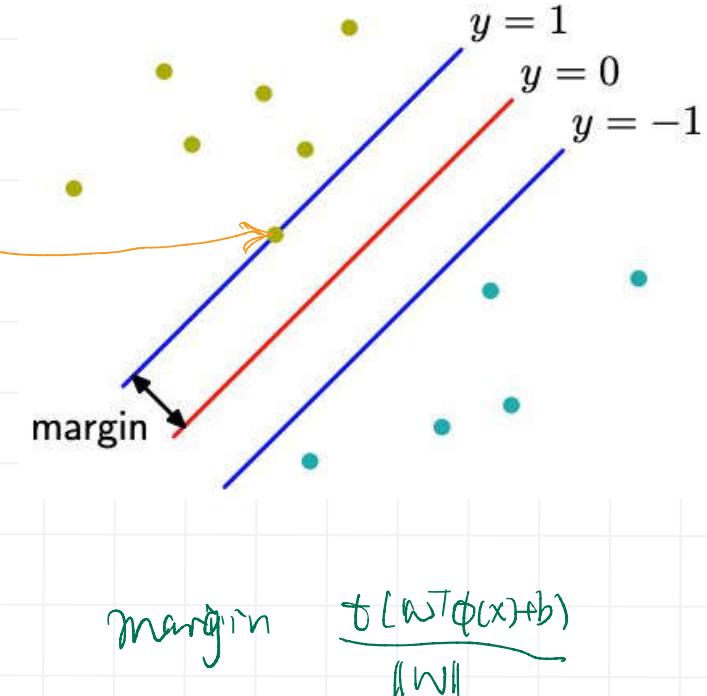
(7.4 holds)

there is at least one constraint active, where  $t_n y(x_n) = 1$

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2$$

(7.6)

$$\begin{aligned} & \text{maximize } \underbrace{\|\mathbf{w}\|^{-1}}_{\text{quadratic obj}} \\ & \min \underbrace{\|\mathbf{w}\|}_{\text{quadratic obj}} \end{aligned}$$



$$\begin{aligned} t \operatorname{sign}(y(x)) &= 1 \\ +1/-1 &\quad +1/-1 \end{aligned}$$

# Outline

Lagrange multipliers, take 2

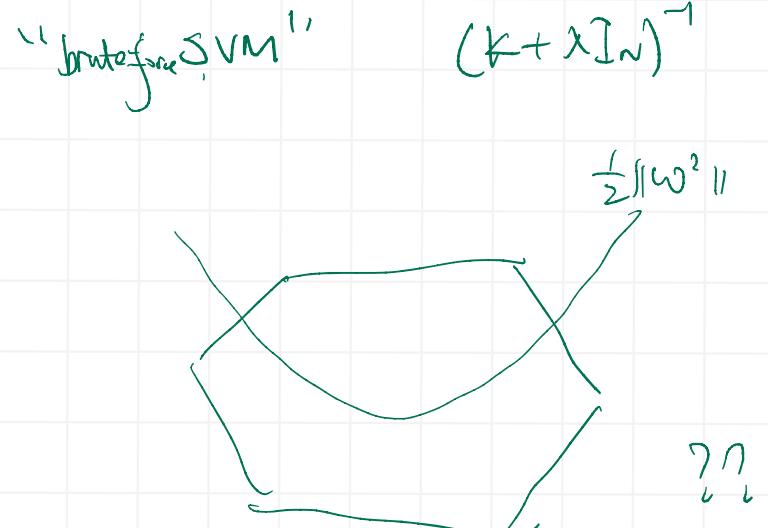
Maximum-margin classifiers (Chap 7.1)

- The intuition of margins
- Constructing the max-margin classifier
- The dual representation
- Support vectors and their geometric intuitions

SVM for overlapping class distributions

Relations to logistic regression

SVM for regression



# Flashback: Lagrange multipliers (appendix E)

The first encounter in SML - we'll see it again in kernel methods.

objective function  
equality constraint

maximize  $f(x)$   
subject to  $g(x)=0$

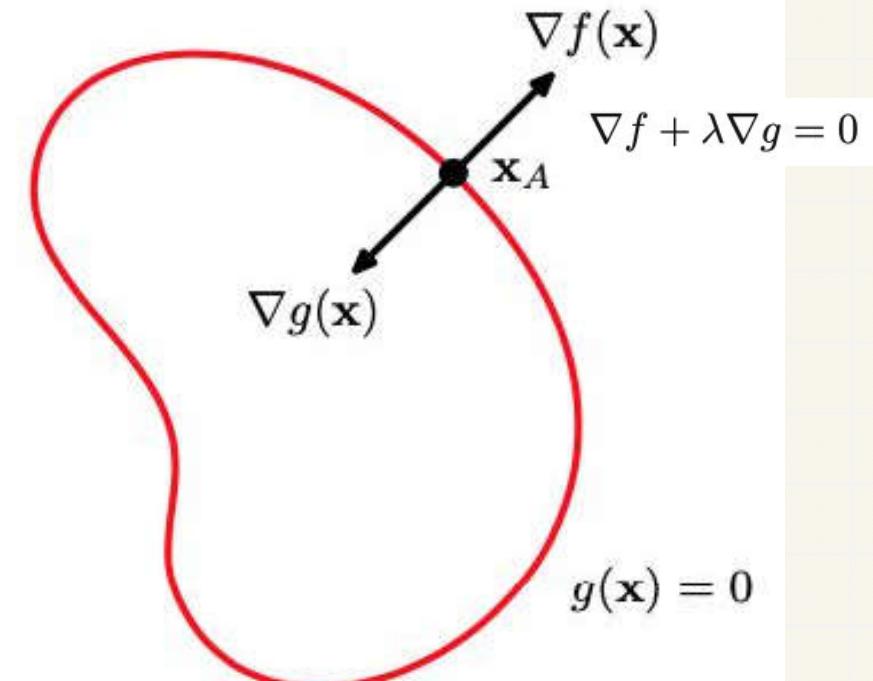


$$\mathcal{L}(x, \lambda) = f(x) + \lambda g(x)$$

$\mathcal{L}$  = Lagrangian

$\lambda$  = Lagrange multiplier

**Figure E.1** A geometrical picture of the technique of Lagrange multipliers in which we seek to maximize a function  $f(\mathbf{x})$ , subject to the constraint  $g(\mathbf{x}) = 0$ . If  $\mathbf{x}$  is  $D$  dimensional, the constraint  $g(\mathbf{x}) = 0$  corresponds to a subspace of dimensionality  $D - 1$ , indicated by the red curve. The problem can be solved by optimizing the Lagrangian function  $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$ .



# Lagrange multipliers with inequality constraints

objective function  
inequality constraint

maximize  $f(x)$   
subject to  $g(x) \geq 0$

max.  $L(x, \lambda) = f(x) + \lambda g(x)$

Karush-Kuhn-Tucker (KKT) conditions

$$g(\mathbf{x}) \geq 0 \quad (\text{E.9})$$

$$\lambda \geq 0 \quad (\text{E.10})$$

$$\lambda g(\mathbf{x}) = 0 \quad (\text{E.11})$$

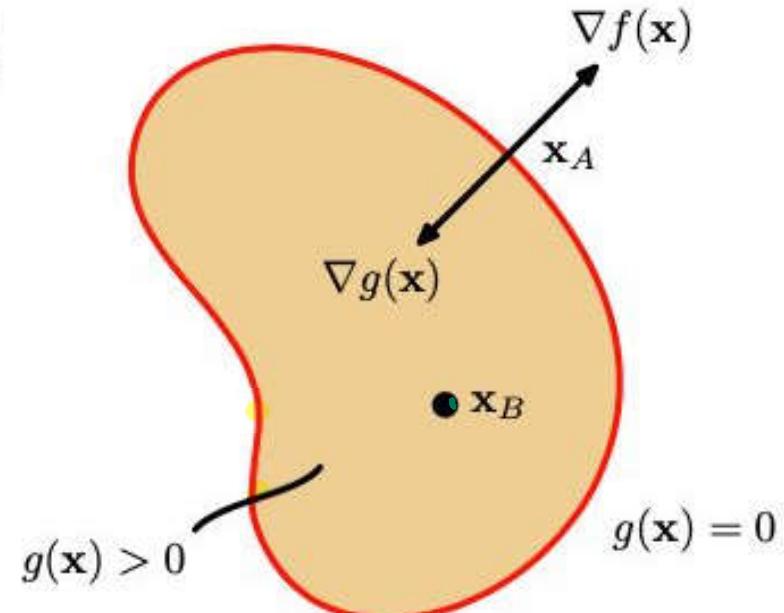
**Figure E.3** Illustration of the problem of maximizing  $f(\mathbf{x})$  subject to the inequality constraint  $g(\mathbf{x}) \geq 0$ .

two cases

stationary point of  $L(x, \lambda)$

Constraint *inactive*:  $g(\mathbf{x}) > 0 \rightarrow \nabla f(\mathbf{x}) = 0, \lambda = 0$

Constraint *active*:  $g(\mathbf{x}) = 0 \rightarrow \nabla f(\mathbf{x}) = -\lambda \nabla g(\mathbf{x}), \lambda > 0$





## Joseph-Louis Lagrange

1736–1813

Although widely considered to be a French mathematician, Lagrange was born in Turin in Italy. By the age of nineteen, he had already made important contributions mathematics and had been appointed as Professor at the Royal Artillery School in Turin. For many

years, Euler worked hard to persuade Lagrange to move to Berlin, which he eventually did in 1766 where he succeeded Euler as Director of Mathematics at the Berlin Academy. Later he moved to Paris, narrowly escaping with his life during the French revolution thanks to the personal intervention of Lavoisier (the French chemist who discovered oxygen) who himself was later executed at the guillotine. Lagrange made key contributions to the calculus of variations and the foundations of dynamics.

The KKT conditions were originally named after Harold W. Kuhn and Albert W. Tucker, who first published the conditions in 1951. Later scholars discovered that the necessary conditions for this problem had been stated by William Karush in his master's thesis in 1939.

[https://en.wikipedia.org/wiki/Karush%E2%80%93Kuhn%E2%80%93Tucker\\_conditions](https://en.wikipedia.org/wiki/Karush%E2%80%93Kuhn%E2%80%93Tucker_conditions)

# The max-margin problem and its Lagrangian (primal)

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad (7.6)$$

$$\text{s.t. } t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1, \quad n = 1, \dots, N. \quad (7.5)$$

The Lagrangian, define one  $a_n >= 0$  for each constraint in (7.5)

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N a_n \{t_n(\mathbf{w}^T \phi(\mathbf{x}_n) + b) - 1\} \quad (7.7)$$

Set its derivative w.r.t  $\mathbf{w}$  and  $b$  to 0

$$\mathbf{w} = \sum_{n=1}^N a_n t_n \phi(\mathbf{x}_n) \quad (7.8)$$

$$0 = \sum_{n=1}^N a_n t_n. \quad (7.9)$$

←  $\{a_n\}$  are still left!

## The dual representation

→ opt problem expressed in Lagrange multipliers  
Here: cultural exposure

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N a_n \{t_n(\mathbf{w}^T \phi(\mathbf{x}_n) + b) - 1\} \quad (7.7)$$

Set the derivative of L w.r.t.  
 $\mathbf{w}$  and  $b$  to 0

$$\begin{aligned} \mathbf{w} &= \sum_{n=1}^N a_n t_n \phi(\mathbf{x}_n) \\ 0 &= \sum_{n=1}^N a_n t_n. \end{aligned}$$

Substitute  $\mathbf{w}$  with (7.8) and use (7.9) - see notes.

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) \quad (7.10)$$

subject to the constraints

$$a_n \geq 0, \quad n = 1, \dots, N, \quad (7.11)$$

$$\sum_{n=1}^N a_n t_n = 0. \quad (7.12)$$

from last page.  
(7.8)  
(7.9)

Hope: lots of  $a_n = 0$   
→ sparsity.

total weights of  
class +1 & class -1  
are equal.

Quadratic program: optimize a quadratic function of  $\mathbf{a}$  subject to a set of linear inequality constraints.

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) \quad (7.10)$$

subject to the constraints

$$a_n \geq 0, \quad n = 1, \dots, N, \quad (7.11)$$

$$\sum_{n=1}^N a_n t_n = 0. \quad (7.12)$$

obtained using QP solver

OR customized SVM solver

Assume we have solutions for  $\mathbf{a}$ , now solve for  $\mathbf{b}$

$$t_n \left( \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) + b \right) = 1 \quad (7.17)$$

$$b = \frac{1}{N_S} \sum_{n \in \mathcal{S}} \left( t_n - \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \right) \quad (7.18)$$

SVM derivation - dual form

$$L(\vec{w}, b, \vec{a}) = \frac{1}{2} \|\vec{w}\|^2 - \sum_{n=1}^N a_n \{ t_n (\vec{w}^\top \phi(x_n) + b) - 1 \} \quad (7.7)$$

$$\downarrow \vec{w} = \sum_{n=1}^N a_n t_n \phi(x_n) \quad \frac{\partial L}{\partial \vec{w}} = \vec{w} - \sum_{n=1}^N a_n t_n \phi(x_n) = 0 \quad (7.8)$$

$$L(b, \vec{a}) = \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m \phi(x_n)^\top \phi(x_m) \\ - \sum_{n=1}^N a_n \left\{ t_n \sum_{m=1}^N a_m t_m \phi(x_m)^\top \phi(x_n) + t_n b - 1 \right\} \quad (7.9)$$

$$= -\frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m \phi(x_n)^\top \phi(x_m) - \left( \sum_{n=1}^N a_n t_n \right) b + \sum_{n=1}^N a_n$$

$$= \sum_{n=1}^N a_n - \frac{1}{2} \sum_{m=1}^N \sum_{n=1}^N a_n a_m t_n t_m k(x_n, x_m) \quad (7.10)$$

left 4  
skipped,  
heading

# KKT conditions → support vectors

Make predictions via:

$$y(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b = \sum_{n=1}^N a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b \quad (7.13)$$

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N a_n \{t_n (\mathbf{w}^\top \phi(\mathbf{x}_n) + b) - 1\} \quad (7.7)$$

KKT conditions

$$a_n \geq 0 \quad (7.14)$$

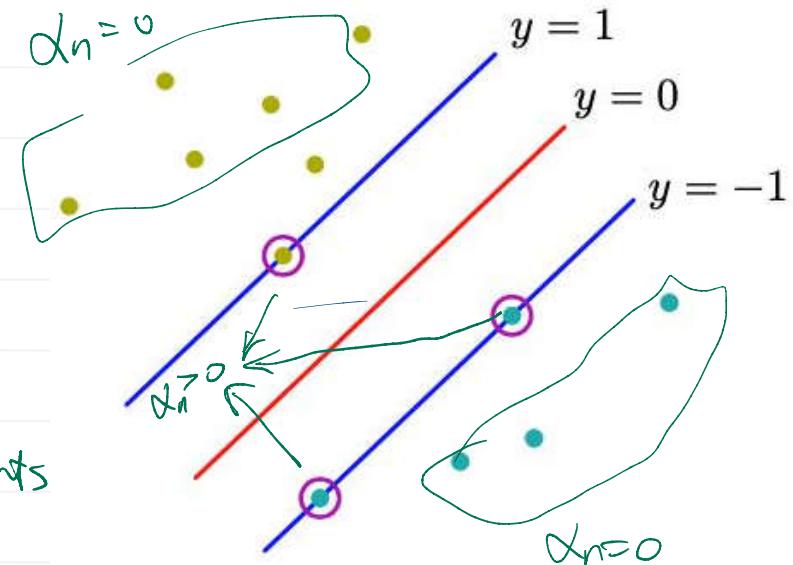
$$t_n y(\mathbf{x}_n) - 1 \geq 0 \quad (7.15)$$

$$a_n \{t_n y(\mathbf{x}_n) - 1\} = 0. \quad (7.16)$$

either  $x_n$  or  $t_n(y(x_n)) - 1 = 0$

$$\mathbf{w} = \sum_{n=1}^N a_n t_n \phi(\mathbf{x}_n) \quad (7.8)$$

ignore training points  
when  $\alpha_n = 0$

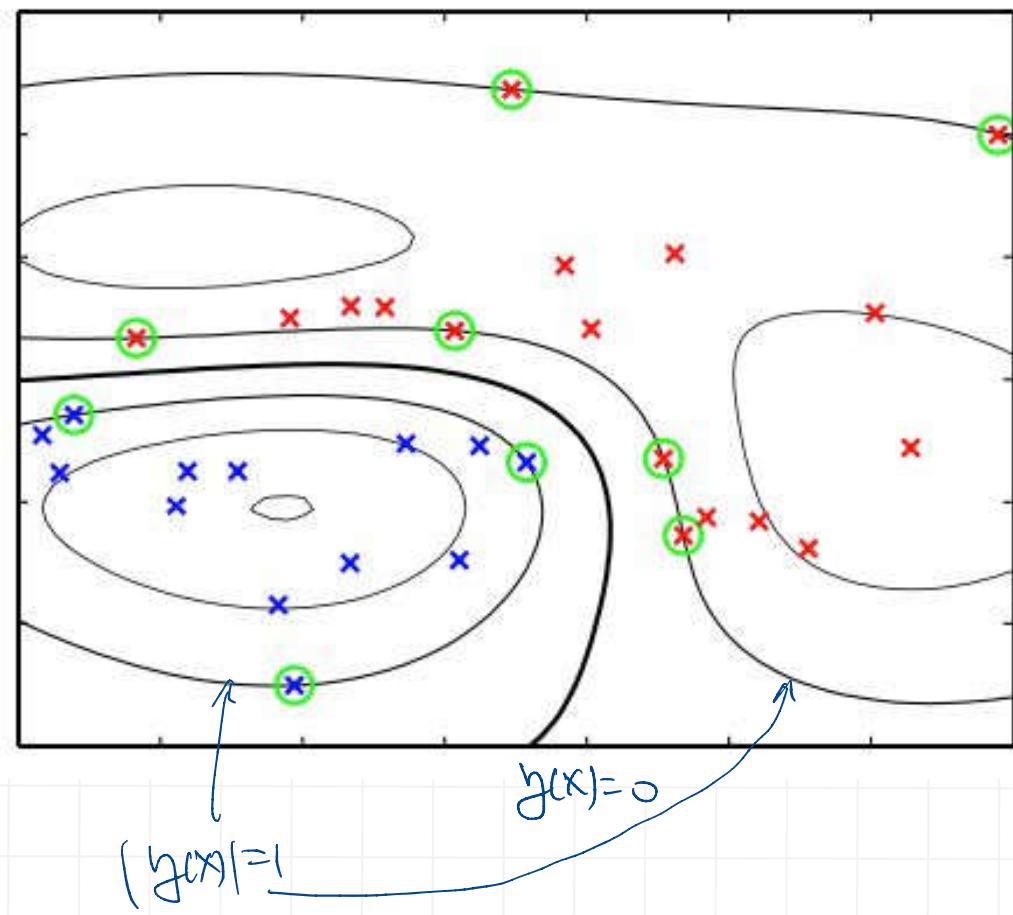


In this example,  
only 3  $\alpha_n > 0$ .

Nice theoretical properties,  
via VC dimensions.

## Geometric insight for sparsity

**Figure 7.2** Example of synthetic data from two classes in two dimensions showing contours of constant  $y(x)$  obtained from a support vector machine having a Gaussian kernel function. Also shown are the decision boundary, the margin boundaries, and the support vectors.



# Outline

Lagrange multipliers, take 2

Maximum-margin classifiers (Chap 7.1)

- The intuition of margins
  - Constructing the max-margin classifier
  - The dual representation
  - Support vectors and their geometric intuitions
- KKT conditions* ↗

SVM for overlapping class distributions

Relations to logistic regression

SVM for regression

# What happens if the classes overlap?

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad (7.6)$$

$$\text{s.t. } t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1, \quad n = 1, \dots, N. \quad (7.5)$$

$$\sum_{n=1}^N E_\infty(y(\mathbf{x}_n)t_n - 1) + \lambda \|\mathbf{w}\|^2$$

(7.19)

**Figure 7.3**

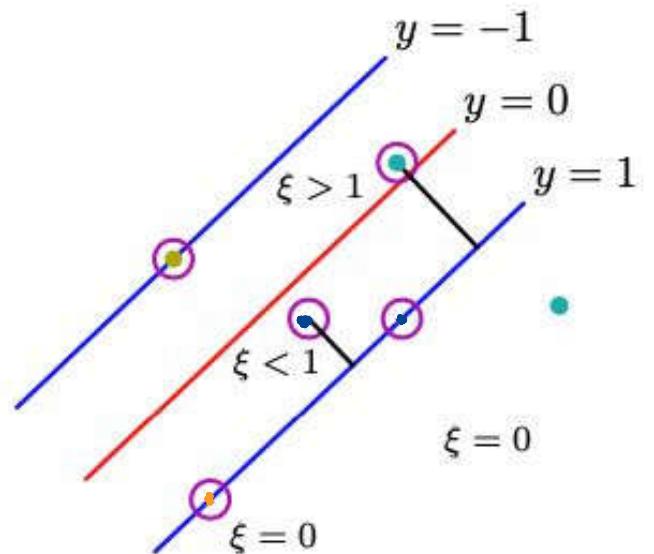
Illustration of the slack variables  $\xi_n \geq 0$ . Data points with circles around them are support vectors.

First, replace the constraints as

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n, \quad n = 1, \dots, N \quad (7.20)$$

$\xi_n \geq 0$

- Allow some data points to be on the 'wrong side' of the decision boundary.
- Increase a penalty with distance from the decision boundary.

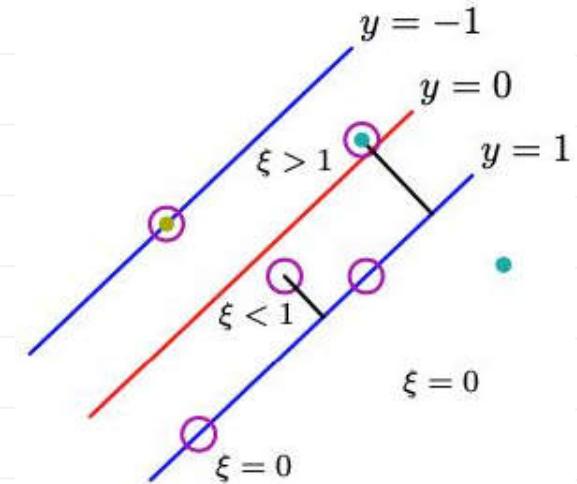


# The “soft margin” optimisation problem

SVM with “hard margins”

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad (7.6)$$

$$\text{s.t. } t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1, \quad n = 1, \dots, N. \quad (7.5)$$



*total slack*      *sane*

$$\text{minimize} \quad C \sum_{n=1}^N \xi_n + \frac{1}{2} \|\mathbf{w}\|^2 \quad (7.21)$$

s.t.       $\xi_n \geq 0$

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n, \quad n = 1, \dots, N \quad (7.20)$$

*margin violation.*

$\xi_n > 1$  if the point is mis-classified. Therefore  $\sum_n \xi_n$  is an upper bound on the number of misclassified points.

C controls the trade-off between the slack variable penalty and the margin -- the objective tries minimise “total slack” across all training points.

# The “soft margin” optimisation problem

$$\text{minimize} \quad C \sum_{n=1}^N \xi_n + \frac{1}{2} \|\mathbf{w}\|^2$$

s.t.

$$\xi_n \geq 0$$

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n, \quad n = 1, \dots, N$$

(7.21)

(7.20)

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n - \sum_{n=1}^N a_n \{ t_n y(\mathbf{x}_n) - 1 + \xi_n \} - \sum_{n=1}^N \mu_n \xi_n \quad (7.22)$$

additional

their own  
L-multiplicators.

KKT conditions

$$\begin{aligned} a_n &\geq 0 \\ t_n y(\mathbf{x}_n) - 1 + \xi_n &\geq 0 \\ a_n (t_n y(\mathbf{x}_n) - 1 + \xi_n) &= 0 \\ \mu_n &\geq 0 \\ \xi_n &\geq 0 \\ \mu_n \xi_n &= 0 \end{aligned}$$

(7.23)

(7.24)

(7.25)

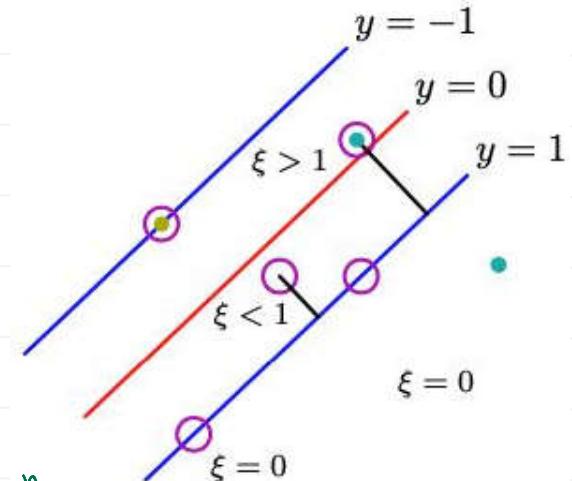
(7.26)

(7.27)

(7.28)

def of L-multiplicators

What changed, if any, in its solution?



# The dual problem of soft margin SVM

Set derivatives of L to zero to eliminate  $\mathbf{w}$ ,  $b$ ,  $\xi_n$ ,  $\mu_n$

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) \quad (7.32)$$

$$0 \leq a_n \leq C$$

$$\sum_{n=1}^N a_n t_n = 0 \quad (7.34)$$

The only change from the separable case, is the **box constraint** via the parameter **C**.

Prediction function unchanged

$$y(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b = \sum_{n=1}^N a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b \quad (7.13)$$

Prediction time:

We need  $k(\mathbf{x}, \mathbf{x}_n), \alpha_n > 0$

We don't need  $k$

Training time: ???

e.g. SMO

You may still need to compute K

Similar -

linear  
poly  
RBF  
graph ...

## Limitations of support vector machines

- Output are decisions, not posterior probabilities.
- Extension to classification with more than two classes is problematic.
- There is a complexity parameter  $C$  (or  $\nu$ ) which must be found (e.g. via cross-validation).
- Predictions are expressed as linear combinations of kernel functions that are centered on the training points.
- Kernel matrix is required to be positive definite.

platt's Scaling

## *Further Reading (No Assessable)*

- Our derivation of the SVM
  - provides a nice example of Lagrangian optimisation
  - yields a neat dual formulation
  - is of historical value
- It is possible to derive an SVM algorithm much more simply however. See e.g.:

Olivier Chapelle

*Training a support vector machine in the primal*  
Neural computation, 2007 - MIT Press.

# Outline

Lagrange multipliers, take 2

Maximum-margin classifiers (Chap 7.1)

- The intuition of margins
- Constructing the max-margin classifier
- The dual representation
- Support vectors and their geometric intuitions

SVM for overlapping class distributions

Relations to logistic regression

SVM for regression

# Re-writing SVM objective with hinge loss

When  $y_n t_n > 1$ ,  $\xi_n = 0$ ; otherwise,  $\xi_n = 1 - y_n t_n$

$$C \sum_{n=1}^N \xi_n + \frac{1}{2} \|\mathbf{w}\|^2 \quad (7.21)$$

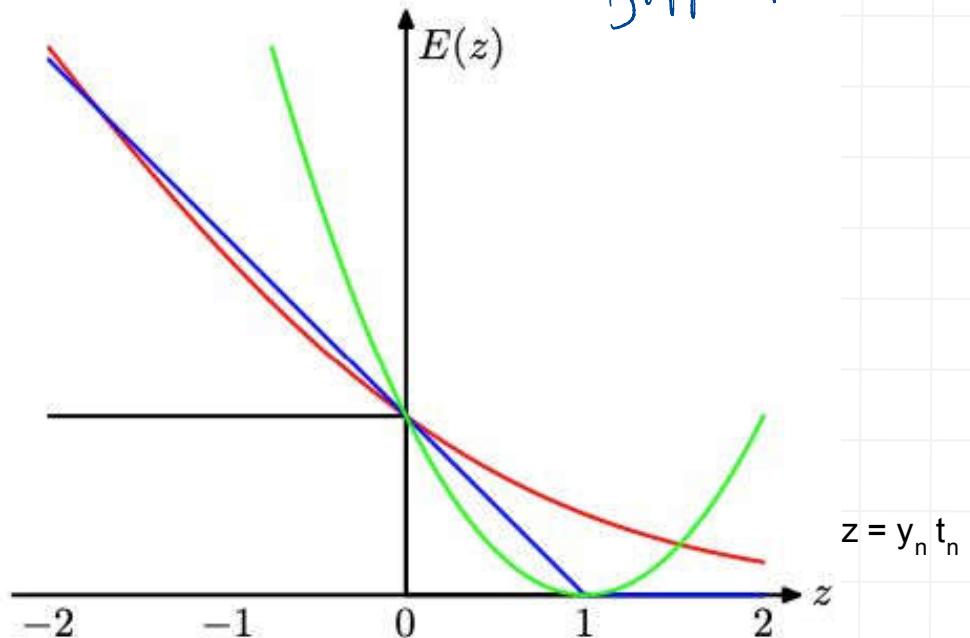
**Figure 7.5** Plot of the 'hinge' error function used in support vector machines, shown in blue, along with the error function for logistic regression, rescaled by a factor of  $1/\ln(2)$  so that it passes through the point  $(0, 1)$ , shown in red. Also shown are the misclassification error in black and the squared error in green.

$$\sum_{n=1}^N E_{\text{SV}}(y_n t_n) + \lambda \|\mathbf{w}\|^2 \quad (7.44)$$

$E_{\text{SV}}(y_n t_n) = [1 - y_n t_n]_+$

(x)\_+ \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}

flipped ReLU.



# Loss functions

- Linear Regression (squared loss)

$$\frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^\top \phi(x_n)\}^2 + \frac{\lambda}{2} \sum_{d=1}^D w_d^2 = \frac{1}{2} (\mathbf{t} - \Phi \mathbf{w})^\top (\mathbf{t} - \Phi \mathbf{w}) + \frac{\lambda}{2} \mathbf{w}^\top \mathbf{w}$$

- Logistic Regression (log loss)

$$-\ln p(\mathbf{t} | \mathbf{w}) = -\sum_{n=1}^N \log (1 + \exp(-t_n \mathbf{w}^\top \phi(x_n))) + \frac{\lambda}{2} \mathbf{w}^\top \mathbf{w}$$

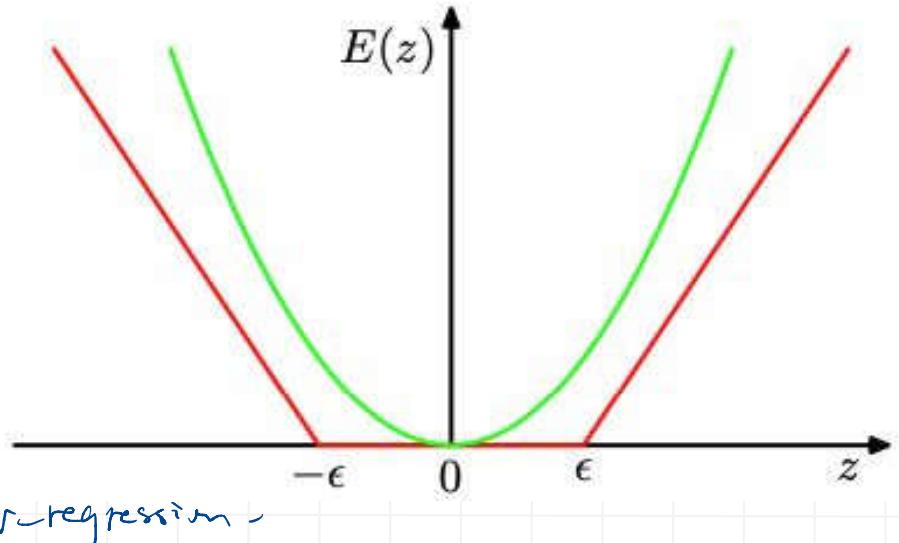
- Support Vector Machine (hinge loss)

$$C \sum_{n=1}^N [1 - t_n \mathbf{w}^\top \phi(x_n)]_+ + \frac{1}{2} \mathbf{w}^\top \mathbf{w}$$

← Covered in (ML?)

## $\epsilon$ -insensitive error function

**Figure 7.6** Plot of an  $\epsilon$ -insensitive error function (in red) in which the error increases linearly with distance beyond the insensitive region. Also shown for comparison is the quadratic error function (in green).



89, loss

$$\frac{1}{2} \sum_{n=1}^N \{y_n - t_n\}^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2.$$

$$(7.50) \longrightarrow C \sum_{n=1}^N E_\epsilon(y(\mathbf{x}_n) - t_n) + \frac{1}{2} \|\mathbf{w}\|^2 \quad (7.52)$$

$$E_\epsilon(y(\mathbf{x}) - t) = \begin{cases} 0, & \text{if } |y(\mathbf{x}) - t| < \epsilon; \\ |y(\mathbf{x}) - t| - \epsilon, & \text{otherwise} \end{cases} \quad (7.51)$$

# SVM regression with two set of slack variables

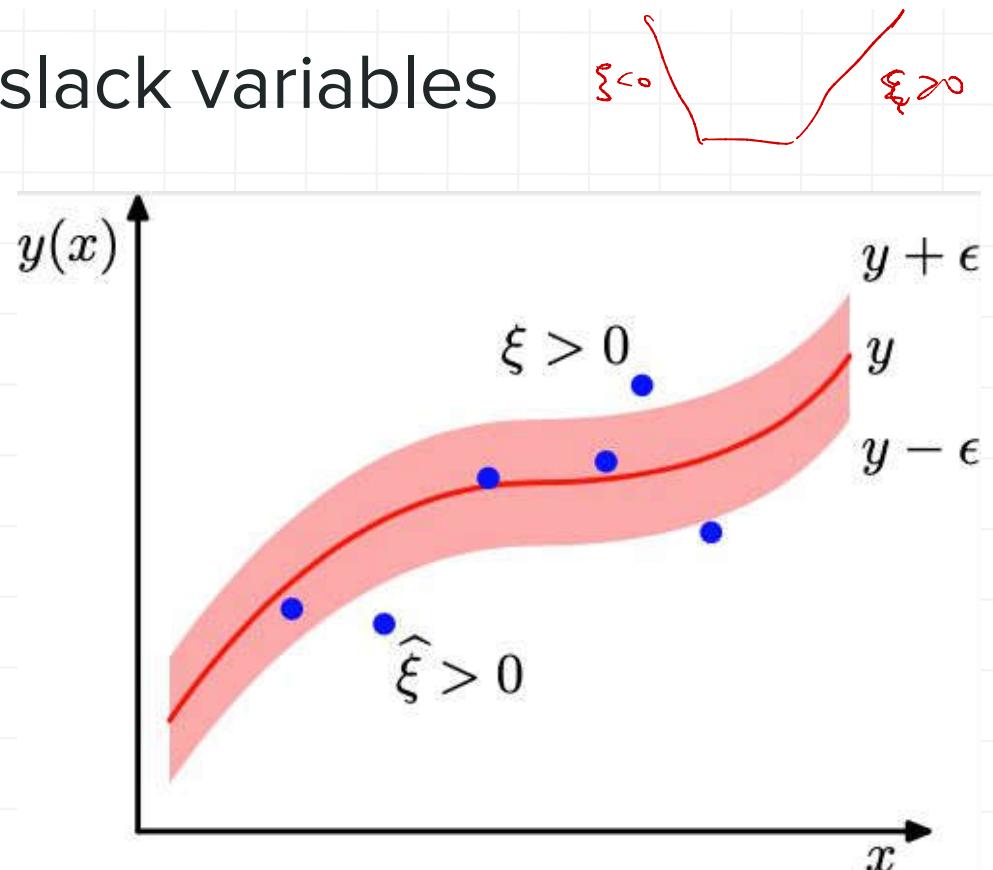
**Figure 7.7** Illustration of SVM regression, showing the regression curve together with the  $\epsilon$ -insensitive ‘tube’. Also shown are examples of the slack variables  $\xi$  and  $\hat{\xi}$ . Points above the  $\epsilon$ -tube have  $\xi > 0$  and  $\hat{\xi} = 0$ , points below the  $\epsilon$ -tube have  $\xi = 0$  and  $\hat{\xi} > 0$ , and points inside the  $\epsilon$ -tube have  $\xi = \hat{\xi} = 0$ .

$$C \sum_{n=1}^N (\xi_n + \hat{\xi}_n) + \frac{1}{2} \|\mathbf{w}\|^2$$

total slack

$$\begin{aligned} \tilde{L}(\mathbf{a}, \hat{\mathbf{a}}) &= -\frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N (a_n - \hat{a}_n)(a_m - \hat{a}_m) k(\mathbf{x}_n, \mathbf{x}_m) \\ &\quad - \epsilon \sum_{n=1}^N (a_n + \hat{a}_n) + \sum_{n=1}^N (a_n - \hat{a}_n) t_n \end{aligned} \tag{7.61}$$

see book

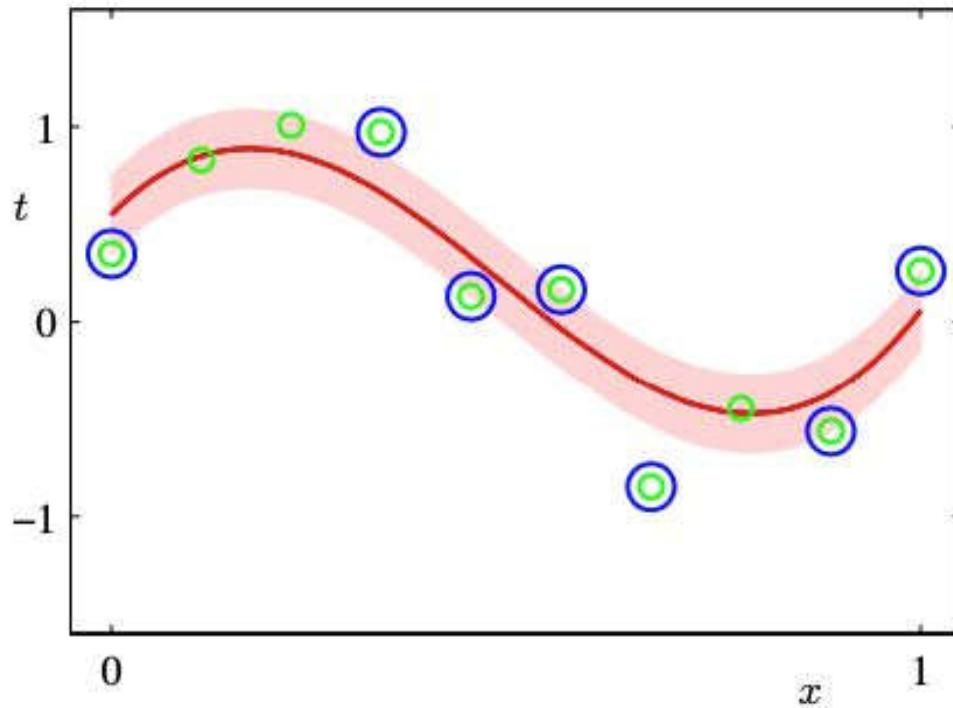


still uses kernels.

Prediction function

$$y(\mathbf{x}) = \sum_{n=1}^N (\underline{a_n} - \widehat{a}_n) k(\mathbf{x}, \mathbf{x}_n) + b \quad (7.64)$$

**Figure 7.8** Illustration of the  $\nu$ -SVM for regression applied to the sinusoidal synthetic data set using Gaussian kernels. The predicted regression curve is shown by the red line, and the  $\epsilon$ -insensitive tube corresponds to the shaded region. Also, the data points are shown in green, and those with support vectors are indicated by blue circles.



# Kernel machines

Lagrange multipliers, take 2

Maximum-margin classifiers (Chap 7.1)

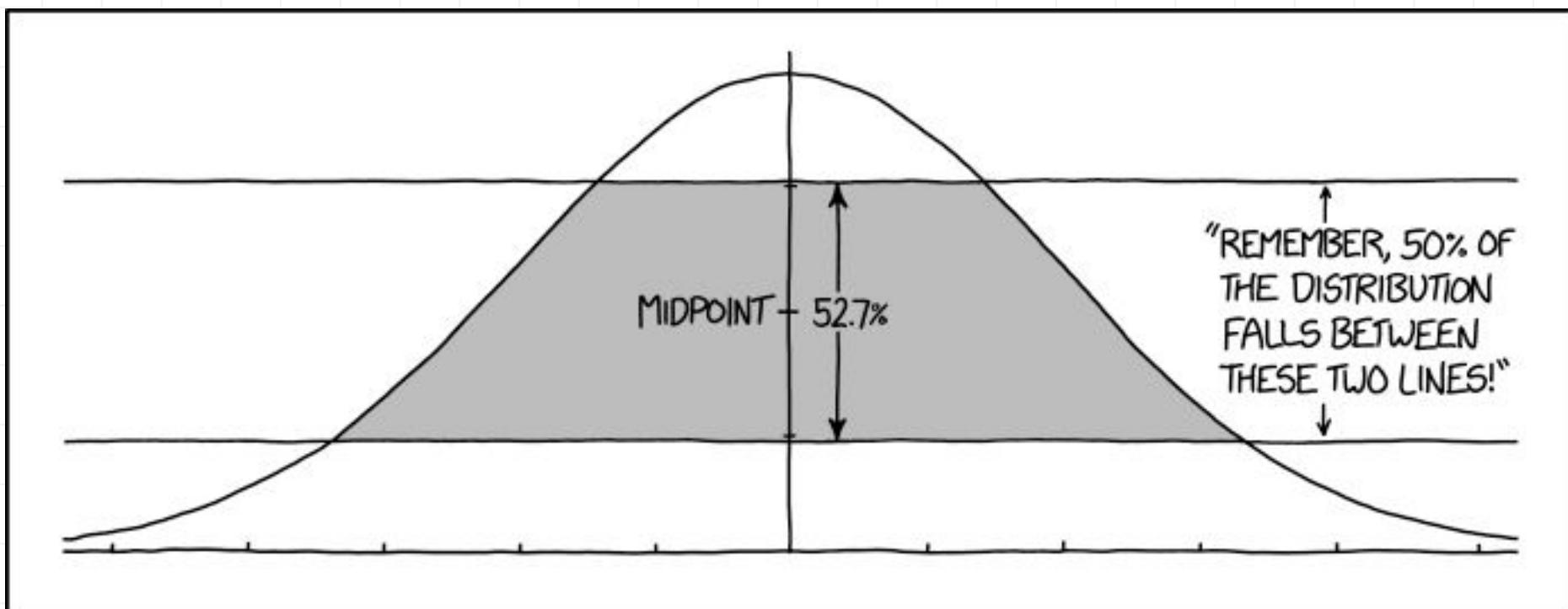
- The intuition of margins
- Constructing the max-margin classifier
- The dual representation
- Support vectors and their geometric intuitions

SVM for overlapping class distributions

Relations to logistic regression

SVM for regression

<https://xkcd.com/2118/>



HOW TO ANNOY A STATISTICIAN

# Announcements

Course schedule updated.

In person lectures - PSYC G8

Week 8 Wed - Gaussian Process 2

Week 10 Wed - Guest lecture: ML for Cyber security

- No tutorial in week 7 – assignment 1 walk-through during drop-in, will be recorded.
- No tutorial in week 8 – enjoy quiz 2 :)
  
- Video assignment released, due in week 12.
- Assignment 2 out soon.
- Quiz 2 released, 9 multiple choice questions.
- Assignment 1 raw grades out, regrade-request open on Gradescope for 2 weeks.

# Thank You for anonymous survey input

IML  $\rightarrow$  SML

Student A: Math too hard!

Student B: "Content is too easy. Does not feel challenging enough to be a 4000 level course.  
Make assessment more difficult, and material more in depth."

? skipped derivations / proofs

Thanks. Let's clarify some expectations for derivations / proofs.

Things that are important and appearing for the first time are generally covered; when a proof is skipped it usually means that we expect students to be able to follow the overall logic, and the model design arguments without the proof.

? motivating the material

Motivations tend to be mathematical in this course. We're happy to help in a number of ways.

? which parts are important

Good question. We'll try to clarify + reemphasize this in the intro/outro slides. If still unclear, do ask questions during class Q&A, during tuts, or on piazza.

# Gaussian Processes - Regression

Motivation

Defining Gaussian Processes (GP)

Kernel functions, sampling

GP regression

GP regression - predictive distribution

Sampling algorithm and computational costs

 in assignment 2

Bishop, Chap 6.4 (6.4.1-6.4.3)

GP book

<http://gaussianprocess.org/gpml/chapters/> Chap 1, 2.1, 2.2, 2.5

# Bayesian Joint Inversions for the Exploration of Earth Resources

Alistair Reid<sup>1</sup>, Simon O'Callaghan<sup>1</sup>, Edwin V. Bonilla<sup>1</sup>, Lachlan McCalman<sup>1</sup>, Tim Rawling<sup>2</sup> and Fabio Ramos<sup>3</sup>

1. NICTA, 2. University of Melbourne, 3. University of Sydney

{ alistair.reid, simon.ocallaghan, edwin.bonilla, lachlan.mccalman }@nicta.com.au

tim.rawling@unimelb.edu.au, f.ramos@acfr.usyd.edu.au

In a geological inversion problem, properties such as temperature, conductivity, density, magnetic susceptibility and permeability are inferred from related observations such as gravity, magnetics and seismic reflexion. ... In this paper we formulate geophysical inversion as a machine learning problem, and propose an approach based on Gaussian processes regression that naturally provides both a predictive distribution over the inverted quantities and a principled method to fuse different types of observations. We apply our method to a real dataset from South Australia containing gravity and drill-hole data with the goal of characterizing rock densities for geothermal target exploration, and also to simulated validation data involving gravity, drill-hole and magnetic observations.

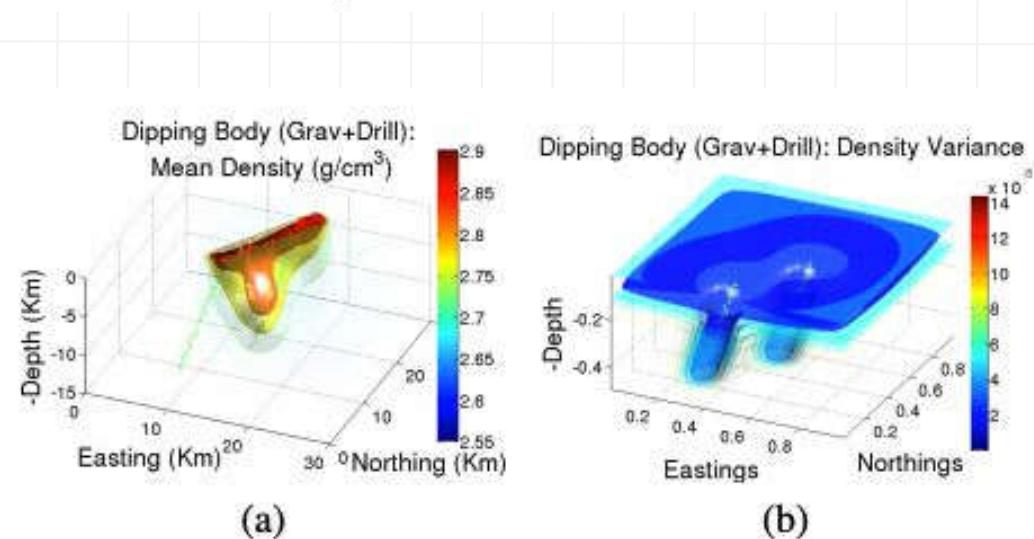


Figure 5: Outputs of the Gaussian process inversion algorithm after fusing gravity and drill observations of the dipping body.

# Two ways to learn a function (regression)

[GPbook, intro]

## Motivating scenarios

- SARCOS robotic arm
  - $x$ : (7 joint positions, 7 joint velocities, 7 joint accelerations)
  - $y$ : 7 joint torques
  - Learning:  $y \sim f(x)$
  - Prediction: compute the torque needed to move the arm along a given trajectory
- Predict the yield of a chemical plant ( $y$ ), given temperature, pressure, amount of catalyst, etc ( $x$ )
  - Hmm, we don't know a functional form of  $y$  (in either input or kernel space)
  - With full uncertainty estimate in  $y$

## Representation choices

- Restrict the class of functions we consider, e.g. linear
- Give prior probability to every possible function

we know how to do this :)

huh?

Un-countably infinite number of functions

Un-countably infinite number of input points (e.g.  $R$ ,  $R^2$ , ...)

Rely on parameter of the function, which can be a function itself

"Pretend" that it's a very long vector

# Prediction function for regression

$$\mathbf{w}^* = (\lambda \mathbf{I} + \Phi^\top \Phi)^{-1} \Phi^\top \mathbf{t}$$

*← normal eq.*

Prediction function for new  $\mathbf{x}$

$$y(\mathbf{x}) = \phi(\mathbf{x})^\top \mathbf{w}^* = \phi(\mathbf{x})^\top (\lambda \mathbf{I} + \Phi^\top \Phi)^{-1} \Phi^\top \mathbf{t}$$

Kernelized version

$$\begin{aligned} \mathbf{w} &= \Phi^\top \mathbf{a} \\ \underline{\mathbf{a}} &= (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t}. \end{aligned} \tag{6.8}$$

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

*point estimate of  $y$*   
*"Bayesian" view desires*  
 $P(y_{N+1} | X_{N+1}, X_{1:N}, Y_{1:N})$

What is the **Bayesian version** of this?

*→ expressed in  $K$*

# Bayesian linear regression (isotropic prior)

GP book  
2.1

if

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^N \mathcal{N}(t_n | \mathbf{w}^T \phi(\mathbf{x}_n), \underline{\beta^{-1}}) \quad (3.10)$$

def. "weight-space"

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w} | \mathbf{0}, \alpha^{-1} \mathbf{I}) \quad (3.52)$$

then

$$\underline{p(\mathbf{w}|\mathbf{t})} = \mathcal{N}(\mathbf{w} | \mathbf{m}_N, \mathbf{S}_N) \quad (3.49)$$

$$\left\{ \begin{array}{l} \mathbf{m}_N = \beta \mathbf{S}_N \Phi^T \mathbf{t} \\ \mathbf{S}_N^{-1} = \alpha \mathbf{I} + \beta \Phi^T \Phi. \end{array} \right. \quad (3.53) \quad \text{regression prob.}$$

$$(3.54)$$

learning : weights  
etc.

Prediction function

for new  $\mathbf{x}$

$$p(t|\mathbf{x}, \mathbf{t}, \alpha, \beta) = \mathcal{N}(t | \mathbf{m}_N^T \phi(\mathbf{x}), \sigma_N^2(\mathbf{x})) \quad (3.58)$$

$$\sigma_N^2(\mathbf{x}) = \frac{1}{\beta} + \phi(\mathbf{x})^T \mathbf{S}_N \phi(\mathbf{x}). \quad (3.59)$$

prediction?  
 $(x_{N+1}, t_{N+1})$   
?

Is there a kernelized version of this? – can I have both kernels and prior

$p(t_{N+1} | \dots)$

# Motivation

In week 6, we talked about kernels in non-probabilistic settings:

$$y(\mathbf{x}) = \mathbf{k}(\mathbf{x})^\top (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t}$$

*test data ,*  
*↓*

*↓*  
 $\Phi \Phi^\top$  *← training ,*

$$y(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b = \sum_{n=1}^N a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b$$

*↑*  
*↓*

Can one use kernels in a probabilistic setting?

"noiseless"

Starting again from linear regression ...

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) \quad (6.49)$$

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \mathbf{0}, \alpha^{-1} \mathbf{I}) \quad (6.50)$$

$$\mathbf{y} = \Phi \mathbf{w} \quad (6.51) \qquad \Phi_{nk} = \phi_k(\mathbf{x}_n)$$

*matrix form* *fixed* *Gaussian*

Claim:  $\mathbf{Y}$  is Gaussian ... only its mean and covariance matters.

$$\mathbb{E}[\mathbf{y}] = \Phi \mathbb{E}[\mathbf{w}] = \mathbf{0} \quad (6.52)$$

$$\text{cov}[\mathbf{y}] = \mathbb{E}[\mathbf{y}\mathbf{y}^T] = \Phi \mathbb{E}[\mathbf{w}\mathbf{w}^T] \Phi^T = \frac{1}{\alpha} \Phi \Phi^T = \mathbf{K} \quad (6.53) \quad p(\mathbf{y}) = \mathcal{N}(\mathbf{y} | \mathbf{0}, \mathbf{K})$$

$$K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m) = \frac{1}{\alpha} \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m) \quad (6.54)$$

$$\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \frac{1}{\alpha} \Phi \Phi^T)$$

Stochastic proc.  $P(y(t) | \theta)$

## Gaussian process (GP)

A Gaussian Process is defined as a probability distribution over functions  $y(x)$  such that the set of values of  $y(x)$  evaluated at an arbitrary set of points  $x_1, \dots, x_N$  jointly have a Gaussian distribution.

- More generally, a stochastic process  $y(x)$  is specified by giving the joint probability distribution for any finite set of values  $y(x_1), \dots, y(x_N)$  in a consistent manner.
- Common choice: set mean to zero, due to no prior knowledge of  $y(x)$   $E[y] = 0$
- The joint distribution of any  $y_1, \dots, y_N$  Fully specified by their second-order statistics
- Input  $x$  in 2-D – Gaussian random field (also called Kriging in statistics)

$x$  in 1-D – easy to visualise

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) \quad (6.49)$$

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \mathbf{0}, \alpha^{-1} \mathbf{I}) \quad (6.50)$$

$$\mathbf{y} = \Phi \mathbf{w} \quad (6.51)$$

$$\tilde{\mathbf{y}} \sim N(\tilde{\mathbf{0}}, \tilde{\mathbf{K}})$$

# Kernel functions in GP

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y} | \mathbf{0}, \mathbf{K})$$

$$\left. \begin{array}{l} y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) \\ \mathbf{y} = \Phi \mathbf{w} \\ p(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \mathbf{0}, \alpha^{-1} \mathbf{I}) \end{array} \right\} \quad (6.50)$$

Implicitly defined by  $\phi(\mathbf{x})$

$$K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m) = \frac{1}{\alpha} \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m) \quad (6.54)$$

One can also explicitly define a kernel

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2 / 2\sigma^2) \quad (6.23)$$

$$k(x, x') = \exp(-\theta |x - x'|) \quad (6.56)$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$$

linear kernel

$$k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}')$$

stationary kernel

$$k(\mathbf{x}, \mathbf{x}') = k(\|\mathbf{x} - \mathbf{x}'\|)$$

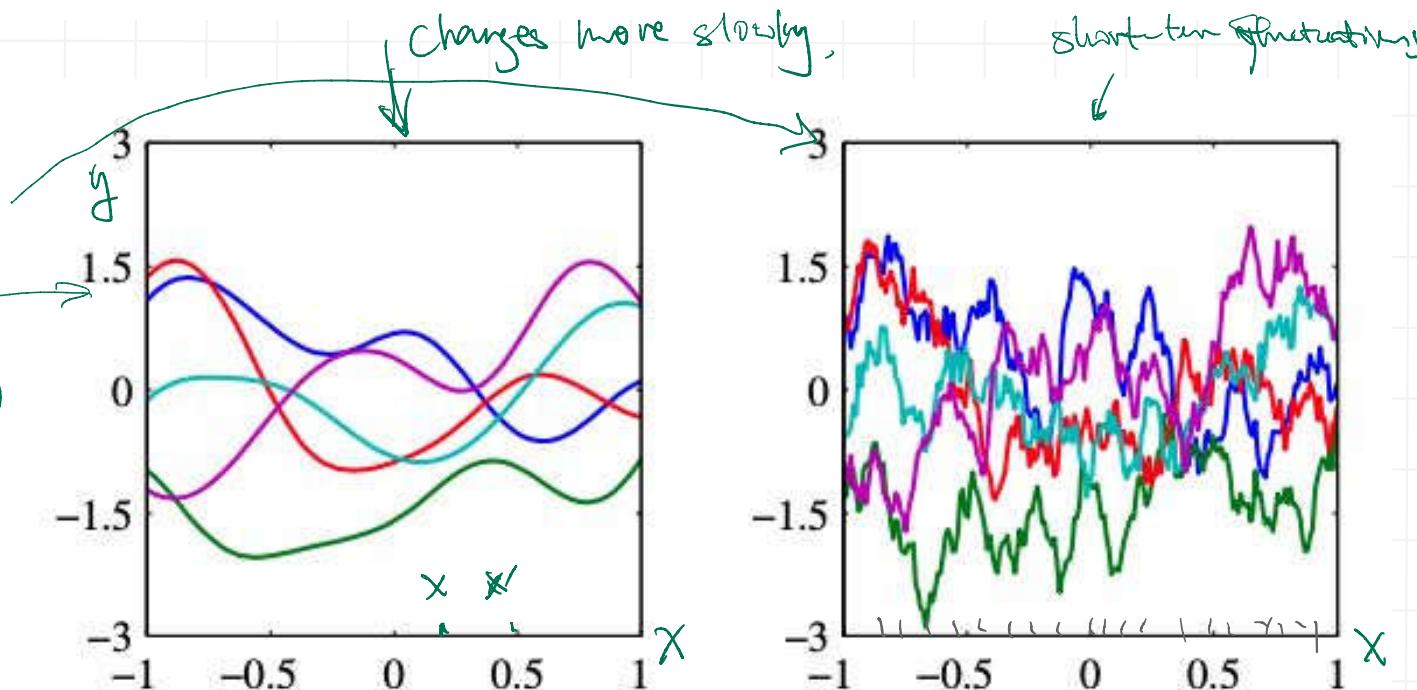
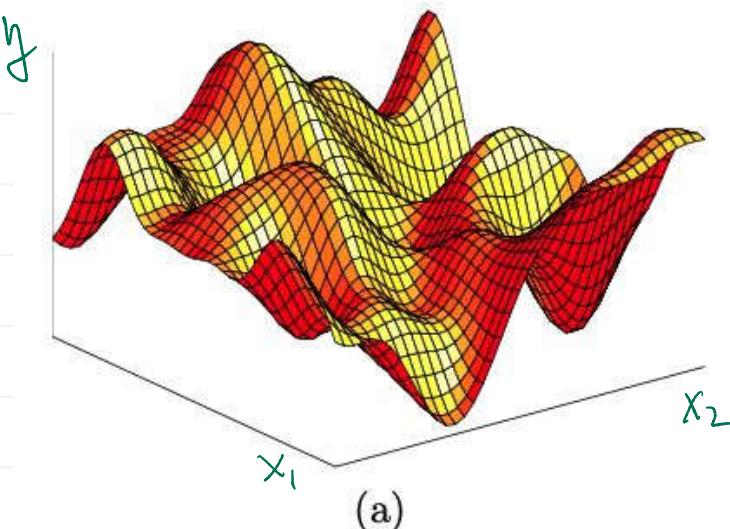
homogeneous kernel

The whole heap of kernel construction tricks apply ...

**Figure 6.4** Samples from Gaussian processes for a ‘Gaussian’ kernel (left) and an exponential kernel (right).

$$k(\mathbf{x}, \mathbf{x}') \quad \text{cov}(y(x), y(x'))$$

[GP book, Fig 1.2(a)]



$$k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2 / 2\sigma^2) \quad (6.23)$$

$$k(x, x') = \exp(-\theta |x - x'|) \quad (6.56)$$

$$y \sim N(\bar{\mu}, K)$$

Q: How to sample from a GP?  
How are these figures produced?

$$\{x_1, x_2, \dots, x_n\}$$

$$K \in \mathbb{R}^{n \times n}$$

[GP book, Sec A.2] <http://gaussianprocess.org/gpml/chapters/RWA.pdf>

To generate samples  $\mathbf{x} \sim \mathcal{N}(\mathbf{m}, K)$  with arbitrary mean  $\mathbf{m}$  and covariance matrix  $K$  using a scalar Gaussian generator (which is readily available in many programming environments) we proceed as follows: first, compute the Cholesky decomposition (also known as the “matrix square root”)  $L$  of the positive definite symmetric covariance matrix  $K = LL^\top$ , where  $L$  is a lower triangular matrix, see section A.4. Then generate  $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, I)$  by multiple separate calls to the scalar Gaussian generator. Compute  $\mathbf{x} = \mathbf{m} + L\mathbf{u}$ , which has the desired distribution with mean  $\mathbf{m}$  and covariance  $L\mathbb{E}[\mathbf{u}\mathbf{u}^\top]L^\top = LL^\top = K$  (by the independence of the elements of  $\mathbf{u}$ ).

In practice it may be necessary to add a small multiple of the identity matrix  $\epsilon I$  to the covariance matrix for numerical reasons. This is because the eigenvalues of the matrix  $K$  can decay very rapidly (see section 4.3.1 for a closely related analytical result) and without this stabilization the Cholesky decomposition fails. The effect on the generated samples is to add additional independent noise of variance  $\epsilon$ . From the context  $\epsilon$  can usually be chosen to have inconsequential effects on the samples, while ensuring numerical stability.

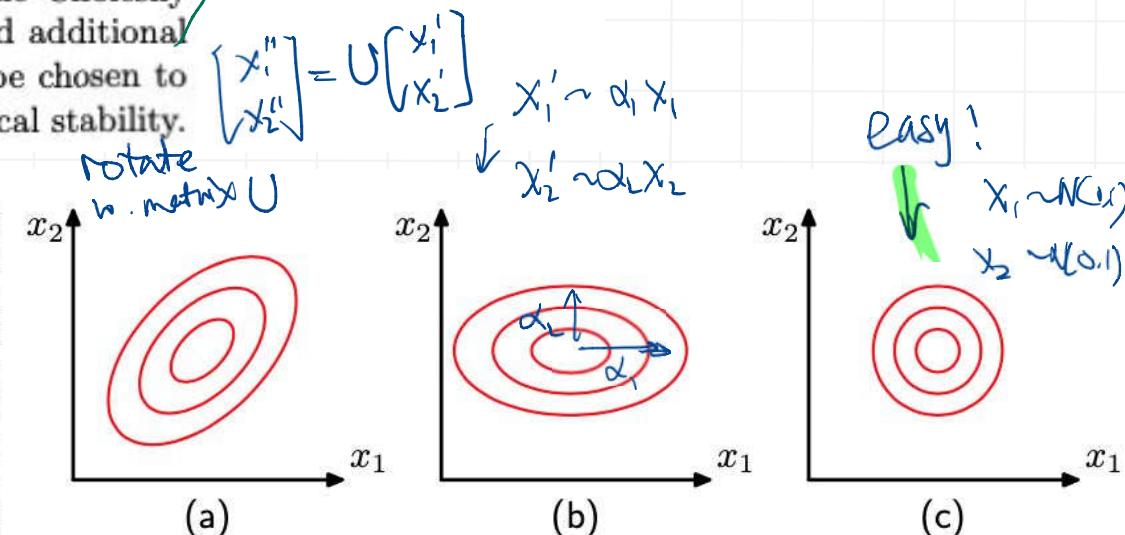
Intuition:

**Figure 2.8** Contours of constant probability density for a Gaussian distribution in two dimensions in which the covariance matrix is (a) of general form, (b) diagonal, in which the elliptical contours are aligned with the coordinate axes, and (c) proportional to the identity matrix, in which the contours are concentric circles.

generating multivariate Gaussian samples

`numpy.random.standard_normal`

`numpy.random.multivariate_normal`



Why Cholesky rather than eigen decomposition?

faster to compute!

$\mathcal{O}(n^3) \leftarrow$  Cholesky  
eigen val  
inverse

Cost of some matrix factorizations and decompositions.  $A$  is  $n \times n$ , except for QR and VD, where it is  $m \times n$  ( $m \geq n$ ).

Factorization/decomposition	Number of flops
LU factorization with partial pivoting ( $PA = LU$ )	$2n^3/3$
LU factorization with partial pivoting of upper Hessenberg matrix ( $PA = LU$ )	$n^2$
Cholesky factorization ( $A = R^*R$ )	$n^3/3$
Householder QR factorization ( $A = QR$ )	$2n^2(m - n/3)$ for $R$ ; $4(m^2n - mn^2 + n^3/3)$ for $m \times m Q$ ; $2n^2(m - n/3)$ for $m \times n Q$ ;
SVD <sup>a</sup> ( $A = P\Sigma Q^*$ )	$2np(2m - n)$ for $QB$ with $m \times p B$ and $Q$ held in factored form.
Hessenberg decomposition ( $A = QHQ^*$ )	$14mn^2 + 8n^3$ ( $P(:, 1:n)$ , $\Sigma$ , and $Q$ ) <sup>b</sup>
Schur decomposition <sup>a</sup> ( $A = QTQ^*$ )	$6mn^2 + 20n^3$ ( $P(:, 1:n)$ , $\Sigma$ , and $Q$ ) <sup>c</sup>
For Hermitian $A$ :	$14n^3/3$ ( $Q$ and $H$ ), $10n^3/3$ ( $H$ only)
Tridiagonal reduction ( $A = QTQ^*$ )	$25n^3$ ( $Q$ and $T$ ), $10n^3$ ( $T$ only)
Spectral decomposition ( $A = QDQ^*$ )	$8n^3/3$ ( $Q$ and $T$ ), $4n^3/3$ ( $T$ only) $9n^3$ ( $Q$ and $D$ ), $4n^3/3$ ( $D$ only)

<sup>a</sup>These costs are estimates taken from Golub and Van Loan [224, 1996].

<sup>b</sup>Golub–Reinsch SVD.

<sup>c</sup>Golub–Reinsch SVD with preliminary QR factorization.

# Gaussian Processes - Regression

Motivation

Defining Gaussian Processes (GP)

Kernel functions, sampling

GP regression

GP regression - predictive distribution

Sampling algorithm and computational costs

Bishop, Chap 6.4 (6.4.1-6.4.3)

GP book

<http://gaussianprocess.org/gpml/chapters/> Chap 1, 2.1, 2.2, 2.5

## GP for regression (w noise)

"noiseless"

$$\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \frac{1}{\alpha} \Phi \Phi^T)$$

$$\sim \mathcal{N}(0, k)$$

$$y_n = y(\mathbf{x}_n) \quad t_n = y_n + \epsilon_n \quad \epsilon_n \sim N(0, \beta^{-1}) \quad (6.57)$$

$$p(t_n|y_n) = \mathcal{N}(t_n|y_n, \beta^{-1}) \quad (6.58)$$

$$p(\mathbf{t}|\mathbf{y}) = \mathcal{N}(\mathbf{t}|\mathbf{y}, \beta^{-1} \mathbf{I}_N) \quad (6.59)$$

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}). \quad (6.60)$$

$$E(Y)=0 \quad E(\epsilon)=0$$

$$p(\mathbf{t}) = \int p(\mathbf{t}|\mathbf{y})p(\mathbf{y}) d\mathbf{y} = \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{C}) \quad (6.61)$$

$$C(\mathbf{x}_n, \mathbf{x}_m) = \underbrace{k(\mathbf{x}_n, \mathbf{x}_m)}_{\text{smooth}} + \underbrace{\beta^{-1} \delta_{nm}}_{\text{noise}}. \quad (6.62)$$

$$\mathbf{C} = \begin{pmatrix} k_{11} + \beta^{-1} & k_{12} & \cdots & \cdots \\ k_{21} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -k_{nn} + \beta^{-1} \end{pmatrix}$$

$$\mathbf{t} \sim \mathcal{N}(\mathbf{0}, \frac{1}{\beta} I_N + K)$$

# GP for regression

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}). \quad (6.60)$$

$$p(\mathbf{t}) = \int p(\mathbf{t}|\mathbf{y})p(\mathbf{y}) d\mathbf{y} = \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{C}) \quad (6.61)$$

$$C(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \beta^{-1}\delta_{nm}. \quad (6.62)$$

*training*

Let's explicitly define a kernel

$$k(\mathbf{x}_n, \mathbf{x}_m) = \theta_0 \exp \left\{ -\frac{\theta_1}{2} \|\mathbf{x}_n - \mathbf{x}_m\|^2 \right\} + \theta_2 + \theta_3 \mathbf{x}_n^T \mathbf{x}_m. \quad (6.63)$$

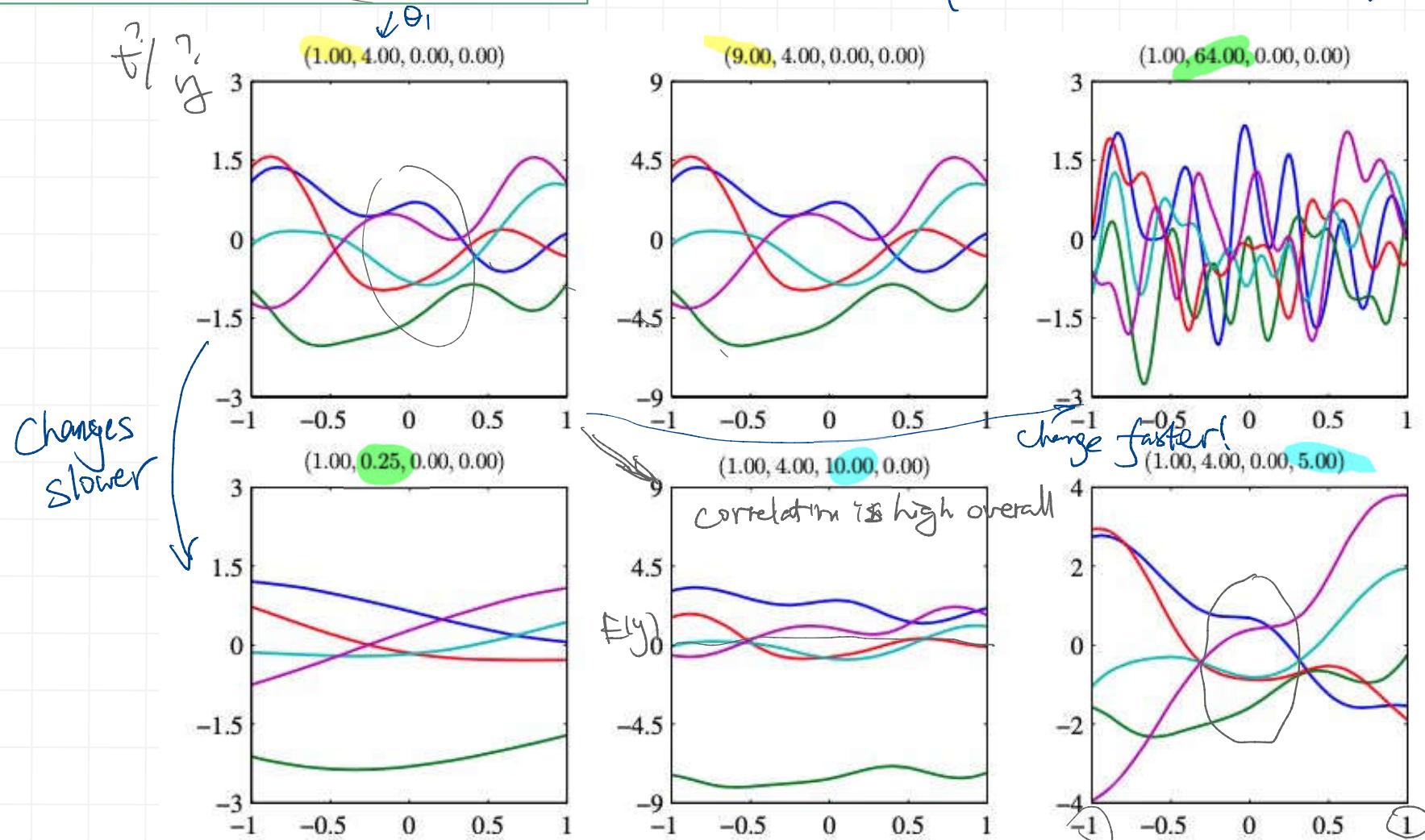
*Sq-exponential*      *const*      *linear*.

We covered representation and “training” (implicit) in GP.

$$k(\mathbf{x}_n, \mathbf{x}_m) = \theta_0 \exp \left\{ -\frac{\theta_1}{2} \|\mathbf{x}_n - \mathbf{x}_m\|^2 \right\} + \theta_2 + \theta_3 \mathbf{x}_n^T \mathbf{x}_m. \quad (6.63)$$

$$\theta_1 \sim \text{Uniform} \sim \theta^{0.5} \text{ scale} \downarrow$$

$y \sim N(0, K)$



**Figure 6.5** Samples from a Gaussian process prior defined by the covariance function (6.63). The title above each plot denotes  $(\theta_0, \theta_1, \theta_2, \theta_3)$ .

## Predictive distribution

$$p(t_{N+1} | \mathbf{t}_N)$$

$$p(\mathbf{t}_{N+1}) = \mathcal{N}(\mathbf{t}_{N+1} | \mathbf{0}, \mathbf{C}_{N+1})$$

$$\mathbf{C}_{N+1} = \begin{pmatrix} \mathbf{C}_N & \mathbf{k} \\ \mathbf{k}^T & c \end{pmatrix}$$

$\mathbf{k} \leftarrow \beta^{-1} \mathbf{I}_N$

kernel val of test pt  $\mathbf{x}_{N+1}$

(6.64)

$$P(\vec{t}_N) \sim \mathcal{N}(0, C_N)$$

(6.65)

vector  $\mathbf{k}$  has elements  $k(\mathbf{x}_n, \mathbf{x}_{N+1})$

$$c = k(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) + \beta^{-1}$$

$$\mu_{a|b} = \mu_a + \Sigma_{ab} \Sigma_{bb}^{-1} (\mathbf{x}_b - \mu_b) \quad (2.81)$$

$$\Sigma_{a|b} = \Sigma_{aa} - \Sigma_{ab} \Sigma_{bb}^{-1} \Sigma_{ba}. \quad (2.82)$$

The conditional mean and variance of  $t_{N+1}$  given  $\mathbf{t}_{1:N}$ , expressed as a function of  $\mathbf{x}_{N+1}$

$$m(\mathbf{x}_{N+1}) = \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{t} \quad \text{"training"}$$

test input, training input

$$\sigma^2(\mathbf{x}_{N+1}) = c - \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{k}.$$

variance reduction due to "training"

$$m(\mathbf{x}_{N+1}) = \sum_{n=1}^N a_n k(\mathbf{x}_n, \mathbf{x}_{N+1}) \quad (6.68)$$

$\mathbf{k}^T$

where  $a_n$  is the  $n^{\text{th}}$  component of  $\mathbf{C}_N^{-1} \mathbf{t}$ .

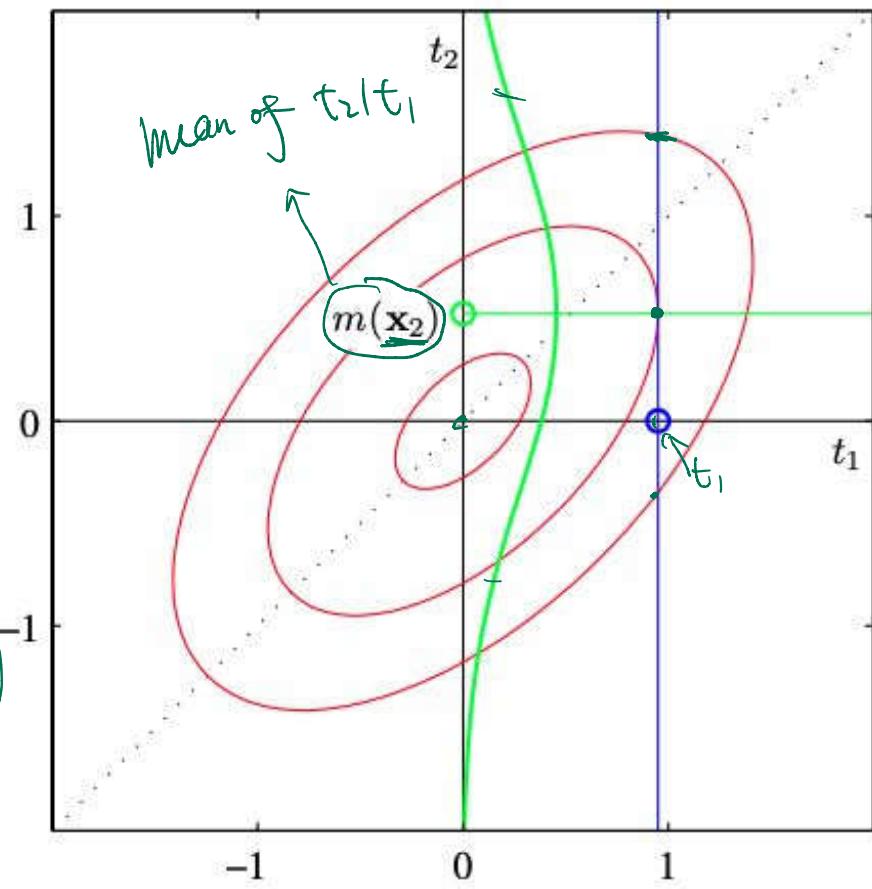
**Figure 6.7**

Illustration of the mechanism of Gaussian process regression for the case of one training point and one test point, in which the red ellipses show contours of the joint distribution  $p(t_1, t_2)$ . Here  $t_1$  is the training data point, and conditioning on the value of  $t_1$ , corresponding to the vertical blue line, we obtain  $p(t_2|t_1)$  shown as a function of  $t_2$  by the green curve.

$$p(\mathbf{t}_{N+1}) = \mathcal{N}(\mathbf{t}_{N+1} | \mathbf{0}, \mathbf{C}_{N+1})$$

$$P\left(\begin{bmatrix} t_1 \\ t_2 \end{bmatrix}\right) \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix}\right)$$

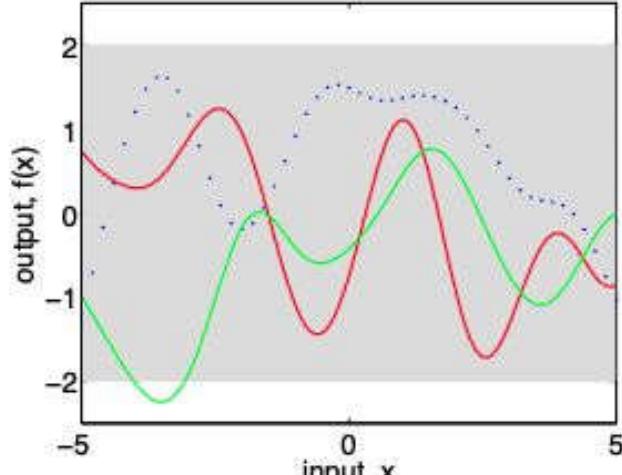
$$P(t_2|t_1)$$



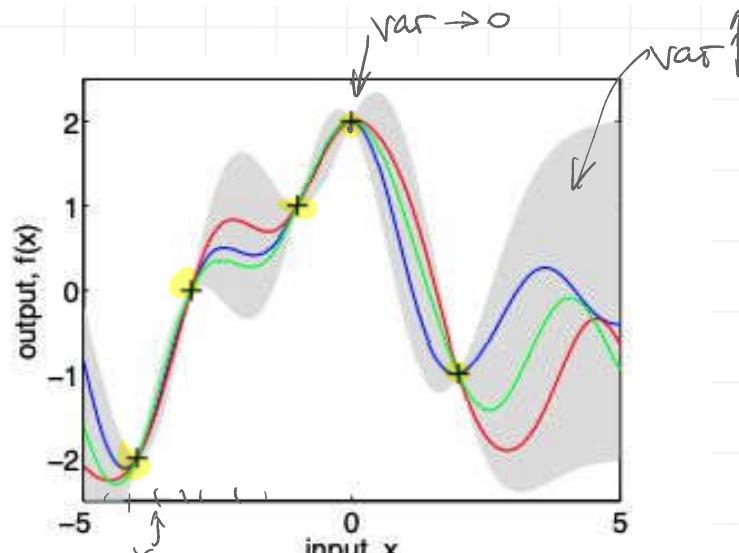
# Notations

	Bishop book	GP book
"train" input	$x, \phi(x)$ $\phi$	$x, X$ $\phi(x)$
Output	$y = \phi w$ $t = y + \epsilon$	$f(x) = \phi^T(x) w$ $y \sim N(f, \sigma_n^2)$
Kernel / cov-fn	$K, C_N, \vec{K} = [k(x_{N+1}, x_{1:N})]$	$K(x, x_*), K(x, X), K(x, x')$
"test" input	$x_{N+1}$	$x_*, x_*$
Output	$t_{N+1}$ (scalar) $w, m(x_{N+1}), \Gamma(x_{N+1})$	$\frac{f_*}{w, f_*} \quad V[f_*]$
Noise cov	$\beta^{-1}$	$\sigma_n^2$

## GP book Chap 2.2



(a), prior



(b), posterior

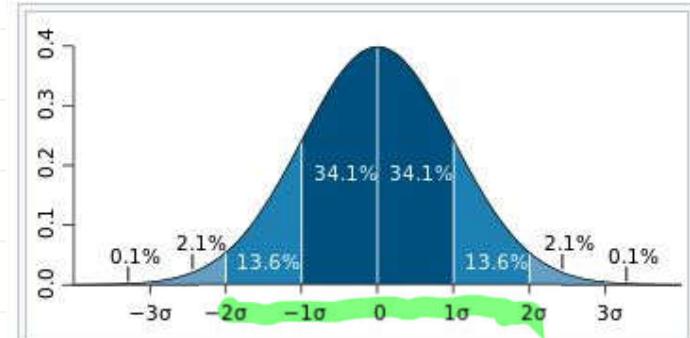
Figure 2.2: Panel (a) shows three functions drawn at random from a GP prior; the dots indicate values of  $y$  actually generated; the two other functions have (less correctly) been drawn as lines by joining a large number of evaluated points. Panel (b) shows three random functions drawn from the posterior, i.e. the prior conditioned on the five noise free observations indicated. In both plots the shaded area represents the pointwise mean plus and minus two times the standard deviation for each input value (corresponding to the 95% confidence region), for the prior and posterior respectively.

$$f_* | X_*, X, \mathbf{f} \sim \mathcal{N}(K(X_*, X) K(X, X)^{-1} \mathbf{f}, K(X_*, X_*) - K(X_*, X) K(X, X)^{-1} K(X, X_*)). \quad (2.19)$$

$$\underline{m(\mathbf{x}_{N+1})} = \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{t} \quad (6.66)$$

$$\sigma^2(\mathbf{x}_{N+1}) = c - \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{k}. \quad (6.67)$$

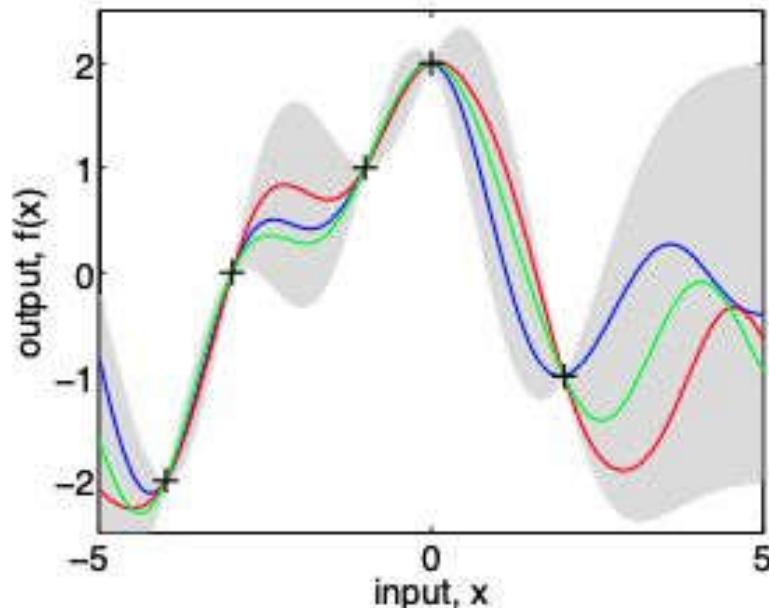
Noise-less:  $\mathbf{C}_N = \mathbf{K}_N$



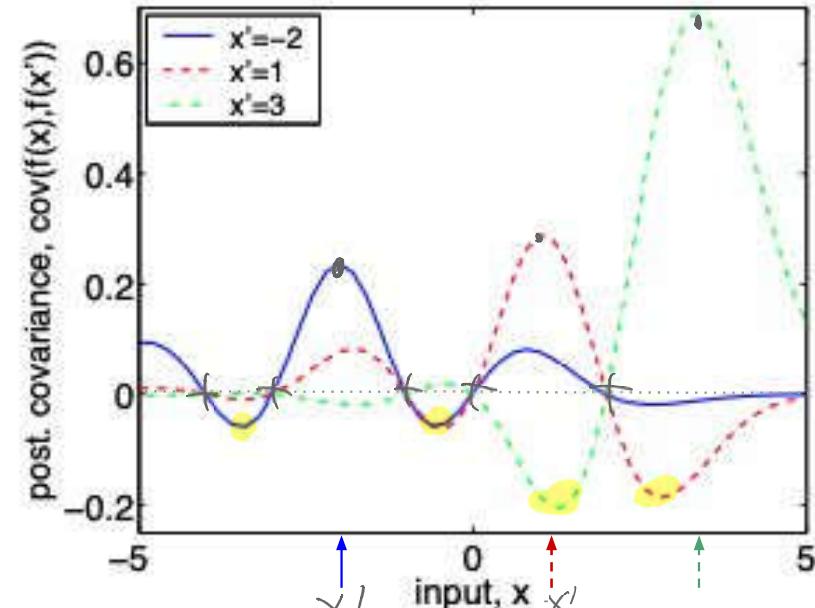
For the normal distribution, the values less than one standard deviation away from the mean account for 68.27% of the set; while two standard deviations from the mean account for 95.45%; and three standard deviations account for 99.73%.  $\square$



$\nearrow$  g. reched by  $\mathcal{O}_{\mathbf{x} \mid \mathbf{x}^n}$ .  $K(\mathbf{x}, \mathbf{x}')$



(a), posterior



(b), posterior covariance

Figure 2.4: Panel (a) is identical to Figure 2.2(b) showing three random functions drawn from the posterior. Panel (b) shows the posterior co-variance between  $f(\mathbf{x})$  and  $f(\mathbf{x}')$  for the same data for three different values of  $\mathbf{x}'$ . Note, that the covariance at close points is high, falling to zero at the training points (where there is no variance, since it is a noise-free process), then becomes negative, etc. This happens because if the smooth function happens to be less than the mean on one side of the data point, it tends to exceed the mean on the other side, causing a reversal of the sign of the covariance at the data points. Note for contrast that the prior covariance is simply of Gaussian shape and never negative.

GP book Chap 2.2

$$k_y(x_p, x_q) = \sigma_f^2 \exp\left(-\frac{1}{2\ell^2}(x_p - x_q)^2\right) + \sigma_n^2 \delta_{pq}.$$

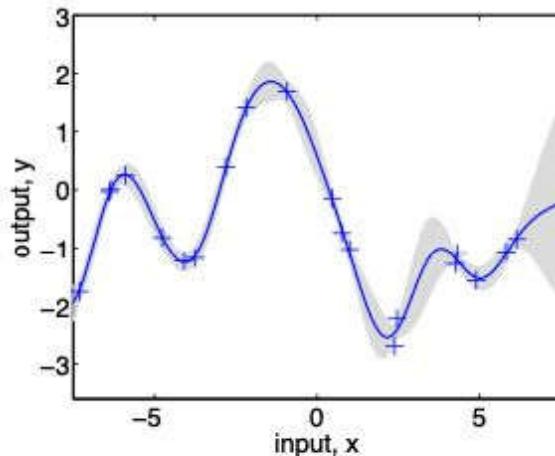
noise var.  
↓

(2.31)

① Sampling w varying kernel param.

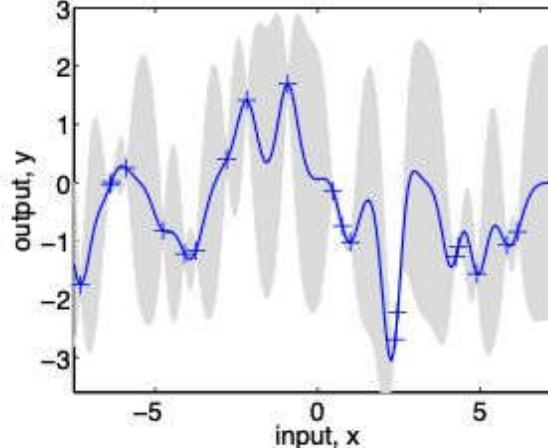
② posterior distribution

given input data.

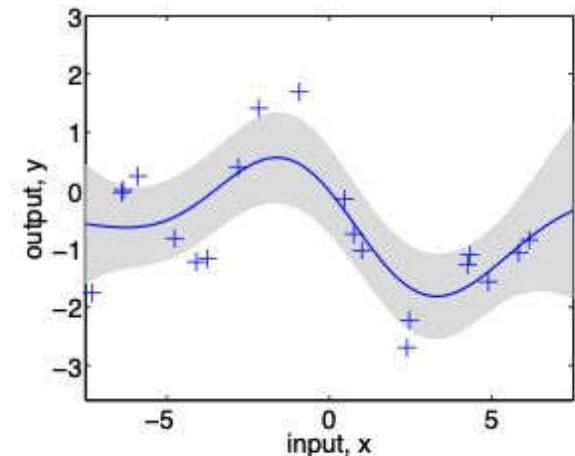


(a),  $\ell = 1$

noise (rw)  
↓



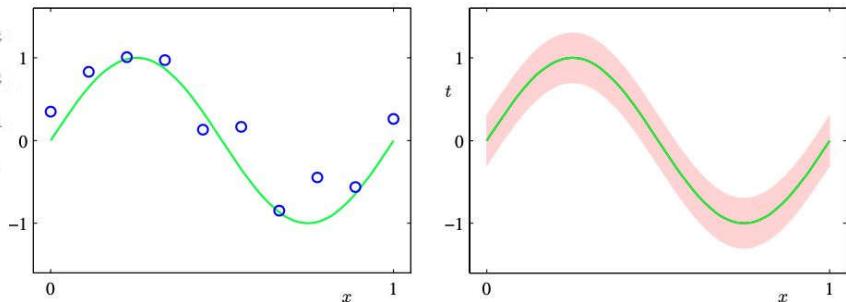
(b),  $\ell = 0.3$



(c),  $\ell = 3$

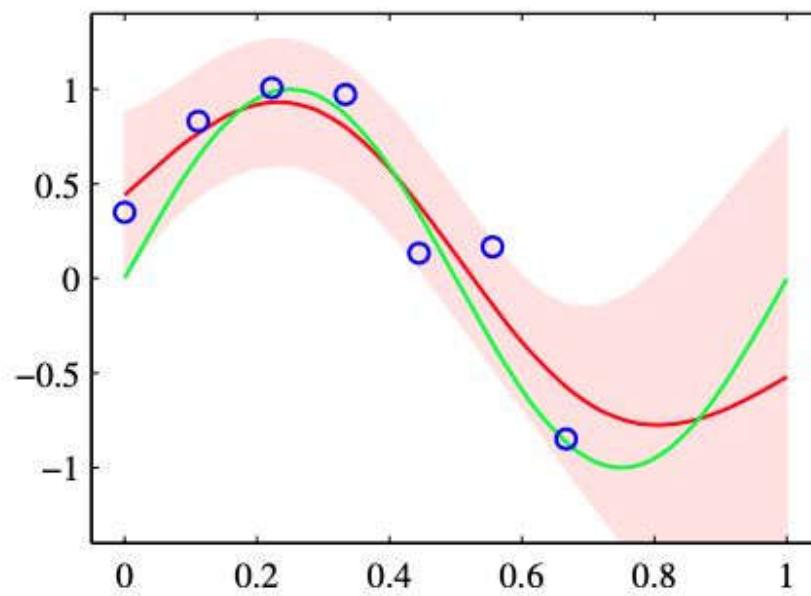
Figure 2.5: (a) Data is generated from a GP with hyperparameters  $(\ell, \sigma_f, \sigma_n) = (1, 1, 0.1)$ , as shown by the + symbols. Using Gaussian process prediction with these hyperparameters we obtain a 95% confidence region for the underlying function  $f$  (shown in grey). Panels (b) and (c) again show the 95% confidence region, but this time for hyperparameter values  $(0.3, 1.08, 0.00005)$  and  $(3.0, 1.16, 0.89)$  respectively.

tion, shown in Figure A.6. The input values  $\{x_n\}$  are generated uniformly in range  $(0, 1)$ , and the corresponding target values  $\{t_n\}$  are obtained by first computing the corresponding values of the function  $\sin(2\pi x)$ , and then adding random noise with a Gaussian distribution having standard deviation 0.3. Various forms of this data set,



**Figure A.6** The left-hand plot shows the synthetic regression data set along with the underlying sinusoidal function from which the data points were generated. The right-hand plot shows the true conditional distribution  $p(t|x)$  from which the labels are generated, in which the green curve denotes the mean, and the shaded region spans one standard deviation on each side of the mean.

**Figure 6.8** Illustration of Gaussian process regression applied to the sinusoidal data set in Figure A.6 in which the three right-most data points have been omitted. The green curve shows the sinusoidal function from which the data points, shown in blue, are obtained by sampling and addition of Gaussian noise. The red line shows the mean of the Gaussian process predictive distribution, and the shaded region corresponds to plus and minus two standard deviations. Notice how the uncertainty increases in the region to the right of the data points.



$$\sigma^2(\mathbf{x}_{N+1}) = c - \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{k}. \quad (6.67)$$

$$C(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \beta^{-1} \delta_{nm}. \quad (6.62)$$

$$c = k(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) + \beta^{-1}$$

[Bishop 6.4]

Implementing

$$m(\mathbf{x}_{N+1}) = \mathbf{k}^T \underline{\mathbf{C}_N^{-1} \mathbf{t}} \quad (6.66)$$

$$\sigma^2(\mathbf{x}_{N+1}) = c - \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{k}. \quad (6.67)$$

GP book Chap 2.2, A.4

**input:**  $X$  (inputs),  $\mathbf{y}$  (targets),  $k$  (covariance function),  $\sigma_n^2$  (noise level),  
 $\mathbf{x}_*$  (test input)

2:  $L := \text{cholesky}(K + \sigma_n^2 I) \quad \xrightarrow{\mathbf{C}_N}$   
 $\alpha := L^T \backslash (L \backslash \mathbf{y}) \quad \leftarrow \mathbf{C}_N^T$   
4:  $\bar{f}_* := \mathbf{k}_*^T \alpha$   
 $\mathbf{v} := L \backslash \mathbf{k}_*$   
6:  $\mathbb{V}[f_*] := k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^T \mathbf{v} \quad \} \text{ predictive mean eq. (2.25)}$   
 $\log p(\mathbf{y}|X) := -\frac{1}{2} \mathbf{y}^T \alpha - \sum_i \log L_{ii} - \frac{n}{2} \log 2\pi \quad \text{eq. (2.30)}$   
8: **return:**  $\bar{f}_*$  (mean),  $\mathbb{V}[f_*]$  (variance),  $\log p(\mathbf{y}|X)$  (log marginal likelihood)

Algorithm 2.1: Predictions and log marginal likelihood for Gaussian process regression. The implementation addresses the matrix inversion required by eq. (2.25) and (2.26) using Cholesky factorization, see section A.4. For multiple test cases lines 4-6 are repeated. The log determinant required in eq. (2.30) is computed from the Cholesky factor (for large  $n$  it may not be possible to represent the determinant itself). The computational complexity is  $n^3/6$  for the Cholesky decomposition in line 2, and  $n^2/2$  for solving triangular systems in line 3 and (for each test case) in line 5.

Note also that the determinant of a positive definite symmetric matrix can be calculated efficiently by

$$|A| = \prod_{i=1}^n L_{ii}^2, \quad \text{or} \quad \log |A| = 2 \sum_{i=1}^n \log L_{ii}, \quad (\text{A.18})$$

where  $L$  is the Cholesky factor from  $A$ .

$A = LL^T$ ,  $L$  lower-triangular

To solve  $Ax = b$

i.e.  $x = A \backslash b$  or  $A^{-1}b$  for square  $A$

First solve  $Ly = b$ ,  $y = L \backslash b$

Then  $L^T x = y$

$x = L^T \backslash y = L^T \backslash (L \backslash b)$

$$Lv = K*$$

$$V^T V = K^T C_N^{-1} K$$

# GP vs Bayesian linear regression

$$m(\mathbf{x}_{N+1}) = \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{t} \quad (6.66)$$

$$\sigma^2(\mathbf{x}_{N+1}) = c - \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{k}. \quad (6.67)$$

Linear regression w basis functions

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N) \quad (3.49)$$

$$\mathbf{m}_N = \mathbf{S}_N (\mathbf{S}_0^{-1} \mathbf{m}_0 + \beta \Phi^T \mathbf{t}) \quad (3.50)$$

$$\mathbf{S}_N^{-1} = \mathbf{S}_0^{-1} + \beta \Phi^T \Phi. \quad (3.51)$$

Bottleneck:

compute  $\mathbf{C}_N^{-1}$ ,  $O(N^3)$  - at training time  
 $O(N^2)$  for each test point

Bottleneck:

compute  $\mathbf{S}_N$ ,  $O(M^3)$  - at training time  
 $O(M^2)$  for each test point

# An example application

GP book Chap 2.5

- SARCOS robotic arm
  - $x$ : (7 joint positions, 7 joint velocities, 7 joint accelerations)
  - $y$ : 7 joint torques
  - Learning:  $y \sim f(x)$
  - Prediction: compute the torque needed to move the arm along a given trajectory

Squared exponential covariance function

- Separate length scales for each input dim,  $\sigma_f^2, \sigma_n^2$
- Optimise marginal likelihood on subset of data

~49K input-output pairs

- 44,484 for training
- 4,449 for testing

"inverting" (cholesky)  
49K x 49K

Method	SMSE	MSLL
LR	0.075	-1.29
RBD	0.104	-
LWPR	0.040	-
GPR	0.011	-2.25

Table 2.1: Test results on the inverse dynamics problem for a number of different methods. The “–” denotes a missing entry, caused by two methods not producing full predictive *distributions*, so MSLL could not be evaluated.

Evaluation metrics

- SMSE - Standardized mean square error
- MSLL - Mean standardised log loss

\*speed up with approximations, see Table 8.1

# Learning hyperparameters

$$p(\mathbf{t}) = \int p(\mathbf{t}|\mathbf{y})p(\mathbf{y}) d\mathbf{y} = \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{C}) \quad (6.61)$$

$$\ln p(\mathbf{t}|\boldsymbol{\theta}) = -\frac{1}{2} \ln |\mathbf{C}_N| - \frac{1}{2} \mathbf{t}^T \mathbf{C}_N^{-1} \mathbf{t} - \frac{N}{2} \ln(2\pi). \quad (6.69)$$

Gaussian  
(likelihood)

$$\frac{\partial}{\partial x} \ln |\mathbf{A}| = \text{Tr} \left( \mathbf{A}^{-1} \frac{\partial \mathbf{A}}{\partial x} \right) \quad (C.22)$$

$$\frac{\partial}{\partial x} (\mathbf{A}^{-1}) = -\mathbf{A}^{-1} \frac{\partial \mathbf{A}}{\partial x} \mathbf{A}^{-1} \quad (C.21)$$

$$\frac{\partial}{\partial \theta_i} \ln p(\mathbf{t}|\boldsymbol{\theta}) = -\frac{1}{2} \text{Tr} \left( \mathbf{C}_N^{-1} \frac{\partial \mathbf{C}_N}{\partial \theta_i} \right) + \frac{1}{2} \underbrace{\mathbf{t}^T \mathbf{C}_N^{-1}}_{\text{green bracket}} \underbrace{\frac{\partial \mathbf{C}_N}{\partial \theta_i}}_{\text{yellow circle}} \underbrace{\mathbf{C}_N^{-1} \mathbf{t}}_{\text{green bracket}}. \quad (6.70)$$

# Gaussian Processes - Regression

Motivation

Defining Gaussian Processes (GP)

Kernel functions, sampling

GP regression

GP regression - predictive distribution

Sampling algorithm and computational costs

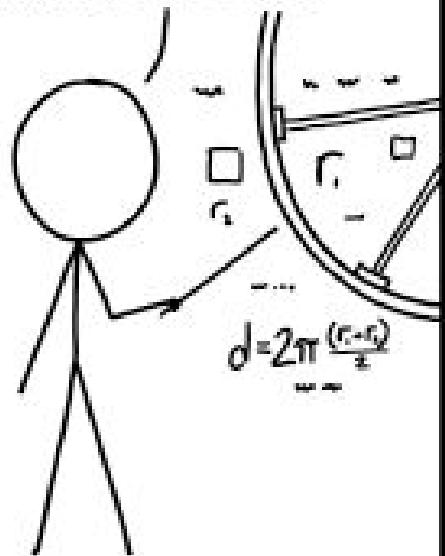
Bishop, Chap 6.4 (6.4.1-6.4.3)

GP book

<http://gaussianprocess.org/gpml/chapters/> Chap 1, 2.1, 2.2, 2.5

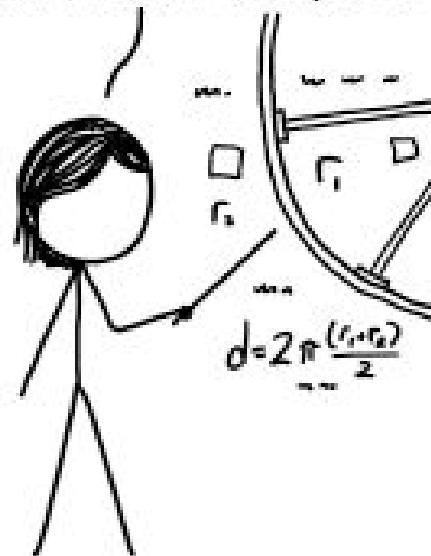
PHYSICIST  
APPROXIMATIONS

WE'LL ASSUME THE  
CURVE OF THIS RAIL  
IS A CIRCULAR ARC  
WITH RADIUS  $R$ .



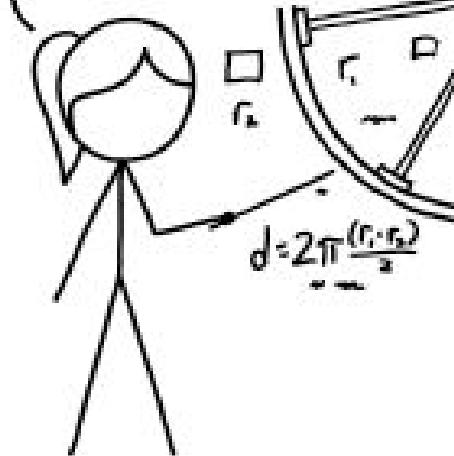
ENGINEER APPROXIMATIONS

LET'S ASSUME THIS  
CURVE DEVIATES FROM  
A CIRCLE BY NO MORE  
THAN 1 PART IN 1,000.



COSMOLOGIST APPROXIMATIONS

ASSUME PI IS ONE.  
PRETTY SURE IT'S  
BIGGER THAN THAT.  
OK, WE CAN MAKE  
IT TEN. WHATEVER.



# Announcements

Released: Quiz 2, Assignment 2, video assignment

Next three lectures: graphical models

Week 10 Wed: guest lecture

# Approximate Inference + GP Classification

Laplace approximation - in general

Bishop, Chap 4.4, 4.5  
6.4 (6.4.5, part of 6.4.6, 6.4.7)

Laplace approximation - Bayesian logistic regression

GP book *chap 3*  
<http://gaussianprocess.org/gpml/chapters/>

GP classification

Laplace approximation - GP classification

Connection to neural networks

## Laplace approximation in general

[Bishop 4.4]

Goal: find a Gaussian approximation to a probability density defined over a set of continuous variables.

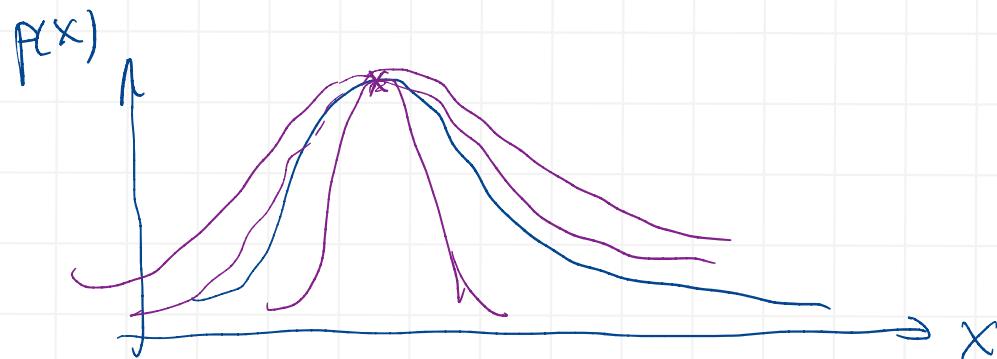
How: find Gaussian pdf  $\underline{q(z)}$ , centred on a mode of the distribution  $\underline{p(z)}$

Consider pdf

$$\underline{p(z)} = \frac{1}{Z} \underline{f(z)}$$

(4.125)

where  $Z = \int f(z) dz$



Gaussian pdf  $\Leftrightarrow$   $\ln(f(z))$  quadratic

$$f(z)$$

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\} \quad (2.42)$$



$$p(z) = \frac{1}{Z} f(z)$$

$$(4.125) \quad \text{where } Z = \int f(z) dz$$

Find mode  $z_0$

$$\frac{df(z)}{dz}\Big|_{z=z_0} = 0. \quad (4.126)$$

$$\text{SCC} |_{z_0} \quad f(z_0) + \frac{1}{1!} f'(z_0) \frac{(x-z_0)}{1!} + \frac{1}{2!} f''(z_0) (x-z_0)^2$$

$$\frac{d \ln(f(z))}{dz} = \frac{1}{f(z)} \frac{df(z)}{dz}$$

Taylor expansion  
of  $\ln f(z)$  at  $z_0$

$$\ln f(z) \simeq \ln f(z_0) - \frac{1}{2} A(z - z_0)^2 \quad (4.127)$$

$$A = -\frac{d^2}{dz^2} \ln f(z)\Big|_{z=z_0}. \quad (4.128)$$

Take exp()

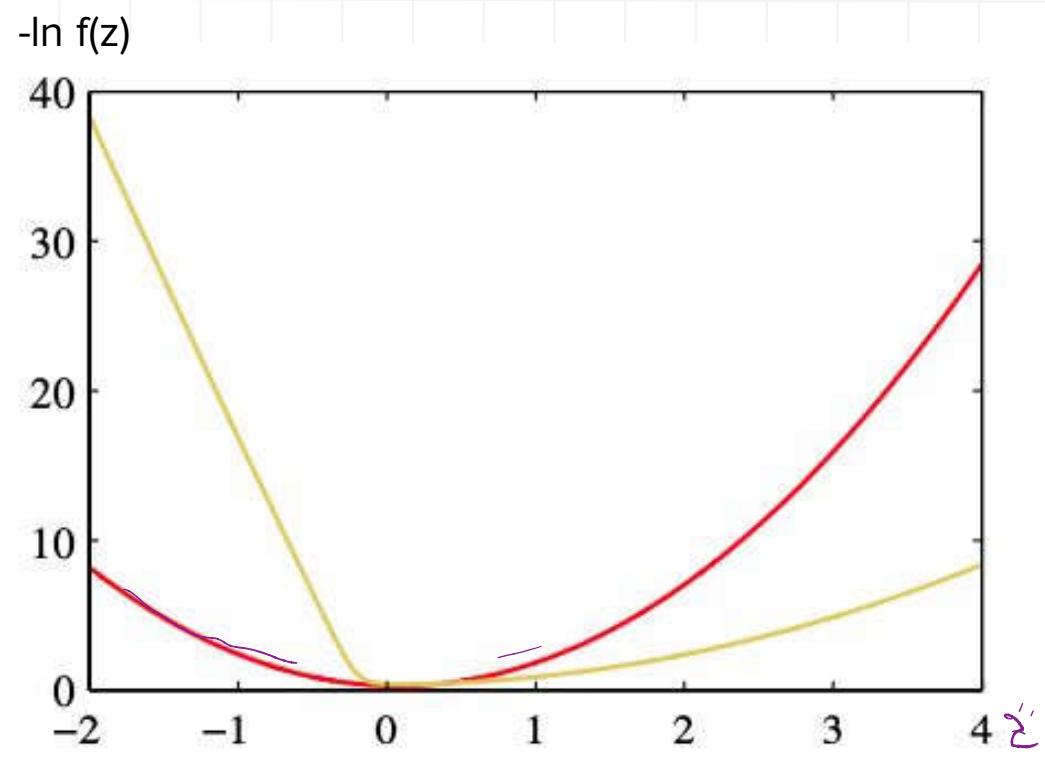
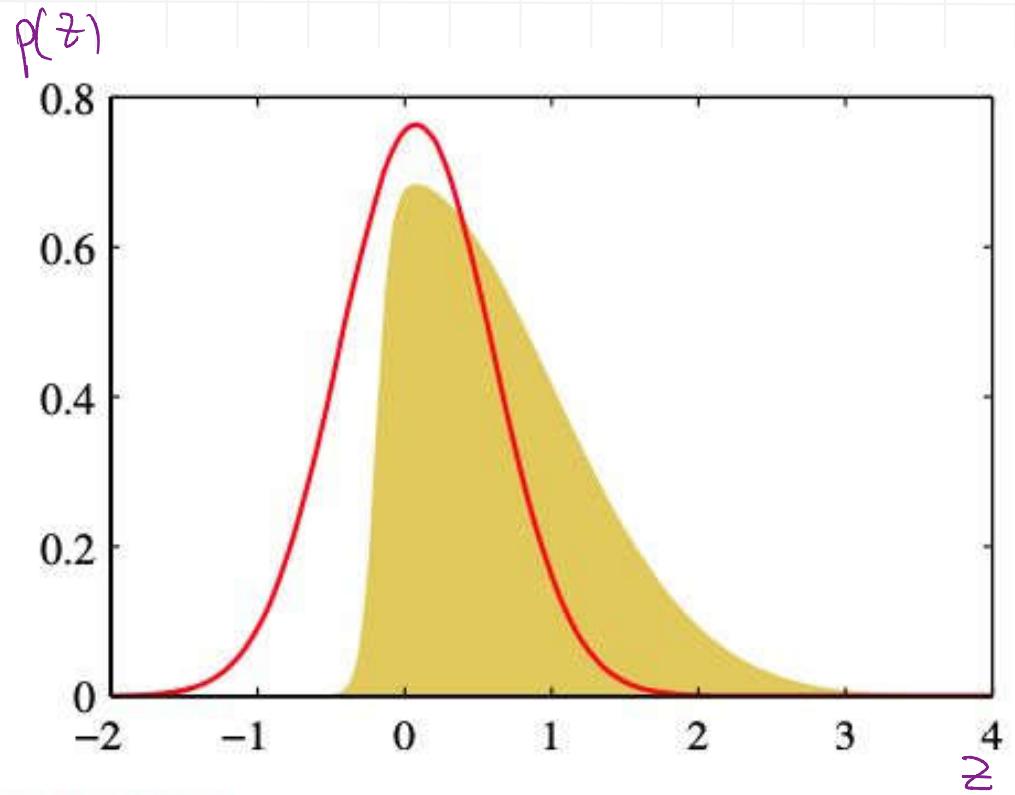
(4.127)

$$f(z) \simeq f(z_0) \exp\left\{-\frac{A}{2}(z - z_0)^2\right\}. \quad (4.129)$$

Normalise to  
obtain  $q(z)$

$$q(z) = \left(\frac{A}{2\pi}\right)^{1/2} \exp\left\{-\frac{A}{2}(z - z_0)^2\right\}. \quad (4.130)$$

Assume  $A > 0$



**Figure 4.14** Illustration of the Laplace approximation applied to the distribution  $p(z) \propto \exp(-z^2/2)\sigma(20z+4)$  where  $\sigma(z)$  is the logistic sigmoid function defined by  $\sigma(z) = (1 + e^{-z})^{-1}$ . The left plot shows the normalized distribution  $p(z)$  in yellow, together with the Laplace approximation centred on the mode  $z_0$  of  $p(z)$  in red. The right plot shows the negative logarithms of the corresponding curves.

## Laplace approximation in higher dimensions

$$P(\vec{z}) = \frac{1}{Z} f(\vec{z})$$

$f(z)$ , with  $z_0$  a stationary point  $\nabla f(z)|_{z=z_0} = 0$

Taylor expansion of  $\ln f(z)$  around  $z_0$

$$\ln f(z) \simeq \ln f(z_0) - \frac{1}{2}(z - z_0)^T \mathbf{A}(z - z_0) \quad (4.131)$$

$\overbrace{\hspace{1cm}}$        $\overset{\text{matrix}}{\uparrow}$        $\boxed{\hspace{1cm}}$

where the  $M \times M$  Hessian matrix  $\mathbf{A}$  is defined by

$$\mathbf{A} = -\nabla\nabla \ln f(z)|_{z=z_0} \quad (4.132)$$

and  $\nabla$  is the gradient operator. Taking the exponential of both sides we obtain

$$f(z) \simeq f(z_0) \exp \left\{ -\frac{1}{2}(z - z_0)^T \mathbf{A}(z - z_0) \right\}. \quad (4.133)$$

$\overset{\text{normalise}}{\uparrow}$

# Laplace approximation in higher dimensions

$$f(\mathbf{z}) \simeq f(\mathbf{z}_0) \exp \left\{ -\frac{1}{2} (\mathbf{z} - \mathbf{z}_0)^T \mathbf{A} (\mathbf{z} - \mathbf{z}_0) \right\}. \quad (4.133)$$

normalise, use (2.43)

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\}$$

$$q(\mathbf{z}) = \underbrace{\frac{|\mathbf{A}|^{1/2}}{(2\pi)^{M/2}}}_{\text{normalising constant}} \exp \left\{ -\frac{1}{2} (\mathbf{z} - \mathbf{z}_0)^T \mathbf{A} (\mathbf{z} - \mathbf{z}_0) \right\} = \mathcal{N}(\mathbf{z}|\mathbf{z}_0, \mathbf{A}^{-1}) \quad (4.134)$$

$q(\mathbf{z})$  is a valid multivariate Gaussian distribution iff A positive semi-definite  
→  $\mathbf{z}_0$  is a local maximum, not local min or saddle point

What about  $f(z)$  with multiple modes? Different Laplace approximations for each mode

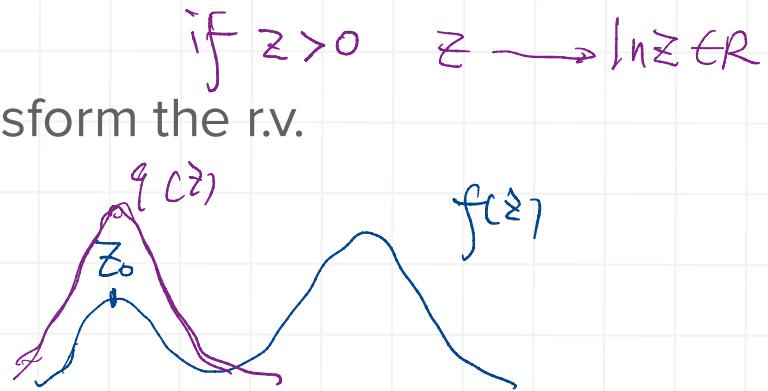
# Pros + cons of Laplace approximation

$$q(\mathbf{z}) = \frac{|\mathbf{A}|^{1/2}}{(2\pi)^{M/2}} \exp \left\{ -\frac{1}{2} (\mathbf{z} - \mathbf{z}_0)^T \mathbf{A} (\mathbf{z} - \mathbf{z}_0) \right\} = \mathcal{N}(\mathbf{z} | \mathbf{z}_0, \mathbf{A}^{-1}) \quad (4.134)$$

- :) Normalisation constant  $Z$  for  $f(z)$  does not need to be known.
- :) CLT  $\rightarrow$  posterior increasingly better approximated by Gaussian as the number of observed data points grow.

:) Assumes the domain of  $z$  is  $\mathbb{R}$ , or  $\mathbb{R}^d$ , may need to transform the r.v.

:) Is based purely on  $f(z)$  around  $z_0$ , no global info



# Approximate Inference + GP Classification

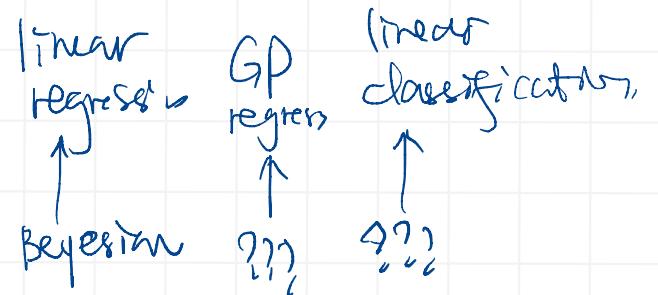
Laplace approximation - in general

Laplace approximation - Bayesian logistic regression

GP classification

Laplace approximation - GP classification

Connection to neural networks





## Recap: Logistic regression

$$p(\mathcal{C}_1|\boldsymbol{\phi}) = y(\boldsymbol{\phi}) = \sigma(\mathbf{w}^T \boldsymbol{\phi}) \quad (4.87)$$

$$\sigma(a) = \frac{1}{1 + \exp(-a)}.$$

data set  $\{\boldsymbol{\phi}_n, t_n\}$ , where  $t_n \in \{0, 1\}$  and  $\boldsymbol{\phi}_n = \boldsymbol{\phi}(\mathbf{x}_n)$ .

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^N y_n^{t_n} \{1 - y_n\}^{1-t_n} \quad (4.89)$$

Negative log-likelihood, or cross-entropy error function

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^N \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\} \quad (4.90)$$

$$\nabla E(\mathbf{w}) = \sum_{n=1}^N (y_n - t_n) \boldsymbol{\phi}_n \quad (4.91)$$

# Bayesian logistic regression

$$p(\mathcal{C}_1|\phi) = y(\phi) = \sigma(\mathbf{w}^T \phi) \quad (4.87)$$

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^N \{t_n \ln y_n + (1-t_n) \ln(1-y_n)\} \quad (4.90)$$

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0) \quad (4.140)$$

$$\underline{p(\mathbf{w}|\mathbf{t})} \propto p(\mathbf{w})p(\mathbf{t}|\mathbf{w}) \quad (4.141) \quad \xleftarrow{\text{P}(\vec{t}) \text{ const}}$$

$$\begin{aligned} \ln p(\mathbf{w}|\mathbf{t}) &= -\frac{1}{2} \underbrace{(\mathbf{w} - \mathbf{m}_0)^T \mathbf{S}_0^{-1} (\mathbf{w} - \mathbf{m}_0)}_{\text{quadratic part of } \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0)} \\ &+ \sum_{n=1}^N \{t_n \ln y_n + (1-t_n) \ln(1-y_n)\} + \text{const} \quad (4.142) \end{aligned}$$

*were from E(w)*

*normaliser of  $\mathcal{N}(\cdot, \cdot)$*

# Laplace approximation for Bayesian logistic regression

$$\begin{aligned} \ln p(\mathbf{w}|\mathbf{t}) &= -\frac{1}{2}(\mathbf{w} - \mathbf{m}_0)^T \mathbf{S}_0^{-1} (\mathbf{w} - \mathbf{m}_0) \\ &\quad + \sum_{n=1}^N \underbrace{\ln \sigma(w^T \phi_n)}_{\textcircled{1}} + \sum_{n=1}^N \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\} + \text{const} \quad (4.142) \end{aligned}$$

$$\sigma(w^T \phi) = \frac{1}{1 + e^{-w^T \phi}}$$

① compute local max  $\mathbf{z}_0$

$$q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{w}_{\text{MAP}}, \mathbf{S}_N). \quad (4.144)$$

② compute Hessian around  $\mathbf{z}_0$

$$\mathbf{S}_N = -\nabla \nabla \ln p(\mathbf{w}|\mathbf{t}) = \mathbf{S}_0^{-1} + \sum_{n=1}^N y_n (1 - y_n) \phi_n \phi_n^T. \quad (4.143)$$

$$y(\phi) = \sigma(w^T \phi)$$

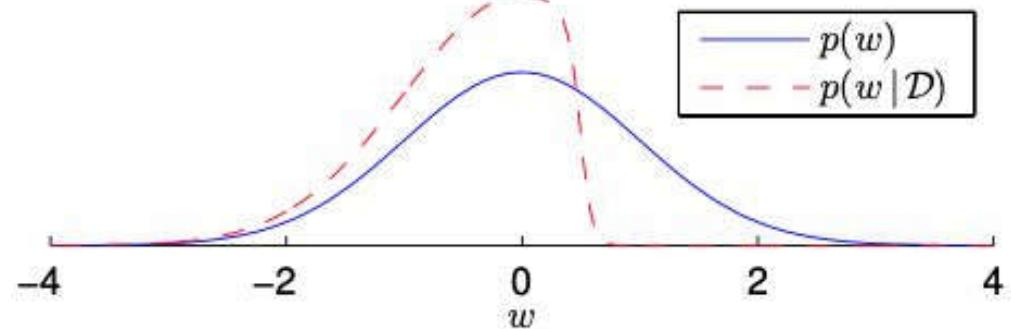
$$\frac{dy}{d\phi} = \sigma(1 - \sigma)$$

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

$$\begin{aligned} \nabla_w t_n \ln \sigma(w^T \phi_n) \\ = t_n \sigma(w^T \phi_n) (1 - \sigma(w^T \phi_n)) \frac{1}{\sigma(w^T \phi_n)} \phi_n \end{aligned}$$

## Example 1

$$p(w) \propto \mathcal{N}(w; 0, 1)$$
$$p(w | \mathcal{D}) \propto \mathcal{N}(w; 0, 1) \sigma(10 - 20w).$$

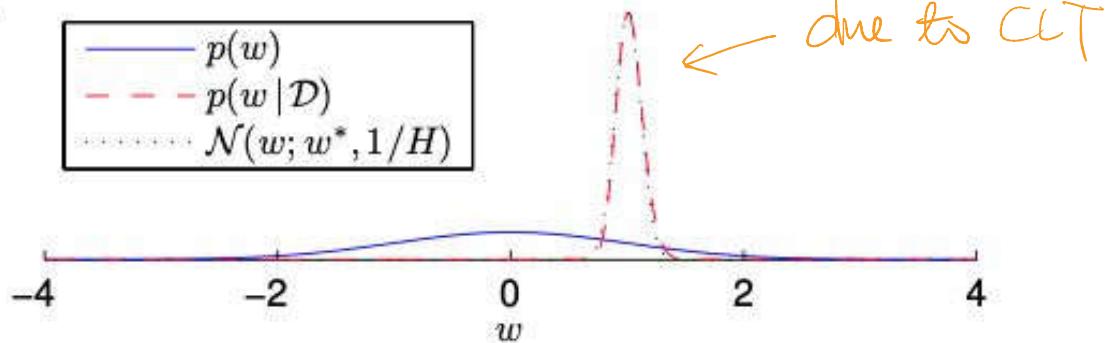


## Example 2

As another example, I generated  $N = 500$  labels,  $\{z^{(n)}\}$ , from a logistic regression model with no bias and with  $w=1$  at  $x^{(n)} \sim \mathcal{N}(0, 10^2)$ . Then,

$$p(w) \propto \mathcal{N}(w; 0, 1)$$

$$p(w | \mathcal{D}) \propto \mathcal{N}(w; 0, 1) \prod_{n=1}^{500} \sigma(wx^{(n)}z^{(n)}), \quad z^{(n)} \in \{\pm 1\}.$$



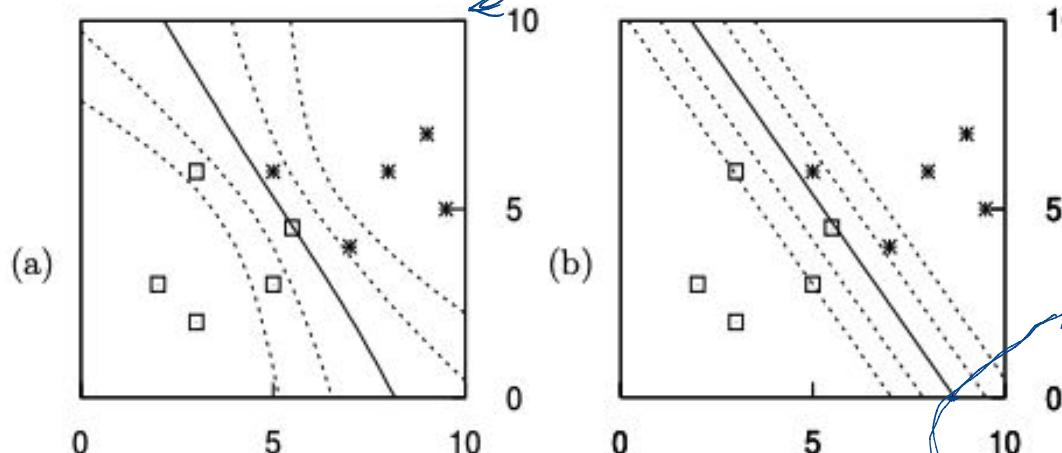
Example courtesy of Edinburgh MLPR course [https://www.inf.ed.ac.uk/teaching/courses/mlpr/2016/notes/w8a\\_bayes\\_logistic\\_regression\\_laplace.pdf](https://www.inf.ed.ac.uk/teaching/courses/mlpr/2016/notes/w8a_bayes_logistic_regression_laplace.pdf)

# What does the predictive distribution look like?

$$p(\mathcal{C}_1 | \phi, \mathbf{t}) = \int p(\mathcal{C}_1 | \phi, \mathbf{w}) p(\mathbf{w} | \mathbf{t}) d\mathbf{w} \simeq \int \sigma(\mathbf{w}^T \phi) q(\mathbf{w}) d\mathbf{w} \quad (4.145)$$

laplace approx.

$$= \int \sigma(a) \mathcal{N}(a | \mu_a, \sigma_a^2) da.$$



assumes  $p(\mathbf{w} | \mathbf{t}) = \mathcal{S}(\mathbf{w}^*)$

$\sigma(\mathbf{w}^* \phi)$   
over  $\phi = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

Again, the axes are the input features  $x_1$  and  $x_2$ . The right hand figure shows  $P(y=1 | \mathbf{x}, \mathbf{w}^*)$  for some fitted weights  $\mathbf{w}^*$ . No matter how these fitted weights are chosen, the contours have to be linear. The parallel contours mean that the uncertainty of predictions falls at the same rate when moving away from the decision boundary, no matter how far we are from the training inputs.

# Approximate Inference + GP Classification

Laplace approximation - in general

Laplace approximation - Bayesian logistic regression

GP classification

Laplace approximation - GP classification

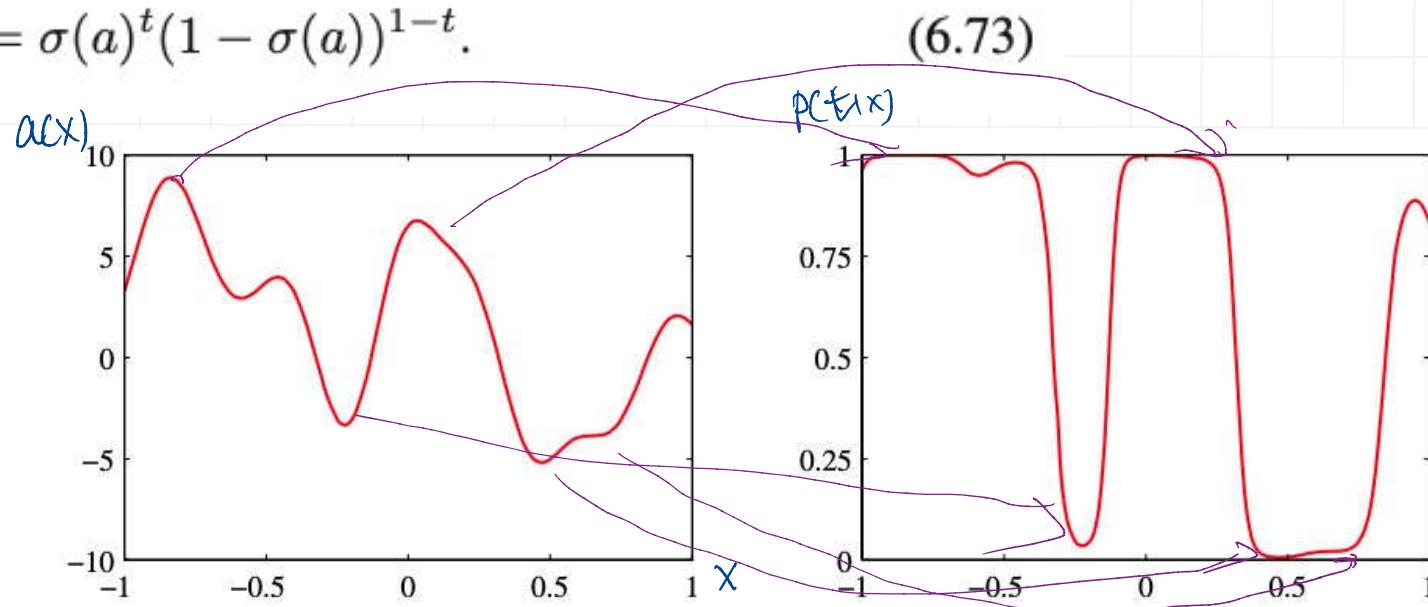
Connection to neural networks

## GP for classification

$$\underbrace{\mathbf{a}(\mathbf{x}) \sim GP(\mathbf{0}, K)}_{\text{mean fn } P \text{ covariance fn}}$$

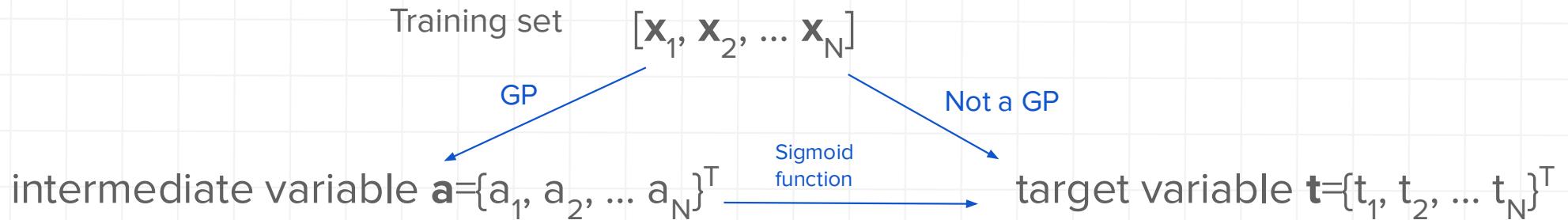
Training set: input  $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ , target variable  $\mathbf{t} = [t_1, t_2, \dots, t_N]^T$ , with  $t_i \in \{0, 1\}$

$$p(t|a) = \sigma(a)^t(1 - \sigma(a))^{1-t}.$$



**Figure 6.11** The left plot shows a sample from a Gaussian process prior over functions  $a(\mathbf{x})$ , and the right plot shows the result of transforming this sample using a logistic sigmoid function.

# What we just said



For regression

$$p(t_{N+1}) = \mathcal{N}(t_{N+1} | \mathbf{0}, \mathbf{C}_{N+1}) \quad (6.64)$$

Rename the variable

$$p(a_{N+1}) = \mathcal{N}(a_{N+1} | \mathbf{0}, \mathbf{C}_{N+1}). \quad (6.74)$$

## Covariance function of the GP a

$$p(\mathbf{a}_{N+1}) = \mathcal{N}(\mathbf{a}_{N+1} | \mathbf{0}, \mathbf{C}_{N+1}). \quad (6.74)$$

last week

$$C(x_n, x_m) = f(x_n, x_m)$$

$$+ \beta^+ \delta_{mn}.$$

↑  
noise -

- Assume it's noise-less
- BUT add diagonal term to ensure that it's positive semi-definite

$$C(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \nu \delta_{nm} \quad (6.75)$$

Assume kernel function  $k(x, x' ; \theta)$

# How to do prediction?

Assume:  $\mathbf{x}_{N+1}$  hidden.

Desired:

$$p(t_{N+1} = 1 | \mathbf{t}_N)$$

Have:

$$p(t_{N+1} = 1 | a_{N+1}) = \sigma(a_{N+1})$$

Plug in GP prediction

$$m(\mathbf{x}_{N+1}) = \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{t} \quad (6.66)$$

$$\sigma^2(\mathbf{x}_{N+1}) = c - \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{k}. \quad (6.67)$$

x is “hidden” inside kernels  $\mathbf{k}$  and  $\mathbf{C}$

$$p(a_{N+1} | \mathbf{a}_N) = \mathcal{N}(a_{N+1} | \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{a}_N, c - \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{k}). \quad (6.78)$$

Use Bayes rule:

$$p(t_{N+1} = 1 | \mathbf{t}_N) = \int p(t_{N+1} = 1 | a_{N+1}) p(a_{N+1} | \mathbf{t}_N) da_{N+1} \quad (6.76)$$

*assumes  $t_{N+1} \perp\!\!\!\perp t_N$  given  $a_{N+1}$*

# Computing posterior for GP classification

*predictive*

$\sigma(a_{N+1})$

$$p(t_{N+1} = 1 | \mathbf{t}_N) = \int p(t_{N+1} = 1 | a_{N+1}) p(a_{N+1} | \mathbf{t}_N) da_{N+1} \quad (6.76)$$

want to  
use

$\underline{p(a_{N+1} | \mathbf{a}_N)}$

$$\begin{aligned} p(a_{N+1} | \mathbf{t}_N) &= \int p(a_{N+1}, \mathbf{a}_N | \mathbf{t}_N) d\mathbf{a}_N \quad \xrightarrow{\text{Bayes rule}} \\ &\stackrel{\text{const}}{=} \frac{1}{p(\mathbf{t}_N)} \int p(a_{N+1}, \mathbf{a}_N) p(\mathbf{t}_N | a_{N+1}, \mathbf{a}_N) d\mathbf{a}_N \\ &= \frac{1}{p(\mathbf{t}_N)} \int p(a_{N+1} | \mathbf{a}_N) p(\mathbf{a}_N) p(\mathbf{t}_N | \mathbf{a}_N) d\mathbf{a}_N \quad \xrightarrow{p(\mathbf{t}_N | a_{N+1}, \mathbf{a}_N) = p(\mathbf{t}_N | \mathbf{a}_N)} \\ &= \int p(a_{N+1} | \mathbf{a}_N) p(\mathbf{a}_N | \mathbf{t}_N) d\mathbf{a}_N \end{aligned} \quad (6.77)$$

$$\xrightarrow{} p(a_{N+1} | \mathbf{a}_N) = \mathcal{N}(a_{N+1} | \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{a}_N, c - \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{k}). \quad (6.78)$$

Laplace approximation

$(x_n, t_n)$  known  
 $p(a_n)$  prior known.

Want to approximate  $p(\mathbf{a}_N | \mathbf{t}_N)$

$\rightarrow$  logistic reg.  
 $p(\mathbf{t}_N | \mathbf{a}_N)$

$P(\mathbf{a}_N) = \mathcal{N}(\mathbf{a}_N | \mathbf{0}, \mathbf{C}_N)$

$$p(\mathbf{t}_N | \mathbf{a}_N) = \prod_{n=1}^N \sigma(a_n)^{t_n} (1 - \sigma(a_n))^{1-t_n} = \prod_{n=1}^N e^{a_n t_n} \sigma(-a_n). \quad (6.79)$$

expand + simplify.

Taylor expansion of  $p(\mathbf{a}_N | \mathbf{t}_N)$

$$\begin{aligned} \Psi(\mathbf{a}_N) &= \ln p(\mathbf{a}_N) + \ln p(\mathbf{t}_N | \mathbf{a}_N) \\ &= -\frac{1}{2} \mathbf{a}_N^T \mathbf{C}_N^{-1} \mathbf{a}_N - \frac{N}{2} \ln(2\pi) - \frac{1}{2} \ln |\mathbf{C}_N| + \mathbf{t}_N^T \mathbf{a}_N \\ &\quad - \sum_{n=1}^N \ln(1 + e^{a_n}) + \text{const.} \end{aligned} \quad (6.80)$$

const  
 $-\ln p(\mathbf{t}_N)$

$\frac{1}{1+e^{-a_n}}$

## Laplace approximation of $p(\mathbf{a}_N | \mathbf{t}_N)$

$$\begin{aligned}\Psi(\mathbf{a}_N) &= \ln p(\mathbf{a}_N) + \ln p(\mathbf{t}_N | \mathbf{a}_N) \\ &= -\frac{1}{2} \mathbf{a}_N^T \mathbf{C}_N^{-1} \mathbf{a}_N - \frac{N}{2} \ln(2\pi) - \frac{1}{2} \ln |\mathbf{C}_N| + \mathbf{t}_N^T \mathbf{a}_N \\ &\quad - \sum_{n=1}^N \ln(1 + e^{a_n}) + \text{const.}\end{aligned}\tag{6.80}$$

$$\nabla \Psi(\mathbf{a}_N) = \mathbf{t}_N - \underline{\sigma}_N - \mathbf{C}_N^{-1} \underline{\mathbf{a}}_N \tag{6.81}$$

$$\nabla \nabla \Psi(\mathbf{a}_N) = -\mathbf{W}_N - \mathbf{C}_N^{-1} \tag{6.82}$$

Claim: -  $\nabla \nabla \Psi(\mathbf{a}_N)$ : positive definite, but posterior is non-Gaussian (since  $\mathbf{W}_N$  depend on  $\mathbf{a}_N$ )

$$q(\mathbf{a}_N) = \mathcal{N}(\mathbf{a}_N | \mathbf{a}_N^*, \mathbf{H}^{-1}). \tag{6.86}$$

$\downarrow$   
valid Laplace approximation,

$x_1, \dots, x_n \rightarrow a_1, \dots, a_m \rightarrow b, \dots, t_n$   
 $\uparrow$   
scalar

(a) find local max

(b)  $\mathbf{H} := -\nabla \nabla$

$$\nabla \Psi(\mathbf{a}_N) = 0 \Rightarrow \mathbf{a}_N^* = \mathcal{N}(\mathbf{t}_N, \mathbf{r}_N)$$

$$\sigma_N = [\sigma(a_n)] \quad \frac{\partial \sigma}{\partial a} = f(1-f)$$

$$\mathbf{W}_N = \underbrace{\mathbf{diag}[\sigma(a_n)(1-\sigma(a_n))]}_{(0, \frac{1}{4})}$$

$$\mathbf{a}_N^* = \mathbf{C}_N (\mathbf{t}_N - \underline{\sigma}_N). \tag{6.84}$$

$$\mathbf{H} = -\nabla \nabla \Psi(\mathbf{a}_N) = \mathbf{W}_N + \mathbf{C}_N^{-1} \tag{6.85}$$

Q: What do we use for  $f(a_n)$

## Bringing it back together

$$p(t_{N+1} = 1 | \mathbf{t}_N) = \int p(t_{N+1} = 1 | a_{N+1}) p(a_{N+1} | \mathbf{t}_N) da_{N+1} \quad (6.76)$$

$$\int p(a_{N+1} | \mathbf{a}_N) p(\mathbf{a}_N | \mathbf{t}_N) d\mathbf{a}_N \quad (6.77)$$

$$\sigma(a_{N+1})$$

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y} | \mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^T) \quad (2.115)$$

$$\mathcal{N}(a_{N+1} | \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{a}_N, c - \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{k})$$

$$\begin{aligned} \mathbb{E}[a_{N+1} | \mathbf{t}_N] &= \mathbf{k}^T (\mathbf{t}_N - \boldsymbol{\sigma}_N) \\ \text{var}[a_{N+1} | \mathbf{t}_N] &= c - \mathbf{k}^T (\mathbf{W}_N^{-1} + \mathbf{C}_N)^{-1} \mathbf{k}. \end{aligned}$$

$$(6.87) \quad (6.88)$$

$$q(\mathbf{a}_N) = \mathcal{N}(\mathbf{a}_N | \mathbf{a}_N^*, \mathbf{H}^{-1}).$$

$$\mathbf{a}_N^* = \mathbf{C}_N (\mathbf{t}_N - \boldsymbol{\sigma}_N).$$

$$\mathbf{H} = -\nabla \nabla \Psi(\mathbf{a}_N) = \mathbf{W}_N + \mathbf{C}_N^{-1}$$

## Bringing it back together

$$p(t_{N+1} = 1 | \mathbf{t}_N) = \int p(t_{N+1} = 1 | a_{N+1}) p(a_{N+1} | \mathbf{t}_N) da_{N+1} \quad (6.76)$$

$$\sigma(a_{N+1})$$

$$\mathbb{E}[a_{N+1} | \mathbf{t}_N] = \mathbf{k}^T(\mathbf{t}_N - \boldsymbol{\sigma}_N) \quad (6.87)$$

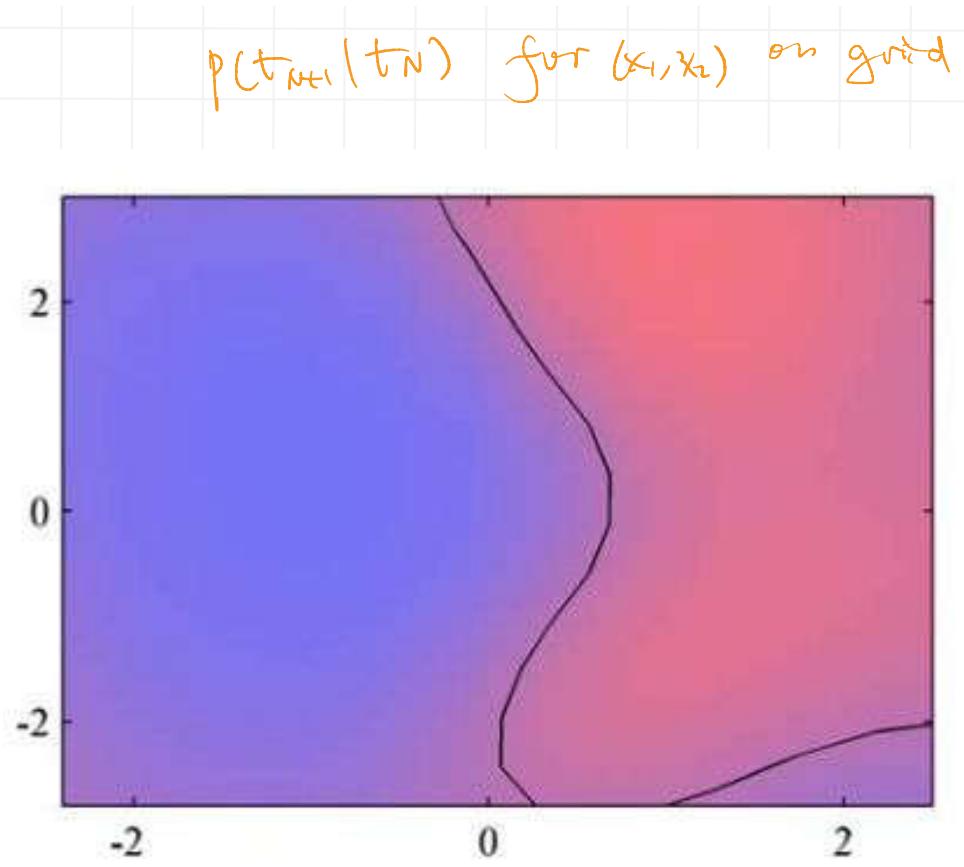
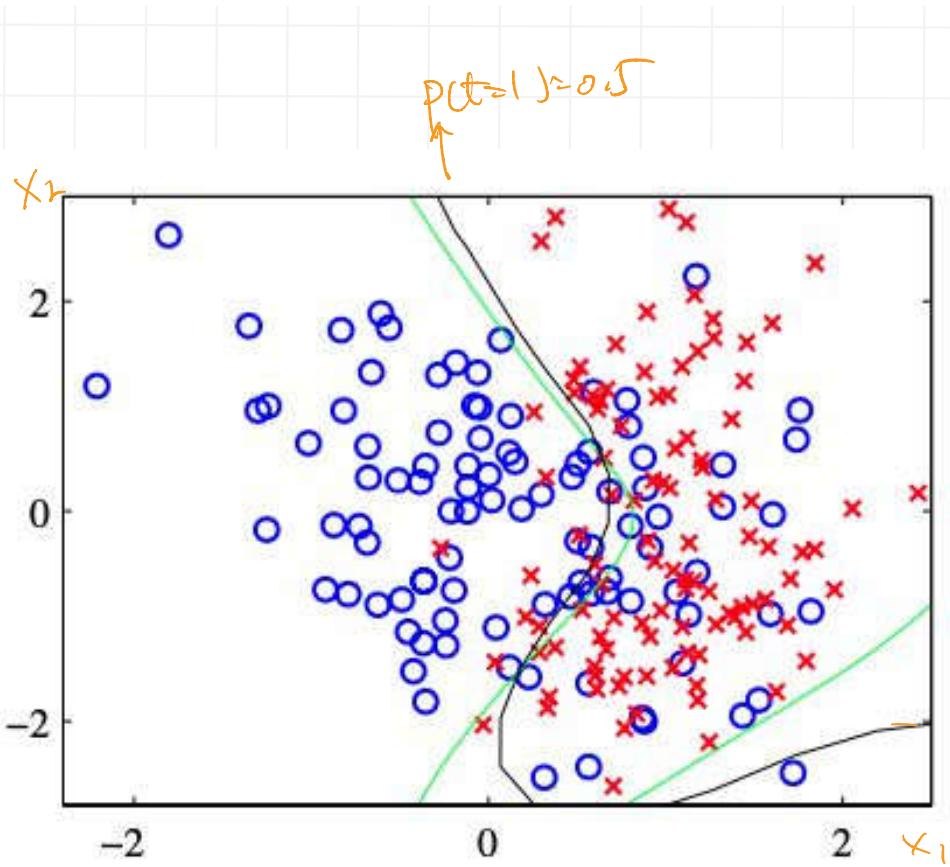
$$\text{var}[a_{N+1} | \mathbf{t}_N] = c - \mathbf{k}^T(\mathbf{W}_N^{-1} + \mathbf{C}_N)^{-1}\mathbf{k}. \quad (6.88)$$

$$\int \sigma(a) \mathcal{N}(a | \mu, \sigma^2) da \simeq \sigma(\kappa(\sigma^2)\mu) \quad (4.153)$$

$$\kappa(\sigma^2) = (1 + \pi\sigma^2/8)^{-1/2}.$$

$$c - \mathbf{k}^T(\mathbf{W}_N^{-1} + \mathbf{C}_N)^{-1}\mathbf{k}$$

$$\mathbf{k}^T(\mathbf{t}_N - \boldsymbol{\sigma}_N)$$



**Figure 6.12** Illustration of the use of a Gaussian process for classification, showing the data on the left together with the optimal decision boundary from the true distribution in green, and the decision boundary from the Gaussian process classifier in black. On the right is the predicted posterior probability for the blue and red classes together with the Gaussian process decision boundary.

## Connection to neural nets

2-layer neural network, with M hidden units

Bayesian neural network function  $f(x, w)$ , with prior over  $w$

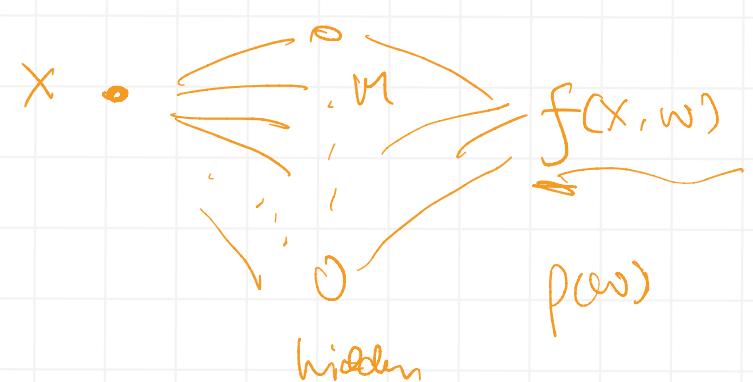
(Neal 1996) for a broad class of prior distributions over  $w$ , the distribution of functions generated by a neural network will tend to a Gaussian process when  $M$  tends to infinity.

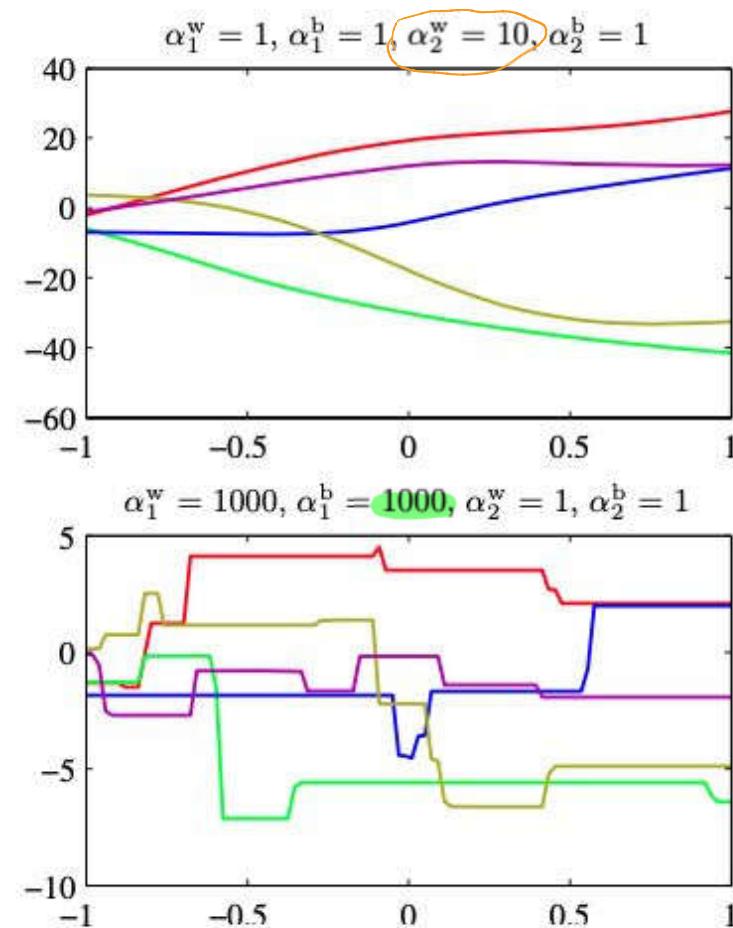
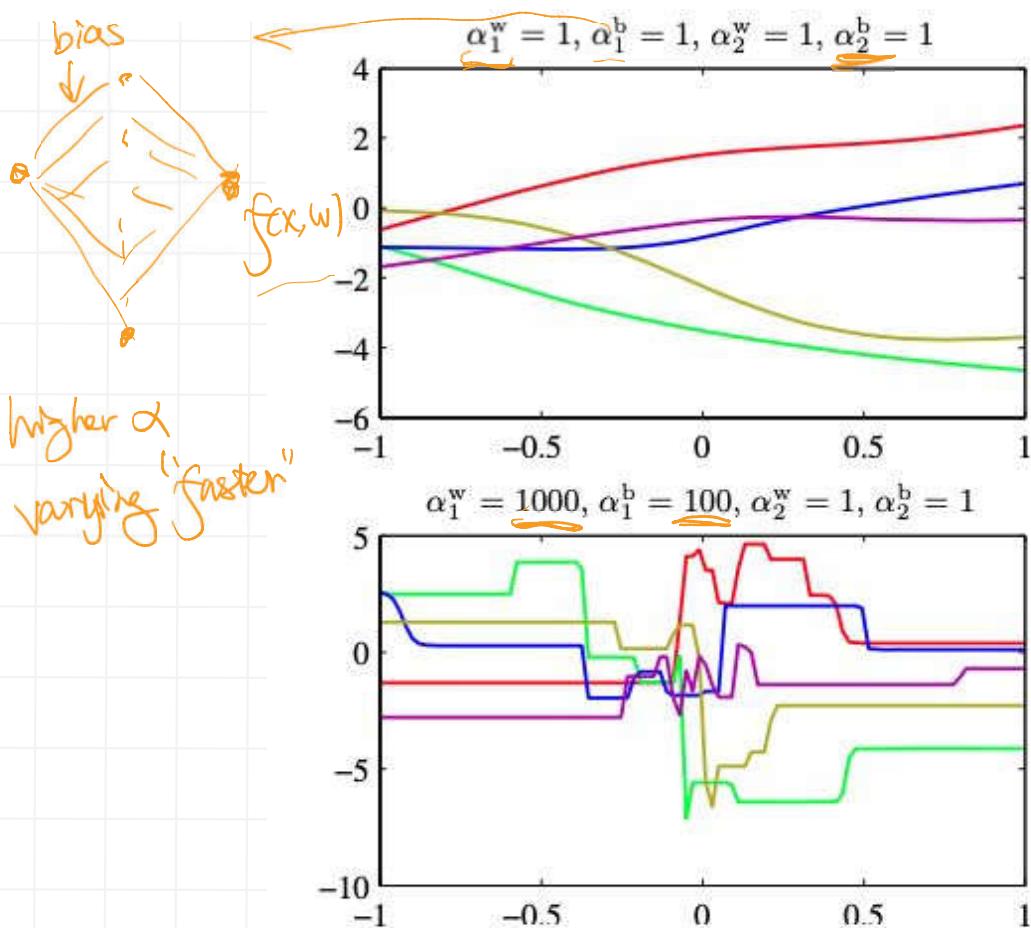
$$\text{e.g. } k(x' - x) = \frac{f(\|x - x'\|)}{\exp(-\frac{\sigma}{2}\|x - x'\|^2)}$$

non-stationary  $k(x, x') = x^T x'$

Covariance function  $k(x, x')$  non-stationary for neural nets with probit and Gaussian activations.

Weight prior determine the lengths scales of the neural net function.





**Figure 5.11** Illustration of the effect of the hyperparameters governing the prior distribution over weights and biases in a two-layer network having a single input, a single linear output, and 12 hidden units having ‘tanh’ activation functions. The priors are governed by four hyperparameters  $\alpha_1^b$ ,  $\alpha_1^w$ ,  $\alpha_2^b$ , and  $\alpha_2^w$ , which represent the precisions of the Gaussian distributions of the first-layer biases, first-layer weights, second-layer biases, and second-layer weights, respectively. We see that the parameter  $\alpha_2^w$  governs the vertical scale of functions (note the different vertical axis ranges on the top two diagrams),  $\alpha_1^w$  governs the horizontal scale of variations in the function values, and  $\alpha_1^b$  governs the horizontal range over which variations occur. The parameter  $\alpha_2^b$ , whose effect is not illustrated here, governs the range of vertical offsets of the functions.

# Gaussian Processes 2

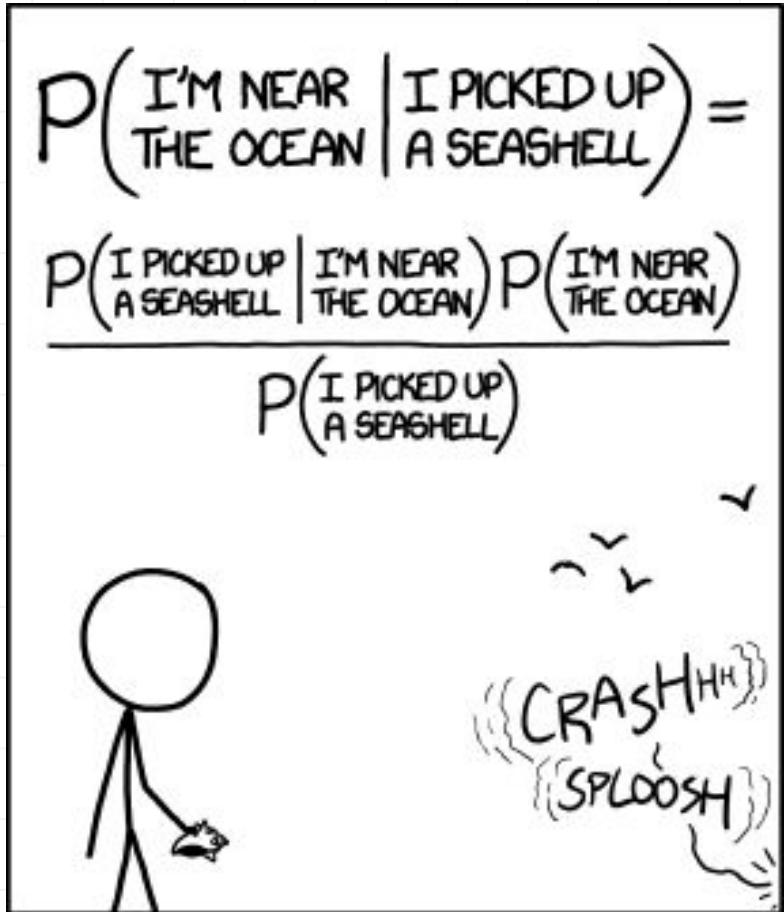
GP classification

Laplace approximation - in general

Laplace approximation - Bayesian logistic regression

Laplace approximation - GP classification

<https://xkcd.com/1236/>



STATISTICALLY SPEAKING, IF YOU PICK UP A  
SEASHELL AND DON'T HOLD IT TO YOUR EAR,  
YOU CAN PROBABLY HEAR THE OCEAN.

# Graphical models

[Bishop 8.1 and 8.2]

What? And Why?

What is graphical models / Bayesian network

Plate notation

Conditional independence

Quiz 2 remark

Everything comes from the sum rule and the product rule ... but graphical models provide a rich set of modeling language and inference procedures.

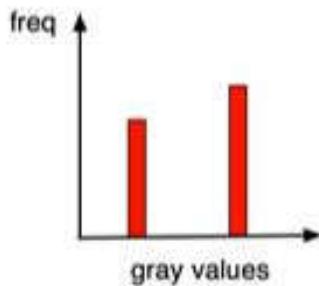
Fig 9.4



# Motivating scenario - computer vision

- **Image Segmentation**

Cluster the gray value representation of an image



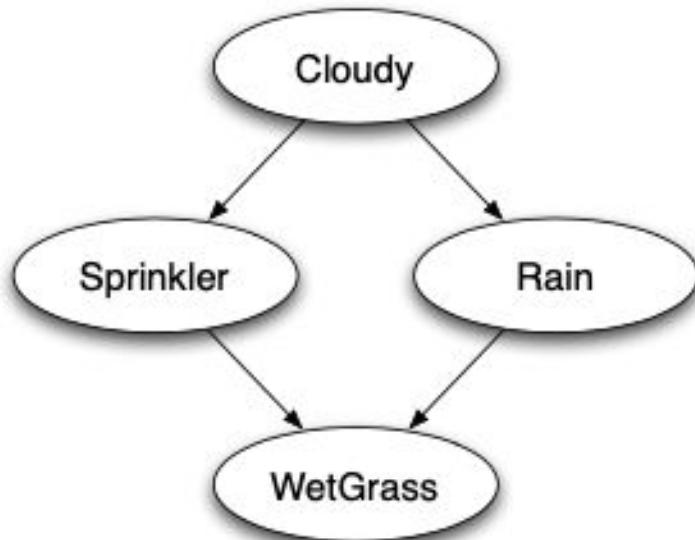
- **Neighbourhood information lost**

Need to use the structure of the image.



# Motivation - encoding (in)dependence

Why is the grass wet?



Introduce four Boolean variables :

$C(\text{loudy}), S(\text{prinkler}), R(\text{ain}), W(\text{etGrass}) \in \{F(\text{false}), T(\text{true})\}.$

# Motivation - encoding (in)dependence

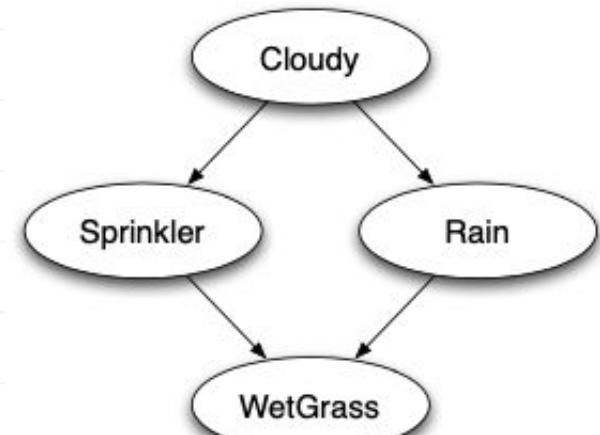
Model the conditional probabilities

	$p(C = F)$	$p(C = T)$
	0.2	0.8

C	$p(S = F)$	$p(S = T)$
F	0.5	0.5
T	0.9	0.1

C	$p(R = F)$	$p(R = T)$
F	0.8	0.2
T	0.2	0.8

S R	$p(W = F)$	$p(W = T)$
F F	1.0	0.0
T F	0.1	0.9
F T	0.1	0.9
T T	0.01	0.99

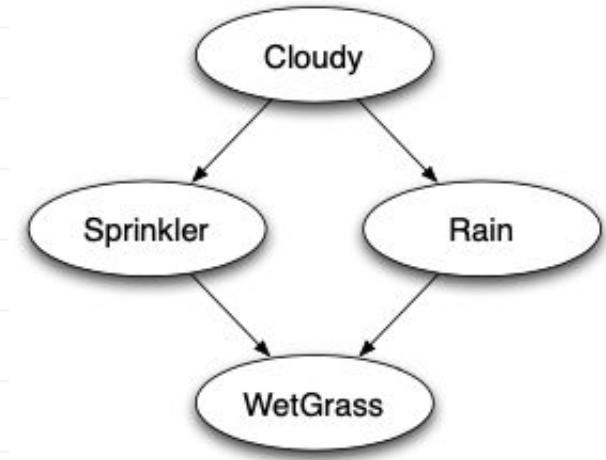
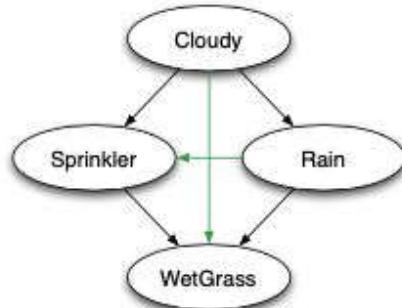


# Motivation - encoding (in)dependence

If everything depends on everything

C S R W	p(C, S, R, W)
F F F F	...
F F F T	...
...	...
T T T F	...
T T T T	...

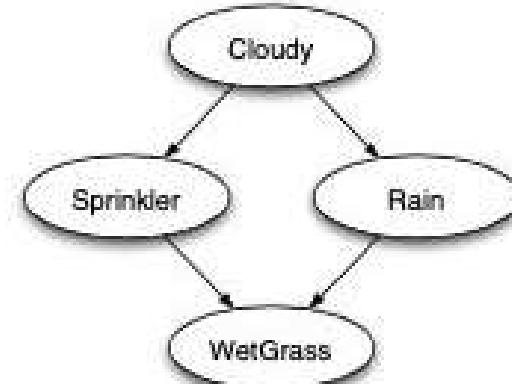
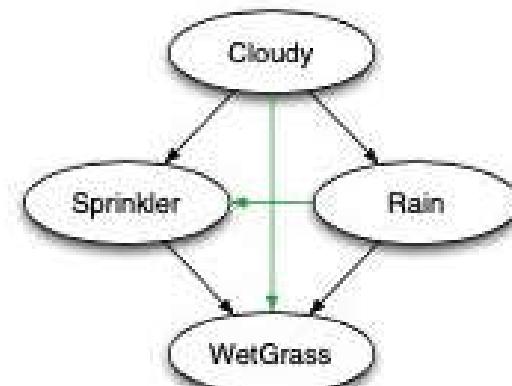
$$\begin{aligned} p(W, S, R, C) &= p(W | S, R, C) p(S, R, C) \\ &= p(W | S, R, C) p(S | R, C) p(R, C) \\ &= p(W | S, R, C) p(S | R, C) p(R | C) p(C) \end{aligned}$$



## Motivation - encoding (in)dependence

$$p(W) = \sum_{S,R,C} p(W | S, R, C) p(S | R, C) p(R | C) p(C)$$

$$p(W) = \sum_{S,R} p(W | S, R) \sum_C p(S | C) p(R | C) p(C)$$



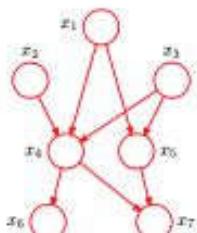
# Probabilistic graphical model - terminologies

Nodes: random variables

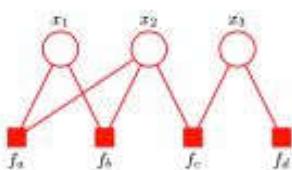
Edges: probabilistic relationships

Graph: captures “structures” in the joint distribution

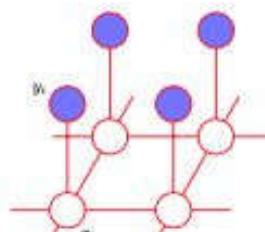
- Directed Graph : **Bayesian Network** (also called **Directed Graphical Model**) expressing **causal** relationship between variables
- Undirected Graph : **Markov Random Field** expressing **soft constraints** between variables
- Factor Graph : convenient for solving **inference** problems (derived from Bayesian Networks or Markov Random Fields).



Bayesian Network



Factor Graph



Markov Random Field

*Inference (Ch 8.4)* - compute the posterior distributions of one or more subsets of other nodes, given values of a subset of nodes (in a known graphical model with known parameters)

*Learning* - estimate the parameters (conditional probabilities) for a given graphical model, from a dataset of observed values.

# Probabilistic graphical model - overarching goals

Nodes: random variables

Edges: probabilistic relationships

Graph: captures “structures” in the joint distribution

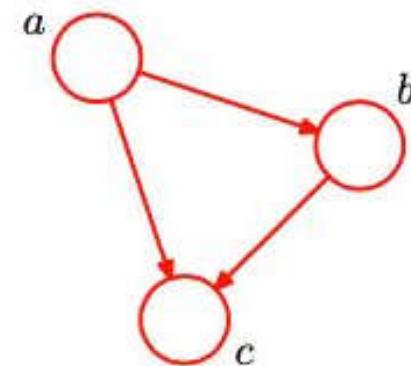
1. They provide a simple way to visualize the structure of a probabilistic model and can be used to design and motivate new models.
2. Insights into the properties of the model, including conditional independence properties, can be obtained by inspection of the graph.
3. Complex computations, required to perform inference and learning in sophisticated models, can be expressed in terms of graphical manipulations, in which underlying mathematical expressions are carried along implicitly.

# Bayesian networks

$$p(a, b, c) = p(c|a, b)p(a, b). \quad (8.1)$$

$$p(a, b, c) = p(c|a, b)p(b|a)p(a). \quad (8.2)$$

**Figure 8.1** A directed graphical model representing the joint probability distribution over three variables  $a$ ,  $b$ , and  $c$ , corresponding to the decomposition on the right-hand side of (8.2).



LHS of (8.2) is symmetric w.r.t  $a$ ,  $b$ ,  $c$ , but RHS is not.

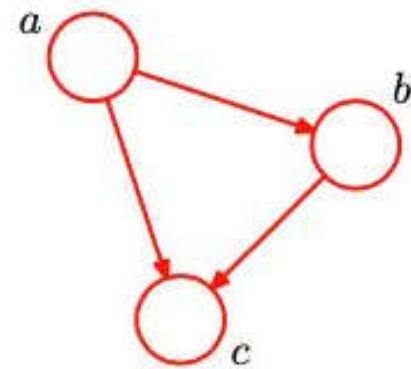
- Fig 8.2 implicitly chose an ordering
- possible to have different ordering, different decomposition, different graphical models

# Bayesian networks for any distribution

$$p(a, b, c) = p(c|a, b)p(a, b). \quad (8.1)$$

$$p(a, b, c) = p(c|a, b)p(b|a)p(a). \quad (8.2)$$

**Figure 8.1** A directed graphical model representing the joint probability distribution over three variables  $a$ ,  $b$ , and  $c$ , corresponding to the decomposition on the right-hand side of (8.2).



$$p(x_1, \dots, x_K) = p(x_K|x_1, \dots, x_{K-1}) \dots p(x_2|x_1)p(x_1). \quad (8.3)$$

This decomposition holds for any distribution over  $(x_1, \dots, x_K)$ .

Graph is “fully connected” -- there is a link (in either direction) between any pair of nodes.

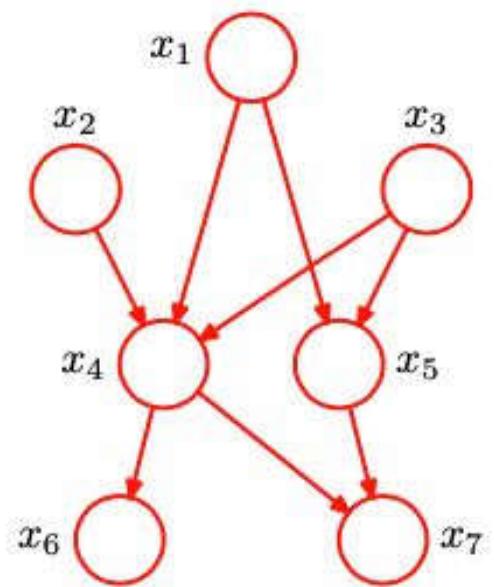
# Factoring joint distributions → Graph

$$p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5). \quad (8.4)$$

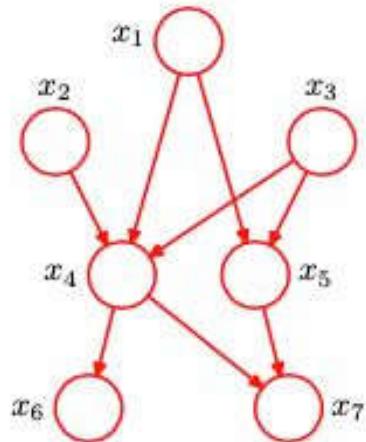
**Figure 8.2** Example of a directed acyclic graph describing the joint distribution over variables  $x_1, \dots, x_7$ . The corresponding decomposition of the joint distribution is given by (8.4).

... it is the *absence* of links in the graph that conveys interesting information about the properties of the class of distributions that the graph represents.

- ➊ Draw a node for each conditional distribution associated with a random variable.
- ➋ Draw an edge **from** each conditional distribution associated with a random variable **to** all other conditional distribution which are conditioned on this variable.



# Graph → joint distributions

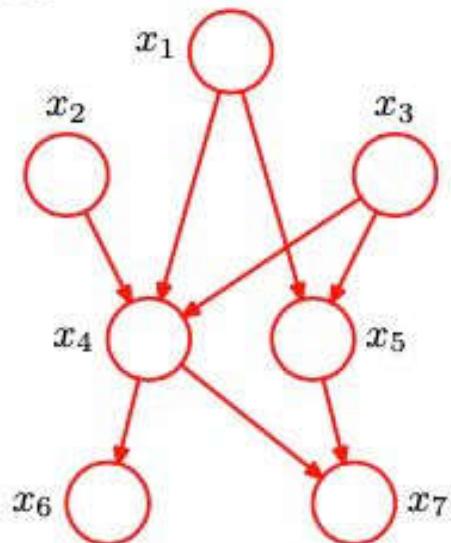


Can we get the expression from the graph?

- ➊ Write a product of probability distributions, one for each associated random variable. ↔ Draw a node for each conditional distribution associated with a random variable.
- ➋ Add all random variables associated with parent nodes to the list of conditioning variables. ↔ Draw an edge **from** each conditional distribution associated with a random variable **to** all other conditional distribution which are conditioned on this variable.

$$p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5)$$

- Restriction : Graph must be a **directed acyclic graph** (DAG).
- There are no closed paths in the graph when moving along the directed edges.
- Or equivalently: There exists an ordering of the nodes such that there are no edges that go from any node to any lower numbered node.



- Extension: Can also have **sets of variables**, or **vectors** at a node.

# General Bayes nets

In general, the joint distribution is given by:

$$p(\mathbf{x}) = \prod_{k=1}^K p(x_k | \text{pa}_k) \quad (8.5)$$

- What does it mean if  $\text{pa}(x_k) \neq x_1, x_2, \dots, x_{k-1}$  in the above equation?
  - Sparse graphical model.
  - $p(\mathbf{x})$  no longer general.
  - Assumption, e.g.  $p(W|C, S, R) = p(W|S, R)$ .
  - **Conditional independence.**

# General Bayes nets

In general, the joint distribution is given by:

$$p(\mathbf{x}) = \prod_{k=1}^K p(x_k | \text{pa}_k) \quad (8.5)$$

Is  $p(\mathbf{x})$  normalised,  $\sum_{\mathbf{x}} p(\mathbf{x}) = 1$ ?

- As graph is DAG, there always exists a node with no outgoing edges, say  $x_i$ .

$$\sum_{\mathbf{x}} p(\mathbf{x}) = \sum_{x_1, \dots, \underset{k \neq i}{x_{i-1}, x_{i+1}, \dots, x_K}} \prod_{k=1}^K p(x_k | \text{pa}(x_k)) \underbrace{\sum_{x_i} p(x_i | \text{pa}(x_i))}_{=1}$$

because  $\sum_{x_i} p(x_i | \text{pa}(x_i)) = \sum_{x_i} \frac{p(x_i, \text{pa}(x_i))}{p(\text{pa}(x_i))} = \frac{p(\text{pa}(x_i))}{p(\text{pa}(x_i))} = 1$

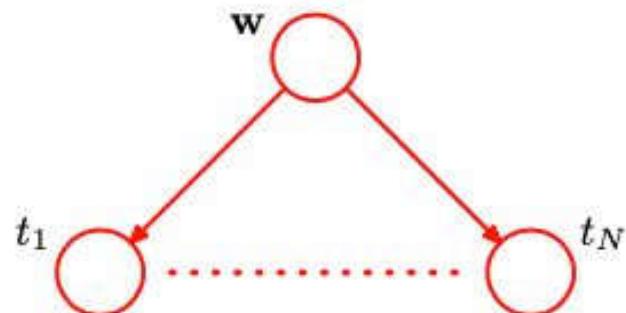
- Repeat, until no node left.

# Plate notation

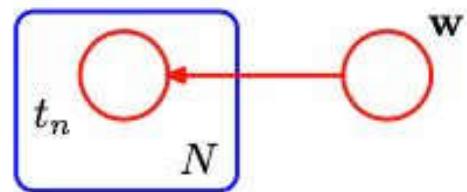
- Bayesian polynomial regression : observed inputs  $\mathbf{x}$ , observed targets  $\mathbf{t}$ , noise variance  $\sigma^2$ , hyperparameter  $\alpha$  controlling the priors for  $\mathbf{w}$ .
- Focusing on  $\mathbf{t}$  and  $\mathbf{w}$  only

$$p(\mathbf{t}, \mathbf{w}) = p(\mathbf{w}) \prod_{n=1}^N p(t_n | \mathbf{w})$$

**Figure 8.3** Directed graphical model representing the joint distribution (8.6) corresponding to the Bayesian polynomial regression model introduced in Section 1.2.6.



**Figure 8.4** An alternative, more compact, representation of the graph shown in Figure 8.3 in which we have introduced a *plate* (the box labelled  $N$ ) that represents  $N$  nodes of which only a single example  $t_n$  is shown explicitly.



# Polynomial regression: more variables

$$p(\mathbf{t}, \mathbf{w} | \mathbf{x}, \alpha, \sigma^2) = p(\mathbf{w} | \alpha) \prod_{n=1}^N p(t_n | \mathbf{w}, x_n, \sigma^2)$$

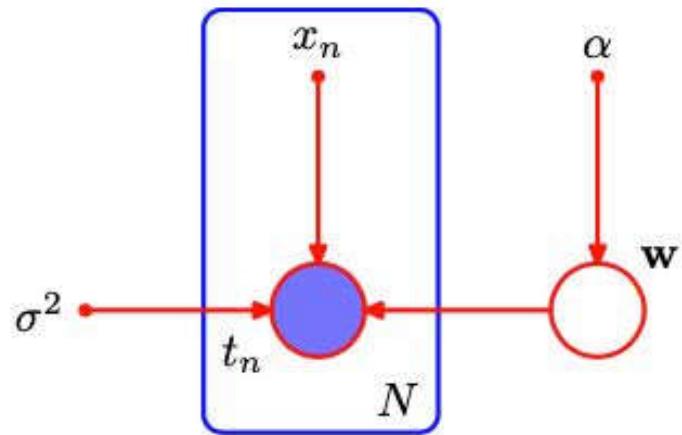
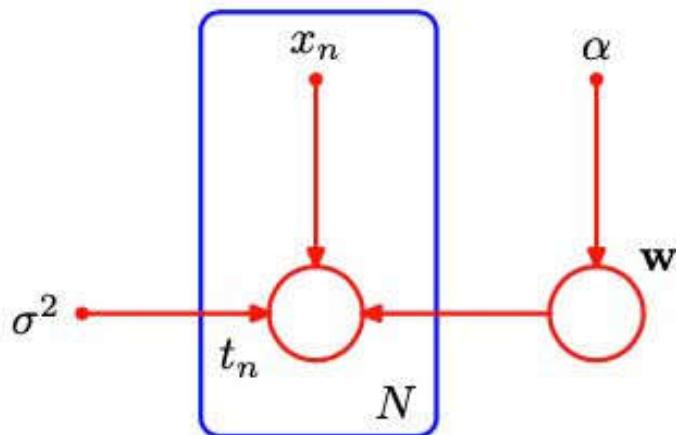
Random variables = open circles

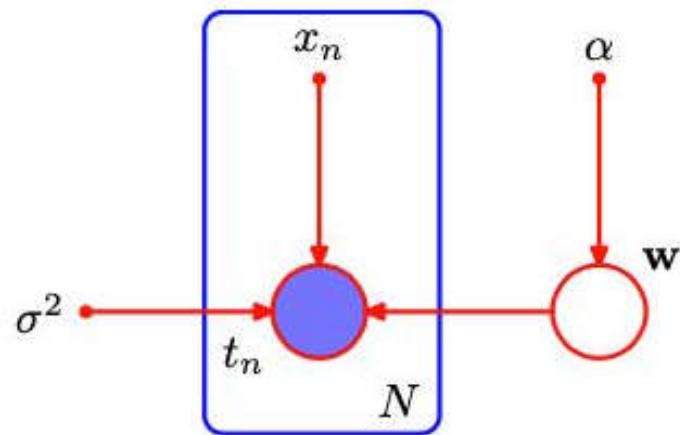
Deterministic variables = smaller solid circles

## Random variables

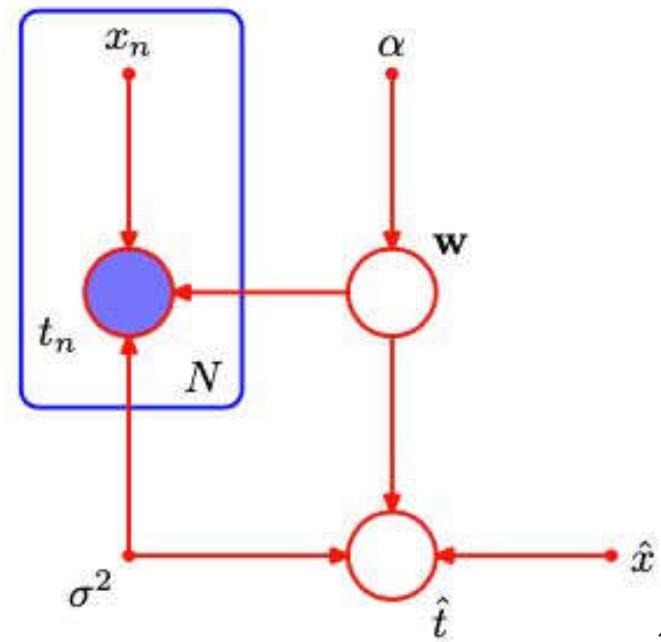
- Observed random variables, e.g.  $\mathbf{t}$
- Unobserved random variables, e.g.  $\mathbf{w}$ ,  
(latent random variables, hidden random variables)

Shade the observed random variables in the graphical model.





**Figure 8.7** The polynomial regression model, corresponding to Figure 8.6, showing also a new input value  $\hat{x}$  together with the corresponding model prediction  $\hat{t}$ .



# Generative models

Bayes net allows us to draw samples from a (joint) distribution.

Ancestral sampling: 
$$p(\mathbf{x}) = \prod_{k=1}^K p(x_k | \text{pa}_k) \quad (8.5)$$

- Work though each node *in order*
- Draw lower-number nodes (parents)
- Draw  $x_k$  with values of its parents fixed
- To obtain samples from marginals, retain the variables in question and disregard the rest

# Generative models

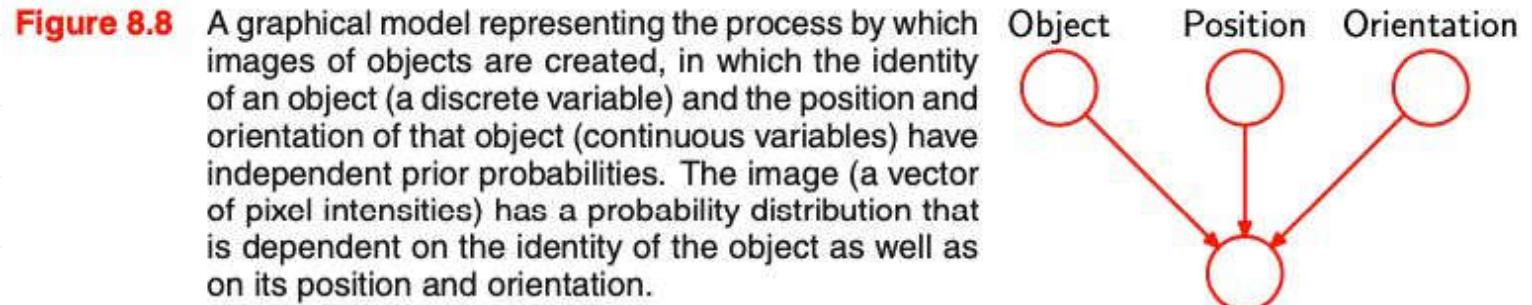
In a typical application

- Higher-numbered nodes: terminal nodes, observed
- Lower-numbered nodes: latent variables, unobserved
  - to allow a complex distribution over the observed to factor into simpler conditional distributions

BN graphs represent causal processes.

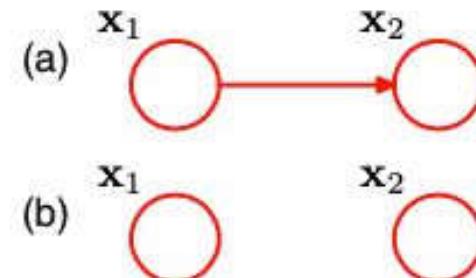
Polynomial regression example isn't a generative model (but can be made so with more complexity)

Latent variables need not have physical interpretation (can be introduced for representational/computational reasons)

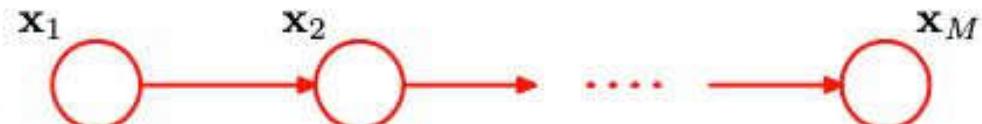


# Number of parameters for discrete variables

**Figure 8.9** (a) This fully-connected graph describes a general distribution over two  $K$ -state discrete variables having a total of  $K^2 - 1$  parameters. (b) By dropping the link between the nodes, the number of parameters is reduced to  $2(K - 1)$ .



**Figure 8.10** This chain of  $M$  discrete nodes, each having  $K$  states, requires the specification of  $K - 1 + (M - 1)K(K - 1)$  parameters, which grows linearly with the length  $M$  of the chain. In contrast, a fully connected graph of  $M$  nodes would have  $K^M - 1$  parameters, which grows exponentially with  $M$ .



Read: 8.1.3 in Bishop

# Graphical models

What? And Why?

What is graphical models / Bayesian network

Plate notation

Conditional independence

### *Definition (Conditional Independence)*

If for three random variables  $a$ ,  $b$ , and  $c$  the following holds

$$p(a | b, c) = p(a | c)$$

then  $a$  is **conditionally independent** of  $b$  given  $c$ .

Notation :  $a \perp\!\!\!\perp b | c$ .

- The above equation must hold for all possible values of  $c$ .
- Consequence :

$$\begin{aligned} p(a, b | c) &= p(a | b, c) p(b | c) \\ &= p(a | c) p(b | c) \end{aligned}$$

- Conditional independence simplifies
  - the structure of the model
  - the computations needed to perform inference/learning.
- NB: **conditional dependence** is a related but different concept we are not concerned with in this course.

## Rules for Conditional Independence

Symmetry :

$$X \perp\!\!\! \perp Y | Z \implies Y \perp\!\!\! \perp X | Z$$

Decomposition :

$$Y, W \perp\!\!\! \perp X | Z \implies Y \perp\!\!\! \perp X | Z \text{ and } W \perp\!\!\! \perp X | Z$$

Weak Union :

$$X \perp\!\!\! \perp Y, W | Z \implies X \perp\!\!\! \perp Y | Z, W$$

Contraction :

$$X \perp\!\!\! \perp W | Z, Y$$

$$\text{and } X \perp\!\!\! \perp Y | Z \implies X \perp\!\!\! \perp W, Y | Z$$

Intersection :

$$X \perp\!\!\! \perp Y | Z, W$$

$$\text{and } X \perp\!\!\! \perp W | Z, Y \implies X \perp\!\!\! \perp Y, W | Z$$

Note: Intersection is only valid for  $p(X), p(Y), p(Z), p(W) > 0$ .

# Three-node graphs

Goal: infer conditional independence of  $a$  and  $b$

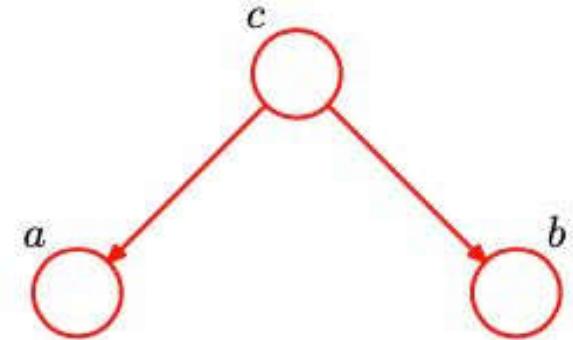
Larger goal: spot general subgraph patterns so as to reason about conditional independence in general.

$$p(a, b, c) = p(a | c) p(b | c) p(c)$$

Marginalise both sides over  $c$

$$p(a, b) = \sum_c p(a | c) p(b | c) p(c) \neq p(a) p(b).$$

Does not hold :  $a \perp\!\!\!\perp b | \emptyset$  (where  $\emptyset$  is the empty set).

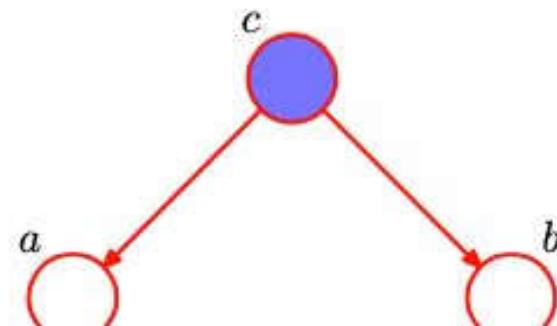


Now condition on  $c$ .

$$p(a, b | c) = \frac{p(a, b, c)}{p(c)} = p(a | c)p(b | c)$$

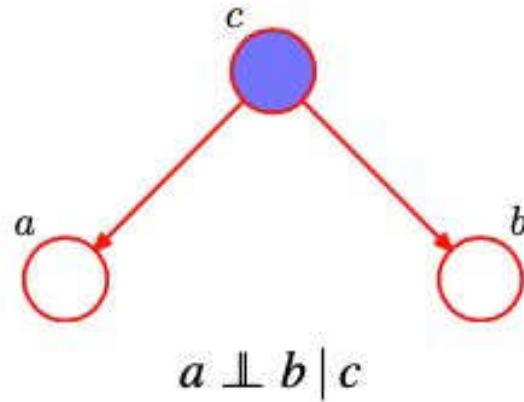
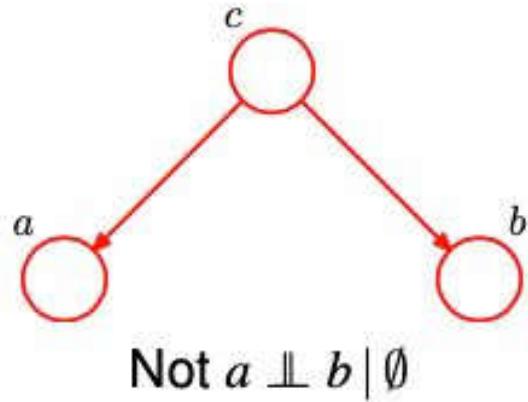
Therefore  $a \perp b | c$ .

**Figure 8.16** As in Figure 8.15 but where we have conditioned on the value of variable  $c$ .



## Graphical interpretation

- In both graphical models there is a **path** from  $a$  to  $b$ .
- The node  $c$  is called **tail-to-tail** (TT) with respect to this path because the node  $c$  is connected to the tails of the arrows in the path.
- The presence of the TT-node  $c$  in the path left renders  $a$  dependent on  $b$  (and  $b$  dependent on  $a$ ).
- Conditioning on  $c$  **blocks** the path from  $a$  to  $b$  and causes  $a$  and  $b$  to become conditionally independent on  $c$ .



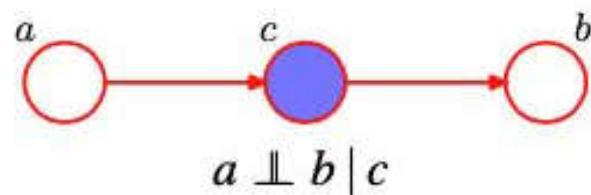
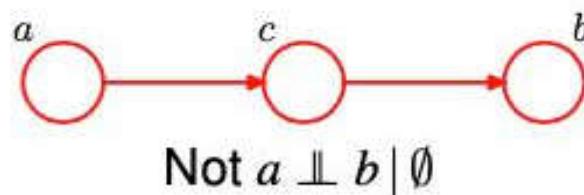
$$p(a, b, c) = p(a) p(c | a) p(b | c)$$

Head-Tail (HT) pattern

Marginalise over  $c$  to test for independence.

$$p(a, b) = p(a) \sum_c p(c | a) p(b | c) = p(a) p(b | a) \neq p(a) p(b)$$

Does not hold :  $a \perp\!\!\!\perp b | \emptyset$ .



Now condition on  $c$ .

$$p(a, b | c) = \frac{p(a, b, c)}{p(c)} = \frac{p(a) p(c | a) p(b | c)}{p(c)} = p(a | c) p(b | c)$$

where we used Bayes' theorem  $p(c | a) = p(a | c) p(c) / p(a)$ .

Therefore  $a \perp\!\!\!\perp b | c$ .

Head-Head (HH) pattern -- a bit more subtle

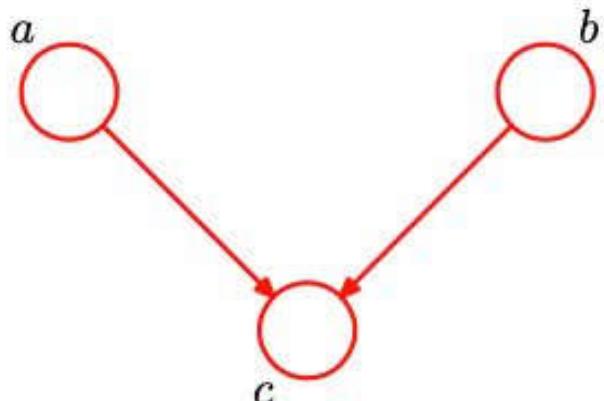
$$p(a, b, c) = p(a)p(b)p(c | a, b)$$

Marginalise over  $c$  to test for independence.

$$\begin{aligned} p(a, b) &= \sum_c p(a)p(b)p(c | a, b) = p(a)p(b)\sum_c p(c | a, b) \\ &= p(a)p(b) \end{aligned}$$

a and b are independent if NO variable is observed:

$$a \perp\!\!\!\perp b | \emptyset.$$

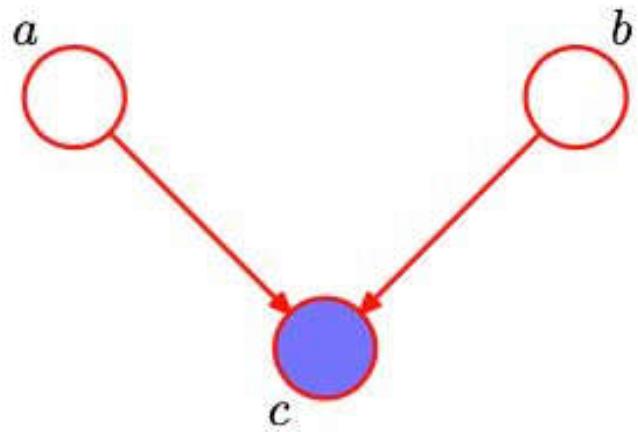


## Head-Head (HH) pattern -- 2 of 4

- Now condition on  $c$ .

$$p(a, b | c) = \frac{p(a, b, c)}{p(c)} = \frac{p(a)p(b)p(c | a, b)}{p(c)} \neq p(a | c)p(b | c).$$

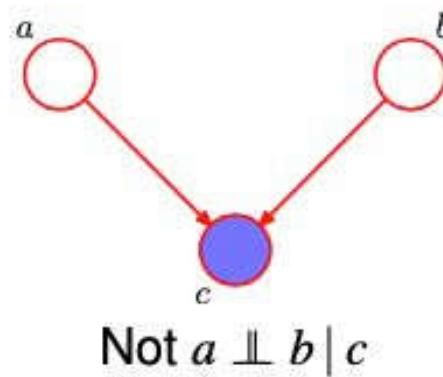
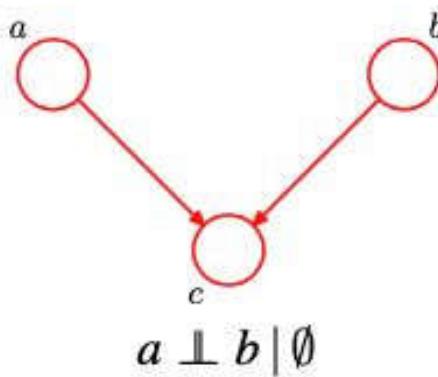
- Does not hold :  $a \perp\!\!\!\perp b | c$ .



## Head-Head (HH) pattern -- 3 of 4

### Graphical interpretation

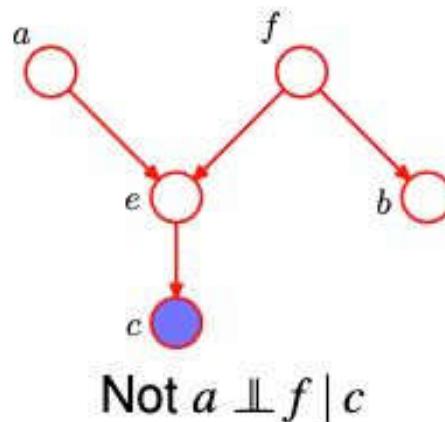
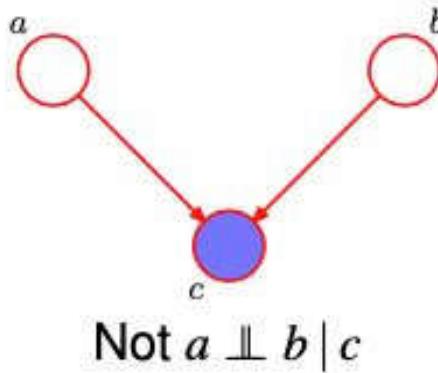
- In both graphical models there is a **path** from  $a$  to  $b$ .
- The node  $c$  is called **head-to-head** (HH) with respect to this path because the node  $c$  is connected to the heads of the arrows in the path.
- The presence of the HH-node  $c$  in the path left makes  $a$  independent of  $b$  (and  $b$  independent of  $a$ ). The unobserved  $c$  **blocks** the path from  $a$  to  $b$ .



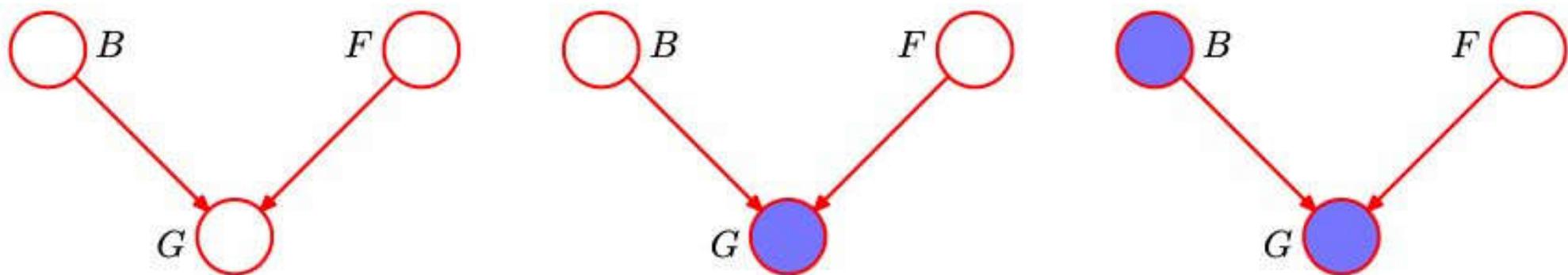
## Head-Head (HH) pattern -- 4 of 4

### Graphical interpretation

- Conditioning on  $c$  **unblocks** the path from  $a$  to  $b$ , and renders  $a$  conditionally dependent on  $b$  given  $c$ .
- Some more terminology: Node  $y$  is a **descendant** of node  $x$  if there is a path from  $x$  to  $y$  in which each step follows the directions of the arrows.
- A HH-path will become unblocked if either the node, or any of its descendants, is observed.



## HH pattern and “explaining away”



**Figure 8.21** An example of a 3-node graph used to illustrate the phenomenon of ‘explaining away’. The three nodes represent the state of the battery ( $B$ ), the state of the fuel tank ( $F$ ) and the reading on the electric fuel gauge ( $G$ ). See the text for details.

# Conditional independence for general graphs

- Conditional independence has been established for
  - all configurations of three variables: (HH, HT, TT)  $\times$  (observed, unobserved)
  - the subtle case of HH junction with observed descendent.
- We can generalise these results to arbitrary Bayesian Networks.
- Roughly: the graph connectivity implies conditional independence for those sets of nodes which are **directionally separated**.

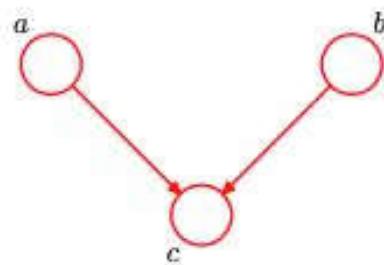
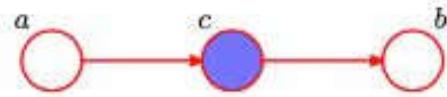
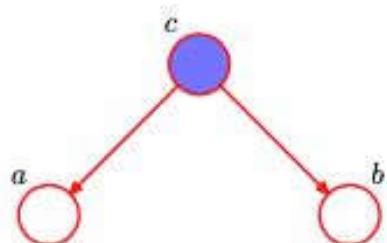
→ known as “D-separation”

## D-separation: blocked paths

### *Definition (Blocked Path)*

A blocked path is a path which contains

- an observed TT- or HT-node, or
- an unobserved HH-node whose descendants are all unobserved.

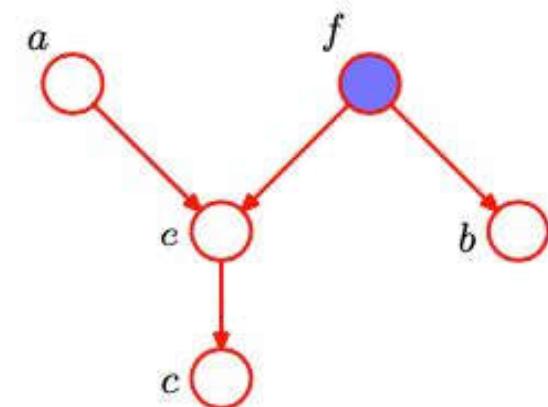
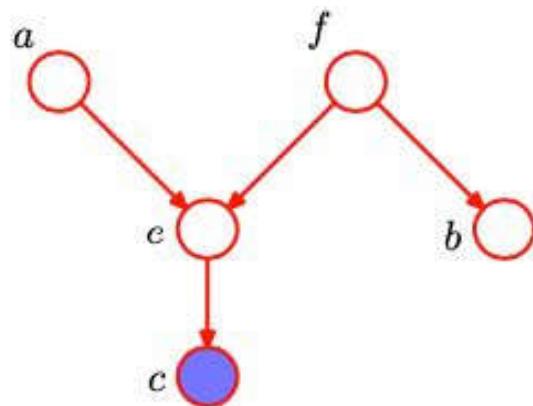


## D-separation

- Consider a general directed graph in which  $A$ ,  $B$ , and  $C$  are arbitrary non-intersecting sets of nodes. (There may be other nodes in the graph which are not contained in the union of  $A$ ,  $B$ , and  $C$ .)
- Consider all possible paths from any node in  $A$  to any node in  $B$ .
- Any such path is blocked, if it includes a node such that either
  - the node is HT or TT, and the node is in set  $C$ , or
  - the node is HH, and neither the node, nor any of the descendants, is in set  $C$ .
- If all paths are blocked, then  $A$  is  $d-$  separated from  $B$  by  $C$ , and the joint distribution over all the variables in the graph will satisfy  $A \perp\!\!\!\perp B | C$ .

## D-separation: example

- The path from  $a$  to  $b$  is not blocked by  $f$  because  $f$  is a TT-node and unobserved.
- The path from  $a$  to  $b$  is not blocked by  $e$  because  $e$  is a HH-node, and although unobserved itself, one of its descendants (node  $c$ ) is observed.



- The path from  $a$  to  $b$  is blocked by  $f$  because  $f$  is a TT-node and observed. Therefore,  $a \perp b | f$ .
- Furthermore, the path from  $a$  to  $b$  is also blocked by  $e$  because  $e$  is a HH-node, and neither it nor its descendants are observed. Therefore  $a \perp b | f$ .

### Theorem (Factorisation $\Rightarrow$ Conditional Independence)

If a probability distribution factorises according to a directed acyclic graph, and if  $A$ ,  $B$  and  $C$  are disjoint subsets of nodes such that  $A$  is d-separated from  $B$  by  $C$  in the graph, then the distribution satisfies  $A \perp\!\!\!\perp B | C$ .

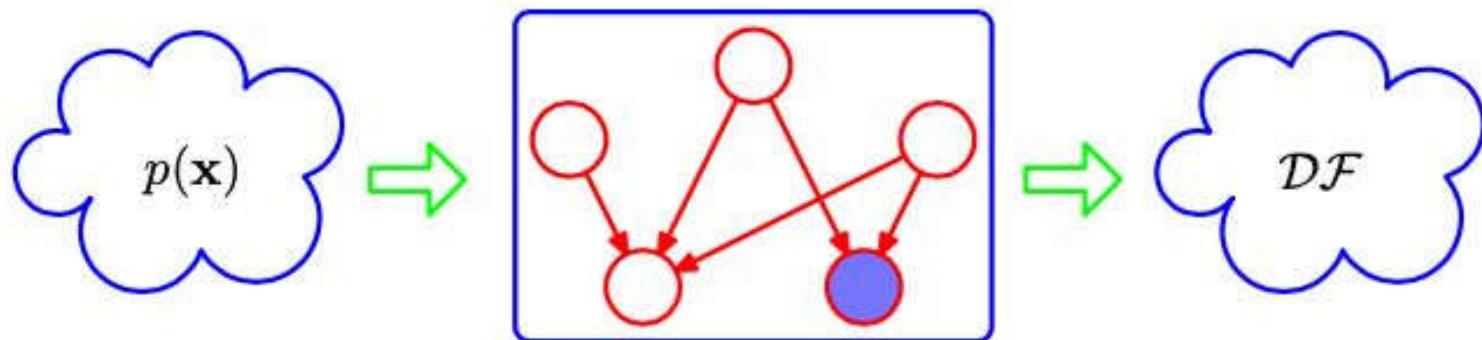
### Theorem (Conditional Independence $\Rightarrow$ Factorisation)

If a probability distribution satisfies the conditional independence statements implied by d-separation over a particular directed graph, then it also factorises according to the graph.

Why is Conditional Independence  $\Leftrightarrow$  Factorisation relevant?

- Conditional Independence statements are usually what a domain expert knows about the problem at hand.
- Needed is a model  $p(\mathbf{x})$  for computation.
- The Conditional Independence  $\Rightarrow$  Factorisation provides  $p(x)$  from Conditional Independence statements.
- One can build a global model for computation from local conditional independence statements.

# Graphical model as filters for probability distributions



**Figure 8.25** We can view a graphical model (in this case a directed graph) as a filter in which a probability distribution  $p(\mathbf{x})$  is allowed through the filter if, and only if, it satisfies the directed factorization property (8.5). The set of all possible probability distributions  $p(\mathbf{x})$  that pass through the filter is denoted  $\mathcal{DF}$ . We can alternatively use the graph to filter distributions according to whether they respect all of the conditional independencies implied by the d-separation properties of the graph. The d-separation theorem says that it is the same set of distributions  $\mathcal{DF}$  that will be allowed through this second kind of filter.

# Graphical models: Bayes Nets

What? And Why?

What is graphical models / Bayesian network

Plate notation

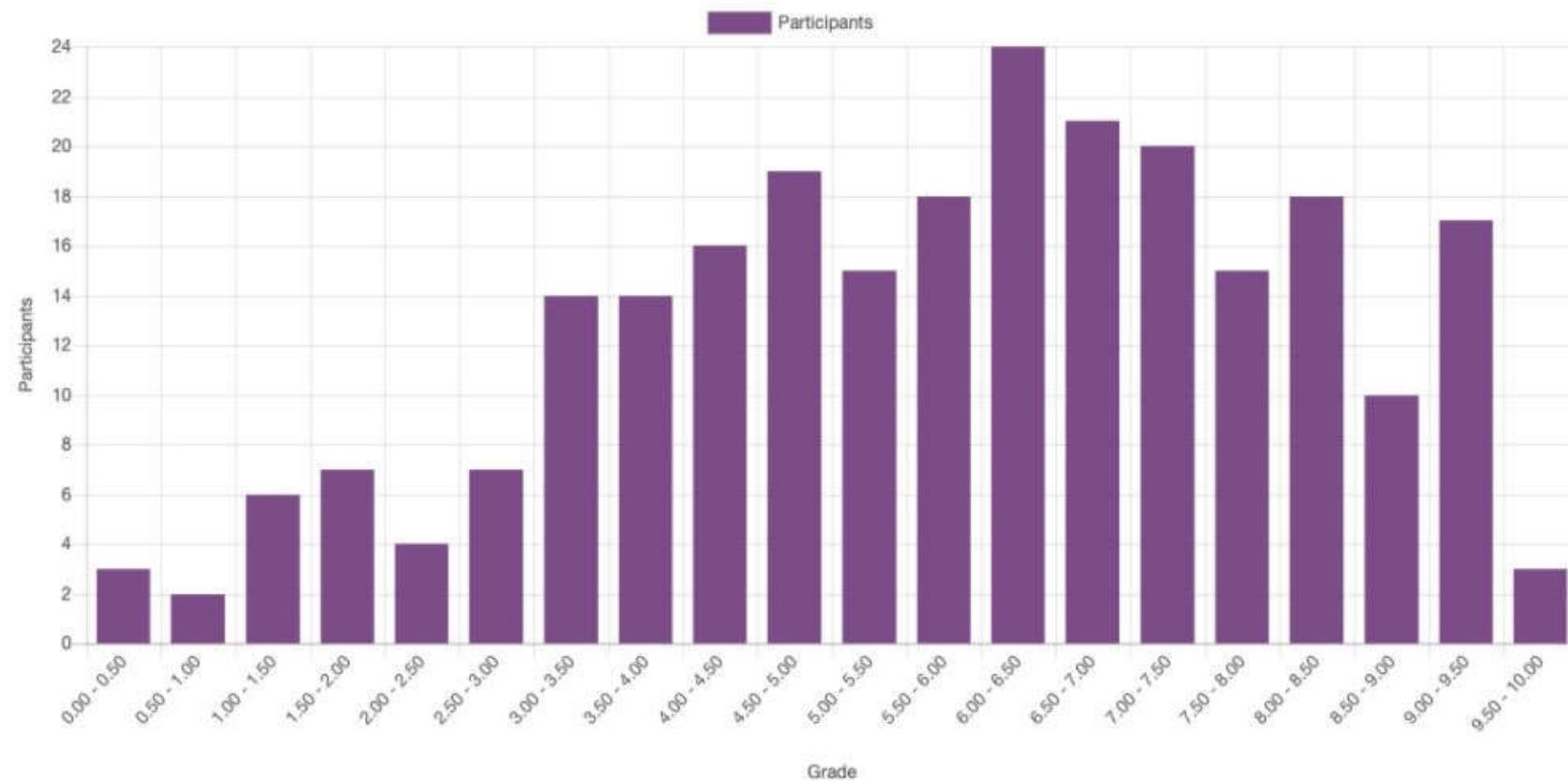
Conditional independence

## quiz 2 stats

253 students completed quiz 2, here are some stats:  
average\* 57.3; median 60 (out of 100)

the easiest question: Q5 linear regression of gaussian variables 81%; Q8, hoeffding's 71%  
the hardest question: Q4 KDE, 23%

## Overall number of students achieving grade ranges

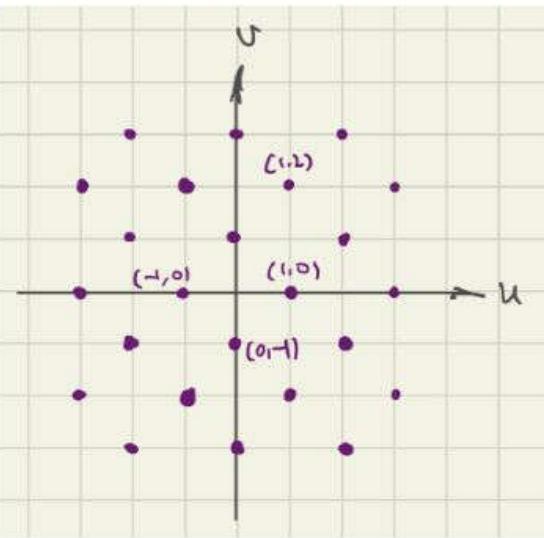


Consider points on a regular grid in the  $(u, v)$  plane. The four data points closest to the origin are  $(1, 0), (-1, 0), (0, 1), (0, -1)$ . For every data point  $\mathbf{x}_i = (u_i, v_i)$ , there are four other data points at  $(u_i \pm 2, v_i)$  and  $(u_i, v_i \pm 2)$ .

Define the  $L_1$ , or manhattan distance between two points as  $\|\mathbf{x}_i - \mathbf{x}_j\|_1 = |u_i - u_j| + |v_i - v_j|$ . We perform kernel density estimation using two different kernels  $k_1(\mathbf{x}, \mathbf{x}_0)$  and  $k_2(\mathbf{x}, \mathbf{x}_0)$ , and obtain estimates  $p_1(u, v)$  and  $p_2(u, v)$ , respectively. Those two kernels are defined as following:

$$\text{Cube kernel: } k_1(\mathbf{x}, \mathbf{x}_0) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{if } \|\mathbf{x} - \mathbf{x}_0\|_1 < 1. \\ 0, & \text{otherwise.} \end{cases}$$

$$\text{Pyramid kernel: } k_2(\mathbf{x}, \mathbf{x}_0) = \max\{0, 1 - \|\mathbf{x} - \mathbf{x}_0\|_1\}$$

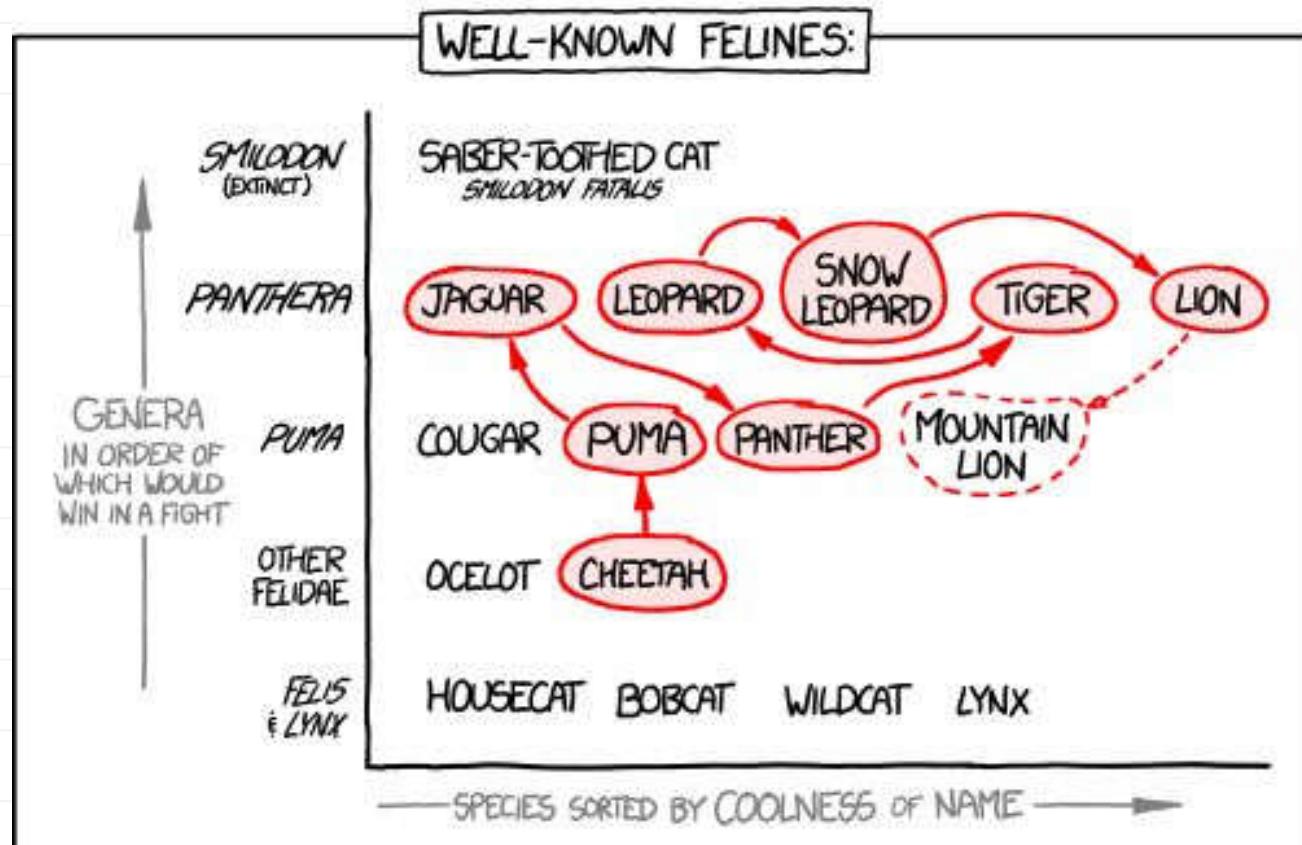


Which of the following statements are correct?

### Select one or more:

- A. None of the other options
- B.  $p_2(u, v)$  is continuous everywhere
- C. The estimate  $p_2(u, v)$  is larger or equal to  $p_1(u, v)$  everywhere
- D. Both kernels are computed from data points within a circle in Euclidean space, centred around  $\mathbf{x}_0$ .
- E.  $p_1(0, 0) > p_2(0, 0)$

<https://xkcd.com/1056/>



### THE OS X PROBLEM

#### Announcements:

- Guest lecture next Wed  
PHYS T + Teams  
**ML for cyber security**  
See you here!
- Two graphical model tutorials:  
week 10 and week 11
- Final exam Fri 3 June  
5:40 - 8:40 pm Canberra
  - On Wattle
  - Self-invigilated, video recording needed
  - Instructions will be available



# Markov random fields

Bishop 8.3, 8.4.1-8.4.2

What are MRFs

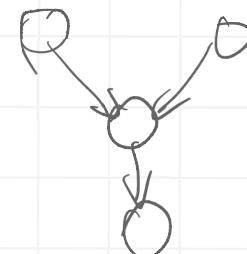
Conditional independence and factorisation

Relation to directed graphs

Inference in graphical models (part 1) - chains and trees

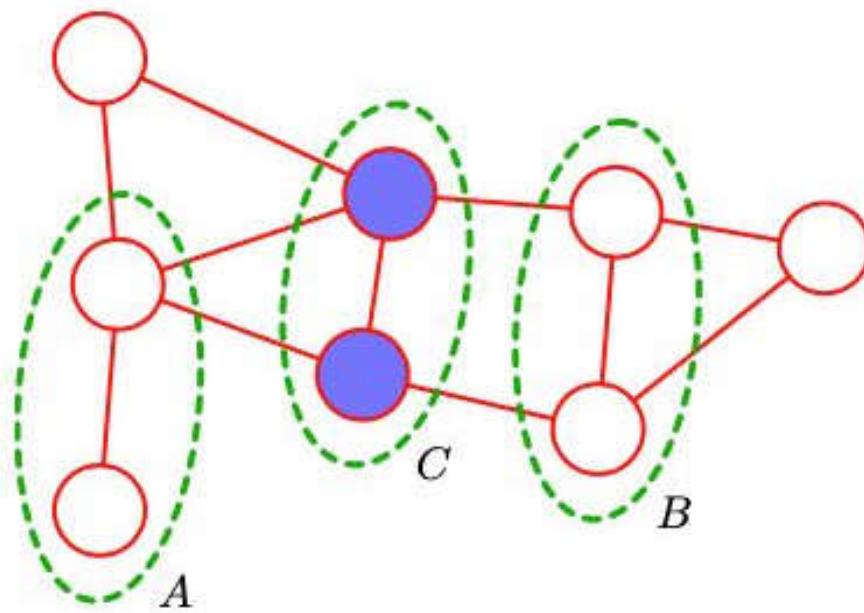
# Markov Random fields

- Markov Random Fields (Markov network, undirected graphical model) are defined over a graph with undirected edges.
- MRFs allow for different conditional independence statements than Bayesian networks.
- In a Bayesian network, the definition of a blocked path was subtle for a HH-node because it did include that all descendants were unobservable.
- Is there an alternative graphical semantics for probability distributions such that conditional independence is determined by simple graph separation?
- Yes, removing the direction from the edges removes the asymmetry between parent and child nodes and subsequently the subtleties associated with the HH-node.



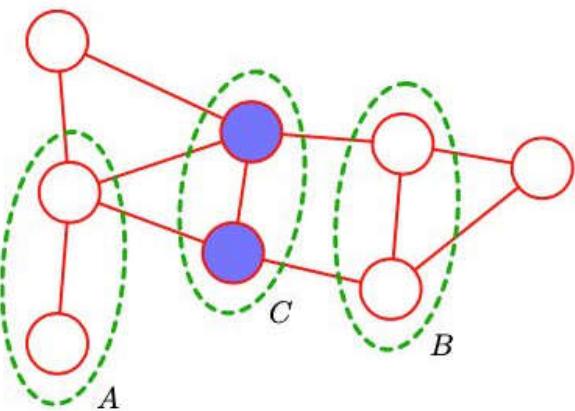
## Definition: Undirected graph separation

In an undirected graph  $G$ , having  $A$ ,  $B$  and  $C$  disjoint subsets of nodes, if every path from  $A$  to  $B$  includes at least one node from  $C$ , then  $C$  is said to separate  $A$  from  $B$  in  $G$ .



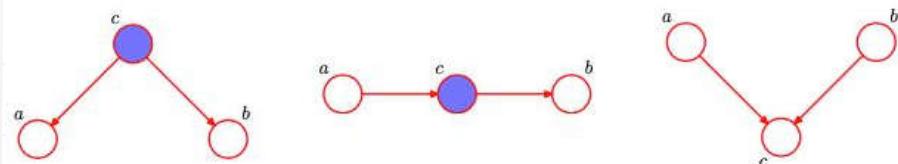
# Separation in undirected vs directed graphs

In an undirected graph  $G$ , having  $A$ ,  $B$  and  $C$  disjoint subsets of nodes, if every path from  $A$  to  $B$  includes at least one node from  $C$ , then  $C$  is said to **separate  $A$  from  $B$**  in  $G$ .



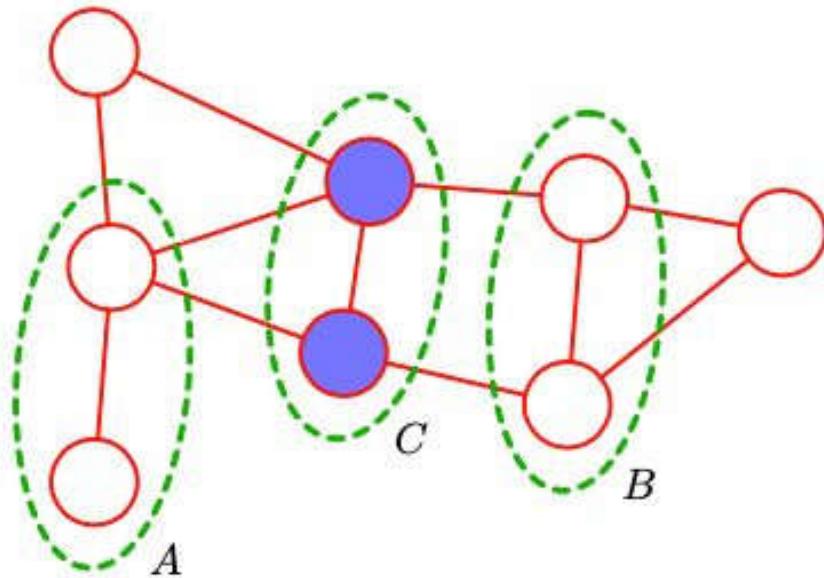
BN , Monday .

- Consider a general directed graph in which  $A$ ,  $B$ , and  $C$  are arbitrary non-intersecting sets of nodes. (There may be other nodes in the graph which are not contained in the union of  $A$ ,  $B$ , and  $C$ .)
- Consider all possible paths from any node in  $A$  to any node in  $B$ .
- Any such path is blocked, if it includes a node such that either
  - the node is HT or TT, and the node is in set  $C$ , or
  - the node is HH, and neither the node, nor any of the descendants, is in set  $C$ .
- If all paths are blocked, then  $A$  is  $d$ -separated from  $B$  by  $C$ , and the joint distribution over all the variables in the graph will satisfy  $A \perp\!\!\!\perp B | C$ .



# Conditional independence in MRFs

**Figure 8.27** An example of an undirected graph in which every path from any node in set  $A$  to any node in set  $B$  passes through at least one node in set  $C$ . Consequently the conditional independence property  $A \perp\!\!\!\perp B \mid C$  holds for any probability distribution described by this graph.



### *Definition (Markov Random Field)*

A Markov Random Field is a set of probability distributions  $\{ p(\mathbf{x}) \mid p(\mathbf{x}) > 0, \forall \mathbf{x} \}$  such that there exists an undirected graph  $G$  with disjoint subsets of nodes  $A$ ,  $B$  and  $C$ , in which whenever  $C$  separates  $A$  from  $B$  in  $G$ ,

$$A \perp\!\!\!\perp B \mid C.$$

- Although we sometimes say "the MRF is such an undirected graph", we mean "the MRF represents the set of all probability distributions whose conditional independency statements are precisely those given by graph separation in the graph".

## Factorisation in an MRF

- Assume two nodes  $x_i$  and  $x_j$  that are not connected by an edge.
- Given all other nodes in the graph,  $x_i$  and  $x_j$  must be conditionally independent as all paths between  $x_i$  and  $x_j$  are blocked by observed nodes.

$$x_i \perp\!\!\!\perp x_j \mid \tilde{x}_{\setminus \{i,j\}}$$

$$p(x_i, x_j | \mathbf{x}_{\setminus \{i,j\}}) = p(x_i | \mathbf{x}_{\setminus \{i,j\}}) p(x_j | \mathbf{x}_{\setminus \{i,j\}}) \quad (8.38)$$

all nodes except  $x_i, x_j$

where  $\mathbf{x}_{\setminus \{i,j\}}$  denotes the set of all variables  $\mathbf{x}$  with  $x_i$  and  $x_j$  removed.

- This is suggestive of the importance of considering sets of connected nodes (cliques). Can we use this for the factorisation of the graph?

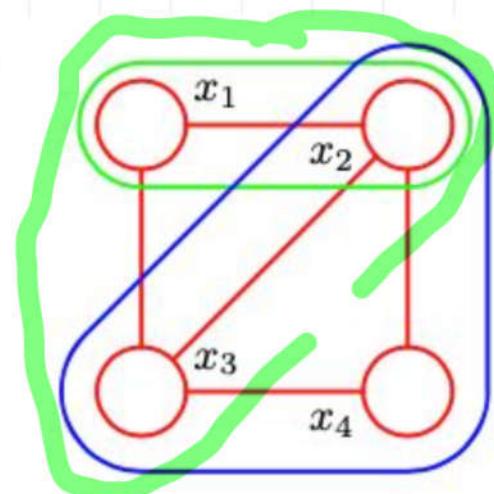
# Cliques in graphs

A **clique** is a subset of nodes in a graph such that there exists an edge between all pairs of nodes in the subset. (The nodes in a clique are fully connected.)

A **maximal clique** of a graph is a clique which is not a proper subset of another clique. (No other nodes of the graph can be added to a maximal clique without destroying the property of full connectedness.)

**Figure 8.29** A four-node undirected graph showing a clique (outlined in green) and a maximal clique (outlined in blue).

~~$\psi(x_1, x_2)$~~   
 $\psi(x_1, x_2, x_3)$   
 $\psi(x_2, x_3, x_4)$



## Factorisation using cliques

$x_C$ : nodes in  
maximal clique

A probability distribution  $p(\mathbf{x})$  **factorises** with respect to a given undirected graph  $G$  if it can be written as

$$p(\mathbf{x}) = \frac{1}{Z} \prod_C \psi_C(\mathbf{x}_C). \quad (8.39)$$

Potential function

$$\psi_C(\mathbf{x}_C) \geq 0$$

where  $\mathcal{C}$  is the set of maximal cliques of  $G$ , and **potential functions**  $\psi_C(\mathbf{x}_C) \geq 0$ . The constant  $Z = \sum_{\mathbf{x}} p(\mathbf{x})$  ensures the correct normalisation of  $p(\mathbf{x})$ .

Normalisation factor / **partition function**

$$Z = \sum_{\mathbf{x}} \prod_C \psi_C(\mathbf{x}_C) \quad (8.40)$$

Major limitation of MRFs, **computing this for the entire graph is usually hard.**

# Conditional independence and factorisation

*Theorem (Factorisation  $\Rightarrow$  Conditional Independence)*

*If a probability distribution factorises according to an undirected graph, and if  $A$ ,  $B$  and  $C$  are disjoint subsets of nodes such that  $C$  separates  $A$  from  $B$  in the graph, then the distribution satisfies  $A \perp\!\!\!\perp B | C$ .*

*Theorem (Conditional Independence  $\Rightarrow$  Factorisation*

*(Hammersley-Clifford Theorem)) (1990)*

*If a strictly positive probability distribution  $p(\mathbf{x}) > 0, \forall \mathbf{x}$ , satisfies the conditional independence statements implied by graph separation over a particular undirected graph, then it also factorises according to the graph.*

## Strictly positive potential functions

As the potential functions  $\psi_C(\mathbf{x}_C)$  are strictly positive, one can express them as exponential

$$\psi_C(\mathbf{x}_C) = \exp \{-E(\mathbf{x}_C)\}$$

of an **energy function**  $E(\mathbf{x}_C)$ .

The exponential distribution is called the **Boltzmann distribution**.

The joint distribution is defined as the product of the potentials, and so the total energy is obtained by adding the energies of each of the maximal cliques.

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(\mathbf{x}_C) = \frac{1}{Z} \exp \left\{ - \sum_{C \in \mathcal{C}} E(\mathbf{x}_C) \right\}$$

Configuration:

$x_1 x_2 x_3$	$\psi(x_1, x_2, x_3)$
0 0 0	0.6
0 0 1	0.2
- - -	
- - ,	
1 1 0	0.2
1 1 1	1.2

Intuition

Potential function: expressing which configurations of the local variables are preferred to others.

Global configurations with relatively high probability: those with a good balance in satisfying the (possibly conflicting) influences of the clique potentials.

## Example: MRF for image denoising

- Given an unknown and noise-free image described by binary pixels  $x_i \in \{-1, +1\}$  for  $i = 1, \dots, D$ .
- Randomly flip the sign of some pixels with some small probability and denote the pixels of the noisy image by  $y_i$  for  $i = 1, \dots, D$ .
- Goal : Recover the original noise-free image.



Original image.



After randomly changing  
10% of the pixels.

**Figure 8.31** An undirected graphical model representing a Markov random field for image de-noising, in which  $x_i$  is a binary variable denoting the state of pixel  $i$  in the unknown noise-free image, and  $y_i$  denotes the corresponding value of pixel  $i$  in the observed noisy image.

$$x_i \in \{1, 0\}$$

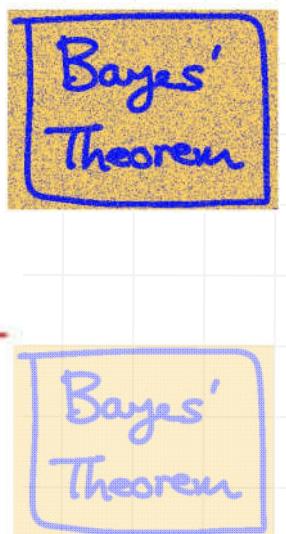
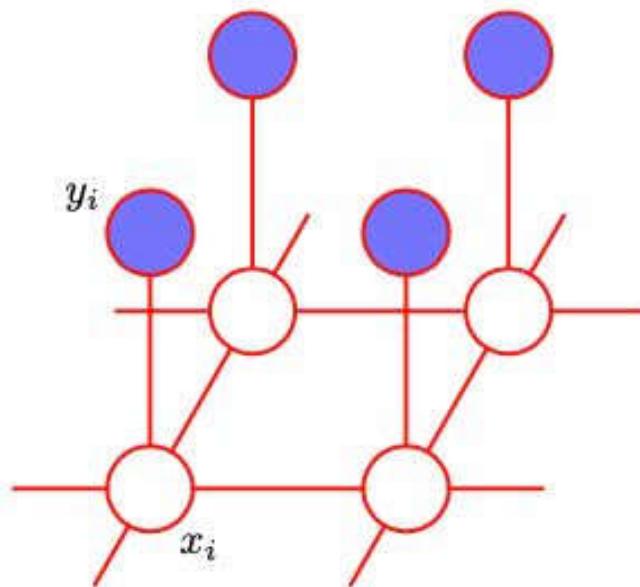
A pixel is more likely to be background.

Neighbouring pixels tend to be the same.

A pixel and the corresponding corrupted pixel tend to be the same

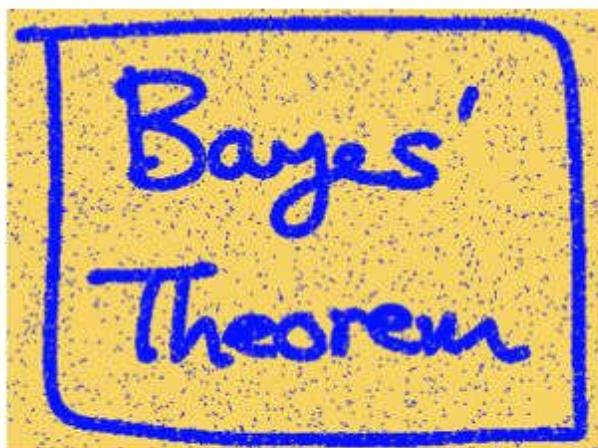
$$E(\mathbf{x}, \mathbf{y}) = h \sum_i x_i - \beta \sum_{\{i,j\}} x_i x_j - \eta \sum_i x_i y_i \quad (8.42)$$

$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \exp\{-E(\mathbf{x}, \mathbf{y})\}. \quad (8.43)$$

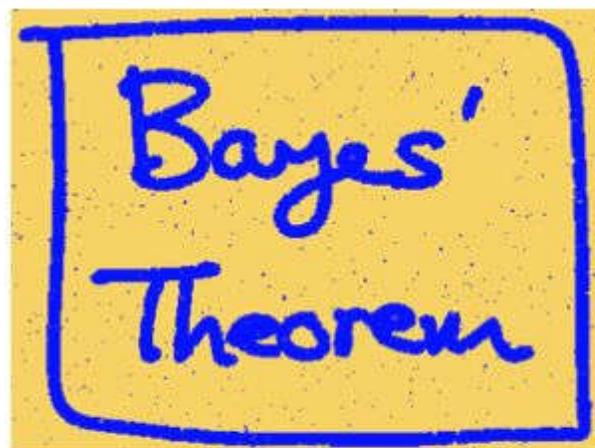


- ① Fix the elements for  $\mathbf{y}$  as we have them observed.  
(Implicitly defines  $p(\mathbf{x} | \mathbf{y})$ ).
- ② Initialise  $x_i = y_i$  for  $i = 1, \dots, D$ .
- ③ Take one node  $x_j$  and evaluate the total energy for both possible states of  $x_j = \{-1, +1\}$  keeping all other variables fixed. Set  $x_j$  to the state having the lower energy.
- ④ Repeat for another node until stopping criterion is satisfied.

$$\beta = 1.0, \eta = 2.1 \text{ and } h = 0.$$



Local minimum (ICM).



Global minimum (graph-cut).

# Graphical model 2

What are MRFs

Conditional independence and factorisation

Relation to directed graphs

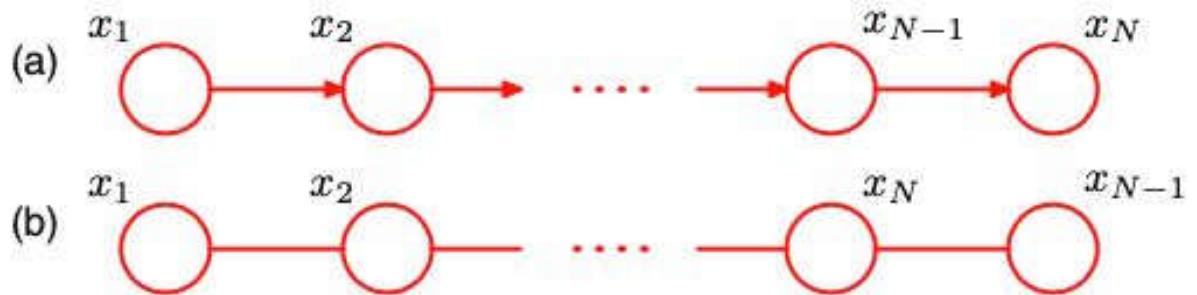
Inference in graphical models (part 1)

# Directed vs undirected chain graphs

$$p(\mathbf{x}) = p(x_1)p(x_2|x_1)p(x_3|x_2) \cdots p(x_N|x_{N-1}). \quad (8.44)$$

$$p(\mathbf{x}) = \frac{1}{Z} \psi_{1,2}(x_1, x_2) \psi_{2,3}(x_2, x_3) \cdots \psi_{N-1,N}(x_{N-1}, x_N). \quad (8.45)$$

**Figure 8.32** (a) Example of a directed graph. (b) The equivalent undirected graph.



$$\psi_{1,2}(x_1, x_2) = p(x_1)p(x_2|x_1)$$

$$\psi_{2,3}(x_2, x_3) = p(x_3|x_2)$$

$\vdots$

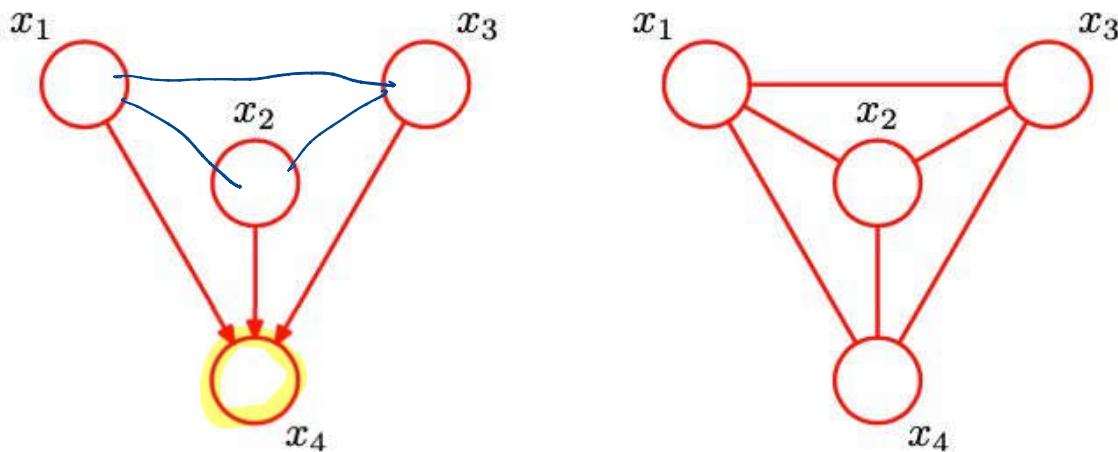
$$\psi_{N-1,N}(x_{N-1}, x_N) = p(x_N|x_{N-1})$$

Directed vs undirected chain graphs  
have the same factors.

Partition function  $Z = 1$

# Moralisation

- For other kind of Bayesian Networks (BNs), create the cliques of a MRF by adding undirected edges between all pairs of parents for each node in the graph.
- This process of 'marrying the parents' is called **moralisation**, and the result is a **moral graph**.
- BUT the resulting MRF may represent different conditional independence statements than the original BN.
- Example: The MRF is fully connected, and exhibits NO conditional independence properties, in contrast to the original directed graph.



BN: DAG  $\rightarrow$  no cliques

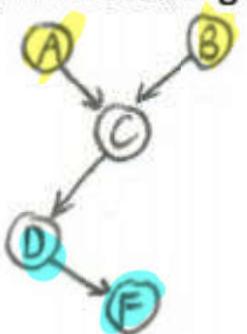
MRF: all about clique,

$$\begin{aligned} & P(x_1) P(x_2) P(x_3) \\ & P(x_4 | x_1, x_2, x_3) \\ \downarrow & \\ & \psi(x_1, x_2, x_3, x_4) \end{aligned}$$

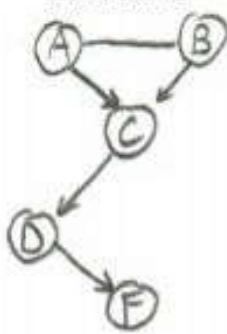
**1. Are A and B conditionally independent, given D and F?**

(Same as " $P(A|BDF) =? P(A|DF)$ " or " $P(B|ADF) =? P(B|DF)$ ")

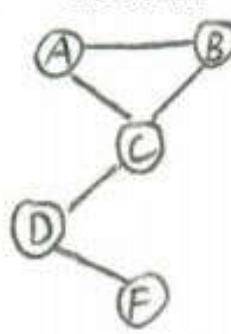
Draw ancestral graph



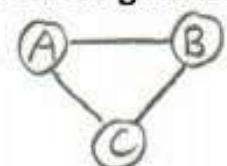
Moralize



Disorient



Delete givens



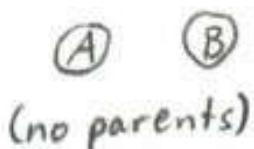
Answer: No, A and B are connected, so they are not required to be conditionally independent given D and F.

**2. Are A and B marginally independent? (Same as " $P(A|B) =? P(A)$ " or " $P(B|A) =? P(B)$ ")**

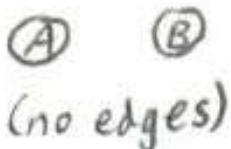
Draw ancestral graph



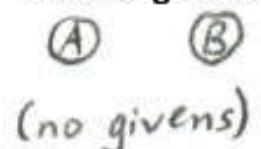
Moralize



Disorient



Delete givens



Answer: Yes, A and B are not connected, so they are marginally independent.

### *Definition (D-map)*

A graph is a **D-map** (dependency map) of a distribution if every conditional independence statement satisfied by the distribution is reflected in the graph.

### *Definition (I-map)*

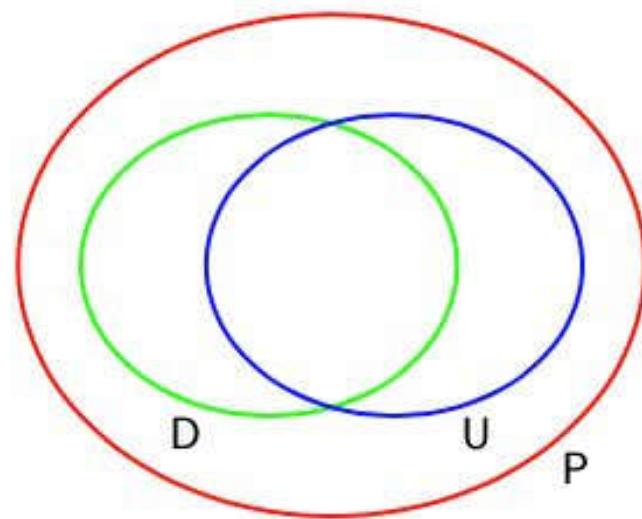
A graph is an **I-map** (independence map) of a distribution if every conditional independence statement implied by the graph is satisfied in the distribution.

### *Definition (P-map)*

A graph is a **P-map** (perfect map) of a distribution if it is both a D-map and an I-map for the distribution.

$$P(\vec{x})$$

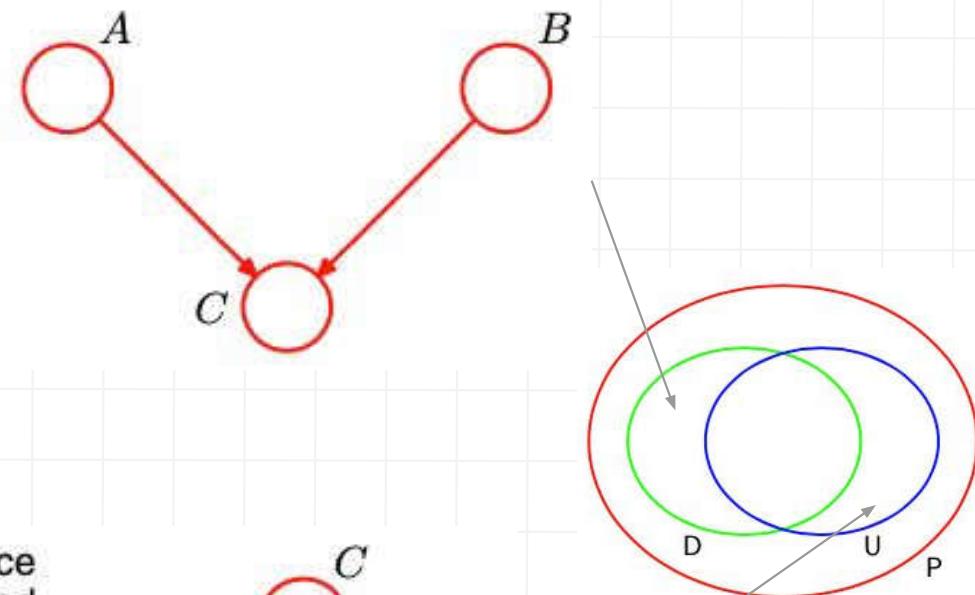
**Figure 8.34** Venn diagram illustrating the set of all distributions  $P$  over a given set of variables, together with the set of distributions  $D$  that can be represented as a perfect map using a directed graph, and the set  $U$  that can be represented as a perfect map using an undirected graph.



Undirected :  $A \perp\!\!\!\perp B \mid C$

**Figure 8.35** A directed graph whose conditional independence properties cannot be expressed using an undirected graph over the same three variables.

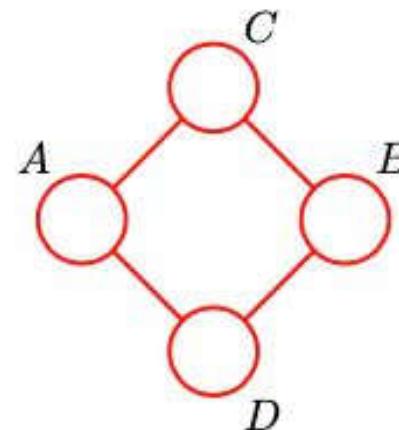
$A \perp\!\!\!\perp B \mid \emptyset$



**Figure 8.36** An undirected graph whose conditional independence properties cannot be expressed in terms of a directed graph over the same variables.

$A \perp\!\!\!\perp B \mid C, D$

$C \perp\!\!\!\perp D \mid A, B$



# Bayes Net and MRFs

In both types of Graphical Models

- A relationship between the conditional independence statements satisfied by a distribution and the associated simplified algebraic structure of the distribution is made in term of graphical objects.
- The conditional independence statements are related to concepts of separation between variables in the graph.
- The simplified algebraic structure (factorisation of  $p(\mathbf{x})$ ) is related to 'local pieces' of the graph (child + its parents in BNs, cliques in MRFs).

## Differences

- The set of probability distributions that can be represented as MRFs is different from the set that can be represented as BNs.
- Although both MRFs and BNs are expressed as a factorisation of local functions on the graph, the MRF has a normalisation constant  $Z = \sum_{\mathbf{x}} \prod_{C \in c} \psi_C(\mathbf{x}_C)$  that couples all factors, whereas the BN has not.
- The local 'pieces' of the BN are probability distributions themselves, whereas in MRFs they need only be non-negative functions (i.e. they may not have range  $[0, 1]$  as probabilities do).

# Graphical model 2

What are MRFs

Conditional independence and factorisation

Relation to directed graphs

Inference in graphical models (part 1)

# Inference in graphical models

$$P(\vec{X})$$

compute marginal

$$P(X_i)$$

$$P(x_i, x_j, \dots)$$

conditionals  $P(X_i | X_j_1, X_j_2, \dots)$

Inference in graphical models: with some of the nodes in a graph are clamped to observed values, compute the posterior distributions of one or more subsets of other nodes.

General ideas:

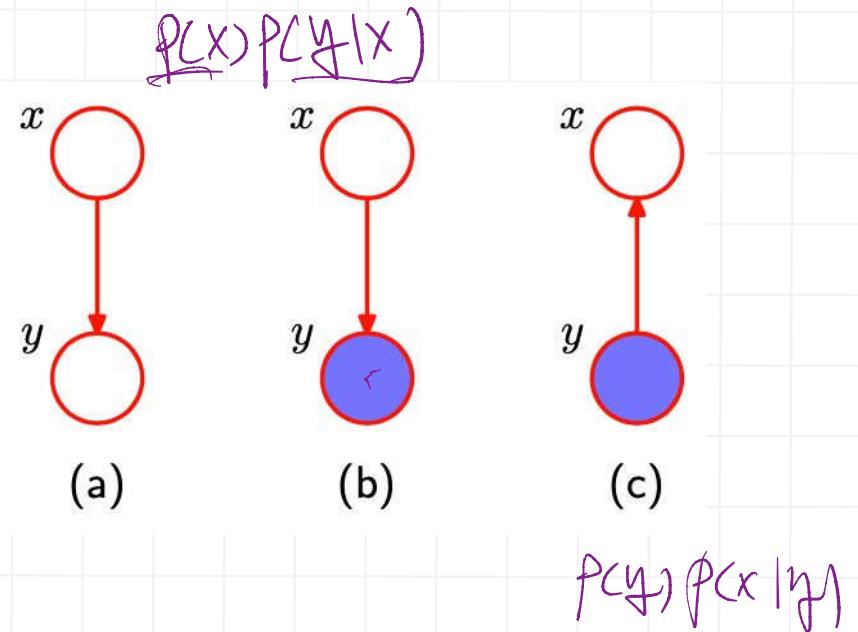
- Exploit graphical structure - efficiency, transparency
- Many algorithms expressed in local “message propagation” around the graph
- Focusing on exact inference in this class  
(see Chap 10 for approximate inference algorithms)

Is an essential step in learning (c.f. EM algorithm)

e.g. image denoising,  
inference:  $(\beta, \gamma, h)$   
learning: estimate  $\beta, \gamma, h$ ,

# Inference on the simplest graphical model

**Figure 8.37** A graphical representation of Bayes' theorem. See the text for details.



$$p(y) = \sum_{x'} p(y|x')p(x') \quad (8.47)$$

$$\underline{p(x|y)} = \frac{p(y|x)p(x)}{p(y)}. \quad (8.48)$$

A handwritten note next to the equation indicates that the term  $p(y|x)p(x)$  is "known".

## Inference on a chain graph

$$p(\vec{x} \mid \psi) = \frac{1}{Z} \left( \sum_{x_1} \psi_{1,2} \psi_2 \dots \underbrace{\psi_{N-1,N}}_{\text{const wrt } x_1} \right)$$

$$p(\mathbf{x}) = \frac{1}{Z} \psi_{1,2}(x_1, x_2) \psi_{2,3}(x_2, x_3) \dots \underbrace{\psi_{N-1,N}(x_{N-1}, x_N)}, \quad (8.45) \text{ also (8.49)}$$

Want to compute  $p(x_n)$ , naive algorithm:  $O(K^{N-1})$

$$p(x_n) = \sum_{x_1} \dots \sum_{x_{n-1}} \sum_{\underbrace{x_{n+1}}_{\text{"nodes before, after" } x_n}} \dots \sum_{x_N} p(\mathbf{x}). \quad (8.50)$$

Fig 8.32(b)



(8.45) also (8.49)

removing  $x_N$

$$\sum_{x_N} \underbrace{\psi_{N-1,N}(x_{N-1}, x_N)}_{\text{after } x_N} \quad (8.51)$$

$$p(x_n) = \frac{1}{Z}$$

$$\left[ \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \cdots \left[ \sum_{x_2} \psi_{2,3}(x_2, x_3) \left[ \sum_{x_1} \psi_{1,2}(x_1, x_2) \right] \cdots \right] \right]$$

$$\mu_\alpha(x_n)$$

$$\left[ \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \cdots \left[ \sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N) \right] \cdots \right]. \quad (8.52)$$

$k$  terms

$$p(x_n) = \frac{1}{Z} \mu_\alpha(x_n) \mu_\beta(x_n).$$

$$(8.54)$$

removing  $x_1$

$\mu_\alpha(x_n)$

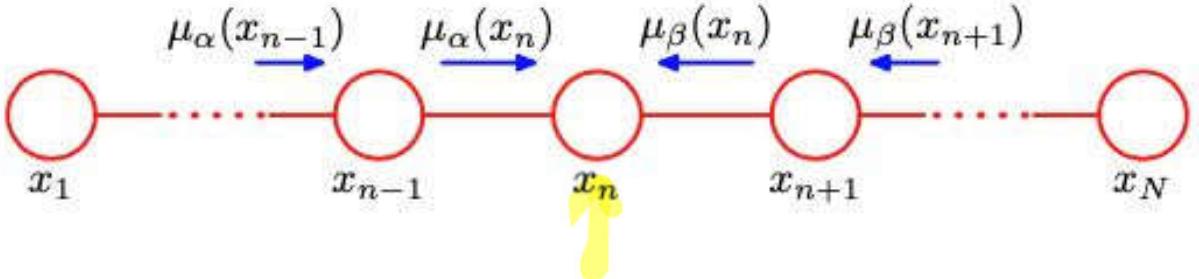
removing  $x_N$

removing  $x_2$

$\mu_\beta(x_n)$

$O(NK^2)$

**Figure 8.38** The marginal distribution  $p(x_n)$  for a node  $x_n$  along the chain is obtained by multiplying the two messages  $\mu_\alpha(x_n)$  and  $\mu_\beta(x_n)$ , and then normalizing. These messages can themselves be evaluated recursively by passing messages from both ends of the chain towards node  $x_n$ .



## Recursive evaluation of messages

$$\mu_\alpha(x_2) = \sum_{x_1} \psi_{1,2}(x_1, x_2) \quad (8.56)$$

$$\begin{aligned} \mu_\alpha(x_3) &= \sum_{x_2} \psi_{2,3}(x_2, x_3) \mu_\alpha(x_2) \\ \mu_\alpha(x_n) &= \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \left[ \sum_{x_{n-2}} \dots \right] \\ &= \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \mu_\alpha(x_{n-1}). \end{aligned} \quad (8.55)$$

$$\begin{aligned} \mu_\beta(x_n) &= \sum_{x_{n+1}} \psi_{n+1,n}(x_{n+1}, x_n) \left[ \sum_{x_{n+2}} \dots \right] \\ &= \sum_{x_{n+1}} \psi_{n+1,n}(x_{n+1}, x_n) \mu_\beta(x_{n+1}). \end{aligned} \quad (8.57)$$

brute-force:

$$O(K^{N-1})$$

## Other marginals and conditional distributions

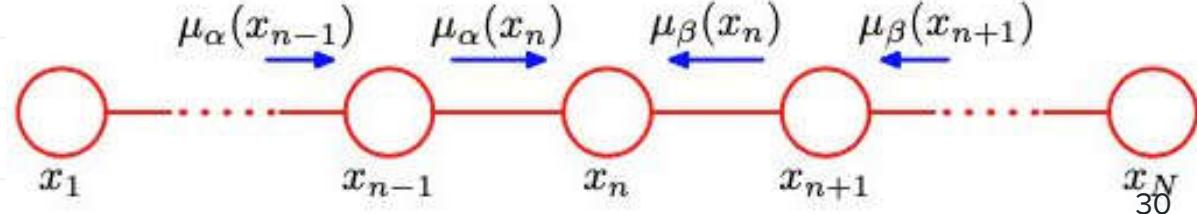
- Computing all marginals
  - Repeat the above N times  $O(N^2K^2)$
  - Storing all intermediate messages:  $O(N^2K^2)$
- Observed nodes
  - Clamp instead of sum over
- Computing joint probabilities

foreach  $p(x_n)$   $O(NK^2)$

$p(x_5 | x_3=2) \rightarrow K$  vector.

$p(x_5 | x_3) \rightarrow K \times K$  table

$$p(x_{n-1}, x_n) = \frac{1}{Z} \mu_\alpha(x_{n-1}) \psi_{n-1,n}(x_{n-1}, x_n) \mu_\beta(x_n). \quad (8.58)$$



# Tree-shaped graphical models

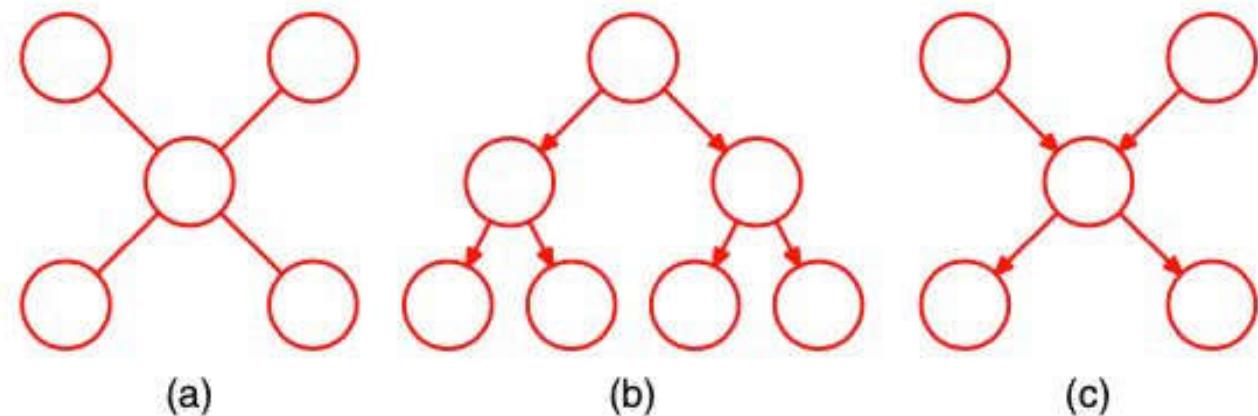
Undirected graph: there is one, and only one, path between any pair of nodes → there is no loops.

Directed graphs: there is a single node, called the root, which has no parents, and all other nodes have one parent.

Moralisation will not add links → convert between directed and undirected tree.

Inference can be done similarly → sum-product algorithm, stay tuned ...

**Figure 8.39** Examples of tree-structured graphs, showing (a) an undirected tree, (b) a directed tree, and (c) a directed polytree.



# Markov random fields

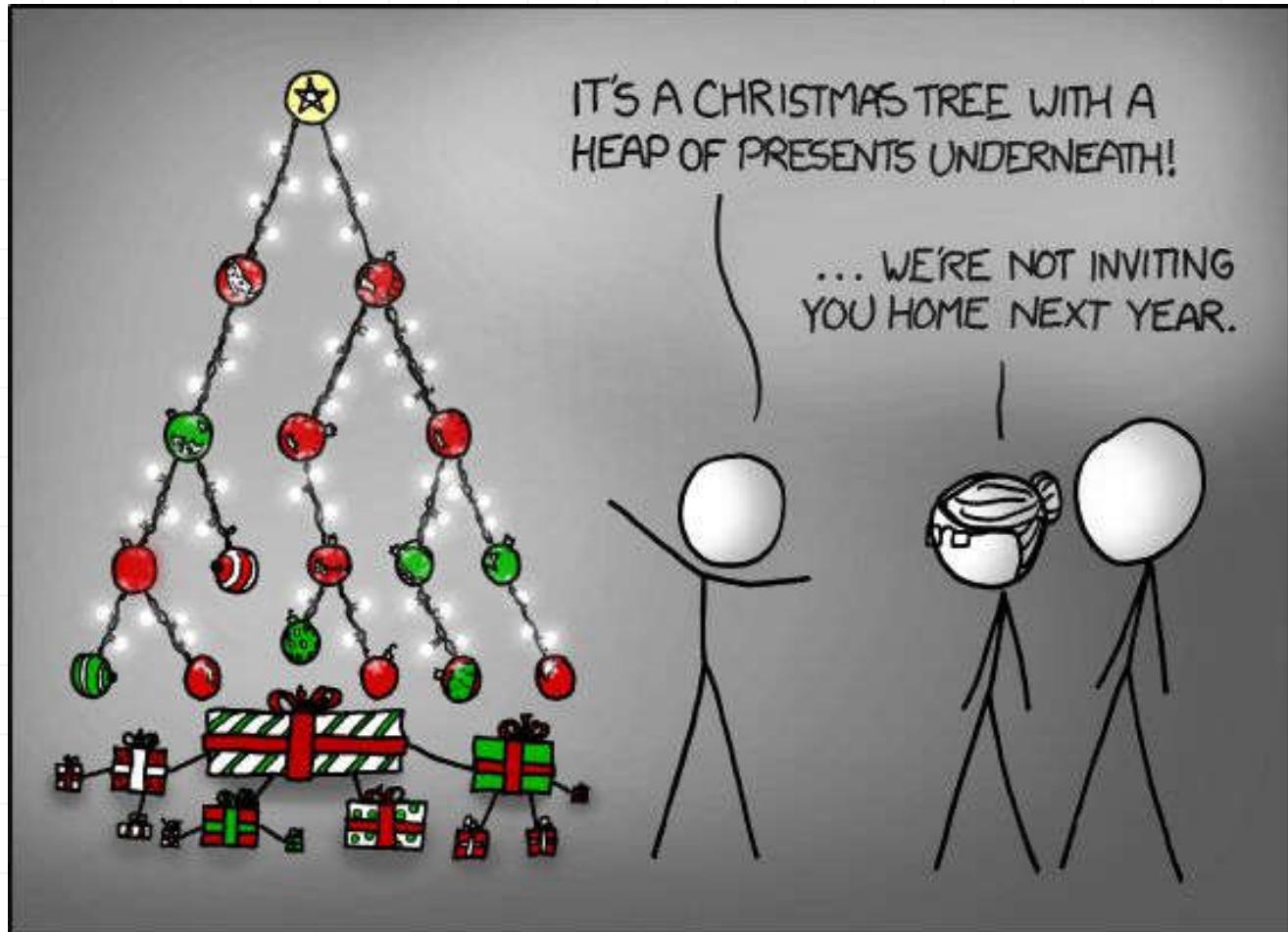
What are MRFs

Conditional independence and factorisation

Relation to directed graphs

Inference in graphical models (part 1) - chains and trees

<https://xkcd.com/835/>



### Announcements:

- Guest lecture this Wed  
PHYS T + Teams  
**ML for cyber security**  
See you here!
- Two graphical model tutorials: this week and next week.
- Hope you're going well in assignment 2
- Final exam Fri 3 June  
5:40 - 8:40 pm Canberra
  - On Wattle
  - Self-invigilated, video recording needed
  - Instructions will be available



# Graphical model inference

Bishop 8.4.3, 8.4.4

Factor graphs

The sum-product algorithm

Other algorithms

High-level goal:

One representation for both directed and undirected graphical models.

Use this representation to facilitate inference.

[Frey, 1998, Kschischang et al 2001]



Factor graphs and the sum-product algorithm

FR Kschischang, BJ Frey, HA Loeliger

IEEE Transactions on information theory 47 (2), 498-519

7780

2001

# Factor graphs

*undirected  
directed graphical model.*

Joint distribution  $\rightarrow$  a product of factors

$$p(\mathbf{x}) = \prod_s f_s(\mathbf{x}_s) \quad (8.59)$$

$$p(\mathbf{x}) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3). \quad (8.60)$$

Contains more info than MRF, because  $f_a(x_1, x_2)$  and  $f_b(x_1, x_2)$  would be in one potential function.

$$\psi_a(x_1, x_2)$$

e.g.  $f_a$  relative humidity  
 $f_b$  temperature

Directed graph: conditionals  $\rightarrow$  factors

$$p(\mathbf{x}) = \prod_{k=1}^K p(x_k | \text{pa}_k) \quad (8.5)$$

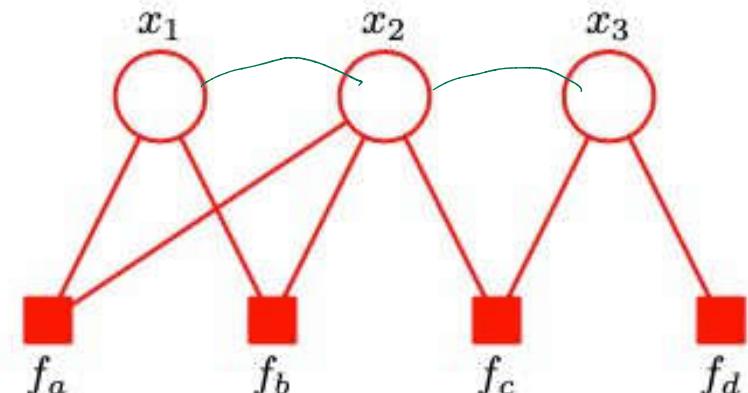
Undirected graph: potential functions over max cliques  $\rightarrow$  factors

$$p(\mathbf{x}) = \frac{1}{Z} \prod_C \psi_C(\mathbf{x}_C). \quad (8.39)$$

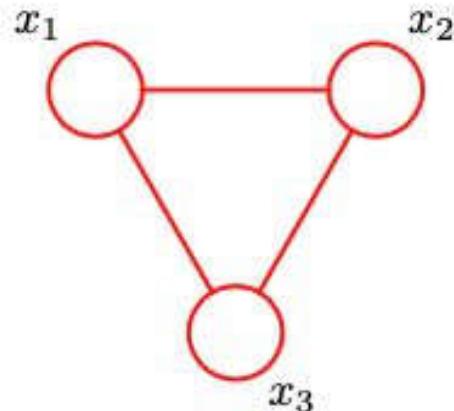
$Z$  is a factor too, over an empty set of variables.

factors are "unnormalised"

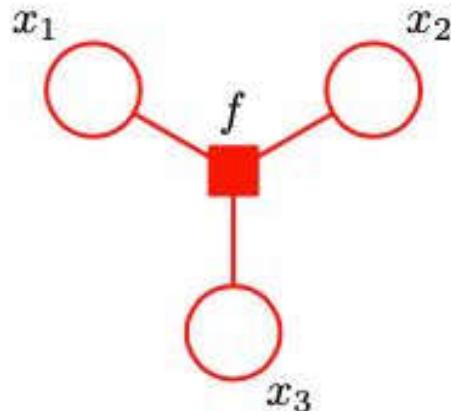
$s$ : a subset of variables



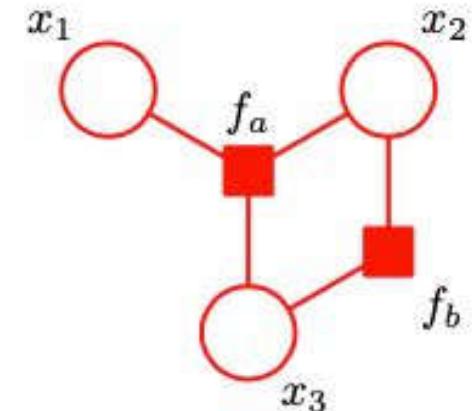
## Example: factor graphs representing the same distribution



(a)



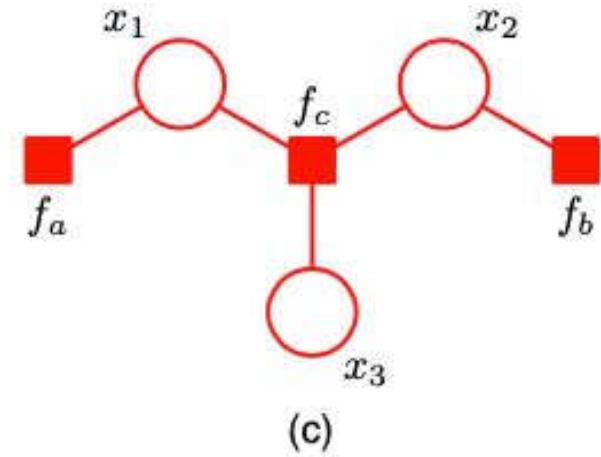
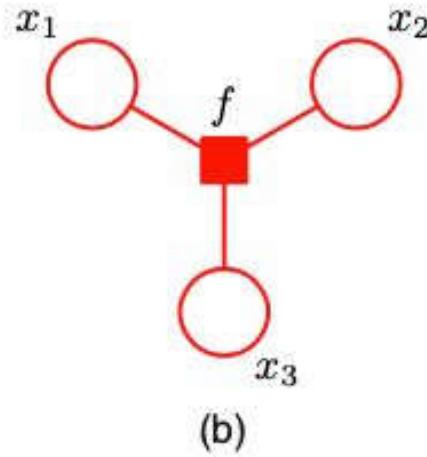
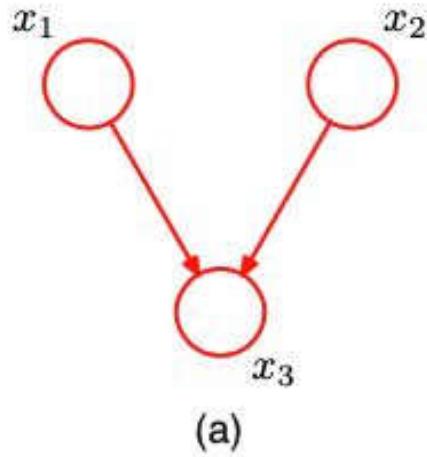
(b)



(c)

**Figure 8.41** (a) An undirected graph with a single clique potential  $\psi(x_1, x_2, x_3)$ . (b) A factor graph with factor  $f(x_1, x_2, x_3) = \psi(x_1, x_2, x_3)$  representing the same distribution as the undirected graph. (c) A different factor graph representing the same distribution, whose factors satisfy  $f_a(x_1, x_2, x_3)f_b(x_1, x_2) = \psi(x_1, x_2, x_3)$ .

## Example: factor graphs representing the same distribution



**Figure 8.42** (a) A directed graph with the factorization  $p(x_1)p(x_2)p(x_3|x_1, x_2)$ . (b) A factor graph representing the same distribution as the directed graph, whose factor satisfies  $f(x_1, x_2, x_3) = p(x_1)p(x_2)p(x_3|x_1, x_2)$ . (c) A different factor graph representing the same distribution with factors  $f_a(x_1) = p(x_1)$ ,  $f_b(x_2) = p(x_2)$  and  $f_c(x_1, x_2, x_3) = p(x_3|x_1, x_2)$ .

# Factor graphs are bipartite

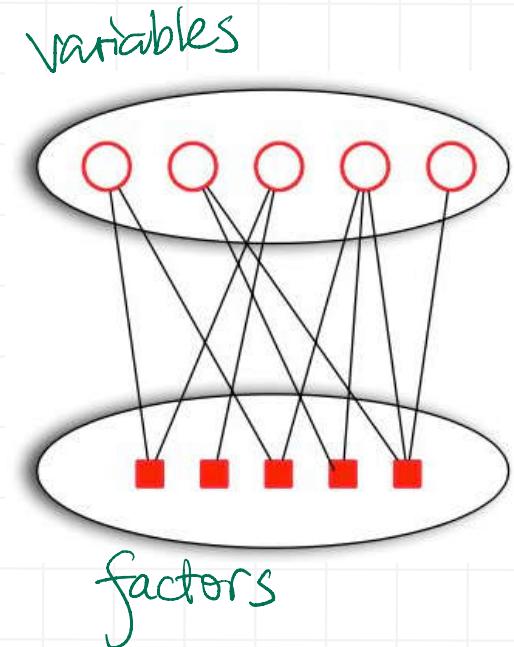
## Definition (Bipartite Graph)

A bipartite graph (or bigraph) is a graph whose vertices can be divided into two disjoint sets  $U$  and  $V$  such that every edge connects a vertex in  $U$  to one in  $V$ .

- ① Create variable nodes for each node in the original graph
- ② Create factor nodes for each maximal clique  $x_s$
- ③ Set the factors  $f_s(x_s)$  to the clique potentials

Note: There may be several different factor graphs corresponding to the same undirected graph.

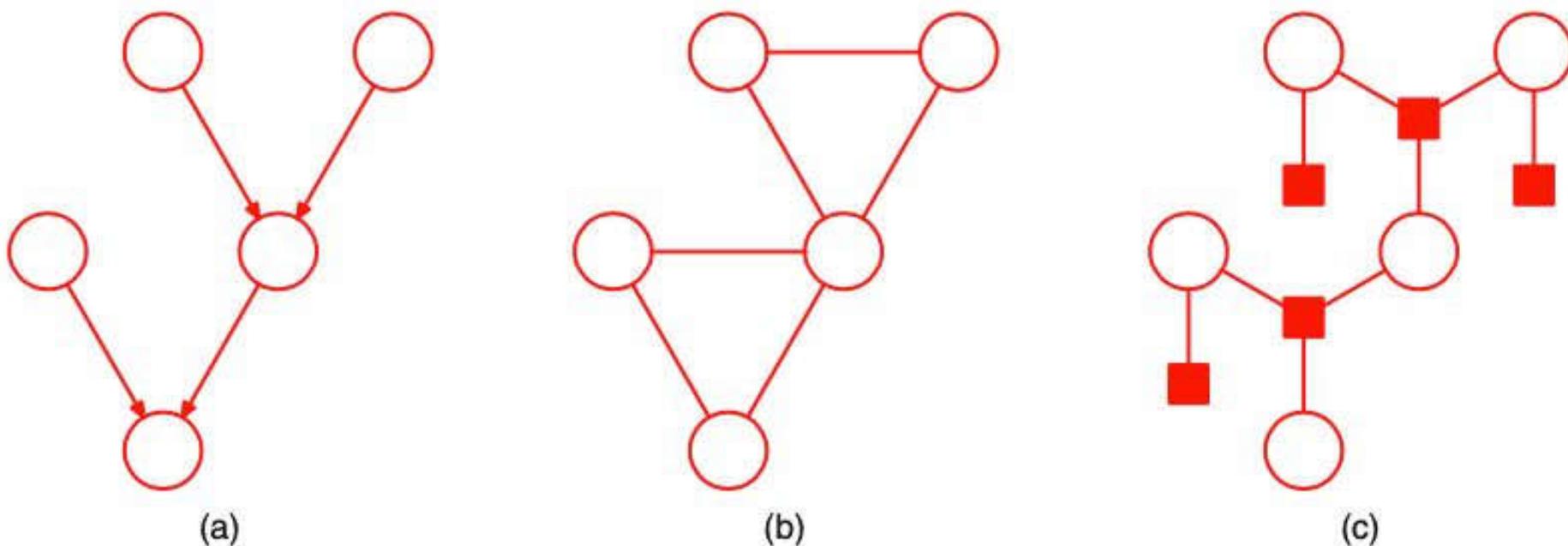
BN  $\xrightarrow{\text{moralisation}}$  MRF  $\rightarrow$  FG



BN+MRF:  
intuitive to think about  
variable dependencies

FG:  
better for constructing inference

## Factor graph for a poly-tree

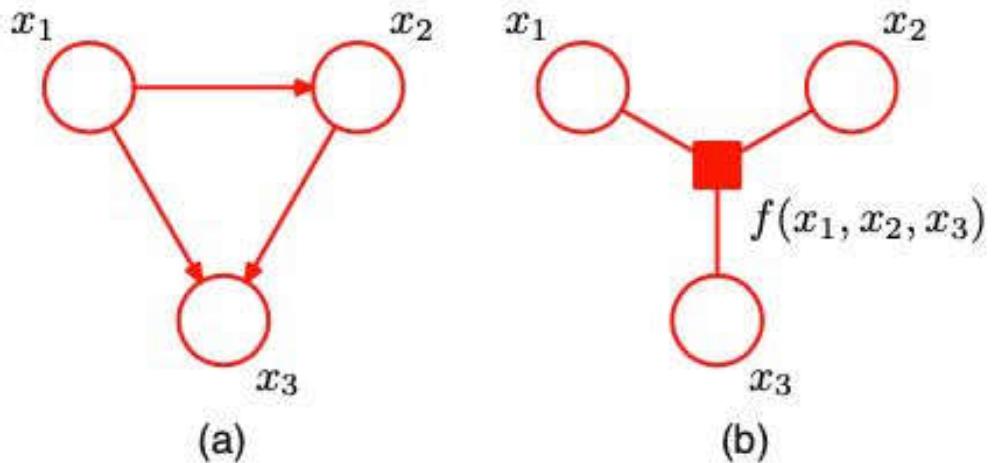


**Figure 8.43** (a) A directed polytree. (b) The result of converting the polytree into an undirected graph showing the creation of loops. (c) The result of converting the polytree into a factor graph, which retains the tree structure.

Factor graphs for directed/undirected trees, and directed poly trees retain a tree structure.

**Figure 8.44**

(a) A fragment of a directed graph having a local cycle. (b) Conversion to a fragment of a factor graph having a tree structure, in which  $f(x_1, x_2, x_3) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)$ .



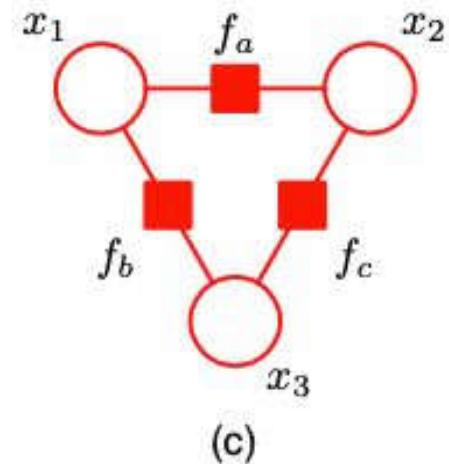
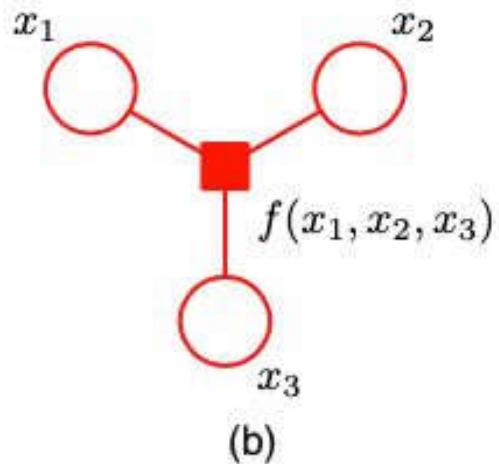
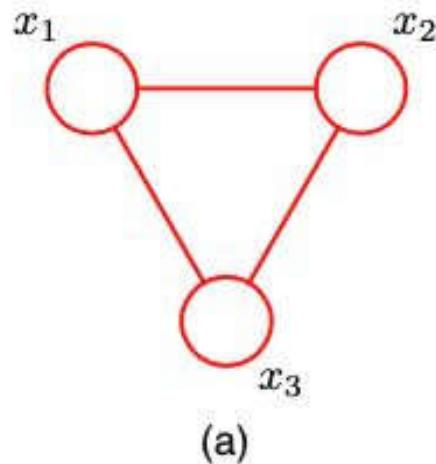
BN ↴ fairly general factorisation

MRF :  $\psi_{123}$



PG  $f(x_1, x_2, x_3)$

## Multiple factor graphs representing the same (undirected) graph



**Figure 8.45** (a) A fully connected undirected graph. (b) and (c) Two factor graphs each of which corresponds to the undirected graph in (a).

Factorization in (c) does not correspond to any conditional independence properties.

FG with loops : “bad” for inference -

$$P(x_1, x_2, x_3, x_4)$$

$$P(x_3) = ?$$

$$P(x_3, x_1) = ?,$$

$$P(x_2 | x_3=1)$$

?

## The sum-product algorithm

Goal: evaluate local marginals over nodes or subsets of nodes.

Variant: find the most probable state  $\rightarrow$  max sum algorithm.

Assume (for instructional convenience): all variables discrete. Marginalization = sums

For continuous variables - perform integration, e.g. linear dynamic systems (e.g. Chap 13.3)

Historic note:

belief propagation (Pearl, 1988; Lauritzen and Spiegelhalter, 1988) perform exact inference on DAGs. It is a special case of the sum product algorithm (Frey, 1998; Kschischang et al., 2001)

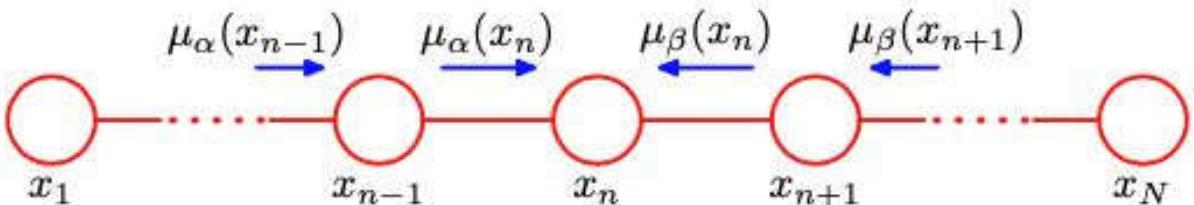
Recall from last lecture ...

$$\begin{aligned}
 p(x_n) &= \frac{1}{Z} \\
 &\left[ \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \cdots \left[ \sum_{x_2} \psi_{2,3}(x_2, x_3) \left[ \sum_{x_1} \psi_{1,2}(x_1, x_2) \right] \right] \cdots \right] \\
 &\underbrace{\left[ \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \cdots \left[ \sum_{x_2} \psi_{2,3}(x_2, x_3) \left[ \sum_{x_1} \psi_{1,2}(x_1, x_2) \right] \right] \cdots \right]}_{\mu_\alpha(x_n)} \\
 &\left[ \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \cdots \left[ \sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N) \right] \cdots \right]. \\
 &\underbrace{\left[ \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \cdots \left[ \sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N) \right] \cdots \right]}_{\mu_\beta(x_n)}. \tag{8.52}
 \end{aligned}$$

$O(NK^2)$

$$p(x_n) = \frac{1}{Z} \mu_\alpha(x_n) \mu_\beta(x_n). \tag{8.54}$$

**Figure 8.38** The marginal distribution  $p(x_n)$  for a node  $x_n$  along the chain is obtained by multiplying the two messages  $\mu_\alpha(x_n)$  and  $\mu_\beta(x_n)$ , and then normalizing. These messages can themselves be evaluated recursively by passing messages from both ends of the chain towards node  $x_n$ .



~~p(x) of 1, 2, .. k~~

## The sum-product algorithm

one var  
want marginal dist.

$$p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x}) = \sum_{\mathbf{x} \setminus x} \prod_s f_s(\mathbf{x}_s) \quad (8.61)$$

O( $k^{N-1}$ )

product of  $\{f_s\}$

$$p(\mathbf{x}) = \prod_{s \in \text{ne}(x)} F_s(x, X_s)$$

$$\begin{aligned} p(x) &= \prod_{s \in \text{ne}(x)} \left[ \sum_{X_s} F_s(x, X_s) \right] \\ &= \prod_{s \in \text{ne}(x)} \mu_{f_s \rightarrow x}(x). \end{aligned} \quad \text{rename} \quad (8.63)$$

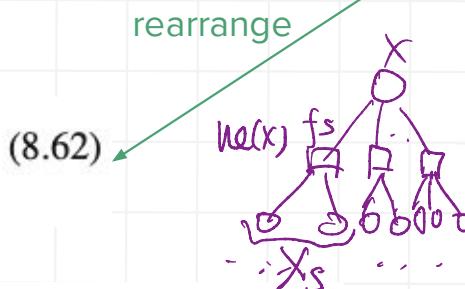
$$\text{where } \mu_{f_s \rightarrow x}(x) \equiv \sum_{X_s} F_s(x, X_s) \quad (8.64)$$

① do summation first (when we can)

② multiply less  $2 \rightarrow 1$

$$ab + ac = a(b + c) \quad (8.53)$$

$$p(\mathbf{x}) = \prod_s f_s(\mathbf{x}_s) \quad (8.59)$$



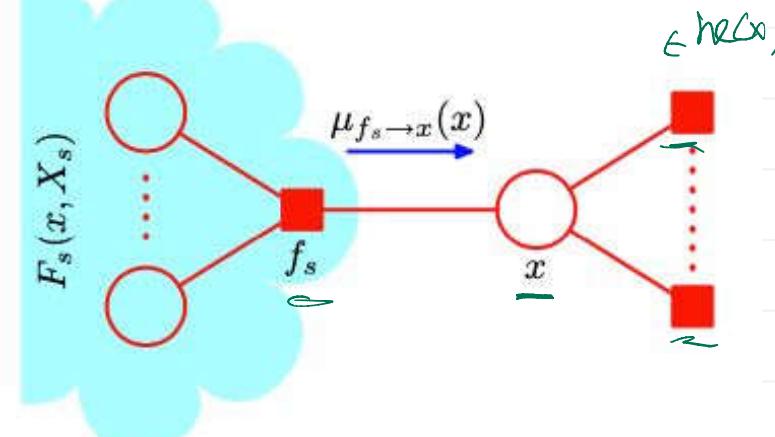
s: a subset of nodes

ne(x): neighbouring factor nodes

Xs: the set of all variables in the subtree connected to the variable node x via the factor node fs

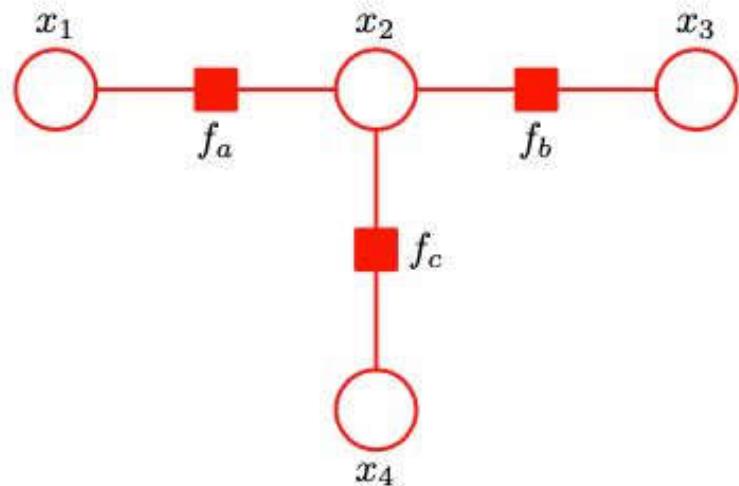
Fs(xs): the product of all the factors in the group associated with factor fs.

Fig 8.46



## A first example of sum-product

$$\begin{aligned}
 p(x_2) &= \sum_{x_1, x_3, x_4} f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4) \\
 &= \left( \sum_{x_1} f_a(x_1, x_2) \right) \left( \sum_{x_3} f_b(x_2, x_3) \right) \left( \sum_{x_4} f_c(x_2, x_4) \right) \\
 &\quad \mu_{f_a \rightarrow x_2}(x_2) \quad \mu_{f_b \rightarrow x_2}(x_2)
 \end{aligned}$$



$$\begin{aligned}
 p(x) &= \prod_{s \in \text{ne}(x)} \left[ \sum_{X_s} F_s(x, X_s) \right] \\
 &= \prod_{s \in \text{ne}(x)} \mu_{f_s \rightarrow x}(x).
 \end{aligned} \tag{8.63}$$

where  $\mu_{f_s \rightarrow x}(x) \equiv \sum_{X_s} F_s(x, X_s)$   

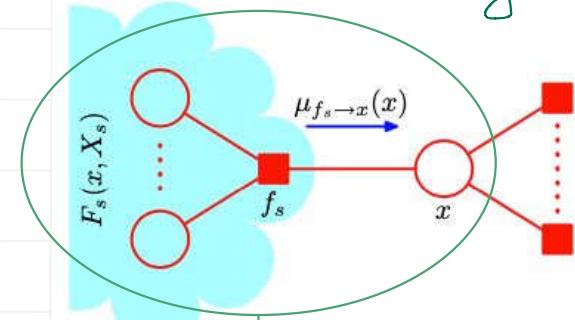
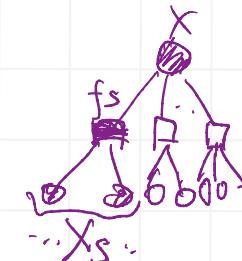
$$\tag{8.64}$$

Fig 8.46

# What about $F_s(x, X_s)$ ?

$F_s(x, X_s)$ : the product of all the factors in the group associated with factor  $f_s$ .

$X_s$ : the set of all variables in the subtree connected to the variable node  $x$  via the factor node  $f_s$



Factorise this factor subgraph:

$$F_s(x, X_s) = \underbrace{f_s(x, x_1, \dots, x_M)}_{(8.65)} G_1(x_1, X_{s1}) \dots G_M(x_M, X_{sM})$$

$$X_s = \{x, x_1, \dots, x_M\}$$

↑ grandchildren of  $x$

Fig 8.47

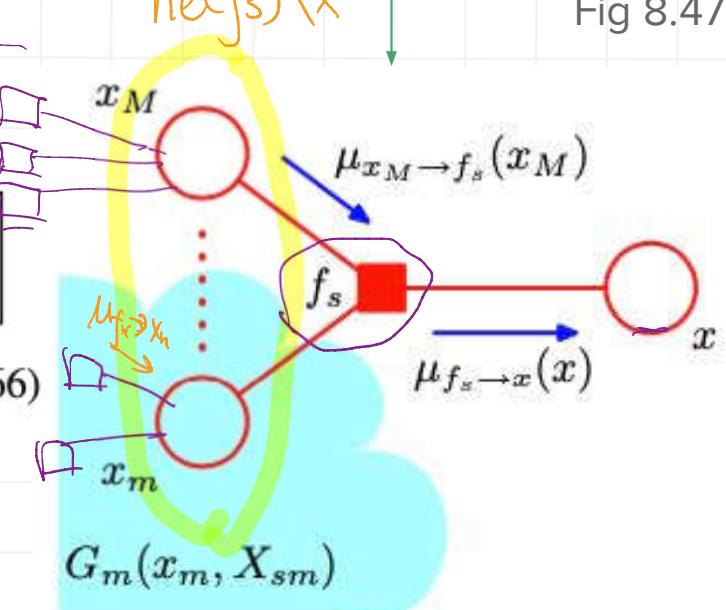
$\text{ne}(f_s)$ : neighbours of factor node  $f_s$

$\text{ne}(f_s) \setminus x$ : neighbours of factor node  $f_s$  except  $x$

$$\begin{aligned} \mu_{f_s \rightarrow x}(x) &= \sum_{x_1} \dots \sum_{x_M} f_s(x, x_1, \dots, x_M) \prod_{m \in \text{ne}(f_s) \setminus x} \left[ \sum_{X_{xm}} G_m(x_m, X_{sm}) \right] \\ &= \sum_{x_1} \dots \sum_{x_M} f_s(x, x_1, \dots, x_M) \prod_{m \in \text{ne}(f_s) \setminus x} \mu_{x_m \rightarrow f_s}(x_m) \end{aligned} \quad (8.66)$$

$$\mu_{x_m \rightarrow f_s}(x_m) \equiv \sum_{X_{sm}} G_m(x_m, X_{sm}). \quad (8.67)$$

node-to-factor messages.



# recap

Two different kinds of messages:

$$\mu_{f_s \rightarrow x}(x) \equiv \sum_{X_s} F_s(x, X_s) \quad (8.64)$$

$$\mu_{x_m \rightarrow f_s}(x_m) \equiv \sum_{X_{sm}} G_m(x_m, X_{sm}). \quad (8.67)$$

Computing factor-to-variable-message

- 1. take the product of the incoming messages along all other links coming into the factor node
  - 2. multiply by the factor associated with that node
  - 3. marginalize over all of the variables associated with the incoming messages
- can send a message once the factor node has received incoming messages from all other neighbouring variable nodes

$$\begin{aligned} \mu_{f_s \rightarrow x}(x) &= \sum_{x_1} \dots \sum_{x_M} f_s(x, x_1, \dots, x_M) \prod_{m \in \text{ne}(f_s) \setminus x} \left[ \sum_{X_{sm}} G_m(x_m, X_{sm}) \right] \\ &= \sum_{x_1} \dots \sum_{x_M} f_s(x, x_1, \dots, x_M) \prod_{m \in \text{ne}(f_s) \setminus x} \mu_{x_m \rightarrow f_s}(x_m) \quad (8.66) \end{aligned}$$

$$\begin{aligned} \mu_{x_m \rightarrow f_s}(x_m) &= \prod_{l \in \text{ne}(x_m) \setminus f_s} \mu_{f_l \rightarrow x_m}(x_m) \\ \mu_{f_s \rightarrow x}(x) &= \sum_{x_1} \dots \sum_{x_M} f_s(x, x_1, \dots, x_M) \prod_{m \in \text{ne}(f_s) \setminus x} \mu_{x_m \rightarrow f_s}(x_m) \end{aligned}$$

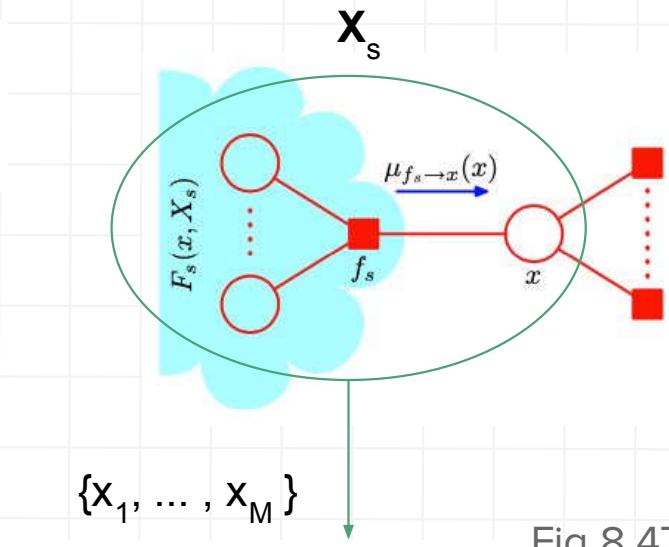
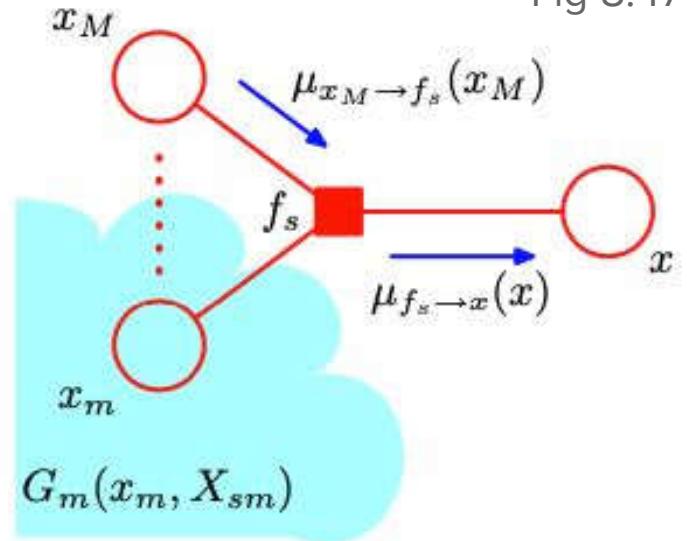


Fig 8.47



*Kschischang & Frey, 2001.*

The message sent from a node  $v$  on an edge  $e$  is the product of the local function at  $v$  (or the unit function if  $v$  is a variable node) with all messages received at  $v$  on edges *other* than  $e$ , summarized for the variable associated with  $e$ .

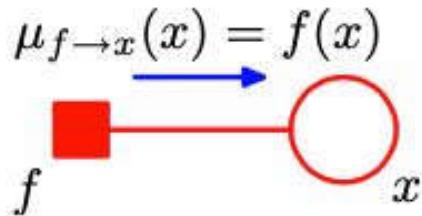
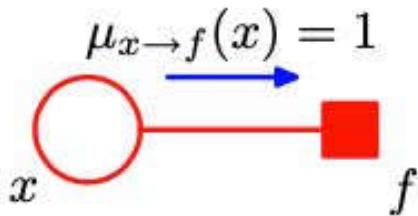
$$\mu_{x_m \rightarrow f_s}(x_m) = \prod_{l \in \text{ne}(x_m) \setminus f_s} \underbrace{\mu_{f_l \rightarrow x_m}(x_m)}_{\text{orange}}$$
$$\mu_{f_s \rightarrow x}(x) = \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \dots, x_M) \prod_{m \in \text{ne}(f_s) \setminus x} \mu_{x_m \rightarrow f_s}(x_m)$$

the marginals are then

$$p(x_m) = \prod_{l \in \text{ne}(x_m)} \underbrace{\mu_{f_l \rightarrow x_m}(x_m)}_{\text{orange}}$$

# How to start, how to normalise?

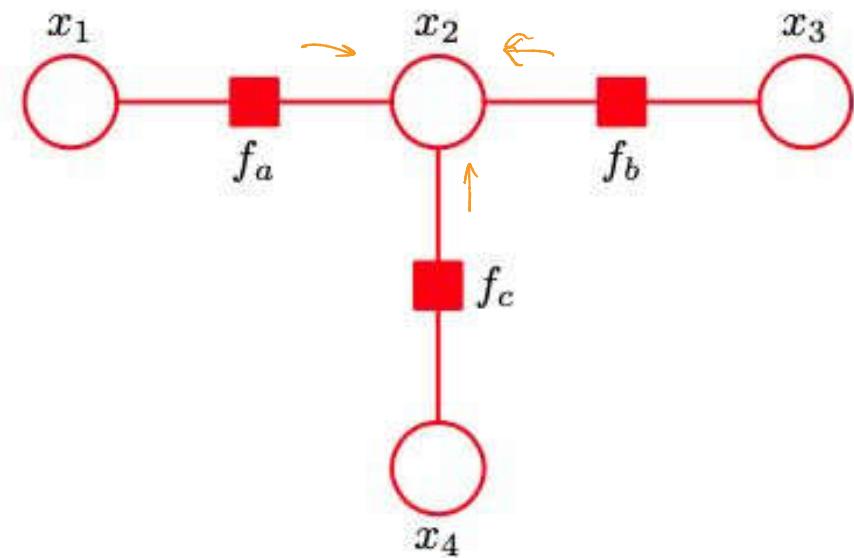
- Consider node  $x$  as the root node of the factor graph.
- Start at the leaf nodes.
  - If the leaf node is a **variable node** then
  - If the leaf node is a **factor node** then



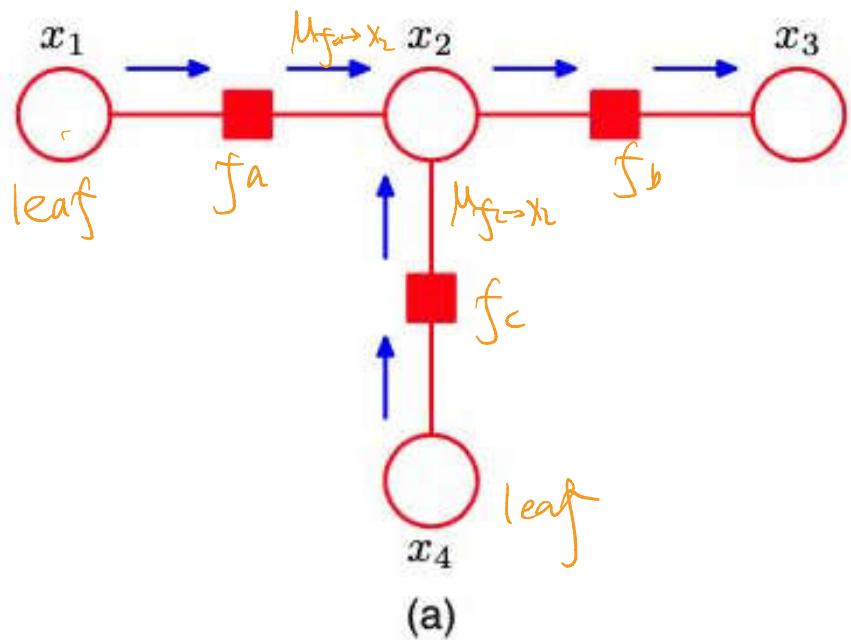
- These initialisation rules follow from the general formulae.
- Normalisation (if the factors are un-normalised):
  - Calculate  $\tilde{p}(x)$  by message passing as before
  - Then  $Z = \sum_x \tilde{p}(x)$  and  $p = \frac{1}{Z} \tilde{p}$

## Example

**Figure 8.51** A simple factor graph used to illustrate the sum-product algorithm.



$$\tilde{p}(\mathbf{x}) = f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4). \quad (8.73)$$



$$\mu_{x_1 \rightarrow f_a}(x_1) = 1 \quad (8.74)$$

$$\mu_{f_a \rightarrow x_2}(x_2) = \sum_{x_1} f_a(x_1, x_2) \quad \text{Π } \underbrace{\mu_{x_1 \rightarrow f_a}}_{f_a}$$

$$\mu_{x_4 \rightarrow f_c}(x_4) = 1 \quad (8.76)$$

$$\mu_{f_c \rightarrow x_2}(x_2) = \sum_{x_4} f_c(x_2, x_4) \quad (8.77)$$

$$\mu_{x_2 \rightarrow f_b}(x_2) = \underbrace{\mu_{f_a \rightarrow x_2}(x_2)}_{f_a} \underbrace{\mu_{f_c \rightarrow x_2}(x_2)}_{f_c}$$

$$\mu_{f_b \rightarrow x_3}(x_3) = \underbrace{\sum_{x_2} f_b(x_2, x_3)}_{f_b} \underbrace{\mu_{x_2 \rightarrow f_b}}_{\text{f\# of } x_2}. \quad (8.79)$$

**Figure 8.52** Flow of messages for the sum-product algorithm applied to the example graph in Figure 8.51. (a) From the leaf nodes  $x_1$  and  $x_4$  towards the root node  $x_3$ . (b) From the root node towards the leaf nodes.

$$\mu_{x_3 \rightarrow f_b}(x_3) = 1$$

(8.80)

$$\mu_{f_b \rightarrow x_2}(x_2) = \sum_{x_3} f_b(x_2, x_3)$$

(8.81)

$$\mu_{x_2 \rightarrow f_a}(x_2) = \mu_{f_b \rightarrow x_2}(x_2) \mu_{f_c \rightarrow x_2}(x_2)$$

(8.82)

$$\mu_{f_a \rightarrow x_1}(x_1) = \sum_{x_2} f_a(x_1, x_2) \mu_{x_2 \rightarrow f_a}(x_2)$$

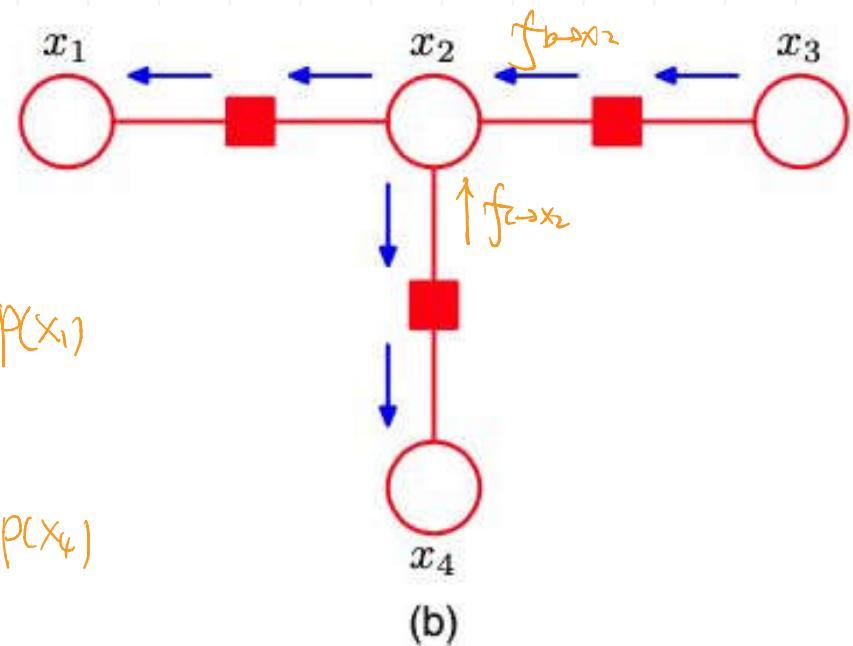
(8.83)  $p(x_1)$

$$\mu_{x_2 \rightarrow f_c}(x_2) = \mu_{f_a \rightarrow x_2}(x_2) \mu_{f_b \rightarrow x_2}(x_2)$$

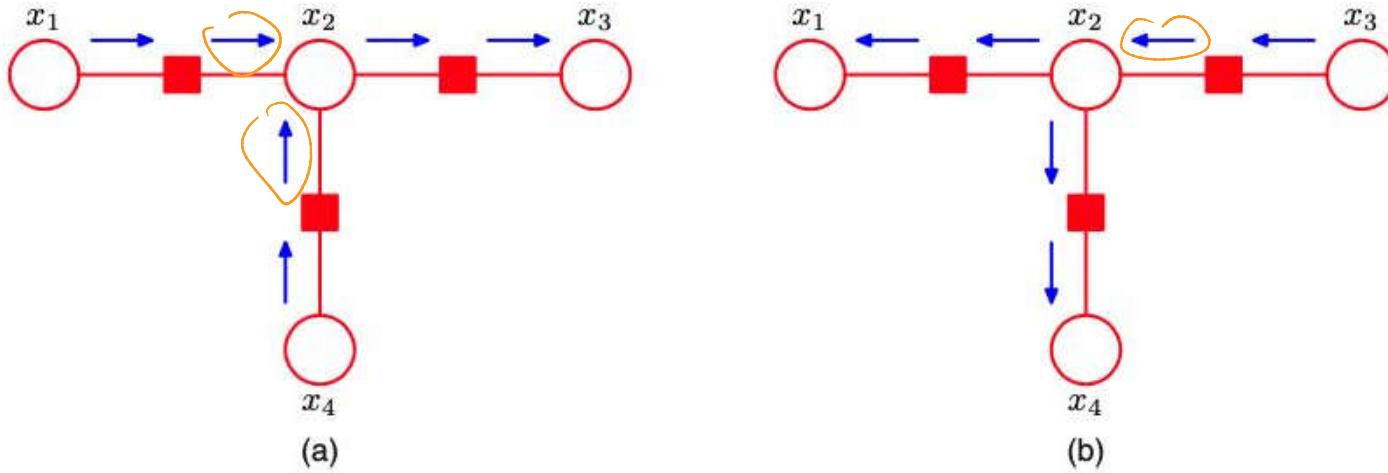
(8.84)

$$\mu_{f_c \rightarrow x_4}(x_4) = \sum_{x_2} f_c(x_2, x_4) \mu_{x_2 \rightarrow f_c}(x_2).$$

(8.85)  $p(x_4)$



**Figure 8.52** Flow of messages for the sum-product algorithm applied to the example graph in Figure 8.51. (a) From the leaf nodes  $x_1$  and  $x_4$  towards the root node  $x_3$ . (b) From the root node towards the leaf nodes.



**Figure 8.52** Flow of messages for the sum-product algorithm applied to the example graph in Figure 8.51. (a) From the leaf nodes  $x_1$  and  $x_4$  towards the root node  $x_3$ . (b) From the root node towards the leaf nodes.

$$\begin{aligned}
 \tilde{p}(x_2) &= \underbrace{\mu_{f_a \rightarrow x_2}(x_2)}_{\text{orange}} \underbrace{\mu_{f_b \rightarrow x_2}(x_2)}_{\text{orange}} \underbrace{\mu_{f_c \rightarrow x_2}(x_2)}_{\text{orange}} \\
 &= \left[ \sum_{x_1} f_a(x_1, x_2) \right] \left[ \sum_{x_3} f_b(x_2, x_3) \right] \left[ \sum_{x_4} f_c(x_2, x_4) \right] \\
 &= \sum_{x_1} \sum_{x_2} \sum_{x_4} f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4) \\
 &= \sum_{x_1} \sum_{x_3} \sum_{x_4} \tilde{p}(\mathbf{x})
 \end{aligned} \tag{8.86}$$

- Find some marginal  $p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x})$
- Idea: interchange summations and products:
  - Push sums as far into products as possible
  - View partial sums as messages
  - Partial sums are coupled by factors
  - Deal with coupling by propagating messages
  - Messages are functions of some variable
  - Exploit the tree structure of the factor graph to divide and conquer
- The tree structure is crucial.
- Generally incorrect for loopy graphs.
- Approach is also known as
  - Belief propagation
  - Message passing
  - Sum product algorithm

## *Marginals for ALL variable nodes in the graph*

- Brute force: run the above algorithm for each node.
- More efficient
  - ① Arbitrarily choose one root in the graph.
  - ② Propagate all messages from leaves to root.
  - ③ Propagate all messages from root to leaves.
  - ④ Local messages give marginals.
- Only doubles the number of computations.

## Generalising Sum-Product: Distributive Property

- For the sum product algorithm, we used the property that multiplication is **distributive** over addition:

$$ab + ac = a(b + c)$$

- Abstract theory

A set with two associative operations '+' and ' $\times$ ' satisfying the **distributive property** is called a **semi-ring**.

- Generalized Distributive Law

In principle, one can replace 'sum' and 'product' in the previous algorithm with other operations.

$a > 0$

$$\max(ab, ac) = a \max(b, c)$$

Aji, S.M.; McEliece, R.J. (Mar 2000). "The generalized distributive law". *IEEE Transactions on Information Theory*. 46 (2): 325–343. doi:10.1109/18.825794.

## Generalising Sum-Product: Max-Product

- $ab + ac = a(b + c) \rightarrow \max(ab, ac) = a \max(b, c)$
- Replace '+' by max, and ' $\times$ ' by  $\sum$
- Max Product algorithm: compute  $\text{argmax}_x p(\mathbf{x})$
- For Hidden Markov Models, this is the Viterbi algorithm.

"decoding"  
+ speech reco.  
Sound  $\rightarrow$  word  $\rightarrow$  sentences  
+ language model  
+ "wireless" radio.  
What was the 0/1 message  
originally sent?

tree of cliques .

- **Junction Tree Algorithm:** exact inference in general undirected graphs. (For directed graphs, first convert to moral graph.) Computational cost is determined by the number of variables in the largest clique of the graph. Grows exponentially with this number for discrete variables.
- **Loopy Belief Propagation:** try sum-product on graphs which are not tree-structured. Graph has cycles and therefore information flows several times through the graph. Initialise by assuming a unit message has been sent over each link in each direction. Convergence is not longer guaranteed in general.

Then "make cliques"  


# Graphical model inference

Factor graphs

The sum-product algorithm

Other algorithms

# Machine Learning for Realistic Cyber Deception

David Liebowitz  
May 2022



# Penten



Canberra-based cyber technology business

Australian Defence and Government focus

# Penten



## Business units

- Secure Mobility
- Tactical Communications Security
- Applied AI

# Secure Mobility

Remote access to classified networks

- AltoCrypt Stik
- AltoCrypt Phone



# Secure Mobility

Portable acoustic suppression

- AltoCrypt pBox

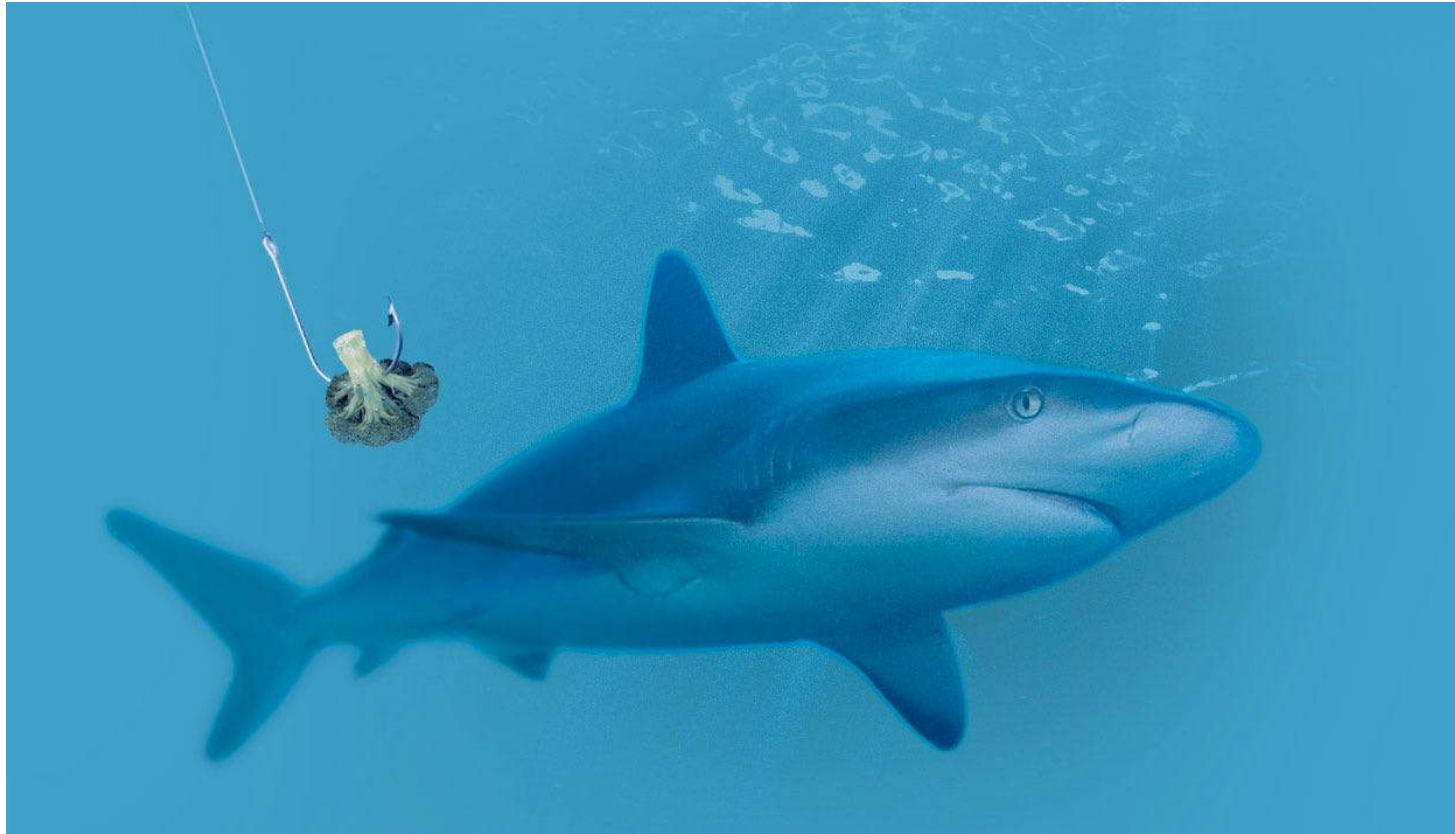


# Tactical Communications Security

Helix



# Applied AI



Cyber deception

# Deception Research



CYBER SECURITY  
COOPERATIVE  
RESEARCH  
CENTRE



UNSW  
SYDNEY

Cyber Deception Research Lab

*<http://www.cse.unsw.edu.au/~dliebowitz/>*

penten

# Deception

My working definition:

Deception is a (deliberate) attempt to manipulate the beliefs of others in order to influence their behaviour.

Deception is everywhere

# Deception in Nature



See: Martin Stevens, *Cheats and Deceits: How Animals and Plants Exploit and Mislead*, Oxford University Press, 2016

# Deception in Nature

Bolas spider



Image: Judy Gallagher

Flickr: <https://www.flickr.com/photos/52450054@N04/24843011518/>

# Deception in Nature

Pitcher plants  
and *batch capture*



[https://en.wikipedia.org/wiki/Nepenthes\\_rafflesiana](https://en.wikipedia.org/wiki/Nepenthes_rafflesiana)

# Evolved to Lie?

*We are thoroughgoing liars, even to ourselves. Our most prized possession—language— not only strengthens our ability to lie but greatly extends its range. We can lie about events distant in space and time, the details and meaning of the behavior of others, our innermost thoughts and desires, and so on.*

- Robert Trivers

# The Trickster



# Warfare

*All warfare is based on deception - Sun Tzu*



AUSTRALIAN WAR MEMORIAL

E04934

# Warfare

*In wartime, truth is so precious that she should always be attended by a bodyguard of lies.* - Winston Churchill

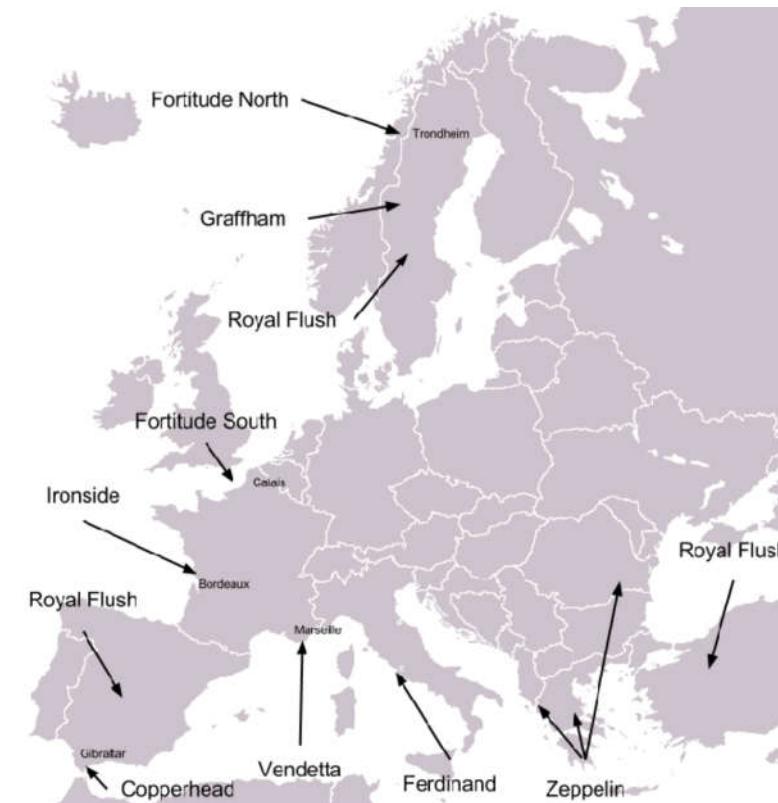


Image: ErrantX ([https://commons.wikimedia.org/wiki/File:Map\\_of\\_Operation\\_Bodyguard\\_subordinate\\_plans.png](https://commons.wikimedia.org/wiki/File:Map_of_Operation_Bodyguard_subordinate_plans.png)), "Map of Operation Bodyguard subordinate plans", <https://creativecommons.org/licenses/by-sa/3.0/legalcode>

# Sport



© Marie-Lan Nguyen / Wikimedia Commons / CC-BY 3.0  
([https://commons.wikimedia.org/wiki/File:Final\\_EMS-EQ\\_2013\\_Fencing\\_WCH\\_t211601.jpg](https://commons.wikimedia.org/wiki/File:Final_EMS-EQ_2013_Fencing_WCH_t211601.jpg)), „Final EMS-EQ 2013 Fencing WCH t211601“, <https://creativecommons.org/licenses/by/3.0/legalcode>

# Deception for Cyber Security

1. Where there is competition or conflict, deception emerges.
2. Deception is a valuable complement to existing security measures.

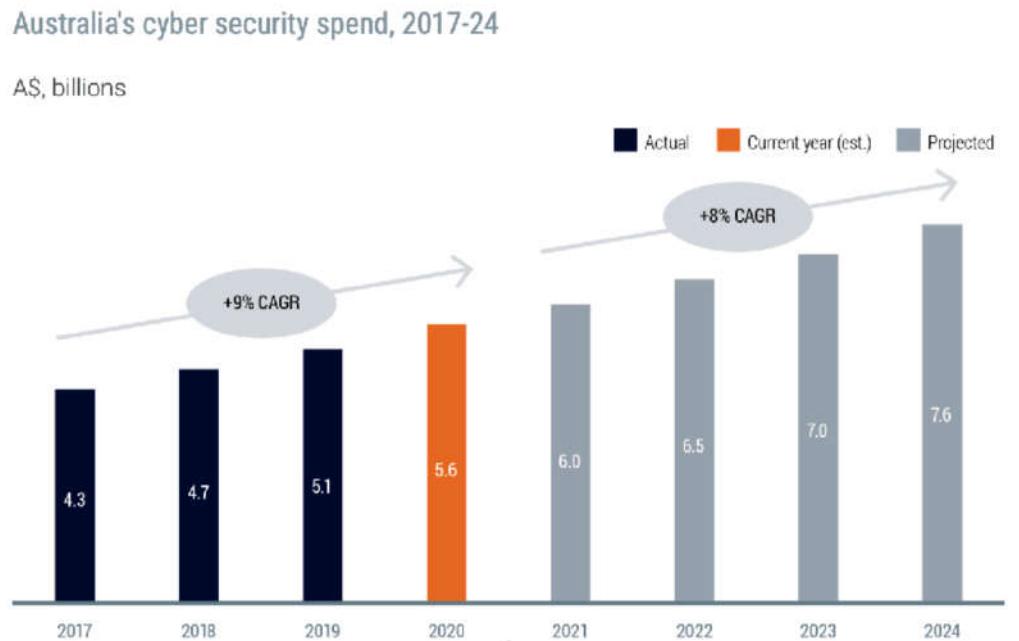
# Deception for Cyber Security

Australia spent

**AUD 5.6 billion**

on cyber security in **2020**

Source: *Australia's Cyber Security Sector Competitiveness Plan 2020*,  
Australian Cyber Security Growth Network



But breaches continue...

# Deception for Cyber Security

The average time to identify and contain a data breach is

**287 days.**

Source: *IBM/Ponemon Cost of a Data Breach Report 2021*

# Deception Can Help

# Honeypots

Cyber Deception usually means honeypots

*A honeypot is security resource whose value lies in being probed, attacked or compromised.*

- Lance Spitzner



# Honeypots

- Put fake stuff on the network
- Legitimate users have no reason to interact with it
- Any interaction is suspicious  
=> **breach discovery**
- Honeypots are selective interaction filters

# Types of Honeypots

- Can be mail/database/web servers
- Can also be **honeytokens** - honeypots that aren't computers:
  - Documents and files - honeyfiles
  - Database records - honeydata
  - Credentials - honeycredentials
  - ...
  - Honey + Anything =  Deception

# Tracers

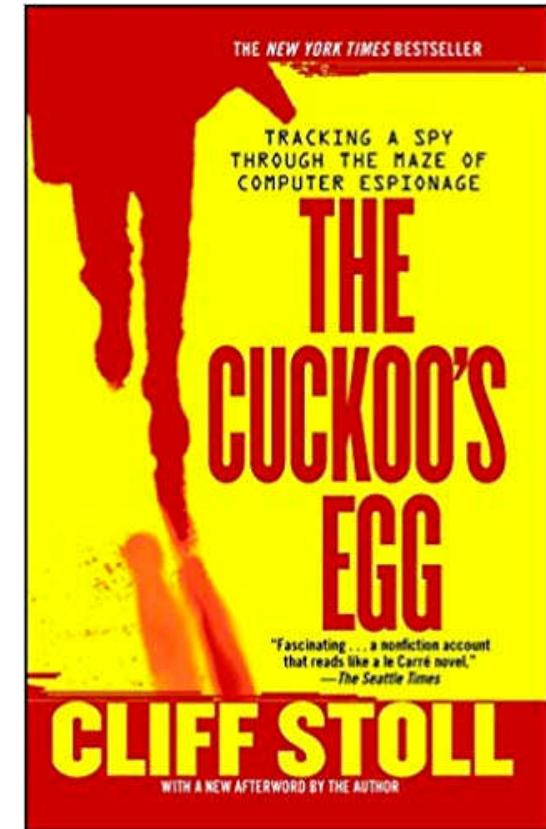
- Designed to be stolen:
  - Create unique tokens
  - Deploy them on the network
  - Track/monitor or beacon
  - Can discover and track data theft

# Other Benefits

- Adversary intent
- Tactics, Tools and Procedures (TTPs)
- Delay and frustrate
- Deceptions must be **realistic**

# The Cuckoos' Egg

- Clifford Stoll, Lawrence Berkeley National Laboratory in California
- Used deceptive documents
- Didn't shut the intruder out



# Now Add Machine Learning

- Realistic deceptions work better
- But tedious and expensive to create
- If only we could learn from real examples...
- ML/AI

# Some ML Tools

- Language models
- Graphs
- Temporal Point Processes

# Language Models

- Models probabilities of word sequences
- Train a model on some text and then generate
- Simple example: character model
  - Pick a *memory length*
  - Go through training data in overlapping windows and
  - For every *memory length* sequence, count instances of the next character

# Training

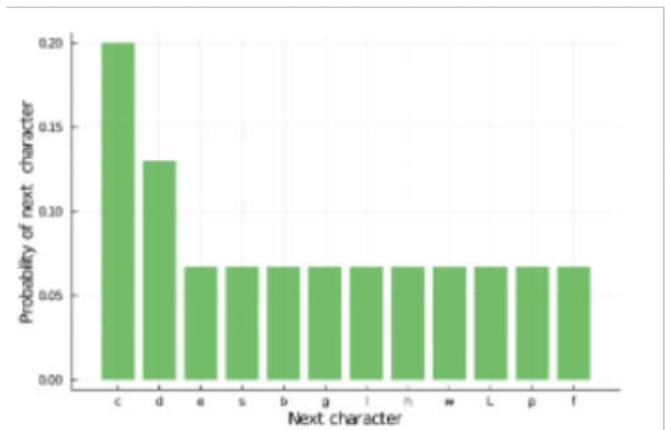
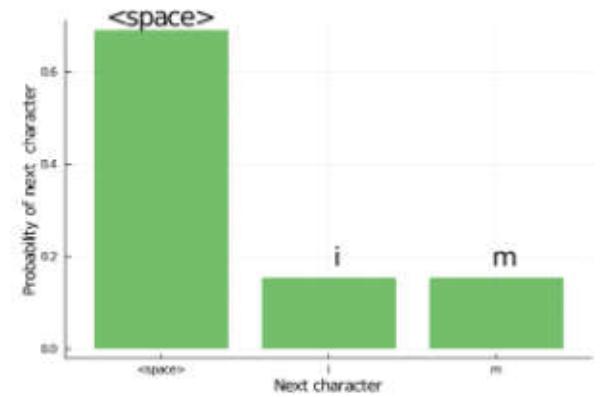
In a hole in the ground there lived a hobbit. Not a nasty, dirty, wet hole, filled with the ends of worms and an oozy smell, nor yet a dry, bare, sandy hole with nothing in it to sit down on or to eat: it was a hobbit-hole, and that means comfort.

# Character Based Model

"led with th"    'e': 1.0

"ed with the"    ' ': 0.69, 'i': 0.154, 'm': 0.154

"d with the ":"    'c': 0.2, 'd': 0.13, 'e': 0.067,  
's': 0.067, 'b': 0.067, 'g': 0.067,  
'l': 0.067, 'h': 0.067, 'w': 0.067,  
'L': 0.067, 'p': 0.067, 'f': 0.067



# Generation Sample

As a matter of fact two nights and the reason. It was lit by a great red fire in their midst and there with thirty or forty armed guards. Gandalf took their leave at last of Beorn, and they were in shining armour, and red light before last. Just think of a really hard one. This he thought, but before long the whole turn of affairs. He had by now had more than enough of the Mountain creaking and cracking his skull on the low arch of the path. “I am the clue-finder, the web-cutter, the stinging!” And he did. He darted backwards and forwards, slashing at spider-threads, hacking at the snapped painter that was made here as a guardroom. There were many stones lying in what appeared. Back swirled the dragon up against the magic it had once obeyed would ever open that door there was nothing to eat as quick as he could, he managed it somehow, he could not stay here, and departed with nothing in it to sit down on or to eat: it was a hobbit-hole.

# With Words

In a hole in the ground there lived a hobbit. Not a nasty, dirty, wet hole, filled with the ends of worms and an oozy smell, nor yet a dry, bare, sandy hole with nothing in it to sit down on or to eat: it was a hobbit-hole, and that means comfort.

# With Words

*"In a hole in the"*    'ground': 0.5, 'rock': 5

Need a lot more text to get variety in generation

# This May Look Familiar...

## The Bell System Technical Journal

*Vol. XXVII*

*July, 1948*

*No. 3*

---

### A Mathematical Theory of Communication

By C. E. SHANNON

#### INTRODUCTION

THE recent development of various methods of modulation such as PCM and PPM which exchange bandwidth for signal-to-noise ratio has intensified the interest in a general theory of communication. A basis for such a theory is contained in the important papers of Nyquist<sup>1</sup> and Hartley<sup>2</sup> on this subject. In the present paper we will extend the theory to include a number of new factors, in particular the effect of noise in the channel, and



# This May Look Familiar...

388

BELL SYSTEM TECHNICAL JOURNAL

### 3. THE SERIES OF APPROXIMATIONS TO ENGLISH

To give a visual idea of how this series of processes approaches a language, typical sequences in the approximations to English have been constructed and are given below. In all cases we have assumed a 27-symbol "alphabet," the 26 letters and a space.

1. Zero-order approximation (symbols independent and equi-probable).

XFOML RXKHRJFFJUJ ZLPWCFWKCYJ  
FFJEYVKCQSGXYD QPAAMKBZAACIBZLHJQD

2. First-order approximation (symbols independent but with frequencies of English text).

OCRO HLI RGWR NMIELWIS EU LL NBNESBYA TH EEI  
ALHENHTTPA OOBTTVA NAH BRL

3. Second-order approximation (digram structure as in English).

ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY  
ACHIN D ILONASIVE TUOOWE AT TEASONARE FUSO  
TIZIN ANDY TOBE SEACE CTISBE

4. Third-order approximation (trigram structure as in English).

IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID  
PONDENOME OF DEMONSTURES OF THE REPTAGIN IS  
REGOACTIONA OF CRE

5. First-Order Word Approximation. Rather than continue with tetra-gram, ...,  $n$ -gram structure it is easier and better to jump at this point to word units. Here words are chosen independently but with their appropriate frequencies.

REPRESENTING AND SPEEDILY IS AN GOOD APT OR  
COME CAN DIFFERENT NATURAL HERE HE THE A IN  
CAME THE TO OF TO EXPERT GRAY COME TO FUR-  
NISHES THE LINE MESSAGE HAD BE THESE.

6. Second-Order Word Approximation. The word transition probabilities are correct but no further structure is included.

1. Zero-order approximation (symbols independent and equi-probable).

XFOML RXKHRJFFJUJ ZLPWCFWKCYJ  
FFJEYVKCQSGXYD QPAAMKBZAACIBZLHJQD

2. First-order approximation (symbols independent but with frequencies of English text).

OCRA ULI RGTH NMIELWIS EU LL NBNESBYA TH EEI  
ALHENHTTPA OOBTTVA NAH BRL

3. Second-order approximation (digram structure as in English).

ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY  
ACHIN D ILONASIVE TUOOWE AT TEASONARE FUSO  
TIZIN ANDY TOBE SEACE CTISBE

4. Third-order approximation (trigram structure as in English).

IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID  
PONDENOME OF DEMONSTURES OF THE REPTAGIN IS  
REGOACTIONA OF CRE

5. First-Order Word Approximation. Rather than continue with tetra-gram, ...,  $n$ -gram structure it is easier and better to jump at this point to word units. Here words are chosen independently but with their appropriate frequencies.

REPRESENTING AND SPEEDILY IS AN GOOD APT OR  
COME CAN DIFFERENT NATURAL HERE HE THE A IN  
CAME THE TO OF TO EXPERT GRAY COME TO FUR-  
NISHES THE LINE MESSAGE HAD BE THESE.

6. Second-Order Word Approximation. The word transition probabilities are correct but no further structure is included.

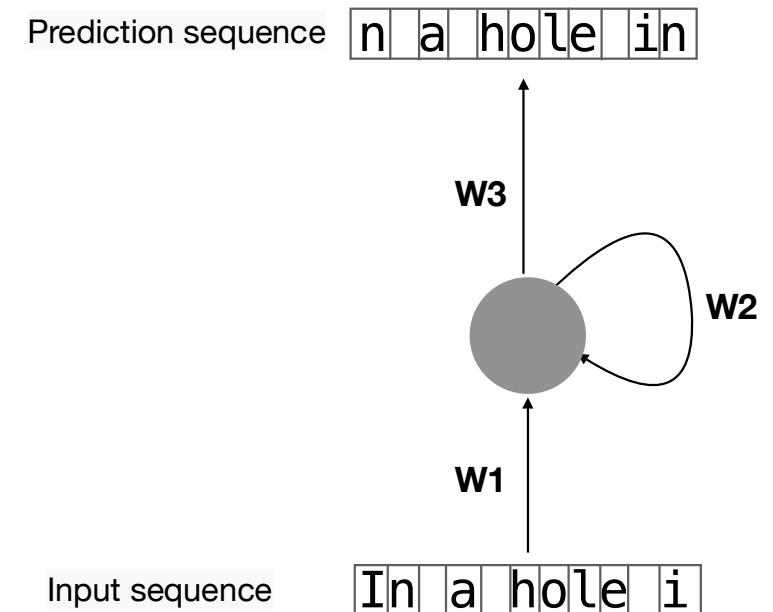
THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH  
WRITER THAT THE CHARACTER OF THIS POINT IS  
THEREFORE ANOTHER METHOD FOR THE LETTERS  
THAT THE TIME OF WHO EVER TOLD THE PROBLEM  
FOR AN UNEXPECTED

The resemblance to ordinary English text increases quite noticeably at each of the above steps. Note that these samples have reasonably good structure out to about twice the range that is taken into account in their construction. Thus in (3) the statistical process insures reasonable text for two-letter sequence, but four-letter sequences from the sample can usually be fitted into good sentences. In (6) sequences of four or more



# Recurrent Neural Networks

- Deep Learning language models
  - RNNs have long memory
- 
- See: Andrey Karpathy, *The Unreasonable Effectiveness of Recurrent Neural Networks*, <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>



# Language is Complicated

During the 1998 Test series against England, Cronje scored five consecutive fifties, having failed to score one in the nine previous Tests against them. In his fiftieth Test, at Trent Bridge he scored 126, his sixth and last Test century and his first in 29 matches.

Extracted from the Hansie Cronje Wikipedia page

# Language is Complicated

During the 1998 Test series against England, Cronje scored five consecutive fifties, having failed to score one in the nine previous Tests against them. In his fiftieth Test, at Trent Bridge he scored 126, his sixth and last Test century and his first in 29 matches.

104 characters and 16 words between England and them.

# Language is Complicated

During the 1998 Test series against England, Cronje scored five consecutive fifties, having failed to score one in the nine previous Tests against them. In his fiftieth Test, at Trent Bridge he scored 126, his sixth and last Test century and his first in 29 matches.

105 characters/17 words between

139 characters/23 words

155 characters/26 words

191 characters/33 words

Cronje and	his
Cronje and	he
Cronje and	his
Cronje and	his

# Transformers

- Uses **attention**
- Can be trained in parallel
- GPT-2, Open AI (June 2018), 1.5 B parameters
- Turing NLG, Microsoft (Feb 2020) ,17 B parameters
- GPT-3, Open AI, (May 2020), 175 B parameters  
and growing...

Can generate coherent paragraphs with a prompt

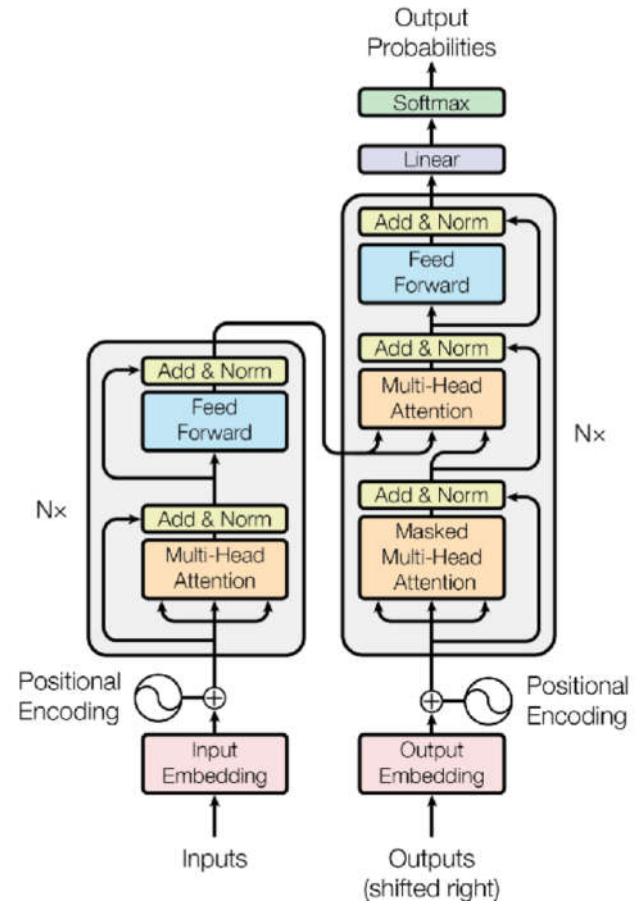
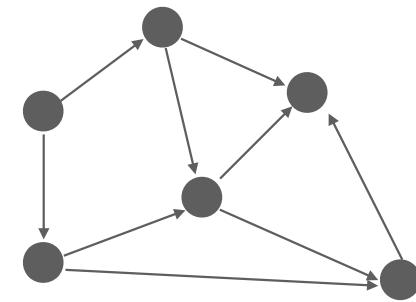


Image from: Vaswani et al, *Attention is all you need*. In Advances in Neural Information Processing Systems, 2017

# Graphs

- Many systems have graph representations
  - Document structure
  - File system
  - IT networks
  - Communications/social networks
  - ...

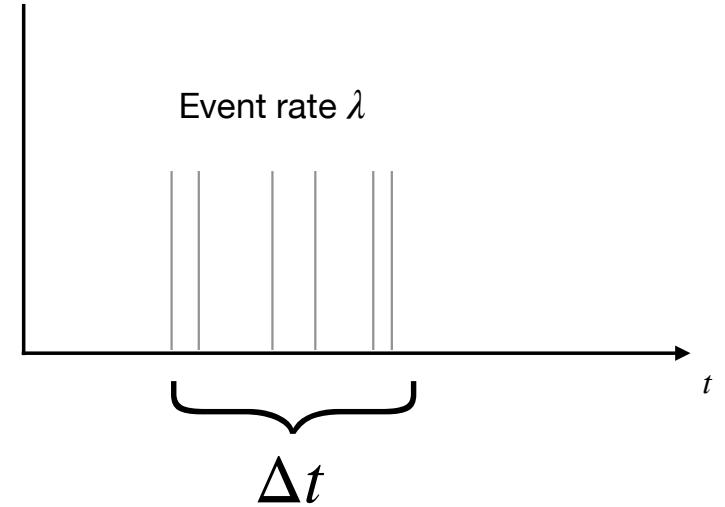


# Graphs

- Deep Learning on graphs
  - Graph Recurrent Attention Network (GRAN) Liao *et al*, NeurIPS, 2019

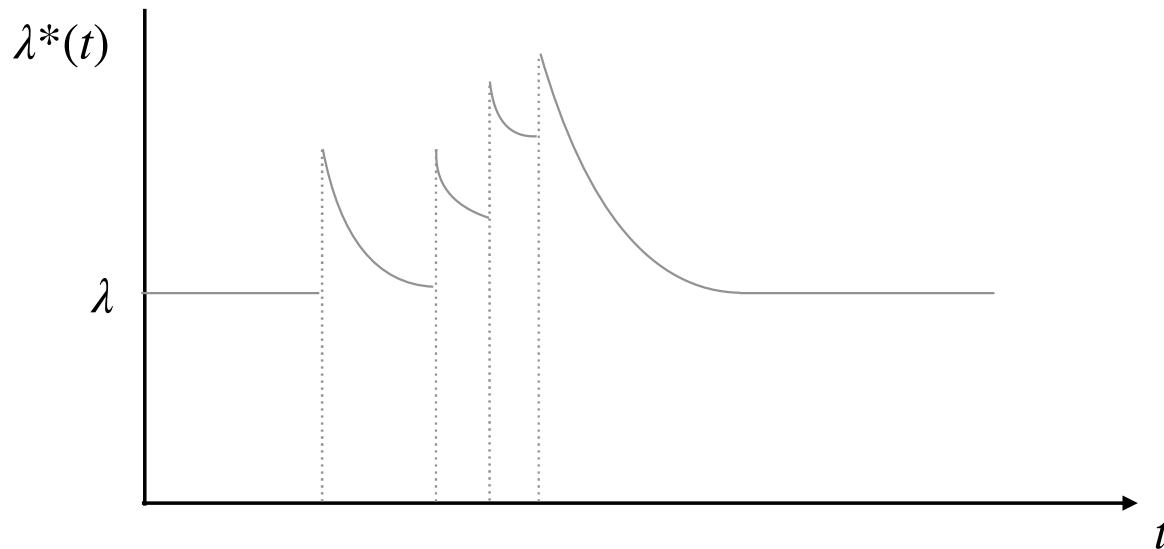
# Communications with TPPs

- When do events take place?
- Temporal Point Processes
  - Poisson process with rate  $\lambda$ :
  - $p(n \text{ events in } \Delta t) = \frac{(\lambda \Delta t)^n}{n!} \exp^{-\lambda \Delta t}$
  - Memoryless
  - Events independent



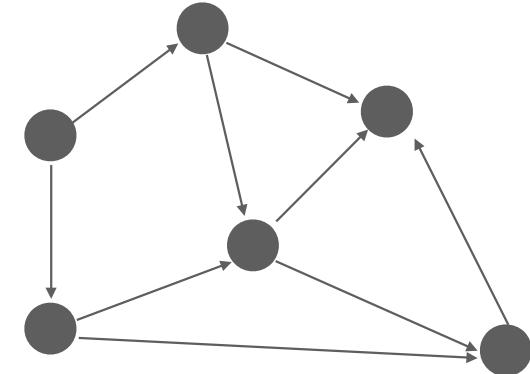
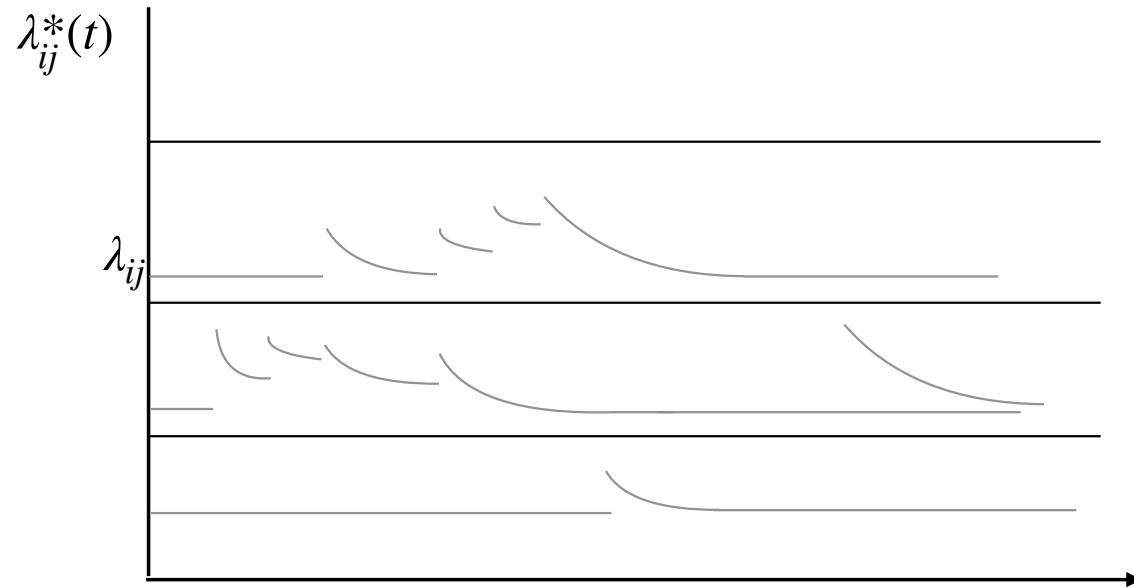
# Communications with TPPs

- Hawkes process:  $\lambda^*(t) = \lambda + \sum_{t_i < t} \mu(t - t_i)$   
with  $\mu(t) = \alpha e^{-\beta t}$
- Base rate  $\lambda$  and *self-exciting* events



# Communications with TPPs

- *Marked* TPPs model temporal interactions on network
- $\alpha_{ij}$  and  $\beta_{ij}$  for pairs of communicating nodes



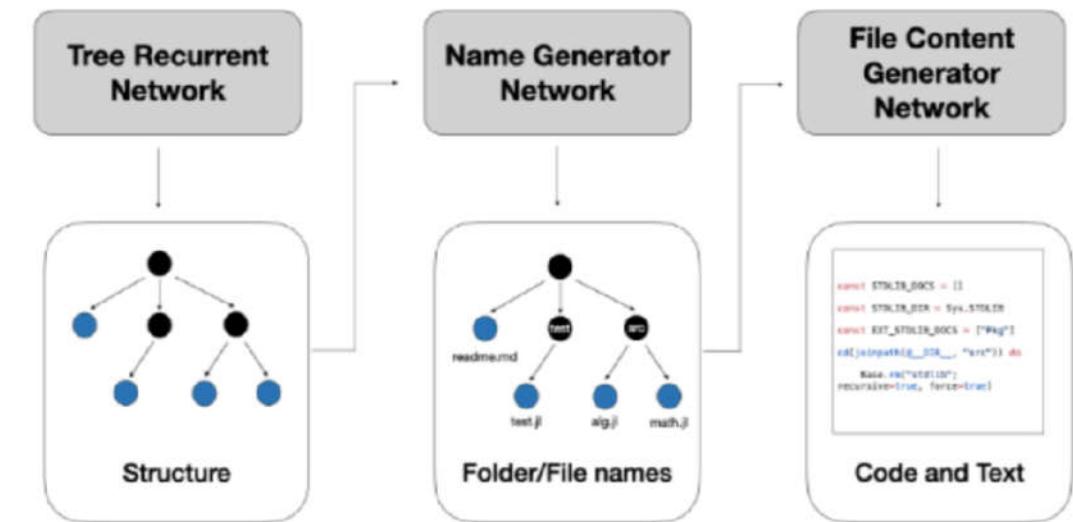
# Communications with TPPs

- Neural Point Processes
- RNN to embed event sequence
- Intensity Free TPP:
  - Embedding vector and metadata parametrise inter-event time distribution
  - Mixture of lognormal

See: Intensity Free TPP from O. Shchur, M. Biloš, and S. Günnemann, *Intensity-free learning of temporal point processes*, ICLR, 2019.

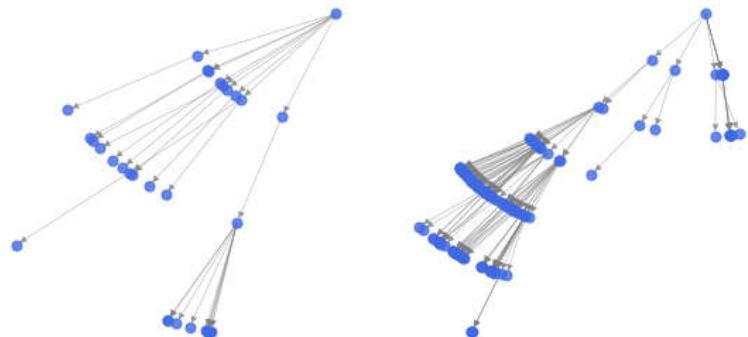
# HoneyCode: Fake Repositories

- Fake file trees, file names and code
- Trees from modified GRAN
- Filenames and code from character RNN



D. Nguyen, D. Liebowitz, S. Nepal, and S. Kanhere, *Honeycode: Automating deceptive software repositories with deep generative models*, in Proceedings of the 54th Hawaii International Conference on SystemSciences, 2021

# HoneyCode: Fake Repositories



```
function sparse!(d::AbstractMixtureModel, x)
    K = ncomponentwise_logpdf!(Matrix{eltype(x)}(undef, nd, n), one(x))
    logc0 = /x * detach(1)
    logc0 = max(length(I0)*length(J)+1

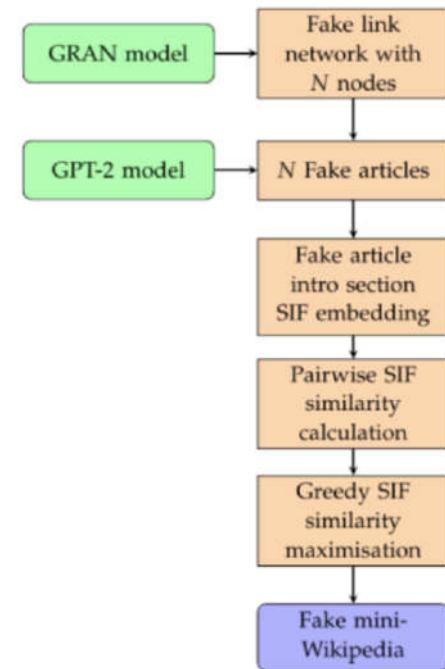
    for i = 1:(length(Acolptr) <= 1)
        VR = similar(A[1], T, nrows, ncols)
    elseif size(X, 2) == 1
        logX = log(u)
        cm2 /= sqrt(1 - abs2(z)/2) * sqrt(z2)
        alpha3 = max(unsqueezeb, Tuple{Float64})
        $fname(pp, pos, durbin, sort)
        return (false, Int[])
    end
    else
        if !in_single_quotes
            const_prop_profitable(buf) == 0 && return false
            write(buffer, ' ')
        end
    end
    return nothing
    return write_project(env)
end
```

D. Nguyen, D. Liebowitz, S. Nepal, and S. Kanhere, *Honeycode: Automating deceptive software repositories with deep generative models*, in Proceedings of the 54th Hawaii International Conference on SystemSciences, 2021

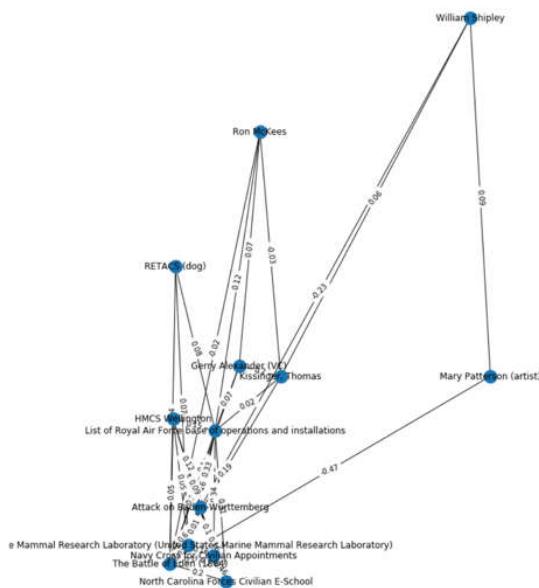


# WikiGen

- Fake page content and structure from GPT-2
- Network from GRAN
- SIF embeddings to match linked content



# WikiGen



== Article Start ==

Albert Le Roux

Albert Le Roux (August 14, 1879 – July 16, 1961) was a Canadian historian and political activist. He served in the Canadian Infantry during the First World War.

== Early life and education ==

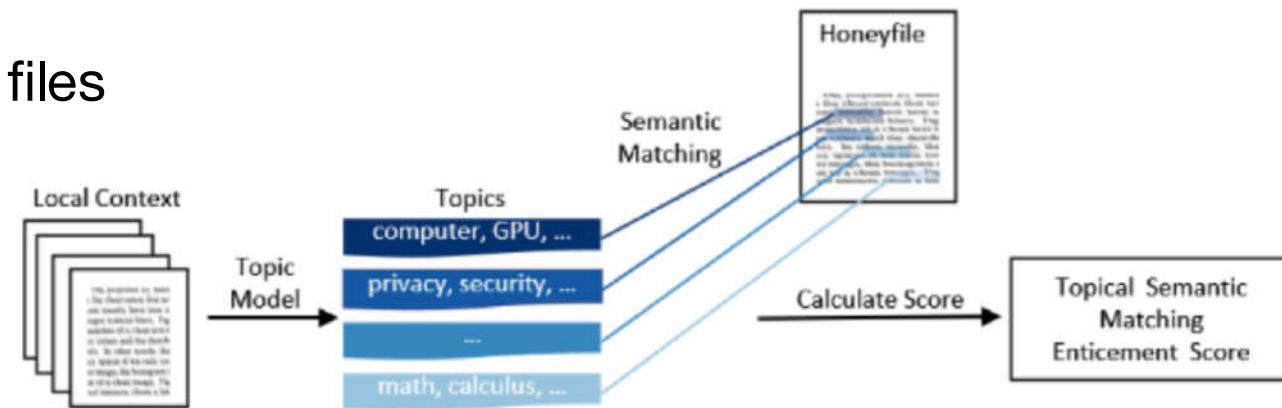
Le Roux was born in 1879 in Ridgeway, Ontario, the son of Alphonse Le Roux. As a young man, Le Roux attended Mount Allison School in Toronto until his father left the family when Le Roux was about twelve. He was subsequently educated at Osgoode Hall and Mount Allison College.

== Military career ==

Le Roux entered the Canadian Infantry in the Second Canadian Expeditionary Force in 1916. Le Roux was commissioned in the men's 1st Battalion in January 1917, and was seconded to the 2nd Battalion. Le Roux was promoted to lieutenant in November 1917 and captain in November 1918. In 1917, Le Roux was in command of a battalion in the Italian Campaign.

# Deception Metrics

- Enticement: drawing adversary to the honeypot
- Compare honeyfile text to real files

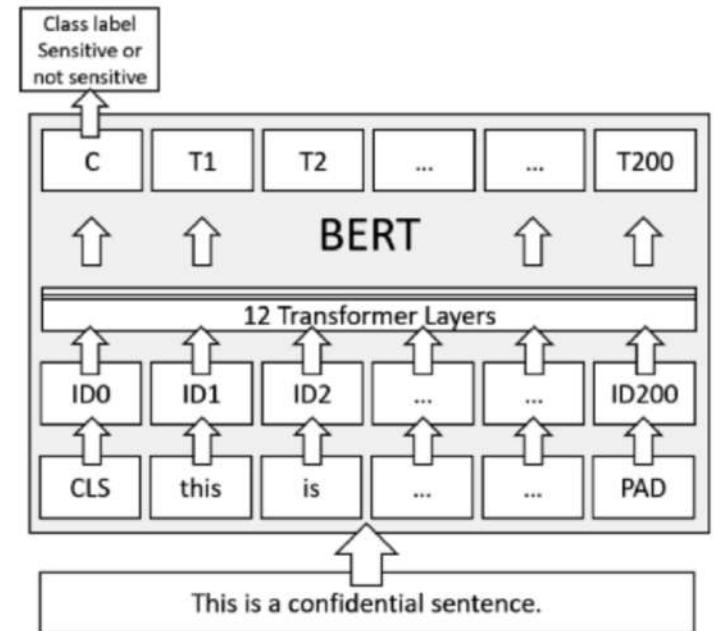


- Topic modelling, semantic matching

R. Timmer, D. Liebowitz, S. Nepal, and S. Kanhere, *TSM: Measuring the Enticement of Honeyfiles with Natural Language Processing*, Proceedings of the 55th Hawaii International Conference on System Sciences, 2022

# Sensitive Content Detection

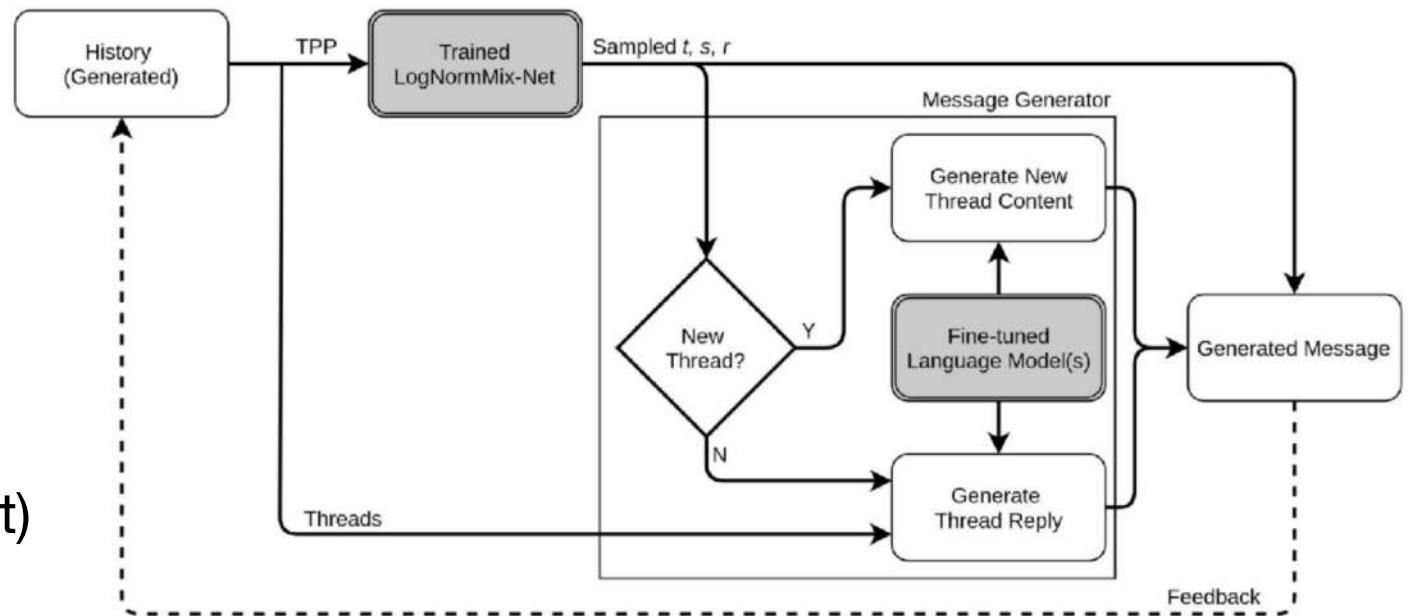
- Automated sensitive content detection
- Monsanto trial data labeled at sentence level
- BERT can do pretty good detection with small training sets



R. Timmer, D. Liebowitz, S. Nepal, and S. Kanhere, *Can pre-trained transformers be used in detecting complex sensitive sentences? - a Monsanto case study*, Proc. the 3rd IEEE International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications, 2021.

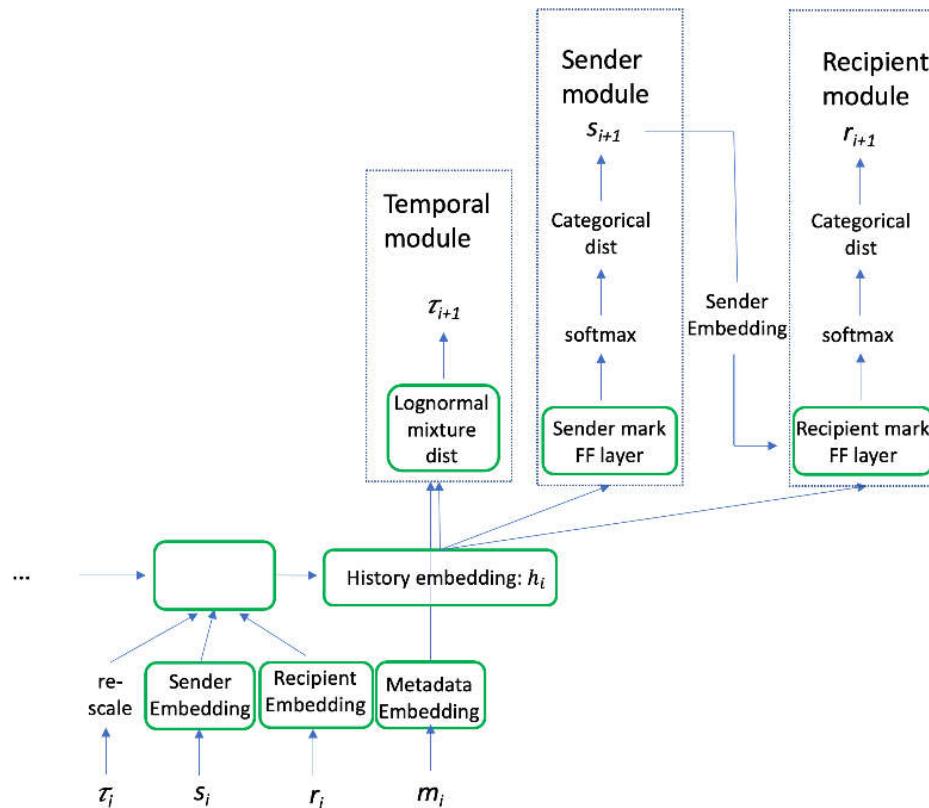
# Networked Communications

- Marked TPP
- Language model
- Simulate Direct Messaging  
(email, social networks chat)



K. Moore, C. J. Christopher, D. Liebowitz, S. Nepal, and R. Selvey,  
*Modelling direct messaging networks with multiple recipients for cyber  
deception*, To appear in 7th IEEE European Symposium on Security and Privacy, June 2022

# Networked Communications



K. Moore, C. J. Christopher, D. Liebowitz, S. Nepal, and R. Selvey,  
*Modelling direct messaging networks with multiple recipients for cyber  
deception*, To appear in 7th IEEE European Symposium on Security and Privacy, June 2022

# Networked Communications

## New Wilson Web Address.

Lauren.Gould@wilfred.com.au

Tues 6/10/2001 4:30PM

To: Tanya.Ali@wilfred.com.au; Paul.Flores@wilfred.com.au; Dylan.Odonnell@wilfred.com.au;  
Nicholas.Wells@wilfred.com.au

Hello Travis,

As you are aware, we are moving into a new system that will make it easier for you to send out customized email messages without using the web address. If you are not currently using either of these extensions or any other means, you will need to either upgrade your e-mail address to the new system, or change your password for your account to be synchronized with the new system.

Best, Lauren.

From: Travis.Davis@wilfred.com.au

Tues 6/10/2001 12:30PM

To: Jill.Perez@wilfred.com.au; Nicholas.Wells@wilfred.com.au; Lauren.Gould@wilfred.com.au;  
Toni.Nicholson@wilfred.com.au; Christina.Eaton@wilfred.com.au

Hi all

We have been working on building our internal email database for the last 5 years. In order to be included as a member in this new system, we will need to fill in this required information from each person, as well as the name of their company, the type of computer they have, and their internet connection.

Travis.

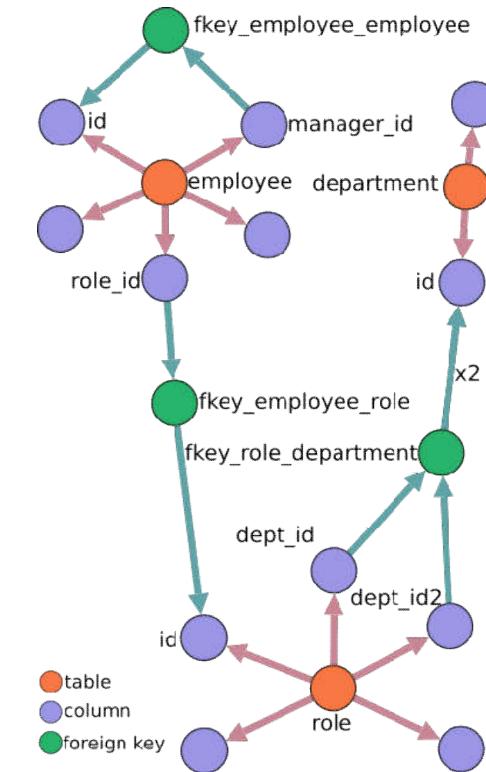
K. Moore, C. J. Christopher, D. Liebowitz, S. Nepal, and R. Selvey,  
*Modelling direct messaging networks with multiple recipients for cyber deception*, To appear in 7th IEEE European Symposium on Security and Privacy, June 2022



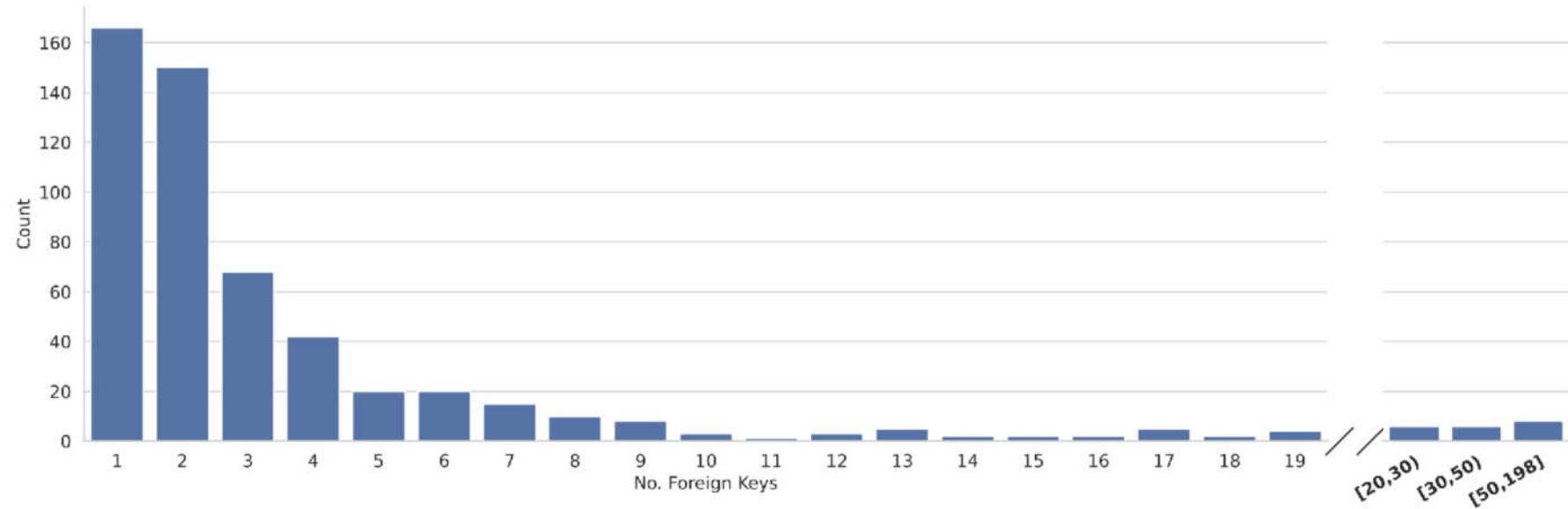
# Database Generation

Towards novel database generation:

- Automatically scrape and parse schema creation SQL scripts on Github
- Standardised format
- Implied foreign keys extracted
- ~ 2500 schemas and growing



# Database Generation



# HoneyTrace



- Data Loss Intelligence
- HoneyTrace.io

# HoneyTrace



---

Track information  
most important to  
your business



---

Extend detection  
beyond your  
network perimeter



---

Helps to answer  
“who, what and  
when?”



---

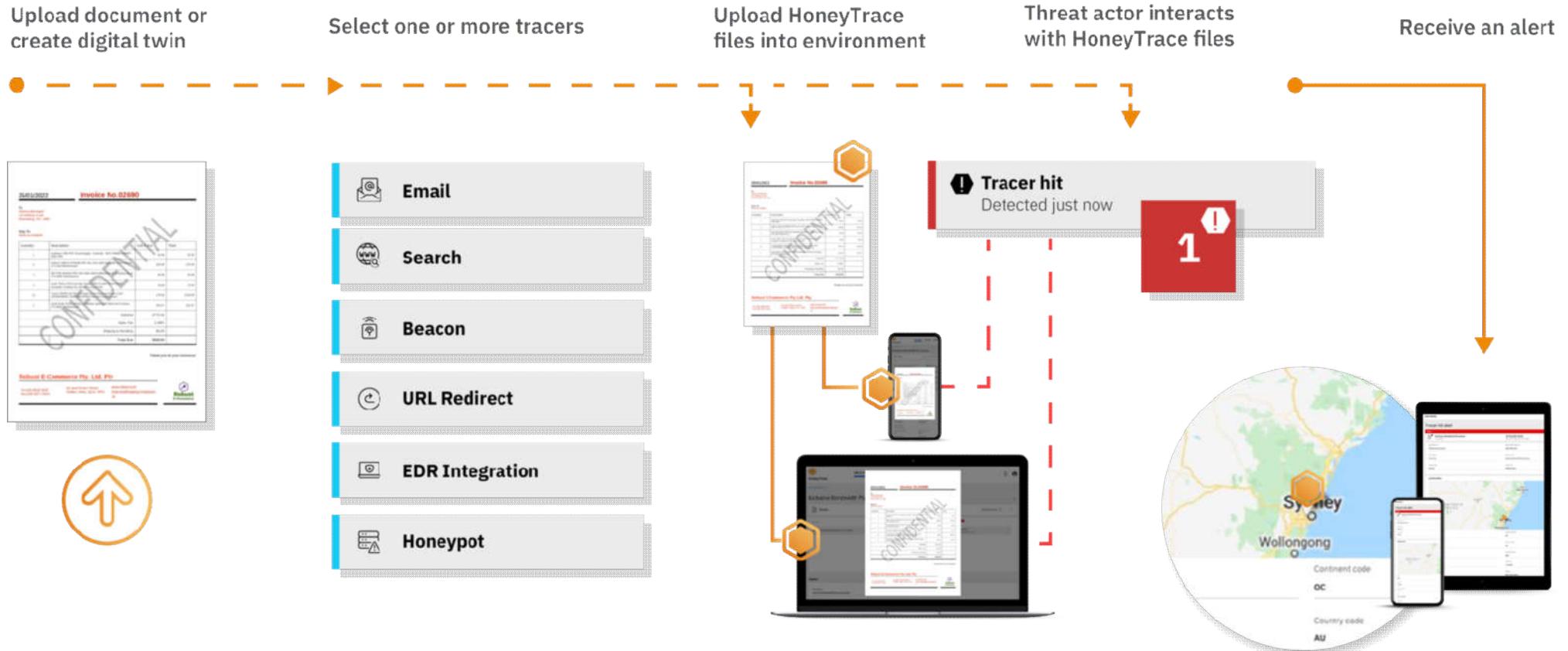
Set up in  
minutes



---

No installation of  
software on your  
system or network

# HoneyTrace



penten

# Questions?

## HOW TO BECOME THE MOST HATED BAND IN THE WORLD:

RECORD AN ALBUM THAT'S NOTHING BUT BRILLIANT, CATCHY INSTANT CLASSICS GUARANTEED POPULARITY AND AIRTIME,



WITH A SAMPLE OF A CAR HORN, CELL PHONE, OR ALARM CLOCK INSERTED RANDOMLY IN EACH SONG.

<https://xkcd.com/780/>

Sampling can be hard to get right ...

Class update:

Exam info posted. Friday “week 13”.  $\rightarrow$  June ,

Week 12 Monday - no lecture, finish your video assignment + good luck with all end-of-S1 deadlines!

Week 12 Wednesday - Q&A, Talk about selected questions in 2021 Exam.

“Week 13” – drop-in sessions will be announced.

# Sampling methods

Motivation, why? Sampling and ML

Basic sampling algorithms

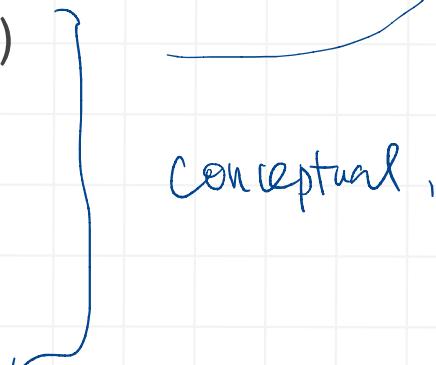
- Sampling standard distributions from  $U(0, 1)$
- Rejection sampling
- Importance sampling

Markov chain Monte Carlo (MCMC)

- Markov chains
- Metropolis-Hastings

Gibbs sampling    11.3

Bishop Chap 11  
Intro, 11.1.1, 11.1.2, 11.1.4, 11.1.6  
11.2, 11.2.1, 11.2.2, 11.2.3  
11.3



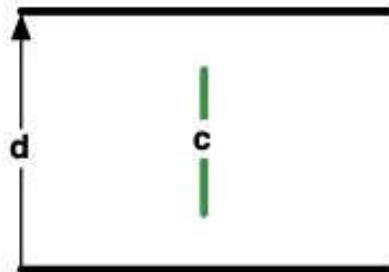
conceptual ,

# Estimating $\pi$ – Buffon's needle

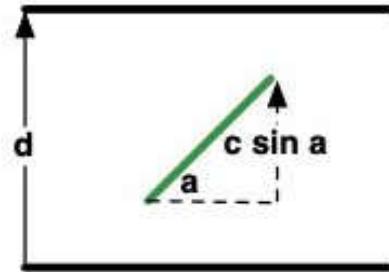
- Drop length  $c$  needle on parallel lines distance  $d$  apart
- Needle falls perpendicular:  
Probability of crossing the line is  $c/d$ .
- Needle falls at an arbitrary angle  $a$ :  
Probability of crossing the line  $c \sin(a)/d$ .
- Every angle is equally probable. Calculate the mean:

$$p(\text{crossing}) = \frac{c}{d} \int_0^\pi \sin(a) dp(a) = \frac{1}{\pi} \frac{c}{d} \int_0^\pi \sin(a) da = \frac{2}{\pi} \frac{c}{d}$$

(iv)  $n$  crossings in  $N$  experiments results in  $\frac{n}{N} \approx \frac{2}{\pi} \frac{c}{d}$



(i) Needle falls  
perpendicular ( $a = \pi/2$ ).



(ii) Needle falls at  
arbitrary angle  $a$ .

- $N \rightarrow \infty$  estimate "correct"
- how fast?

<https://mathworld.wolfram.com/BuffonsNeedleProblem.html>

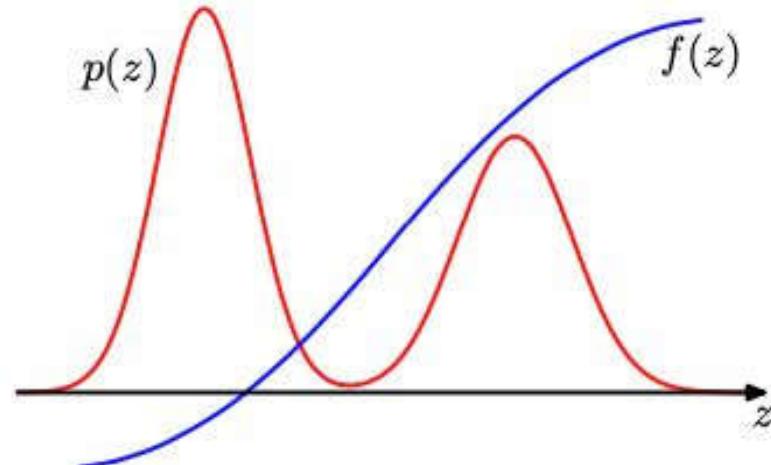
# Why do we need sampling?

Distributions can be quite complex, e.g. posterior distributions, mixture distributions, graphical models (coming next).

Many ML tasks can be stated as estimating the expectation of functions under a distribution, e.g. posterior mean and variance, moments.

$$\mathbb{E}[f] = \int f(\mathbf{z}) p(\mathbf{z}) d\mathbf{z} \quad (11.1)$$

**Figure 11.1** Schematic illustration of a function  $f(z)$  whose expectation is to be evaluated with respect to a distribution  $p(z)$ .



# Bayesian logistic regression

(GP2)

$$p(\mathcal{C}_1|\phi) = y(\phi) = \sigma(\mathbf{w}^T \phi) \quad (4.87)$$

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^N \{t_n \ln y_n + (1-t_n) \ln(1-y_n)\} \quad (4.90)$$

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0) \quad (4.140) \quad \rightarrow \text{prior}$$

$$p(\mathbf{w}|\mathbf{t}) \propto p(\mathbf{w})p(\mathbf{t}|\mathbf{w}) \quad (4.141)$$

$$\begin{aligned} \ln p(\mathbf{w}|\mathbf{t}) &= -\frac{1}{2}(\mathbf{w} - \mathbf{m}_0)^T \mathbf{S}_0^{-1} (\mathbf{w} - \mathbf{m}_0) \\ &+ \sum_{n=1}^N \{t_n \ln y_n + (1-t_n) \ln(1-y_n)\} + \text{const} \end{aligned} \quad (4.142)$$

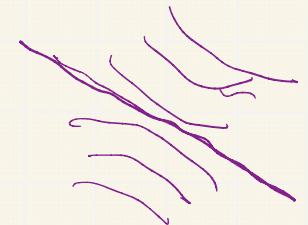
$$\mathbf{S}_N = -\nabla \nabla \ln p(\mathbf{w}|\mathbf{t}) = \mathbf{S}_0^{-1} + \sum_{n=1}^N y_n(1-y_n) \phi_n \phi_n^T. \quad (4.143)$$

Laplace approximation  $\underline{q(\mathbf{w}) = \mathcal{N}(\mathbf{w}_{\text{MAP}}, \mathbf{S}_N)}.$  (4.144)

$$\underline{p(\mathcal{C}_1|\phi, \mathbf{t})} = \int p(\mathcal{C}_1|\phi, \mathbf{w}) p(\mathbf{w}|\mathbf{t}) d\mathbf{w} \simeq \int \sigma(\mathbf{w}^T \phi) q(\mathbf{w}) dw \quad (4.145)$$

$$\longrightarrow = \int \underbrace{\sigma(a)}_{\text{sample}} \mathcal{N}(a|\mu_a, \sigma_a^2) da.$$

$$\mathbb{E}[f] = \int f(\mathbf{z}) p(\mathbf{z}) dz \quad (11.1)$$



# Sampling 101

Goal: estimate

$$\mathbb{E}[f] = \int f(\mathbf{z}) p(\mathbf{z}) d\mathbf{z} \quad (11.1)$$

Simply

$$\hat{f} = \frac{1}{L} \sum_{l=1}^L f(\mathbf{z}^{(l)}). \quad (11.2)$$

sample from  $p(\mathbf{z})$  i.i.d.

$$\mathbb{E}[\hat{f}] = \mathbb{E}[f]$$

$$\underline{\text{var}}[\hat{f}] = \frac{1}{L} \mathbb{E} [(f - \mathbb{E}[f])^2] \quad (11.3)$$

The good:

- the accuracy of the estimator does not depend on the dimensionality of  $\mathbf{z}$
- in principle, high accuracy may be achievable with a relatively small number of samples

"effective # of samples"

The bad:

- samples  $\{\mathbf{z}^{(l)}\}$  might not be independent, and so the effective sample size might be much smaller than the apparent sample size.

The good:

- the accuracy of the estimator does not depend on the dimensionality of  $z$
- in principle, high accuracy may be achievable with a relatively small number of samples

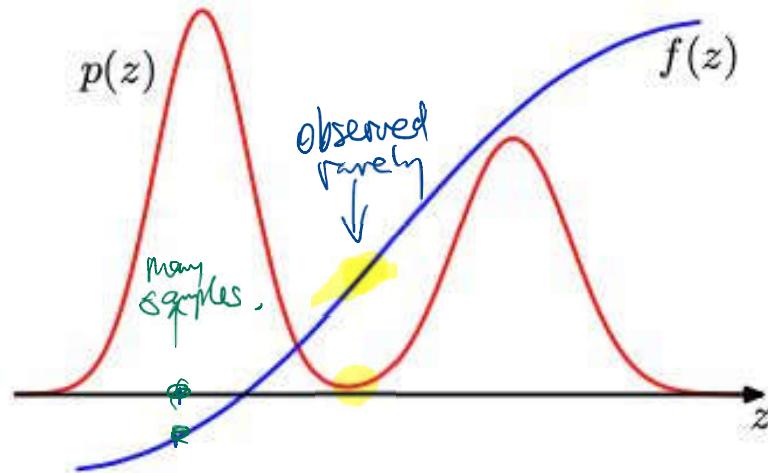
The bad:

- samples  $\{z^{(i)}\}$  might not be independent, and so the effective sample size might be much smaller than the apparent sample size.

The ugly:

$f(z)$  is small in regions where  $p(z)$  is large, and vice versa, then the expectation may be dominated by regions of small probability, implying that **relatively large sample sizes will be required to achieve sufficient accuracy**.

**Figure 11.1** Schematic illustration of a function  $f(z)$  whose expectation is to be evaluated with respect to a distribution  $p(z)$ .



# Sampling from uniform distributions

- In a computer usually via pseudorandom number generator : an algorithm generating a sequence of numbers that approximates the properties of random numbers.
- Use a mathematically well crafted pseudorandom number generator.
- From now on we will assume that we have a good pseudorandom number generator for uniformly distributed data available.
- If you don't trust any algorithm :  
Three carefully adjusted radio receivers picking up atmospheric noise to provide real random numbers at  
<http://www.random.org/>

For this class: assume we can generate  $z \sim U(0, 1)$

Q: given  $z \sim U(0, 1)$   
How do you obtain  $z \sim U(a, b)$  ?

# Sampling a distribution $p(y)$

$$p(y) = p(z) \left| \frac{dz}{dy} \right| \quad (11.5)$$

$$z = h(y) \equiv \int_{-\infty}^y p(\hat{y}) d\hat{y} \quad (11.6)$$

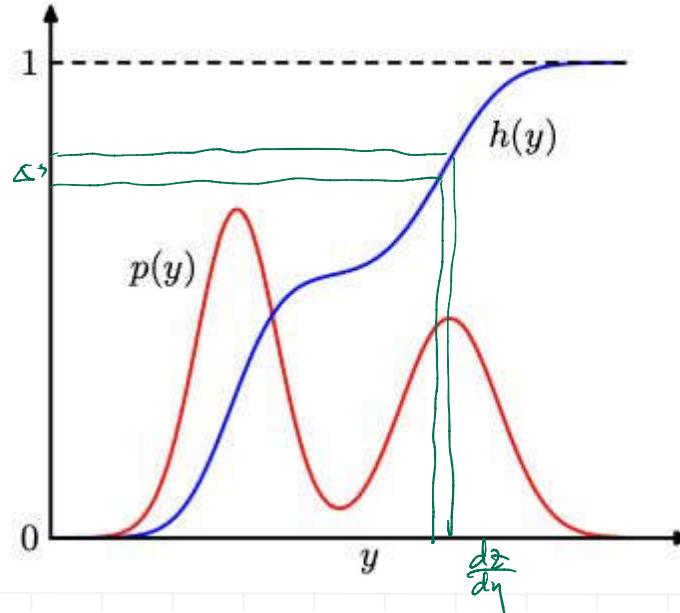
Multiple variables

$$p(y_1, \dots, y_M) = p(z_1, \dots, z_M) \left| \frac{\partial(z_1, \dots, z_M)}{\partial(y_1, \dots, y_M)} \right|. \quad (11.9)$$

*iid U(0,1)*

determinant of Jacobian  
 $M \times M \quad ij \quad \frac{\partial z_i}{\partial y_j}$

**Figure 11.2** Geometrical interpretation of the transformation method for generating nonuniformly distributed random numbers.  $h(y)$  is the indefinite integral of the desired distribution  $p(y)$ . If a uniformly distributed random variable  $z$  is transformed using  $y = h^{-1}(z)$ , then  $y$  will be distributed according to  $p(y)$ .



# Example: exponential distribution

$$p(y) = \lambda \exp(-\lambda y) \quad (11.7) \quad 0 \leq y < \infty.$$

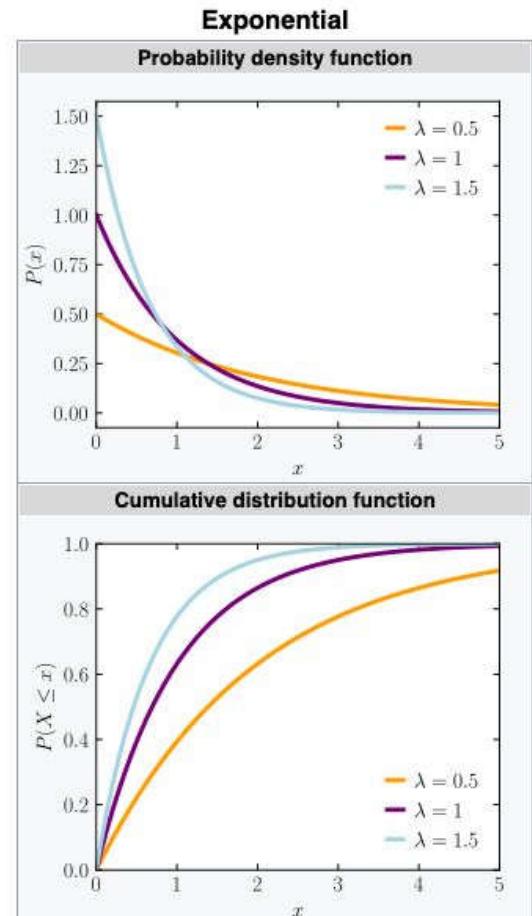
cdf  $h(y) = 1 - \exp(-\lambda y)$

$$y = -\lambda^{-1} \ln(1 - z)$$

$\xleftarrow{\quad}$

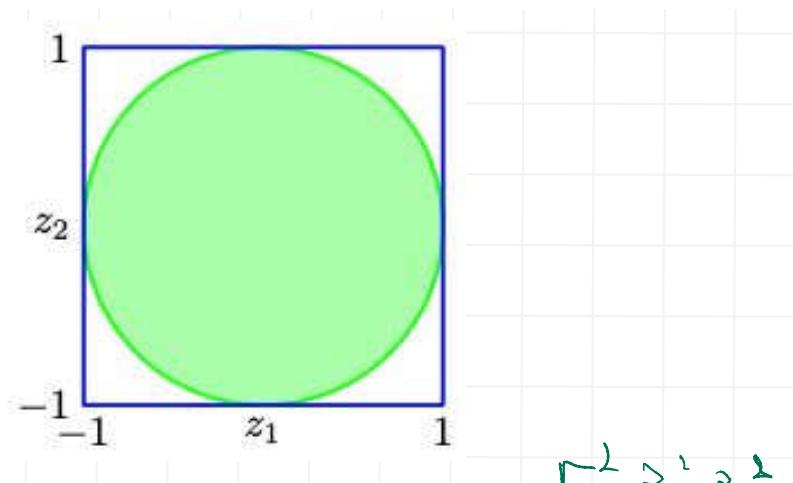
$1 - z = \exp^{-\lambda y}$

$\log(1 - z) = -\lambda y$ .



[https://en.wikipedia.org/wiki/Exponential\\_distribution](https://en.wikipedia.org/wiki/Exponential_distribution)

**Figure 11.3** The Box-Muller method for generating Gaussian distributed random numbers starts by generating samples from a uniform distribution inside the unit circle.



$$r^2 \geq z_1^2 + z_2^2$$

Typo in the book (online pdf version, fixed in the latest print version)

$$y_1 = z_1 \left( \frac{-2 \ln z_1}{r^2} \right)^{1/2} \quad (11.10)$$

$$y_2 = z_2 \left( \frac{-2 \ln z_2}{r^2} \right)^{1/2} \quad (11.11)$$



$$y_1 = z_1 \left( \frac{-2 \ln r^2}{r^2} \right)^{1/2}$$

$$y_2 = z_2 \left( \frac{-2 \ln r^2}{r^2} \right)^{1/2}$$

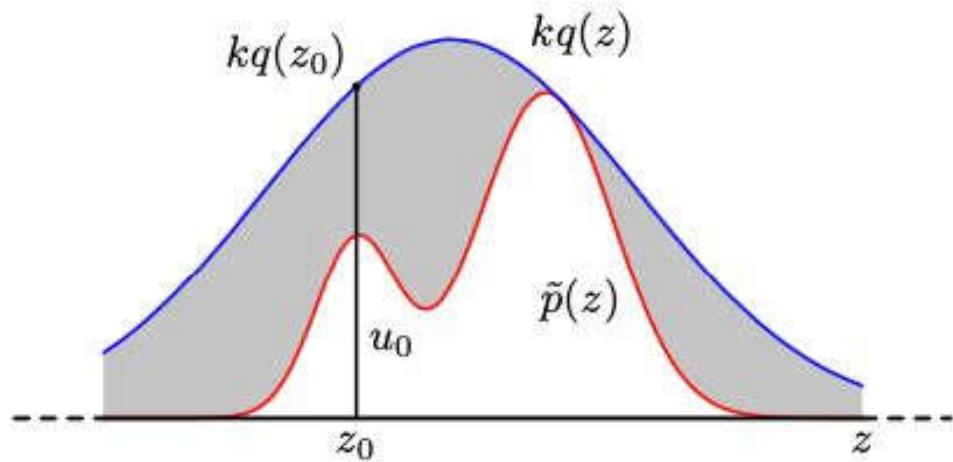
$$\begin{aligned} p(y_1, y_2) &= p(z_1, z_2) \left| \frac{\partial(z_1, z_2)}{\partial(y_1, y_2)} \right| \\ &= \left[ \frac{1}{\sqrt{2\pi}} \exp(-y_1^2/2) \right] \left[ \frac{1}{\sqrt{2\pi}} \exp(-y_2^2/2) \right] \end{aligned} \quad (11.12)$$

Full derivation: through change of variable to the polar coordinates ( $r, \theta$ ), somewhat involved

$$r \sim U(0, 1), \quad \theta \sim U(0, 2\pi)$$

# Rejection sampling

**Figure 11.4** In the rejection sampling method, samples are drawn from a simple distribution  $q(z)$  and rejected if they fall in the grey area between the unnormalized distribution  $\tilde{p}(z)$  and the scaled distribution  $kq(z)$ . The resulting samples are distributed according to  $p(z)$ , which is the normalized version of  $\tilde{p}(z)$ .



Assumption 1 : Sampling directly from  $p(z)$  is difficult, but we can evaluate  $p(z)$  up to some unknown normalisation constant  $Z_p$

$$p(z) = \frac{1}{Z_p} \tilde{p}(z)$$

Assumption 2 : We can draw samples from a simpler distribution  $q(z)$  and for some constant  $k$  and all  $z$  holds

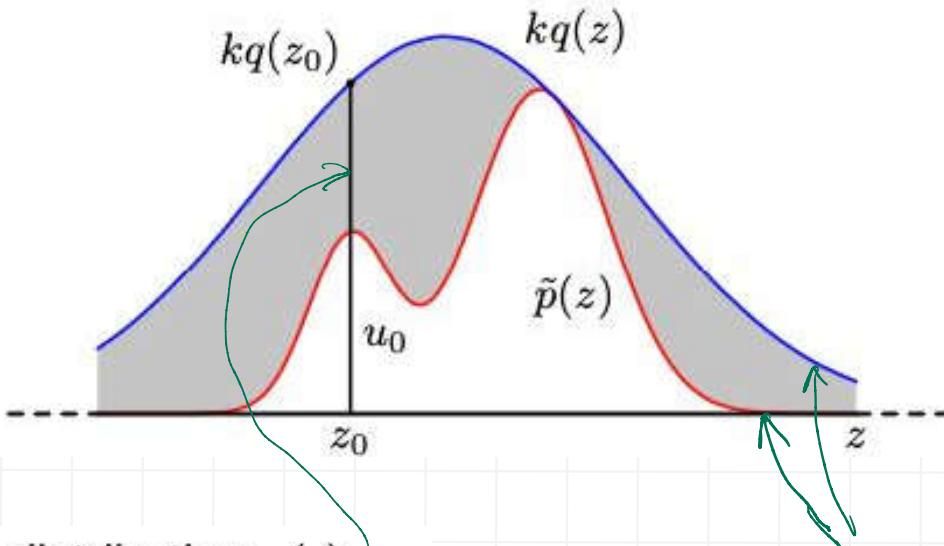
$$\underline{kq(z)} \geq \tilde{p}(z)$$

Impossible:

$$q(z) \geq p(z) \quad \forall z$$

# Rejection sampling

**Figure 11.4** In the rejection sampling method, samples are drawn from a simple distribution  $q(z)$  and rejected if they fall in the grey area between the unnormalized distribution  $\tilde{p}(z)$  and the scaled distribution  $kq(z)$ . The resulting samples are distributed according to  $p(z)$ , which is the normalized version of  $\tilde{p}(z)$ .



$q(z)$  should have  
“heavier” tails  
than  $p(z)$

- ① Generate a random number  $z_0$  from the distribution  $q(z)$ .
- ② Generate a number from the  $u_0$  from the uniform distribution over  $[0, k q(z_0)]$ .
- ③ If  $u_0 > \tilde{p}(z_0)$  then reject the pair  $(z_0, u_0)$ .
- ④ The remaining pairs have uniform distribution under the curve  $\tilde{p}(z)$ .
- ⑤ The  $z$  values are distributed according to  $p(z)$ .

# Rejection sampling example

**Figure 11.5** Plot showing the gamma distribution given by (11.15) as the green curve, with a scaled Cauchy proposal distribution shown by the red curve. Samples from the gamma distribution can be obtained by sampling from the Cauchy and then applying the rejection sampling criterion.

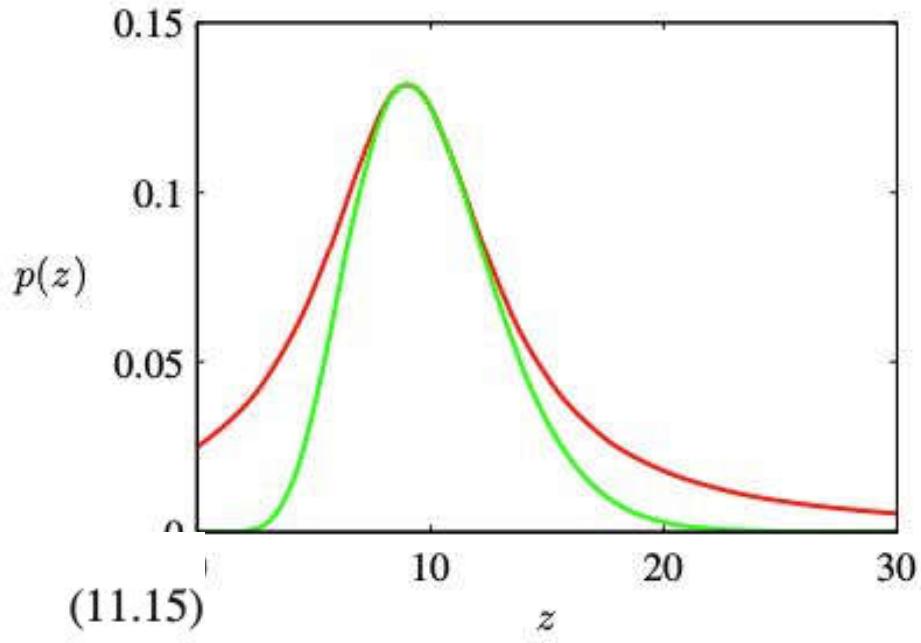
Sample Gamma distribution with  $a > 1$

$$\text{Gam}(z|a, b) = \frac{b^a z^{a-1} \exp(-bz)}{\Gamma(a)}$$

Proposal distribution: Cauchy  $p(y) = \frac{1}{\pi} \frac{1}{1 + y^2}$

Or,

$$q(z) = \frac{k}{1 + (z - c)^2/b^2}$$



(11.15)

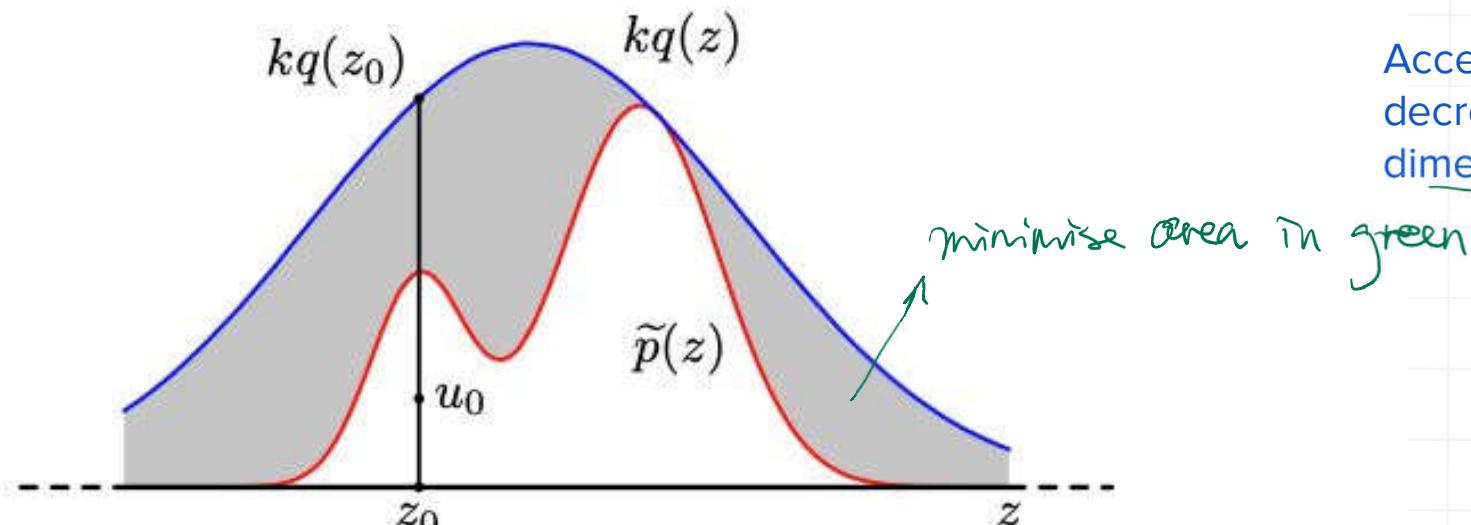
(11.8)

$$u \sim U[0, 1]$$
$$z = b \tan u + c$$

## Rejection sampling - challenges

- Need to find a proposal distribution  $q(z)$  which is a **close upper bound** to  $p(z)$ ; otherwise many samples are rejected.
- Curse of dimensionality for multivariate distributions.

Proposal distribution need to have heavier tails than the target distribution.



Acceptance rate can decreases exponentially as dimensionality increases.

# Importance sampling

Desired:  $\mathbb{E}[f] \simeq \sum_{l=1}^L p(\mathbf{z}^{(l)}) f(\mathbf{z}^{(l)}).$  (11.18)

- Directly calculate  $\mathbb{E}_p [f(z)]$  w.r.t. some  $p(z).$
- Does not sample  $p(z)$  as an intermediate step.
- Again use samples  $z^{(l)}$  from some proposal  $q(z).$

All samples retained.

$$\begin{aligned}\mathbb{E}[f] &= \int f(\mathbf{z}) p(\mathbf{z}) d\mathbf{z} \\ &= \int f(\mathbf{z}) \frac{p(\mathbf{z})}{q(\mathbf{z})} q(\mathbf{z}) d\mathbf{z} \\ &\simeq \frac{1}{L} \sum_{l=1}^L \frac{p(\mathbf{z}^{(l)})}{q(\mathbf{z}^{(l)})} f(\mathbf{z}^{(l)}).\end{aligned}\quad (11.19)$$

Importance weights

$$r_l = \frac{p(\mathbf{z}^{(l)})}{q(\mathbf{z}^{(l)})}$$

$r_l$  can be  $> 1$  ↗

# Estimating the (ratio of) normalization constants

Easy to evaluate distribution up to a **normalising constant**, but hard to know what the constant is.  $p(\mathbf{z}) = \tilde{p}(\mathbf{z})/Z_p$        $\underbrace{q(\mathbf{z})}_{\text{simple.}} = \tilde{q}(\mathbf{z})/Z_q$

$$\begin{aligned}\mathbb{E}[f] &= \int f(\mathbf{z}) p(\mathbf{z}) d\mathbf{z} \\ &= \frac{Z_q}{Z_p} \int f(\mathbf{z}) \frac{\tilde{p}(\mathbf{z})}{\tilde{q}(\mathbf{z})} q(\mathbf{z}) d\mathbf{z} \\ &\approx \frac{Z_q}{Z_p} \frac{1}{L} \sum_{l=1}^L \tilde{r}_l f(\mathbf{z}^{(l)}). \tag{11.20}\end{aligned}$$

$$\begin{aligned}\tilde{r}_l &\neq r_l = \frac{p(\mathbf{z})}{q(\mathbf{z})} \\ &\rightarrow \frac{Z_p}{Z_q} r_l.\end{aligned}$$

$$\begin{aligned}\frac{Z_p}{Z_q} &= \frac{1}{Z_q} \int \tilde{p}(\mathbf{z}) d\mathbf{z} = \int \frac{\tilde{p}(\mathbf{z})}{\tilde{q}(\mathbf{z})} q(\mathbf{z}) d\mathbf{z} \\ &\approx \frac{1}{L} \sum_{l=1}^L \tilde{r}_l \quad \xrightarrow{\text{average of unnormalized importance weights!}} \tag{11.21}\end{aligned}$$

Using the same samples!

## Importance sampling - comments

! Support of  
 $q(z)$   
is a superset of  
support of  $p(z)$

- Importance weights  $r_l$  correct the bias introduced by sampling from the proposal distribution  $q(z)$  instead of the wanted distribution  $p(z)$ .
- Success depends on how well  $q(z)$  approximates  $p(z)$ .
- If  $p(z) > 0$  in same region, then  $q(z) > 0$  necessary.

effective sample size can be much smaller than the apparent sample size  $L$ . The problem is even more severe if none of the samples falls in the regions where  $p(z)f(z)$  is large. In that case, the apparent variances of  $r_l$  and  $r_l f(z^{(l)})$  may be small even though the estimate of the expectation may be severely wrong. Hence a major drawback of the importance sampling method is the potential to produce results that are arbitrarily in error and with no diagnostic indication. This also highlights a key requirement for the sampling distribution  $q(z)$ , namely that it should not be small or zero in regions where  $p(z)$  may be significant.

## Sampling and the EM algorithm

$$Q(\theta, \theta^{\text{old}}) = \int p(\mathbf{Z}|\mathbf{X}, \theta^{\text{old}}) \ln p(\mathbf{Z}, \mathbf{X}|\theta) d\mathbf{Z}. \quad (11.28)$$

$p(\mathbf{Z})$        $f(\mathbf{z})$

Sample this       $\sum^{(l)}$

Monte Carlo EM

$$\underbrace{Q(\theta, \theta^{\text{old}})}_{\text{---}} \simeq \frac{1}{L} \sum_{l=1}^L \ln p(\mathbf{Z}^{(l)}, \mathbf{X}|\theta). \quad (11.29)$$

M-step: as usual,

# Being Bayesian in EM

## IP Algorithm

**I-step.** We wish to sample from  $p(\mathbf{Z}|\mathbf{X})$  but we cannot do this directly. We therefore note the relation

$$p(\mathbf{Z}|\mathbf{X}) = \int p(\mathbf{Z}|\boldsymbol{\theta}, \mathbf{X})p(\boldsymbol{\theta}|\mathbf{X}) d\boldsymbol{\theta} \quad (11.30)$$

and hence for  $l = 1, \dots, L$  we first draw a sample  $\boldsymbol{\theta}^{(l)}$  from the current estimate for  $p(\boldsymbol{\theta}|\mathbf{X})$ , and then use this to draw a sample  $\mathbf{Z}^{(l)}$  from  $p(\mathbf{Z}|\boldsymbol{\theta}^{(l)}, \mathbf{X})$ .

**P-step.** Given the relation

$$p(\boldsymbol{\theta}|\mathbf{X}) = \int p(\boldsymbol{\theta}|\mathbf{Z}, \mathbf{X})p(\mathbf{Z}|\mathbf{X}) d\mathbf{Z} \quad (11.31)$$

we use the samples  $\{\mathbf{Z}^{(l)}\}$  obtained from the I-step to compute a revised estimate of the posterior distribution over  $\boldsymbol{\theta}$  given by

$$p(\boldsymbol{\theta}|\mathbf{X}) \simeq \frac{1}{L} \sum_{l=1}^L p(\boldsymbol{\theta}|\mathbf{Z}^{(l)}, \mathbf{X}). \quad (11.32)$$

By assumption, it will be feasible to sample from this approximation in the I-step.

# Sampling methods

Motivation, why? Sampling and ML

Basic sampling algorithms

- Sampling standard distributions from  $U(0, 1)$
- Rejection sampling
- Importance sampling

Markov chain Monte Carlo (MCMC)

- Markov chains
- Metropolis-Hastings

Gibbs sampling

assume: ① can evaluate  $P(z)$

for any given  $z$ .

② evaluate  $P(z)|_{z=z_0}$

is easier than

Sampling  $z \sim P(z)$



involves Inverse CDF

# Markov Chain Monte Carlo (MCMC)

$$E_p[f(z)]$$

Goal: sample from  $p(z)$ .

Generate a sequence using a Markov chain.

- ① Generate a new sample  $z^* \sim q(z | z^{(l)})$ , conditional on the previous sample  $z^{(l)}$ .
- ② Accept or reject the new sample according to some appropriate criterion.

$$\underline{z}^{(l+1)} = \begin{cases} z^* & \text{if accepted} \\ z^{(l)} & \text{if rejected} \end{cases}$$

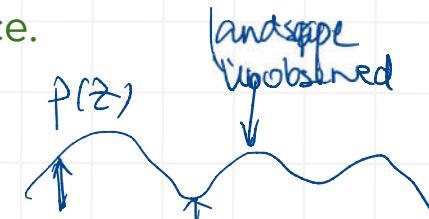
*— copy of the prev value.*

- ③ For an appropriate proposal and corresponding acceptance criterion, as  $l \rightarrow \infty$ ,  $\underline{z}^{(l)}$  approaches an independent sample of  $p(z)$ .

Motivation:

Cover high-probability regions of  $p(z)$  -- can we move towards it?

Scale better with the dimensionality of the sample space.



# Metropolis Algorithm

- 1 Choose a symmetric proposal distribution

$$q(z_A | z_B) = q(z_B | z_A).$$

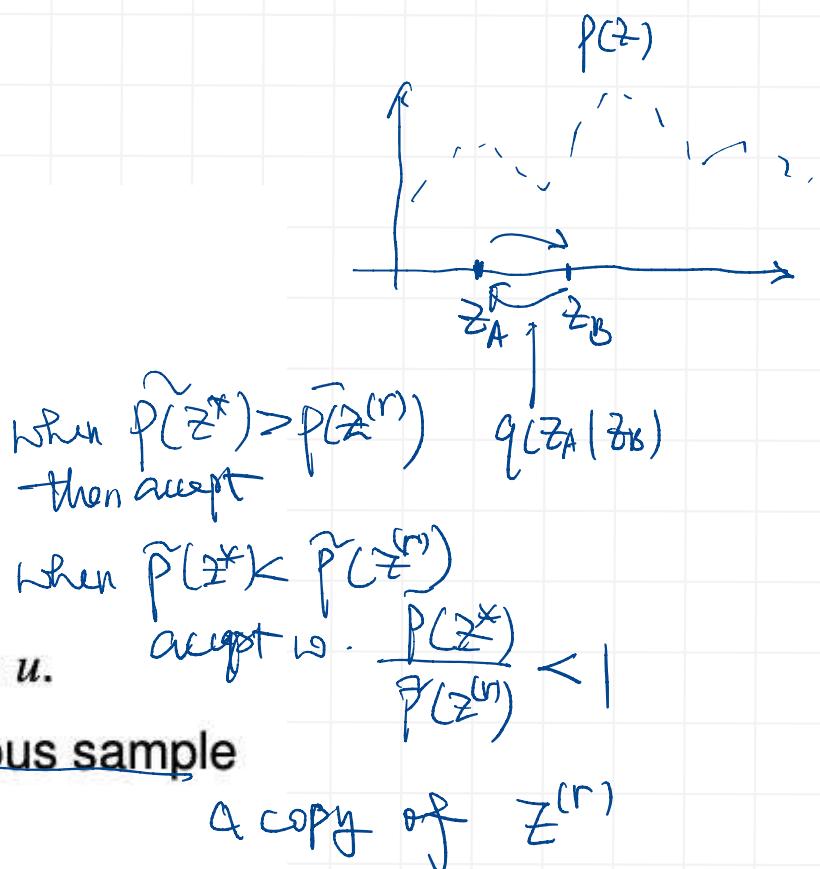
- 2 Accept the new sample  $z^*$  with probability

$$A(z^*, z^{(r)}) = \min \left( 1, \frac{\tilde{p}(z^*)}{\tilde{p}(z^{(r)})} \right)$$

e.g., let  $u \sim \text{Uniform}(0, 1)$  and accept if  $\frac{\tilde{p}(z^*)}{\tilde{p}(z^{(r)})} > u$ .

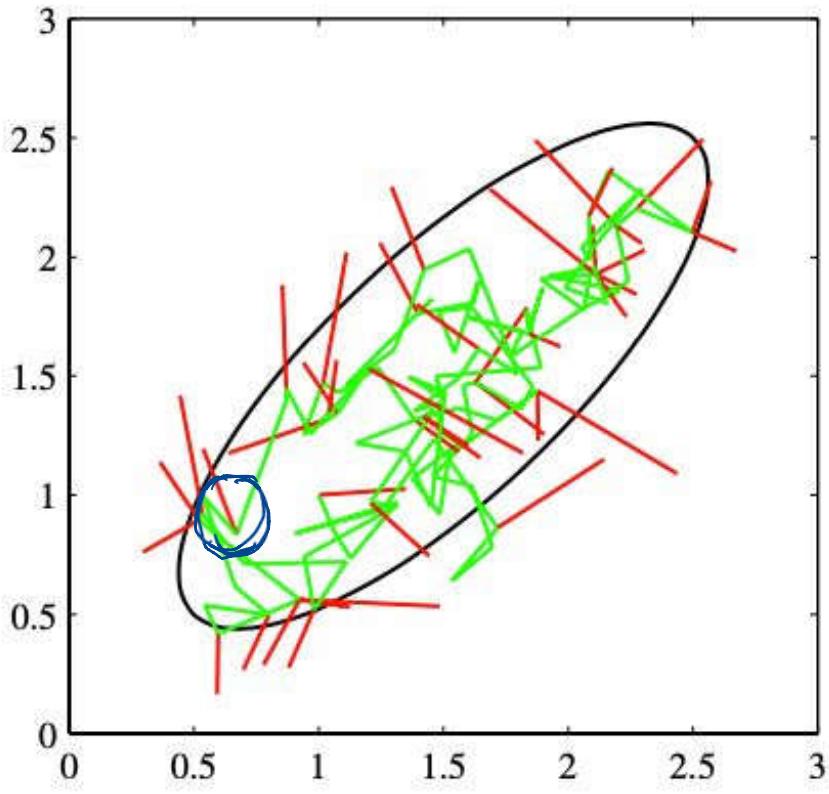
- 3 Unlike rejection sampling we include the previous sample on rejection of the proposal:

$$z^{(r+1)} = \begin{cases} z^* & \text{if accepted} \\ z^{(r)} & \text{if rejected} \end{cases}$$



# Metropolis Algorithm - illustration

**Figure 11.9** A simple illustration using Metropolis algorithm to sample from a Gaussian distribution whose one standard-deviation contour is shown by the ellipse. The proposal distribution is an isotropic Gaussian distribution whose standard deviation is 0.2. Steps that are accepted are shown as green lines, and rejected steps are shown in red. A total of 150 candidate samples are generated, of which 43 are rejected.



# Random walk as a Markov Chain

Consider a state space  $z$  consisting of the integers, with probabilities

$$p(z^{(\tau+1)} = z^{(\tau)}) = 0.5 \quad (11.34)$$

$$p(z^{(\tau+1)} = z^{(\tau)} + 1) = 0.25 \quad (11.35)$$

$$p(z^{(\tau+1)} = z^{(\tau)} - 1) = 0.25 \quad (11.36)$$

if  $z^{(1)} = 0$ ,

then  $\mathbb{E}[z^{(\tau)}] = 0$

$\mathbb{E}[(z^{(\tau)})^2] = \tau/2.$

after  $\tau$  steps, the random walk has only travelled a distance that on average is proportional to the square root of  $\tau$   
→ Random walks are very inefficient in exploring the state space.

Design goal of MCMC algorithms: avoid random-walk behaviour.

# First order Markov Chain

A series of random variable  $\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(M)}$  such that

$$p(\mathbf{z}^{(m+1)} | \mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}) = p(\mathbf{z}^{(m+1)} | \mathbf{z}^{(m)}). \quad (11.37)$$

chain shaped  
graphical model!



Marginal probability

$$\begin{aligned} p(z^{(m+1)}) &= \sum_{z^{(m)}} p(z^{(m+1)} | z^{(m)}) p(z^{(m)}) \\ &= \sum_{z^{(m)}} T_m(z^{(m)} | z^{(m+1)}) p(z^{(m)}) \end{aligned}$$

where  $T_m(z^{(m)} | z^{(m+1)})$  are the transition probabilities.

# MCMC - why it works

- Marginal probability

$$p(z^{(m+1)}) = \sum_{z^{(m)}} T_m(z^{(m)} | z^{(m+1)}) p(z^{(m)})$$

*drop  $z^{(m+1)}$*

- A Markov chain is called **homogeneous** if the transition probabilities are the same for all  $m$ , denoted by  $T(z', z)$ .
- A distribution is **invariant**, or **stationary**, with respect to a Markov chain if each step leaves the distribution invariant.
- For a homogeneous Markov chain, the distribution  $p^*(z)$  is invariant if

$$\underbrace{p^*(z)}_{z'} = \sum_{z'} \underbrace{T(z', z)}_{z'} \underbrace{p^*(z')}_{z'}. \quad (11.39)$$

(Note: There can be many. If  $T$  is the identity matrix, every distribution is invariant.)

*↑ we stay put at all values of  $z$ .*

## MCMC - why it works

- **Detailed balance**

$$p(z \rightarrow z')$$

$$p(z' \rightarrow z)$$

$$\underbrace{p^*(z)T(z, z')}_{\text{Detailed balance}} = \underbrace{p^*(z')T(z', z)}$$

is sufficient (but not necessary) for  $p^*(z)$  to be invariant (to check, put (2) into (1)). (A Markov chain that respects the detailed balance is called **reversible**.)

- A Markov chain is **ergodic** if it converges to the invariant distribution irrespective of the choice of the initial conditions. The invariant distribution is then called **equilibrium**.
- An ergodic Markov chain can **have only one equilibrium distribution**.
- Why is it working? Choose the transition probabilities  $T$  to satisfy the detailed balance for our goal distribution  $p(z)$ .

We want

$$p^*(z) = \sum_{z'} T(z', z)p^*(z').$$

(11.39)

$\forall z$

(11.40)

repeat many steps.

$$p^*(z) \rightarrow p(z).$$

Desired: Avoid rejection

## The Metropolis-Hastings Algorithm

- Choose a symmetric proposal distribution  $q(z_A | z_B) = q(z_B | z_A)$ .

- Generalisation of the Metropolis algorithm for nonsymmetric proposal distributions  $q_k$ .
- At step  $\tau$ , draw a sample  $z^*$  from the distribution  $q_k(z | z^{(\tau)})$  where  $k$  labels the set of possible transitions.
- Accept with probability

Metropolis requires  $q_k(z^{(t)} | z^*) = q_k(z^* | z^{(t)})$

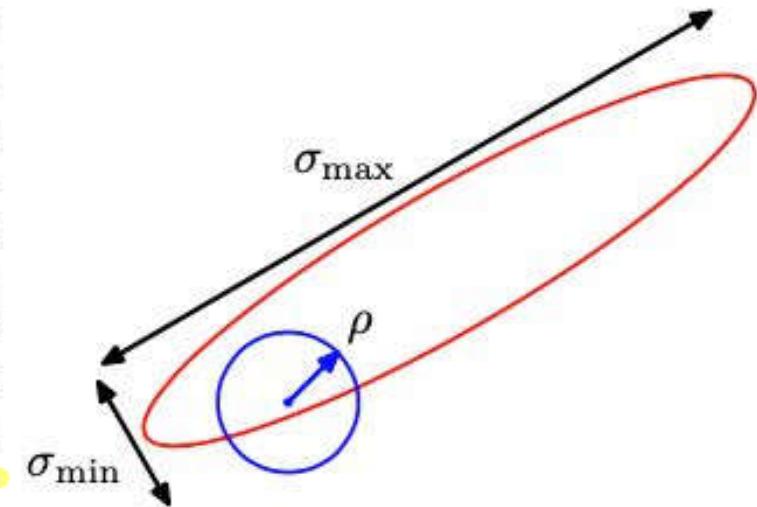
$$A_k(z^*, z^{(\tau)}) = \min \left( 1, \frac{\tilde{p}(z^*) q_k(z^{(\tau)} | z^*)}{\tilde{p}(z^{(\tau)}) q_k(z^* | z^{(\tau)})} \right). \quad (11.44)$$

- Accept the new sample  $z^*$  with probability

$$A(z^*, z^{(\tau)}) = \min \left( 1, \frac{\tilde{p}(z^*)}{\tilde{p}(z^{(\tau)})} \right)$$

- Choice of proposal distribution critical.
- Common choice : Gaussian centered on the current state.
  - small variance  $\rightarrow$  high acceptance rate, but slow walk through the state space; samples not independent
  - large variance  $\rightarrow$  high rejection rate

**Figure 11.10** Schematic illustration of the use of an isotropic Gaussian proposal distribution (blue circle) to sample from a correlated multivariate Gaussian distribution (red ellipse) having very different standard deviations in different directions, using the Metropolis-Hastings algorithm. In order to keep the rejection rate low, the scale  $\rho$  of the proposal distribution should be on the order of the smallest standard deviation  $\sigma_{\min}$ , which leads to random walk behaviour in which the number of steps separating states that are approximately independent is of order  $(\sigma_{\max}/\sigma_{\min})^2$  where  $\sigma_{\max}$  is the largest standard deviation.



# Metropolis-Hastings: why does it work?

$$A_k(\mathbf{z}^*, \mathbf{z}^{(\tau)}) = \min \left( 1, \frac{\tilde{p}(\mathbf{z}^*) q_k(\mathbf{z}^{(\tau)} | \mathbf{z}^*)}{\tilde{p}(\mathbf{z}^{(\tau)}) q_k(\mathbf{z}^* | \mathbf{z}^{(\tau)})} \right)$$

- Transition probability of this Markov chain is

$$T(z, z') = q_k(z' | z) A_k(z', z)$$

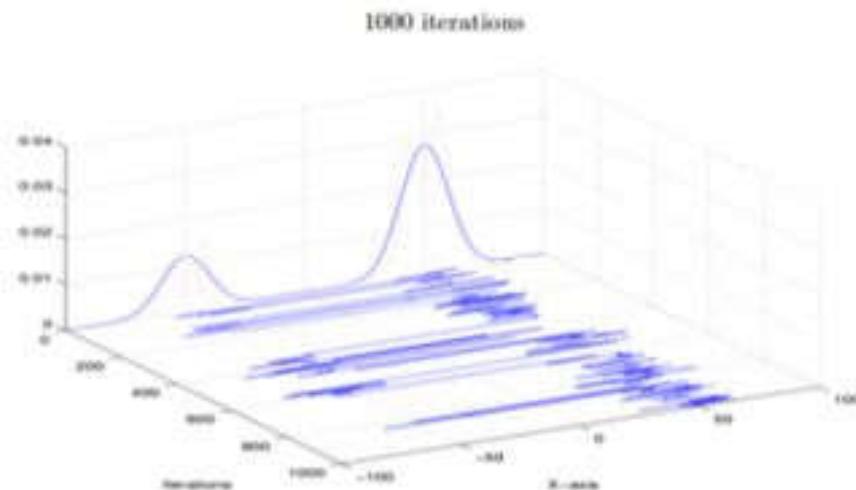
- Prove that  $p(z)$  is the invariant distribution if the detailed balance holds

$$p(z) T(z, z') = T(z', z) p(z').$$

- Using the symmetry  $\min(a, b) = \min(b, a)$  it can be shown that the detailed balance holds

$$\begin{aligned} p(\mathbf{z}) q_k(\mathbf{z} | \mathbf{z}') A_k(\mathbf{z}', \mathbf{z}) &= \min(p(\mathbf{z}) q_k(\mathbf{z} | \mathbf{z}'), p(\mathbf{z}') q_k(\mathbf{z}' | \mathbf{z})) \\ &= \min(p(\mathbf{z}') q_k(\mathbf{z}' | \mathbf{z}), p(\mathbf{z}) q_k(\mathbf{z} | \mathbf{z}')) \\ &= p(\mathbf{z}') q_k(\mathbf{z}' | \mathbf{z}) A_k(\mathbf{z}, \mathbf{z}') \end{aligned} \tag{11.45}$$

## Metropolis-Hastings for sampling from mixture of Gaussians:



<http://www.cs.ubc.ca/~arnaud/stat535/slides10.pdf>

- With a random walk  $q$  we may get stuck in one mode.
- We could have proposal be mixture between random walk and "mode jumping".

## Gibbs Sampling

- Goal: sample from a joint distribution  $p(\mathbf{z}) = p(z_1, \dots, z_M)$
- How? sample one variable from the distribution conditioned on all the other variable
- Example : given  $p(z_1, z_2, z_3)$
- At step  $\tau$  we have samples  $z_1^{(\tau)}$ ,  $z_2^{(\tau)}$  and  $z_3^{(\tau)}$ .
- Get samples for the next step  $\tau + 1$

$$z_1^{(\tau+1)} \sim p(z_1^{(\tau)} | z_2^{(\tau)}, z_3^{(\tau)})$$

$$z_2^{(\tau+1)} \sim p(z_2^{(\tau)} | z_1^{(\tau+1)}, z_3^{(\tau)})$$

$$z_3^{(\tau+1)} \sim p(z_3^{(\tau)} | z_1^{(\tau+1)}, z_2^{(\tau+1)})$$

## Gibbs sampling - why does it work?

- ①  $p(\mathbf{z})$  is invariant of each of the Gibbs sampling steps and hence of the whole Markov chain : a)  $p(z_i | \{\mathbf{z}_{\setminus i}\})$  is invariant because the marginal distribution  $p(\mathbf{z}_{\setminus i})$  does not change, and b) by definition each step samples from  $p(z_i | \{\mathbf{z}_{\setminus i}\})$ .
- ② Ergodicity: sufficient that none of the conditional distribution is zero anywhere.
- ③ Gibbs sampling is a Metropolis-Hastings sampling in which each step is accepted.

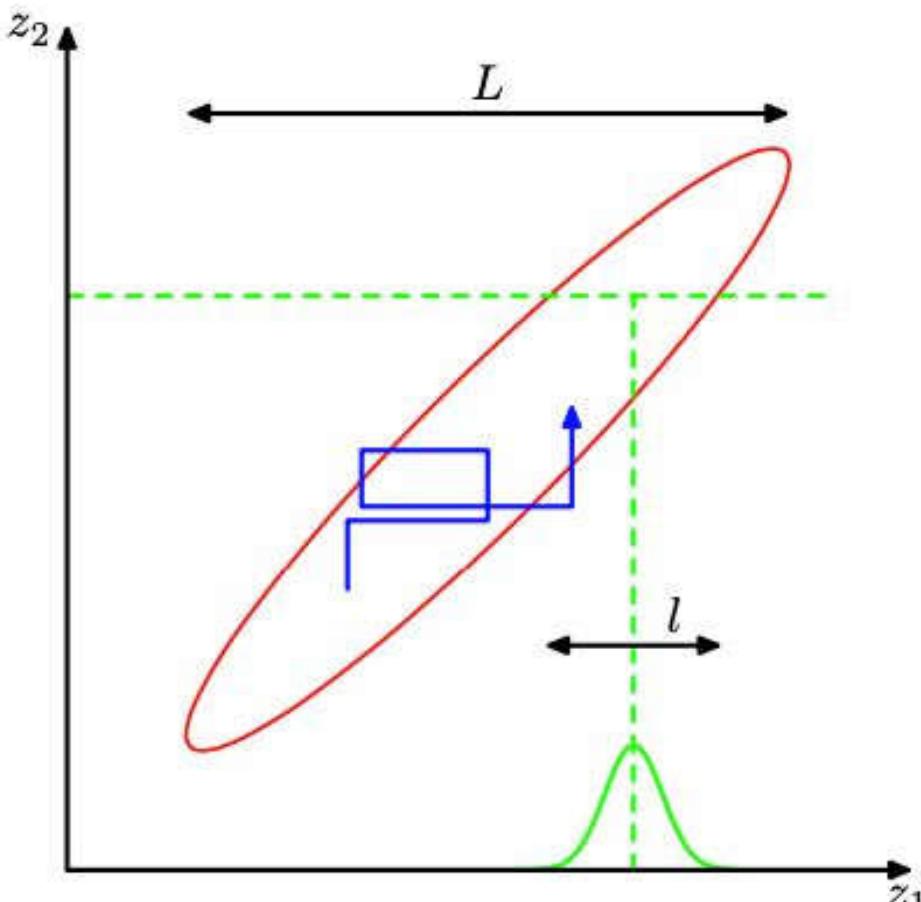
$$A(\mathbf{z}^*, \mathbf{z}) = \frac{p(\mathbf{z}^*) q_k(\mathbf{z} | \mathbf{z}^*)}{p(\mathbf{z}) q_k(\mathbf{z}^* | \mathbf{z})} = \frac{p(z_k^* | \mathbf{z}_{\setminus k}^*) p(\mathbf{z}_{\setminus k}^*) p(z_k | \mathbf{z}_{\setminus k}^*)}{p(z_k | \mathbf{z}_{\setminus k}) p(\mathbf{z}_{\setminus k}) p(z_k^* | \mathbf{z}_{\setminus k})} = 1 \quad (11.49)$$

**Figure 11.11**

Illustration of Gibbs sampling by alternate updates of two variables whose distribution is a correlated Gaussian. The step size is governed by the standard deviation of the conditional distribution (green curve), and is  $O(l)$ , leading to slow progress in the direction of elongation of the joint distribution (red ellipse). The number of steps needed to obtain an independent sample from the distribution is  $O((L/l)^2)$ .

initial goal  $E_p[f(x)]$

practically: keep one sample every  $L$  samples



## Gibbs Sampling

1. Initialize  $\{z_i : i = 1, \dots, M\}$
2. For  $\tau = 1, \dots, T$ :
  - Sample  $z_1^{(\tau+1)} \sim p(z_1 | z_2^{(\tau)}, z_3^{(\tau)}, \dots, z_M^{(\tau)})$ .
  - Sample  $z_2^{(\tau+1)} \sim p(z_2 | z_1^{(\tau+1)}, z_3^{(\tau)}, \dots, z_M^{(\tau)})$ .
  - ⋮
  - Sample  $z_j^{(\tau+1)} \sim p(z_j | z_1^{(\tau+1)}, \dots, z_{j-1}^{(\tau+1)}, z_{j+1}^{(\tau)}, \dots, z_M^{(\tau)})$ .
  - ⋮
  - Sample  $z_M^{(\tau+1)} \sim p(z_M | z_1^{(\tau+1)}, z_2^{(\tau+1)}, \dots, z_{M-1}^{(\tau+1)})$ .

# Sampling methods

Motivation, why? Sampling and ML

Basic sampling algorithms

- Sampling standard distributions from  $U(0, 1)$
- Rejection sampling
- Importance sampling



Markov chain Monte Carlo (MCMC)

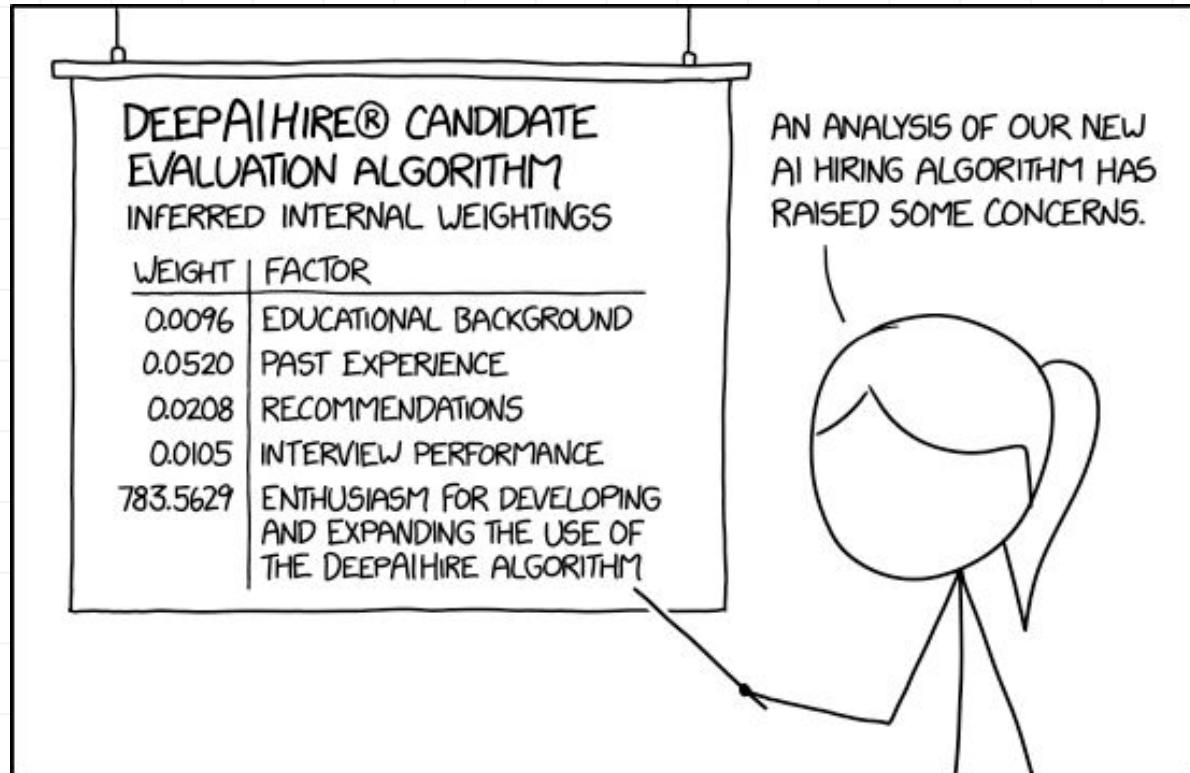
- Markov chains
- Metropolis-Hastings

Gibbs sampling

<https://xkcd.com/1838/>



<https://xkcd.com/2237/>

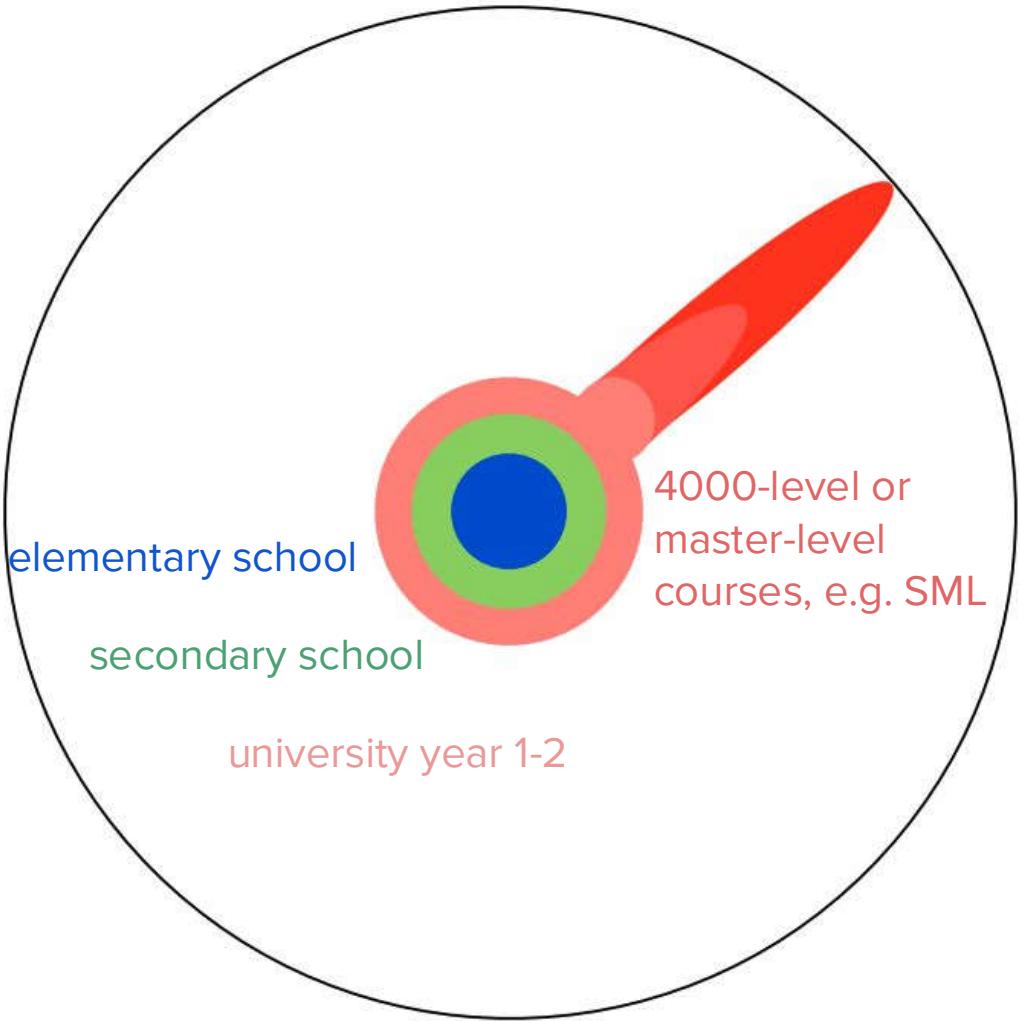


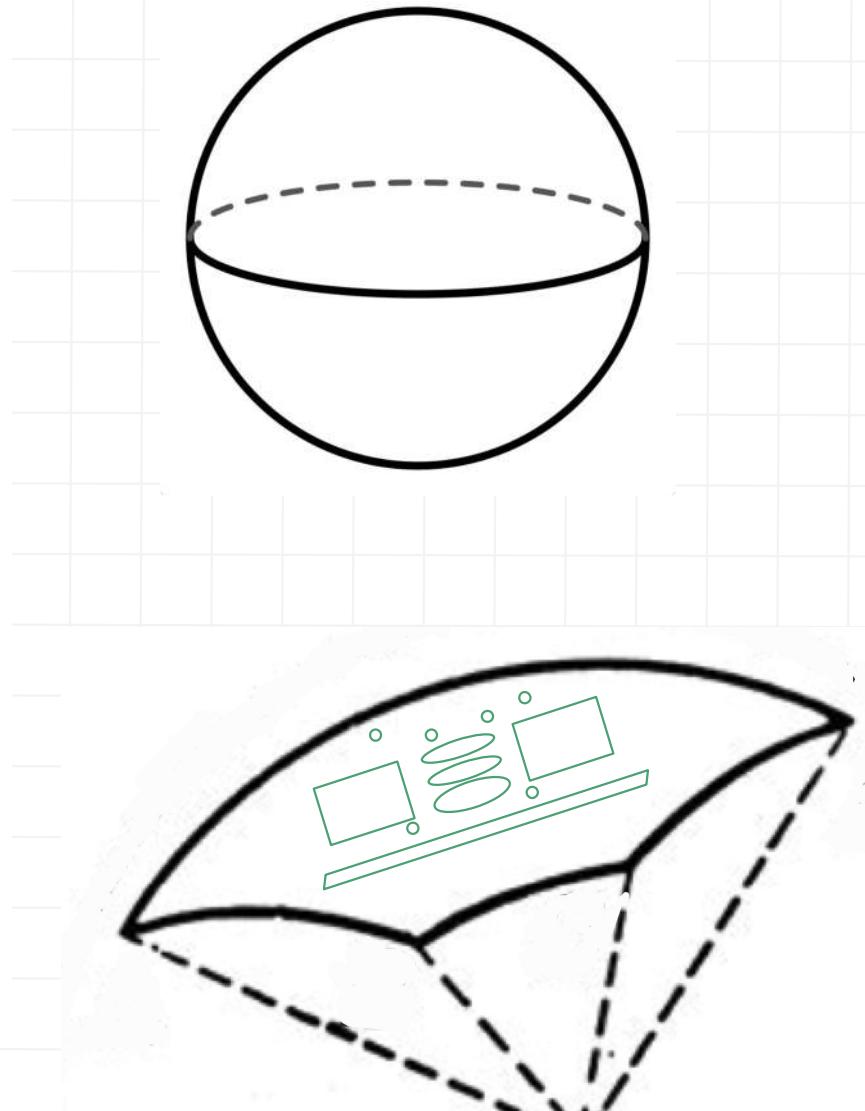
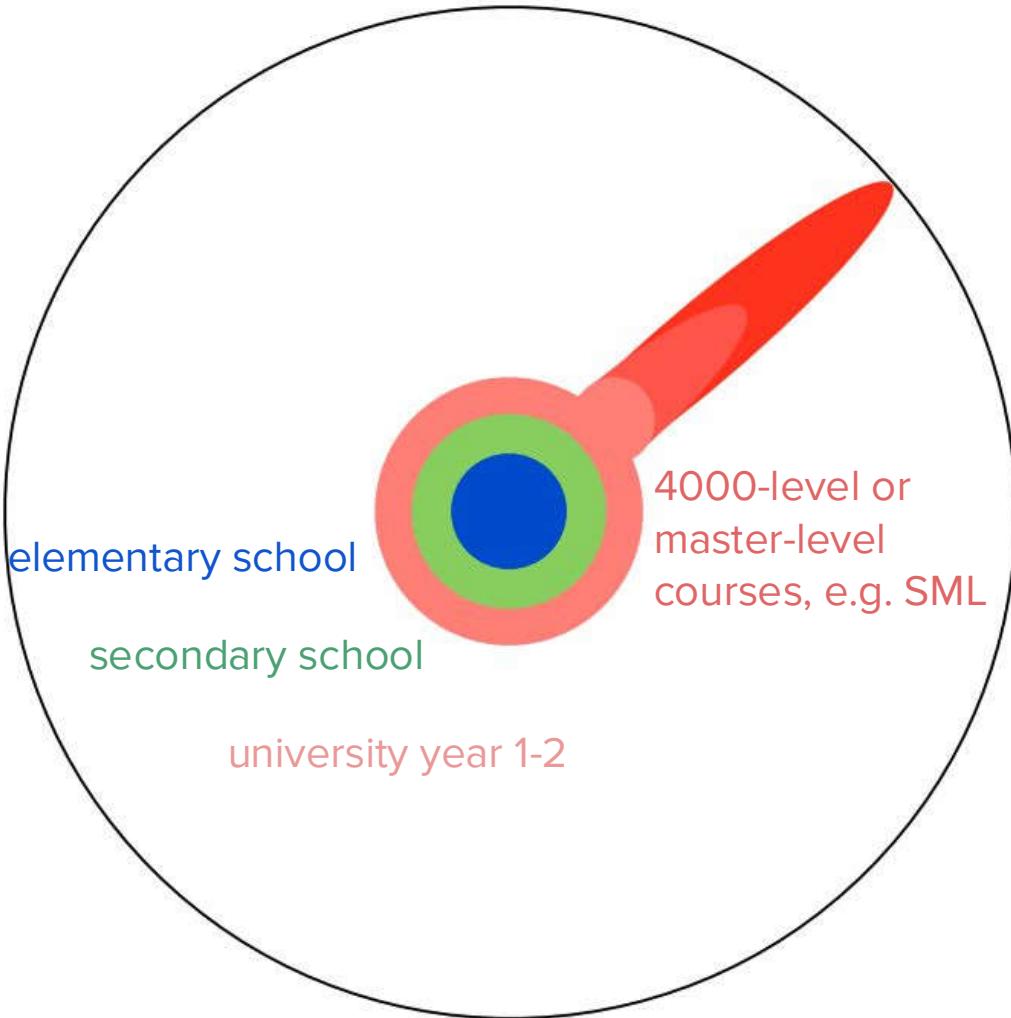
# Plan for today

Reflecting SML (aka course review)

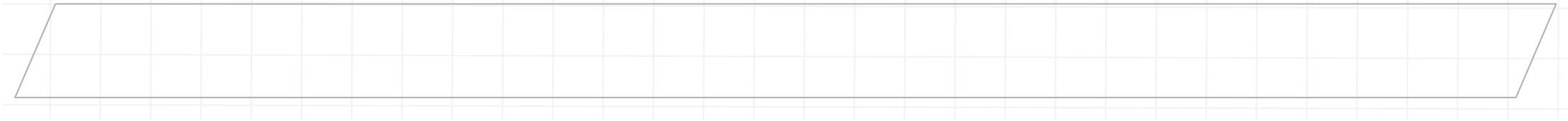
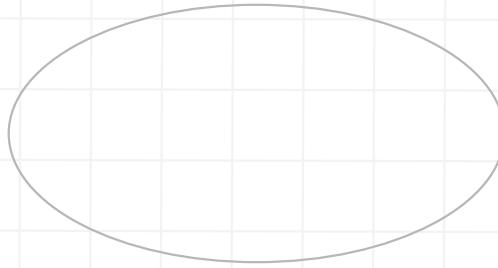
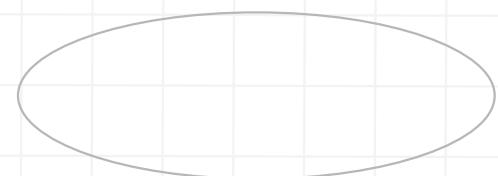
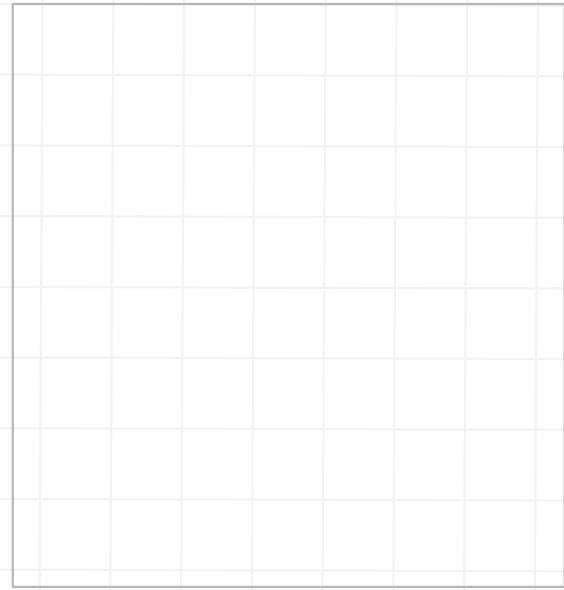
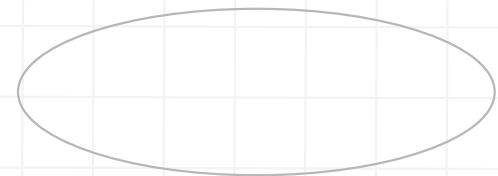
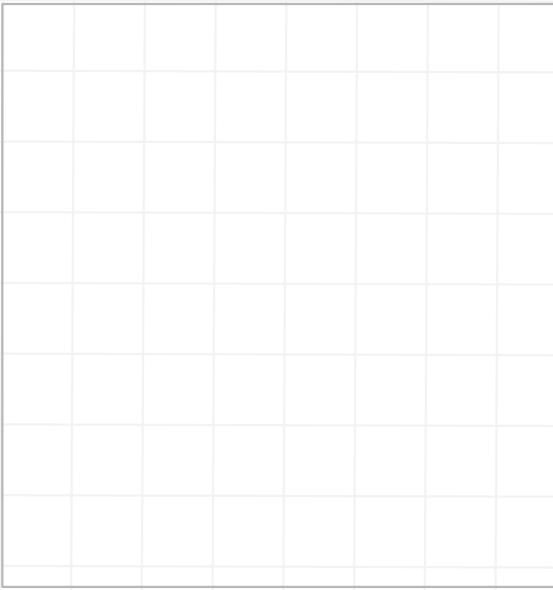
How to debug ML for an application?

Communicating ML (to your family and relatives, friends studying X, your boss's boss's boss at work, the public)





The Illustrated Guide to a Ph.D. by [Matt Might](#)  
<https://matt.might.net/articles/phd-school-in-pictures/>



# Debugging Learning Algorithms

Slides 3-19 from <http://cs229.stanford.edu/materials/ML-advice.pdf>

By Andrew Ng

# **Advice for applying Machine Learning**

Andrew Ng

Stanford University

# Today's Lecture

---

- Advice on how getting learning algorithms to different applications.
- Most of today's material is not very mathematical. But it's also some of the hardest material in this class to understand.
- Some of what I'll say today is debatable.
- Some of what I'll say is not good advice for doing novel machine learning research.
- Key ideas:
  1. Diagnostics for debugging learning algorithms.
  2. Error analyses and ablative analysis.
  3. How to get started on a machine learning problem.
    - Premature (statistical) optimization.



# **Debugging Learning Algorithms**

# Debugging learning algorithms

Motivating example:

- Anti-spam. You carefully choose a small set of 100 words to use as features. (Instead of using all 50000+ words in English.)
- Bayesian logistic regression, implemented with gradient descent, gets 20% test error, which is unacceptably high.

$$\max_{\theta} \sum_{i=1}^m \log p(y^{(i)} | x^{(i)}, \theta) - \lambda \|\theta\|^2$$

- What to do next?



# Fixing the learning algorithm

- Bayesian logistic regression:

$$\max_{\theta} \sum_{i=1}^m \log p(y^{(i)}|x^{(i)}, \theta) - \lambda \|\theta\|^2$$

- Common approach: Try improving the algorithm in different ways.
  - Try getting more training examples.
  - Try a smaller set of features.
  - Try a larger set of features.
  - Try changing the features: Email header vs. email body features.
  - Run gradient descent for more iterations.
  - Try Newton's method.
  - Use a different value for  $\lambda$ .
  - Try using an SVM.
- This approach might work, but it's very time-consuming, and largely a matter of luck whether you end up fixing what the problem really is.



# Diagnostic for bias vs. variance

Better approach:

- Run diagnostics to figure out what the problem is.
- Fix whatever the problem is.

Bayesian logistic regression's test error is 20% (unacceptably high).

Suppose you suspect the problem is either:

- Overfitting (high variance).
- Too few features to classify spam (high bias).

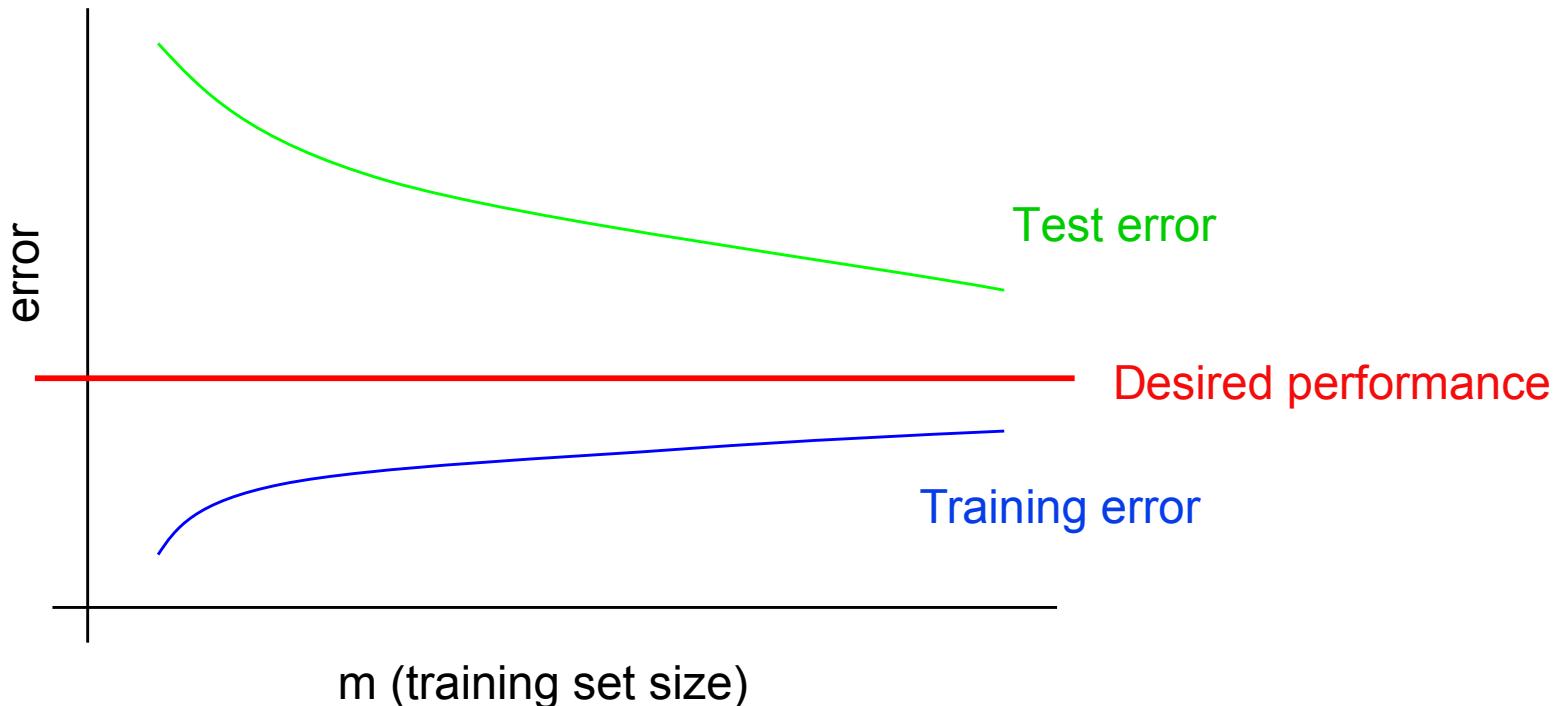
Diagnostic:

- Variance: Training error will be much lower than test error.
- Bias: Training error will also be high.



# More on bias vs. variance

Typical learning curve for high variance:

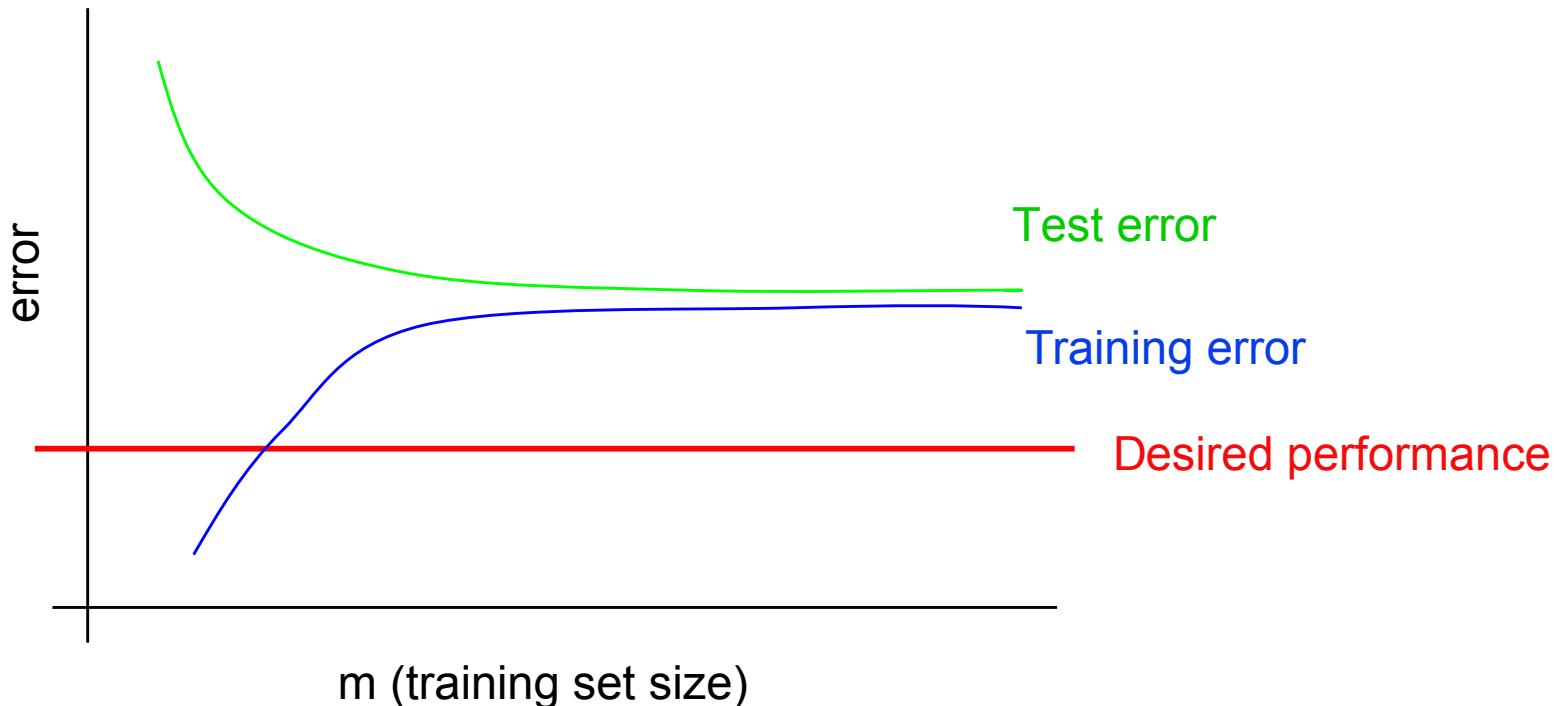


- Test error still decreasing as  $m$  increases. Suggests larger training set will help.
- Large gap between training and test error.



# More on bias vs. variance

Typical learning curve for high bias:



- Even training error is unacceptably high.
- Small gap between training and test error.



# Diagnostics tell you what to try next

Bayesian logistic regression, implemented with gradient descent.

Fixes to try:

- Try getting more training examples. Fixes high variance.
- Try a smaller set of features. Fixes high variance.
- Try a larger set of features. Fixes high bias.
- Try email header features. Fixes high bias.
- Run gradient descent for more iterations.
- Try Newton's method.
- Use a different value for  $\lambda$ .
- Try using an SVM.



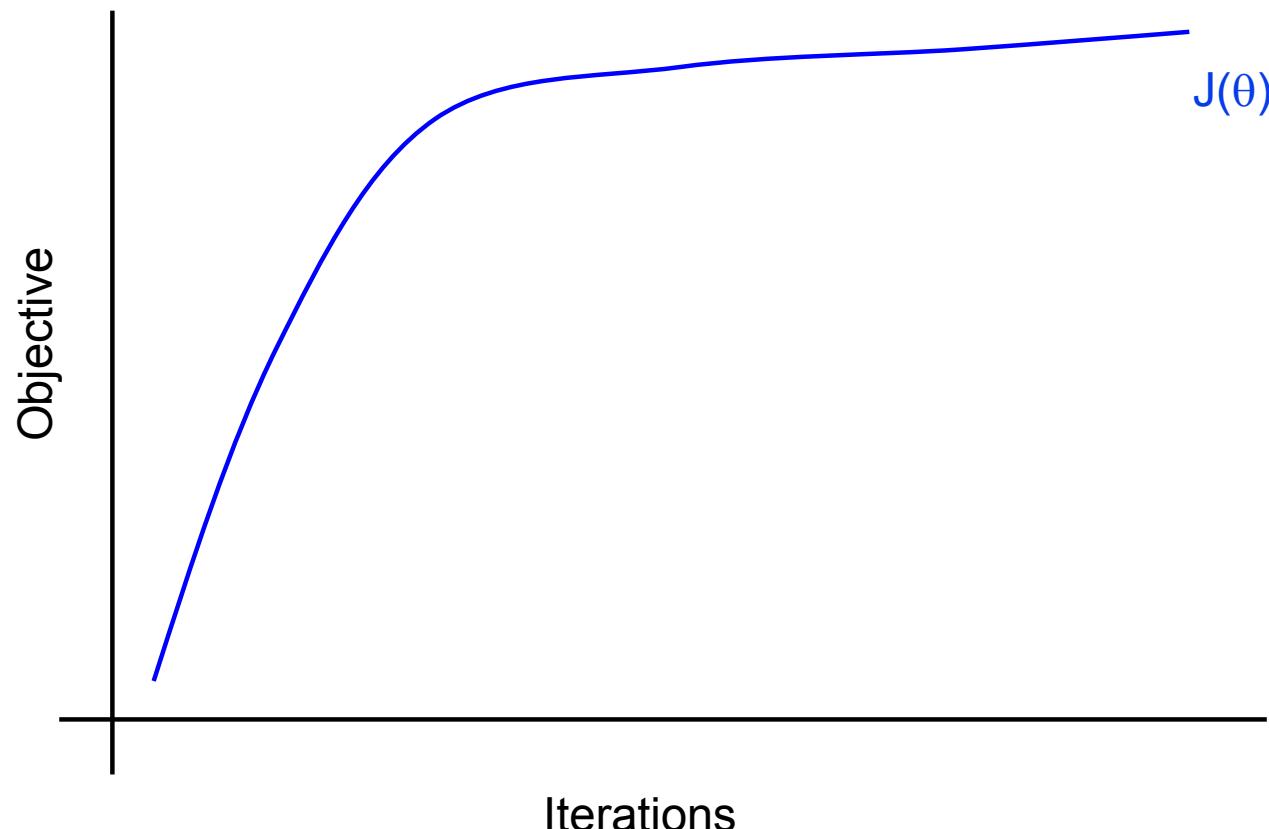
# Optimization algorithm diagnostics

- Bias vs. variance is one common diagnostic.
- For other problems, it's usually up to your own ingenuity to construct your own diagnostics to figure out what's wrong.
- Another example:
  - Bayesian logistic regression gets 2% error on spam, and 2% error on non-spam. (Unacceptably high error on non-spam.)
  - SVM using a linear kernel gets 10% error on spam, and 0.01% error on non-spam. (Acceptable performance.)
  - But you want to use logistic regression, because of computational efficiency, etc.
- What to do next?



# More diagnostics

- Other common questions:
  - Is the algorithm (gradient descent for logistic regression) converging?



It's often very hard to tell if an algorithm has converged yet by looking at the objective.



# More diagnostics

- Other common questions:
  - Is the algorithm (gradient descent for logistic regression) converging?
  - Are you optimizing the right function?
  - I.e., what you care about:

$$a(\theta) = \sum_i w^{(i)} \mathbf{1}\{h_\theta(x^{(i)}) = y^{(i)}\}$$

(weights  $w^{(i)}$  higher for non-spam than for spam).

- Bayesian logistic regression? Correct value for  $\lambda$ ?

$$\max_{\theta} J(\theta) = \sum_{i=1}^m \log p(y^{(i)}|x^{(i)}, \theta) - \lambda \|\theta\|^2$$

- SVM? Correct value for  $C$ ?

$$\begin{aligned} \min_{w,b} \quad & \|w\|^2 + C \sum_{i=1}^m \xi_i \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} - b) \geq 1 - \xi_i \end{aligned}$$



# Diagnostic

An SVM outperforms Bayesian logistic regression, but you really want to deploy Bayesian logistic regression for your application.

Let  $\theta_{\text{SVM}}$  be the parameters learned by an SVM.

Let  $\theta_{\text{BLR}}$  be the parameters learned by Bayesian logistic regression.

You care about weighted accuracy:

$$a(\theta) = \max_{\theta} \sum_i w^{(i)} \mathbb{1}\{h_{\theta}(x^{(i)}) = y^{(i)}\}$$

$\theta_{\text{SVM}}$  outperforms  $\theta_{\text{BLR}}$ . So:

$$a(\theta_{\text{SVM}}) > a(\theta_{\text{BLR}})$$

BLR tries to maximize:

$$J(\theta) = \sum_{i=1}^m \log p(y^{(i)}|x^{(i)}, \theta) - \lambda \|\theta\|^2$$

Diagnostic:

$$J(\theta_{\text{SVM}}) > J(\theta_{\text{BLR}})?$$



## Two cases

---

---

Case 1:  $a(\theta_{\text{SVM}}) > a(\theta_{\text{BLR}})$   
 $J(\theta_{\text{SVM}}) > J(\theta_{\text{BLR}})$

But BLR was trying to maximize  $J(\theta)$ . This means that  $\theta_{\text{BLR}}$  fails to maximize  $J$ , and the problem is with the convergence of the algorithm. **Problem is with optimization algorithm.**

Case 2:  $a(\theta_{\text{SVM}}) > a(\theta_{\text{BLR}})$   
 $J(\theta_{\text{SVM}}) \leq J(\theta_{\text{BLR}})$

This means that BLR succeeded at maximizing  $J(\theta)$ . But the SVM, which does worse on  $J(\theta)$ , actually does better on weighted accuracy  $a(\theta)$ .

This means that  $J(\theta)$  is the wrong function to be maximizing, if you care about  $a(\theta)$ .  
**Problem is with objective function of the maximization problem.**



# Diagnostics tell you what to try next

Bayesian logistic regression, implemented with gradient descent.

Fixes to try:

- Try getting more training examples. Fixes high variance.
- Try a smaller set of features. Fixes high variance.
- Try a larger set of features. Fixes high bias.
- Try email header features. Fixes high bias.
- Run gradient descent for more iterations. Fixes optimization algorithm.
- Try Newton's method. Fixes optimization algorithm.
- Use a different value for  $\lambda$ . Fixes optimization objective.
- Try using an SVM. Fixes optimization objective.



# More on diagnostics

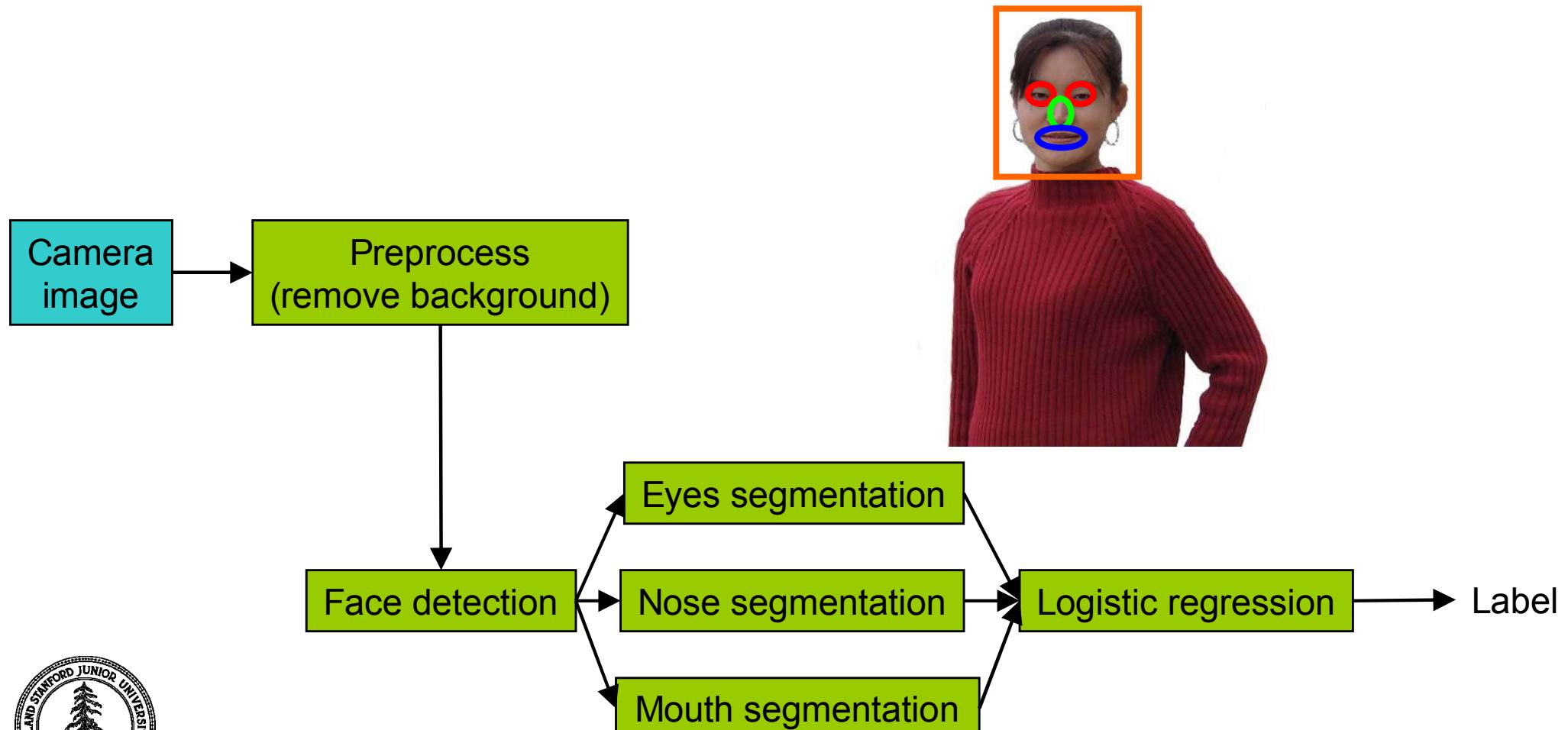
- Quite often, you'll need to come up with your own diagnostics to figure out what's happening in an algorithm.
- Even if a learning algorithm is working well, you might also run diagnostics to make sure you understand what's going on. This is useful for:
  - Understanding your application problem: If you're working on one important ML application for months/years, it's very valuable for you personally to get a intuitive understand of what works and what doesn't work in your problem.
  - Writing research papers: Diagnostics and error analysis help convey insight about the problem, and justify your research claims.
  - I.e., Rather than saying "Here's an algorithm that works," it's more interesting to say "Here's an algorithm that works because of component X, and here's my justification."
- Good machine learning practice: Error analysis. Try to understand what your sources of error are.



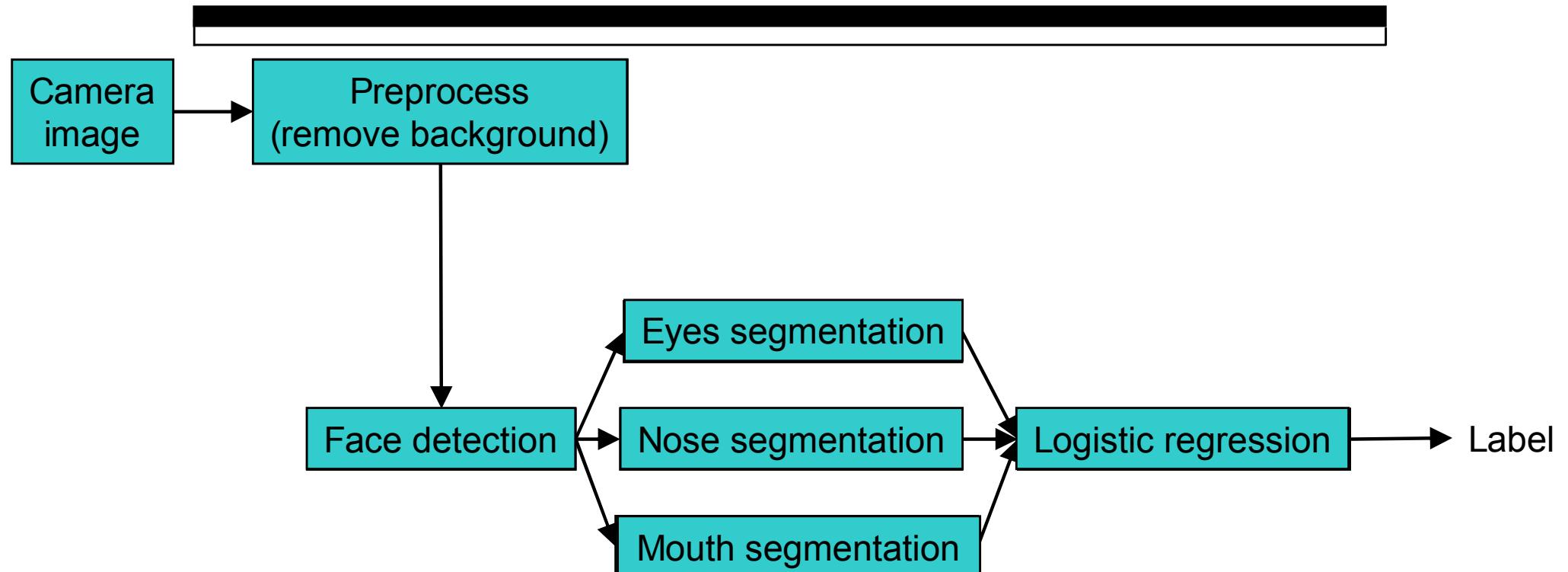
# Error Analysis

# Error analysis

Many applications combine many different learning components into a “pipeline.” E.g., Face recognition from images: [contrived example]



# Error analysis



How much error is attributable to each of the components?

Plug in ground-truth for each component, and see how accuracy changes.

Conclusion: Most room for improvement in face detection and eyes segmentation.

Component	Accuracy
Overall system	85%
Preprocess (remove background)	85.1%
Face detection	91%
Eyes segmentation	95%
Nose segmentation	96%
Mouth segmentation	97%
Logistic regression	100%

# Ablative analysis

Error analysis tries to explain the difference between current performance and perfect performance.

Ablative analysis tries to explain the difference between some baseline (much poorer) performance and current performance.

E.g., Suppose that you've build a good anti-spam classifier by adding lots of clever features to logistic regression:

- Spelling correction.
- Sender host features.
- Email header features.
- Email text parser features.
- Javascript parser.
- Features from embedded images.

Question: How much did each of these components really help?



# Ablative analysis

Simple logistic regression without any clever features get 94% performance.

Just what accounts for your improvement from 94 to 99.9%?

Ablative analysis: Remove components from your system one at a time, to see how it breaks.

Component	Accuracy
Overall system	99.9%
Spelling correction	99.0
Sender host features	98.9%
Email header features	98.9%
Email text parser features	95%
Javascript parser	94.5%
Features from images	94.0%

[baseline]

Conclusion: The email text parser features account for most of the improvement.

# **Getting started on a learning problem**

# Getting started on a problem

Approach #1: Careful design.

- Spend a long term designing exactly the right features, collecting the right dataset, and designing the right algorithmic architecture.
- Implement it and hope it works.
- **Benefit:** Nicer, perhaps more scalable algorithms. May come up with new, elegant, learning algorithms; contribute to basic research in machine learning.

Approach #2: Build-and-fix.

- Implement something quick-and-dirty.
- Run error analyses and diagnostics to see what's wrong with it, and fix its errors.
- **Benefit:** Will often get your application problem working more quickly. Faster time to market.



# Getting started on a problem

Approach #1: Careful design.

- Spend a long term designing exactly the right features, collecting the right dataset, and designing the right algorithmic architecture.
- Implement it and hope it works.
- **Benefit:** Nicer, perhaps more scalable algorithms. May come up with new, elegant, learning algorithms; contribute to basic research in machine learning.

Approach #2: Build-and-fix.

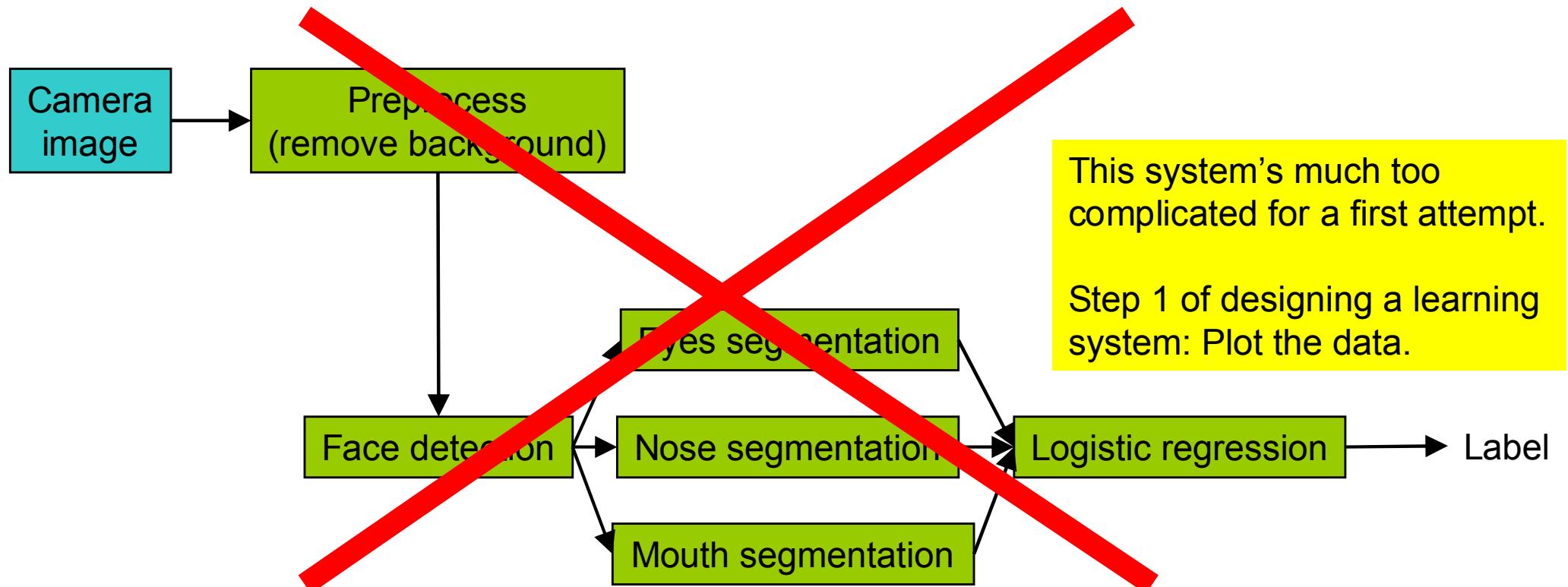
- Implement something quick-and-dirty.
- Run error analyses and diagnostics to see what's wrong with it, and fix its errors.
- **Benefit:** Will often get your application problem working more quickly. Faster time to market.

**"A well running ML system is a rewritten poorly running ML system."** - Chris Re, Stanford



# Premature statistical optimization

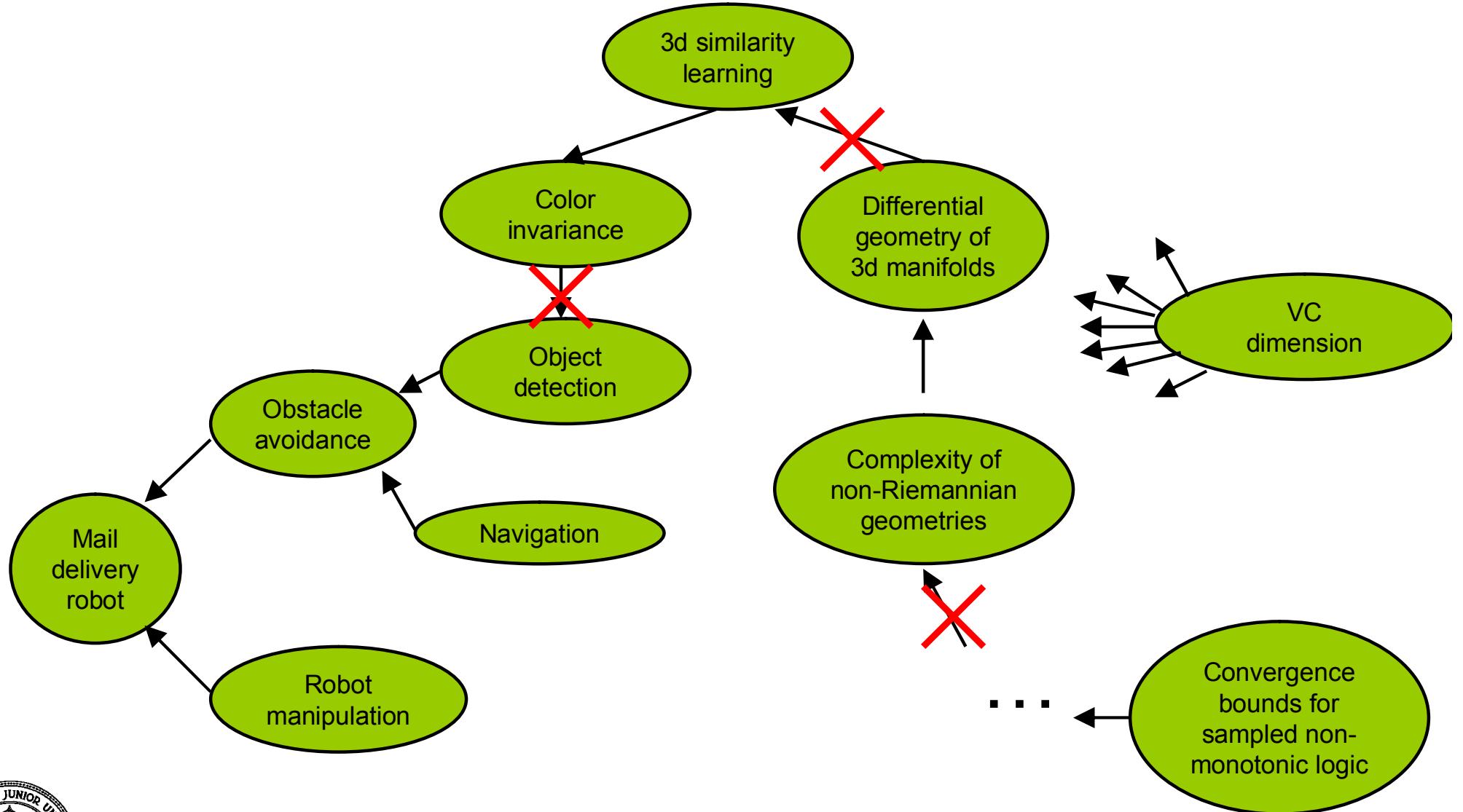
Very often, it's not clear what parts of a system are easy or difficult to build, and which parts you need to spend lots of time focusing on. E.g.,



The only way to find out what needs work is to implement something quickly, and find out what parts break.

[But this may be bad advice if your goal is to come up with new machine learning algorithms.]

# The danger of over-theorizing



[Based on Papadimitriou, 1995]

Andrew Y. Ng

# Summary

---

- Time spent coming up with diagnostics for learning algorithms is time well-spent.
- It's often up to your own ingenuity to come up with right diagnostics.
- Error analyses and ablative analyses also give insight into the problem.
- Two approaches to applying learning algorithms:
  - Design very carefully, then implement.
    - Risk of premature (statistical) optimization.
  - Build a quick-and-dirty prototype, diagnose, and fix.



Also recommended: ML advice by Christopher Ré

[http://cs229.stanford.edu/notes2020spring/ML\\_Advice\\_lecture.pdf](http://cs229.stanford.edu/notes2020spring/ML_Advice_lecture.pdf)

## 7 Steps of ML Systems.



- Step 1: Acquire Data
- Step 2: Look at your data\* -- after every step.
- Step 3: Create train/dev/test splits
- Step 4: Create/Refine a specification
- Step 5: Build model (simplest that works!)
- Step 6: Measurement
- Step 7: Repeat.

# Hidden Technical Debt in Machine Learning Systems

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips  
{dsculley, gholt, dg, edavydov, toddphillips}@google.com  
Google, Inc.

Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, Dan Dennison  
{ebner, vchaudhary, mwyoung, jfcrespo, dennison}@google.com  
Google, Inc.

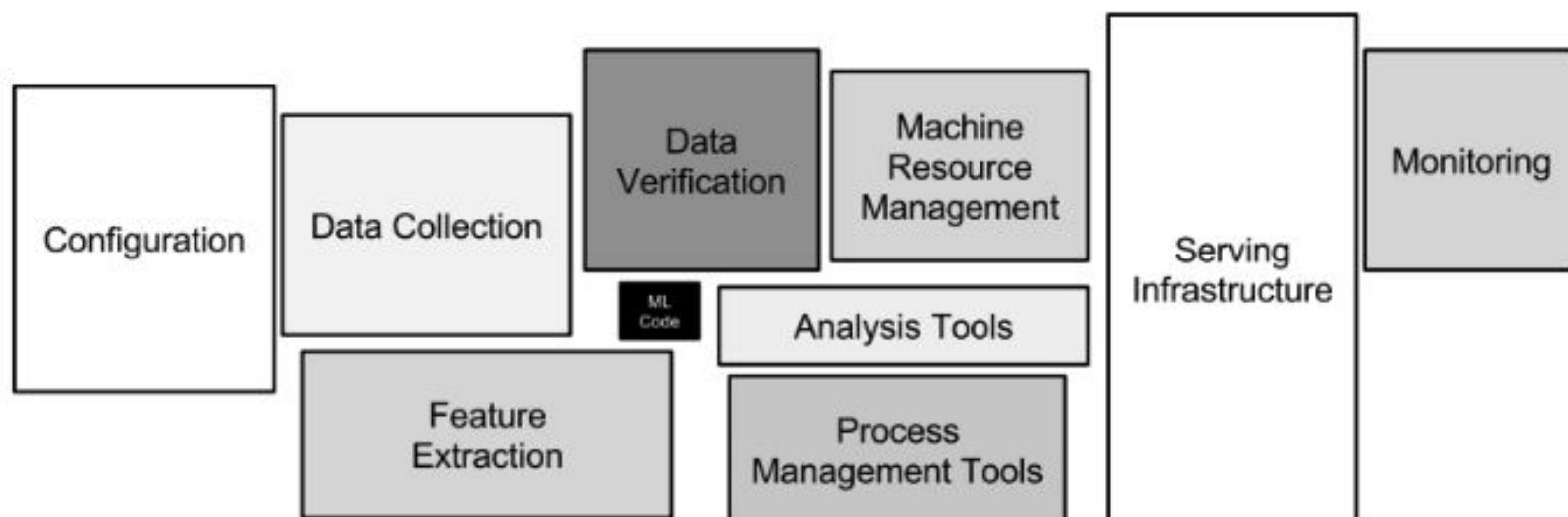


Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

# NEWS

[Home](#) | [Coronavirus](#) | [Video](#) | [World](#) | [Asia](#) | [UK](#) | [Business](#) | [Tech](#) | [Science](#) | [Stories](#) | [Entertainment & Arts](#)

Tech

## Stephen Hawking warns artificial intelligence could end mankind

By Rory Cellan-Jones  
Technology correspondent

① 2 December 2014 | [Comments](#)



Stephen Hawking: "Humans, who are limited by slow biological evolution, couldn't compete and would be superseded"

Borrowed from Michael Wooldridge, AAAI 2021 talk.

The choice *to communicate about AI/ML or not* depend on **your worldview**.

In public communication, “AI” is often conflated with “machine learning” (ML), a subfield of artificial intelligence.

There has been real progress in ML, driven by scientific advances (e.g. the “deep” in “deep learning”), larger datasets, cheap and easily available compute power.

These advances are narrow, and does not deliver “grand dream” of AI.

Don’t be distracted by media hype, AI advances are real and exciting, but they aren’t magic.

Long term speculative concerns shouldn’t distract us from near-term worries (bias, brittleness, employment, climate ... )

# Google 'fixed' its racist algorithm by removing gorillas from its image-labeling tech

51 ▾

Nearly three years after the company was called out, it hasn't gone beyond a quick workaround

By James Vincent | Jan 12, 2018, 10:35am EST

<https://www.theverge.com/2018/1/12/16882408/google-racist-gorillas-photo-recognition-algorithm-ai>

f t  SHARE



The AI algorithms in Google Photos sort images by a number of categories. | Photo by Vjeran Pavic / The Verge

Back in 2015, software engineer Jacky Alciné [pointed out](#) that the image recognition

AD

  
**verge  
deals**

Subscribe to get the best Verge-approved tech deals of the week.

Email (required)

By signing up, you agree to our [Privacy Notice](#) and European users agree to the data transfer



10 MINUTES AGO

30 MINUTES AGO

RETAIL

OCTOBER 11, 2018 / 10:04 AM / UPDATED 3 YEARS AGO

## Amazon scraps secret AI recruiting tool that showed bias against women

By Jeffrey Dastin



SAN FRANCISCO (Reuters) - Amazon.com Inc's [AMZN.O](#) machine-learning specialists uncovered a big problem: their new recruiting engine did not like women.

The team had been building computer programs since 2014 to review job applicants' resumes with the aim of mechanizing the search for top talent, five people familiar with the effort told Reuters.

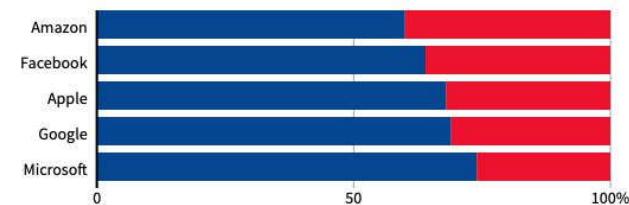
<https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G>

### Dominated by men

Top U.S. tech companies have yet to close the gender gap in hiring, a disparity most pronounced among technical staff such as software developers where men far outnumber women. Amazon's experimental recruiting engine followed the same pattern, learning to penalize resumes including the word "women's" until the company discovered the problem.

#### GLOBAL HEADCOUNT

■ Male ■ Female



#### EMPLOYEES IN TECHNICAL ROLES

■ Apple

■ Facebook

■ Google

■ Microsoft



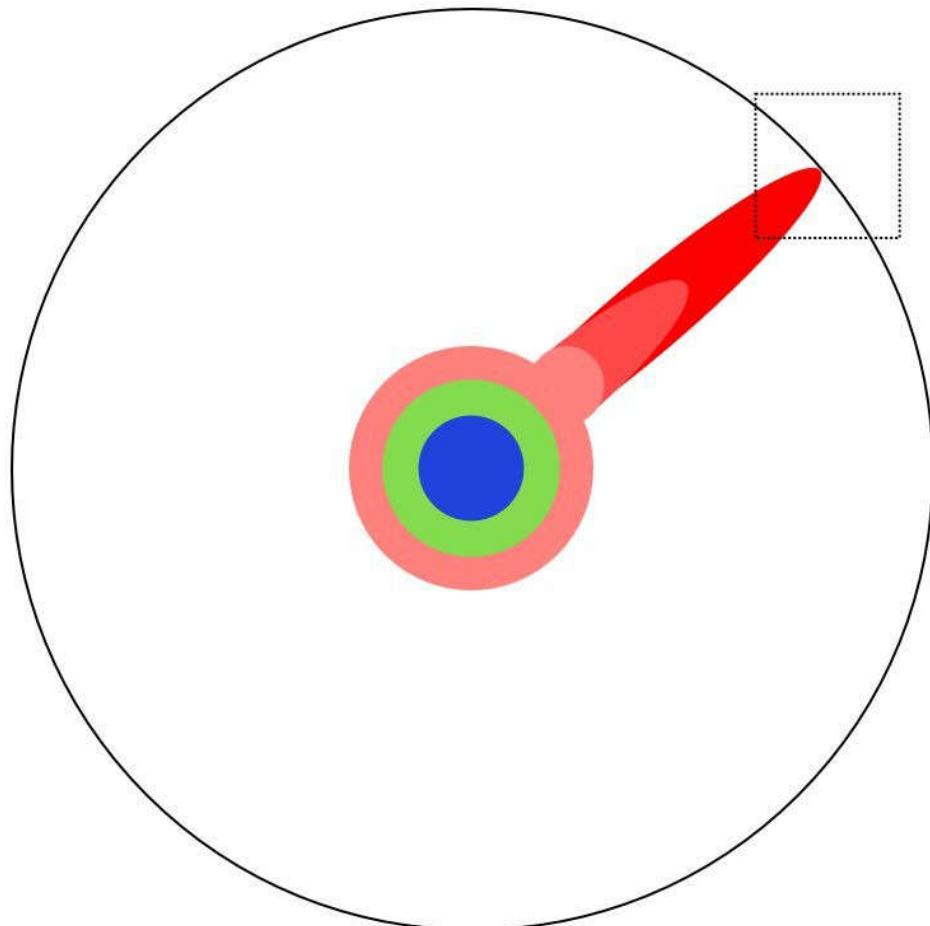
Note: Amazon does not disclose the gender breakdown of its technical workforce.

Source: Latest data available from the companies, since 2017.

By Han Huang | REUTERS GRAPHICS

Lessons from history abound: crash test dummies, mice models, clinical studies that under-represent minorities.

# Keep pushing



The Illustrated Guide to a Ph.D. by [Matt Might](#)  
<https://matt.might.net/articles/phd-school-in-pictures/>

