

# *Introduction to Statistical Machine Learning*

Christfried Webers

Statistical Machine Learning Group  
NICTA  
and  
College of Engineering and Computer Science  
The Australian National University

Canberra  
February – June 2012

(Many figures from C. M. Bishop, "Pattern Recognition and Machine Learning")



## *Outlines*

*Overview*  
*Introduction*  
*Linear Algebra*  
*Probability*  
*Linear Regression 1*  
*Linear Regression 2*  
*Linear Classification 1*  
*Linear Classification 2*  
*Neural Networks 1*  
*Neural Networks 2*  
*Kernel Methods*  
*Sparse Kernel Methods*  
*Graphical Models 1*  
*Graphical Models 2*  
*Graphical Models 3*  
*Mixture Models and EM 1*  
*Mixture Models and EM 2*  
*Approximate Inference*  
*Sampling*  
*Principal Component Analysis*  
*Sequential Data 1*  
*Sequential Data 2*  
*Combining Models*  
*Selected Topics*  
*Discussion and Summary*



Outlines

- 1 Examples
- 2 What is common to this examples?
- 3 Definition
- 4 Related Fields
- 5 Some Basic Notation
- 6 Some Fundamental Types of Learning
- 7 Training Regimes
- 8 Journals, Conferences
- 9 Python
- 10 Elefant

Overview

Introduction

Linear Algebra

Probability

Linear Regression 1

Linear Regression 2

Linear Classification 1

Linear Classification 2

Neural Networks 1

Neural Networks 2

Kernel Methods

Sparse Kernel Methods

Graphical Models 1

Graphical Models 2

Graphical Models 3

Mixture Models and EM 1

Mixture Models and EM 2

Approximate Inference

Sampling

Principal Component Analysis

Sequential Data 1

Sequential Data 2

Combining Models

Selected Topics

Discussion and Summary



Outlines

[Overview](#)

[Introduction](#)

[Linear Algebra](#)

[Probability](#)

[Linear Regression 1](#)

[Linear Regression 2](#)

[Linear Classification 1](#)

[Linear Classification 2](#)

[Neural Networks 1](#)

[Neural Networks 2](#)

[Kernel Methods](#)

[Sparse Kernel Methods](#)

[Graphical Models 1](#)

[Graphical Models 2](#)

[Graphical Models 3](#)

[Mixture Models and EM 1](#)

[Mixture Models and EM 2](#)

[Approximate Inference](#)

[Sampling](#)

[Principal Component Analysis](#)

[Sequential Data 1](#)

[Sequential Data 2](#)

[Combining Models](#)

[Selected Topics](#)

[Discussion and Summary](#)

# Introduction

## 11 Polynomial Curve Fitting

## 12 Probability Theory

## 13 Probability Densities

## 14 Expectations and Covariances



# Linear Algebra

15 Basic Concepts

16 Linear Transformations

17 Trace

18 Inner Product

19 Projection

20 Rank, Determinant, Trace

21 Matrix Inverse

22 Eigenvectors

23 Singular Value Decomposition

24 Directional Derivative, Gradient

25 Books

## Outlines

[Overview](#)

[Introduction](#)

[Linear Algebra](#)

[Probability](#)

[Linear Regression 1](#)

[Linear Regression 2](#)

[Linear Classification 1](#)

[Linear Classification 2](#)

[Neural Networks 1](#)

[Neural Networks 2](#)

[Kernel Methods](#)

[Sparse Kernel Methods](#)

[Graphical Models 1](#)

[Graphical Models 2](#)

[Graphical Models 3](#)

[Mixture Models and EM 1](#)

[Mixture Models and EM 2](#)

[Approximate Inference](#)

[Sampling](#)

[Principal Component Analysis](#)

[Sequential Data 1](#)

[Sequential Data 2](#)

[Combining Models](#)

[Selected Topics](#)

[Discussion and Summary](#)



# Probability

26 Boxes with Apples and Oranges

27 Bayes' Theorem

28 Bayes' Probabilities

29 Probability Distributions

30 Gaussian Distribution over a Vector

31 Decision Theory

32 Model Selection - Key Ideas

## Outlines

[Overview](#)

[Introduction](#)

[Linear Algebra](#)

[Probability](#)

[Linear Regression 1](#)

[Linear Regression 2](#)

[Linear Classification 1](#)

[Linear Classification 2](#)

[Neural Networks 1](#)

[Neural Networks 2](#)

[Kernel Methods](#)

[Sparse Kernel Methods](#)

[Graphical Models 1](#)

[Graphical Models 2](#)

[Graphical Models 3](#)

[Mixture Models and EM 1](#)

[Mixture Models and EM 2](#)

[Approximate Inference](#)

[Sampling](#)

[Principal Component Analysis](#)

[Sequential Data 1](#)

[Sequential Data 2](#)

[Combining Models](#)

[Selected Topics](#)

[Discussion and Summary](#)



# Linear Regression 1

33 Linear Basis Function Models

34 Maximum Likelihood and Least Squares

35 Geometry of Least Squares

36 Sequential Learning

37 Regularized Least Squares

38 Multiple Outputs

39 Loss Function for Regression

40 The Bias-Variance Decomposition

## Outlines

Overview

Introduction

Linear Algebra

Probability

Linear Regression 1

Linear Regression 2

Linear Classification 1

Linear Classification 2

Neural Networks 1

Neural Networks 2

Kernel Methods

Sparse Kernel Methods

Graphical Models 1

Graphical Models 2

Graphical Models 3

Mixture Models and EM 1

Mixture Models and EM 2

Approximate Inference

Sampling

Principal Component Analysis

Sequential Data 1

Sequential Data 2

Combining Models

Selected Topics

Discussion and Summary



# Linear Regression 2

41 Bayesian Regression

42 Sequential Update of the Posterior

43 Predictive Distribution

44 Proof of the Predictive Distribution

45 Predictive Distribution with Simplified Prior

46 Limitations of Linear Basis Function Models

## Outlines

Overview

Introduction

Linear Algebra

Probability

Linear Regression 1

Linear Regression 2

Linear Classification 1

Linear Classification 2

Neural Networks 1

Neural Networks 2

Kernel Methods

Sparse Kernel Methods

Graphical Models 1

Graphical Models 2

Graphical Models 3

Mixture Models and EM 1

Mixture Models and EM 2

Approximate Inference

Sampling

Principal Component Analysis

Sequential Data 1

Sequential Data 2

Combining Models

Selected Topics

Discussion and Summary



## Outlines

[Overview](#)  
[Introduction](#)  
[Linear Algebra](#)  
[Probability](#)  
[Linear Regression 1](#)  
[Linear Regression 2](#)  
[Linear Classification 1](#)  
[Linear Classification 2](#)  
[Neural Networks 1](#)  
[Neural Networks 2](#)  
[Kernel Methods](#)  
[Sparse Kernel Methods](#)  
[Graphical Models 1](#)  
[Graphical Models 2](#)  
[Graphical Models 3](#)  
[Mixture Models and EM 1](#)  
[Mixture Models and EM 2](#)  
[Approximate Inference](#)  
[Sampling](#)  
[Principal Component Analysis](#)  
[Sequential Data 1](#)  
[Sequential Data 2](#)  
[Combining Models](#)  
[Selected Topics](#)  
[Discussion and Summary](#)

# Linear Classification 1

47 *Classification*

48 *Generalised Linear Model*

49 *Inference and Decision*

50 *Discriminant Functions*

51 *Fisher's Linear Discriminant*

52 *The Perceptron Algorithm*



# Linear Classification 2

53 Probabilistic Generative Models

54 Continuous Input

55 Discrete Features

56 Probabilistic Discriminative Models

57 Logistic Regression

58 Iterative Reweighted Least Squares

59 Laplace Approximation

60 Bayesian Logistic Regression

## Outlines

Overview

Introduction

Linear Algebra

Probability

Linear Regression 1

Linear Regression 2

Linear Classification 1

Linear Classification 2

Neural Networks 1

Neural Networks 2

Kernel Methods

Sparse Kernel Methods

Graphical Models 1

Graphical Models 2

Graphical Models 3

Mixture Models and EM 1

Mixture Models and EM 2

Approximate Inference

Sampling

Principal Component Analysis

Sequential Data 1

Sequential Data 2

Combining Models

Selected Topics

Discussion and Summary



## 61 Neural Networks

## 62 Weight-space Symmetries

## 63 Parameter Optimisation

## 64 Gradient Descent Optimisation

### Outlines

*Overview*

*Introduction*

*Linear Algebra*

*Probability*

*Linear Regression 1*

*Linear Regression 2*

*Linear Classification 1*

*Linear Classification 2*

*Neural Networks 1*

*Neural Networks 2*

*Kernel Methods*

*Sparse Kernel Methods*

*Graphical Models 1*

*Graphical Models 2*

*Graphical Models 3*

*Mixture Models and EM 1*

*Mixture Models and EM 2*

*Approximate Inference*

*Sampling*

*Principal Component Analysis*

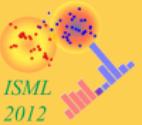
*Sequential Data 1*

*Sequential Data 2*

*Combining Models*

*Selected Topics*

*Discussion and Summary*



## 65 Error Backpropagation

### Outlines

[Overview](#)

[Introduction](#)

[Linear Algebra](#)

[Probability](#)

[Linear Regression 1](#)

[Linear Regression 2](#)

[Linear Classification 1](#)

[Linear Classification 2](#)

[Neural Networks 1](#)

[Neural Networks 2](#)

[Kernel Methods](#)

[Sparse Kernel Methods](#)

[Graphical Models 1](#)

[Graphical Models 2](#)

[Graphical Models 3](#)

[Mixture Models and EM 1](#)

[Mixture Models and EM 2](#)

[Approximate Inference](#)

[Sampling](#)

[Principal Component Analysis](#)

[Sequential Data 1](#)

[Sequential Data 2](#)

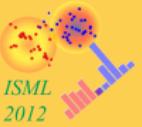
[Combining Models](#)

[Selected Topics](#)

[Discussion and Summary](#)

## 66 Regularisation in Neural Networks

## 67 Bayesian Neural Networks



# Kernel Methods

## 68 Nonparametric Probability Density Estimation

## 69 The Role of Training Data

## 70 Dual Representations

## 71 Kernels

## 72 Lagrange Multipliers

### Outlines

[Overview](#)

[Introduction](#)

[Linear Algebra](#)

[Probability](#)

[Linear Regression 1](#)

[Linear Regression 2](#)

[Linear Classification 1](#)

[Linear Classification 2](#)

[Neural Networks 1](#)

[Neural Networks 2](#)

[Kernel Methods](#)

[Sparse Kernel Methods](#)

[Graphical Models 1](#)

[Graphical Models 2](#)

[Graphical Models 3](#)

[Mixture Models and EM 1](#)

[Mixture Models and EM 2](#)

[Approximate Inference](#)

[Sampling](#)

[Principal Component Analysis](#)

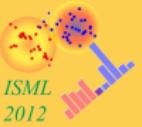
[Sequential Data 1](#)

[Sequential Data 2](#)

[Combining Models](#)

[Selected Topics](#)

[Discussion and Summary](#)



# Sparse Kernel Methods

## 73 Sparse Kernel Machines

## 74 Maximum Margin Classifiers

## 75 Overlapping Class Distributions

## 76 Relevance Vector Machines - Overview

### Outlines

*Overview*

*Introduction*

*Linear Algebra*

*Probability*

*Linear Regression 1*

*Linear Regression 2*

*Linear Classification 1*

*Linear Classification 2*

*Neural Networks 1*

*Neural Networks 2*

*Kernel Methods*

*Sparse Kernel Methods*

*Graphical Models 1*

*Graphical Models 2*

*Graphical Models 3*

*Mixture Models and EM 1*

*Mixture Models and EM 2*

*Approximate Inference*

*Sampling*

*Principal Component Analysis*

*Sequential Data 1*

*Sequential Data 2*

*Combining Models*

*Selected Topics*

*Discussion and Summary*

# Graphical Models 1

Introduction to Statistical  
Machine Learning

©2012

Christfried Webers  
NICTA

The Australian National  
University



## 77 Motivation

## 78 Bayesian Network

## 79 Plate Notation

## 80 Conditional Independence

Outlines

[Overview](#)

[Introduction](#)

[Linear Algebra](#)

[Probability](#)

[Linear Regression 1](#)

[Linear Regression 2](#)

[Linear Classification 1](#)

[Linear Classification 2](#)

[Neural Networks 1](#)

[Neural Networks 2](#)

[Kernel Methods](#)

[Sparse Kernel Methods](#)

[Graphical Models 1](#)

[Graphical Models 2](#)

[Graphical Models 3](#)

[Mixture Models and EM 1](#)

[Mixture Models and EM 2](#)

[Approximate Inference](#)

[Sampling](#)

[Principal Component Analysis](#)

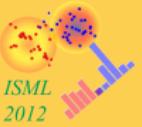
[Sequential Data 1](#)

[Sequential Data 2](#)

[Combining Models](#)

[Selected Topics](#)

[Discussion and Summary](#)



## Outlines

[Overview](#)  
[Introduction](#)  
[Linear Algebra](#)  
[Probability](#)  
[Linear Regression 1](#)  
[Linear Regression 2](#)  
[Linear Classification 1](#)  
[Linear Classification 2](#)  
[Neural Networks 1](#)  
[Neural Networks 2](#)  
[Kernel Methods](#)  
[Sparse Kernel Methods](#)  
[Graphical Models 1](#)  
[Graphical Models 2](#)  
[Graphical Models 3](#)  
[Mixture Models and EM 1](#)  
[Mixture Models and EM 2](#)  
[Approximate Inference](#)  
[Sampling](#)  
[Principal Component Analysis](#)  
[Sequential Data 1](#)  
[Sequential Data 2](#)  
[Combining Models](#)  
[Selected Topics](#)  
[Discussion and Summary](#)

# Graphical Models 2

## 81 Markov Random Fields

## 82 Bayesian Networks vs. Markov Random Fields

# Graphical Models 3

Introduction to Statistical  
Machine Learning

©2012

Christfried Webers  
NICTA

The Australian National  
University



ISML  
2012

## 83 Factor Graphs

## 84 The Sum-Product Algorithm

## 85 Similar Algorithms

## 86 Learning the Graph Structure

Outlines

Overview

Introduction

Linear Algebra

Probability

Linear Regression 1

Linear Regression 2

Linear Classification 1

Linear Classification 2

Neural Networks 1

Neural Networks 2

Kernel Methods

Sparse Kernel Methods

Graphical Models 1

Graphical Models 2

Graphical Models 3

Mixture Models and EM 1

Mixture Models and EM 2

Approximate Inference

Sampling

Principal Component Analysis

Sequential Data 1

Sequential Data 2

Combining Models

Selected Topics

Discussion and Summary

# Mixture Models and EM 1

Introduction to Statistical  
Machine Learning

©2012

Christfried Webers  
NICTA

The Australian National  
University



## 87 Motivation

## 88 K-means Clustering

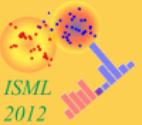
## 89 K-means Applications

## 90 Mixture of Gaussians

### Outlines

[Overview](#)  
[Introduction](#)  
[Linear Algebra](#)  
[Probability](#)  
[Linear Regression 1](#)  
[Linear Regression 2](#)  
[Linear Classification 1](#)  
[Linear Classification 2](#)  
[Neural Networks 1](#)  
[Neural Networks 2](#)  
[Kernel Methods](#)  
[Sparse Kernel Methods](#)  
[Graphical Models 1](#)  
[Graphical Models 2](#)  
[Graphical Models 3](#)  
[\*\*Mixture Models and EM 1\*\*](#)  
[Mixture Models and EM 2](#)  
[Approximate Inference](#)  
[Sampling](#)  
[Principal Component Analysis](#)  
[Sequential Data 1](#)  
[Sequential Data 2](#)  
[Combining Models](#)  
[Selected Topics](#)  
[Discussion and Summary](#)

# Mixture Models and EM 2



## 91 EM for Gaussian Mixtures

## 92 EM for Gaussian Mixtures - Relation to K-Means

## 93 Mixture of Bernoulli Distributions

## 94 EM for Gaussian Mixtures - Latent Variables

## 95 Convergence of EM

### Outlines

[Overview](#)

[Introduction](#)

[Linear Algebra](#)

[Probability](#)

[Linear Regression 1](#)

[Linear Regression 2](#)

[Linear Classification 1](#)

[Linear Classification 2](#)

[Neural Networks 1](#)

[Neural Networks 2](#)

[Kernel Methods](#)

[Sparse Kernel Methods](#)

[Graphical Models 1](#)

[Graphical Models 2](#)

[Graphical Models 3](#)

[Mixture Models and EM 1](#)

[Mixture Models and EM 2](#)

[Approximate Inference](#)

[Sampling](#)

[Principal Component Analysis](#)

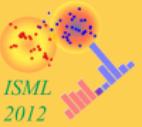
[Sequential Data 1](#)

[Sequential Data 2](#)

[Combining Models](#)

[Selected Topics](#)

[Discussion and Summary](#)



# Approximate Inference

96 Approximate Inference

97 Approximation Schemes

98 Variational Optimisation

99 Calculus of Variation

100 Variational Optimisation applied to Inference

101 Exponential Family

102 Expectation Propagation

## Outlines

Overview

Introduction

Linear Algebra

Probability

Linear Regression 1

Linear Regression 2

Linear Classification 1

Linear Classification 2

Neural Networks 1

Neural Networks 2

Kernel Methods

Sparse Kernel Methods

Graphical Models 1

Graphical Models 2

Graphical Models 3

Mixture Models and EM 1

Mixture Models and EM 2

Approximate Inference

Sampling

Principal Component Analysis

Sequential Data 1

Sequential Data 2

Combining Models

Selected Topics

Discussion and Summary



# Sampling

103 Motivation

104 Sampling from the Uniform Distribution

105 Sampling from Standard Distributions

106 Rejection Sampling

107 Adaptive Rejection Sampling

108 Importance Sampling

109 Markov Chain Monte Carlo - The Idea

110 Gibbs Sampling

## Outlines

Overview

Introduction

Linear Algebra

Probability

Linear Regression 1

Linear Regression 2

Linear Classification 1

Linear Classification 2

Neural Networks 1

Neural Networks 2

Kernel Methods

Sparse Kernel Methods

Graphical Models 1

Graphical Models 2

Graphical Models 3

Mixture Models and EM 1

Mixture Models and EM 2

Approximate Inference

Sampling

Principal Component Analysis

Sequential Data 1

Sequential Data 2

Combining Models

Selected Topics

Discussion and Summary

# Principal Component Analysis

Introduction to Statistical  
Machine Learning

©2012

Christfried Webers  
NICTA

The Australian National  
University



## III Motivation

## III Principal Component Analysis (PCA)

### Outlines

[Overview](#)

[Introduction](#)

[Linear Algebra](#)

[Probability](#)

[Linear Regression 1](#)

[Linear Regression 2](#)

[Linear Classification 1](#)

[Linear Classification 2](#)

[Neural Networks 1](#)

[Neural Networks 2](#)

[Kernel Methods](#)

[Sparse Kernel Methods](#)

[Graphical Models 1](#)

[Graphical Models 2](#)

[Graphical Models 3](#)

[Mixture Models and EM 1](#)

[Mixture Models and EM 2](#)

[Approximate Inference](#)

[Sampling](#)

[Principal Component Analysis](#)

[Sequential Data 1](#)

[Sequential Data 2](#)

[Combining Models](#)

[Selected Topics](#)

[Discussion and Summary](#)



Outlines

- [Overview](#)
- [Introduction](#)
- [Linear Algebra](#)
- [Probability](#)
- [Linear Regression 1](#)
- [Linear Regression 2](#)
- [Linear Classification 1](#)
- [Linear Classification 2](#)
- [Neural Networks 1](#)
- [Neural Networks 2](#)
- [Kernel Methods](#)
- [Sparse Kernel Methods](#)
- [Graphical Models 1](#)
- [Graphical Models 2](#)
- [Graphical Models 3](#)
- [Mixture Models and EM 1](#)
- [Mixture Models and EM 2](#)
- [Approximate Inference](#)
- [Sampling](#)
- [Principal Component Analysis](#)
- [Sequential Data 1](#)
- [Sequential Data 2](#)
- [Combining Models](#)
- [Selected Topics](#)
- [Discussion and Summary](#)

# Sequential Data 1

I13 Motivation

I14 Stationary versus Nonstationary

I15 Markov Model

I16 State Space Model

I17 Hidden Markov Model

I18 HMM - Generative View

I19 HMM - Handwritten Digits



# Sequential Data 2

I20 A Simple Example

I21 Maximum Likelihood for HMM

I22 Forward-Backward HMM

I23 Conditional Independence

I24 Alpha-Beta HMM

I25 How to train a HMM using EM?

I26 HMM - Viterbi Algorithm

## Outlines

Overview

Introduction

Linear Algebra

Probability

Linear Regression 1

Linear Regression 2

Linear Classification 1

Linear Classification 2

Neural Networks 1

Neural Networks 2

Kernel Methods

Sparse Kernel Methods

Graphical Models 1

Graphical Models 2

Graphical Models 3

Mixture Models and EM 1

Mixture Models and EM 2

Approximate Inference

Sampling

Principal Component Analysis

Sequential Data 1

Sequential Data 2

Combining Models

Selected Topics

Discussion and Summary



# Combining Models

I27 Motivation

I28 Model Combination vs. Bayesian Model Averaging

I29 Committees

I30 Boosting

I31 Tree-based Models

I32 Conditional Mixture Models

## Outlines

Overview

Introduction

Linear Algebra

Probability

Linear Regression 1

Linear Regression 2

Linear Classification 1

Linear Classification 2

Neural Networks 1

Neural Networks 2

Kernel Methods

Sparse Kernel Methods

Graphical Models 1

Graphical Models 2

Graphical Models 3

Mixture Models and EM 1

Mixture Models and EM 2

Approximate Inference

Sampling

Principal Component Analysis

Sequential Data 1

Sequential Data 2

Combining Models

Selected Topics

Discussion and Summary



# Selected Topics

133 Selected Topics

134 Occam's Razor

135 Reinforcement Learning

136 PageRank

137 Envelope Paradox

## Outlines

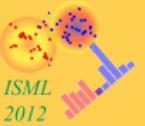
- [Overview](#)
- [Introduction](#)
- [Linear Algebra](#)
- [Probability](#)
- [Linear Regression 1](#)
- [Linear Regression 2](#)
- [Linear Classification 1](#)
- [Linear Classification 2](#)
- [Neural Networks 1](#)
- [Neural Networks 2](#)
- [Kernel Methods](#)
- [Sparse Kernel Methods](#)
- [Graphical Models 1](#)
- [Graphical Models 2](#)
- [Graphical Models 3](#)
- [Mixture Models and EM 1](#)
- [Mixture Models and EM 2](#)
- [Approximate Inference](#)
- [Sampling](#)
- [Principal Component Analysis](#)
- [Sequential Data 1](#)
- [Sequential Data 2](#)
- [Combining Models](#)
- [Selected Topics](#)
- [Discussion and Summary](#)



## Outlines

- 138 Tools
- 139 Models for Decision Problems
- 140 Learning
- 141 Linear Regression and Classification
- 142 Density Estimation
- 143 Kernels
- 144 Factorising Distributions
- 145 Non-Factorising Distributions
- 146 Dimensionality Reduction
- 147 Sequential Data
- 148 Combining Models
- 149 What we did not cover
- 150 From Bayes to HMM

Overview  
Introduction  
Linear Algebra  
Probability  
Linear Regression 1  
Linear Regression 2  
Linear Classification 1  
Linear Classification 2  
Neural Networks 1  
Neural Networks 2  
Kernel Methods  
Sparse Kernel Methods  
Graphical Models 1  
Graphical Models 2  
Graphical Models 3  
Mixture Models and EM 1  
Mixture Models and EM 2  
Approximate Inference  
Sampling  
Principal Component Analysis  
Sequential Data 1  
Sequential Data 2  
Combining Models  
Selected Topics  
Discussion and Summary



# Part I

## *Overview*

*Examples*

*What is common to this examples?*

*Definition*

*Related Fields*

*Some Basic Notation*

*Some Fundamental Types  
of Learning*

*Training Regimes*

*Journals, Conferences*

*Python*

*Elefant*

# Search Machine Ranking – 2009

Introduction to Statistical  
Machine Learning

- Given a number of web sites which match some search phrase:  
Learn which pages most of the users are looking for.



Web Images Maps News Shopping Gmail more ▾ Sign in

Google Introduction to Statistical Machine Learning Search Advanced Search Preferences

Web Results 1 - 10 of about 5,870,000 for [Introduction to Statistical Machine Learning](#). (0.22 seconds)

**Introduction to Statistical Machine Learning**  
15 May 2008 ... The other speakers will detail or built upon this [introduction](#). **Statistical machine learning** is concerned with the development of algorithms ...  
[videolectures.net/mlss08au\\_hutter\\_isml/](#) - 75k - [Cached](#) - [Similar pages](#)

**Statistical Machine Learning (SML) Group, NICTA**  
This course provides a broad but thorough [introduction](#) to the methods and practice of **statistical machine learning**. Topics covered will include Bayesian ...  
[smi.nicta.com.au/isml09.html](#) - 24k - [Cached](#) - [Similar pages](#)

[PDF] **An Introduction to Statistical Machine Learning - Introduction -**  
File Format: PDF/Adobe Acrobat - [View as HTML](#)  
**Statistical Machine Learning. - Introduction ..** Samy Bengio. [bengio@idiap.ch](mailto:bengio@idiap.ch). Dalle Molle Institute for Perceptual Artificial Intelligence (IDIAP) ...  
[bengio.abracadabrou.com/lectures/old/tex\\_intro.pdf](#) - [Similar pages](#)

Examples

What is common to this examples?

Definition

Related Fields

Some Basic Notation

Some Fundamental Types of Learning

Training Regimes

Journals, Conferences

Python

Elefant

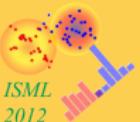
# Search Machine Ranking – 2010

Introduction to Statistical  
Machine Learning

©2012

Christfried Webers  
NICTA

The Australian National  
University



Google has learned

Google | Introduction to Statistical Machine Learning | Search | Advanced Search

Search:  the web  pages from Australia

[View customizations](#)

Web [Show options...](#) Results 1 - 10 of about 3,840,000 for **Introduction to Statistical Machine Learning**. (0.41 seconds)

**Introduction to Statistical Machine Learning** - Statistical Machine ... - 2:21am  
8 Jun 2009 ... This course provides a broad but thorough **introduction** to the methods and practice of **statistical machine learning**. ...  
[smi.nicta.com.au/isml09.html](http://smi.nicta.com.au/isml09.html) - [Cached](#) - [Similar](#) -

[PDF] **Introduction to Statistical Machine Learning**  
File Format: PDF/Adobe Acrobat  
**Machine Learning** c 2009. Christfried Webers. NICTA. The Australian National. University.  
1of 600. **Introduction to Statistical Machine Learning** ...  
[smi.nicta.com.au/isml09/lectures/20.pdf](http://smi.nicta.com.au/isml09/lectures/20.pdf) -

[Show more results from smi.nicta.com.au](#)

[PDF] **An Introduction to Statistical Machine Learning** - **Introduction** - 2:21am  
File Format: PDF/Adobe Acrobat - [Quick View](#)  
by S Bengio - [Related articles](#) - [All 5 versions](#)  
**Statistical Machine Learning** - **Introduction** - Samy Bengio [bengio@idiap.ch](mailto:bengio@idiap.ch). Dalle Molle Institute for Perceptual Artificial Intelligence (IDIAP) ...  
[bengio.abracadoucou.com/lectures/old/tex\\_intro.pdf](http://bengio.abracadoucou.com/lectures/old/tex_intro.pdf) - [Similar](#) -

## Examples

What is common to this examples?

Definition

Related Fields

Some Basic Notation

Some Fundamental Types of Learning

Training Regimes

Journals, Conferences

Python

Elefant

# Search Machine Ranking – 2012

Has Google learned more?

Google search results for "Introduction to Statistical Machine Learning".

Search results:

- Scholarly articles for Introduction to Statistical Machine Learning**
  - [Introduction to statistical relational learning](#) - Getoor - Cited by 486
  - [Introduction to statistical learning theory and support ...](#) - Xuegong - Cited by 329
- Introduction to statistical machine learning - Statistical Machine ...**
  - [smi.nicta.com.au/sml09.html](#)  
8 Jun 2009 – **Introduction to Statistical Machine Learning 2009** (ANU COMP6467/4670). Course Coordinators: Christfried Webers, Marcus Hutter ...
- [PDF] Introduction to Statistical Machine Learning - (SML) Group, NI ...**
  - [smi.nicta.com.au/sml09/lectures/25.pdf](#)  
File Format: PDF/Adobe Acrobat
  - Introduction to Statistical Machine Learning** c 2009. Christfried Webers, NICTA. The Australian National University. 1 of 748. **Introduction to Statistical Machine ...**
- [PDF] An Introduction to Statistical Machine Learning - EM for GMMs -**
  - [bengio.abracadoucoum.com/lectures/old/tex\\_gmm.pdf](#)  
File Format: PDF/Adobe Acrobat - Quick View
  - by S Bengio - **Cited by 3 - Related articles**
  - An Introduction to Statistical Machine Learning.** - EM for GMMs -, Samy Bengio bengio@idiap.ch. Dalle Molle Institute for Perceptual Artificial Intelligence (IDIAP) ...
- [PDF] An Introduction to Statistical Machine Learning - Introduction -**
  - [bengio.abracadoucoum.com/lectures/old/tex\\_intro.pdf](#)  
File Format: PDF/Adobe Acrobat - Quick View
  - by S Bengio - **Related articles**
  - An Introduction to Statistical Machine Learning.** - Introduction -, Samy Bengio bengio@idiap.ch. Dalle Molle Institute for Perceptual Artificial Intelligence (IDIAP) ...

## Examples

What is common to this examples?

Definition

Related Fields

Some Basic Notation

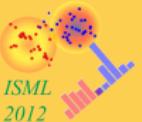
Some Fundamental Types of Learning

Training Regimes

Journals, Conferences

Python

Elefant



*Examples**What is common to this examples?**Definition**Related Fields**Some Basic Notation**Some Fundamental Types  
of Learning**Training Regimes**Journals, Conferences**Python**Elefant*

	From / To	Date Sent	Thread
2949	✉ T → Support	10/02/09 7:45 +0...	Message from eBay.com.au
2950	✉ T → Ken.Johnston	10/02/09 14:12 +...	Fool them once, fool them twice, fool...
2951	✉ T → christfried.web...	10/02/09 3:14 -0...	Assistance, Petersen
2952	✉ T → Air Sep	9/02/09 4:53 -0800	Un negocio de por vida 100% Renta...
2953	✉ T → Osita John	10/02/09 17:33 +...	Now Contact my secretary ask him fo...
2954	✉ T → Air Sep	9/02/09 0:38 -0800	Un negocio de por vida 100% Renta...
2955	✉ T → Air Sep	9/02/09 10:12 -0...	Un negocio de por vida 100% Renta...
2956	✉ T → MISS MERCY...	29/01/09 23:13 -...	Urgent Attention(YOUR FILE HAVE...
2957	✉ → PEPSI BOTTL...	25/07/08 11:23 -...	OEP00934/UK
2958	✉ → JOSEPH POON	11/02/09 12:04 +...	MY PROPOSAL!!!
2959	✉ → MADAM ERL...	11/02/09 13:41 +...	LOOKING FOR A TRUSTWORTHY...
2960	✉ T → REBECA RO...	11/02/09 18:48 +...	Dear sir/madam:
2961	✉ T → REBECA RO...	11/02/09 18:48 +...	Dear sir/madam:
2962	✉ T → Elinor Shannon	11/02/09 22:37 +...	I shall look forward to hearing from you
2963	✉ T → Air Sep	10/02/09 14:37 -...	Un negocio de por vida 100% Renta...
2964	✉ → Foreign Payme...	1/02/09 16:13 +0...	Goodday,
2965	✉ T → JANET KUEN	12/02/09 16:11 +...	Dear sir/madam:
2966	✉ → Abubakar Mar...	10/02/09 19:04 +...	OUR DEAR FRIEND
2967	✉ → JAMES ROBE...	12/02/09 23:12 -...	From James Roberts
2968	✉ T → Bases de Email...	13/02/09 10:50 -...	Nuevas Bases de Datos de Mexico
2969	✉ → Barrister Willi...	15/02/09 1:23 +0...	WILL AND TESTAMENT
2970	✉ → Isolde	15/02/09 9:45 -0...	A Valentine's Day Ecard Special Deli...
2971	✉ T → NTI eNews	15/02/09 12:25 -...	Super Sweet Deals From NTIus.com

- Given some examples what the user defined as junk mail.
- From these examples, learn to identify new incoming junk mail.



### Examples

What is common to this examples?

Definition

Related Fields

Some Basic Notation

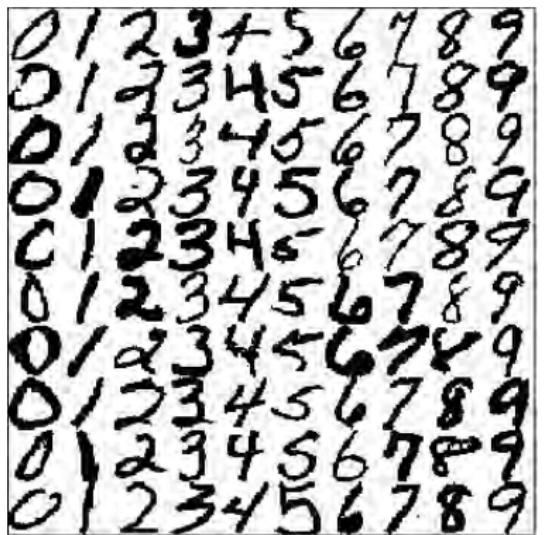
Some Fundamental Types  
of Learning

Training Regimes

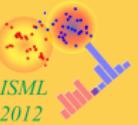
Journals, Conferences

Python

Elefant



- Given handwritten ZIP codes on letters, money amounts on cheques etc.
- Learn to recognise the correct digit written by hand.



- World best computer program TD-GAMMON (Tesauro 1992, 1995) played over a million games against itself.
- Plays now on the level of human world champion.

*Examples*

*What is common to this examples?*

*Definition*

*Related Fields*

*Some Basic Notation*

*Some Fundamental Types of Learning*

*Training Regimes*

*Journals, Conferences*

*Python*

*Elefant*



ISML  
2012

### Examples

What is common to this examples?

Definition

Related Fields

Some Basic Notation

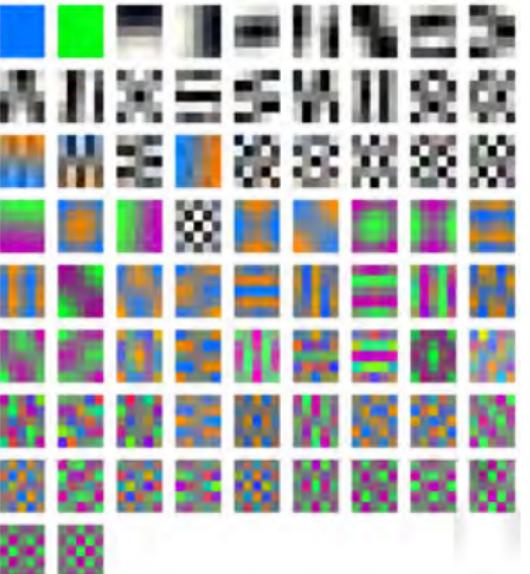
Some Fundamental Types  
of Learning

Training Regimes

Journals, Conferences

Python

Elefant



50 000 patches ( $5 \times 5 \times 3$ ) from the Berkeley Segmentation Database.  
McAuley et. al., "Learning High-Order MRF Priors of Color Images", ICML2006



# Image Denoising

2 Use this knowledge to denoise a **yet unseen** image

Original image



Noise added



Denoised



## Examples

What is common to these examples?

Definition

Related Fields

Some Basic Notation

Some Fundamental Types of Learning

Training Regimes

Journals, Conferences

Python

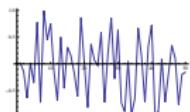
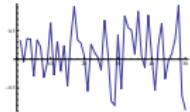
Elefant

McAuley et. al., "Learning High-Order MRF Priors of Color Images", ICML2006

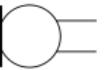
# Separating Audio Sources

Cocktail Party Problem (human brains may do it differently ;–)

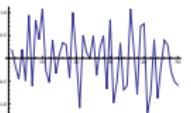
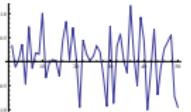
Audio Sources



Microphones



Audio Mixtures



## Examples

What is common to this examples?

Definition

Related Fields

Some Basic Notation

Some Fundamental Types  
of Learning

Training Regimes

Journals, Conferences

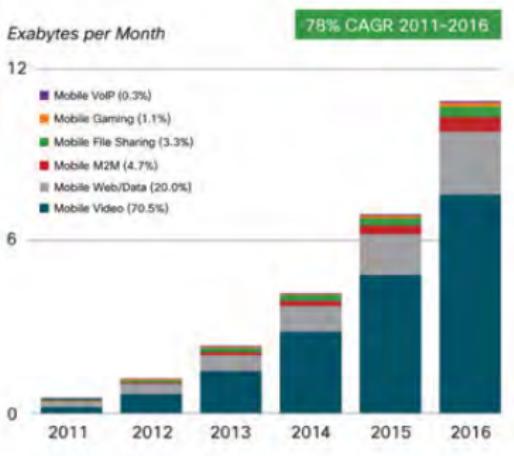
Python

Elefant

(J. Steinbauer et. al., <http://cnx.org/content/m15712/latest/>)

# Smart Mobile Content Delivery

- 70% of all traffic on mobile networks is video
- Streaming of video is awful because of congestions
- Preload content with spare network capacity
- Play from local storage
- Learn the user behaviour on the device (privacy!)



## Examples

What is common to this examples?

Definition

Related Fields

Some Basic Notation

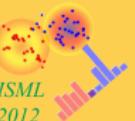
Some Fundamental Types of Learning

Training Regimes

Journals, Conferences

Python

Elefant



# Smart Mobile Content Delivery

- Trial on Android OS

- <http://www.incoming-media.com/>



## Examples

What is common to this examples?

Definition

Related Fields

Some Basic Notation

Some Fundamental Types  
of Learning

Training Regimes

Journals, Conferences

Python

Elefant



*Examples*

*What is common to this examples?*

*Definition*

*Related Fields*

*Some Basic Notation*

*Some Fundamental Types  
of Learning*

*Training Regimes*

*Journals, Conferences*

*Python*

*Elefant*

# *Other applications of Machine Learning*

- autonomous robotics,
- detecting credit card fraud,
- detecting network intrusion,
- bioinformatics,
- neuroscience,
- medical diagnosis,
- stock market analysis,
- playing games by self-play: Checker and Backgammon.



*Examples*

*What is common to this  
examples?*

*Definition*

*Related Fields*

*Some Basic Notation*

*Some Fundamental Types  
of Learning*

*Training Regimes*

*Journals, Conferences*

*Python*

*Elefant*

# What is common to this examples?

- ➊ Given some data (e.g. hand written digits).
- ➋ Possibly some extra information (e.g. which digit does this number represent)
- ➌ Goal: Built a machine which can learn from the given data utilising the extra information (if available).



Examples

What is common to this examples?

Definition

Related Fields

Some Basic Notation

Some Fundamental Types  
of Learning

Training Regimes

Journals, Conferences

Python

Elefant

# What is Machine Learning?

## Definition (First Try)

Machine learning is concerned with the design and development of algorithms that allow machines to learn.

- machines? computers? HAL?
- to learn?
- need to quantify "learning"
- to improve their performance over time

## Definition (Second Try)

Machine learning is concerned with the design and development of algorithms that allow computers (machines) to improve their performance over time.



Examples

What is common to this  
examples?

Definition

Related Fields

Some Basic Notation

Some Fundamental Types  
of Learning

Training Regimes

Journals, Conferences

Python

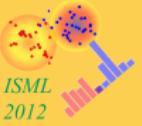
Elefant

# What is Machine Learning?

- What is the source of the improved performance?
- New insights by the algorithm designer?

## Definition (Final Version)

Machine learning is concerned with the design and development of algorithms that allow computers (machines) to improve their performance over time based on data.



# What is Machine Learning?

## Definition (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  $E$ .

Examples

What is common to these examples?

Definition

Related Fields

Some Basic Notation

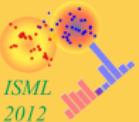
Some Fundamental Types of Learning

Training Regimes

Journals, Conferences

Python

Elefant



*Examples*

*What is common to this examples?*

*Definition*

*Related Fields*

*Some Basic Notation*

*Some Fundamental Types  
of Learning*

*Training Regimes*

*Journals, Conferences*

*Python*

*Elefant*

# What is the challenge?

- Given only some examples.
- Need to derive a relation for many more (possibly infinite) unseen examples.
- Occam's Razor (William Ockham, circa 1285 – 1349):  
“Plurality must never be posited without necessity”
- “The simplest explanation or strategy tends to be the best one.”
- By the way: Often cited “Entities should not be multiplied unnecessarily.” can not be found literally in Ockham’s works. Multiple versions ;-)



Examples

What is common to this  
examples?

Definition

Related Fields

Some Basic Notation

Some Fundamental Types  
of Learning

Training Regimes

Journals, Conferences

Python

Elefant

# 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)

# Why Machine Learning?



Machine Learning is essential when

- humans are unable to explain their expertise (speech recognition).
- humans are not around for help (navigation on Mars, underwater robotics).
- large amount of data with possible hidden relationships and correlations
- environment changes (fast) in time (mobile phone network).
- solutions need to be adapted to many particular cases (junk mail).

Example: It is easier to write a program that learns to play checkers or backgammon well by self-play rather than converting the expertise of a master player to a program.

Examples

What is common to this examples?

Definition

Related Fields

Some Basic Notation

Some Fundamental Types of Learning

Training Regimes

Journals, Conferences

Python

Elefant



- Artificial Intelligence - AI
- Statistics
- Game Theory
- Neuroscience, Psychology
- Data Mining
- Computer Science
- Adaptive Control Theory
- ...

*Examples*

*What is common to this examples?*

*Definition*

*Related Fields*

*Some Basic Notation*

*Some Fundamental Types  
of Learning*

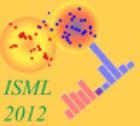
*Training Regimes*

*Journals, Conferences*

*Python*

*Elefant*

# Artificial Intelligence - AI



- Artificial intelligence is the intelligence of machines and the branch of computer science which aims to create it.
- The field was founded on the **claim** that human intelligence can be so precisely described that it can be simulated by a machine.
- Central areas: reasoning, knowledge, planning, **learning**, communication, perception and the ability to move and manipulate objects (autonomous robotics).
- Philosophical questions: Can a machine have a mind and consciousness? Are there limits to how intelligent machines can be?

Examples

What is common to this examples?

Definition

Related Fields

Some Basic Notation

Some Fundamental Types  
of Learning

Training Regimes

Journals, Conferences

Python

Elefant



Examples

What is common to this  
examples?

Definition

Related Fields

Some Basic Notation

Some Fundamental Types  
of Learning

Training Regimes

Journals, Conferences

Python

Elefant

- Descriptive Statistics: Summarize or describe a collection of data
- Inferential Statistics: Draw inferences about a process taking randomness and uncertainty into account
- What can be inferred from data and some modelling assumptions?
- How reliable are the results?



## Examples

What is common to this examples?

## Definition

## Related Fields

Some Basic Notation

Some Fundamental Types  
of Learning

Training Regimes

Journals, Conferences

Python

Elefant

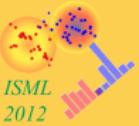
- How do animals learn?
- Modelling of human learning (e.g. Bayesian models of human inductive learning)
- increasing interaction between Statistical Machine Learning and Neuroscience, Psychology  
e.g. NIPS - Neural Information Processing Systems Conference 2009:
  - “Discriminative Network Models of Schizophrenia”,
  - “Functional network reorganization in motor cortex can be explained by reward-modulated Hebbian learning”,
  - “Canonical Time Warping for Alignment of Human Behavior”
- new technologies (e.g. functional magnetic imaging resonance [fMRI])

# Data Mining

©2012

Christfried Webers  
NICTA

The Australian National  
University



## Examples

What is common to this examples?

## Definition

## Related Fields

## Some Basic Notation

## Some Fundamental Types of Learning

## Training Regimes

## Journals, Conferences

## Python

## Elefant

- searching through large data sets (databases)
- goal: extracting hidden patterns from data
- examples: bioinformatics, genetics, medicine
- genetics: how do differences in the DNA between humans relate to different risks of getting diseases such as cancer
- no magic: can not uncover patterns which are not already present in the data

# Computer Science



- "What can be (efficiently) automated?"
- Algorithms
- Data Structures
- Computational complexity theory

Examples

What is common to this examples?

Definition

Related Fields

Some Basic Notation

Some Fundamental Types  
of Learning

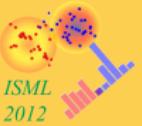
Training Regimes

Journals, Conferences

Python

Elefant

# Adaptive Control Theory



- consider systems with parameters changing (slowly) over time or being uncertain
- Example: aircraft which changes its weight over time depending on fuel consumption (which in turn depends on the wind)
- how to control such a system?
- how to estimate the parameters?

Examples

What is common to this examples?

Definition

Related Fields

Some Basic Notation

Some Fundamental Types  
of Learning

Training Regimes

Journals, Conferences

Python

Elefant



## Examples

What is common to this  
examples?

Definition

Related Fields

Some Basic Notation

Some Fundamental Types  
of Learning

Training Regimes

Journals, Conferences

Python

Elefant

# Some Basic Notation - Data

- The set of all input data is denoted as  $\mathcal{X}$ . For instance,  $\mathcal{X} = \{x \mid x \text{ is an image containing a handwritten digit}\}$ .
- One data point with  $D$  elements :

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \dots \\ x_D \end{bmatrix} = (x_1, \dots, x_D)^T.$$

- Data matrix : A set of  $N$  data points  $\mathbf{x}_i$ , where  $i = 1 \dots N$ ,

$$\mathbf{X} = \begin{bmatrix} x_{1,1} \dots x_{1,D} \\ x_{2,1} \dots x_{2,D} \\ \dots \\ x_{N,1} \dots x_{N,D} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^T \\ \dots \\ \mathbf{x}_N^T \end{bmatrix}.$$

(Note : Each data point  $\mathbf{x}_i$  is a column vector, but appears as a row vector in  $\mathbf{X}$ .)

- If  $D = 1$ ,  $\mathbf{X}$  is a vector of  $N$  scalar data points. We write

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \dots \\ \mathbf{x}_N \end{bmatrix}.$$

# Some Basic Notation - Targets



- A target can be from a finite discrete set ('labels') or from  $\mathbb{R}$ .  
(Note: Can extend this idea to  $m$ -dimensional labels and  $\mathbb{R}^m$ .)

- Set of Targets  $\mathcal{T}$ , e.g.

$$\mathcal{T} = \{\text{one, two, three, four, five, six, seven, eight, nine, zero}\}.$$

- An ordered set of  $N$  scalar labels  $\mathbf{t} = \begin{bmatrix} t_1 \\ \dots \\ t_N \end{bmatrix} = (t_1, \dots, t_N)^T$ .

Examples

What is common to this examples?

Definition

Related Fields

Some Basic Notation

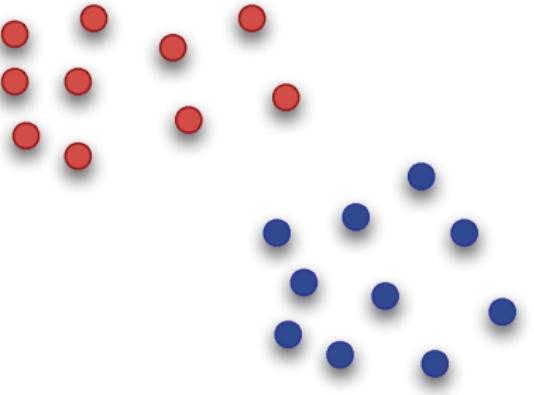
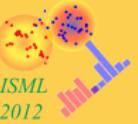
Some Fundamental Types  
of Learning

Training Regimes

Journals, Conferences

Python

Elefant



- Given are pairs of data  $x_i \in \mathcal{X}$  and targets  $t_i \in \mathcal{T}$  in the form  $(x_i, t_i)$ , where  $i = 1..N$ .
- Learn a mapping between the data  $X$  and the target  $t$  which generalises well to new data.

Examples

What is common to this examples?

Definition

Related Fields

Some Basic Notation

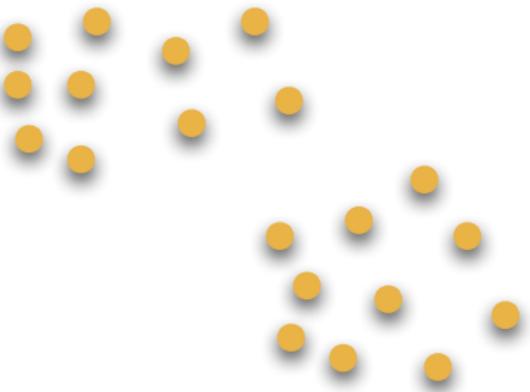
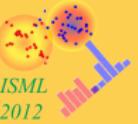
Some Fundamental Types  
of Learning

Training Regimes

Journals, Conferences

Python

Elefant



Examples

What is common to this examples?

Definition

Related Fields

Some Basic Notation

Some Fundamental Types  
of Learning

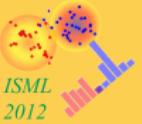
Training Regimes

Journals, Conferences

Python

Elefant

- Given only the data  $x_i \in \mathcal{X}$ .
- Discover (=learn) some interesting structure inherent in the data  $X$ .



Examples

What is common to this examples?

Definition

Related Fields

Some Basic Notation

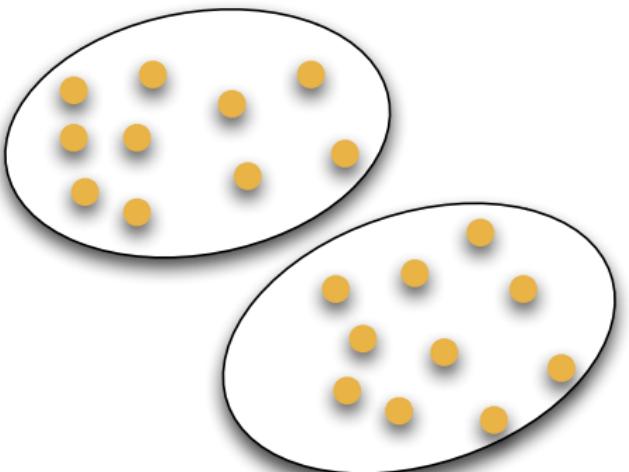
Some Fundamental Types  
of Learning

Training Regimes

Journals, Conferences

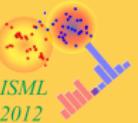
Python

Elefant



- Given only the data  $x_i \in \mathcal{X}$ .
- Discover (=learn) some interesting structure inherent in the data.

# Testing - Supervised versus Unsupervised Learning



Examples

What is common to this examples?

Definition

Related Fields

Some Basic Notation

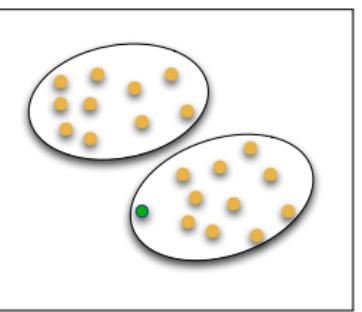
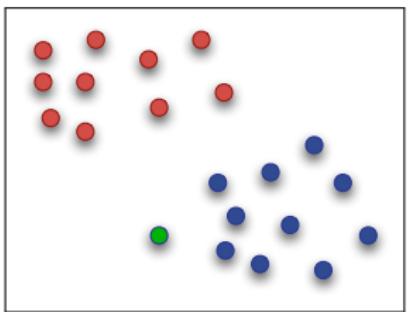
Some Fundamental Types  
of Learning

Training Regimes

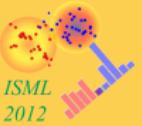
Journals, Conferences

Python

Elefant



# Reinforcement Learning



- Example: Game playing. There is one reward at the end of the game (negative or positive).
- Find suitable actions in a given environment with the goal of maximising some reward.
- correct input/output pairs never presented
- Reward might only come after many actions.
- Current action may not only influence the current reward, but future rewards too.

Examples

What is common to this examples?

Definition

Related Fields

Some Basic Notation

Some Fundamental Types  
of Learning

Training Regimes

Journals, Conferences

Python

Elefant



Examples

What is common to this examples?

Definition

Related Fields

Some Basic Notation

Some Fundamental Types  
of Learning

Training Regimes

Journals, Conferences

Python

Elefant



- Exploration versus Exploitation.
- Well suited for problems with a long-term versus short-term reward trade-off.
- Naturally focusing on online performance.



Examples

What is common to this  
examples?

Definition

Related Fields

Some Basic Notation

Some Fundamental Types  
of Learning

Training Regimes

Journals, Conferences

Python

Elefant

# Other Machine Learning Types

- 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.

[Examples](#)[What is common to this examples?](#)[Definition](#)[Related Fields](#)[Some Basic Notation](#)[Some Fundamental Types  
of Learning](#)[Training Regimes](#)[Journals, Conferences](#)[Python](#)[Elefant](#)

# Training Regimes

## • Batch Learning

- All training data  $X = \{x_1, \dots, x_n\}$  and targets  $t = \{t_1, \dots, t_n\}$  are given.
- Learn a mapping from  $x_i$  to  $t_i$  which can then be applied to yet unseen data  $X' = \{x'_1, \dots, x'_m\}$  to find  $t' = \{t'_1, \dots, t'_m\}$ .

## • Online Processing

- Pairs of  $(x_i, t_i)$  become available one at a time.
- At each step, learn and refine a mapping from  $x_i$  to  $t_i$  which can then be applied to yet unseen data  $x'_i$ .



*Examples*

*What is common to this  
examples?*

*Definition*

*Related Fields*

*Some Basic Notation*

*Some Fundamental Types  
of Learning*

*Training Regimes*

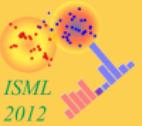
*Journals, Conferences*

*Python*

*Elefant*

# *Journals*

- Journal of Machine Learning Research
- Machine Learning
- IEEE Transactions on Pattern Analysis and Machine Intelligence
- Neural Computation
- Neural Networks
- IEEE Transactions on Neural Networks
- Annals of Statistics
- Journal of the American Statistical Association
- SIAM Journal on Applied Mathematics (SIAP)



*Examples*

*What is common to this examples?*

*Definition*

*Related Fields*

*Some Basic Notation*

*Some Fundamental Types  
of Learning*

*Training Regimes*

*Journals, Conferences*

*Python*

*Elefant*

# *Conferences*

- Algorithmic Learning Theory (ALT)
- Computational Learning Theory (COLT)
- Uncertainty in Artificial Intelligence (UAI)
- Neural Information Processing Systems (NIPS)
- European Conference on Machine Learning (ECML)
- International Conference on Machine Learning (ICML)
- International Joint Conference on Artificial Intelligence (IJCAI)
- International Conference on Artificial Neural Networks (ICANN)

# Python



- dynamically typed programming language (no declarations for variables)
- supports object oriented, imperative, and functional programming style
- many built-in data types (str, tuple, list, set, dict, ... )
- packages for scientific programming (numpy, scipy)
- easily extensible to use code written in C and C++ (or FORTRAN for that matter)
- Python runs on Windows, Linux/Unix, Mac OS X, OS/2, Amiga, Palm Handhelds, and Nokia mobile phones
- OSI-approved Open Source License

*Examples*

*What is common to this examples?*

*Definition*

*Related Fields*

*Some Basic Notation*

*Some Fundamental Types of Learning*

*Training Regimes*

*Journals, Conferences*

*Python*

*Elefant*



- Efficient Learning, Large-scale Inference, and Optimization Toolkit
- Mozilla Public License
- Two Layer
  - Functional Interface
  - Graphical User Interface

*Examples*

*What is common to this examples?*

*Definition*

*Related Fields*

*Some Basic Notation*

*Some Fundamental Types  
of Learning*

*Training Regimes*

*Journals, Conferences*

*Python*

*Elefant*

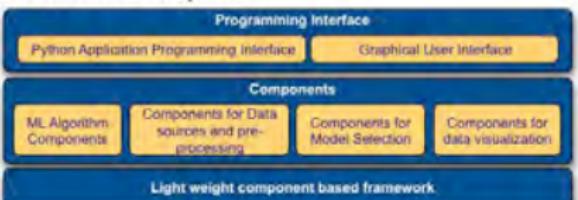


# Elefant - System Diagram

## Applications



## ELEFANT Toolkit in Python



## Core ELEFANT Modules

Various machine learning algorithms  
Support Vector Machines, Feature selections, Estimation, Classification, Quantile and Novelty detection, Neural Networks, Decision Trees, CRF, Graphical Models, etc.

Kernels - Linear, RBF, Dot Product, String, Multi class      Optimizers and Solvers      Algorithms written in C or C++ or JAVA or Fortran

Linear Algebra Library:  
PyPETSc, PyTAO, PySLEPc, PyOOQP  
Scipy and Numpy      Natural Language processing tools:  
Interface to GateD and UIMA

## External packages

Packages for extending C, C++, Fortran and JAVA code into python:  
CTypeLib, SWIG, JPy, F2Py

C or C++ External Packages:  
PETSc, TAO, SLEPc, OOQP      JAVA External Packages:  
UIMA, GATE, Mallet      Fortran External Modules:  
BLAS/LAPACK

## Examples

What is common to this examples?

## Definition

## Related Fields

## Some Basic Notation

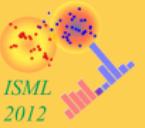
Some Fundamental Types of Learning

## Training Regimes

## Journals, Conferences

## Python

## Elefant



## Part II

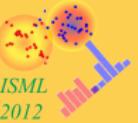
### *Introduction*

*Polynomial Curve Fitting*

*Probability Theory*

*Probability Densities*

*Expectations and  
Covariances*

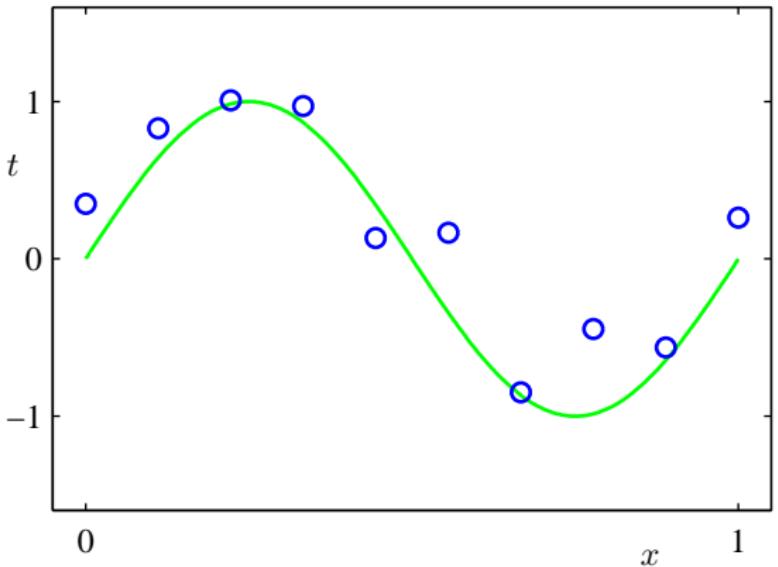


Polynomial Curve Fitting  
Probability Theory  
Probability Densities  
Expectations and Covariances

# Polynomial Curve Fitting

- some artificial data created from the function

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

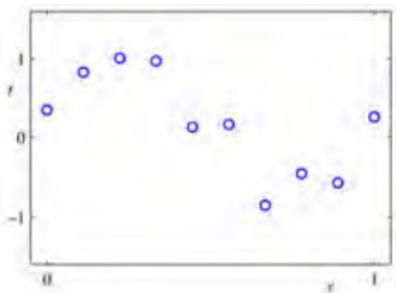




$$N = 10$$

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

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

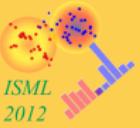


Polynomial Curve Fitting

Probability Theory

Probability Densities

Expectations and  
Covariances



Polynomial Curve Fitting

Probability Theory

Probability Densities

Expectations and  
Covariances

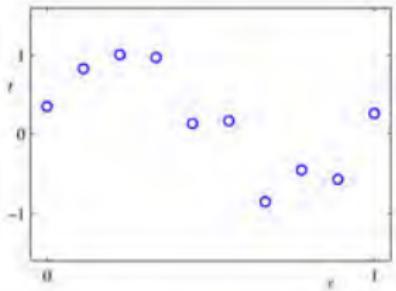
$$N = 10$$

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

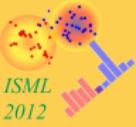
$$\mathbf{t} \equiv (t_1, \dots, t_N)^T$$

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

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



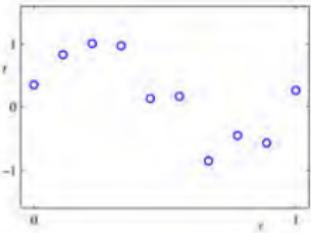
# Polynomial Curve Fitting - Model Specification



M : order of polynomial

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

$$= \sum_{m=0}^M w_m x^m$$



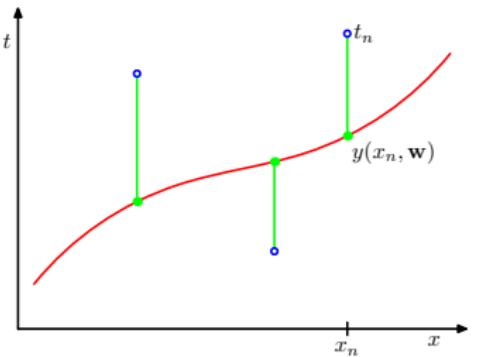
- nonlinear function of  $x$
- linear function of the unknown model parameter  $\mathbf{w}$
- How can we find good parameters  $\mathbf{w} = (w_1, \dots, w_M)^T$ ?



Polynomial Curve Fitting

Probability Theory

Probability Densities

Expectations and  
Covariances

- Performance measure : Error between target and prediction of the model for the training data

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

- unique minimum of  $E(\mathbf{w})$  for argument  $\mathbf{w}^*$



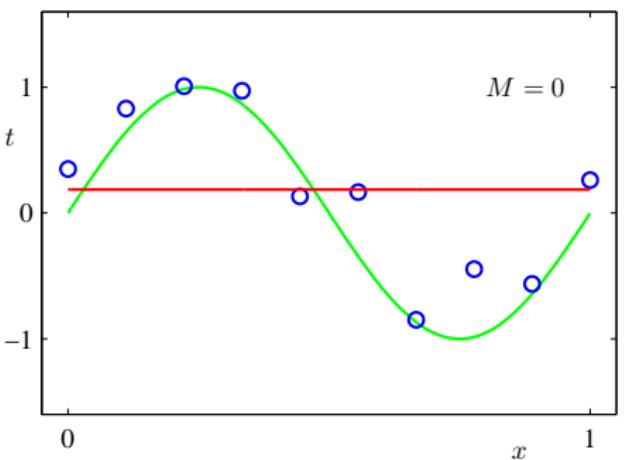
# Model Comparison or Model Selection

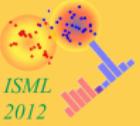
$$y(x, \mathbf{w}) = \sum_{m=0}^M w_m x^m \quad \Big|_{M=0}$$
$$= w_0$$

Polynomial Curve Fitting

Probability Theory

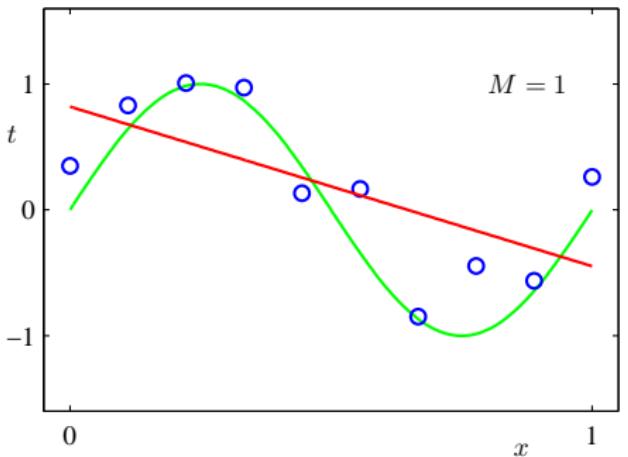
Probability Densities

Expectations and  
Covariances



# Model Comparison or Model Selection

$$y(x, \mathbf{w}) = \sum_{m=0}^M w_m x^m \quad \Bigg|_{M=1}$$
$$= w_0 + w_1 x$$

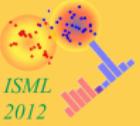


Polynomial Curve Fitting

Probability Theory

Probability Densities

Expectations and  
Covariances

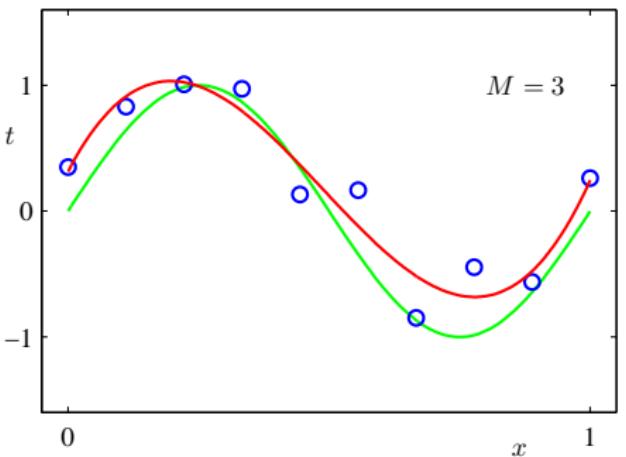


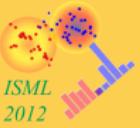
$$y(x, \mathbf{w}) = \sum_{m=0}^M w_m x^m \quad \Big|_{M=3}$$
$$= w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

Polynomial Curve Fitting

Probability Theory

Probability Densities

Expectations and  
Covariances



Polynomial Curve Fitting

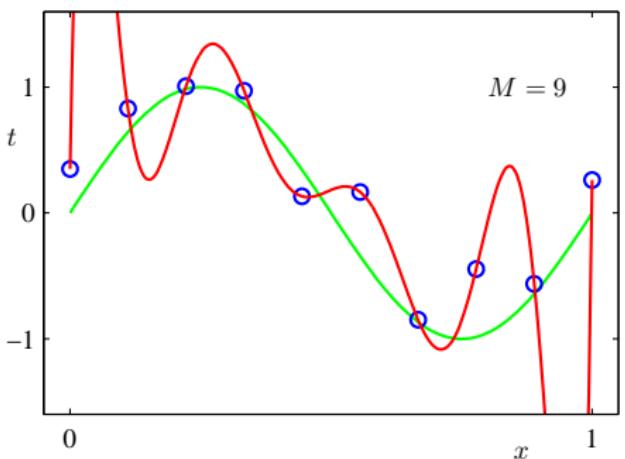
Probability Theory

Probability Densities

Expectations and  
Covariances

$$y(x, \mathbf{w}) = \sum_{m=0}^M w_m x^m \quad \Bigg|_{M=9}$$
$$= w_0 + w_1 x + \cdots + w_8 x^8 + w_9 x^9$$

- overfitting





Polynomial Curve Fitting

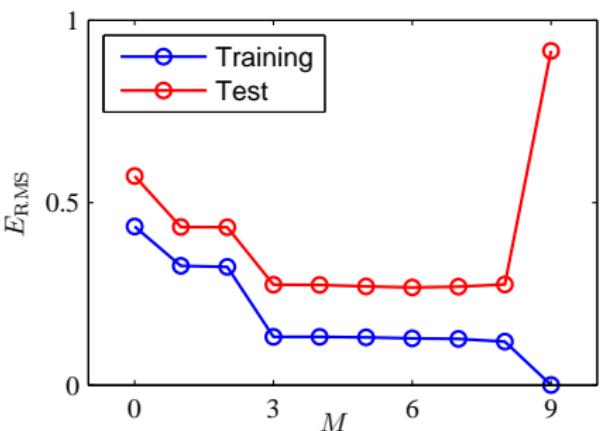
Probability Theory

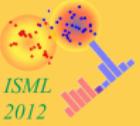
Probability Densities

Expectations and  
Covariances

- Train the model and get  $\mathbf{w}^*$
- Get 100 new data points
- Root-mean-square (RMS) error

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





Polynomial Curve Fitting

Probability Theory

Probability Densities

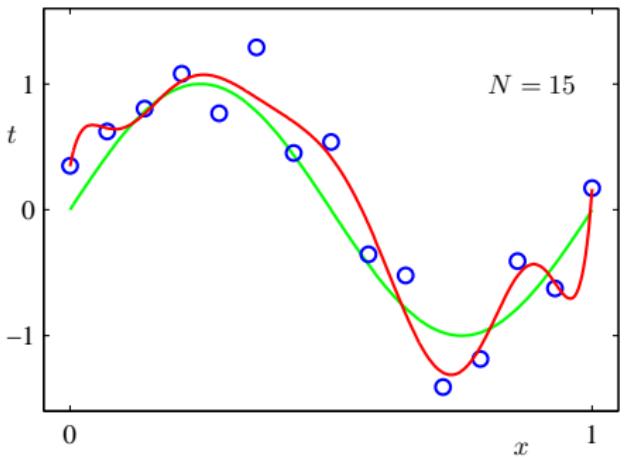
Expectations and  
Covariances

	M = 0	M = 1	M = 3	M = 9
$w_0^*$	0.19	0.82	0.31	0.35
$w_1^*$		-1.27	7.99	232.37
$w_2^*$			-25.43	-5321.83
$w_3^*$			17.37	48568.31
$w_4^*$				-231639.30
$w_5^*$				640042.26
$w_6^*$				-1061800.52
$w_7^*$				1042400.18
$w_8^*$				-557682.99
$w_9^*$				125201.43

Table : Coefficients  $w^*$  for polynomials of various order.



- $N = 15$

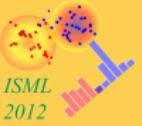


Polynomial Curve Fitting

Probability Theory

Probability Densities

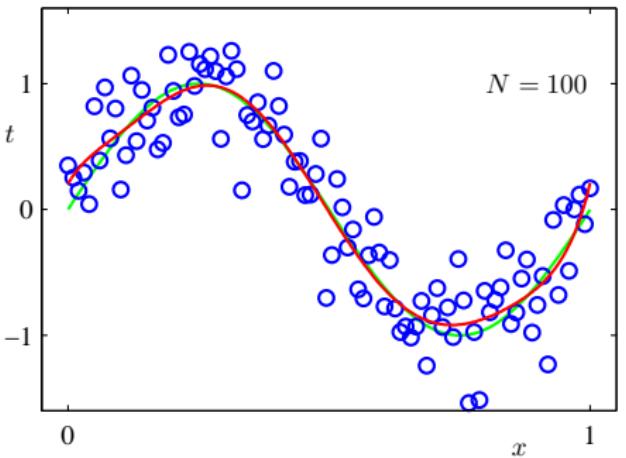
Expectations and  
Covariances



Polynomial Curve Fitting

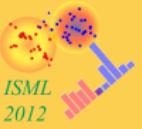
Probability Theory

Probability Densities

Expectations and  
Covariances

# More Data

- $N = 100$
- heuristics : have no less than 5 to 10 times as many data points than parameters
- but number of parameters is not necessarily the most appropriate measure of model complexity !
- later: Bayesian approach



Polynomial Curve Fitting

Probability Theory

Probability Densities

Expectations and  
Covariances

# Regularisation

- How to constrain the growing of the coefficients  $\mathbf{w}$  ?
- Add a **regularisation** term to the error function

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

- Squared norm of the parameter vector  $\mathbf{w}$

$$\|\mathbf{w}\|^2 \equiv \mathbf{w}^T \mathbf{w} = w_0^2 + w_1^2 + \cdots + w_M^2$$



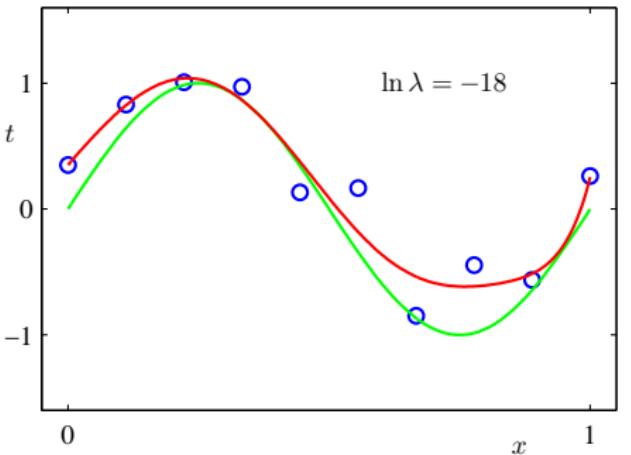
Polynomial Curve Fitting

Probability Theory

Probability Densities

Expectations and  
Covariances

- $M = 9$





Polynomial Curve Fitting

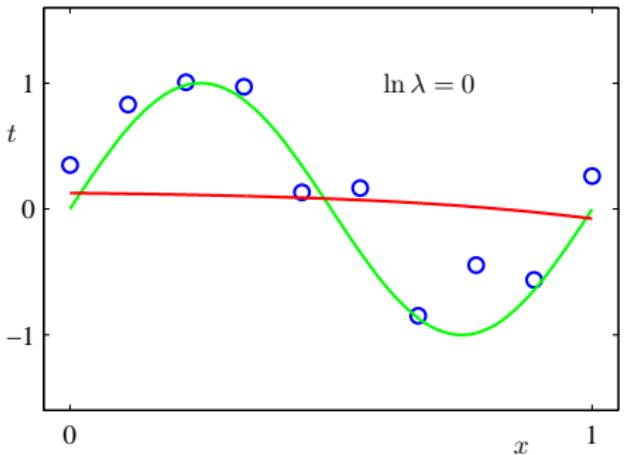
Probability Theory

Probability Densities

Expectations and  
Covariances

# Regularisation

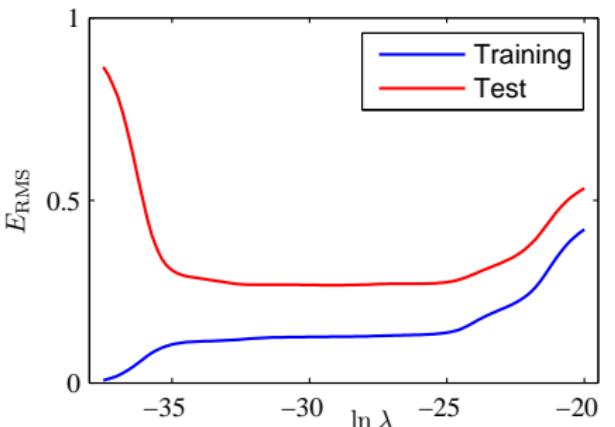
- $M = 9$





# Regularisation

- $M = 9$

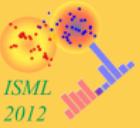


Polynomial Curve Fitting

Probability Theory

Probability Densities

Expectations and  
Covariances

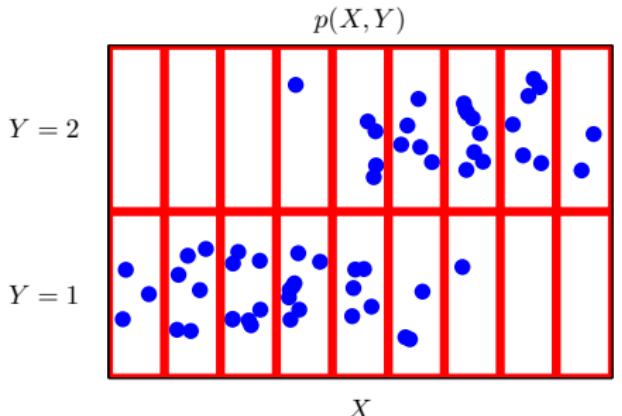


Polynomial Curve Fitting

Probability Theory

Probability Densities

Expectations and  
Covariances





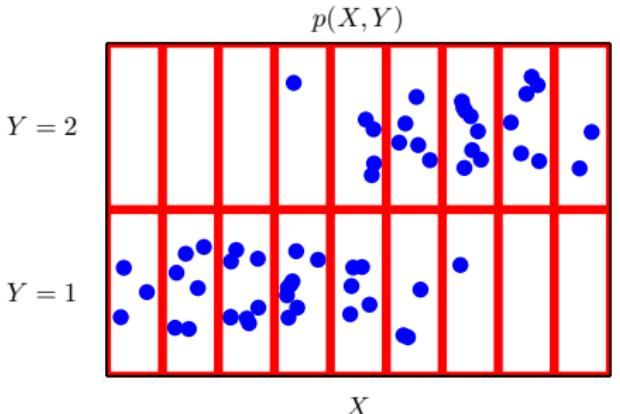
*Polynomial Curve Fitting*

*Probability Theory*

*Probability Densities*

*Expectations and  
Covariances*

Y vs. X	a	b	c	d	e	f	g	h	i	sum
2	0	0	0	1	4	5	8	6	2	26
1	3	6	8	8	5	3	1	0	0	34
sum	3	6	8	9	9	8	9	6	2	60





Polynomial Curve Fitting

Probability Theory

Probability Densities

Expectations and  
Covariances

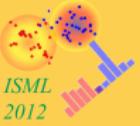
Y vs. X	a	b	c	d	e	f	g	h	i	sum
2	0	0	0	1	4	5	8	6	2	26
1	3	6	8	8	5	3	1	0	0	34
sum	3	6	8	9	9	8	9	6	2	60

$$p(X = d, Y = 1) = 8/60$$

$$\begin{aligned} p(X = d) &= p(X = d, Y = 2) + p(X = d, Y = 1) \\ &= 1/60 + 8/60 \end{aligned}$$

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

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



*Polynomial Curve Fitting*

*Probability Theory*

*Probability Densities*

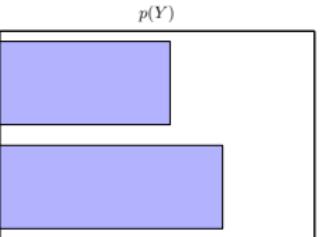
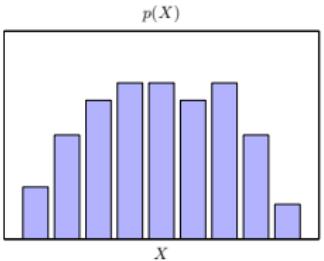
*Expectations and  
Covariances*

# Sum Rule

Y vs. X	a	b	c	d	e	f	g	h	i	sum
2	0	0	0	1	4	5	8	6	2	26
1	3	6	8	8	5	3	1	0	0	34
sum	3	6	8	9	9	8	9	6	2	60

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

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





Polynomial Curve Fitting

Probability Theory

Probability Densities

Expectations and  
Covariances

# Product Rule

Y vs. X	a	b	c	d	e	f	g	h	i	sum
2	0	0	0	1	4	5	8	6	2	26
1	3	6	8	8	5	3	1	0	0	34
sum	3	6	8	9	9	8	9	6	2	60

## Conditional Probability

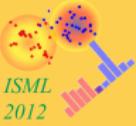
$$p(X = d \mid Y = 1) = 8/34$$

Calculate  $p(Y = 1)$ :

$$p(Y = 1) = \sum_X p(X, Y = 1) = 34/60$$

$$p(X = d, Y = 1) = p(X = d \mid Y = 1)p(Y = 1)$$

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



Polynomial Curve Fitting

Probability Theory

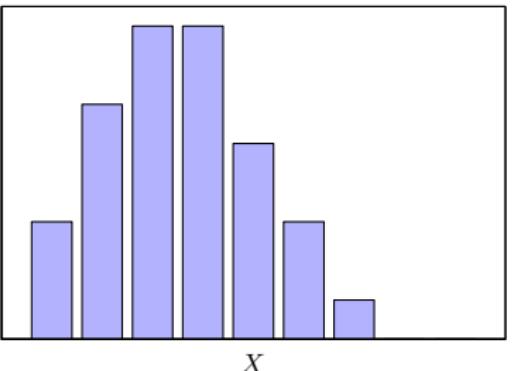
Probability Densities

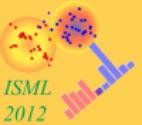
Expectations and  
Covariances

Y vs. X	a	b	c	d	e	f	g	h	i	sum
2	0	0	0	1	4	5	8	6	2	26
1	3	6	8	8	5	3	1	0	0	34
sum	3	6	8	9	9	8	9	6	2	60

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

$$p(X|Y=1)$$





Polynomial Curve Fitting

Probability Theory

Probability Densities

Expectations and  
Covariances

# Sum Rule and Product Rule

- Sum Rule

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

- Product Rule

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



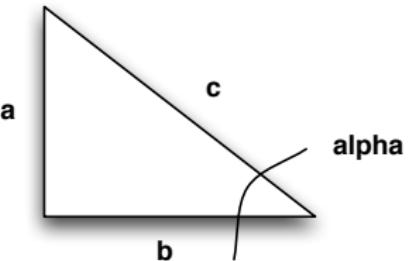
Polynomial Curve Fitting

Probability Theory

Probability Densities

Expectations and  
Covariances

- Why not using pairs of numbers  $(s, t)$  such that  $p(X, Y) = s/t$  (e.g.  $s = 8, t = 60$  )?
- Why not using pairs of numbers  $(a, c)$  instead of  $\sin(\text{alpha}) = a/c$ ?





Polynomial Curve Fitting

Probability Theory

Probability Densities

Expectations and  
Covariances

# Bayes Theorem

Use product rule

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

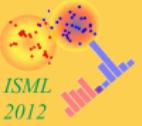
Bayes Theorem

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

and

$$p(X) = \sum_Y p(X, Y) \quad (\text{sum rule})$$

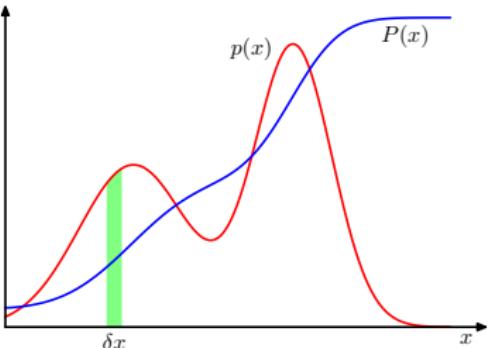
$$= \sum_Y p(X \mid Y) p(Y) \quad (\text{product rule})$$



# Probability Densities

- Real valued variable  $x \in \mathbb{R}$
- Probability of  $x$  to fall in the interval  $(x, x + \delta x)$  is given by  $p(x)\delta x$  for infinitesimal small  $\delta x$ .
- 

$$p(x \in (a, b)) = \int_a^b p(x) \, dx.$$



Polynomial Curve Fitting

Probability Theory

Probability Densities

Expectations and  
Covariances



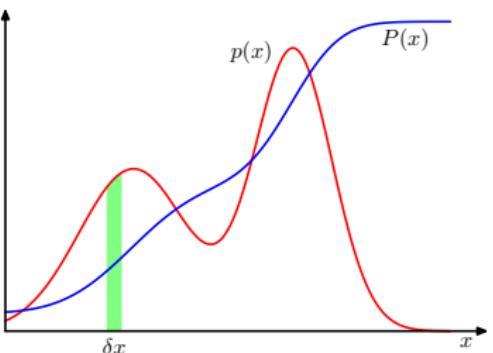
# Constraints on $p(x)$

- Nonnegative

$$p(x) \geq 0$$

- Normalisation

$$\int_{-\infty}^{\infty} p(x) \, dx = 1.$$



Polynomial Curve Fitting

Probability Theory

Probability Densities

Expectations and  
Covariances

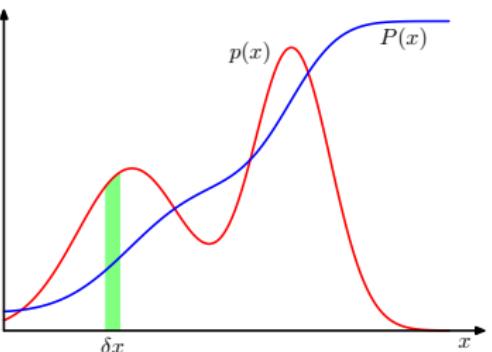


# Cumulative distribution function $P(x)$

$$P(x) = \int_{-\infty}^x p(z) \, dz$$

or

$$\frac{d}{dx} P(x) = p(x)$$



Polynomial Curve Fitting

Probability Theory

Probability Densities

Expectations and  
Covariances



Polynomial Curve Fitting

Probability Theory

Probability Densities

Expectations and  
Covariances

# Multivariate Probability Density

- Vector  $\mathbf{x} \equiv (x_1, \dots, x_D)^T = \begin{bmatrix} x_1 \\ \vdots \\ x_D \end{bmatrix}$

- Nonnegative

$$p(\mathbf{x}) \geq 0$$

- Normalisation

$$\int_{-\infty}^{\infty} p(\mathbf{x}) \, d\mathbf{x} = 1.$$

- This means

$$\int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} p(\mathbf{x}) \, dx_1 \dots \, dx_D = 1.$$

# Sum and Product Rule for Probability Densities



Polynomial Curve Fitting

Probability Theory

Probability Densities

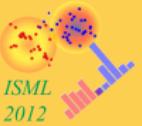
Expectations and  
Covariances

- Sum Rule

$$p(x) = \int_{-\infty}^{\infty} p(x, y) \, dy$$

- Product Rule

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



Polynomial Curve Fitting

Probability Theory

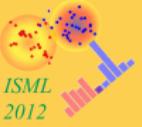
Probability Densities

Expectations and  
Covariances

- Weighted average of a function  $f(x)$  under the probability distribution  $p(x)$

$$\mathbb{E}[f] = \sum_x p(x)f(x) \quad \text{discrete distribution } p(x)$$

$$\mathbb{E}[f] = \int p(x)f(x) \, dx \quad \text{probability density } p(x)$$



Polynomial Curve Fitting

Probability Theory

Probability Densities

Expectations and  
Covariances

- Given a finite number  $N$  of points  $x_n$  drawn from the probability distribution  $p(x)$ .
- Approximate the expectation by a finite sum:

$$\mathbb{E}[f] \simeq \frac{1}{N} \sum_{n=1}^N f(x_n)$$

- How to draw points from a probability distribution  $p(x)$  ?  
Lecture coming about “Sampling”

# *Expectation of a function of several variables*



- arbitrary function  $f(x, y)$

$$\mathbb{E}_x [f(x, y)] = \sum_x p(x) f(x, y) \quad \text{discrete distribution } p(x)$$

$$\mathbb{E}_x [f(x, y)] = \int p(x) f(x, y) \, dx \quad \text{probability density } p(x)$$

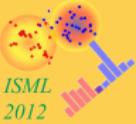
- Note that  $\mathbb{E}_x [f(x, y)]$  is a function of  $y$ .

*Polynomial Curve Fitting*

*Probability Theory*

*Probability Densities*

*Expectations and  
Covariances*



Polynomial Curve Fitting

Probability Theory

Probability Densities

Expectations and  
Covariances

# Conditional Expectation

- arbitrary function  $f(x)$

$$\mathbb{E}_x [f \mid y] = \sum_x p(x \mid y) f(x) \quad \text{discrete distribution } p(x)$$

$$\mathbb{E}_x [f \mid y] = \int p(x \mid y) f(x) \, dx \quad \text{probability density } p(x)$$

- Note that  $\mathbb{E}_x [f \mid y]$  is a function of  $y$ .
- Other notation used in the literature :  $\mathbb{E}_{x|y} [f]$ .
- What is  $\mathbb{E} [\mathbb{E} [f(x) \mid y]]$  ? Can we simplify it?
- This must mean  $\mathbb{E}_y [\mathbb{E}_x [f(x) \mid y]]$ . (Why?)

$$\begin{aligned}\mathbb{E}_y [\mathbb{E}_x [f(x) \mid y]] &= \sum_y p(y) \mathbb{E}_x [f \mid y] = \sum_y p(y) \sum_x p(x|y) f(x) \\ &= \sum_{x,y} f(x) p(x,y) = \sum_x f(x) p(x) \\ &= \mathbb{E}_x [f(x)]\end{aligned}$$



Polynomial Curve Fitting

Probability Theory

Probability Densities

Expectations and  
Covariances

# Variance

- arbitrary function  $f(x)$

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

- Special case:  $f(x) = x$

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



Polynomial Curve Fitting

Probability Theory

Probability Densities

Expectations and  
Covariances

# Covariance

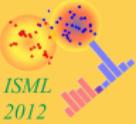
- Two random variables  $x \in \mathbb{R}$  and  $y \in \mathbb{R}$

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

- With  $\mathbb{E}[x] = a$  and  $\mathbb{E}[y] = b$

$$\begin{aligned}\text{cov}[x, y] &= \mathbb{E}_{x,y} [(x - a)(y - b)] \\ &= \mathbb{E}_{x,y} [xy] - \mathbb{E}_{x,y} [xb] - \mathbb{E}_{x,y} [ay] + \mathbb{E}_{x,y} [ab] \\ &= \mathbb{E}_{x,y} [xy] - b \underbrace{\mathbb{E}_{x,y} [x]}_{=\mathbb{E}_x[x]} - a \underbrace{\mathbb{E}_{x,y} [y]}_{=\mathbb{E}_y[y]} + ab \underbrace{\mathbb{E}_{x,y} [1]}_{=1} \\ &= \mathbb{E}_{x,y} [xy] - ab - ab + ab = \mathbb{E}_{x,y} [xy] - ab \\ &= \mathbb{E}_{x,y} [xy] - \mathbb{E}[x]\mathbb{E}[y]\end{aligned}$$

- Expresses how strongly  $x$  and  $y$  vary together. If  $x$  and  $y$  are independent, their covariance vanishes.



- Two random variables  $\mathbf{x} \in \mathbb{R}^D$  and  $\mathbf{y} \in \mathbb{R}^D$

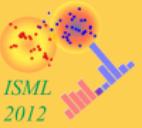
$$\begin{aligned}\text{cov}[\mathbf{x}, \mathbf{y}] &= \mathbb{E}_{\mathbf{x}, \mathbf{y}} [(\mathbf{x} - \mathbb{E} [\mathbf{x}])(\mathbf{y}^T - \mathbb{E} [\mathbf{y}^T])] \\ &= \mathbb{E}_{\mathbf{x}, \mathbf{y}} [\mathbf{x} \mathbf{y}^T] - \mathbb{E} [\mathbf{x}] \mathbb{E} [\mathbf{y}^T]\end{aligned}$$

Polynomial Curve Fitting

Probability Theory

Probability Densities

Expectations and  
Covariances



## Part III

# *Linear Algebra*

*Basic Concepts*

*Linear Transformations*

*Trace*

*Inner Product*

*Projection*

*Rank, Determinant, Trace*

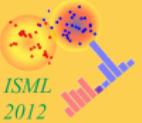
*Matrix Inverse*

*Eigenvectors*

*Singular Value  
Decomposition*

*Directional Derivative,  
Gradient*

*Books*



Basic Concepts

Linear Transformations

Trace

Inner Product

Projection

Rank, Determinant, Trace

Matrix Inverse

Eigenvectors

Singular Value  
Decomposition

Directional Derivative,  
Gradient

Books

# Intuition

## Geometry

- Points and Lines
- Vector addition and scaling
- Humans have experience with 3 dimensions (less with 1, 2 though)

Generalisation to  $N$  dimensions (possibly  $N \rightarrow \infty$ )

- Line  $\rightarrow$  vector space  $\mathbb{V}$
- Point  $\rightarrow$  vector  $x \in \mathbb{V}$
- Example :  $X \in \mathbb{R}^{n \times m}$
- Space of matrices  $\mathbb{R}^{n \times m}$  and the space of vectors  $\mathbb{R}^{n \cdot m}$  are isomorphic



# Vector Space $\mathcal{V}$ , underlying Field $\mathcal{F}$

Given vectors  $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathcal{V}$  and scalars  $\alpha, \beta \in \mathcal{F}$ , the following holds:

- Associativity of addition :  $\mathbf{u} + (\mathbf{v} + \mathbf{w}) = (\mathbf{u} + \mathbf{v}) + \mathbf{w}$  .
- Commutativity of addition :  $\mathbf{v} + \mathbf{w} = \mathbf{w} + \mathbf{v}$  .
- Identity element of addition :  $\exists \mathbf{0}$  such that  $\mathbf{v} + \mathbf{0} = \mathbf{v}, \forall \mathbf{v} \in \mathcal{V}$ .
- Inverse of addition : For all  $\mathbf{v} \in \mathcal{V}, \exists \mathbf{w} \in \mathcal{V}$ , such that  $\mathbf{v} + \mathbf{w} = \mathbf{0}$ . The additive inverse is denoted  $-\mathbf{v}$ .
- Distributivity of scalar multiplication with respect to vector addition :  $\alpha(\mathbf{v} + \mathbf{w}) = \alpha\mathbf{v} + \alpha\mathbf{w}$  .
- Distributivity of scalar multiplication with respect to field addition :  $(\alpha + \beta)\mathbf{v} = \alpha\mathbf{v} + \beta\mathbf{v}$  .
- Compatibility of scalar multiplication with field multiplication :  $\alpha(\beta\mathbf{v}) = (\alpha\beta)\mathbf{v}$  .
- Identity element of scalar multiplication :  $1\mathbf{v} = \mathbf{v}$ , where 1 is the identity in  $\mathcal{F}$  .

Basic Concepts

Linear Transformations

Trace

Inner Product

Projection

Rank, Determinant, Trace

Matrix Inverse

Eigenvectors

Singular Value  
Decomposition

Directional Derivative,  
Gradient

Books

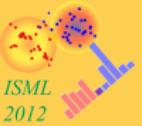
*Basic Concepts**Linear Transformations**Trace**Inner Product**Projection**Rank, Determinant, Trace**Matrix Inverse**Eigenvectors**Singular Value  
Decomposition**Directional Derivative,  
Gradient**Books*

# Matrix-Vector Multiplication

```
1  for i in xrange(m):
    R[i] = 0.0;
    for j in xrange(n):
        R[i] = R[i] + A[i,j] * V[j]
```

Listing 1: Code for elementwise matrix multiplication.

$$\begin{array}{c} A \qquad \qquad \qquad V \qquad \qquad \qquad R \\[10pt]
 \left[ \begin{array}{cccc} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{array} \right] \left[ \begin{array}{c} v_1 \\ v_2 \\ \vdots \\ v_n \end{array} \right] = \left[ \begin{array}{c} a_{11}v_1 + a_{12}v_2 + \dots + a_{1n}v_n \\ a_{21}v_1 + a_{22}v_2 + \dots + a_{2n}v_n \\ \vdots \\ a_{m1}v_1 + a_{m2}v_2 + \dots + a_{mn}v_n \end{array} \right] \end{array}$$



Basic Concepts

Linear Transformations

Trace

Inner Product

Projection

Rank, Determinant, Trace

Matrix Inverse

Eigenvectors

Singular Value  
DecompositionDirectional Derivative,  
Gradient

Books

$$A \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = R$$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} a_{11}v_1 + a_{12}v_2 + \cdots + a_{1n}v_n \\ a_{21}v_1 + a_{22}v_2 + \cdots + a_{2n}v_n \\ \cdots \\ a_{m1}v_1 + a_{m2}v_2 + \cdots + a_{mn}v_n \end{bmatrix}$$

```

1 R = A[:,0] * V[0];
  for j in xrange(1, n):
    R += A[:, j] * V[j];

```

Listing 2: Code for columnwise matrix multiplication.



- Denote the  $n$  columns of  $A$  by  $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n$
- Each  $\mathbf{a}_i$  is now a (column) vector

Basic Concepts

Linear Transformations

Trace

Inner Product

Projection

Rank, Determinant, Trace

Matrix Inverse

Eigenvectors

Singular Value  
DecompositionDirectional Derivative,  
Gradient

Books

$$A \quad V = R$$
$$\begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1 v_1 + \mathbf{a}_2 v_2 + \dots + \mathbf{a}_n v_n \end{bmatrix}$$



# Transpose of R

- Given  $R = AV$ ,

$$R = \begin{bmatrix} \mathbf{a}_1 v_1 + \mathbf{a}_2 v_2 + \cdots + \mathbf{a}_n v_n \end{bmatrix}$$

- What is  $R^T$  ?

- 

$$R^T = [v_1 \mathbf{a}_1^T + v_2 \mathbf{a}_2^T + \cdots + v_n \mathbf{a}_n^T]$$

- NOT equal to  $A^T V^T$  ! (In fact,  $A^T V^T$  is not even defined.)

Basic Concepts

Linear Transformations

Trace

Inner Product

Projection

Rank, Determinant, Trace

Matrix Inverse

Eigenvectors

Singular Value  
Decomposition

Directional Derivative,  
Gradient

Books



- Reverse order rule :  $(AV)^T = V^T A^T$

$$\begin{matrix} V^T & & A^T & = & R^T \\ \left[ v_1 \quad v_2 \dots v_n \right] & & \left[ \begin{array}{c} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \vdots \\ \mathbf{a}_n^T \end{array} \right] & = & \left[ v_1 \mathbf{a}_1^T + v_2 \mathbf{a}_2^T + \dots + v_n \mathbf{a}_n^T \right] \end{matrix}$$

Basic Concepts

Linear Transformations

Trace

Inner Product

Projection

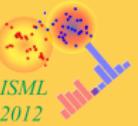
Rank, Determinant, Trace

Matrix Inverse

Eigenvectors

Singular Value  
DecompositionDirectional Derivative,  
Gradient

Books



- The **trace** of a square matrix  $A$  is the sum of all diagonal elements of  $A$ .

$$\text{tr} \{A\} = \sum_{k=1}^n A_{kk}$$

- Example

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$\text{tr} \{A\} = 1 + 5 + 9 = 15$$

- The trace does not exist for a non-square matrix.

Basic Concepts

Linear Transformations

Trace

Inner Product

Projection

Rank, Determinant, Trace

Matrix Inverse

Eigenvectors

Singular Value  
Decomposition

Directional Derivative,  
Gradient

Books



Basic Concepts

Linear Transformations

Trace

Inner Product

Projection

Rank, Determinant, Trace

Matrix Inverse

Eigenvectors

Singular Value  
DecompositionDirectional Derivative,  
Gradient

Books

# vec( $x$ ) operator

- Define  $\text{vec}(X)$  as the vector which results from stacking all columns of a matrix  $A$  on top of each other.
- Example

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$\text{vec}(A) = \begin{bmatrix} 1 \\ 4 \\ 7 \\ 2 \\ 5 \\ 8 \\ 3 \\ 6 \\ 9 \end{bmatrix}$$

- Given two matrices  $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{n \times p}$ , the trace of the product  $\mathbf{X}^T \mathbf{Y}$  can be written as

$$\text{tr}\{\mathbf{X}^T \mathbf{Y}\} = \text{vec}(\mathbf{X})^T \text{vec}(\mathbf{Y})$$



- Mapping from two vectors  $x, y \in \mathbb{V}$  to a field of scalars  $\mathbb{F}$   
( $\mathbb{F} = \mathbb{R}$  or  $\mathbb{F} = \mathbb{C}$ ):

$$\langle \cdot, \cdot \rangle : \mathbb{V} \times \mathbb{V} \rightarrow \mathbb{F}$$

- Conjugate Symmetry :

$$\langle x, y \rangle = \overline{\langle y, x \rangle}$$

- Linearity :

$$\langle ax, y \rangle = a \langle x, y \rangle, \text{ and } \langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle$$

- Positive-definiteness :

$$\langle x, x \rangle \geq 0, \text{ and } \langle x, x \rangle = 0 \text{ for } x = 0 \text{ only.}$$

Basic Concepts

Linear Transformations

Trace

Inner Product

Projection

Rank, Determinant, Trace

Matrix Inverse

Eigenvectors

Singular Value  
Decomposition

Directional Derivative,  
Gradient

Books



Basic Concepts

Linear Transformations

Trace

Inner Product

Projection

Rank, Determinant, Trace

Matrix Inverse

Eigenvectors

Singular Value  
DecompositionDirectional Derivative,  
Gradient

Books

# Canonical Inner Products

- Inner product for real numbers  $x, y \in \mathbb{R}$ :

$$\langle x, y \rangle = xy$$

- Dot product between two vectors:

Given  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y} \equiv \sum_{k=1}^n x_k y_k = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$$

- Canonical inner product for matrices :

Given  $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{n \times p}$

$$\begin{aligned} \langle \mathbf{X}, \mathbf{Y} \rangle &= \text{tr} \{ \mathbf{X}^T \mathbf{Y} \} = \sum_{k=1}^p (\mathbf{X}^T \mathbf{Y})_{kk} = \sum_{k=1}^p \sum_{l=1}^n (\mathbf{X}^T)_{kl} (\mathbf{Y})_{lk} \\ &= \sum_{k=1}^p \sum_{l=1}^n \mathbf{X}_{lk} \mathbf{Y}_{lk} \end{aligned}$$



Basic Concepts

Linear Transformations

Trace

Inner Product

Projection

Rank, Determinant, Trace

Matrix Inverse

Eigenvectors

Singular Value  
DecompositionDirectional Derivative,  
Gradient

Books

# Calculations with Matrices

- Denote  $(i,j)^{\text{th}}$  element of matrix  $\mathbf{X}$  by  $\mathbf{X}_{ij}$
- Transpose :  $(\mathbf{X}^T)_{ij} = \mathbf{X}_{ji}$
- Product :  $(\mathbf{XY})_{ij} = \sum_{k=1}^m \mathbf{X}_{ik} \mathbf{Y}_{kj}$
- Proof of the linearity of the canonical matrix inner product :

$$\langle \mathbf{X} + \mathbf{Y}, \mathbf{Z} \rangle = \text{tr} \left\{ (\mathbf{X} + \mathbf{Y})^T \mathbf{Z} \right\} = \sum_{k=1}^p ((\mathbf{X} + \mathbf{Y})^T \mathbf{Z})_{kk}$$

$$= \sum_{k=1}^p \sum_{l=1}^n ((\mathbf{X} + \mathbf{Y})^T)_{kl} \mathbf{Z}_{lk} = \sum_{k=1}^p \sum_{l=1}^n (\mathbf{X}_{lk} + \mathbf{Y}_{lk}) \mathbf{Z}_{lk}$$

$$= \sum_{k=1}^p \sum_{l=1}^n \mathbf{X}_{lk} \mathbf{Z}_{lk} + \mathbf{Y}_{lk} \mathbf{Z}_{lk}$$

$$= \sum_{k=1}^p \sum_{l=1}^n (\mathbf{X}^T)_{kl} \mathbf{Z}_{lk} + (\mathbf{Y}^T)_{kl} \mathbf{Z}_{lk}$$

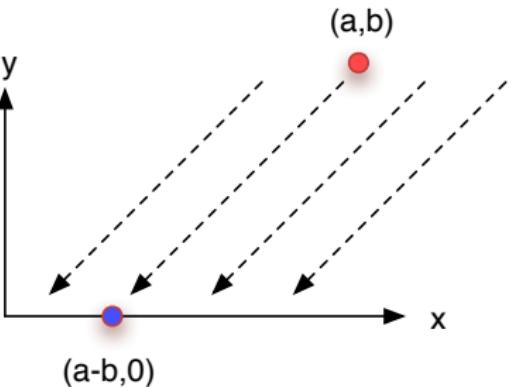
$$= \sum_{k=1}^p (\mathbf{X}^T \mathbf{Z})_{kk} + (\mathbf{Y}^T \mathbf{Z})_{kk}$$

$$= \text{tr} \left\{ \mathbf{X}^T \mathbf{Z} \right\} + \text{tr} \left\{ \mathbf{Y}^T \mathbf{Z} \right\} = \langle \mathbf{X}, \mathbf{Z} \rangle + \langle \mathbf{Y}, \mathbf{Z} \rangle$$



- In linear algebra and functional analysis, a projection is a linear transformation  $P$  from a vector space  $\mathbb{V}$  to itself such that

$$P^2 = P$$



$$A \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a - b \\ 0 \end{bmatrix} \quad A = \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix}$$

[Basic Concepts](#)[Linear Transformations](#)[Trace](#)[Inner Product](#)[Projection](#)[Rank, Determinant, Trace](#)[Matrix Inverse](#)[Eigenvectors](#)[Singular Value  
Decomposition](#)[Directional Derivative,  
Gradient](#)[Books](#)



Basic Concepts

Linear Transformations

Trace

Inner Product

Projection

Rank, Determinant, Trace

Matrix Inverse

Eigenvectors

Singular Value  
DecompositionDirectional Derivative,  
Gradient

Books

# Orthogonal Projection

- Orthogonality : need an **inner product**  $\langle x, y \rangle$
- Choose two arbitrary vectors  $x$  and  $y$ . Then  $Px$  and  $y - Py$  are orthogonal.

$$0 = \langle Px, y - Py \rangle = (Px)^T(y - Py) = x^T(P - P^T P)y$$

- Orthogonal Projection

$$P^2 = P \quad P = P^T$$

- Example : Given some unit vector  $u \in \mathbb{R}^n$  characterising a line through the origin in  $\mathbb{R}^n$
- project an arbitrary vector  $x \in \mathbb{R}^n$  onto this line by

$$P = uu^T$$

- Proof :  $P = P^T$ , and

$$P^2x = (uu^T)(uu^T)x = uu^Tx = Px$$

*Basic Concepts**Linear Transformations**Trace**Inner Product**Projection**Rank, Determinant, Trace**Matrix Inverse**Eigenvectors**Singular Value  
Decomposition**Directional Derivative,  
Gradient**Books*

# Orthogonal Projection

- Given a matrix  $\mathbf{A} \in \mathbb{R}^{n \times p}$  and a vector  $\mathbf{x}$ . What is the closest point  $\tilde{\mathbf{x}}$  to  $\mathbf{x}$  in the column space of  $\mathbf{A}$  ?
- Orthogonal Projection into the column space of  $\mathbf{A}$

$$\tilde{\mathbf{x}} = \mathbf{A}(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{x}$$

- Projection matrix

$$P = \mathbf{A}(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T$$

- Proof

$$P^2 = \mathbf{A}(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{A}(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T = \mathbf{A}(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T = P$$

- Orthogonal projection ?

$$P^T = (\mathbf{A}(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T)^T = \mathbf{A}(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T = P$$



Basic Concepts

Linear Transformations

Trace

Inner Product

Projection

Rank, Determinant, Trace

Matrix Inverse

Eigenvectors

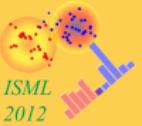
Singular Value  
DecompositionDirectional Derivative,  
Gradient

Books

$$a_1 \mathbf{v}_1 + a_2 \mathbf{v}_2 + \cdots + a_k \mathbf{v}_k = \mathbf{0} \equiv \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

has only the trivial solution

$$a_1 = a_2 = a_k = 0$$



*Basic Concepts*

*Linear Transformations*

*Trace*

*Inner Product*

*Projection*

*Rank, Determinant, Trace*

*Matrix Inverse*

*Eigenvectors*

*Singular Value  
Decomposition*

*Directional Derivative,  
Gradient*

*Books*

# Rank

- The **column rank** of a matrix  $A$  is the maximal number of linearly independent columns of  $A$ .
- The **row rank** of a matrix  $A$  is the maximal number of linearly independent rows of  $A$ .
- For every matrix  $A$ , the row rank and column rank are equal, called **the rank** of  $A$ .



Let  $S_n$  be the set of all permutation of the numbers  $\{1, \dots, n\}$ .  
The determinant of the square matrix  $A$  is then given by

$$\det \{A\} = \sum_{\sigma \in S_n} \text{sgn}(\sigma) \prod_{i=1}^n A_{i,\sigma(i)}$$

where  $\text{sgn}$  is the signature of the permutation (+1 if the permutation is even, -1 if the permutation is odd).

- $\det \{AB\} = \det \{A\} \det \{B\}$  for  $A, B$  square matrices.
- $\det \{A^{-1}\} = \det \{A\}^{-1}$ .
- $\det \{A^T\} = \det \{A\}$ .

*Basic Concepts*

*Linear Transformations*

*Trace*

*Inner Product*

*Projection*

*Rank, Determinant, Trace*

*Matrix Inverse*

*Eigenvectors*

*Singular Value  
Decomposition*

*Directional Derivative,  
Gradient*

*Books*

*Basic Concepts**Linear Transformations**Trace**Inner Product**Projection**Rank, Determinant, Trace**Matrix Inverse**Eigenvectors**Singular Value  
Decomposition**Directional Derivative,  
Gradient**Books*

# Matrix Inverse

- Identity Matrix  $I$
- $I = AA^{-1} = A^{-1}A$
- The matrix inverse  $A^{-1}$  does only exist for square matrices which are NOT singular.
- Singular matrix
  - at least one eigenvalue is zero,
  - determinant  $|A| = 0$ .
- Inversion 'reverses the order' (like transposition)

$$(AB)^{-1} = B^{-1}A^{-1}$$

(Only then is  $(AB)(AB)^{-1} = (AB)B^{-1}A^{-1} = AA^{-1} = I$ .)



Basic Concepts

Linear Transformations

Trace

Inner Product

Projection

Rank, Determinant, Trace

Matrix Inverse

Eigenvectors

Singular Value  
Decomposition

Directional Derivative,  
Gradient

Books

# Matrix Inverse and Transpose

- $(A^{-1})^T$  ?
- need to assume that  $(A^{-1})$  exists
- rule :  $(A^{-1})^T = (A^T)^{-1}$



$$(A^{-1} + B^T C^{-1} B)^{-1} B^T C^{-1} = A B^T (B A B^T + C)^{-1}$$

- How to analyse and prove such an equation?
- $A^{-1}$  must exist, therefore  $A$  must be square, say  $A \in \mathbb{R}^{n \times n}$ .
- Same for  $C^{-1}$ , so let's assume  $C \in \mathbb{R}^{m \times m}$ .
- Therefore,  $B \in \mathbb{R}^{m \times n}$ .

Basic Concepts

Linear Transformations

Trace

Inner Product

Projection

Rank, Determinant, Trace

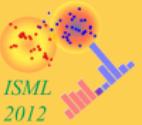
Matrix Inverse

Eigenvectors

Singular Value  
Decomposition

Directional Derivative,  
Gradient

Books



$$(A^{-1} + B^T C^{-1} B)^{-1} B^T C^{-1} = AB^T (BAB^T + C)^{-1}$$

- Multiply by  $(BAB^T + C)$

$$(A^{-1} + B^T C^{-1} B)^{-1} B^T C^{-1} (BAB^T + C) = AB^T$$

- Simplify the left-hand side

$$\begin{aligned} & (A^{-1} + B^T C^{-1} B)^{-1} [B^T C^{-1} B (AB^T) + B^T] \\ &= (A^{-1} + B^T C^{-1} B)^{-1} [B^T C^{-1} B + A^{-1}] AB^T \\ &= AB^T \end{aligned}$$

Basic Concepts

Linear Transformations

Trace

Inner Product

Projection

Rank, Determinant, Trace

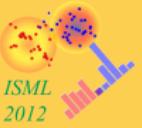
Matrix Inverse

Eigenvectors

Singular Value  
Decomposition

Directional Derivative,  
Gradient

Books



Basic Concepts

Linear Transformations

Trace

Inner Product

Projection

Rank, Determinant, Trace

Matrix Inverse

Eigenvectors

Singular Value  
Decomposition

Directional Derivative,  
Gradient

Books

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

- Useful if  $A$  is diagonal and easy to invert, and  $B$  is very tall (many rows, but only a few columns) and  $C$  is very wide (few rows, many columns).



- Don't multiply by a matrix which does not have full rank.  
Why? In the general case, you will loose solutions.
- Don't multiply by nonsquare matrices.
- Don't assume a matrix inverse exists because a matrix is square. The matrix might be singular and you are making the same mistake as if deducing  $a = b$  from  $a 0 = b 0$ .

*Basic Concepts*

*Linear Transformations*

*Trace*

*Inner Product*

*Projection*

*Rank, Determinant, Trace*

*Matrix Inverse*

*Eigenvectors*

*Singular Value  
Decomposition*

*Directional Derivative,  
Gradient*

*Books*



Basic Concepts

Linear Transformations

Trace

Inner Product

Projection

Rank, Determinant, Trace

Matrix Inverse

Eigenvectors

Singular Value  
DecompositionDirectional Derivative,  
Gradient

Books

# Eigenvectors

- Every square matrix  $A \in \mathbb{R}^{n \times n}$  has an Eigenvector decomposition

$$Ax = \lambda x$$

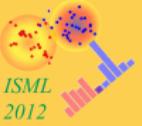
where  $x \in \mathbb{R}^n$  and  $\lambda \in \mathbb{C}$ .

- Example:

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} x = \lambda x$$

$$\lambda = \{-i, i\}$$

$$x = \left\{ \begin{bmatrix} i \\ 1 \end{bmatrix}, \begin{bmatrix} -i \\ 1 \end{bmatrix} \right\}$$



- How many eigenvalue/eigenvector pairs?
- 

$$Ax = \lambda x$$

is equivalent to

$$(A - \lambda I)x = 0$$

- Has only non-trivial solution for  $\det\{A - \lambda I\} = 0$
- polynom of  $n$ th order; at most  $n$  distinct solutions

Basic Concepts

Linear Transformations

Trace

Inner Product

Projection

Rank, Determinant, Trace

Matrix Inverse

Eigenvectors

Singular Value  
Decomposition

Directional Derivative,  
Gradient

Books



Basic Concepts

Linear Transformations

Trace

Inner Product

Projection

Rank, Determinant, Trace

Matrix Inverse

Eigenvectors

Singular Value  
DecompositionDirectional Derivative,  
Gradient

Books

# Real Eigenvalues

- How can we enforce real eigenvalues?
- Let's look at matrices with complex entries  $A \in \mathbb{C}^{n \times n}$ .
- Transposition is replaced by Hermitian adjoint, e.g.

$$\begin{bmatrix} 1 + i2 & 3 + i4 \\ 5 + i6 & 7 + i8 \end{bmatrix}^H = \begin{bmatrix} 1 - i2 & 5 - i6 \\ 3 - i4 & 7 - i8 \end{bmatrix}$$

- Denote the complex conjugate of a complex number  $\lambda$  by  $\bar{\lambda}$ .



Basic Concepts

Linear Transformations

Trace

Inner Product

Projection

Rank, Determinant, Trace

Matrix Inverse

Eigenvectors

Singular Value  
DecompositionDirectional Derivative,  
Gradient

Books

# Real Eigenvalues

- How can we enforce real eigenvalues?
- Let's assume  $A \in \mathbb{C}^{n \times n}$ , Hermitian ( $A^H = A$ ).
- Calculate

$$x^H A x = \lambda x^H x$$

for an eigenvector  $x \in \mathbb{C}^n$  of  $A$ .

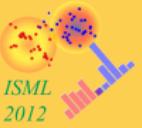
- Another possibility to calculate  $x^H A x$

$$\begin{aligned} x^H A x &= x^H A^H x && (A \text{ is Hermitian}) \\ &= (x^H A x)^H && (\text{reverse order}) \\ &= (\lambda x^H x)^H && (\text{eigenvalue}) \\ &= \bar{\lambda} x^H x \end{aligned}$$

- and therefore

$$\lambda = \bar{\lambda} \quad (\lambda \text{ is real}).$$

- If  $A$  is Hermitian, then all eigenvalues are real.
- Special case: If  $A$  has only real entries and is symmetric, then all eigenvalues are real.



ISML  
2012

Basic Concepts

Linear Transformations

Trace

Inner Product

Projection

Rank, Determinant, Trace

Matrix Inverse

Eigenvectors

Singular Value  
Decomposition

Directional Derivative,  
Gradient

Books

# Singular Value Decomposition

Every matrix  $A \in \mathbb{R}^{n \times p}$  can be decomposed into a product of three matrices

$$A = U\Sigma V^T$$

where  $U \in \mathbb{R}^{n \times n}$  and  $V \in \mathbb{R}^{p \times p}$  are orthogonal matrices ( $U^T U = I$  and  $V^T V = I$ ), and  $\Sigma \in \mathbb{R}^{n \times p}$  has nonnegative numbers on the diagonal.

*Basic Concepts**Linear Transformations**Trace**Inner Product**Projection**Rank, Determinant, Trace**Matrix Inverse**Eigenvectors**Singular Value  
Decomposition**Directional Derivative,  
Gradient**Books*

# How to calculate a gradient?

- Given a vector space  $\mathcal{V}$ , and a function  $f : \mathcal{V} \rightarrow \mathbb{R}$ .
- Example:  $X \in \mathbb{R}^{n \times p}$ , arbitrary  $C \in \mathbb{R}^{n \times n}$ ,

$$f(X) = \text{tr} \{ X^T C X \}.$$

- How to calculate the gradient of  $f$  ?
- Ill-defined question. There is no gradient in a vector space.
- Only a **Directional Derivative** at point  $X \in \mathcal{V}$  in direction  $\xi \in \mathcal{V}$ .

$$\mathcal{D}f(X)(\xi) = \lim_{h \rightarrow 0} \frac{f(X + h\xi) - f(X)}{h}$$

- For the example  $f(X) = \text{tr} \{ X^T C X \}$

$$\begin{aligned}\mathcal{D}f(X)(\xi) &= \lim_{h \rightarrow 0} \frac{\text{tr} \{ (X + h\xi)^T C (X + h\xi) \} - \text{tr} \{ X^T C X \}}{h} \\ &= \text{tr} \{ \xi^T C X + X^T C \xi \} = \text{tr} \{ X^T (C^T + C) \xi \}\end{aligned}$$

*Basic Concepts**Linear Transformations**Trace**Inner Product**Projection**Rank, Determinant, Trace**Matrix Inverse**Eigenvectors**Singular Value  
Decomposition**Directional Derivative,  
Gradient**Books*

# How to calculate a gradient?

- Given a vector space  $\mathcal{V}$ , and a function  $f : \mathcal{V} \rightarrow \mathbb{R}$ .
  - How to calculate the gradient of  $f$  ?
- ① Calculate the Directional Derivative
  - ② Define an **inner product**  $\langle A, B \rangle$  on the vector space
  - ③ The **gradient** is now defined as

$$\mathcal{D}f(X)(\xi) = \langle \text{grad } f, \xi \rangle$$

- For the example  $f(X) = \text{tr} \{ X^T C X \}$
- Define the inner product  $\langle A, B \rangle = \text{tr} \{ A^T B \}$  .
- Write the Directional Derivative as an inner product with  $\xi$

$$\begin{aligned}\mathcal{D}f(X)(\xi) &= \text{tr} \{ X^T (C^T + C) \xi \} \\ &= \langle (C + C^T) X, \xi \rangle = \langle \text{grad } f, \xi \rangle\end{aligned}$$



- Gilbert Strang, "Introduction to Linear Algebra", Wellesley Cambridge, 2009.
- David C. Lay, "Linear Algebra and Its Applications", Addison Wesley, 2005.

*Basic Concepts*

*Linear Transformations*

*Trace*

*Inner Product*

*Projection*

*Rank, Determinant, Trace*

*Matrix Inverse*

*Eigenvectors*

*Singular Value  
Decomposition*

*Directional Derivative,  
Gradient*

*Books*



## Part IV

# *Probability and Uncertainty*

*Boxes with Apples and  
Oranges*

*Bayes' Theorem*

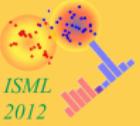
*Bayes' Probabilities*

*Probability Distributions*

*Gaussian Distribution  
over a Vector*

*Decision Theory*

*Model Selection - Key  
Ideas*



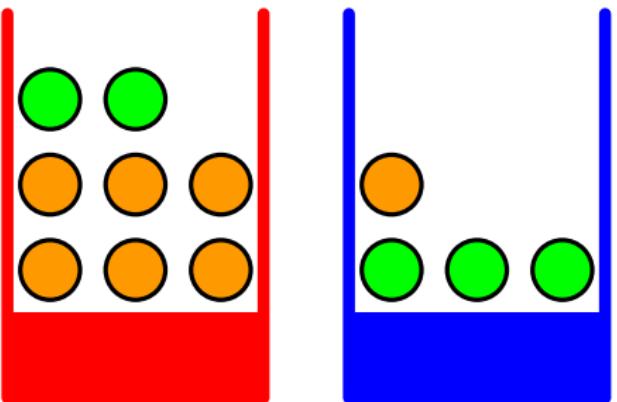
# Simple Experiment

1 Choose a box.

- Red box  $p(B = r) = 4/10$
- Blue box  $p(B = b) = 6/10$

2 Choose any item of the selected box with equal probability.

- Given that we have chosen an orange, what is the probability that the box we chose was the blue one?



Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

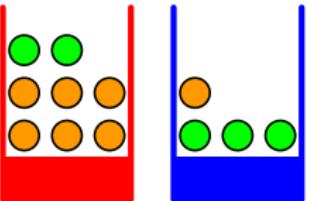
Model Selection - Key  
Ideas



# What do we know?

- $p(F = o | B = b) = 1/4$
- $p(F = a | B = b) = 3/4$
- $p(F = o | B = r) = 3/4$
- $p(F = a | B = r) = 1/4$
- Given that we have chosen an orange, what is the probability that the box we chose was the blue one?

$$p(B = b | F = o)?$$



Boxes with Apples and  
Oranges

Bayes' Theorem

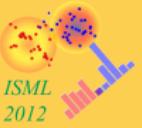
Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas

Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas

# Calculating the Posterior $p(B = b \mid F = o)$

- Bayes' Theorem

$$p(B = b \mid F = o) = \frac{p(F = o \mid B = b)p(B = b)}{p(F = o)}$$

- Sum Rule for the denominator

$$\begin{aligned} p(F = o) &= p(F = o, B = b) + p(F = o, B = r) \\ &= p(F = o \mid B = b)p(B = b) \\ &\quad + p(F = o \mid B = r)p(B = r) \\ &= \frac{1}{4} \times \frac{6}{10} + \frac{3}{4} \times \frac{4}{10} = \frac{9}{20} \end{aligned}$$



$$p(B = b \mid F = o) = \frac{1}{4} \times \frac{6}{10} \times \frac{20}{9} = \frac{1}{3}$$

Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

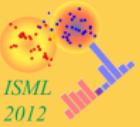
Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas

- Before choosing an item from a box: most complete information in  $p(B)$  (prior).
- Note, that in our example  $p(B = b) = \frac{6}{10}$ . Therefore choosing the box from the prior, we would opt for the blue box.
- Once we observe some data (e.g. choose an orange), we can calculate  $p(B = b | F = o)$  (posterior probability) via Bayes' theorem.
- After observing an orange, the posterior probability  $p(B = b | F = o) = \frac{1}{3}$  and therefore  $p(B = r | F = o) = \frac{2}{3}$ .
- Observing an orange it is now more likely that the orange came from the red box.



Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

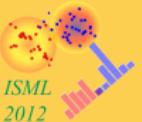
Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas

$$\underbrace{p(Y|X)}_{\text{posterior}} = \frac{\underbrace{p(X|Y)}_{\text{likelihood}} \underbrace{p(Y)}_{\text{prior}}}{\underbrace{p(X)}_{\text{normalisation}}} = \frac{p(X|Y)p(Y)}{\sum_Y p(X|Y)p(Y)}$$

# Bayes' Probabilities



- classical or frequentist interpretation of probabilities
- Bayesian view: probabilities represent uncertainty
- Example: Will the Arctic ice cap have disappeared by the end of the century?
- fresh evidence can change the opinion on ice loss
- goal: quantify uncertainty and revise uncertainty in light of new evidence
- use Bayesian interpretation of probability

Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas



Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas

# Andrey Kolmogorov - Axiomatic Probability Theory (1933)

- Let  $(\Omega, F, P)$  be a measure space with  $P(\Omega) = 1$ .
- Then  $(\Omega, F, P)$  is a probability space, with sample space  $\Omega$ , event space  $F$  and probability measure  $P$ .
- 1. Axiom  $P(E) \geq 0 \quad \forall E \in F$ .
- 2. Axiom  $P(\Omega) = 1$ .
- 3. Axiom  $P(E_1 \cup E_2 \cup \dots) = \sum_i P(E_i)$  for any countable sequence of pairwise disjoint events  $E_1, E_2, \dots$



Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas

- Assume numerical values are used to represent degrees of belief.
- Define a set of axioms encoding common sense properties of such beliefs.
- Results in a set of rules for manipulating degrees of belief which are equivalent to the sum and product rule of probability.
- many other authors have proposed different sets of axioms and properties
- Result: the numerical quantities behave all according to the rules of probability
- Denote these quantities as Bayesian probabilities.

Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas

# Curve fitting - revisited

- uncertainty about the parameter  $\mathbf{w}$  captured in the prior probability  $p(\mathbf{w})$
- observed data  $\mathcal{D} = \{t_1, \dots, t_N\}$
- calculate the uncertainty in  $\mathbf{w}$  **after** the data  $\mathcal{D}$  have been observed

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

- $p(\mathcal{D} | \mathbf{w})$  as a function of  $\mathbf{w}$  : **likelihood function**
- likelihood expresses how probable the data are for different values of  $\mathbf{w}$
- **not** a probability function over  $\mathbf{w}$



## Likelihood function $p(\mathcal{D} | \mathbf{w})$

### Frequentist Approach

- $\mathbf{w}$  considered fixed parameter
- value defined by some 'estimator'
- error bars on the estimated  $\mathbf{w}$  obtained from the distribution of possible data sets  $\mathcal{D}$

### Bayesian Approach

- only one single data set  $\mathcal{D}$
- uncertainty in the parameters comes from a probability distribution over  $\mathbf{w}$

Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas

# Frequentist Estimator - Maximum Likelihood



- choose  $\mathbf{w}$  for which the likelihood  $p(\mathcal{D} | \mathbf{w})$  is maximal
- choose  $\mathbf{w}$  for which the probability of the observed data is maximal
- Machine Learning: error function is negative log of likelihood function
- log is a monoton function
- maximising the likelihood  $\iff$  minimising the error
- Example: Fair-looking coin is tossed three times, always landing on heads.
- Maximum likelihood estimate of the probability of landing heads will give 1.

Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas



Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

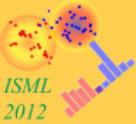
Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas

# Bayesian Approach

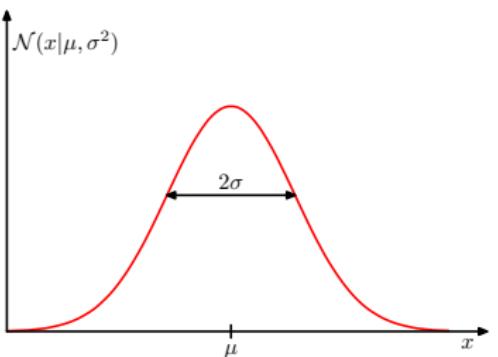
- including prior knowledge easy (via prior  $w$ )
- BUT: if prior is badly chosen, can lead to bad results
- subjective choice of prior
- sometimes choice of prior motivated by convenient mathematical form
- need to sum/integrate over the whole parameter space
- advances in sampling (Markov Chain Monte Carlo methods)
- advances in approximation schemes (Variational Bayes, Expectation Propagation)



# The Gaussian Distribution

- $x \in \mathbb{R}$
- Gaussian Distribution with mean  $\mu$  and variance  $\sigma^2$

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



Boxes with Apples and  
Oranges

Bayes' Theorem

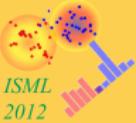
Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas

Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas

# The Gaussian Distribution

- $\mathcal{N}(x | \mu, \sigma^2) > 0$
- $\int_{-\infty}^{\infty} \mathcal{N}(x | \mu, \sigma^2) dx = 1$
- Expectation over  $x$

$$\mathbb{E}[x] = \int_{-\infty}^{\infty} \mathcal{N}(x | \mu, \sigma^2) x dx = \mu$$

- Expectation over  $x^2$

$$\mathbb{E}[x^2] = \int_{-\infty}^{\infty} \mathcal{N}(x | \mu, \sigma^2) x^2 dx = \mu^2 + \sigma^2$$

- Variance of  $x$

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

Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

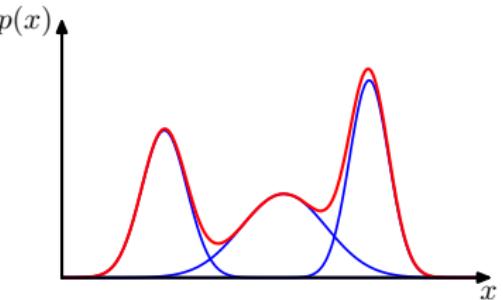
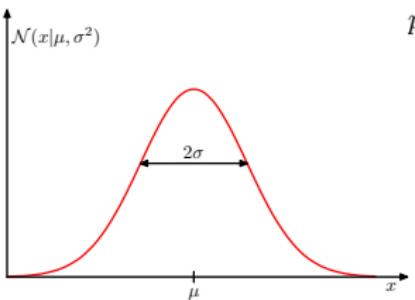
Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas

# The Mode of a Probability Distribution

- Mode of a distribution : the value that occurs the most frequently in a probability distribution.
- For a probability density function: the value  $x$  at which the probability density attains its maximum.
- Gaussian Distribution has **one** mode (unimodular); the mode is  $\mu$ .
- If there are multiple local maxima in the probability distribution, the probability distribution is called **multimodal** (example: mixture of three Gaussians).



Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas

# The Bernoulli Distribution

- Two possible outcomes  $x \in \{0, 1\}$  (e.g. coin which may be damaged).
- $p(x = 1 | \mu) = \mu$  for  $0 \leq \mu \leq 1$
- $p(x = 0 | \mu) = 1 - \mu$
- Bernoulli Distribution

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

- Expectation over  $x$

$$\mathbb{E}[x] = \mu$$

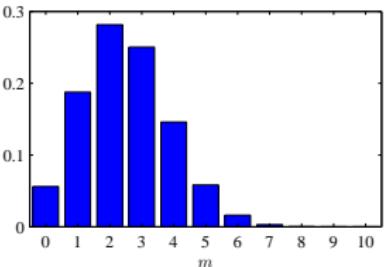
- Variance of  $x$

$$\text{var}[x] = \mu(1 - \mu)$$



- Flip a coin  $N$  times. What is the distribution to observe heads exactly  $m$  times?
- This is a distribution over  $m = \{0, \dots, N\}$ .
- Binomial Distribution

$$\text{Bin}(m | N, \mu) = \binom{N}{m} \mu^m (1 - \mu)^{N-m}$$



$$N = 10, \mu = 0.25$$

Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

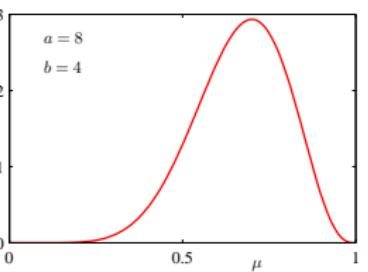
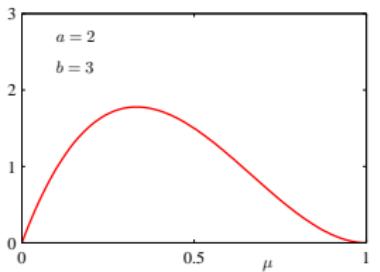
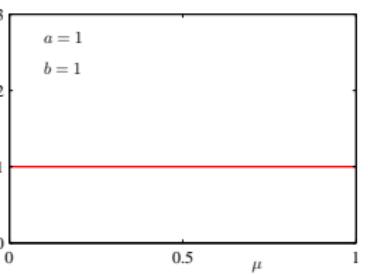
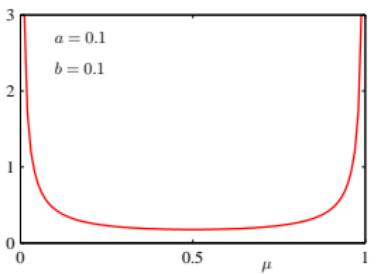
Model Selection - Key  
Ideas



# The Beta Distribution

## • Beta Distribution

$$\text{Beta}(\mu | a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \mu^{a-1} (1-\mu)^{b-1}$$



Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas

Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

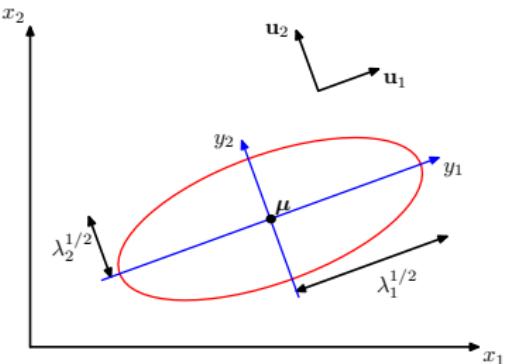
Model Selection - Key  
Ideas

# The Gaussian Distribution over a Vector $\mathbf{x}$

- $\mathbf{x} \in \mathbb{R}^D$
- Gaussian Distribution with mean  $\mu \in \mathbb{R}^D$  and covariance matrix  $\Sigma \in \mathbb{R}^{D \times D}$

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

where  $|\Sigma|$  is the determinant of  $\Sigma$ .



Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas

# The Gaussian Distribution over a vector $\mathbf{x}$

- Can find a linear transformation to a new coordinate system  $\mathbf{y}$  in which the  $\mathbf{x}$  becomes

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

- $\mathbf{U}$  is the eigenvector matrix for the covariance matrix  $\Sigma$  with eigenvalue matrix  $\mathbf{E}$

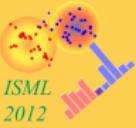
$$\Sigma \mathbf{U} = \mathbf{U} \mathbf{E} = \mathbf{U} \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \lambda_D \end{bmatrix}$$

- $\mathbf{U}$  can be made an orthogonal matrix, therefore the columns  $u_i$  of  $\mathbf{U}$  are unit vectors which are orthogonal to

each other  $u_i^T u_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$

- Now we can write  $\Sigma$  and its inverse (prove that  $\Sigma \Sigma^{-1} = \mathbf{I}$ )

$$\Sigma = \mathbf{U} \mathbf{E} \mathbf{U}^T = \sum_{i=1}^n \lambda_i u_i u_i^T \quad \Sigma^{-1} = \mathbf{U} \mathbf{E}^{-1} \mathbf{U}^T = \sum_{i=1}^n \frac{1}{\lambda_i} u_i u_i^T$$

Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas

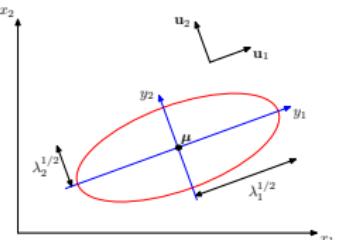
# The Gaussian Distribution over a vector $\mathbf{x}$

- Now use the linear transformation  $\mathbf{y} = \mathbf{U}^T (\mathbf{x} - \boldsymbol{\mu})$  and  $\Sigma^{-1} = \mathbf{U} \mathbf{E}^{-1} \mathbf{U}^T$  to transform the exponent  $(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})$  into

$$\mathbf{y}^T \mathbf{E} \mathbf{y} = \sum_{j=1}^n \frac{y_j^2}{\lambda_j}$$

- Now exponentiating the sum (and taking care of the factors) results in a product of scalar valued Gaussian distributions in orthogonal directions  $u_i$

$$p(\mathbf{y}) = \prod_{j=1}^D \frac{1}{(2\pi\lambda_j)^{1/2}} \exp\left\{-\frac{y_j^2}{2\lambda_j}\right\}$$



Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas

# Partitioned Gaussians

- Given a joint Gaussian distribution  $\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma})$ , where  $\boldsymbol{\mu}$  is the mean vector,  $\boldsymbol{\Sigma}$  the covariance matrix, and  $\boldsymbol{\Lambda} \equiv \boldsymbol{\Sigma}^{-1}$  the precision matrix
- Assume that the variables can be partitioned into two sets

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{pmatrix}, \boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{pmatrix}, \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{pmatrix}, \boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{pmatrix}$$

$$\boldsymbol{\Lambda}_{aa} = (\boldsymbol{\Sigma}_{aa} - \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}\boldsymbol{\Sigma}_{ba})^{-1}$$

$$\boldsymbol{\Lambda}_{ab} = -(\boldsymbol{\Sigma}_{aa} - \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}\boldsymbol{\Sigma}_{ba})^{-1}\boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}$$

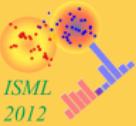
- Conditional distribution

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

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

- Marginal distribution

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

Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

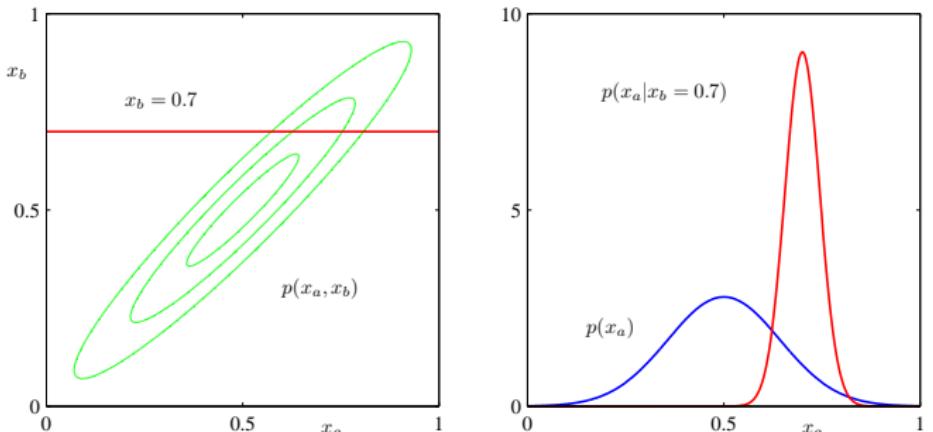
Probability Distributions

Gaussian Distribution  
over a Vector

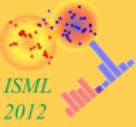
Decision Theory

Model Selection - Key  
Ideas

# Partitioned Gaussians



Contours of a Gaussian distribution over two variables  $x_a$  and  $x_b$  (left), and marginal distribution  $p(x_a)$  and conditional distribution  $p(x_a | x_b)$  for  $x_b = 0.7$  (right).



# Nonlinear Change of Variables in Distributions

- Given some  $p_x(x)$ .
- Consider a nonlinear change of variables

$$x = g(y)$$

- What is the new probability distribution  $p_y(y)$  in terms of the variable  $y$  ?

$$\begin{aligned} p_y(y) &= p_x(x) \left| \frac{dx}{dy} \right| \\ &= p_x(g(y)) |g'(y)| \end{aligned}$$

- For vector valued  $\mathbf{x}$  and  $\mathbf{y}$

$$p_y(\mathbf{y}) = p_x(\mathbf{x}) |\mathbf{J}|$$

where  $J_{ij} = \frac{\partial x_i}{\partial y_j}$ .

Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas

# Decision Theory - Key Ideas

©2012

Christfried Webers  
NICTA

The Australian National  
University



- Two classes  $\mathcal{C}_1$  and  $\mathcal{C}_2$
- joint distribution  $p(\mathbf{x}, \mathcal{C}_k)$
- using Bayes' theorem

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

- Example: cancer treatment ( $k = 2$ )
- data  $\mathbf{x}$  : an X-ray image
- $\mathcal{C}_1$  : patient has cancer ( $\mathcal{C}_1$  : patient has no cancer)
- $p(\mathcal{C}_1)$  is the prior probability of a person having cancer
- $p(\mathcal{C}_1 | \mathbf{x})$  is the posterior probability of a person having cancer after having seen the X-ray data

Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

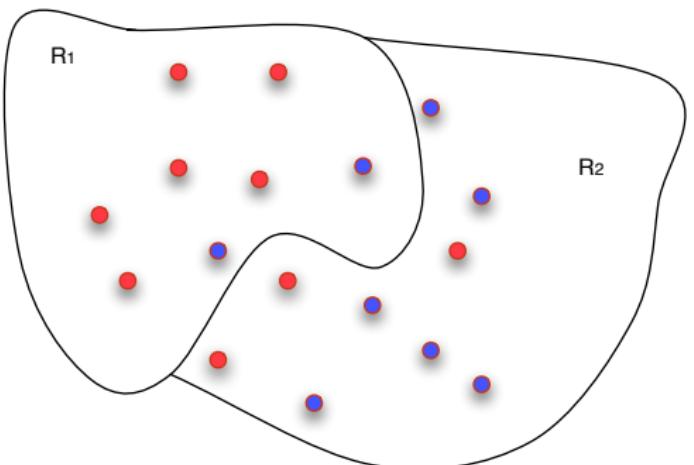
Decision Theory

Model Selection - Key  
Ideas

# Decision Theory - Key Ideas

- Need a rule which assigns each value of the input  $\mathbf{x}$  to one of the available classes.
- The input space is partitioned into decision regions  $\mathcal{R}_k$ .
- Leads to decision boundaries or decision surfaces
- probability of a mistake

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

Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

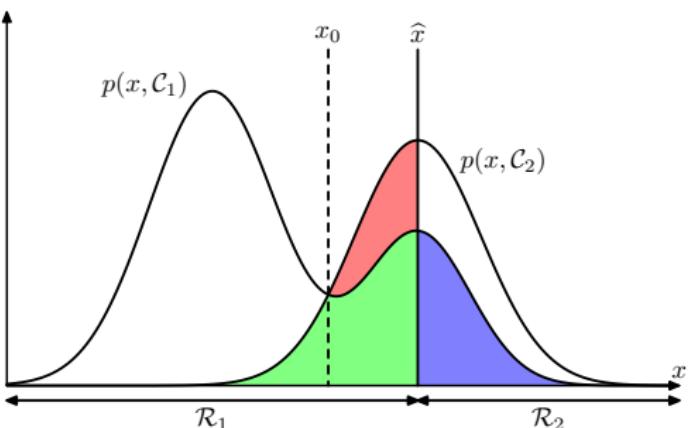
Model Selection - Key  
Ideas

# Decision Theory - Key Ideas

- probability of a mistake

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

- goal: minimize  $p(\text{mistake})$



Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

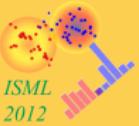
Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas

# Decision Theory - Key Ideas



- multiple classes
- instead of minimising the probability of mistakes, maximise the probability of correct classification

$$\begin{aligned} p(\text{correct}) &= \sum_{k=1}^K p(\mathbf{x} \in \mathcal{R}_k, \mathcal{C}_k) \\ &= \sum_{k=1}^K \int_{\mathcal{R}_k} p(\mathbf{x}, \mathcal{C}_k) \, d\mathbf{x} \end{aligned}$$

Boxes with Apples and  
Oranges

Bayes' Theorem

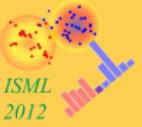
Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas



- Not all mistakes are equally costly.
- Weight each misclassification of  $\mathbf{x}$  to the wrong class  $\mathcal{C}_j$  instead of assigning it to the correct class  $\mathcal{C}_k$  by a factor  $L_{kj}$ .
- The expected loss is now

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

- Goal: minimize the expected loss  $\mathbb{E} [L]$

Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas

Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

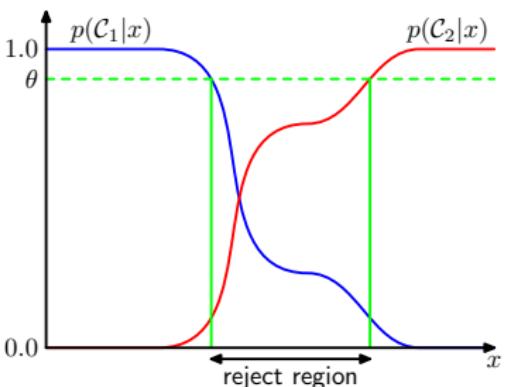
Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas

# The Reject Region

- Avoid making automated decisions on difficult cases.
- Difficult cases:
  - posterior probabilities  $p(\mathcal{C}_k | \mathbf{x})$  are very small
  - joint distributions  $p(\mathbf{x}, \mathcal{C}_k)$  have comparable values





Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

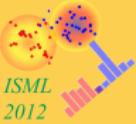
Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas

# S-fold Cross Validation

- Given a set of  $N$  data items and targets.
- Goal: Find the best model (type of model, number of parameters like the order  $p$  of the polynomial or the regularisation constant  $\lambda$ ). Avoid overfitting.
- Solution: Train a machine learning algorithm with some of the data, evaluate it with the rest.
- If we have many data
  - ➊ Train a range of models or a model with a range of parameters.
  - ➋ Compare the performance on an independent data set (**validation set**) and choose the one with the best predictive performance.
  - ➌ Still, overfitting to the validation set can occur. Therefore, use a third test set for final evaluation. (Keep the test set in a safe and never give it to the developers ;–)



Boxes with Apples and  
Oranges

Bayes' Theorem

Bayes' Probabilities

Probability Distributions

Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas

# S-fold Cross Validation

- For few data, there is a dilemma: Few training data or few test data.
- Solution is **cross-validation**.
- Use a portion of  $(S - 1)/S$  of the available data for training, but use all the data to asses the performance.
- For very scarce data one may use  $S = N$ , which is also called the **leave-one-out** technique.



Boxes with Apples and  
Oranges

Bayes' Theorem

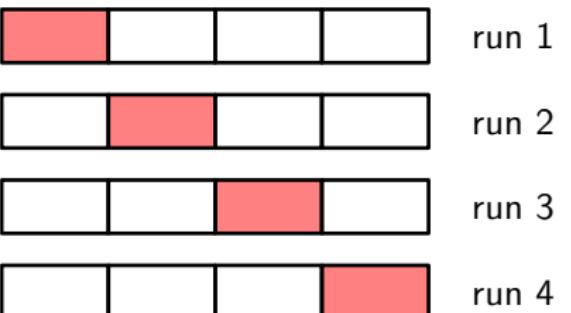
Bayes' Probabilities

Probability Distributions

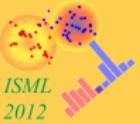
Gaussian Distribution  
over a Vector

Decision Theory

Model Selection - Key  
Ideas



Example for  $S = 4$ .



# Part V

## *Linear Regression 1*

*Linear Basis Function  
Models*

*Maximum Likelihood and  
Least Squares*

*Geometry of Least  
Squares*

*Sequential Learning*

*Regularized Least  
Squares*

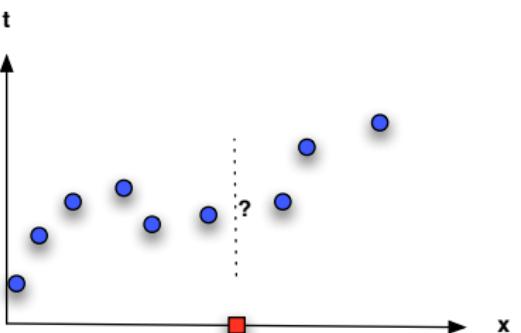
*Multiple Outputs*

*Loss Function for  
Regression*

*The Bias-Variance  
Decomposition*

# Regression

- Given a training data set of  $N$  observations  $\{\mathbf{x}_n\}$  and target values  $t_n$ .
- Goal : Learn to predict the value of one ore more target values  $t$  given a new value of the input  $\mathbf{x}$ .
- Example: Polynomial curve fitting (see Introduction).



Linear Basis Function  
Models

Maximum Likelihood and  
Least Squares

Geometry of Least  
Squares

Sequential Learning

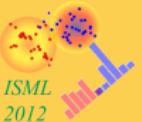
Regularized Least  
Squares

Multiple Outputs

Loss Function for  
Regression

The Bias-Variance  
Decomposition

# Supervised Learning



Linear Basis Function  
Models

Maximum Likelihood and  
Least Squares

Geometry of Least  
Squares

Sequential Learning

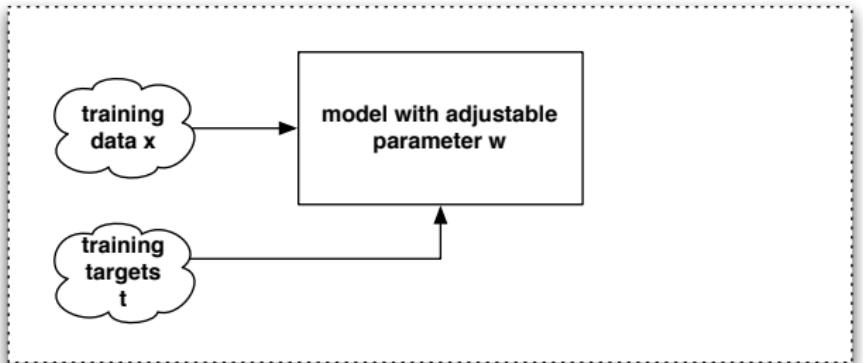
Regularized Least  
Squares

Multiple Outputs

Loss Function for  
Regression

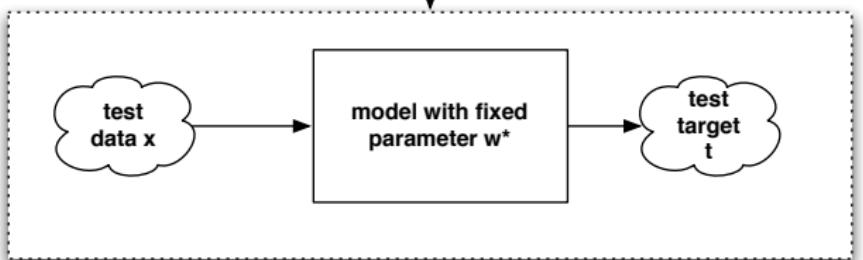
The Bias-Variance  
Decomposition

## Training Phase



fix the most appropriate  $w^*$

## Test Phase



# Linear Basis Function Models



- Linear combination of **fixed** nonlinear basis functions

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

- 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**

Linear Basis Function  
Models

Maximum Likelihood and  
Least Squares

Geometry of Least  
Squares

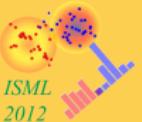
Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
Regression

The Bias-Variance  
Decomposition

Linear Basis Function  
ModelsMaximum Likelihood and  
Least SquaresGeometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

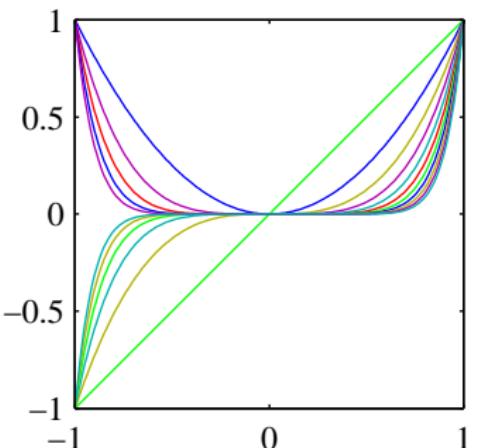
Loss Function for  
RegressionThe Bias-Variance  
Decomposition

# Polynomial Basis Functions

- Scalar input variable  $x$

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

- Limitation : Polynomials are global functions of the input variable  $x$ .
- Extension: Split the input space into regions and fit a different polynomial to each region (spline functions).

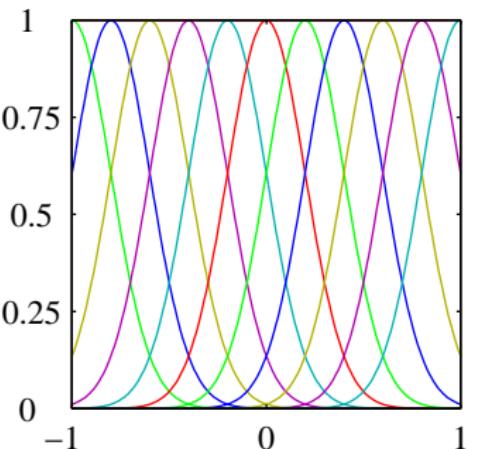


# 'Gaussian' Basis Functions

- 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  $w$ .



Linear Basis Function  
Models

Maximum Likelihood and  
Least Squares

Geometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
Regression

The Bias-Variance  
Decomposition



# Sigmoidal Basis Functions

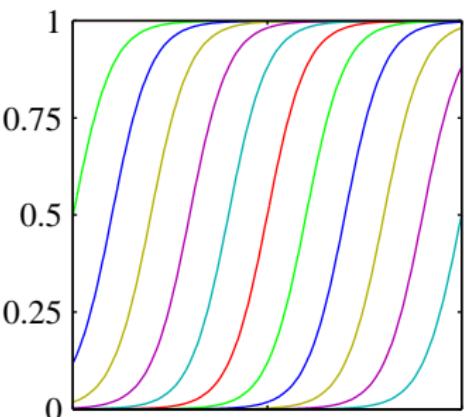
- Scalar input variable  $x$

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

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  
 $\tanh(a) = 2\sigma(a) - 1$ .



Linear Basis Function  
Models

Maximum Likelihood and  
Least Squares

Geometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
Regression

The Bias-Variance  
Decomposition

Linear Basis Function  
ModelsMaximum Likelihood and  
Least SquaresGeometry of Least  
Squares

Sequential Learning

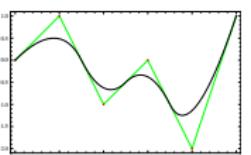
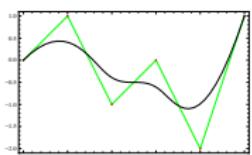
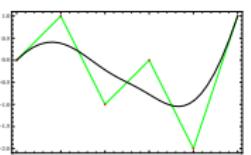
Regularized Least  
Squares

Multiple Outputs

Loss Function for  
RegressionThe Bias-Variance  
Decomposition

# 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 → conditions on the derivatives).

Linear  
SplinesQuadratic  
SplinesCubic  
SplinesQuartic  
Splines

Approximate the points

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



- No special assumption about the basis functions  $\phi_j(\mathbf{x})$ . In the simplest case, one can think of  $\phi_j(\mathbf{x}) = \mathbf{x}$ .
- Assume target  $t$  is given by

$$t = \underbrace{y(\mathbf{x}, \mathbf{w})}_{\text{deterministic}} + \underbrace{\epsilon}_{\text{noise}}$$

where  $\epsilon$  is a zero-mean Gaussian random variable with precision (inverse variance)  $\beta$ .

- Thus

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

Linear Basis Function  
Models

Maximum Likelihood and  
Least Squares

Geometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
Regression

The Bias-Variance  
Decomposition

Linear Basis Function  
ModelsMaximum Likelihood and  
Least SquaresGeometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
RegressionThe Bias-Variance  
Decomposition

# Maximum Likelihood and Least Squares

- Likelihood of one target  $t$  given the data  $\mathbf{x}$

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

- Set of inputs  $\mathbf{X}$  with corresponding target values  $\mathbf{t}$ .
- Assume data are **independent and identically distributed** (i.i.d.) (means : data are drawn independent and from the same distribution). The likelihood of the target  $\mathbf{t}$  is then

$$\begin{aligned} p(\mathbf{t} \mid \mathbf{X}, \mathbf{w}, \beta) &= \prod_{n=1}^N \mathcal{N}(t_n \mid y(\mathbf{x}_n, \mathbf{w}), \beta^{-1}) \\ &= \prod_{n=1}^N \mathcal{N}(t_n \mid \mathbf{w}^T \phi(\mathbf{x}_n), \beta^{-1}) \end{aligned}$$

- From now on drop the conditioning variable  $\mathbf{X}$  from the notation, as with supervised learning we do not seek to model the distribution of the input data.

Linear Basis Function  
ModelsMaximum Likelihood and  
Least SquaresGeometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
RegressionThe Bias-Variance  
Decomposition

# Maximum Likelihood and Least Squares

- Consider the **logarithm of the likelihood**  $p(\mathbf{t} | \mathbf{w}, \beta)$  (the logarithm is a monoton function! )

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

where the **sum-of-squares error function** is

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

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



- Goal: Find a more compact representation.
- Rewrite the **error function**

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(x_n)\}^2 = \frac{1}{2} (\mathbf{t} - \Phi \mathbf{w})^T (\mathbf{t} - \Phi \mathbf{w})$$

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

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

Linear Basis Function  
Models

Maximum Likelihood and  
Least Squares

Geometry of Least  
Squares

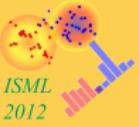
Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
Regression

The Bias-Variance  
Decomposition

Linear Basis Function  
ModelsMaximum Likelihood and  
Least SquaresGeometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
RegressionThe Bias-Variance  
Decomposition

# Maximum Likelihood and Least Squares

- The log likelihood is now

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

- Find critical points of  $\ln p(\mathbf{t} | \mathbf{w}, \beta)$ .
- Directional derivative in direction  $\xi$

$$\mathcal{D} \ln p(\mathbf{t} | \mathbf{w}, \beta)(\xi) = \beta \xi^T (\Phi^T \mathbf{t} - \Phi^T \Phi \mathbf{w})$$

shall be zero in all directions  $\xi$ . Therefore

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

- which results in

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

where  $\Phi^\dagger$  is the Moore-Penrose pseudo-inverse of the matrix  $\Phi$ .



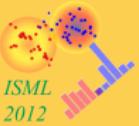
# Maximum Likelihood and Least Squares

- Found a critical point  $\mathbf{w}_{ML}$  for  $\ln p(\mathbf{t} | \mathbf{w}, \beta)$ . Is it a maximum, a minimum, or a saddle point?
- Calculate the second directional derivative

$$\begin{aligned}\mathcal{D}^2 \ln p(\mathbf{t} | \mathbf{w}, \beta)(\boldsymbol{\xi}, \boldsymbol{\xi}) &= -\beta \boldsymbol{\xi}^T \boldsymbol{\Phi}^T \boldsymbol{\Phi} \boldsymbol{\xi} \\ &= -\beta (\boldsymbol{\Phi} \boldsymbol{\xi})^T \boldsymbol{\Phi} \boldsymbol{\xi} \\ &= -\beta \|\boldsymbol{\Phi} \boldsymbol{\xi}\|^2 \leq 0\end{aligned}$$

- We found a maximum.
- (Is it enough to check the second directional derivative in two directions  $\boldsymbol{\xi}$  which are the same? Yes, because for any bilinear function  $g(\boldsymbol{\xi}, \boldsymbol{\eta})$ , symmetric in its arguments,

$$\begin{aligned}&\frac{1}{2} [g(\boldsymbol{\xi}, \boldsymbol{\xi}) + g(\boldsymbol{\eta}, \boldsymbol{\eta}) - g(\boldsymbol{\xi} - \boldsymbol{\eta}, \boldsymbol{\xi} - \boldsymbol{\eta})] \\&= \frac{1}{2} [g(\boldsymbol{\xi}, \boldsymbol{\xi}) + g(\boldsymbol{\eta}, \boldsymbol{\eta}) - g(\boldsymbol{\xi}, \boldsymbol{\xi}) - g(\boldsymbol{\eta}, \boldsymbol{\eta}) + g(\boldsymbol{\xi}, \boldsymbol{\eta}) + g(\boldsymbol{\eta}, \boldsymbol{\xi})] \\&= \frac{1}{2} [g(\boldsymbol{\xi}, \boldsymbol{\eta}) + g(\boldsymbol{\eta}, \boldsymbol{\xi})] = g(\boldsymbol{\xi}, \boldsymbol{\eta})\end{aligned}$$

Linear Basis Function  
ModelsMaximum Likelihood and  
Least SquaresGeometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
RegressionThe Bias-Variance  
Decomposition

# Maximum Likelihood and Least Squares

- The log likelihood with the optimal  $\mathbf{w}_{ML}$  is now

$$\ln p(\mathbf{t} | \mathbf{w}_{ML}, \beta)$$

$$= \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta \frac{1}{2} (\mathbf{t} - \Phi \mathbf{w}_{ML})^T (\mathbf{t} - \Phi \mathbf{w}_{ML})$$

- Find critical points of  $\ln p(\mathbf{t} | \mathbf{w}, \beta)$  wrt  $\beta$ ,

$$\frac{\partial \ln p(\mathbf{t} | \mathbf{w}_{ML}, \beta)}{\partial \beta} = 0$$

results in

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

- Note: We can first find the maximum likelihood for  $\mathbf{w}$  as this does **not depend** on  $\beta$ . Then we can use  $\mathbf{w}_{ML}$  to find the maximum likelihood solution for  $\beta$ .
- Could we have chosen optimisation wrt  $\beta$  first, and then wrt to  $\mathbf{w}$  ?

Linear Basis Function  
ModelsMaximum Likelihood and  
Least SquaresGeometry of Least  
Squares

Sequential Learning

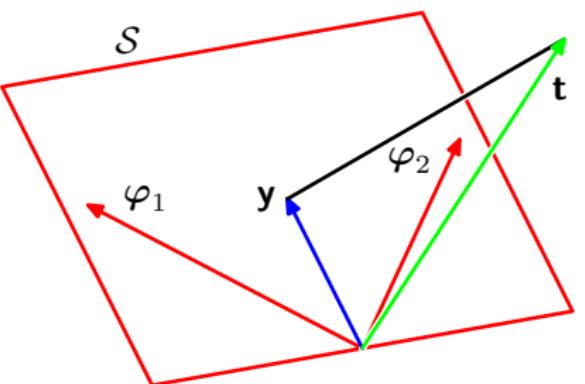
Regularized Least  
Squares

Multiple Outputs

Loss Function for  
RegressionThe Bias-Variance  
Decomposition

# Geometry of Least Squares

- Target vector  $\mathbf{t} = (t_1, \dots, t_N)^T \in \mathbb{R}^N$
- Basis vectors  $\varphi_j = (\Phi_{j1}, \dots, \Phi_{jN})^T = (\phi_j(\mathbf{x}_1), \dots, \phi_j(\mathbf{x}_N))^T \in \mathbb{R}^N$  span a subspace  $\mathcal{S}$
- Find  $\mathbf{w}$  such that  $\mathbf{y} = (y(\mathbf{x}_1, \mathbf{w}), \dots, y(\mathbf{x}_N, \mathbf{w}))^T \in \mathcal{S}$  is closest to  $\mathbf{t}$ .



# Sequential Learning - Stochastic Gradient Descent



- For large data sets, calculating the maximum likelihood parameters  $\mathbf{w}_{ML}$  and  $\beta_{ML}$  may be costly.
- For real-time 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

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n,$$

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

Linear Basis Function  
Models

Maximum Likelihood and  
Least Squares

Geometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
Regression

The Bias-Variance  
Decomposition

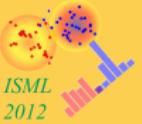
# Sequential Learning - Stochastic Gradient Descent

Introduction to Statistical  
Machine Learning

©2012

Christfried Webers  
NICTA

The Australian National  
University



Linear Basis Function  
Models

Maximum Likelihood and  
Least Squares

Geometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
Regression

The Bias-Variance  
Decomposition

- For the sum-of-squares error function, stochastic gradient descent results in

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

- The value for the learning rate must be chosen carefully. A too large learning rate may prevent the algorithm from converging. A too small learning rate does not follow the data too slowly.



# Regularized Least Squares

- Add regularisation in order to prevent overfitting

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

with regularisation coefficient  $\lambda$ .

- Simple quadratic regulariser

$$E_W(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$$

- Maximum likelihood solution

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

Linear Basis Function  
Models

Maximum Likelihood and  
Least Squares

Geometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
Regression

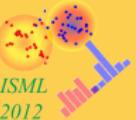
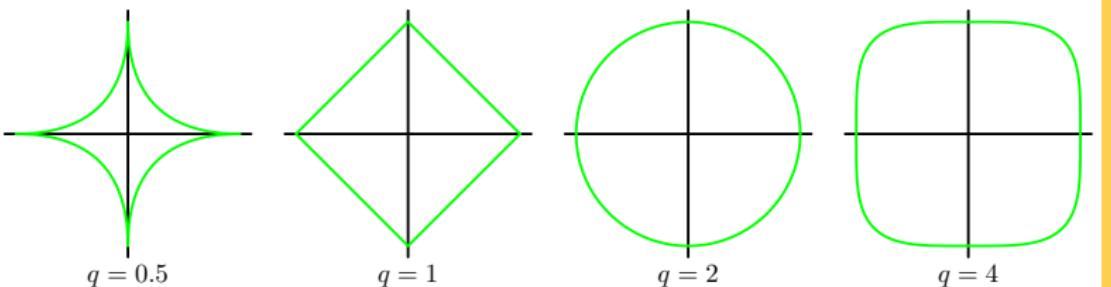
The Bias-Variance  
Decomposition

# Regularized Least Squares

- More general regulariser

$$E_W(\mathbf{w}) = \frac{1}{2} \sum_{j=1}^M |w_j|^q$$

- $q = 1$  (lasso) leads to a sparse model if  $\lambda$  large enough.



Linear Basis Function  
Models

Maximum Likelihood and  
Least Squares

Geometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
Regression

The Bias-Variance  
Decomposition

# Comparison of Quadratic and Lasso Regulariser



Linear Basis Function  
Models

Maximum Likelihood and  
Least Squares

Geometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

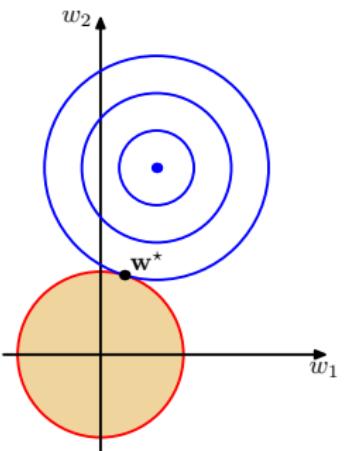
Loss Function for  
Regression

The Bias-Variance  
Decomposition

Assume a sufficiently large regularisation coefficient  $\lambda$ .

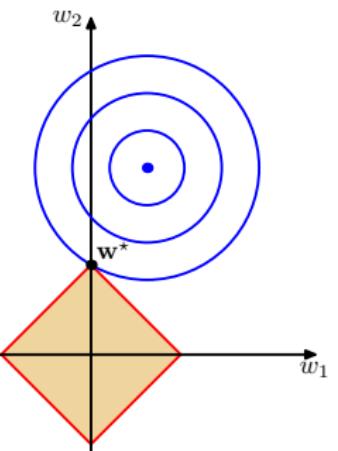
Quadratic regulariser

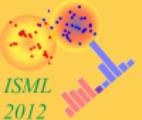
$$\frac{1}{2} \sum_{j=1}^M w_j^2$$



Lasso regulariser

$$\frac{1}{2} \sum_{j=1}^M |w_j|$$



Linear Basis Function  
ModelsMaximum Likelihood and  
Least SquaresGeometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
RegressionThe Bias-Variance  
Decomposition

# Multiple Outputs

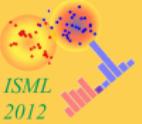
- More than 1 target variable per data point.
- $\mathbf{y}$  becomes a vector instead of a scalar. Each dimension can be treated with a different set of basis functions (and that may be necessary if the data in the different target dimensions represent very different types of information.)
- Here we restrict ourselves to the SAME basis functions

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

where  $\mathbf{y}$  is a  $K$ -dimensional column vector,  $\mathbf{W}^T$  is an  $M \times K$  matrix of model parameters, and

$\phi(\mathbf{x}) = (\phi_0(\mathbf{x}), \dots, \phi_{M-1}(\mathbf{x}), \phi_0(\mathbf{x}) = 1)$ , as before.

- Define target matrix  $\mathbf{T}$  containing the target vector  $\mathbf{t}_n^T$  in the  $n^{th}$  row.



- Suppose the conditional distribution of the target vector is an isotropic Gaussian of the form

$$p(\mathbf{t} \mid \mathbf{x}, \mathbf{W}, \beta) = \mathcal{N}(\mathbf{t} \mid \mathbf{W}^T \phi(\mathbf{x}), \beta^{-1} \mathbf{I}).$$

- The log likelihood is then

$$\begin{aligned}\ln p(\mathbf{T} \mid \mathbf{X}, \mathbf{W}, \beta) &= \sum_{n=1}^N \ln \mathcal{N}(\mathbf{t}_n \mid \mathbf{W}^T \phi(\mathbf{x}_n), \beta^{-1} \mathbf{V} \mathbf{I}) \\ &= \frac{NK}{2} \ln \left( \frac{\beta}{2\pi} \right) - \frac{\beta}{2} \sum_{n=1}^N \|\mathbf{t}_n - \mathbf{W}^T \phi(\mathbf{x}_n)\|^2\end{aligned}$$

Linear Basis Function  
ModelsMaximum Likelihood and  
Least SquaresGeometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
RegressionThe Bias-Variance  
Decomposition

# Multiple Outputs



# Multiple Outputs

- Maximisation with respect to  $\mathbf{W}$  results in

$$\mathbf{W}_{ML} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{T}.$$

- For each target variable  $\mathbf{t}_k$ , we get

$$\mathbf{w}_k = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{t}_k = \Phi^\dagger \mathbf{t}_k.$$

- The solution between the different target variables decouples.
- Holds also for a general Gaussian noise distribution with arbitrary covariance matrix.
- Why?  $\mathbf{W}$  defines the mean of the Gaussian noise distribution. And the maximum likelihood solution for the mean of a multivariate Gaussian is independent of the covariance.

Linear Basis Function  
Models

Maximum Likelihood and  
Least Squares

Geometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
Regression

The Bias-Variance  
Decomposition



- Over-fitting results from a large number of basis functions and a relatively small training set.
- Regularisation can prevent overfitting, but how to find the correct value for the regularisation constant  $\lambda$  ?
- Frequentists viewpoint of the model complexity is the **bias-variance** trade-off.

Linear Basis Function  
Models

Maximum Likelihood and  
Least Squares

Geometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
Regression

The Bias-Variance  
Decomposition

Linear Basis Function  
ModelsMaximum Likelihood and  
Least SquaresGeometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
RegressionThe Bias-Variance  
Decomposition

# Loss Function for Regression

- Choose an estimator  $y(\mathbf{x})$  to estimate the target value  $t$  for each input  $\mathbf{x}$ .
- Choose a loss function  $L(t, y(\mathbf{x}))$  which measures the difference between the target  $t$  and the estimate  $y(\mathbf{x})$ .
- The **expected loss** is then

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

- Common choice: **Squared Loss**

$$L(t, y(\mathbf{x})) = \{y(\mathbf{x}) - t\}^2.$$

- Expected loss for squared loss function

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



- Expected loss for squared loss function

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

- Minimise  $\mathbb{E} [L]$  by choosing the regression function

$$y(\mathbf{x}) = \frac{\int t p(\mathbf{x}, t) \, dt}{p(\mathbf{x})} = \int t p(t \mid \mathbf{x}) \, dt = \mathbb{E}_t [t \mid \mathbf{x}]$$

(use calculus of variations to derive this result).

Linear Basis Function  
Models

Maximum Likelihood and  
Least Squares

Geometry of Least  
Squares

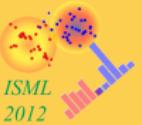
Sequential Learning

Regularized Least  
Squares

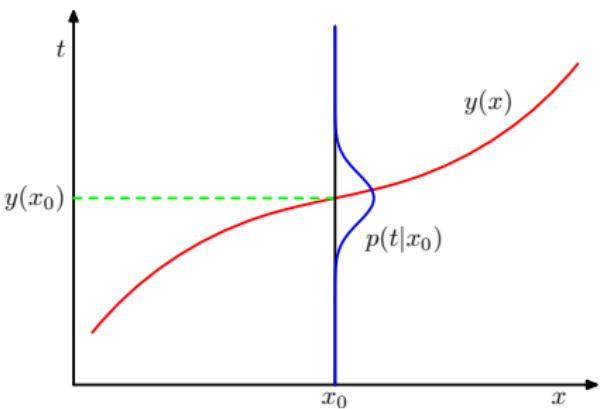
Multiple Outputs

Loss Function for  
Regression

The Bias-Variance  
Decomposition



- The regression function which minimises the expected squared loss, is given by the mean of the conditional distribution  $p(t | \mathbf{x})$ .



Linear Basis Function  
Models

Maximum Likelihood and  
Least Squares

Geometry of Least  
Squares

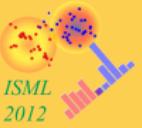
Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
Regression

The Bias-Variance  
Decomposition

Linear Basis Function  
ModelsMaximum Likelihood and  
Least SquaresGeometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
RegressionThe Bias-Variance  
Decomposition

# Loss Function for Regression

- Analyse the expected loss

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

- Rewrite the squared loss

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

- Claim

$$\int \int \{y(\mathbf{x}) - \mathbb{E}[t | \mathbf{x}]\} \{\mathbb{E}[t | \mathbf{x}] - t\} p(\mathbf{x}, t) \, d\mathbf{x} \, dt = 0.$$



# Loss Function for Regression

- Claim

$$\int \int \{y(\mathbf{x}) - \mathbb{E}[t | \mathbf{x}]\} \{\mathbb{E}[t | \mathbf{x}] - t\} p(\mathbf{x}, t) d\mathbf{x} dt = 0.$$

- Separate functions depending on  $t$  from function depending on  $\mathbf{x}$

$$\int \{y(\mathbf{x}) - \mathbb{E}[t | \mathbf{x}]\} \left( \int \{\mathbb{E}[t | \mathbf{x}] - t\} p(\mathbf{x}, t) dt \right) d\mathbf{x}$$

- Calculate the integral over  $t$

$$\begin{aligned} \int \{\mathbb{E}[t | \mathbf{x}] - t\} p(\mathbf{x}, t) dt &= \mathbb{E}[t | \mathbf{x}] p(\mathbf{x}) - p(\mathbf{x}) \int \frac{t p(\mathbf{x}, t)}{p(\mathbf{x})} dt \\ &= \mathbb{E}[t | \mathbf{x}] p(\mathbf{x}) - p(\mathbf{x}) \mathbb{E}[t | \mathbf{x}] \\ &= 0 \end{aligned}$$

Linear Basis Function  
Models

Maximum Likelihood and  
Least Squares

Geometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
Regression

The Bias-Variance  
Decomposition



- The expected loss is now

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

- Minimise first term by choosing appropriate  $y(\mathbf{x})$ .
- Second term represents the intrinsic variability of the target data (can be regarded as noise). Independent of the choice  $y(\mathbf{x})$ , can not be reduced by learning a better  $y(\mathbf{x})$ .

Linear Basis Function  
Models

Maximum Likelihood and  
Least Squares

Geometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
Regression

The Bias-Variance  
Decomposition

Linear Basis Function  
ModelsMaximum Likelihood and  
Least SquaresGeometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
RegressionThe Bias-Variance  
Decomposition

# The Bias-Variance Decomposition

- Consider now the dependency on the data set  $\mathcal{D}$ .
- Prediction function now  $y(\mathbf{x}; \mathcal{D})$ .
- Consider again squared loss for which the optimal prediction is given by the conditional expectation  $h(\mathbf{x})$

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

- BUT: we can not know  $h(x)$  exactly, as we would need an infinite number of training data to learn it accurately.
- Evaluate performance of algorithm by taking the expectation  $\mathbb{E}_{\mathcal{D}} [L]$  over all data sets  $\mathcal{D}$

Linear Basis Function  
ModelsMaximum Likelihood and  
Least SquaresGeometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
RegressionThe Bias-Variance  
Decomposition

# The Bias-Variance Decomposition

- Taking the expectation over all data sets  $\mathcal{D}$

$$\begin{aligned}\mathbb{E}_{\mathcal{D}} [\mathbb{E} [L]] &= \int \mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2] p(\mathbf{x}) d\mathbf{x} \\ &\quad + \int \int \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt\end{aligned}$$

- Again, add and subtract the expectation  $\mathbb{E}_{\mathcal{D}} [y(\mathbf{x}; \mathcal{D})]$

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

and show that the mixed term does vanish under the expectation  $\mathbb{E}_{\mathcal{D}} [\dots]$ .

Linear Basis Function  
ModelsMaximum Likelihood and  
Least SquaresGeometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
RegressionThe Bias-Variance  
Decomposition

# The Bias-Variance Decomposition

- Expected loss  $\mathbb{E}_{\mathcal{D}} [L]$  over all data sets  $\mathcal{D}$

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

where

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

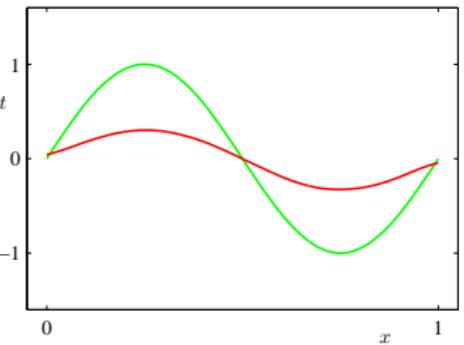
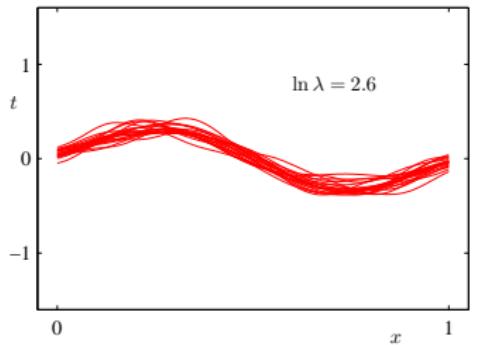
$$\text{variance} = \int \mathbb{E}_{\mathcal{D}} \left[ \{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}} [y(\mathbf{x}; \mathcal{D})]\}^2 \right] p(\mathbf{x}) d\mathbf{x}$$

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

- squared bias** : how much does the average prediction over all data sets differ from the desired regression function ?
- variance** : how much do solutions for individual data sets vary around their average ?

# The Bias-Variance Decomposition

Dependence of bias and variance on the model complexity



Left: Result of fitting the model to 100 data sets, only 25 shown.  
Right: Average of the 100 fits in red, the sinusoidal function from where the data were created in green.

Linear Basis Function  
Models

Maximum Likelihood and  
Least Squares

Geometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
Regression

The Bias-Variance  
Decomposition

Linear Basis Function  
ModelsMaximum Likelihood and  
Least SquaresGeometry of Least  
Squares

Sequential Learning

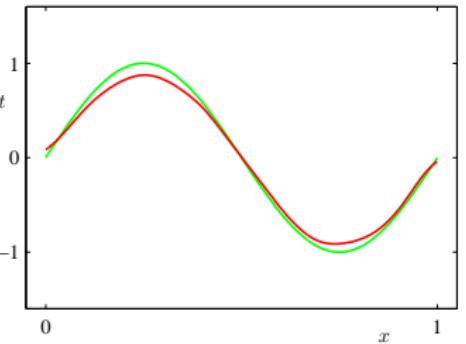
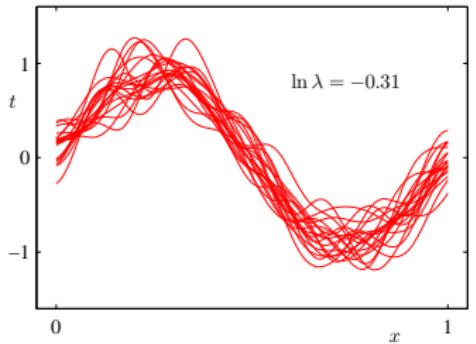
Regularized Least  
Squares

Multiple Outputs

Loss Function for  
RegressionThe Bias-Variance  
Decomposition

# The Bias-Variance Decomposition

Dependence of bias and variance on the model complexity



Left: Result of fitting the model to 100 data sets, only 25 shown.  
Right: Average of the 100 fits in red, the sinusoidal function from where the data were created in green.

Linear Basis Function  
ModelsMaximum Likelihood and  
Least SquaresGeometry of Least  
Squares

Sequential Learning

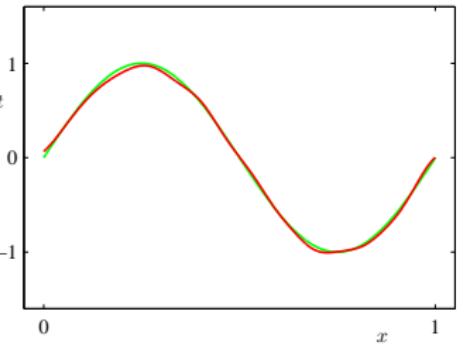
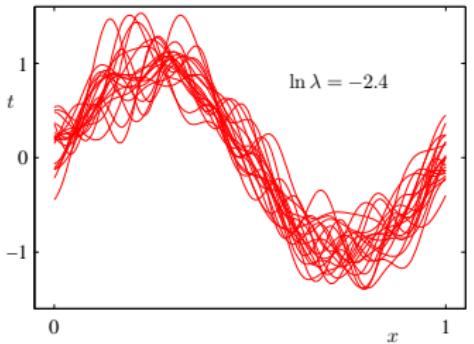
Regularized Least  
Squares

Multiple Outputs

Loss Function for  
RegressionThe Bias-Variance  
Decomposition

# The Bias-Variance Decomposition

Dependence of bias and variance on the model complexity

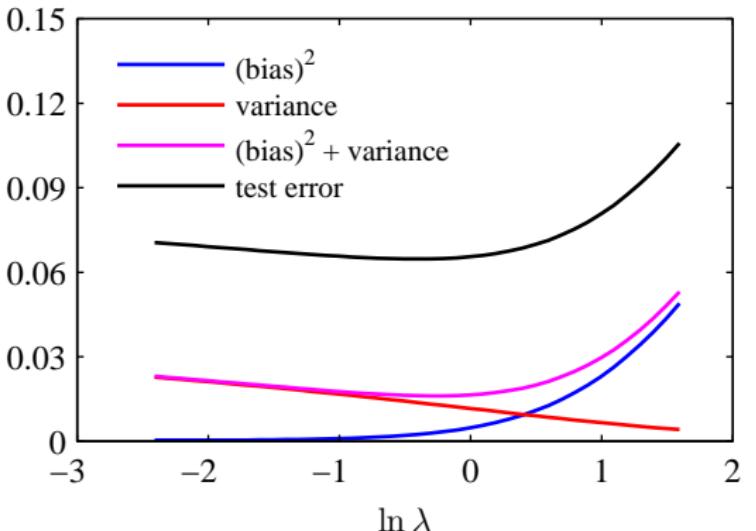


Left: Result of fitting the model to 100 data sets, only 25 shown.  
Right: Average of the 100 fits in red, the sinusoidal function  
from where the data were created in green.



# The Bias-Variance Decomposition

- 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



Linear Basis Function  
Models

Maximum Likelihood and  
Least Squares

Geometry of Least  
Squares

Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
Regression

The Bias-Variance  
Decomposition

# The Bias-Variance Decomposition



- Tradeoff between bias and variance
  - simple models have high bias and low variance
  - complex models have low bias and high variance
- The sum of bias and variance has a minimum at some model complexity.
- The bias-variance decomposition needs many data sets, which are not always available.

Linear Basis Function  
Models

Maximum Likelihood and  
Least Squares

Geometry of Least  
Squares

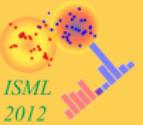
Sequential Learning

Regularized Least  
Squares

Multiple Outputs

Loss Function for  
Regression

The Bias-Variance  
Decomposition



## Part VI

# *Linear Regression 2*

*Bayesian Regression*

*Sequential Update of the  
Posterior*

*Predictive Distribution*

*Proof of the Predictive  
Distribution*

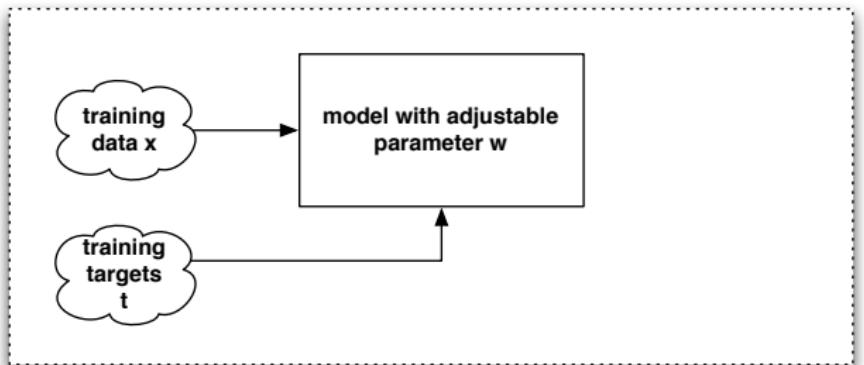
*Predictive Distribution  
with Simplified Prior*

*Limitations of Linear  
Basis Function Models*

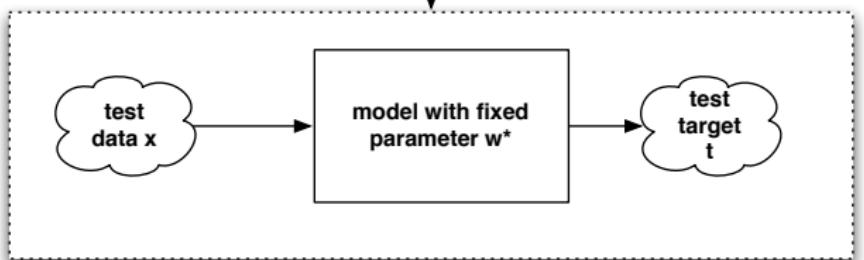
# Bayesian Regression



## Training Phase



## Test Phase



Bayesian Regression

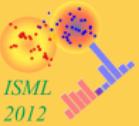
Sequential Update of the  
Posterior

Predictive Distribution

Proof of the Predictive  
Distribution

Predictive Distribution  
with Simplified Prior

Limitations of Linear  
Basis Function Models



# Bayesian Regression

- Bayes Theorem

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{normalisation}}$$

$$p(\mathbf{w} | \mathbf{t}) = \frac{p(\mathbf{t} | \mathbf{w}) p(\mathbf{w})}{p(\mathbf{t})}$$

- likelihood for i.i.d. data

$$\begin{aligned} p(\mathbf{t} | \mathbf{w}) &= \prod_{n=1}^N \mathcal{N}(t_n | y(\mathbf{x}_n, \mathbf{w}), \beta^{-1}) \\ &= \prod_{n=1}^N \mathcal{N}(t_n | \mathbf{w}^T \phi(\mathbf{x}_n), \beta^{-1}) \\ &= \text{const} \times \exp\left\{-\beta \frac{1}{2} (\mathbf{t} - \Phi \mathbf{w})^T (\mathbf{t} - \Phi \mathbf{w})\right\} \end{aligned}$$

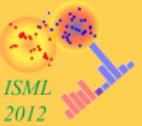
where we left out the conditioning on  $\mathbf{x}$  (always assumed), and  $\beta$ , which is assumed to be constant.

Bayesian Regression

Sequential Update of the  
Posterior

Predictive Distribution

Proof of the Predictive  
DistributionPredictive Distribution  
with Simplified PriorLimitations of Linear  
Basis Function Models



Bayesian Regression

Sequential Update of the  
Posterior

Predictive Distribution

Proof of the Predictive  
Distribution

Predictive Distribution  
with Simplified Prior

Limitations of Linear  
Basis Function Models

# How to choose a prior?

- Can we find a prior for the given likelihood which
  - makes sense for the problem at hand
  - allows us to find a posterior in a 'nice' form

An answer to the second question:

## 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  $p(w)$ .



Bayesian Regression

Sequential Update of the  
Posterior

Predictive Distribution

Proof of the Predictive  
Distribution

Predictive Distribution  
with Simplified Prior

Limitations of Linear  
Basis Function Models

# Examples of Conjugate Prior Distributions

Table : Discrete likelihood distributions

Likelihood	Conjugate Prior
Bernoulli	Beta
Binomial	Beta
Poisson	Gamma
Multinomial	Dirichlet

Table : Continuous likelihood distributions

Likelihood	Conjugate Prior
Uniform	Pareto
Exponential	Gamma
Normal	Normal
Multivariate normal	Multivariate normal

# Conjugate Prior to a Gaussian Distribution



Bayesian Regression

Sequential Update of the  
Posterior

Predictive Distribution

Proof of the Predictive  
Distribution

Predictive Distribution  
with Simplified Prior

Limitations of Linear  
Basis Function Models

- Example : The Gaussian family is conjugate to itself with respect to a Gaussian likelihood function: if the likelihood function is Gaussian, choosing a Gaussian prior will ensure that the posterior distribution is also Gaussian.
- Given a marginal distribution for  $\mathbf{x}$  and a conditional Gaussian distribution for  $\mathbf{y}$  given  $\mathbf{x}$  in the form

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

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

- we get

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

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

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



Bayesian Regression

Sequential Update of the  
Posterior

Predictive Distribution

Proof of the Predictive  
DistributionPredictive Distribution  
with Simplified PriorLimitations of Linear  
Basis Function Models

# Bayesian Regression

- Choose a Gaussian prior with mean  $\mathbf{m}_0$  and covariance  $\mathbf{S}_0$

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

- After having seen  $N$  training data pairs  $(\mathbf{x}_n, t_n)$ , the posterior for the given likelihood is now

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

where

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

$$\mathbf{S}_N^{-1} = \mathbf{S}_0^{-1} + \beta \Phi^T \Phi$$

- The posterior is Gaussian, therefore mode = mean.
- The maximum posterior weight vector  $\mathbf{w}_{MAP} = \mathbf{m}_N$ .
- Assume infinitely broad prior  $\mathbf{S}_0 = \alpha^{-1} \mathbf{I}$  with  $\alpha \rightarrow 0$ , the mean reduces to the maximum likelihood  $\mathbf{w}_{ML}$ .



- If we have not yet seen any data point ( $N = 0$ ), the posterior is equal to the prior.
- Sequential arrival of data points : Each posterior distribution calculated after the arrival of a data point and target value, acts as the prior distribution for the subsequent data point.
- Nicely fits a sequential learning framework.

Bayesian Regression

Sequential Update of the  
Posterior

Predictive Distribution

Proof of the Predictive  
Distribution

Predictive Distribution  
with Simplified Prior

Limitations of Linear  
Basis Function Models



Bayesian Regression

Sequential Update of the  
Posterior

Predictive Distribution

Proof of the Predictive  
DistributionPredictive Distribution  
with Simplified PriorLimitations of Linear  
Basis Function Models

# Bayesian Regression

- Special simplified prior in the remainder,  $\mathbf{m}_0 = 0$  and  $\mathbf{S}_0 = \alpha^{-1} \mathbf{I}$ ,

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

- The parameters of the posterior distribution  $p(\mathbf{w} | \mathbf{t}) = \mathcal{N}(\mathbf{w} | \mathbf{m}_n, \mathbf{S}_N)$  are now

$$\mathbf{m}_N = \beta \mathbf{S}_N \Phi^T \mathbf{t}$$

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

- For  $\alpha \rightarrow 0$  we get

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

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

$$\ln p(\mathbf{w} | \mathbf{t}) = -\frac{\beta}{2} (\mathbf{t} - \Phi \mathbf{w})^T (\mathbf{t} - \Phi \mathbf{w}) - \frac{\alpha}{2} \mathbf{w}^T \mathbf{w} + \text{const}$$

# Bayesian Regression

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

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

- Maximising the posterior distribution with respect to  $\mathbf{w}$  corresponds to minimising the sum-of-squares error function with the addition of a quadratic regularisation term  $\lambda = \alpha/\beta$ .



Bayesian Regression

Sequential Update of the  
Posterior

Predictive Distribution

Proof of the Predictive  
Distribution

Predictive Distribution  
with Simplified Prior

Limitations of Linear  
Basis Function Models

# Sequential Update of the Posterior



- Example of a linear basis function model
- Single input  $x$ , single output  $t$
- Linear model  $y(x, \mathbf{w}) = w_0 + w_1x$ .
- Data creation

- ➊ Choose an  $x_n$  from the uniform distribution  $\mathcal{U}(x | -1, 1)$ .
- ➋ Calculate  $f(x_n, \mathbf{a}) = a_0 + a_1x_n$ , where  $a_0 = -0.3$ ,  $a_1 = 0.5$ .
- ➌ Add Gaussian noise with standard deviation  $\sigma = 0.2$ ,

$$t_n = \mathcal{N}(x_n | f(x_n, \mathbf{a}), 0.04)$$

- Set the precision of the uniform prior to  $\alpha = 2.0$ .

Bayesian Regression

Sequential Update of the  
Posterior

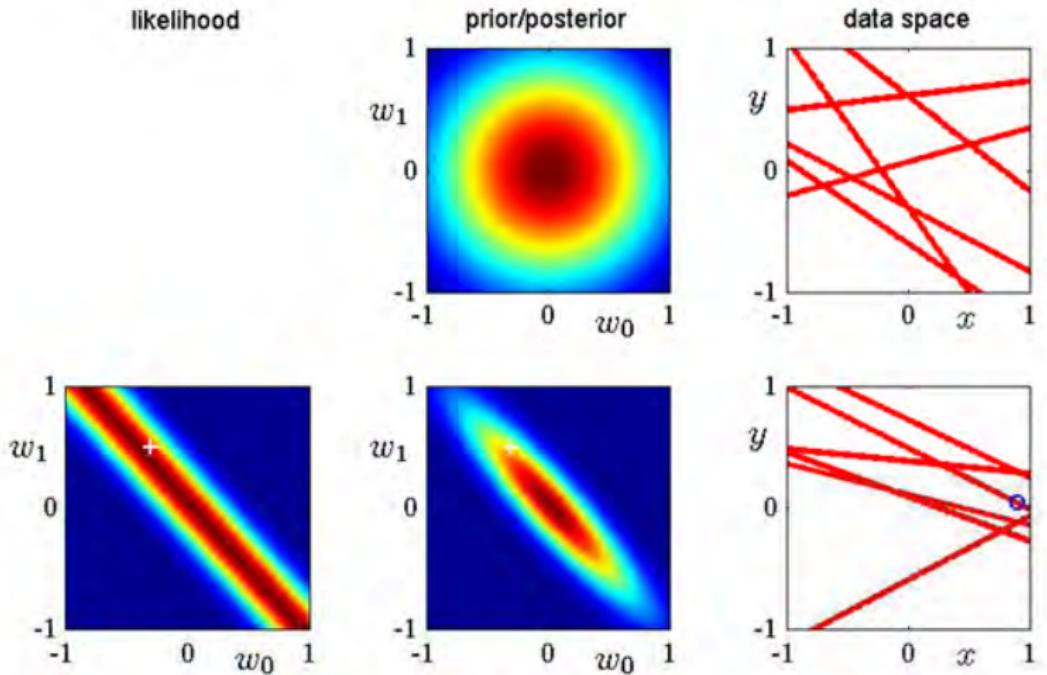
Predictive Distribution

Proof of the Predictive  
Distribution

Predictive Distribution  
with Simplified Prior

Limitations of Linear  
Basis Function Models

# Sequential Update of the Posterior





Bayesian Regression

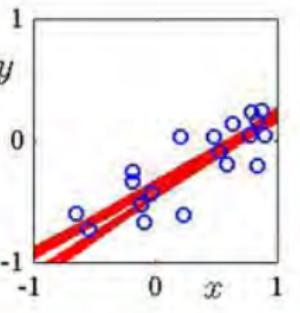
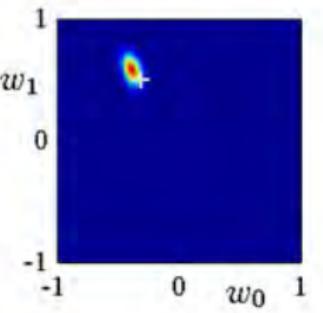
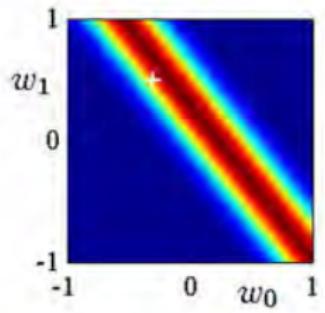
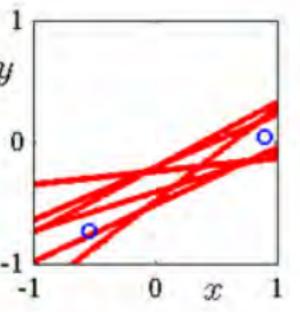
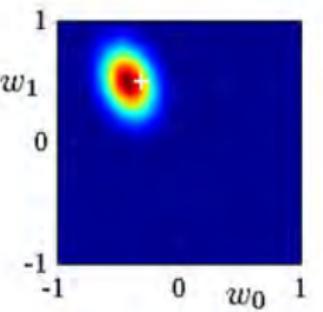
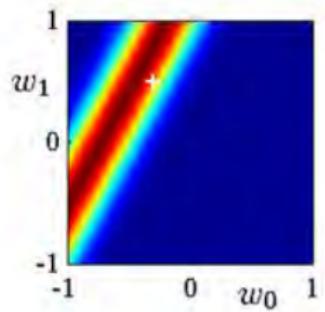
Sequential Update of the  
Posterior

Predictive Distribution

Proof of the Predictive  
Distribution

Predictive Distribution  
with Simplified Prior

Limitations of Linear  
Basis Function Models





Bayesian Regression

Sequential Update of the  
Posterior

Predictive Distribution

Proof of the Predictive  
Distribution

Predictive Distribution  
with Simplified Prior

Limitations of Linear  
Basis Function Models

- In the training phase, data  $\mathbf{x}$  and targets  $\mathbf{t}$  are provided
- In the test phase, a new data value  $x$  is given and the corresponding target value  $t$  is asked for
- Bayesian approach: Find the **probability** of the test target  $t$  given the test data  $x$ , the training data  $\mathbf{x}$  and the training targets  $\mathbf{t}$

$$p(t | x, \mathbf{x}, \mathbf{t})$$

- This is the **Predictive Distribution**.



Bayesian Regression

Sequential Update of the  
Posterior

Predictive Distribution

Proof of the Predictive  
DistributionPredictive Distribution  
with Simplified PriorLimitations of Linear  
Basis Function Models

# How to calculate the Predictive Distribution?

- Introduce the model parameter  $\mathbf{w}$  via the sum rule

$$\begin{aligned} p(t | x, \mathbf{x}, \mathbf{t}) &= \int p(t, \mathbf{w} | x, \mathbf{x}, \mathbf{t}) d\mathbf{w} \\ &= \int p(t | \mathbf{w}, x, \mathbf{x}, \mathbf{t}) p(\mathbf{w} | x, \mathbf{x}, \mathbf{t}) d\mathbf{w} \end{aligned}$$

- The test target  $t$  depends only on the test data  $x$  and the model parameter  $\mathbf{w}$ , but not on the training data and the training targets

$$p(t | \mathbf{w}, x, \mathbf{x}, \mathbf{t}) = p(t | \mathbf{w}, x)$$

- The model parameter  $\mathbf{w}$  are learned with the training data  $\mathbf{x}$  and the training targets  $\mathbf{t}$  only

$$p(\mathbf{w} | x, \mathbf{x}, \mathbf{t}) = p(\mathbf{w} | \mathbf{x}, \mathbf{t})$$

- Predictive Distribution

$$p(t | x, \mathbf{x}, \mathbf{t}) = \int p(t | \mathbf{w}, x) p(\mathbf{w} | \mathbf{x}, \mathbf{t}) d\mathbf{w}$$



Bayesian Regression

Sequential Update of the  
Posterior

Predictive Distribution

Proof of the Predictive  
DistributionPredictive Distribution  
with Simplified PriorLimitations of Linear  
Basis Function Models

# Proof of the Predictive Distribution

- How to prove the Predictive Distribution in the general form?

$$p(t | x, \mathbf{x}, \mathbf{t}) = \int p(t | \mathbf{w}, x, \mathbf{x}, \mathbf{t}) p(\mathbf{w} | x, \mathbf{x}, \mathbf{t}) d\mathbf{w}$$

- Convert each conditional probability on the right-hand-side into a joint probability.

$$\begin{aligned} & \int p(t | \mathbf{w}, x, \mathbf{x}, \mathbf{t}) p(\mathbf{w} | x, \mathbf{x}, \mathbf{t}) d\mathbf{w} \\ &= \int \frac{p(t, \mathbf{w}, x, \mathbf{x}, \mathbf{t})}{p(\mathbf{w}, x, \mathbf{x}, \mathbf{t})} \frac{p(\mathbf{w}, x, \mathbf{x}, \mathbf{t})}{p(x, \mathbf{x}, \mathbf{t})} d\mathbf{w} \\ &= \int \frac{p(t, \mathbf{w}, x, \mathbf{x}, \mathbf{t})}{p(x, \mathbf{x}, \mathbf{t})} d\mathbf{w} \\ &= \frac{p(t, x, \mathbf{x}, \mathbf{t})}{p(x, \mathbf{x}, \mathbf{t})} \\ &= p(t | x, \mathbf{x}, \mathbf{t}) \end{aligned}$$



Bayesian Regression

Sequential Update of the  
Posterior

Predictive Distribution

Proof of the Predictive  
DistributionPredictive Distribution  
with Simplified PriorLimitations of Linear  
Basis Function Models

# Predictive Distribution with Simplified Prior

- Find the predictive distribution

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

(remember : The conditioning on the input variables  $\mathbf{x}$  is often suppressed to simplify the notation.)

- Now we know

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

- and the posterior was

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

where

$$\mathbf{m}_N = \beta \mathbf{S}_N \boldsymbol{\Phi}^T \mathbf{t}$$

$$\mathbf{S}_N^{-1} = \alpha \mathbf{I} + \beta \boldsymbol{\Phi}^T \boldsymbol{\Phi}$$

# Predictive Distribution with Simplified Prior



- If we do the convolution of the two Gaussians, we get for the predictive distribution

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

where the variance  $\sigma_N^2(\mathbf{x})$  is given by

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

Bayesian Regression

Sequential Update of the  
Posterior

Predictive Distribution

Proof of the Predictive  
Distribution

Predictive Distribution  
with Simplified Prior

Limitations of Linear  
Basis Function Models



Bayesian Regression

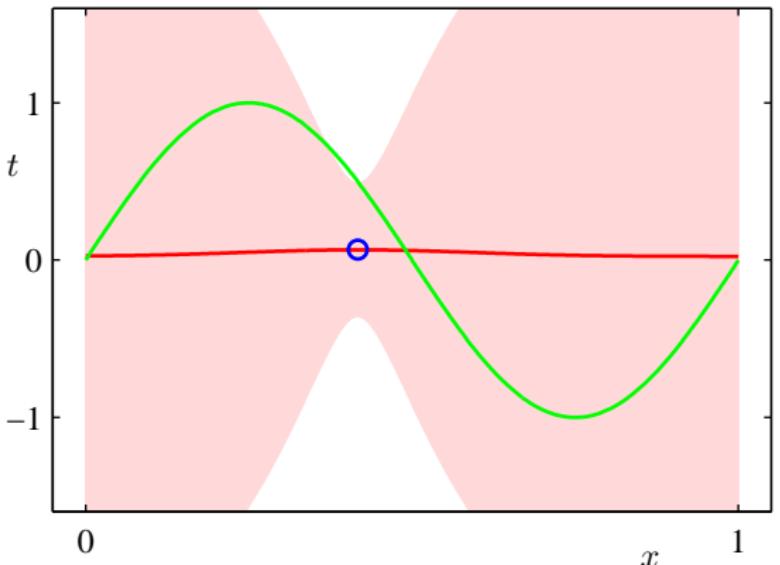
Sequential Update of the  
Posterior

Predictive Distribution

Proof of the Predictive  
DistributionPredictive Distribution  
with Simplified PriorLimitations of Linear  
Basis Function Models

# Predictive Distribution with Simplified Prior

Example with artificial sinusoidal data from  $\sin(2\pi x)$  (green) and added noise. Number of data points  $N = 1$ .



Mean of the predictive distribution (red) and regions of one standard deviation from mean (red shaded).



Bayesian Regression

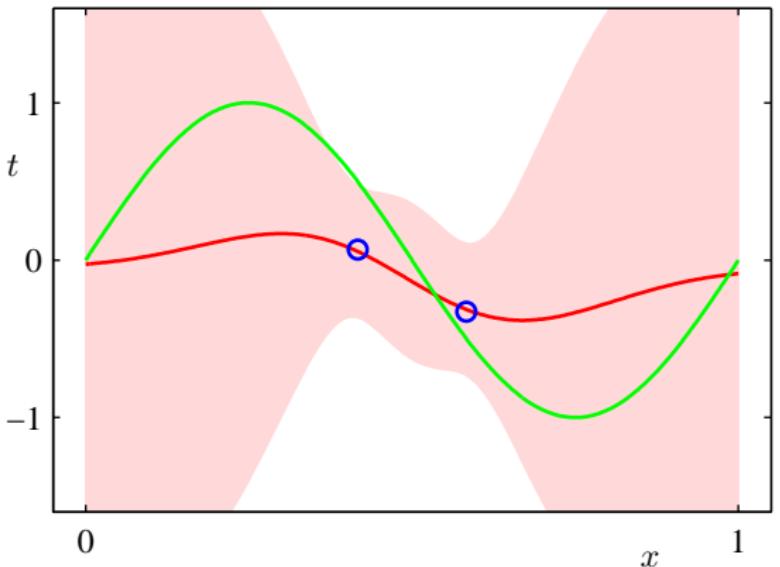
Sequential Update of the  
Posterior

Predictive Distribution

Proof of the Predictive  
DistributionPredictive Distribution  
with Simplified PriorLimitations of Linear  
Basis Function Models

# Predictive Distribution with Simplified Prior

Example with artificial sinusoidal data from  $\sin(2\pi x)$  (green) and added noise. Number of data points  $N = 2$ .



Mean of the predictive distribution (red) and regions of one standard deviation from mean (red shaded).



Bayesian Regression

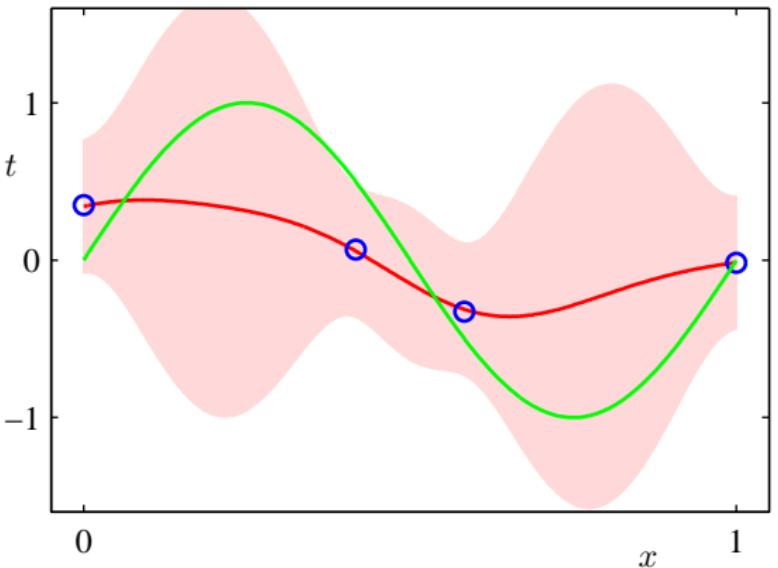
Sequential Update of the  
Posterior

Predictive Distribution

Proof of the Predictive  
DistributionPredictive Distribution  
with Simplified PriorLimitations of Linear  
Basis Function Models

# Predictive Distribution with Simplified Prior

Example with artificial sinusoidal data from  $\sin(2\pi x)$  (green) and added noise. Number of data points  $N = 4$ .



Mean of the predictive distribution (red) and regions of one standard deviation from mean (red shaded).



Bayesian Regression

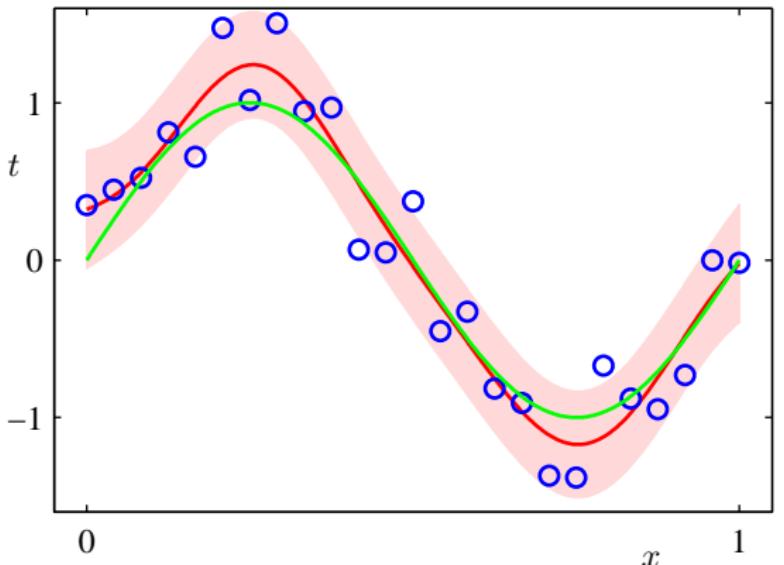
Sequential Update of the  
Posterior

Predictive Distribution

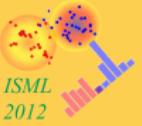
Proof of the Predictive  
DistributionPredictive Distribution  
with Simplified PriorLimitations of Linear  
Basis Function Models

# Predictive Distribution with Simplified Prior

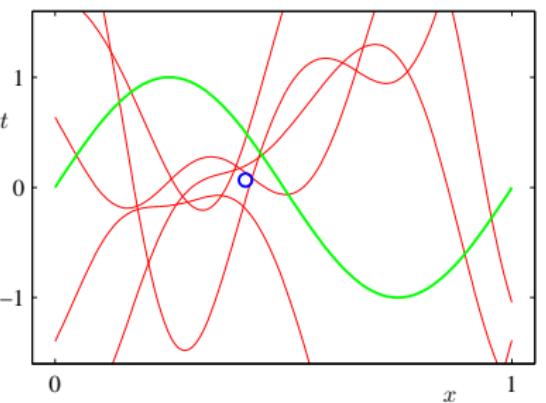
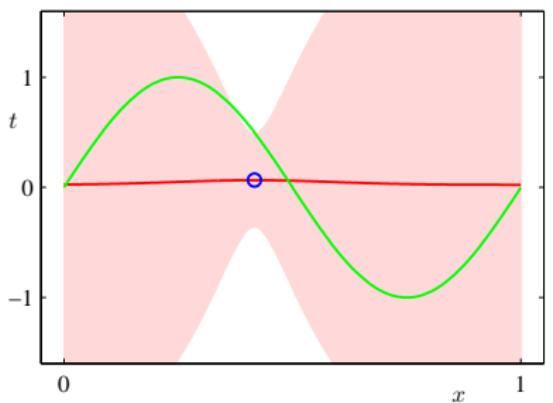
Example with artificial sinusoidal data from  $\sin(2\pi x)$  (green) and added noise. Number of data points  $N = 25$ .



Mean of the predictive distribution (red) and regions of one standard deviation from mean (red shaded).



Plots of the function  $y(x, \mathbf{w})$  using samples from the posterior distribution over  $\mathbf{w}$ . Number of data points  $N = 1$ .



Bayesian Regression

Sequential Update of the Posterior

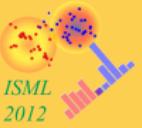
Predictive Distribution

Proof of the Predictive Distribution

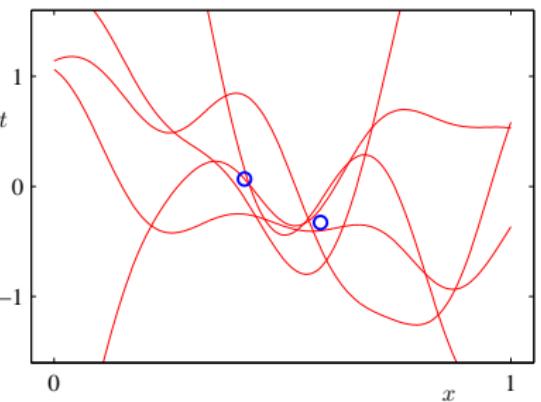
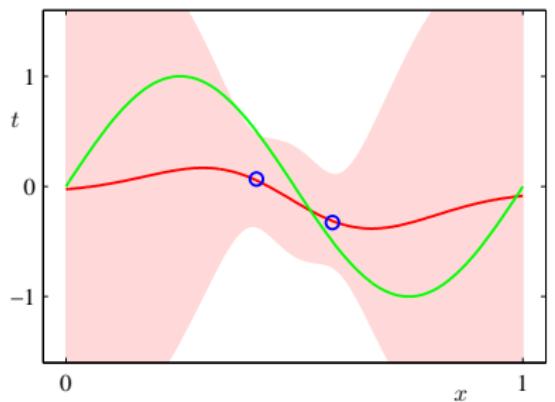
Predictive Distribution with Simplified Prior

Limitations of Linear Basis Function Models

# Predictive Distribution with Simplified Prior



Plots of the function  $y(x, \mathbf{w})$  using samples from the posterior distribution over  $\mathbf{w}$ . Number of data points  $N = 2$ .



Bayesian Regression

Sequential Update of the Posterior

Predictive Distribution

Proof of the Predictive Distribution

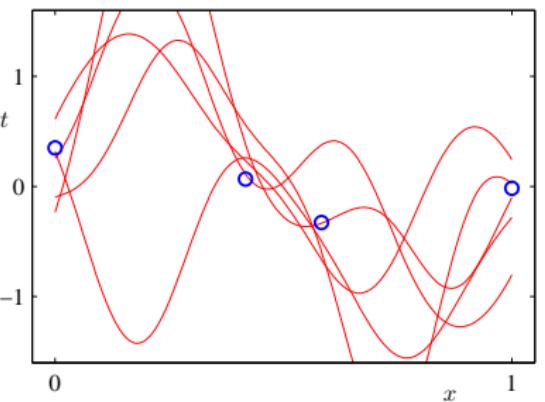
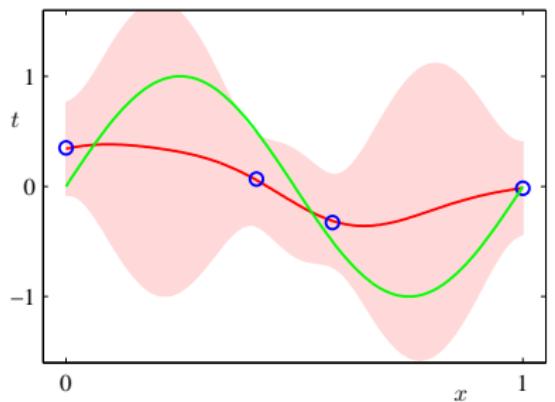
Predictive Distribution with Simplified Prior

Limitations of Linear Basis Function Models

# Predictive Distribution with Simplified Prior



Plots of the function  $y(x, \mathbf{w})$  using samples from the posterior distribution over  $\mathbf{w}$ . Number of data points  $N = 4$ .



Bayesian Regression

Sequential Update of the Posterior

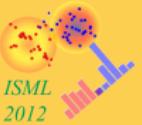
Predictive Distribution

Proof of the Predictive Distribution

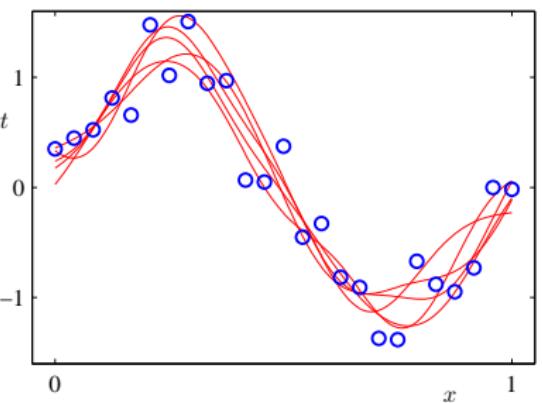
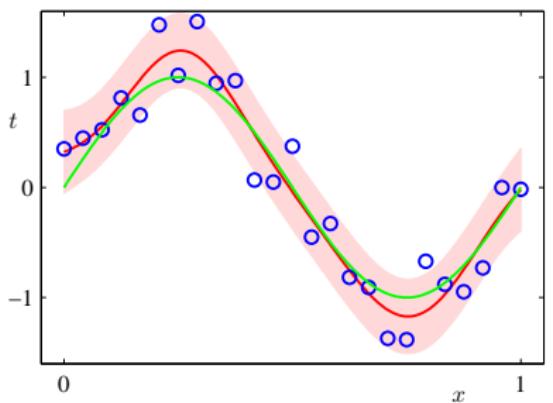
Predictive Distribution with Simplified Prior

Limitations of Linear Basis Function Models

# Predictive Distribution with Simplified Prior



Plots of the function  $y(x, \mathbf{w})$  using samples from the posterior distribution over  $\mathbf{w}$ . Number of data points  $N = 25$ .



Bayesian Regression

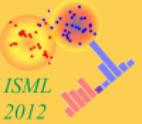
Sequential Update of the Posterior

Predictive Distribution

Proof of the Predictive Distribution

Predictive Distribution with Simplified Prior

Limitations of Linear Basis Function Models



Bayesian Regression

Sequential Update of the  
Posterior

Predictive Distribution

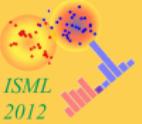
Proof of the Predictive  
Distribution

Predictive Distribution  
with Simplified Prior

Limitations of Linear  
Basis Function Models

# Limitations of Linear Basis Function Models

- Basis function  $\phi_j(\mathbf{x})$  are fixed before the training data set is observed.
- Curse of dimensionality : Number of basis function grows rapidly, often exponentially, with the dimensionality  $D$ .
- But typical data sets have two nice properties which can be exploited if the basis functions are not fixed :
  - Data lie close to a nonlinear manifold with intrinsic dimension much smaller than  $D$ . Need algorithms which place basis functions only where data are (e.g. radial basis function networks, support vector machines, relevance vector machines, neural networks).
  - Target variables may only depend on a few significant directions within the data manifold. Need algorithms which can exploit this property (Neural networks).



Bayesian Regression

Sequential Update of the  
Posterior

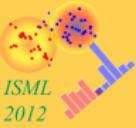
Predictive Distribution

Proof of the Predictive  
DistributionPredictive Distribution  
with Simplified PriorLimitations of Linear  
Basis Function Models

# 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  $V_D(r) = K_D r^D$ .

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



Bayesian Regression

Sequential Update of the  
Posterior

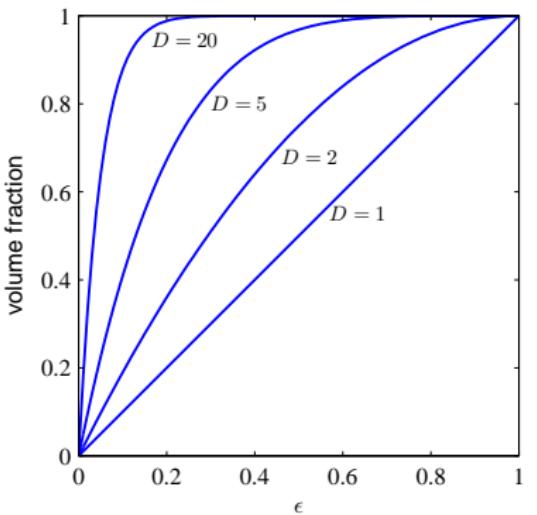
Predictive Distribution

Proof of the Predictive  
DistributionPredictive Distribution  
with Simplified PriorLimitations of Linear  
Basis Function Models

# Curse of Dimensionality

- Fraction of the volume of the sphere in a  $D$ -dimensional space which lies between radius  $r = 1$  and  $r = 1 - \epsilon$

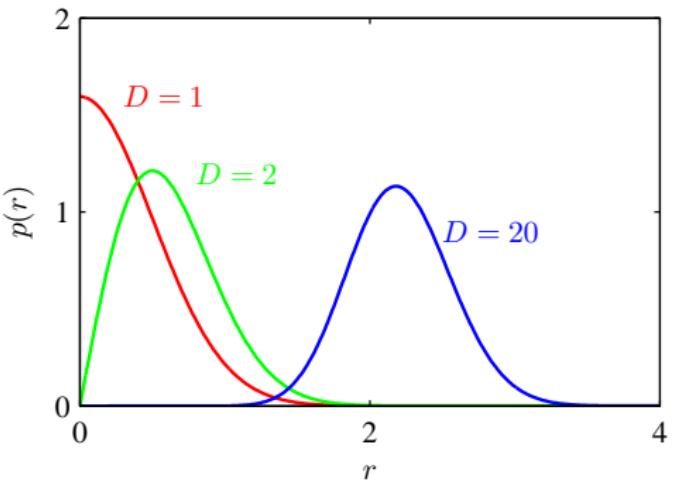
$$\frac{V_D(1) - V_D(1 - \epsilon)}{V_D(1)} = 1 - (1 - \epsilon)^D$$





# Curse of Dimensionality

- Probability density with respect to radius  $r$  of a Gaussian distribution for various values of the dimensionality  $D$ .



Bayesian Regression

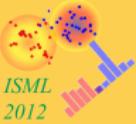
Sequential Update of the  
Posterior

Predictive Distribution

Proof of the Predictive  
Distribution

Predictive Distribution  
with Simplified Prior

Limitations of Linear  
Basis Function Models



Bayesian Regression

Sequential Update of the  
Posterior

Predictive Distribution

Proof of the Predictive  
DistributionPredictive Distribution  
with Simplified PriorLimitations of Linear  
Basis Function Models

# Curse of Dimensionality

- Probability density with respect to radius  $r$  of a Gaussian distribution for various values of the dimensionality  $D$ .
- Example:  $D = 2$ ; assume  $\mu = 0, \Sigma = I$

$$\mathcal{N}(x | 0, I) = \frac{1}{2\pi} \exp \left\{ -\frac{1}{2} x^T x \right\} = \frac{1}{2\pi} \exp \left\{ -\frac{1}{2} (x_1^2 + x_2^2) \right\}$$

- Coordinate transformation

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

- Probability in the new coordinates

$$p(r, \phi | 0, I) = \mathcal{N}(r(x), \phi(x) | 0, I) | J |$$

where  $|J| = r$  is the determinant of the Jacobian for the given coordinate transformation.

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



Bayesian Regression

Sequential Update of the  
Posterior

Predictive Distribution

Proof of the Predictive  
DistributionPredictive Distribution  
with Simplified PriorLimitations of Linear  
Basis Function Models

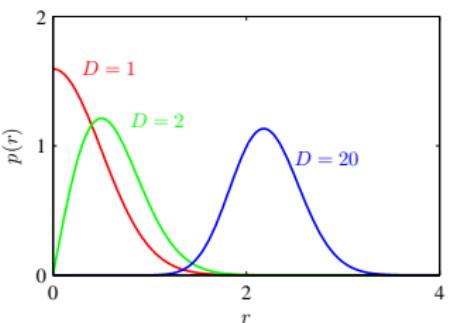
# Curse of Dimensionality

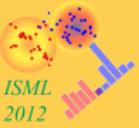
- Probability density with respect to radius  $r$  of a Gaussian distribution for  $D = 2$  (and  $\mu = 0, \Sigma = I$ )

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

- Integrate over all angles  $\phi$

$$p(r | 0, I) = \int_0^{2\pi} \frac{1}{2\pi} r \exp \left\{ -\frac{1}{2} r^2 \right\} d\phi = r \exp \left\{ -\frac{1}{2} r^2 \right\}$$





# Part VII

## *Linear Classification 1*

*Classification*

*Generalised Linear  
Model*

*Inference and Decision*

*Discriminant Functions*

*Fisher's Linear  
Discriminant*

*The Perceptron  
Algorithm*



# 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.

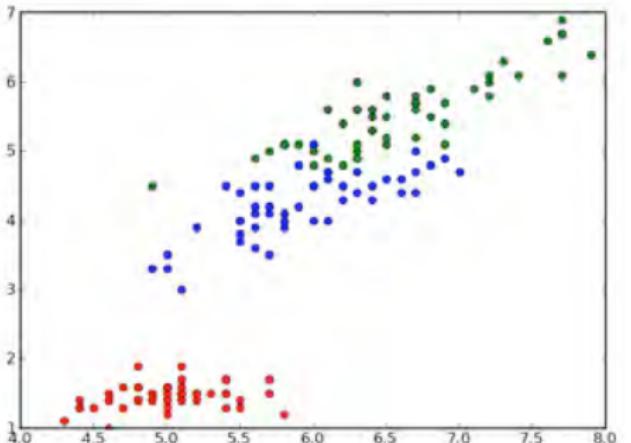


Figure : Length of the petal [in cm] for a given sepal [cm] for iris flowers (Iris Setosa, Iris Versicolor, Iris Virginica).

Classification

Generalised Linear  
Model

Inference and Decision

Discriminant Functions

Fisher's Linear  
Discriminant

The Perceptron  
Algorithm



*Classification*

*Generalised Linear  
Model*

*Inference and Decision*

*Discriminant Functions*

*Fisher's Linear  
Discriminant*

*The Perceptron  
Algorithm*

# How to represent binary class labels?

- Class labels are no longer real values as in regression, but a discrete set.
- Two classes :  $t \in \{0, 1\}$   
(  $t = 1$  represents class  $\mathcal{C}_1$  and  $t = 0$  represents class  $\mathcal{C}_2$  )
- Can interpret the value of  $t$  as the probability of class  $\mathcal{C}_1$ , with only two values possible for the probability, 0 or 1.
- Note: Other conventions to map classes into integers possible, check the setup.

*Classification**Generalised Linear  
Model**Inference and Decision**Discriminant Functions**Fisher's Linear  
Discriminant**The Perceptron  
Algorithm*

# How to represent multi-class labels?

- If there are more than two classes ( $K > 2$ ), we call it a multi-class setup.
- Often used: 1-of- $K$  coding scheme in which  $\mathbf{t}$  is a vector of length  $K$  which has all values 0 except for  $t_j = 1$ , where  $j$  comes from the membership in class  $C_j$  to encode.
- Example: Given 5 classes,  $\{C_1, \dots, C_5\}$ . Membership in class  $C_2$  will be encoded as the target vector

$$\mathbf{t} = (0, 1, 0, 0, 0)^T$$

- Note: Other conventions to map multi-classes into integers possible, check the setup.



Classification

Generalised Linear  
Model

Inference and Decision

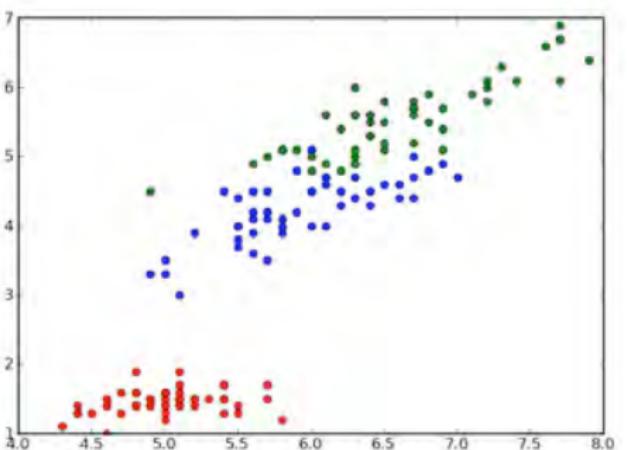
Discriminant Functions

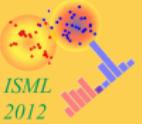
Fisher's Linear  
DiscriminantThe Perceptron  
Algorithm

- Idea: Use again a **Linear Model** as in regression:  $y(\mathbf{x}, \mathbf{w})$  is a linear function of the parameters  $\mathbf{w}$

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

- But generally  $y(\mathbf{x}_n, \mathbf{w}) \in \mathbb{R}$ .  
Example: Which class is  $y(\mathbf{x}, \mathbf{w}) = 0.71623$  ?





Classification

Generalised Linear  
Model

Inference and Decision

Discriminant Functions

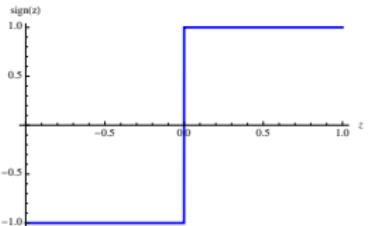
Fisher's Linear  
DiscriminantThe Perceptron  
Algorithm

Figure : Example of an activation function  $f(z) = \text{sign}(z)$  .



Classification

Generalised Linear  
Model

Inference and Decision

Discriminant Functions

Fisher's Linear  
DiscriminantThe Perceptron  
Algorithm

# Three Models for Decision Problems

In increasing order of complexity

- Find a **discriminant function**  $f(\mathbf{x})$  which maps each input directly onto a class label.
- Discriminative Models
  - ➊ Solve the inference problem of determining the posterior class probabilities  $p(\mathcal{C}_k | \mathbf{x})$ .
  - ➋ Use decision theory to assign each new  $\mathbf{x}$  to one of the classes.
- Generative Models
  - ➌ Solve the inference problem of determining the class-conditional probabilities  $p(\mathbf{x} | \mathcal{C}_k)$ .
  - ➍ Also, infer the prior class probabilities  $p(\mathcal{C}_k)$ .
  - ➎ Use Bayes' theorem to find the posterior  $p(\mathcal{C}_k | \mathbf{x})$ .
  - ➏ Alternatively, model the joint distribution  $p(\mathbf{x}, \mathcal{C}_k)$  directly.
  - ➐ Use decision theory to assign each new  $\mathbf{x}$  to one of the classes.



Classification

Generalised Linear  
Model

Inference and Decision

Discriminant Functions

Fisher's Linear  
DiscriminantThe Perceptron  
Algorithm

# Two Classes

## Definition

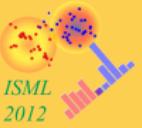
A **discriminant** is a function that maps from an input vector  $\mathbf{x}$  to one of  $K$  classes, denoted by  $\mathcal{C}_k$ .

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

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

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** )



Classification

Generalised Linear  
Model

Inference and Decision

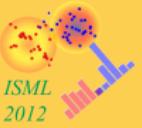
Discriminant Functions

Fisher's Linear  
Discriminant

The Perceptron  
Algorithm

- Decision boundary  $y(\mathbf{x}) = 0$  is a  $(D - 1)$ -dimensional hyperplane in a  $D$ -dimensional input space (**decision surface**).
- $\mathbf{w}$  is orthogonal to any vector lying in the decision surface.
- Proof: Assume  $\mathbf{x}_A$  and  $\mathbf{x}_B$  are two points lying in the decision surface. Then,

$$0 = y(\mathbf{x}_A) - y(\mathbf{x}_B) = \mathbf{w}^T (\mathbf{x}_A - \mathbf{x}_B)$$



Classification

Generalised Linear  
Model

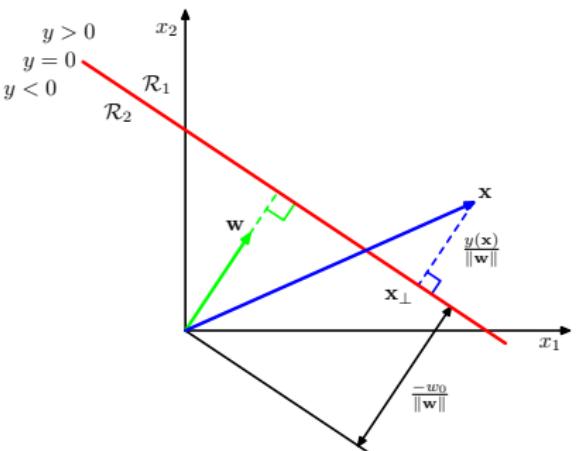
Inference and Decision

Discriminant Functions

Fisher's Linear  
DiscriminantThe Perceptron  
Algorithm

- The normal distance from the origin to the decision surface is

$$\frac{\mathbf{w}^T \mathbf{x}}{\|\mathbf{w}\|} = -\frac{w_0}{\|\mathbf{w}\|}$$





Classification

Generalised Linear  
Model

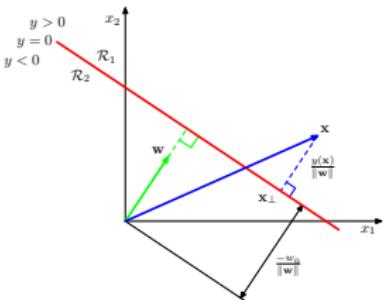
Inference and Decision

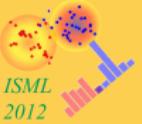
Discriminant Functions

Fisher's Linear  
DiscriminantThe Perceptron  
Algorithm

- The value of  $y(\mathbf{x})$  gives a signed measure of the perpendicular distance  $r$  of the point  $\mathbf{x}$  from the decision surface,  $r = y(\mathbf{x})/\|\mathbf{w}\|$ .

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





Classification

Generalised Linear  
Model

Inference and Decision

Discriminant Functions

Fisher's Linear  
Discriminant

The Perceptron  
Algorithm

# Two Classes

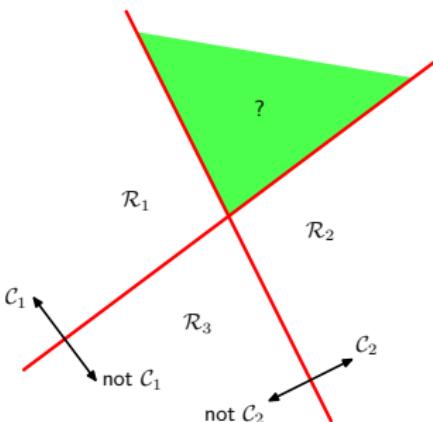
- More compact notation : Add an extra dimension to the input space and set the value to  $x_0 = 1$ .
- Also define  $\tilde{\mathbf{w}} = (w_0, \mathbf{w})$  and  $\tilde{\mathbf{x}} = (1, \mathbf{x})$

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

- Decision surface is now a  $D$ -dimensional hyperplane in a  $D + 1$ -dimensional expanded input space.



- Number of classes  $K > 2$
- Can we combine a number of two-class discriminant functions using  $K - 1$  one-versus-the-rest classifiers?



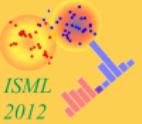
Classification

Generalised Linear  
Model

Inference and Decision

Discriminant Functions

Fisher's Linear  
DiscriminantThe Perceptron  
Algorithm



Classification

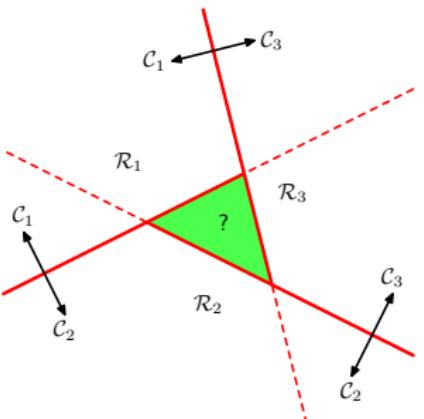
Generalised Linear  
Model

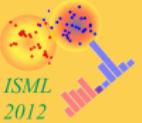
Inference and Decision

Discriminant Functions

Fisher's Linear  
DiscriminantThe Perceptron  
Algorithm

- Number of classes  $K > 2$
- Can we combine a number of two-class discriminant functions using  $K(K - 1)/2$  one-versus-one classifiers?





Classification

Generalised Linear  
Model

Inference and Decision

Discriminant Functions

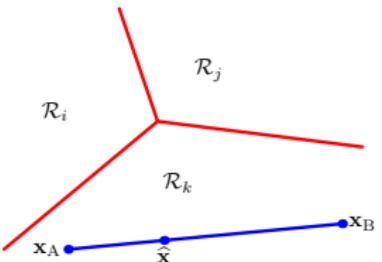
Fisher's Linear  
DiscriminantThe Perceptron  
Algorithm

# Multi-Class

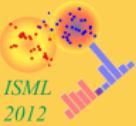
- Number of classes  $K > 2$
- Solution: Use  $K$  linear functions

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

- Assign input  $\mathbf{x}$  to class  $\mathcal{C}_k$  if  $y_k(\mathbf{x}) > y_j(\mathbf{x})$  for all  $j \neq k$ .
- Decision boundary between class  $\mathcal{C}_k$  and  $\mathcal{C}_j$  given by  
 $y_k(\mathbf{x}) = y_j(\mathbf{x})$



# Least Squares for Classification



- Regression with a linear function of the model parameters and minimisation of sum-of-squares error function resulted in a closed-form solution for the parameter values.
- Is this also possible for classification?
- Given input data  $\mathbf{x}$  belonging to one of  $K$  classes  $\mathcal{C}_k$ .
- Use 1-of- $K$  binary coding scheme.
- Each class is described by its own linear model

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0} \quad k = 1, \dots, K$$

Classification

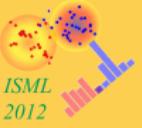
Generalised Linear  
Model

Inference and Decision

Discriminant Functions

Fisher's Linear  
Discriminant

The Perceptron  
Algorithm



Classification

Generalised Linear  
Model

Inference and Decision

Discriminant Functions

Fisher's Linear  
DiscriminantThe Perceptron  
Algorithm

# Least Squares for Classification

- With the conventions

$$\tilde{\mathbf{w}}_k = \begin{bmatrix} w_{k0} \\ \mathbf{w}_k \end{bmatrix} \in \mathbb{R}^{D+1}$$

$$\tilde{\mathbf{x}} = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix} \in \mathbb{R}^{D+1}$$

$$\tilde{\mathbf{W}} = [\tilde{\mathbf{w}}_1 \quad \dots \quad \tilde{\mathbf{w}}_K] \in \mathbb{R}^{(D+1) \times K}$$

- we get for the discriminant function (vector valued)

$$\mathbf{y}(\mathbf{x}) = \tilde{\mathbf{W}}^T \tilde{\mathbf{x}} \in \mathbb{R}^K.$$

- For a new input  $\mathbf{x}$ , the class is then defined by the index of the largest value in the row vector  $\mathbf{y}(\mathbf{x})$



Classification

Generalised Linear  
Model

Inference and Decision

Discriminant Functions

Fisher's Linear  
DiscriminantThe Perceptron  
Algorithm

# Determine $\tilde{\mathbf{W}}$

- Given a training set  $\{\mathbf{x}_n, \mathbf{t}\}$  where  $n = 1, \dots, N$ , and  $\mathbf{t}$  is the class in the 1-of- $K$  coding scheme.
- Define a matrix  $\mathbf{T}$  where row  $n$  corresponds to  $\mathbf{t}_n^T$ .
- The sum-of-squares error can now be written as

$$E_D(\tilde{\mathbf{W}}) = \frac{1}{2} \text{tr} \left\{ (\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T})^T (\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}) \right\}$$

- The minimum of  $E_D(\tilde{\mathbf{W}})$  will be reached for

$$\tilde{\mathbf{W}} = (\tilde{\mathbf{X}}^T \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^T \mathbf{T} = \tilde{\mathbf{X}}^\dagger \mathbf{T}$$

where  $\tilde{\mathbf{X}}^\dagger$  is the pseudo-inverse of  $\tilde{\mathbf{X}}$ .



# Discriminant Function for Multi-Class

- The discriminant function  $\mathbf{y}(\mathbf{x})$  is therefore

$$\mathbf{y}(\mathbf{x}) = \tilde{\mathbf{W}}^T \tilde{\mathbf{x}} = \mathbf{T}^T (\tilde{\mathbf{X}}^\dagger)^T \tilde{\mathbf{x}},$$

where  $\tilde{\mathbf{X}}$  is given by the training data, and  $\tilde{\mathbf{x}}$  is the new input.

- Interesting property: If for every  $\mathbf{t}_n$  the same linear constraint  $\mathbf{a}^T \mathbf{t}_n + b = 0$  holds, then the prediction  $\mathbf{y}(\mathbf{x})$  will also obey the same constraint

$$\mathbf{a}^T \mathbf{y}(\mathbf{x}) + b = 0.$$

- For the 1-of- $K$  coding scheme, the sum of all components in  $\mathbf{t}_n$  is one, and therefore all components of  $\mathbf{y}(\mathbf{x})$  will sum to one. BUT: the components are not probabilities, as they are not constraint to the interval  $(0, 1)$ .

Classification

Generalised Linear  
Model

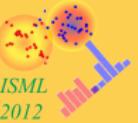
Inference and Decision

Discriminant Functions

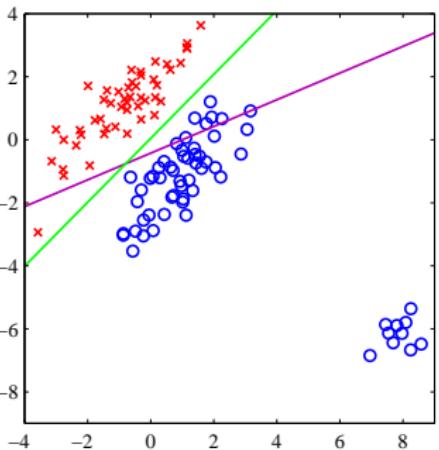
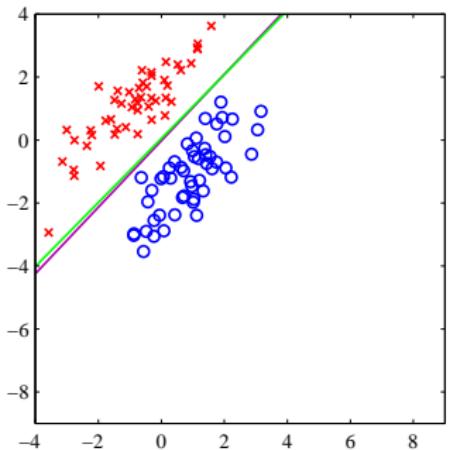
Fisher's Linear  
Discriminant

The Perceptron  
Algorithm

# Deficiencies of the Least Squares Approach



Magenta curve : Decision Boundary for the least squares approach ( Green curve : Decision boundary for the logistic regression model described later)



Classification

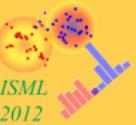
Generalised Linear  
Model

Inference and Decision

Discriminant Functions

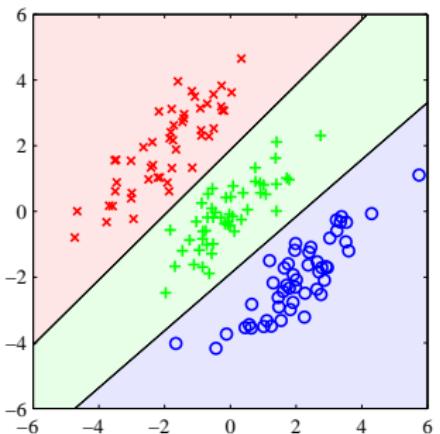
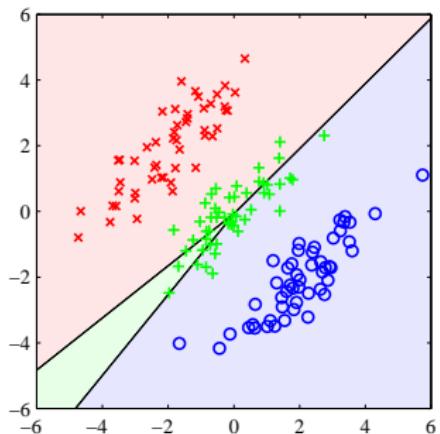
Fisher's Linear  
Discriminant

The Perceptron  
Algorithm



# Deficiencies of the Least Squares Approach

Magenta curve : Decision Boundary for the least squares approach ( Green curve : Decision boundary for the logistic regression model described later)



Classification

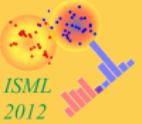
Generalised Linear  
Model

Inference and Decision

Discriminant Functions

Fisher's Linear  
Discriminant

The Perceptron  
Algorithm



Classification

Generalised Linear  
Model

Inference and Decision

Discriminant Functions

Fisher's Linear  
DiscriminantThe Perceptron  
Algorithm

# Fisher's Linear Discriminant

- View linear classification as dimensionality reduction.

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

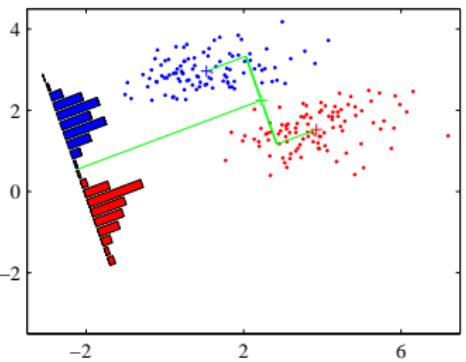
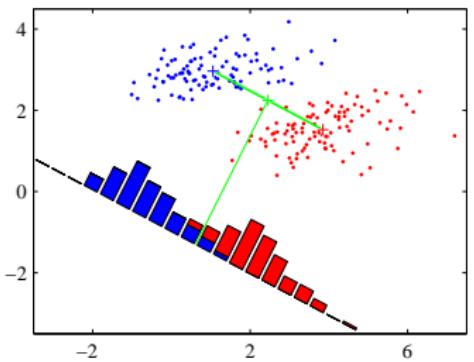
If  $y \geq -w_0$  then class  $\mathcal{C}_1$ , otherwise  $\mathcal{C}_2$ .

- But there are many projections from a  $D$ -dimensional input space onto one dimension.
- Projection always means loss of information.
- For classification we want to preserve the class separation in one dimension.
- Can we find a projection which maximally preserves the class separation ?



# Fisher's Linear Discriminant

Samples from two classes in a two-dimensional input space and their histogram when projected to two different one-dimensional spaces.



Classification

Generalised Linear  
Model

Inference and Decision

Discriminant Functions

Fisher's Linear  
Discriminant

The Perceptron  
Algorithm



Classification

Generalised Linear  
Model

Inference and Decision

Discriminant Functions

Fisher's Linear  
DiscriminantThe Perceptron  
Algorithm

# Fisher's Linear Discriminant - First Try

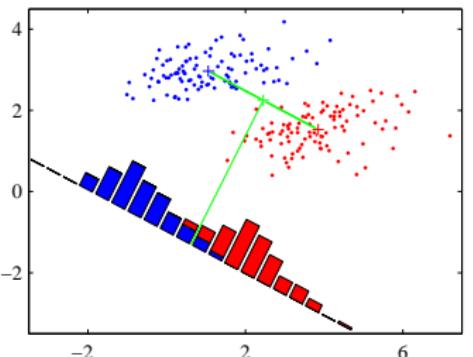
- Given  $N_1$  input data of class  $\mathcal{C}_1$ , and  $N_2$  input data of class  $\mathcal{C}_2$ , calculate the centres of the two classes

$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n, \quad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n$$

- Choose  $\mathbf{w}$  so as to maximise the projection of the class means onto  $\mathbf{w}$

$$m_1 - m_2 = \mathbf{w}^T (\mathbf{m}_1 - \mathbf{m}_2)$$

- Problem with non-uniform covariance



# Fisher's Linear Discriminant

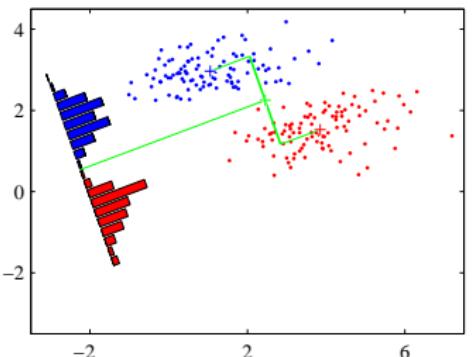
- Measure also the within-class variance for each class

$$s_k^2 = \sum_{n \in C_k} (y_n - m_k)^2$$

where  $y_n = \mathbf{w}^T \mathbf{x}_n$ .

- Maximise the Fisher criterion

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$



Classification

Generalised Linear  
Model

Inference and Decision

Discriminant Functions

Fisher's Linear  
DiscriminantThe Perceptron  
Algorithm



Classification

Generalised Linear  
Model

Inference and Decision

Discriminant Functions

Fisher's Linear  
DiscriminantThe Perceptron  
Algorithm

# Fisher's Linear Discriminant

- The Fisher criterion can be rewritten as

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

- $\mathbf{S}_B$  is the **between-class** covariance

$$\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$$

- $\mathbf{S}_W$  is the **within-class** covariance

$$\mathbf{S}_W = \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^T$$



Classification

Generalised Linear  
Model

Inference and Decision

Discriminant Functions

Fisher's Linear  
DiscriminantThe Perceptron  
Algorithm

# Fisher's Linear Discriminant

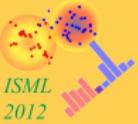
- The Fisher criterion

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

has a maximum for Fisher's linear discriminant

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

- Fisher's linear discriminant is NOT a discriminant, but can be used to construct one by choosing a threshold  $y_0$  in the projection space.

*Classification**Generalised Linear  
Model**Inference and Decision**Discriminant Functions**Fisher's Linear  
Discriminant**The Perceptron  
Algorithm*

# Fisher's Discriminant For Multi-Class

- Assume that the dimensionality of the input space  $D$  is greater than the number of classes  $K$ .
- Use  $D' > 1$  linear 'features'  $y_k = \mathbf{w}^T \mathbf{x}$  and write everything in vector form (no bias involved!)

$$\mathbf{y} = \mathbf{W}^T \mathbf{x}.$$

- The within-class covariance is then the sum of the covariances for all  $K$  classes

$$\mathbf{S}_W = \sum_{k=1}^K \mathbf{S}_k$$

where

$$\mathbf{S}_k = \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^T$$

$$\mathbf{m}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n$$



Classification

Generalised Linear  
Model

Inference and Decision

Discriminant Functions

Fisher's Linear  
DiscriminantThe Perceptron  
Algorithm

# Fisher's Discriminant For Multi-Class

- Between-class covariance

$$\mathbf{S}_B = \sum_{k=1}^K N_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T.$$

where  $\mathbf{m}$  is the total mean of the input data

$$\mathbf{m} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n.$$

- One possible way to define a function of  $\mathbf{W}$  which is large when the between-class covariance is large and the within-class covariance is small is given by

$$J(\mathbf{W}) = \text{tr} \left\{ (\mathbf{W}^T \mathbf{S}_W \mathbf{W})^{-1} (\mathbf{W}^T \mathbf{S}_B \mathbf{W}) \right\}$$

- The maximum of  $J(\mathbf{W})$  is determined by the  $D'$  eigenvectors of  $\mathbf{S}_W^{-1} \mathbf{S}_B$  with the largest eigenvalues.



Classification

Generalised Linear  
Model

Inference and Decision

Discriminant Functions

Fisher's Linear  
Discriminant

The Perceptron  
Algorithm

# Fisher's Discriminant For Multi-Class

- How many linear 'features' can one find with this method?
- $\mathbf{S}_B$  is of rank at most  $K - 1$  because of the sum of  $K$  rank one matrices and the global constraint via  $\mathbf{m}$ .
- Projection onto the subspace spanned by  $\mathbf{S}_B$  can not have more than  $K - 1$  linear features.

# The Perceptron Algorithm

- Frank Rosenblatt (1928 - 1969)
- "Principles of neurodynamics: Perceptrons and the theory of brain mechanisms" (Spartan Books, 1962)



Classification

Generalised Linear  
Model

Inference and Decision

Discriminant Functions

Fisher's Linear  
Discriminant

The Perceptron  
Algorithm

# The Perceptron Algorithm

- Perceptron ("MARK 1") was the first computer which could learn new skills by trial and error



Classification

Generalised Linear  
Model

Inference and Decision

Discriminant Functions

Fisher's Linear  
Discriminant

The Perceptron  
Algorithm



# The Perceptron Algorithm

- Two class model
- Create feature vector  $\phi(\mathbf{x})$  by a fixed nonlinear transformation of the input  $\mathbf{x}$ .
- Generalised linear model

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

with  $\phi(\mathbf{x})$  containing some bias element  $\phi_0(\mathbf{x}) = 1$ .

- nonlinear **activation** function

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

- Target coding for perceptron

$$t = \begin{cases} +1, & \text{if } \mathcal{C}_1 \\ -1, & \text{if } \mathcal{C}_2 \end{cases}$$

Classification

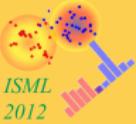
Generalised Linear  
Model

Inference and Decision

Discriminant Functions

Fisher's Linear  
DiscriminantThe Perceptron  
Algorithm

# The Perceptron Algorithm - Error Function



Classification

Generalised Linear  
Model

Inference and Decision

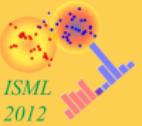
Discriminant Functions

Fisher's Linear  
Discriminant

The Perceptron  
Algorithm

- Idea : Minimise total number of misclassified patterns.
- Problem : As a function of  $\mathbf{w}$ , this is piecewise constant and therefore the gradient is zero almost everywhere.
- Better idea: Using the  $(-1, +1)$  target coding scheme, we want all patterns to satisfy  $\mathbf{w}^T \phi(\mathbf{x}_n) t_n > 0$ .
- **Perceptron Criterion** : Add the errors for all patterns belonging to the set of misclassified patterns  $\mathcal{M}$

$$E_P(\mathbf{w}) = - \sum_{n \in \mathcal{M}} \mathbf{w}^T \phi(\mathbf{x}_n) t_n$$



Classification

Generalised Linear  
Model

Inference and Decision

Discriminant Functions

Fisher's Linear  
DiscriminantThe Perceptron  
Algorithm

# Perceptron - Stochastic Gradient Descent

- Perceptron Criterion (with notation  $\phi_n = \phi(\mathbf{x}_n)$ )

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

- One iteration at step  $\tau$

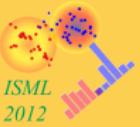
- ➊ Choose a training pair  $(\mathbf{x}_n, t_n)$
- ➋ Update the weight vector  $\mathbf{w}$  by

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

- As  $y(\mathbf{x}, \mathbf{w})$  does not depend on the norm of  $\mathbf{w}$ , one can set  $\eta = 1$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \phi_n t_n$$

# The Perceptron Algorithm - Update 1



Classification

Generalised Linear  
Model

Inference and Decision

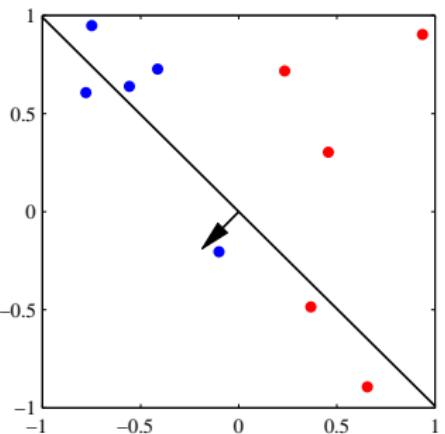
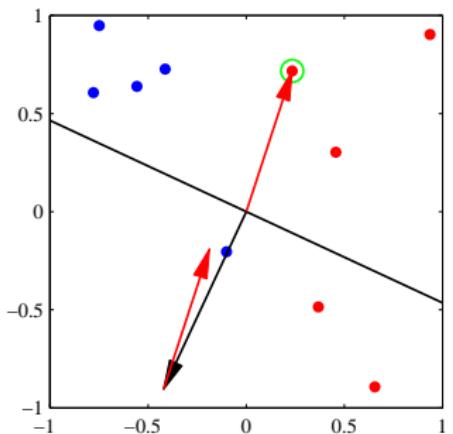
Discriminant Functions

Fisher's Linear  
Discriminant

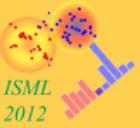
The Perceptron  
Algorithm

Update of the perceptron weights from a misclassified pattern  
(green)

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \phi_n t_n$$



# The Perceptron Algorithm - Update 2



Classification

Generalised Linear  
Model

Inference and Decision

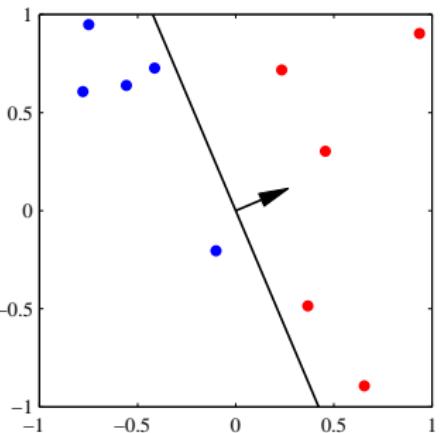
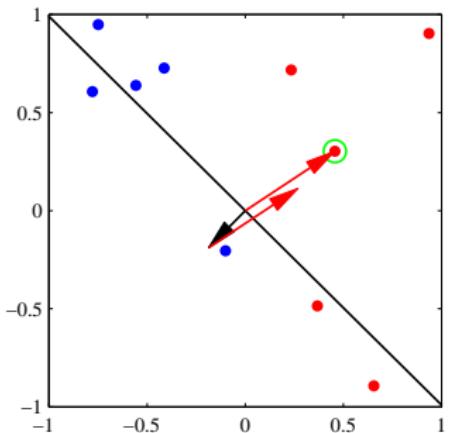
Discriminant Functions

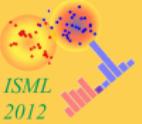
Fisher's Linear  
Discriminant

The Perceptron  
Algorithm

Update of the perceptron weights from a misclassified pattern  
(green)

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \phi_n t_n$$





Classification

Generalised Linear  
Model

Inference and Decision

Discriminant Functions

Fisher's Linear  
DiscriminantThe Perceptron  
Algorithm

# The Perceptron Algorithm - Convergence

- Does the algorithm converge ?
- For a single update step

$$-\mathbf{w}^{(\tau+1)T} \phi_n t_n = -\mathbf{w}^{(\tau)T} \phi_n t_n - (\phi_n t_n)^T \phi_n t_n < -\mathbf{w}^{(\tau)T} \phi_n t_n$$

because  $(\phi_n t_n)^T \phi_n t_n = \|\phi_n t_n\| > 0$ .

- BUT: contributions to the error from the other misclassified patterns might have increased.
- AND: some correctly classified patterns might now be misclassified.
- **Perceptron Convergence Theorem** : If the training set is linearly separable, the perceptron algorithm is guaranteed to find a solution in a finite number of steps.



# Part VIII

## *Linear Classification 2*

*Probabilistic Generative  
Models*

*Continuous Input*

*Discrete Features*

*Probabilistic  
Discriminative Models*

*Logistic Regression*

*Iterative Reweighted  
Least Squares*

*Laplace Approximation*

*Bayesian Logistic  
Regression*



# Three Models for Decision Problems

In increasing order of complexity

- Find a discriminant function  $f(\mathbf{x})$  which maps each input directly onto a class label.

- **Discriminative Models**

- ➊ Solve the inference problem of determining the posterior class probabilities  $p(\mathcal{C}_k | \mathbf{x})$ .
- ➋ Use decision theory to assign each new  $\mathbf{x}$  to one of the classes.

- **Generative Models**

- ➊ Solve the inference problem of determining the class-conditional probabilities  $p(\mathbf{x} | \mathcal{C}_k)$ .
- ➋ Also, infer the prior class probabilities  $p(\mathcal{C}_k)$ .
- ➌ Use Bayes' theorem to find the posterior  $p(\mathcal{C}_k | \mathbf{x})$ .
- ➍ Alternatively, model the joint distribution  $p(\mathbf{x}, \mathcal{C}_k)$  directly.
- ➎ Use decision theory to assign each new  $\mathbf{x}$  to one of the classes.

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

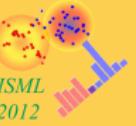
# Probabilistic Generative Models

- Generative approach: model class-conditional densities  $p(\mathbf{x} | \mathcal{C}_k)$  and priors  $p(\mathcal{C}_k)$  to calculate the posterior probability for class  $\mathcal{C}_1$

$$\begin{aligned} p(\mathcal{C}_1 | \mathbf{x}) &= \frac{p(\mathbf{x} | \mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x} | \mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x} | \mathcal{C}_2)p(\mathcal{C}_2)} \\ &= \frac{1}{1 + \exp(-a(\mathbf{x}))} = \sigma(a(\mathbf{x})) \end{aligned}$$

where  $a$  and the logistic sigmoid function  $\sigma(a)$  are given by

$$\begin{aligned} a(\mathbf{x}) &= \ln \frac{p(\mathbf{x} | \mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x} | \mathcal{C}_2)p(\mathcal{C}_2)} = \ln \frac{p(\mathbf{x}, \mathcal{C}_1)}{p(\mathbf{x}, \mathcal{C}_2)} \\ \sigma(a) &= \frac{1}{1 + \exp(-a)}. \end{aligned}$$

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

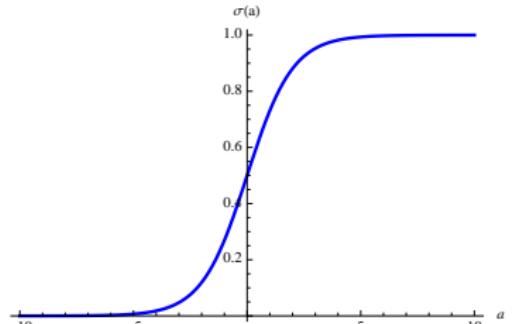
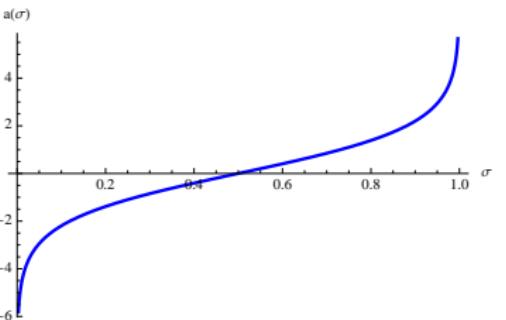
Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

- The logistic sigmoid function  $\sigma(a) = \frac{1}{1+\exp(-a)}$
- "squashing function" because it maps the real axis into a finite interval  $(0, 1)$
- $\sigma(-a) = 1 - \sigma(a)$
- Derivative  $\frac{d}{da}\sigma(a) = \sigma(a)\sigma(-a) = \sigma(a)(1 - \sigma(a))$
- Inverse is called logit function  $a(\sigma) = \ln\left(\frac{\sigma}{1-\sigma}\right)$

Logistic Sigmoid  $\sigma(a)$ Logit  $a(\sigma)$

# Probabilistic Generative Models - Multiclass



- The **normalised exponential** is given by

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

where

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

- Also called **softmax function** as it is a smoothed version of the max function.
- Example: If  $a_k \gg a_j$  for all  $j \neq k$ , then  $p(\mathcal{C}_k \mid \mathbf{x}) \simeq 1$ , and  $p(\mathcal{C}_j \mid \mathbf{x}) \simeq 0$ .

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

# Probabil. Generative Model - Continuous Input



- Assume class-conditional probabilities are Gaussian, all classes share the **same covariance**. What can we say about the posterior probabilities?

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

where we separated the quadratic term in  $\mathbf{x}$  and the linear term.

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

# Probabil. Generative Model - Continuous Input

Introduction to Statistical  
Machine Learning

©2012  
Christfried Webers  
NICTA

The Australian National  
University



- For two classes

$$p(\mathcal{C}_1 | \mathbf{x}) = \sigma(a(\mathbf{x}))$$

- and  $a(\mathbf{x})$  is

$$\begin{aligned} a(\mathbf{x}) &= \ln \frac{p(\mathbf{x} | \mathcal{C}_1) p(\mathcal{C}_1)}{p(\mathbf{x} | \mathcal{C}_2) p(\mathcal{C}_2)} \\ &= \ln \frac{\exp \left\{ \boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1} \mathbf{x} - \frac{1}{2} \boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 \right\}}{\exp \left\{ \boldsymbol{\mu}_2^T \boldsymbol{\Sigma}^{-1} \mathbf{x} - \frac{1}{2} \boldsymbol{\mu}_2^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_2 \right\}} + \ln \frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)} \end{aligned}$$

- Therefore

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

where

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

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

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

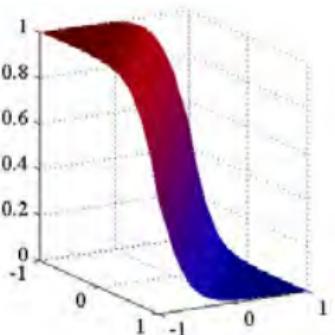
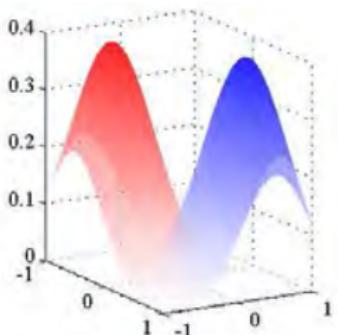
Laplace Approximation

Bayesian Logistic  
Regression

# Probabil. Generative Model - Continuous Input



Class-conditional densities for two classes (left). Posterior probability  $p(\mathcal{C}_1 | \mathbf{x})$  (right). Note the logistic sigmoid of a linear function of  $\mathbf{x}$ .



Probabilistic Generative  
Models

Continuous Input

Discrete Features

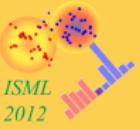
Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

# General Case - K Classes, Shared Covariance

- Use the normalised exponential

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

where

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

- to get a linear function of  $\mathbf{x}$

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

where

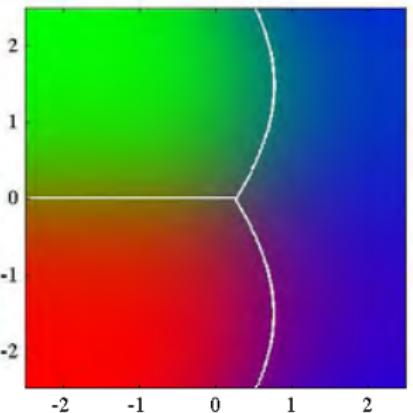
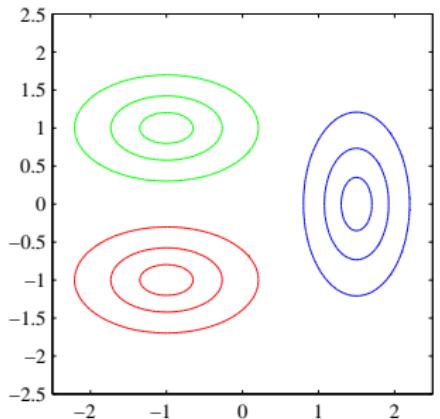
$$\mathbf{w}_k = \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k$$

$$w_{k0} = -\frac{1}{2} \boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k + p(\mathcal{C}_k).$$



# General Case - K Classes, Different Covariance

- If each class-conditional probability has a different covariance, the quadratic terms  $-\frac{1}{2}\mathbf{x}^T \Sigma^{-1} \mathbf{x}$  do not longer cancel each other out.
- We get a quadratic discriminant.



Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

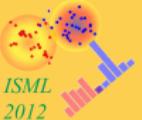
Bayesian Logistic  
Regression

# Maximum Likelihood Solution

- Given the functional form of the class-conditional densities  $p(\mathbf{x} | \mathcal{C}_k)$ , can we determine the parameters  $\mu$  and  $\Sigma$  ?
- Not without data ;-)
- Given also a data set  $(\mathbf{x}_n, t_n)$  for  $n = 1, \dots, N$ . (Using the coding scheme where  $t_n = 1$  corresponds to class  $\mathcal{C}_1$  and  $t_n = 0$  denotes class  $\mathcal{C}_2$ .)
- Assume the class-conditional densities to be Gaussian with the same covariance, but different mean.
- Denote the prior probability  $p(\mathcal{C}_1) = \pi$ , and therefore  $p(\mathcal{C}_2) = 1 - \pi$ .
- Then

$$p(\mathbf{x}_n, \mathcal{C}_1) = p(\mathcal{C}_1)p(\mathbf{x}_n | \mathcal{C}_1) = \pi \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_1, \boldsymbol{\Sigma})$$

$$p(\mathbf{x}_n, \mathcal{C}_2) = p(\mathcal{C}_2)p(\mathbf{x}_n | \mathcal{C}_2) = (1 - \pi) \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_2, \boldsymbol{\Sigma})$$

Probabilistic Generative  
Models

Continuous Input

Discrete Features

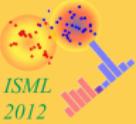
Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

# Maximum Likelihood Solution

- Thus the likelihood for the whole data set  $\mathbf{X}$  and  $\mathbf{t}$  is given by

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

- Maximise the log likelihood
- The term depending on  $\pi$  is

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

- which is maximal for

$$\pi = \frac{1}{N} \sum_{n=1}^N t_n = \frac{N_1}{N} = \frac{N_1}{N_1 + N_2}$$

where  $N_1$  is the number of data points in class  $\mathcal{C}_1$ .

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

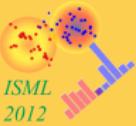
# Maximum Likelihood Solution

- Similarly, we can maximise the log likelihood (and thereby the likelihood  $p(\mathbf{t}, \mathbf{X} | \pi, \mu_1, \mu_2, \Sigma)$ ) depending on the mean  $\mu_1$  or  $\mu_2$ , and get

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

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

- For each class, this are the means of all input vectors assigned to this class.



- Finally, the log likelihood  $\ln p(\mathbf{t}, \mathbf{X} | \pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma)$  can be maximised for the covariance  $\Sigma$  resulting in

$$\Sigma = \frac{N_1}{N} \mathbf{S}_1 + \frac{N_2}{N} \mathbf{S}_2$$

$$\mathbf{S}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T$$

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

- 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 **Naive Bayes** assumption is that all features conditioned on the class  $\mathcal{C}_k$  are independent of each other.

$$p(\mathbf{x} | \mathcal{C}_k) = \prod_{i=1}^D \mu_{k_i}^{x_i} (1 - \mu_{k_i})^{1-x_i}$$

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

# Discrete Features - Naive Bayes

- With the naive Bayes

$$p(\mathbf{x} \mid \mathcal{C}_k) = \prod_{i=1}^D \mu_{k_i}^{x_i} (1 - \mu_{k_i})^{1-x_i}$$

- we can then again find the factors  $a_k$  in the normalised exponential

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

- as a linear function of the  $x_i$

$$a_k(\mathbf{x}) = \sum_{i=1}^D \{x_i \ln \mu_{k_i} + (1 - x_i) \ln(1 - \mu_{k_i})\} + \ln p(\mathcal{C}_k).$$



# Three Models for Decision Problems

In increasing order of complexity

- Find a discriminant function  $f(\mathbf{x})$  which maps each input directly onto a class label.

- **Discriminative Models**

- ➊ Solve the inference problem of determining the posterior class probabilities  $p(\mathcal{C}_k | \mathbf{x})$ .
- ➋ Use decision theory to assign each new  $\mathbf{x}$  to one of the classes.

- **Generative Models**

- ➊ Solve the inference problem of determining the class-conditional probabilities  $p(\mathbf{x} | \mathcal{C}_k)$ .
- ➋ Also, infer the prior class probabilities  $p(\mathcal{C}_k)$ .
- ➌ Use Bayes' theorem to find the posterior  $p(\mathcal{C}_k | \mathbf{x})$ .
- ➍ Alternatively, model the joint distribution  $p(\mathbf{x}, \mathcal{C}_k)$  directly.
- ➎ Use decision theory to assign each new  $\mathbf{x}$  to one of the classes.

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

# Probabilistic Discriminative Models



- Maximise a likelihood function defined through the conditional distribution  $p(\mathcal{C}_k | \mathbf{x})$  directly.
- **Discriminative** training
- Typically fewer parameters to be determined.
- As we learn the posterior  $p(\mathcal{C}_k | \mathbf{x})$  directly, prediction may be better than with a generative model where the class-conditional density assumptions  $p(\mathbf{x} | \mathcal{C}_k)$  poorly approximate the true distributions.
- But: discriminative models can not create synthetic data, as  $p(\mathbf{x})$  is not modelled.

Probabilistic Generative  
Models

Continuous Input

Discrete Features

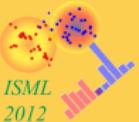
Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

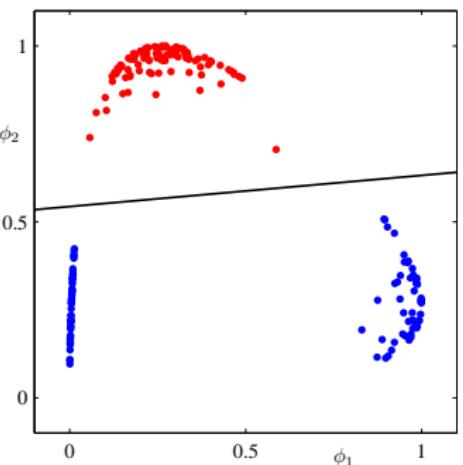
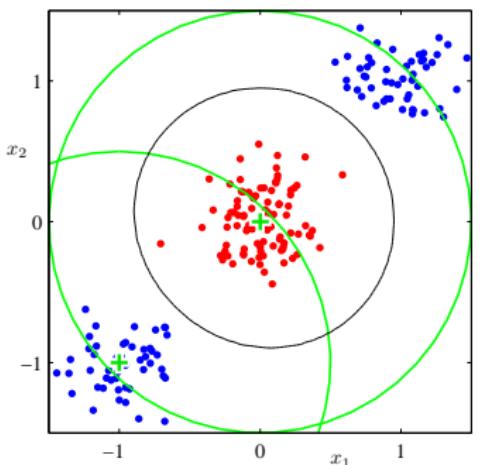
Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

- Used direct input  $\mathbf{x}$  until now.
- All classification algorithms work also if we first apply a fixed nonlinear transformation of the inputs using a vector of basis functions  $\phi(\mathbf{x})$ .
- Example: Use two Gaussian basis functions centered at the green crosses in the input space.



Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

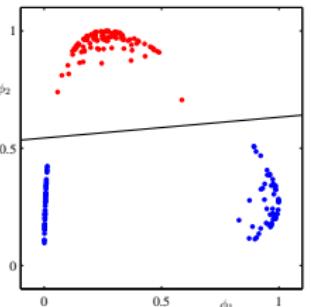
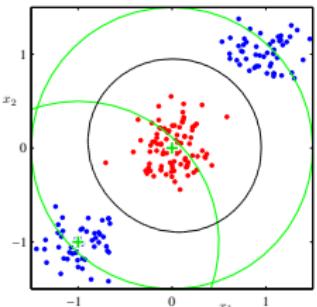
Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

# Original Input versus Feature Space

- Linear decision boundaries in the feature space correspond to nonlinear decision boundaries in the input space.
- Classes which are NOT linearly separable in the input space can become linearly separable in the feature space.
- BUT: If classes overlap in input space, they will also overlap in feature space.
- Nonlinear features  $\phi(\mathbf{x})$  can not remove the overlap; but they may increase it !



# Original Input versus Feature Space



- Fixed basis functions do not adapt to the data and therefore have important limitations (see discussion in Linear Regression).
- Understanding of more advanced algorithms becomes easier if we introduce the feature space now and use it instead of the original input space.
- Some applications use fixed features successfully by avoiding the limitations.
- We will therefore use  $\phi$  instead of  $x$  from now on.

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

# Logistic Regression is Classification

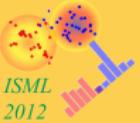
- Two classes where the posterior of class  $\mathcal{C}_1$  is a logistic sigmoid  $\sigma()$  acting on a linear function of the feature vector  $\phi$

$$p(\mathcal{C}_1 | \phi) = y(\phi) = \sigma(\mathbf{w}^T \phi)$$

- $p(\mathcal{C}_2 | \phi) = 1 - p(\mathcal{C}_1 | \phi)$
- Model dimension is equal to dimension of the feature space  $M$ .
- Compare this to fitting two Gaussians

$$\underbrace{2M}_{\text{means}} + \underbrace{M(M+1)/2}_{\text{shared covariance}} = M(M+5)/2$$

- For larger  $M$ , the logistic regression model has a clear advantage.

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

# Logistic Regression is Classification

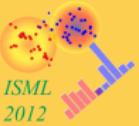
- Determine the parameter via maximum likelihood for data  $(\phi_n, t_n)$ ,  $n = 1, \dots, N$ , where  $\phi_n = \phi(\mathbf{x}_n)$ . The class membership is coded as  $t_n \in \{0, 1\}$ .
- Likelihood function

$$p(\mathbf{t} | \mathbf{w}) = \prod_{n=1}^N y_n^{t_n} (1 - y_n)^{1-t_n}$$

where  $y_n = p(\mathcal{C}_1 | \phi_n)$ .

- Error function : negative log likelihood resulting in the cross-entropy error function

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

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

# *Logistic Regression is Classification*

- Error function (**cross-entropy** error )

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

- $y_n = p(\mathcal{C}_1 | \phi_n) = \sigma(\mathbf{w}^T \phi_n)$
- Gradient of the error function (using  $\frac{d\sigma}{da} = \sigma(1 - \sigma)$  )

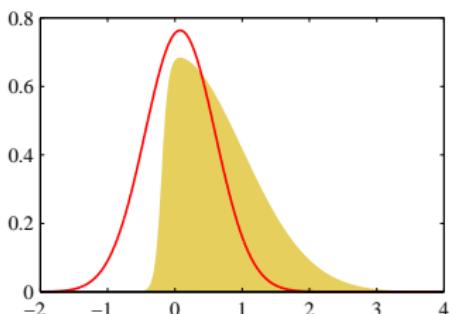
$$\nabla E(\mathbf{w}) = \sum_{n=1}^N (y_n - t_n) \phi_n$$

- gradient does not contain any sigmoid function
- for each data point error is product of deviation  $y_n - t_n$  and basis function  $\phi_n$ .
- BUT : maximum likelihood solution can exhibit over-fitting even for many data points; should use regularised error or MAP then.

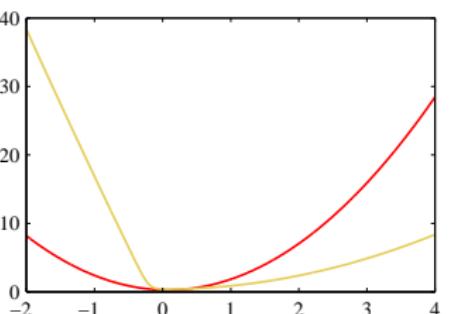


# Laplace Approximation

- Given a continuous distribution  $p(x)$  which is not Gaussian, can we approximate it by a Gaussian  $q(x)$  ?
- Need to find a mode of  $p(x)$ . Try to find a Gaussian with the same mode.



Non-Gaussian (yellow) and Gaussian approximation (red).



Negative log of the Non-Gaussian (yellow) and Gaussian approx. (red).

Probabilistic Generative Models

Continuous Input

Discrete Features

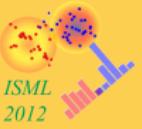
Probabilistic Discriminative Models

Logistic Regression

Iterative Reweighted Least Squares

Laplace Approximation

Bayesian Logistic Regression

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

# Laplace Approximation

- Assume  $p(x)$  can be written as

$$p(z) = \frac{1}{Z} f(z)$$

with normalisation  $Z = \int f(z) dz$ .

- Furthermore, assume  $Z$  is unknown !
- A mode of  $p(z)$  is at a point  $z_0$  where  $p'(z_0) = 0$ .
- Taylor expansion of  $\ln f(z)$  at  $z_0$

$$\ln f(z) \simeq \ln f(z_0) - \frac{1}{2} A(z - z_0)^2$$

where

$$A = -\frac{d^2}{dz^2} \ln f(z) |_{z=z_0}$$

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

# Laplace Approximation

- Exponentiating

$$\ln f(z) \simeq \ln f(z_0) - \frac{1}{2} A(z - z_0)^2$$

- we get

$$f(z) \simeq f(z_0) \exp\left\{-\frac{A}{2}(z - z_0)^2\right\}.$$

- And after normalisation we get the Laplace approximation

$$q(z) = \left(\frac{A}{2\pi}\right)^{1/2} \exp\left\{-\frac{A}{2}(z - z_0)^2\right\}.$$

- Only defined for precision  $A > 0$  as only then  $p(z)$  has a maximum.

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

# Laplace Approximation - Vector Space

- Approximate  $p(\mathbf{z})$  for  $\mathbf{z} \in \mathbb{R}^M$

$$p(\mathbf{z}) = \frac{1}{Z} f(\mathbf{z}).$$

- we get the Taylor expansion

$$\ln f(\mathbf{z}) \simeq \ln f(\mathbf{z}_0) - \frac{1}{2} (\mathbf{z} - \mathbf{z}_0)^T \mathbf{A} (\mathbf{z} - \mathbf{z}_0)$$

- where the Hessian  $\mathbf{A}$  is defined as

$$\mathbf{A} = -\nabla \nabla \ln f(\mathbf{z}) \mid_{\mathbf{z}=\mathbf{z}_0}.$$

- The Laplace approximation of  $p(\mathbf{z})$  is then

$$\begin{aligned} q(\mathbf{z}) &= \frac{|\mathbf{A}|^{1/2}}{(2\pi)^{M/2}} \exp \left\{ -\frac{1}{2} (\mathbf{z} - \mathbf{z}_0)^T \mathbf{A} (\mathbf{z} - \mathbf{z}_0) \right\} \\ &= \mathcal{N}(\mathbf{z} \mid \mathbf{z}_0, \mathbf{A}^{-1}) \end{aligned}$$

# Bayesian Logistic Regression



- Exact Bayesian inference for the logistic regression is intractable.
- Why? Need to normalise a product of prior probabilities and likelihoods which itself are a product of logistic sigmoid functions, one for each data point.
- Evaluation of the predictive distribution also intractable.
- Therefore we will use the Laplace approximation.

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression



- Assume a Gaussian prior because we want a Gaussian posterior.

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

for fixed **hyperparameter**  $\mathbf{m}_0$  and  $\mathbf{S}_0$ .

- Hyperparameters** are parameters of a prior distribution. In contrast to the model parameters  $\mathbf{w}$ , they are not learned.
- For a set of training data  $(\mathbf{x}_n, t_n)$ , where  $n = 1, \dots, N$ , the posterior is given by

$$p(\mathbf{w} | \mathbf{t}) \propto p(\mathbf{w})p(\mathbf{t} | \mathbf{w})$$

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

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

# Bayesian Logistic Regression

- Using our previous result for the cross-entropy function

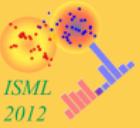
$$E(\mathbf{w}) = -\ln p(\mathbf{t} | \mathbf{w}) = -\sum_{n=1}^N \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}$$

we can now calculate the log of the posterior

$$p(\mathbf{w} | \mathbf{t}) \propto p(\mathbf{w})p(\mathbf{t} | \mathbf{w})$$

using the notation  $y_n = \sigma(\mathbf{w}^T \phi_n)$  as

$$\begin{aligned}\ln p(\mathbf{w} | \mathbf{t}) &= -\frac{1}{2}(\mathbf{w} - \mathbf{m}_0)^T \mathbf{S}_0^{-1} (\mathbf{w} - \mathbf{m}_0) \\ &\quad + \sum_{n=1}^N \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}\end{aligned}$$

Probabilistic Generative  
Models

Continuous Input

Discrete Features

Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression

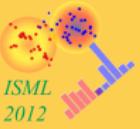
# Bayesian Logistic Regression

- To obtain a Gaussian approximation to

$$\begin{aligned}\ln p(\mathbf{w} \mid \mathbf{t}) = & -\frac{1}{2}(\mathbf{w} - \mathbf{m}_0)^T \mathbf{S}_0^{-1}(\mathbf{w} - \mathbf{m}_0) \\ & + \sum_{n=1}^N \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}\end{aligned}$$

- Find  $\mathbf{w}_{MAP}$  which maximises  $\ln p(\mathbf{w} \mid \mathbf{t})$ . This defines the mean of the Gaussian approximation. (Note: This is a nonlinear function in  $\mathbf{w}$  because  $y_n = \sigma(\mathbf{w}^T \phi_n)$ .)
- Calculate the second derivative of the negative log likelihood to get the inverse covariance of the Laplace approximation

$$\mathbf{S}_N = -\nabla \nabla \ln p(\mathbf{w} \mid \mathbf{t}) = \mathbf{S}_0^{-1} + \sum_{n=1}^N y_n(1 - y_n) \phi_n \phi_n^T.$$



- The approximated Gaussian (via Laplace approximation) of the posterior distribution is now

$$q(\mathbf{w} | \phi) = \mathcal{N}(\mathbf{w} | \mathbf{w}_{MAP}, \mathbf{S}_N)$$

where

$$\mathbf{S}_N = -\nabla \nabla \ln p(\mathbf{w} | \mathbf{t}) = \mathbf{S}_0^{-1} + \sum_{n=1}^N y_n(1-y_n)\phi_n\phi_n^T.$$

Probabilistic Generative  
Models

Continuous Input

Discrete Features

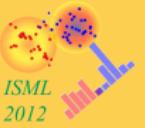
Probabilistic  
Discriminative Models

Logistic Regression

Iterative Reweighted  
Least Squares

Laplace Approximation

Bayesian Logistic  
Regression



## Part IX

### *Neural Networks I*

*Neural Networks*

*Weight-space Symmetries*

*Parameter Optimisation*

*Gradient Descent  
Optimisation*



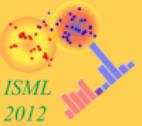
*Neural Networks*

*Weight-space Symmetries*

*Parameter Optimisation*

*Gradient Descent  
Optimisation*

- The basis functions play a crucial role in the algorithms explored so far.
- Number and parameters of basis functions fixed before learning starts (e.g. Linear Regression and Linear Classification).
- Number of basis functions fixed, parameters of the basis functions are adaptive (e.g. Neural Networks).
- Center basis function on the data, select a subset of basis functions in the training phase (e.g. Support Vector Machines, Relevance Vector Machines).



Neural Networks

Weight-space Symmetries

Parameter Optimisation

Gradient Descent  
Optimisation

- The functional form of the network model (including special parametrisation of the basis functions).
- How to determine the network parameters within the maximum likelihood framework? (Solution of a nonlinear optimisation problem.)
- **Error backpropagation** : efficiently evaluate the derivatives of the log likelihood function with respect to the network parameters.
- Various approaches to regularise neural networks.



Neural Networks

Weight-space Symmetries

Parameter Optimisation

Gradient Descent  
Optimisation

# Feed-forward Network Functions

- Same goal as before: Decompose the target  $t$

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

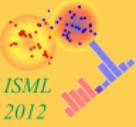
where  $\epsilon(\mathbf{x})$  is the residual error.

- (Generalised) Linear Model

$$y(\mathbf{x}, \mathbf{w}) = f \left( \sum_{j=0}^M w_j \phi_j(\mathbf{x}) \right)$$

where  $\phi = (\phi_0, \dots, \phi_M)^T$  is the fixed model basis and  $\mathbf{w} = (w_0, \dots, w_M)^T$  are the model parameter.

- For regression:  $f(\cdot)$  is the identity function.
- For classification:  $f(\cdot)$  is a nonlinear activation function.
- Goal : Let  $\phi_j(\mathbf{x})$  depend on parameters, and then adjust these parameters together with  $\mathbf{w}$ .



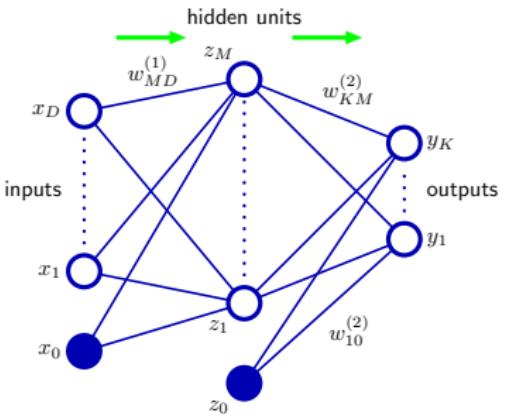
Neural Networks

Weight-space Symmetries

Parameter Optimisation

Gradient Descent  
Optimisation

- Goal : Let  $\phi_j(\mathbf{x})$  depend on parameters, and then adjust these parameters together with  $\mathbf{w}$ .
- Many ways to do this.
- Neural networks use basis functions which follow the same form as the (generalised) linear model.
- EACH basis function is itself a nonlinear function of an adaptive linear combination of the inputs.





Neural Networks

Weight-space Symmetries

Parameter Optimisation

Gradient Descent  
Optimisation

# Functional Transformations

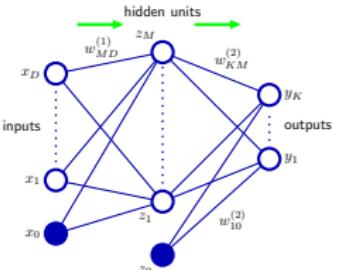
- Construct  $M$  linear combinations of the input variables  $x_1, \dots, x_D$  in the form

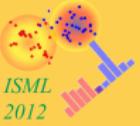
$$\underbrace{a_j}_{\text{activations}} = \sum_{i=1}^D \underbrace{w_{ji}^{(1)}}_{\text{weights}} x_i + \underbrace{w_{j0}^{(1)}}_{\text{bias}} \quad j = 1, \dots, M$$

- Apply a differentiable, nonlinear **activation function**  $h(\cdot)$  to get the output of the **hidden units**

$$z_j = h(a_j)$$

- $h(\cdot)$  is typically sigmoidal or tanh.





Neural Networks

Weight-space Symmetries

Parameter Optimisation

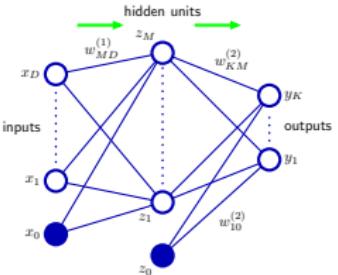
Gradient Descent  
Optimisation

- The outputs of the hidden units are again linearly combined

$$a_k = \sum_{j=1}^M w_{kj}^{(2)} z_j + w_{k0}^{(2)} \quad k = 1, \dots, K$$

- Apply again a differentiable, nonlinear activation function  $g(\cdot)$  to get the network outputs  $y_k$

$$y_k = g(a_k)$$





Neural Networks

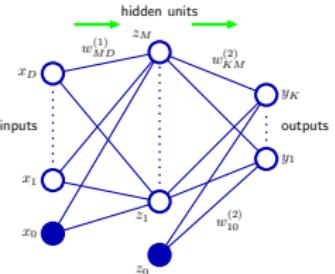
Weight-space Symmetries

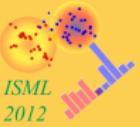
Parameter Optimisation

Gradient Descent  
Optimisation

- The activation function  $g(\cdot)$  is determined by the nature of the data and the distribution of the target variables.
- For standard regression:  $g(\cdot)$  is the identity function so that  $y_k = a_k$ .
- For multiple binary classification,  $g(\cdot)$  is a logistic sigmoid function

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





Neural Networks

Weight-space Symmetries

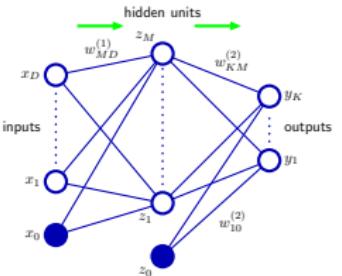
Parameter Optimisation

Gradient Descent  
Optimisation

- Combine all transformations into one formula

$$y_k(\mathbf{x}, \mathbf{w}) = g \left( \sum_{j=1}^M w_{kj}^{(2)} h \left( \sum_{i=1}^D w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right)$$

where  $\mathbf{w}$  contains all weight and bias parameters.





Neural Networks

Weight-space Symmetries

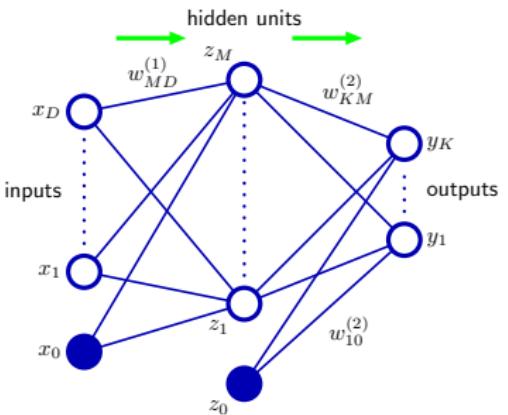
Parameter Optimisation

Gradient Descent  
Optimisation

- As before, the biases can be absorbed into the weights by introducing an extra input  $x_0 = 1$  and a hidden unit  $z_0 = 1$ .

$$y_k(\mathbf{x}, \mathbf{w}) = g \left( \sum_{j=0}^M w_{kj}^{(2)} h \left( \sum_{i=0}^D w_{ji}^{(1)} x_i \right) \right)$$

where  $\mathbf{w}$  now contains all weight and bias parameters.





Neural Networks

Weight-space Symmetries

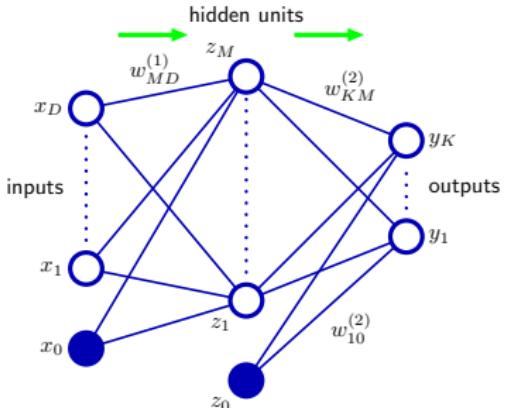
Parameter Optimisation

Gradient Descent  
Optimisation

- A neural network looks like a **multilayer perceptron**.
- But perceptron's nonlinear activation function was a step function. Not smooth. Not differentiable.

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

- The activation functions  $h(\cdot)$  and  $g(\cdot)$  of a neural network are smooth and differentiable.



# Linear Activation Functions



- If all activation functions are linear functions then there exists an equivalent network without hidden units.  
(Composition of linear functions is a linear function.)
- But if the number of hidden units in this case is smaller than the number of input or output units, the resulting linear function are not the most general.
- Dimensionality reduction.
- Principal Component Analysis (comes later in the lecture).
- Generally, most neural networks use nonlinear activation functions as the goal is to approximate a nonlinear mapping from the input space to the outputs.

Neural Networks

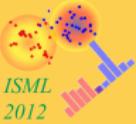
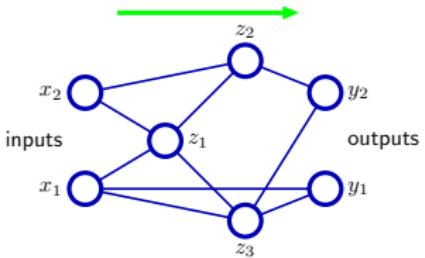
Weight-space Symmetries

Parameter Optimisation

Gradient Descent  
Optimisation

# Extensions of the Neural Network Architecture

- Add more hidden layers.
- Some units may be not fully connected to the next layer.
- Some links may skip over one or several subsequent layer(s).
- Still in the framework of **feed-forward** networks.
- Note: If information is allowed to flow also backwards, we get a graph with cycles. This is called a **recurrent** neural network which can exhibit a very different dynamical behaviour (e.g. may have state, may exhibit chaos). Not further considered here.

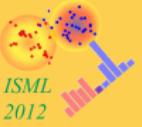


Neural Networks

Weight-space Symmetries

Parameter Optimisation

Gradient Descent  
Optimisation



Neural Networks

Weight-space Symmetries

Parameter Optimisation

Gradient Descent  
Optimisation

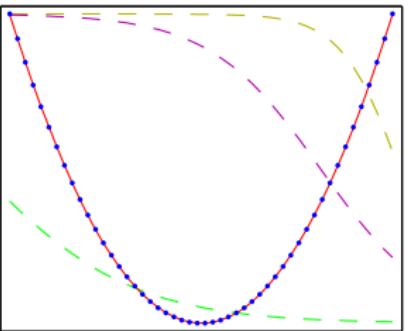
- Feed-forward neural networks are **universal approximators**.
- Example: A two-layer neural network with linear outputs can uniformly approximate any continuous function on a compact input domain to arbitrary accuracy if it has enough hidden units.
- Holds for a wide range of hidden unit activation functions, but NOT for polynomials.
- Remaining big question : Where do we get the appropriate settings for the weights from? With other words, how do we learn the weights from training examples?



# Aproximation Capabilities of Neural Networks

- Neural network approximating

$$f(x) = x^2$$



Two-layer network with 3 hidden units (tanh activation functions) and linear outputs trained on 50 data points sampled from the interval  $(-1, 1)$ . Red: resulting output. Dashed: Output of the hidden units.

Neural Networks

Weight-space Symmetries

Parameter Optimisation

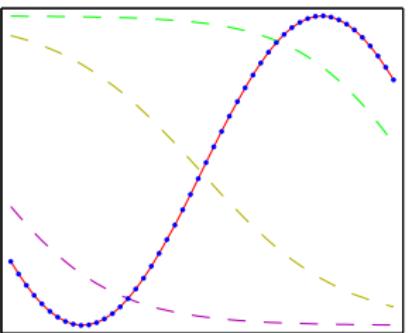
Gradient Descent  
Optimisation



# Aproximation Capabilities of Neural Networks

- Neural network approximating

$$f(x) = \sin(x)$$



Two-layer network with 3 hidden units (tanh activation functions) and linear outputs trained on 50 data points sampled from the interval  $(-1, 1)$ . Red: resulting output. Dashed: Output of the hidden units.

Neural Networks

Weight-space Symmetries

Parameter Optimisation

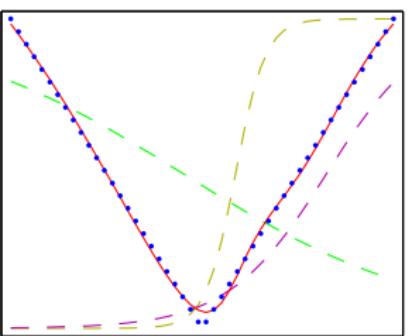
Gradient Descent  
Optimisation



# Aproximation Capabilities of Neural Networks

- Neural network approximating

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



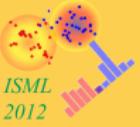
Two-layer network with 3 hidden units (tanh activation functions) and linear outputs trained on 50 data points sampled from the interval  $(-1, 1)$ . Red: resulting output. Dashed: Output of the hidden units.

Neural Networks

Weight-space Symmetries

Parameter Optimisation

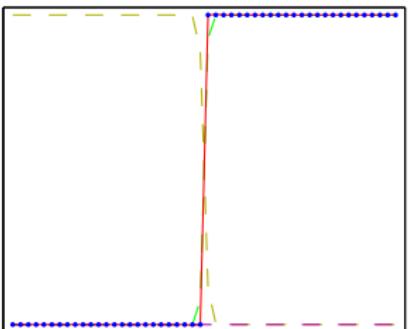
Gradient Descent  
Optimisation



Neural Networks

Weight-space Symmetries

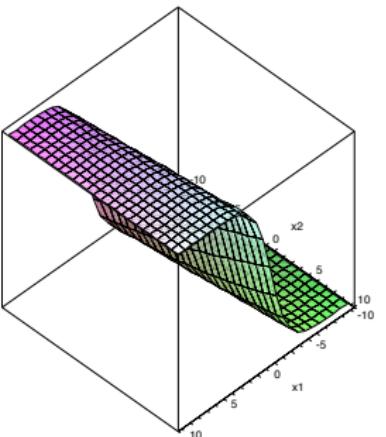
Parameter Optimisation

Gradient Descent  
Optimisation

Two-layer network with 3 hidden units (tanh activation functions) and linear outputs trained on 50 data points sampled from the interval  $(-1, 1)$ . Red: resulting output. Dashed: Output of the hidden units.



- Hidden layer nodes represent parametrised basis functions



$$z = \sigma(w_0 + w_1 x_1 + w_2 x_2) \text{ for } (w_0, w_1, w_2) = (0.0, 1.0, 0.1)$$

Neural Networks

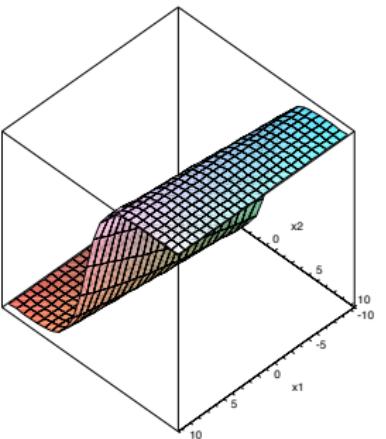
Weight-space Symmetries

Parameter Optimisation

Gradient Descent  
Optimisation



- Hidden layer nodes represent parametrised basis functions



$$z = \sigma(w_0 + w_1x_1 + w_2x_2) \text{ for } (w_0, w_1, w_2) = (0.0, 0.1, 1.0)$$

Neural Networks

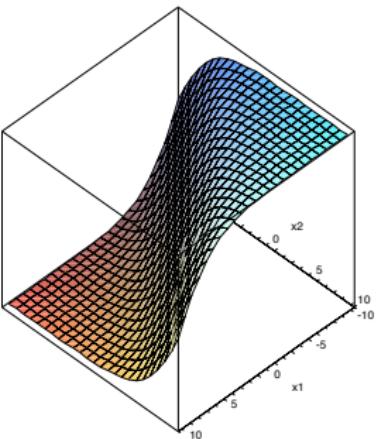
Weight-space Symmetries

Parameter Optimisation

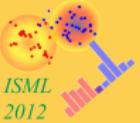
Gradient Descent  
Optimisation

# Variable Basis Functions in a Neural Networks

- Hidden layer nodes represent parametrised basis functions



$$z = \sigma(w_0 + w_1 x_1 + w_2 x_2) \text{ for } (w_0, w_1, w_2) = (0.0, -0.5, 0.5)$$



Neural Networks

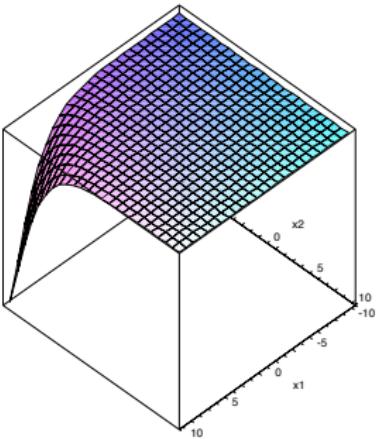
Weight-space Symmetries

Parameter Optimisation

Gradient Descent  
Optimisation



- Hidden layer nodes represent parametrised basis functions



$$z = \sigma(w_0 + w_1 x_1 + w_2 x_2) \text{ for } (w_0, w_1, w_2) = (10.0, -0.5, 0.5)$$

Neural Networks

Weight-space Symmetries

Parameter Optimisation

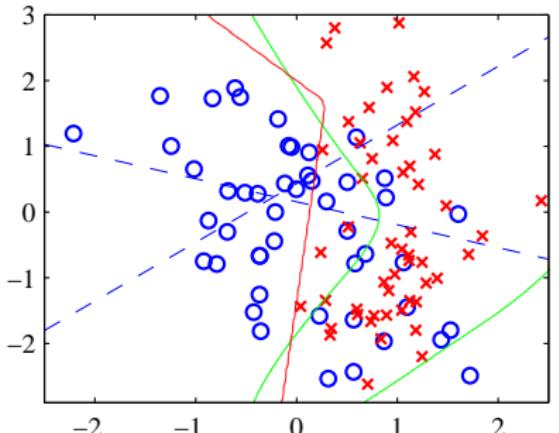
Gradient Descent  
Optimisation



Neural Networks

Weight-space Symmetries

Parameter Optimisation

Gradient Descent  
Optimisation

Red:  $y = 0.5$  decision boundary. Dashed blue:  $z = 0.5$  hidden unit contours. Green: Optimal decision boundary from the known data distribution.



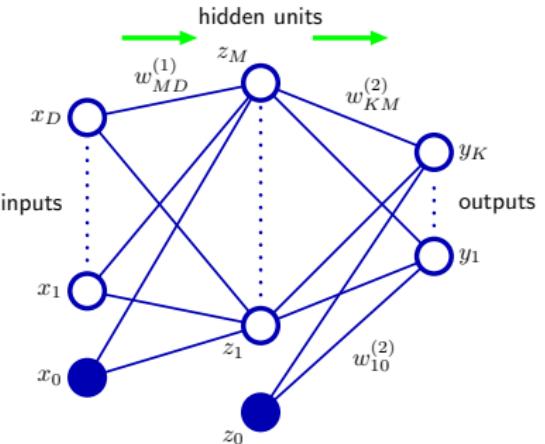
Neural Networks

Weight-space Symmetries

Parameter Optimisation

Gradient Descent  
Optimisation

- Given a set of weights  $w$ . This fixes a mapping from the input space to the output space.
- Does there exist another set of weights realising the same mapping?
- Assume tanh activation function for the hidden units. As tanh is an odd function:  $\tanh(-a) = -\tanh(a)$ .
- Change the sign of all inputs to a hidden unit and outputs of this hidden unit: Mapping stays the same.





Neural Networks

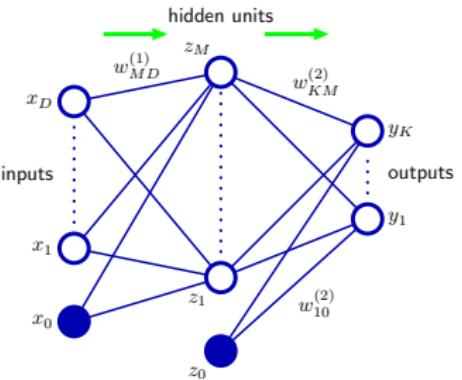
Weight-space Symmetries

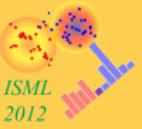
Parameter Optimisation

Gradient Descent  
Optimisation

- $M$  hidden units, therefore  $2^M$  equivalent weight vectors.
- Furthermore, exchange all of the weights going into and out of a hidden unit with the corresponding weights of another hidden unit. Mapping stays the same.  $M!$  symmetries.
- Overall weight space symmetry :  $M! 2^M$

$M$	1	2	3	4	5	6	7
$M! 2^M$	2	8	48	384	3840	46080	645120





Neural Networks

Weight-space Symmetries

Parameter Optimisation

Gradient Descent  
Optimisation

# Parameter Optimisation

- Assume the error  $E(\mathbf{w})$  is a smooth function of the weights.
- Smallest value will occur at a **critical point** for which

$$\nabla E(\mathbf{w}) = 0.$$

- This could be a minimum, maximum, or saddle point.
- Furthermore, because of symmetry in weight space, there are at least  $M! 2^M$  other critical points with the same value for the error.



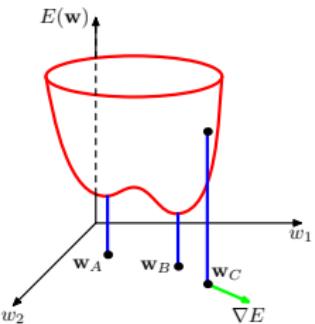
# Parameter Optimisation

## 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  $\mathbf{w}^*$ .



Neural Networks

Weight-space Symmetries

Parameter Optimisation

Gradient Descent  
Optimisation



Neural Networks

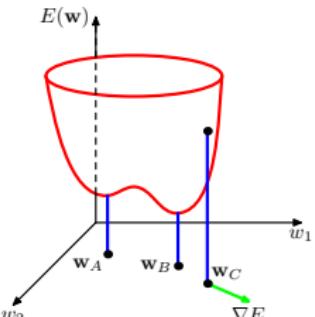
Weight-space Symmetries

Parameter Optimisation

Gradient Descent  
Optimisation

- Finding the global minimum is difficult in general (would have to check everywhere) unless the error function comes from a special class (e.g. smooth convex functions have only one minimum).
- Error functions for neural networks are not convex (symmetries!).
- But finding a local minimum might be sufficient.
- Use iterative methods with weight vector update  $\Delta \mathbf{w}^{(\tau)}$  to find a local minimum.

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \Delta \mathbf{w}^{(\tau)}$$





Neural Networks

Weight-space Symmetries

Parameter Optimisation

Gradient Descent  
Optimisation

# Local Quadratic Approximation

- Around a minimum  $\mathbf{w}^*$  we can approximate

$$E(\mathbf{w}) \simeq E(\mathbf{w}^*) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^T \mathbf{H}(\mathbf{w} - \mathbf{w}^*),$$

where the Hessian  $\mathbf{H}$  is evaluated at  $\mathbf{w}^*$ .

- Using a set  $\{\mathbf{u}_i\}$  of orthonormal eigenvectors of  $\mathbf{H}$ ,

$$\mathbf{H}\mathbf{u}_i = \lambda_i \mathbf{u}_i,$$

to expand

$$\mathbf{w} - \mathbf{w}^* = \sum_i \alpha_i \mathbf{u}_i.$$

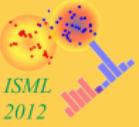
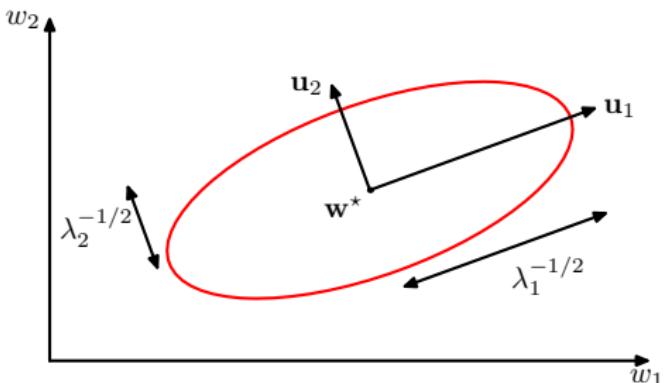
- We get

$$E(\mathbf{w}) = E(\mathbf{w}^*) + \frac{1}{2} \sum_i \lambda_i \alpha_i^2.$$

# Local Quadratic Approximation

- Around a minimum  $\mathbf{w}^*$  we can approximate

$$E(\mathbf{w}) = E(\mathbf{w}^*) + \frac{1}{2} \sum_i \lambda_i \alpha_i^2.$$



Neural Networks

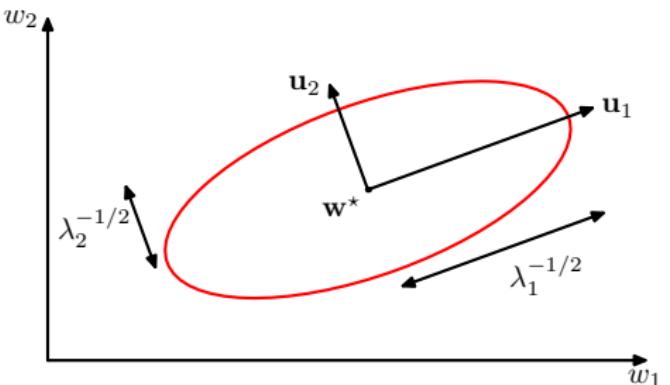
Weight-space Symmetries

Parameter Optimisation

Gradient Descent  
Optimisation



- Around a minimum  $\mathbf{w}^*$ , the Hessian  $\mathbf{H}$  must be positive definite if evaluated at  $\mathbf{w}^*$ .



Neural Networks

Weight-space Symmetries

Parameter Optimisation

Gradient Descent  
Optimisation

# Gradient Information improves Performances



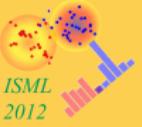
Neural Networks

Weight-space Symmetries

Parameter Optimisation

Gradient Descent  
Optimisation

- Hessian is symmetric and contains  $W(W + 1)/2$  independent entries where  $W$  is the total number of weights in the network.
- Need to gather this  $O(W^2)$  pieces of information by doing  $O(W)$  function evaluations if nothing else is known. Get order  $O(W^3)$ .
- The gradient  $\nabla E$  provides  $W$  pieces of information at once. Still need  $O(W)$  steps, but the order is now  $O(W^2)$ .



Neural Networks

Weight-space Symmetries

Parameter Optimisation

Gradient Descent  
Optimisation

# Gradient Descent Optimisation

- Batch processing : Update the weight vector with a small step in the direction of the negative gradient

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)})$$

where  $\eta$  is the learning rate.

- After each step, re-evaluate the gradient  $\nabla E(\mathbf{w}^{(\tau)})$  again.
- Gradient Descent has problems in 'long valleys'.

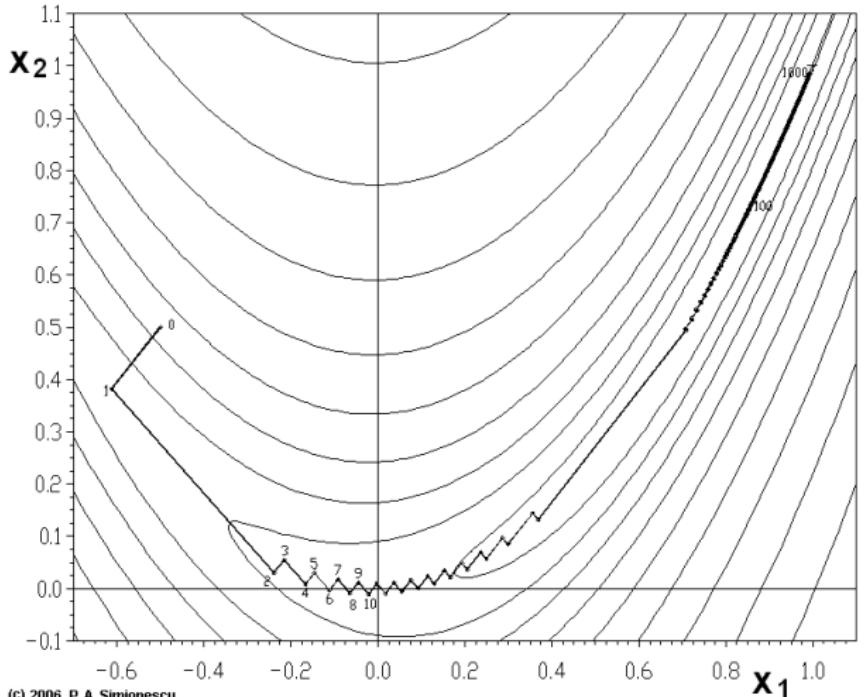


Neural Networks

Weight-space Symmetries

Parameter Optimisation

Gradient Descent  
Optimisation



(c) 2006 P.A. Simionescu

Example of zig-zag of Gradient Descent Algorithm.



Neural Networks

Weight-space Symmetries

Parameter Optimisation

Gradient Descent  
Optimisation

- Use Conjugate Gradient Descent instead of Gradient Descent to avoid zig-zag behaviour.
- Use Newton method which also calculates the inverse Hessian in each iteration (but inverting the Hessian is usually costly).
- Use Quasi-Newton methods (e.g. BFGS) which also calculates an estimate of the inverse Hessian while iterating.
- Run the algorithm from a set of starting points to find the smallest local minimum.

# Nonlinear Optimisation

©2012

Christfried Webers  
NICTAThe Australian National  
University

Neural Networks

Weight-space Symmetries

Parameter Optimisation

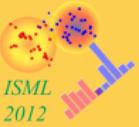
Gradient Descent  
Optimisation

- Remaining big problem: Error function is defined over the whole training set. Therefore, need to process the whole training set for each calculation of the gradient  $\nabla E(\mathbf{w}^{(\tau)})$ .
- If the error function is a sum of errors for each data point

$$E(\mathbf{w}) = \sum_{n=1}^N E_n(\mathbf{w})$$

we can use **on-line gradient descent** (also called **sequential gradient descent** or **stochastic gradient descent**) updating the weights by one data point at a time

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n(\mathbf{w}^{(\tau)}).$$



ISML  
2012

# Part X

## *Neural Networks 2*

*Error Backpropagation*

*Regularisation in Neural  
Networks*

*Bayesian Neural  
Networks*



## Error Backpropagation

Regularisation in Neural Networks

Bayesian Neural Networks

- Goal: Efficiently update the weights in order to find a local minimum of some error function  $E(\mathbf{w})$  utilizing the gradient of the error function.
- Core ideas :
  - ➊ Propagate the errors backwards through the network to efficiently calculate the gradient.
  - ➋ Update the weights using the calculated gradient.
- Sequential procedure : Calculate gradient and update weights for each data/target pair.
- Batch procedure : Collect gradient information for all data/target pairs for the same weight setting. Then adjust the weights.
- Main question in both cases: How to calculate the gradient of  $E(\mathbf{w})$  given one data/target pair?



# Error Backpropagation

- Assume the error is a sum over errors for each data/target pair

$$E(\mathbf{w}) = \sum_{n=1}^N E_n(\mathbf{w}).$$

- After applying input  $\mathbf{x}_n$  to the network, we get the output  $\mathbf{y}_n$  and calculate the error  $E_n(\mathbf{w})$ .
- What is the gradient for one such term  $E_n(\mathbf{w})$ ?
- Note : In the following, we will drop the  $n$  on weights  $\mathbf{w}$  and targets  $\mathbf{t}$  in order to unclutter the equations.
- Notation: Input pattern is  $\mathbf{x}_n$ .  
Scalar  $x_i$  is the  $i^{th}$  component of the input pattern  $\mathbf{x}_n$ .



Error Backpropagation

Regularisation in Neural  
NetworksBayesian Neural  
Networks

# Error Backpropagation - Simplified Model

- Simple linear model **without** hidden layers
- One layer only, no nonlinear activation function!

$$y_k = \sum_l w_{kl} x_l,$$

and error after applying input  $\mathbf{x}_n$

$$E_n(\mathbf{w}) = \frac{1}{2} \sum_{k=1} (y_k - t_k)^2.$$

- The gradient with respect to  $w_{ji}$  is now

$$\begin{aligned} \frac{\partial E_n(\mathbf{w})}{\partial w_{ji}} &= \sum_{k=1} (y_k - t_k) \frac{\partial}{\partial w_{ji}} y_k = \sum_{k=1} (y_k - t_k) \frac{\partial}{\partial w_{ji}} \sum_l w_{kl} x_l \\ &= \sum_{k=1} (y_k - t_k) \sum_l x_l \delta_{jk} \delta_{il} \\ &= (y_j - t_j) x_i. \end{aligned}$$



## Error Backpropagation

Regularisation in Neural  
NetworksBayesian Neural  
Networks

# Error Backpropagation - Simplified Model

- Do the same using the directional derivative:  
input vector  $\mathbf{x} \in \mathbb{R}^{D_1}$ , output vector  $\mathbf{y} \in \mathbb{R}^{D_2}$

$$\mathbf{y} = \mathbf{W}^T \mathbf{x},$$

and error after applying input training pair  $(\mathbf{x}, \mathbf{t})$

$$E_n(\mathbf{W}) = \frac{1}{2} (\mathbf{y} - \mathbf{t})^T (\mathbf{y} - \mathbf{t}).$$

- The directional derivative with respect to  $\mathbf{W}$  is now

$$\mathcal{D}E_n(\mathbf{W})(\xi) = \frac{1}{2} ((\xi^T \mathbf{x})^T (\mathbf{y} - \mathbf{t}) + (\mathbf{y} - \mathbf{t})^T \xi^T \mathbf{x}) = (\mathbf{y} - \mathbf{t})^T \xi^T \mathbf{x}$$

- With canonical inner product  $\langle A, B \rangle = \text{tr} \{ A^T B \}$  the gradient of  $E_n(\mathbf{W})(\xi)$  is

$$\mathcal{D}E_n(\mathbf{W})(\xi) = \text{tr} \left\{ \underbrace{(\mathbf{y} - \mathbf{t})^T \xi^T \mathbf{x}}_{\text{just a number}} \right\} = \text{tr} \left\{ \xi^T \mathbf{x} \underbrace{(\mathbf{y} - \mathbf{t})^T}_{\text{gradient}} \right\}$$



# Error Backpropagation

- The gradient

$$\frac{\partial E_n(\mathbf{w})}{\partial w_{ji}} = (y_j - t_j) x_i.$$

looks like the product of the output error ( $y_j - t_j$ ) with the input  $x_i$  associated with an edge for  $w_{ji}$  in the network diagram.

- Can we generalise this idea?

Error Backpropagation

Regularisation in Neural  
NetworksBayesian Neural  
Networks



Error Backpropagation

Regularisation in Neural  
NetworksBayesian Neural  
Networks

# Error Backpropagation

- Now consider a network with nonlinear activation functions  $h(\cdot)$  composed with the sum over the inputs  $z_i$  in one layer and  $z_j$  in the next layer connected by edges with weights  $w_{ji}$

$$a_j = \sum_i w_{ji} z_i$$

$$z_j = h(a_j).$$

- Use the chain rule to calculate the gradient

$$\frac{\partial E_n(\mathbf{w})}{\partial w_{ji}} = \frac{\partial E_n(\mathbf{w})}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} = \delta_j z_i,$$

where we defined the **error**  $\delta_j = \frac{\partial E_n(\mathbf{w})}{\partial a_j}$

- Same intuition as before: gradient is output error times the input associated with the edge for  $w_{ji}$ .



Error Backpropagation

Regularisation in Neural  
NetworksBayesian Neural  
Networks

# Error Backpropagation

- Need to calculate the errors  $\delta$  in EACH layer.

$$\frac{\partial E_n(\mathbf{w})}{\partial w_{ji}} = \delta_j z_i \quad \delta_j = \frac{\partial E_n(\mathbf{w})}{\partial a_j}$$

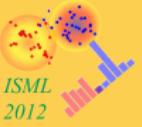
- For the output units, we have

$$\delta_k = y_k - t_k.$$

- For the hidden units we calculate

$$\delta_j = \frac{\partial E_n(\mathbf{w})}{\partial a_j} = \sum_k \frac{\partial E_n(\mathbf{w})}{\partial a_k} \frac{\partial a_k}{\partial a_j} = \sum_k \delta_k \frac{\partial a_k}{\partial a_j},$$

using the definition of  $\delta_k$ .



## Error Backpropagation

Regularisation in Neural  
NetworksBayesian Neural  
Networks

# Error Backpropagation

- Express  $a_k$  as a function of the incoming  $a_j$

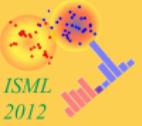
$$a_k = \sum_j w_{kj} z_j = \sum_j w_{kj} h(a_j),$$

- and differentiate

$$\frac{\partial a_k}{\partial a_j} = w_{kj} \frac{\partial h(a_j)}{\partial a_j} = w_{kj} \frac{\partial h(s)}{\partial s} \Big|_{s=a_j} = w_{kj} h'(a_j).$$

- Finally, we get for the error in the previous layer

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k.$$

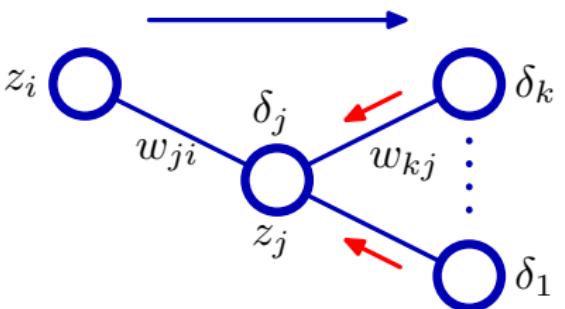


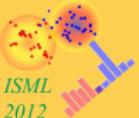
# Error Backpropagation

- The backpropagation formula

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k.$$

- Functional form of  $h'(\cdot)$  is known, because we choose the activation function  $h(\cdot)$ .



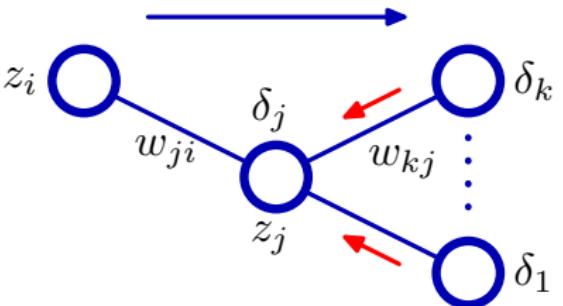


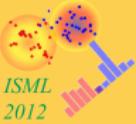
# Error Backpropagation Algorithms

- ➊ Apply the input vector  $\mathbf{x}$  to the network and forward propagate through the network to calculate all activations and outputs of each unit.
- ➋ Backpropagate the errors through the network.
- ➌ Calculate all components of  $\nabla E_n$  by

$$\frac{\partial E_n(\mathbf{w})}{\partial w_{ji}} = \delta_j z_i$$

- ➍ Update the weights  $\mathbf{w}$  using  $\frac{\partial E_n(\mathbf{w})}{\partial w_{ji}}$ .





# Error Backpropagation

- For batch processing, we repeat backpropagation for each pattern in the training set and then sum over all patterns

$$\frac{\partial E(\mathbf{w})}{\partial w_{ji}} = \sum_{n=1}^N \frac{\partial E_n(\mathbf{w})}{\partial w_{ji}}$$

- Backpropagation can be generalised by assuming that each node has a different activation function  $h(\cdot)$ .

Error Backpropagation

Regularisation in Neural  
Networks

Bayesian Neural  
Networks



Error Backpropagation

Regularisation in Neural  
NetworksBayesian Neural  
Networks

# Efficiency of Error Backpropagation

- As the number of weights is usually much larger than the number of units (the network is well connected), the complexity of calculating the gradient  $\frac{\partial E_n(\mathbf{w})}{\partial w_{ji}}$  via error backpropagation is of  $O(W)$  where  $W$  is the number of weights.
- Compare this to numerical differentiation using

$$\frac{\partial E_n(\mathbf{w})}{\partial w_{ji}} = \frac{E_n(w_{ji} + \epsilon) - E_n(w_{ji})}{\epsilon} + O(\epsilon)$$

or the numerically more stable (fewer round-off errors)  
**symmetric differences**

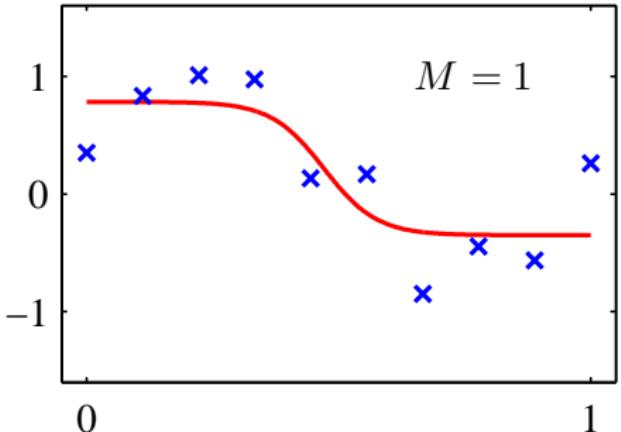
$$\frac{\partial E_n(\mathbf{w})}{\partial w_{ji}} = \frac{E_n(w_{ji} + \epsilon) - E_n(w_{ji} - \epsilon)}{2\epsilon} + O(\epsilon^2)$$

which both need  $O(W^2)$  operations.

# Regularisation in Neural Networks



- Number of input and output nodes determined by the application.
- Number of hidden nodes is a free parameter.



Training a two-layer network with 1 hidden node.

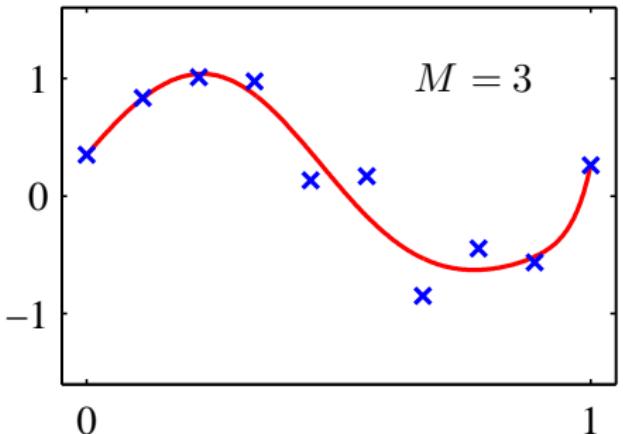
Error Backpropagation

Regularisation in Neural  
Networks

Bayesian Neural  
Networks



Error Backpropagation

Regularisation in Neural  
NetworksBayesian Neural  
Networks

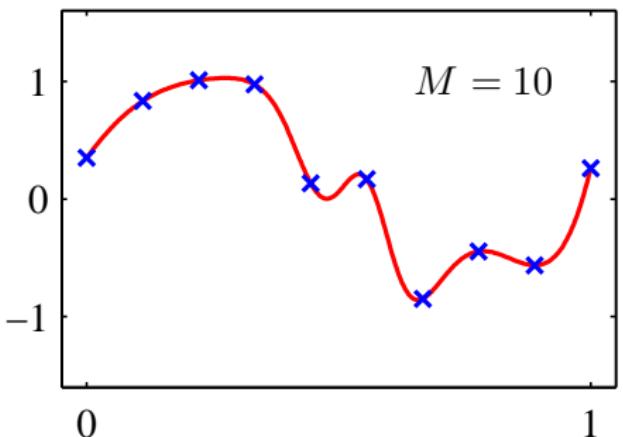
Training a two-layer network with 3 hidden nodes.



Error Backpropagation

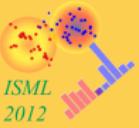
Regularisation in Neural  
Networks

Bayesian Neural  
Networks

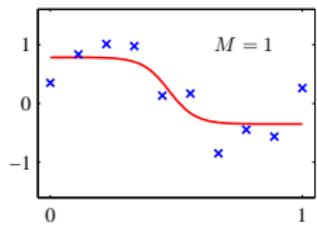


Training a two-layer network with 10 hidden nodes.

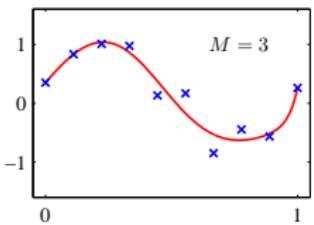
# Regularisation in Neural Networks



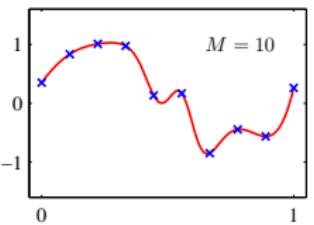
- Model complexity matters again.



$M = 1$



$M = 3$

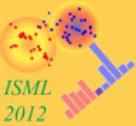


$M = 10$

Error Backpropagation

Regularisation in Neural  
Networks

Bayesian Neural  
Networks



Error Backpropagation

Regularisation in Neural  
Networks

Bayesian Neural  
Networks

- But the relation between generalisation error and the number of hidden units  $M$  is more complex than for fitting a polynomial. Reason : presence of local minima in the error function for the neural network.
- As before, we can use the **regularised error**

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \frac{1}{2} \mathbf{w}^T \mathbf{w}$$

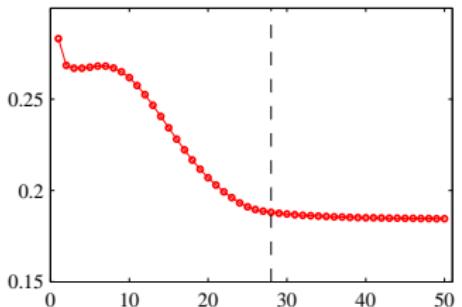


Error Backpropagation

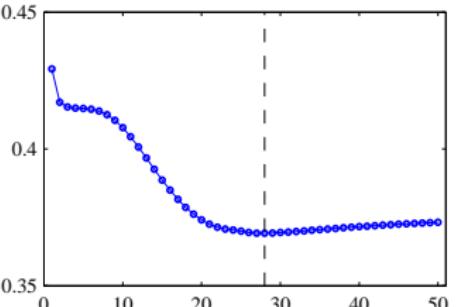
Regularisation in Neural  
Networks

Bayesian Neural  
Networks

- Stop training at the minimum of the validation set error.



Training set error.



Validation set error.

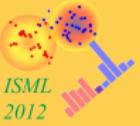


Error Backpropagation

Regularisation in Neural  
Networks

Bayesian Neural  
Networks

- If input data should be invariant with respect to some transformations, we can utilise this for training.
- Use training patterns including these transformations (e.g. handwritten digits translated in the input space).
- Or create extra artificial input data by applying several transformations to the original input data.
- Alternatively, preprocess the input data to remove the transformation.
- Or use **convolutional neural networks** (e.g. in image processing where close pixels are more correlated than far away pixels; therefore extract local features first and later feed into a network extracting higher-order features).

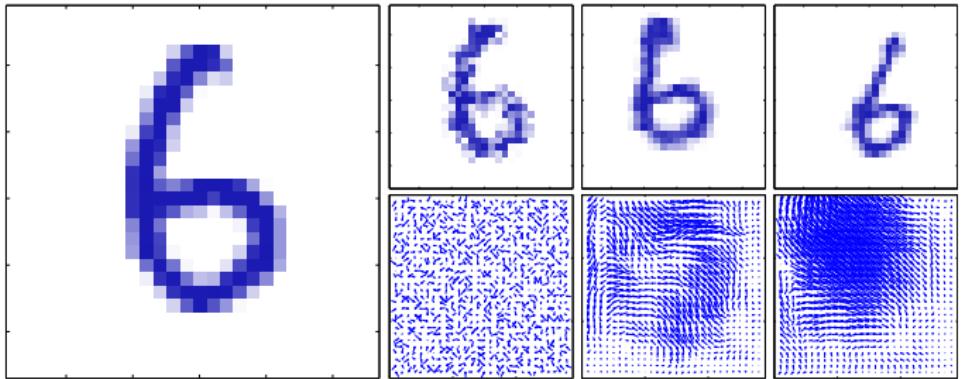


Error Backpropagation

Regularisation in Neural  
Networks

Bayesian Neural  
Networks

- Create synthetic data by warping handwritten digits.



Left: Original digitised image. Right : Examples of warped images (above) and their corresponding displacement fields (below).



Error Backpropagation

Regularisation in Neural  
NetworksBayesian Neural  
Networks

# Bayesian Neural Networks

- Predict a single target  $t$  from a vector of inputs  $\mathbf{x}$
- Assume conditional distribution to be Gaussian with precision  $\beta$

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

- Prior distribution over weights  $\mathbf{w}$  is also assumed to be Gaussian

$$p(\mathbf{w} \mid \alpha) = \mathcal{N}(\mathbf{w} \mid \mathbf{0}, \alpha^{-1})$$

- 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

$$p(\mathcal{D} \mid \mathbf{w}, \beta) = \prod_{n=1}^N \mathcal{N}(t_n \mid y(\mathbf{x}_n, \mathbf{w}), \beta^{-1})$$

- Posterior distribution

$$p(\mathbf{w} \mid \mathcal{D}, \alpha, \beta) \propto p(\mathbf{w} \mid \alpha) p(\mathcal{D} \mid \mathbf{w}, \beta)$$



Error Backpropagation

Regularisation in Neural  
NetworksBayesian Neural  
Networks

# Bayesian Neural Networks

- But  $y(\mathbf{x}, \mathbf{w})$  is nonlinear, and therefore we can no longer calculate the posterior in closed form.
- Use Laplace approximation
  - ① Find a (local) maximum  $\mathbf{w}_{MAP}$  of the posterior via numerical optimisation.
  - ② Evaluate the matrix of second derivatives of the negative log posterior distribution.
- Find a (local) maximum using the log-posterior

$$\ln p(\mathbf{w} | \mathcal{D}, \alpha, \beta) = -\frac{\alpha}{2} \mathbf{w}^T \mathbf{w} - \frac{\beta}{2} \sum_{n=1}^N (y(\mathbf{x}, \mathbf{w}) - t_n)^2 + \text{const}$$

- Find the matrix of second derivatives of the negative log posterior distribution

$$\mathbf{A} = -\nabla \nabla \ln p(\mathbf{w} | \mathcal{D}, \alpha, \beta) = \alpha \mathbf{I} + \beta \mathbf{H}$$

where  $\mathbf{H}$  is the Hessian matrix of the sum-of-squares error function with respect to the components of  $\mathbf{w}$ .



Error Backpropagation

Regularisation in Neural  
NetworksBayesian Neural  
Networks

# Bayesian Neural Networks

- Having  $\mathbf{w}_{MAP}$ , and  $\mathbf{A}$ , we can approximate the posterior by a Gaussian

$$q(\mathbf{w} \mid \mathcal{D}, \alpha, \beta) = \mathcal{N}(\mathbf{w} \mid \mathbf{w}_{MAP}, \mathbf{A}^{-1})$$

- Similarly for the predictive distribution (without proof)

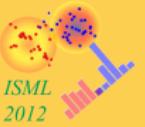
$$p(t \mid \mathbf{x}, \mathcal{D}, \alpha, \beta) = \mathcal{N}(t \mid y(\mathbf{x}, \mathbf{w}_{MAP}), \sigma^2(\mathbf{x}))$$

where

$$\sigma^2(\mathbf{x}) = \beta^{-1} + \mathbf{g}^T \mathbf{A}^{-1} \mathbf{g}$$

and

$$\mathbf{g} = \nabla_{\mathbf{w}} y(\mathbf{x}, \mathbf{w}) \Big|_{\mathbf{w}=\mathbf{w}_{MAP}}.$$



## Part XI

### *Kernel Methods*

*Nonparametric  
Probability Density  
Estimation*

*The Role of Training  
Data*

*Dual Representations*

*Kernels*

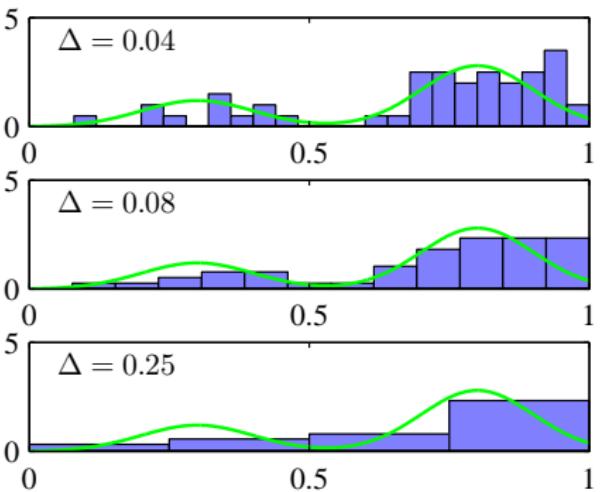
*Lagrange Multipliers*

Nonparametric  
Probability Density  
EstimationThe Role of Training  
Data

Dual Representations

Kernels

Lagrange Multipliers



Histogram of 50 data points generated from the distribution shown by the green curve for varying common bin width  $\Delta$



# Nonparametric Density Estimation – Histogram

## Advantages

- Data can be discarded after calculating the  $p_i$ .
- Algorithm can be applied to sequentially arriving data.

## Disadvantages

- Dependency on bin width  $\Delta_i$ .
- Discontinuities due to the bin edges.
- Exponential scaling with the dimensionality  $D$  of the data.  
Need  $M^D$  bins for  $D$  dimensions and  $M$  bins per dimension.

Nonparametric  
Probability Density  
Estimation

The Role of Training  
Data

Dual Representations

Kernels

Lagrange Multipliers

Nonparametric  
Probability Density  
EstimationThe Role of Training  
Data

Dual Representations

Kernels

Lagrange Multipliers

# Nonparametric Density Estimation - Refined

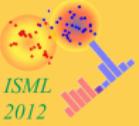
- Draw data from some unknown probability distribution  $p(\mathbf{x})$  in a  $D$ -dimensional space.
- Consider a small region  $\mathcal{R}$  containing  $\mathbf{x}$ . Probability mass associated with this region

$$P = \int_{\mathcal{R}} p(\mathbf{x}) \, d\mathbf{x}$$

- Data set of  $N$  observations drawn from  $p(\mathbf{x})$ . Total number  $K$  of points found inside of  $\mathcal{R}$  is distributed according to the binomial distribution

$$\text{Bin}(K | N, P) = \frac{N!}{K!(N-K)!} P^K (1-P)^{N-K}$$

- Expectation of  $K$  :  $\mathbb{E}[K/N] = P$
- Variance of  $K$  :  $\text{var}[K/N] = P(1-P)$

Nonparametric  
Probability Density  
EstimationThe Role of Training  
Data

Dual Representations

Kernels

Lagrange Multipliers

# Nonparametric Density Estimation - Refined

- Expectation of  $K$  :  $\mathbb{E}[K/N] = P$
- Variance of  $K$  :  $\text{var}[K/N] = P(1 - P)$
- For large  $N$ , the distribution will be sharply peaked and therefore

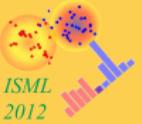
$$K \approx NP$$

- Assuming also that the region has volume  $V$  and the region is small enough for  $p(\mathbf{x})$  to be roughly constant, then

$$P \approx p(\mathbf{x})V$$

- Combining two contradictory assumptions
  - Region  $\mathcal{R}$  is small enough for  $p(\mathbf{x})$  to be roughly constant.
  - Region  $\mathcal{R}$  is large enough to have enough  $K$  points falling into it to get a sharp peak for the binomial distribution.

$$p(\mathbf{x}) \approx \frac{K}{NV}$$



- Two ways to exploit

$$p(\mathbf{x}) \approx \frac{K}{NV}$$

- ① Fix  $V$  and determine  $K$  from the data :  
**kernel density estimation**
- ② Fix  $K$  and determine the volume  $V$  from the data :  
 **$K$ -nearest-neighbours density estimation**

Nonparametric  
Probability Density  
Estimation

The Role of Training  
Data

Dual Representations

Kernels

Lagrange Multipliers

Nonparametric  
Probability Density  
EstimationThe Role of Training  
Data

Dual Representations

Kernels

Lagrange Multipliers

# Nonparametric Estimation – Parzen Estimator

- Define region  $\mathcal{R}$  to be a small hypercube around  $\mathbf{x}$
- Define **Parzen window (kernel function)**

$$k(\mathbf{u}) = \begin{cases} 1, & |u_i| \leq 1/2, \\ 0, & \text{otherwise} \end{cases} \quad i = 1, \dots, D$$

- Total number of data points inside of the hypercube centered at  $\mathbf{x}$

$$K = \sum_{n=1}^N k\left(\frac{\mathbf{x} - \mathbf{x}_n}{h}\right)$$

- Density estimate for  $p(\mathbf{x})$

$$p(\mathbf{x}) \approx \frac{K}{NV} = \frac{1}{N} \sum_{n=1}^N \frac{1}{h^D} k\left(\frac{\mathbf{x} - \mathbf{x}_n}{h}\right)$$

- Interpret as sum over  $N$  cubes centered at each of the  $\mathbf{x}_n$ .

# Nonparametric Estimation – Parzen Estimator



- Remaining problem: Discontinuities because of the hypercube (either **in** or **out**).
- Choose a smoother kernel function (and normalise correctly).
- Common choice : Gaussian kernel

$$p(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^N \frac{1}{(2\pi h^2)^{D/2}} \exp \left\{ -\frac{\|\mathbf{x} - \mathbf{x}_n\|}{2h^2} \right\}$$

- Can choose any other kernel function  $k(\mathbf{u})$  obeying

$$k(\mathbf{u}) \geq 0,$$

$$\int k(\mathbf{u}) \, d\mathbf{u} = 1$$

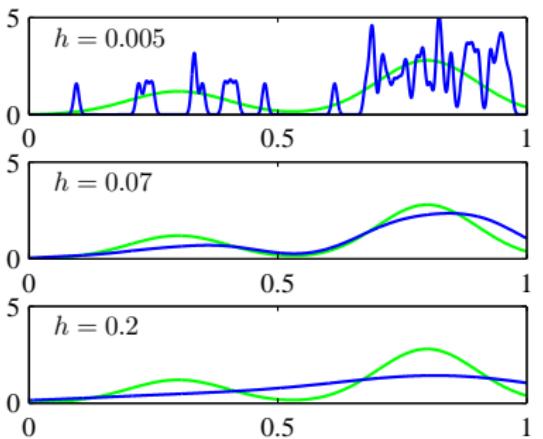
Nonparametric  
Probability Density  
Estimation

The Role of Training  
Data

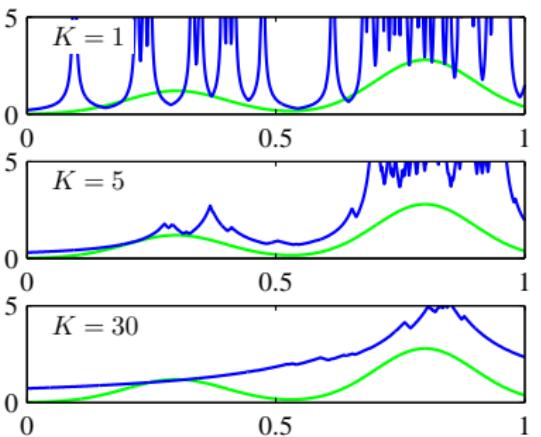
Dual Representations

Kernels

Lagrange Multipliers



Kernel density model with Gaussian kernel for different  $h$ .



Nearest neighbour density model for different  $K$ .

# The Role of Training Data



- Parametric methods

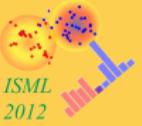
- ➊ Learn the model parameter  $w$  from the training data  $t$ .
- ➋ Discard the training data  $t$ .

- Nonparametric methods : Use training data directly for prediction.

- $k$ -nearest neighbours : use  $k$ -closest data from the 'training' set for classification
- Parzen probability density model : set of functions centered on the training data

- Kernel methods

- Base prediction on linear combination of kernel functions evaluated at the training data.

Nonparametric  
Probability Density  
EstimationThe Role of Training  
Data

Dual Representations

Kernels

Lagrange Multipliers

# Dual Representations

- Consider a linear regression model with regularised sum-of-squares error

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

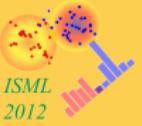
where  $\lambda \geq 0$ .

- We could also write this in more compact form as

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

with the target vector  $\mathbf{t} = (t_1, \dots, t_N)^T$ , and the design matrix

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

Nonparametric  
Probability Density  
EstimationThe Role of Training  
Data

Dual Representations

Kernels

Lagrange Multipliers

# Dual Representations

- Critical points for  $J(\mathbf{w})$

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

can be found as

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

by introducing the new vector  $\mathbf{a} = (a_1, \dots, a_N)^T$  with components

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

Nonparametric  
Probability Density  
EstimationThe Role of Training  
Data

Dual Representations

Kernels

Lagrange Multipliers

# Dual Representations

- Now express  $J(\mathbf{w})$  as a function of this new variable  $\mathbf{a}$  instead of  $\mathbf{w}$  via the relation  $\mathbf{w} = \Phi^T \mathbf{a}$

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

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

- Define the  $N \times N$  Gram matrix  $\mathbf{K} = \Phi \Phi^T$  with elements

$$K_{nm} = \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m).$$

- Express  $J(\mathbf{a})$  now as

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



# The Kernel Function

- The kernel function is defined over two points,  $\mathbf{x}$  and  $\mathbf{x}'$ , of the input space

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}').$$

- $k(\mathbf{x}, \mathbf{x}')$  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.)

Nonparametric  
Probability Density  
Estimation

The Role of Training  
Data

Dual Representations

Kernels

Lagrange Multipliers

Nonparametric  
Probability Density  
EstimationThe Role of Training  
Data

Dual Representations

Kernels

Lagrange Multipliers

# Critical Points of $J(\mathbf{a})$

- Let's calculate the critical points for

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

- Directional derivative

$$\mathcal{D}J(\mathbf{a})(\boldsymbol{\xi}) = \boldsymbol{\xi}^T \mathbf{K} \mathbf{K} \mathbf{a} - \boldsymbol{\xi}^T \mathbf{K} \mathbf{t} + \lambda \boldsymbol{\xi}^T \mathbf{K} \mathbf{a}$$

should be zero in all possible directions  $\boldsymbol{\xi}$ .

- Therefore  $\mathbf{K}(\mathbf{K} \mathbf{a} - \mathbf{t} + \lambda \mathbf{a}) = 0$  and as  $\mathbf{K}$  has full rank

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

- Second directional derivative (using  $\mathbf{K} = \Phi \Phi^T$ )

$$\mathcal{D}^2 J(\mathbf{a})(\boldsymbol{\xi}, \boldsymbol{\xi}) = \boldsymbol{\xi}^T \mathbf{K} \mathbf{K} \boldsymbol{\xi} + \lambda \boldsymbol{\xi}^T \mathbf{K} \boldsymbol{\xi} = \|\mathbf{K} \boldsymbol{\xi}\|^2 + \lambda \|\Phi^T \boldsymbol{\xi}\| > 0.$$

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

# Prediction for the Linear Regression Model



- Inserting the argument  $\mathbf{a}$  which minimises the error  $J(\mathbf{a})$  into the prediction model for the linear regression, we get for the prediction

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

where we defined the vector  $\mathbf{k}(\mathbf{x})$  with elements

$$k_n(\mathbf{x}) = k(\mathbf{x}_n, \mathbf{x}) = \phi(\mathbf{x}_n)^T \phi(\mathbf{x}).$$

- The prediction  $y(\mathbf{x})$  can be expressed entirely in terms of the kernel function  $k(\mathbf{x}, \mathbf{x}')$  evaluated at the training and test data.
- Looks familiar? See Bayesian Linear Regression.

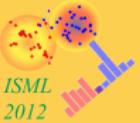
Nonparametric  
Probability Density  
Estimation

The Role of Training  
Data

Dual Representations

Kernels

Lagrange Multipliers



Nonparametric  
Probability Density  
Estimation

The Role of Training  
Data

Dual Representations

Kernels

Lagrange Multipliers

# Dual Representation

- What have we gained by the dual representation?
- Need to invert an  $N \times N$  matrix now, where  $N$  is the number of data points. Can be large!
- In the parameter space formulation, we 'only' needed to invert an  $M \times M$  matrix, where  $M$  was the number of basis functions.
- BUT : a kernel corresponds to an inner product of basis functions. So we can use a large number of basis functions, even infinitely many.
- We can construct new valid kernels directly from given ones (whatever the corresponding basis functions of the new kernel might be).
- As a kernel defines a kind of 'distance' between two points in the input space, we can define kernels over graphs, sets, strings, and text documents.

# Kernels from Basis Functions

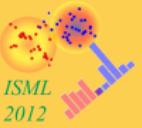


- 1 Choose a set of basis functions

$$\{\phi_1, \dots, \phi_M\}$$

- 2 Find a new kernel as an inner product between vectors of basis functions evaluated at  $x$  and  $x'$

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

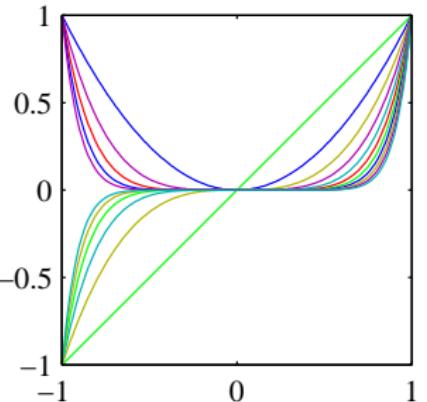
Nonparametric  
Probability Density  
EstimationThe Role of Training  
Data

Dual Representations

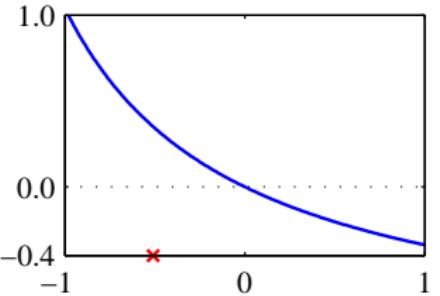
Kernels

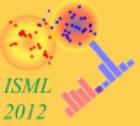
Lagrange Multipliers

## Polynomial basis functions



Corresponding kernel  
 $k(x, x')$  as function of  $x$  for  
 $x' = -0.5$  (red cross).



Nonparametric  
Probability Density  
EstimationThe Role of Training  
Data

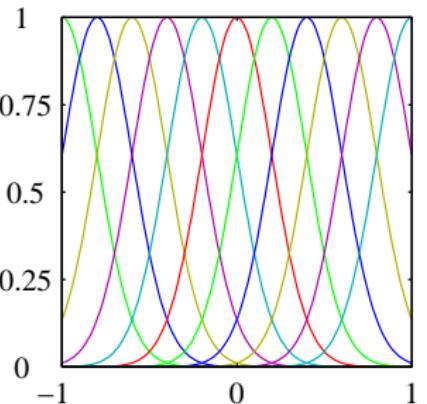
Dual Representations

Kernels

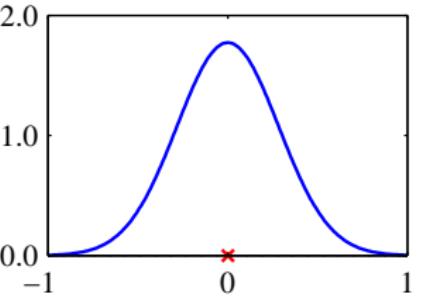
Lagrange Multipliers

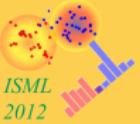
# Kernels from Basis Functions

## Gaussian basis functions



Corresponding kernel  
 $k(x, x')$  as function of  $x$  for  
 $x' = 0.0$  (red cross).





Nonparametric  
Probability Density  
Estimation

The Role of Training  
Data

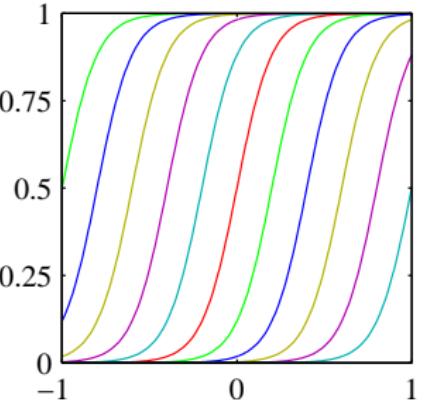
Dual Representations

Kernels

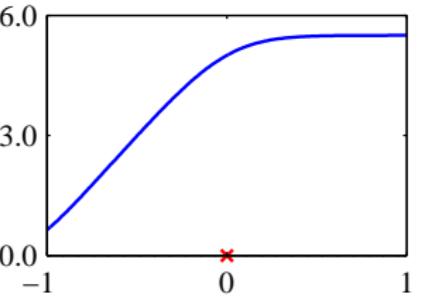
Lagrange Multipliers

# Kernels from Basis Functions

Logistic Sigmoid  
basis functions



Corresponding kernel  
 $k(x, x')$  as function of  $x$  for  
 $x' = 0.0$  (red cross).



# Kernels by Guessing a Kernel Function



- 1 Choose a mapping from two points of the input space to a real number, which is symmetric in its arguments, e.g.

$$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z})^2 = k(\mathbf{z}, \mathbf{x})$$

- 2 Try to write this as an inner product of a vector valued function evaluated at the arguments  $\mathbf{x}$  and  $\mathbf{z}$ , e.g.

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

with the feature mapping  $\phi(\mathbf{x}) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)^T$ .

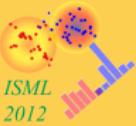
Nonparametric  
Probability Density  
Estimation

The Role of Training  
Data

Dual Representations

Kernels

Lagrange Multipliers

Nonparametric  
Probability Density  
EstimationThe Role of Training  
Data

Dual Representations

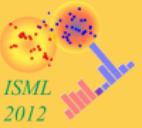
Kernels

Lagrange Multipliers

# New Kernels From Theory

- 1 A necessary and sufficient condition for  $k(\mathbf{x}, \mathbf{x}')$  to be a valid kernel is that the Gram matrix  $\mathbf{K}$ , whose elements are  $k(\mathbf{x}_n, \mathbf{x}_m)$ , should be positive semidefinite for all possible choices of the set  $\{\mathbf{x}_n\}$ .

Note: The Gram matrix  $\mathbf{K}$  was defined with the help of the input data  $\mathbf{K} = \Phi \Phi^T$ . The kernel function  $k(\mathbf{x}_n, \mathbf{x}_m)$  defines the entries in the Gram matrix  $K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m)$  depending on two input data points  $\mathbf{x}_n$  and  $\mathbf{x}_m$ . The above therefore says, that  $k(\mathbf{x}, \mathbf{x}')$  is a valid kernel if the Gram matrix is positive semidefinite for any set of input data.

Nonparametric  
Probability Density  
EstimationThe Role of Training  
Data

Dual Representations

Kernels

Lagrange Multipliers

# New Kernels From Other Kernels

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

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

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

 $c > 0$  constant

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

 $f(\cdot)$  any function

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

 $q(\cdot)$  polynomial with  
nonneg. coeff.

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

 $\phi(\mathbf{x})$  any function to  $\mathbb{R}^M$ 

$$k(\mathbf{x}, \mathbf{x}') = k_3(\phi(\mathbf{x}), \phi(\mathbf{x}'))$$

 $k_3(\cdot, \cdot)$  valid kernel in  $\mathbb{R}^M$ 

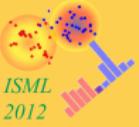
$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{A} \mathbf{x}'$$

 $\mathbf{A} = \mathbf{A}^T, \mathbf{A} \geq 0$ 

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

 $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$ 

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

Nonparametric  
Probability Density  
EstimationThe Role of Training  
Data

Dual Representations

Kernels

Lagrange Multipliers

# New Kernels From Other Kernels

## Further examples of kernels

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}')^M \quad \text{only terms of degree } M$$

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + c)^M \quad \text{all terms up to degree } M$$

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2 / 2\sigma^2) \quad \text{Gaussian kernel}$$

$$k(\mathbf{x}, \mathbf{x}') = \tanh(a \mathbf{x}^T \mathbf{x}' + b) \quad \text{Sigmoidal kernel}$$

Generally, we call

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}' \quad \text{linear kernel}$$

$$k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}') \quad \text{stationary kernel}$$

$$k(\mathbf{x}, \mathbf{x}') = (\|\mathbf{x} - \mathbf{x}'\|) \quad \text{homogeneous kernel}$$

# Kernels over Graphs, Sets, Strings, Texts



- 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.

Nonparametric  
Probability Density  
Estimation

The Role of Training  
Data

Dual Representations

Kernels

Lagrange Multipliers

Nonparametric  
Probability Density  
EstimationThe Role of Training  
Data

Dual Representations

Kernels

Lagrange Multipliers

# 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  $L$ .

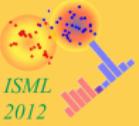
Nonparametric  
Probability Density  
EstimationThe Role of Training  
Data

Dual Representations

Kernels

Lagrange Multipliers

- Find the stationary points for a function  $f(x_1, x_2)$  subject to one or more constraints on the variables  $x_1$  and  $x_2$  written in the form  $g(x_1, x_2) = 0$ .
- Direct approach
  - ➊ Solve  $g(x_1, x_2) = 0$  for one of the variables to get  $x_2 = h(x_1)$ .
  - ➋ Insert the result into  $f(x_1, x_2)$  to get a function of one variable  $f(x_1, h(x_1))$ .
  - ➌ Find the stationary point(s)  $x_1^*$  of  $f(x_1, h(x_1))$  with corresponding value  $x_2^* = h(x_1^*)$ .
- Finding  $x_2 = h(x_1)$  may be hard.
- Symmetry in the variables  $x_1$  and  $x_2$  is lost.

Nonparametric  
Probability Density  
EstimationThe Role of Training  
Data

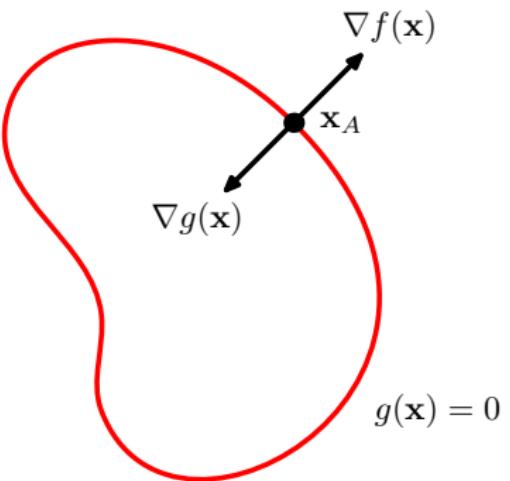
Dual Representations

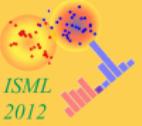
Kernels

Lagrange Multipliers

# Lagrange Multipliers

- Assume  $D$ -dimensional variable  $\mathbf{x} = (x_1, \dots, x_D)^T$ .
- The constraint  $g(\mathbf{x}) = 0$  is a  $(D - 1)$ -dimensional surface in the  $\mathbf{x}$ -space.
- The gradient  $\nabla g(\mathbf{x})$  will be orthogonal to the surface because if both  $g(\mathbf{x} + \epsilon)$  and  $g(\mathbf{x})$  lie on the surface, then  $g(\mathbf{x} + \epsilon) \simeq g(\mathbf{x}) + \epsilon^T \nabla g(\mathbf{x})$ .



Nonparametric  
Probability Density  
EstimationThe Role of Training  
Data

Dual Representations

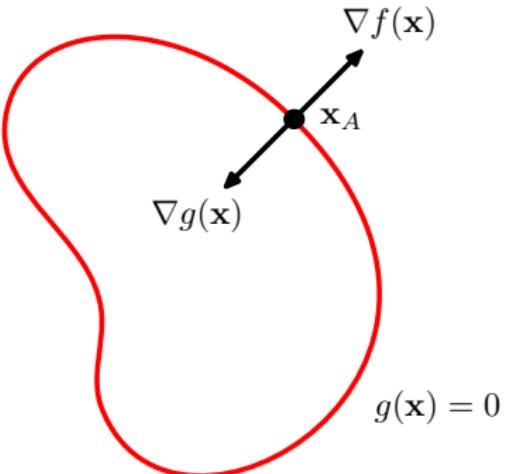
Kernels

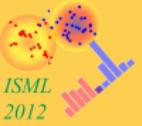
Lagrange Multipliers

# Lagrange Multipliers

- Assume  $\mathbf{x}^*$  maximises  $f(\mathbf{x})$ . Then  $\nabla f(\mathbf{x}^*)$  will be orthogonal to the surface. Otherwise we could increase the value by  $f(\mathbf{x}^* + \epsilon) \simeq f(\mathbf{x}^*) + \epsilon^T \nabla f(\mathbf{x}^*)$ .
- Thus  $\nabla f(\mathbf{x}^*)$  and  $\nabla g(\mathbf{x})$  must be parallel (or anti-parallel) and therefore at  $\mathbf{x} = \mathbf{x}^*$  we have with the Lagrange multiplier  $\lambda \neq 0$ ,

$$\nabla f(\mathbf{x}) + \lambda \nabla g(\mathbf{x}) = 0.$$



Nonparametric  
Probability Density  
EstimationThe Role of Training  
Data

Dual Representations

Kernels

Lagrange Multipliers

# Lagrange Multipliers

- Introduce the **Lagrangian** function

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

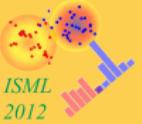
from which we get the constraint stationary conditions

$$\nabla_{\mathbf{x}} L(\mathbf{x}, \lambda) = \nabla f(\mathbf{x}) + \lambda \nabla g(\mathbf{x}) = 0$$

and the constraint itself

$$\frac{\partial L(\mathbf{x}, \lambda)}{\partial \lambda} = g(\mathbf{x}) = 0.$$

- This are  $D$  equations resulting from  $\nabla_{\mathbf{x}} L(\mathbf{x}, \lambda)$  and one equation from  $\frac{\partial L(\mathbf{x}, \lambda)}{\partial \lambda}$ , together determining  $\mathbf{x}^*$  and  $\lambda$ .

Nonparametric  
Probability Density  
EstimationThe Role of Training  
Data

Dual Representations

Kernels

Lagrange Multipliers

# Lagrange Multipliers - Example

- Given  $f(x_1, x_2) = 1 - x_1^2 - x_2^2$  subject to the constraint  $g(x_1, x_2) = x_1 + x_2 - 1 = 0$ .
- Define the Lagrangian function

$$L(\mathbf{x}, \lambda) = 1 - x_1^2 - x_2^2 + \lambda(x_1 + x_2 - 1).$$

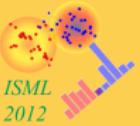
- A stationary solution with respect to  $x_1$ ,  $x_2$ , and  $\lambda$  must satisfy

$$-2x_1 + \lambda = 0$$

$$-2x_2 + \lambda = 0$$

$$x_1 + x_2 - 1 = 0.$$

- Therefore  $(x_1^*, x_2^*) = (\frac{1}{2}, \frac{1}{2})$  and  $\lambda = 1$ .

Nonparametric  
Probability Density  
EstimationThe Role of Training  
Data

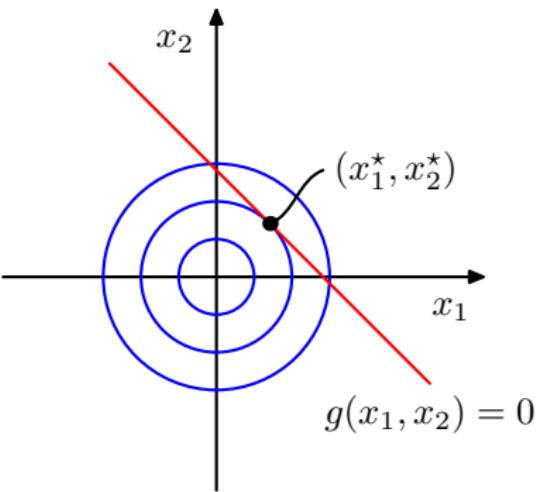
Dual Representations

Kernels

Lagrange Multipliers

# Lagrange Multipliers - Example

- Given  $f(x_1, x_2) = 1 - x_1^2 - x_2^2$  subject to the constraint  $g(x_1, x_2) = x_1 + x_2 - 1 = 0$ .
- Lagrangian  $L(\mathbf{x}, \lambda) = 1 - x_1^2 - x_2^2 + \lambda(x_1 + x_2 - 1)$
- $(x_1^*, x_2^*) = (\frac{1}{2}, \frac{1}{2})$ .



$$g(x_1, x_2) = 0$$

Nonparametric  
Probability Density  
EstimationThe Role of Training  
Data

Dual Representations

Kernels

Lagrange Multipliers

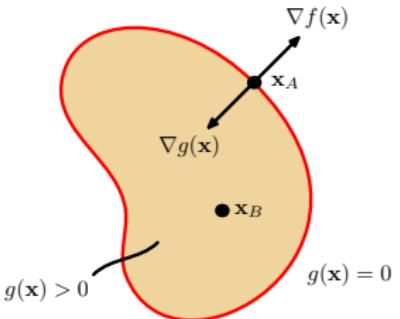
# Lagrange Multipliers for Inequality Constraints

- Inequality constraint  $g(\mathbf{x}) \geq 0$ .

- Two cases

- If  $g(\mathbf{x}) > 0$ , constraint is **inactive**. Constraint plays no role. Solution is  $\nabla f(\mathbf{x}) = 0$ . Corresponds to Lagrangian with  $\lambda = 0$ .
- If  $g(\mathbf{x}) = 0$ , constraint is **active**. Solution lies on the boundary, but now the sign of  $\lambda$  is crucial. Only a maximum if its gradient is oriented away from the region  $g(\mathbf{x}) > 0$ . Therefore,  $\nabla f(\mathbf{x}) = -\lambda \nabla g(\mathbf{x})$  for some  $\lambda > 0$ .

- For either of the cases  $\lambda g(\mathbf{x}) = 0$ .





Nonparametric  
Probability Density  
Estimation

The Role of Training  
Data

Dual Representations  
Kernels

Lagrange Multipliers

# Lagrange Multipliers for Inequality Constraints

- Maximise  $f(\mathbf{x})$  subject to the constraint  $g(\mathbf{x}) \geq 0$ .
- Define the Lagrangian

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

- Solve for  $\mathbf{x}$  and  $\lambda$  subject to the constraints  
([Karush-Kuhn-Tucker](#) or KKT conditions )

$$g(\mathbf{x}) \geq 0$$

$$\lambda \geq 0$$

$$\lambda g(\mathbf{x}) = 0$$

Nonparametric  
Probability Density  
EstimationThe Role of Training  
Data

Dual Representations

Kernels

Lagrange Multipliers

# Lagrange Multipliers for General Case

- Maximise  $f(\mathbf{x})$  subject to the constraints  $g_j(\mathbf{x}) = 0$  for  $j = 1, \dots, J$ , and  $h_k(\mathbf{x}) \geq 0$  for  $k = 1, \dots, K$ .
- Define the Lagrange multipliers  $\{\lambda_j\}$  and  $\{\mu_k\}$ , and the Lagrangian

$$L(\mathbf{x}, \{\lambda_j\}, \{\mu_k\}) = f(\mathbf{x}) + \sum_{j=1}^J \lambda_j g_j(\mathbf{x}) + \sum_{k=1}^K \mu_k h_k(\mathbf{x}).$$

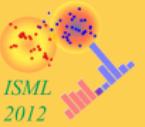
- Solve for  $\mathbf{x}$ ,  $\{\lambda_j\}$ , and  $\{\mu_k\}$  subject to the constraints (Karush-Kuhn-Tucker or KKT conditions )

$$\mu_k \geq 0$$

$$\mu_k h_k(\mathbf{x}) = 0$$

for  $k = 1, \dots, K$ .

- For minimisation of  $f(\mathbf{x})$ , change the sign in front of the Lagrange multipliers.



*Sparse Kernel Machines*

*Maximum Margin  
Classifiers*

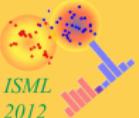
*Overlapping Class  
Distributions*

*Relevance Vector  
Machines - Overview*

## Part XII

### *Sparse Kernel Machines*

# Sparse Kernel Machines - Motivation



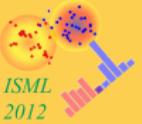
- Nonlinear kernels extended our toolbox of methods considerably. Wherever an inner product was used in an algorithms, we can replace it with a kernel.
- Kernels act as a kind of 'similarity' measure and can be defined over graphs, sets, strings, and documents.
- But the kernel matrix is a square matrix with dimensions equal to the number of data points  $N$ . In order to calculate it, the kernel function must be evaluated for all pairs of training inputs.
- **Sparse Kernel Machines** implement learning algorithms which are based on sparse kernels. For prediction, these kernels are only evaluated at a subset of the training data.

Sparse Kernel Machines

Maximum Margin  
Classifiers

Overlapping Class  
Distributions

Relevance Vector  
Machines - Overview



# Maximum Margin Classifiers

- Return to the two-class classification with a linear model

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

where  $\phi(\mathbf{x})$  is some fixed feature space mapping, and  $b$  is the bias.

- 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 there exists a linear decision boundary. That means, there exist  $\mathbf{w}$  and  $b$  such that

$$t_n \text{ sign}(y(\mathbf{x}_n)) > 0 \quad n = 1, \dots, N.$$

Sparse Kernel Machines

Maximum Margin  
ClassifiersOverlapping Class  
DistributionsRelevance Vector  
Machines - Overview

# Maximum Margin Classifiers

- There may exist many solutions  $w$  and  $b$  for which the linear classifier perfectly separates the two classes.
- 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)?



©2012  
Christfried Webers  
NICTA

The Australian National  
University

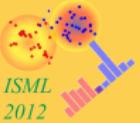


Sparse Kernel Machines

Maximum Margin  
Classifiers

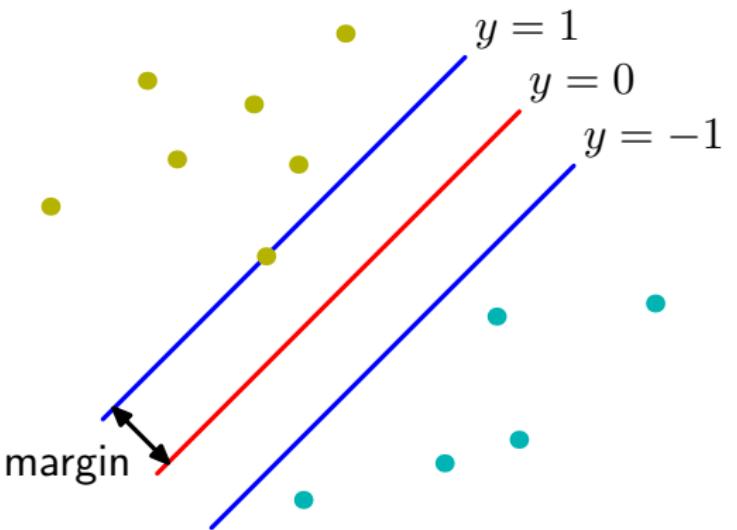
Overlapping Class  
Distributions

Relevance Vector  
Machines - Overview



# Maximum Margin Classifiers

- The margin is the smallest distance between the decision boundary and any of the samples. (We will see later why  $y = \pm 1$  on the margin.)



Sparse Kernel Machines

Maximum Margin  
ClassifiersOverlapping Class  
DistributionsRelevance Vector  
Machines - Overview

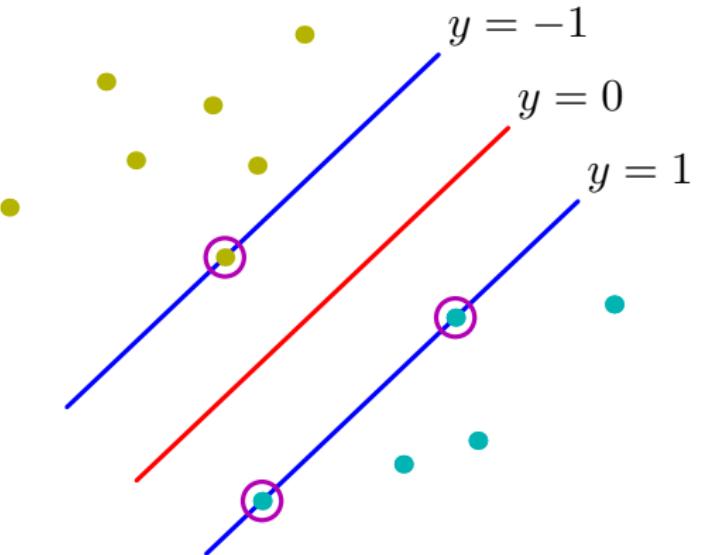


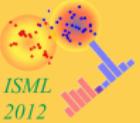
Sparse Kernel Machines

Maximum Margin  
ClassifiersOverlapping Class  
DistributionsRelevance Vector  
Machines - Overview

# Maximum Margin Classifiers

- Support Vector Machines choose the decision boundary which maximises the margin.



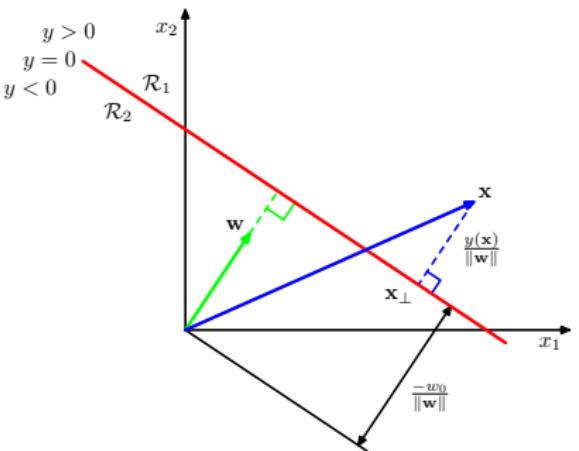


Sparse Kernel Machines

Maximum Margin  
ClassifiersOverlapping Class  
DistributionsRelevance Vector  
Machines - Overview

# Reminder - Linear Classification

- $\mathbf{w}$  is orthogonal to the decision boundary ( $0 = \mathbf{w}^T \mathbf{x}_1 + w_0 = \mathbf{w}^T \mathbf{x}_2 + w_0$  and  $\mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2) = 0.$ )
- Length of the projection of  $\mathbf{x}$  onto  $\mathbf{w}$  is  $\frac{\mathbf{w}^T \mathbf{x}}{\|\mathbf{w}\|}.$
- If  $\mathbf{x}$  sits on the decision boundary then  $0 = y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$  and therefore the distance from the origin orthogonal to the decision boundary (and parallel to  $\mathbf{w}$ ) is  $\frac{\mathbf{w}^T \mathbf{x}}{\|\mathbf{w}\|} = -\frac{w_0}{\|\mathbf{w}\|}.$
- The distance of  $\mathbf{x}$  from the decision boundary is therefore  $\frac{\mathbf{w}^T \mathbf{x}}{\|\mathbf{w}\|} - \left(-\frac{w_0}{\|\mathbf{w}\|}\right) = \frac{y(\mathbf{x})}{\|\mathbf{w}\|}.$





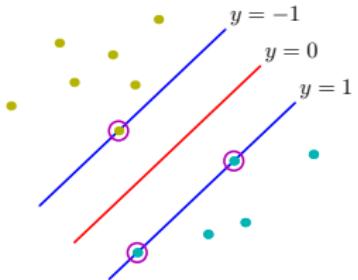
Sparse Kernel Machines

Maximum Margin  
ClassifiersOverlapping Class  
DistributionsRelevance Vector  
Machines - Overview

# Maximum Margin Classifiers

- Support Vector Machines choose the decision boundary which maximises the smallest distance to samples in both classes.
- We calculated the distance of a point  $\mathbf{x}$  from the hyperplane  $y(\mathbf{x}) = 0$  as  $|y(\mathbf{x})|/\|\mathbf{w}\|$ .
- Perfect classification means  $t_n y(\mathbf{x}_n) > 0$  for all  $n$ .
- Thus, the distance of  $\mathbf{x}_n$  from the decision boundary is

$$\frac{t_n y(\mathbf{x}_n)}{\|\mathbf{w}\|} = \frac{t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b)}{\|\mathbf{w}\|}$$





- Support Vector Machines choose the decision boundary which maximises the smallest distance to samples in both classes.
- For the maximum margin solution, solve

$$\arg \max_{\mathbf{w}, b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n [t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b)] \right\}.$$

- How can we solve this?

Sparse Kernel Machines

Maximum Margin  
ClassifiersOverlapping Class  
DistributionsRelevance Vector  
Machines - Overview



Sparse Kernel Machines

Maximum Margin  
ClassifiersOverlapping Class  
DistributionsRelevance Vector  
Machines - Overview

# Maximum Margin Classifiers

- Maximum margin solution

$$\arg \max_{\mathbf{w}, b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n [t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b)] \right\}$$

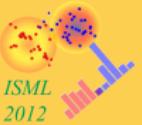
- Observation: Rescaling  $\mathbf{w} \rightarrow \kappa \mathbf{w}$  and  $b \rightarrow \kappa b$  does not change the distance from the hypersurface.
- Scale in such a way that for the closest point

$$t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) = 1.$$

- The canonical representation for the decision hyperplane is therefore

$$t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1 \quad n = 1, \dots, N.$$

- The constraints are active, if the equality holds, otherwise they are inactive.



Sparse Kernel Machines

Maximum Margin  
ClassifiersOverlapping Class  
DistributionsRelevance Vector  
Machines - Overview

# Maximum Margin Classifiers

- Maximum margin solution

$$\arg \max_{\mathbf{w}, b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n [t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b)] \right\}$$

- Transformed once

$$\arg \max_{\mathbf{w}, b} \left\{ \frac{1}{\|\mathbf{w}\|} \right\} \quad \text{s.t. } t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1.$$

- Transformed again

$$\arg \min_{\mathbf{w}, b} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 \right\} \quad \text{s.t. } t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1.$$



- Quadratic Programming (QP) problem: minimise a quadratic function subject to linear constraints.

$$\arg \min_{\mathbf{w}, b} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 \right\} \quad \text{s.t. } t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1.$$

- Introduce Lagrange multipliers  $\{a_n \geq 0\}$  to get the Lagrangian

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N a_n \{t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) - 1\}.$$

Sparse Kernel Machines

Maximum Margin  
ClassifiersOverlapping Class  
DistributionsRelevance Vector  
Machines - Overview



# Maximum Margin Classifiers

- Lagrangian

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N a_n \left\{ t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) - 1 \right\}.$$

- Derivatives with respect to  $\mathbf{w}$  and  $b$  are

$$\mathbf{w} = \sum_{n=1}^N a_n t_n \phi(\mathbf{x}_n) \quad 0 = \sum_{n=1}^N a_n t_n$$

- Dual representation (again QP problem): Maximise

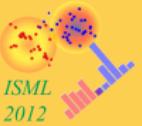
$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

$$\sum_{n=1}^N a_n t_n = 0$$

$$a_n \geq 0 \quad n = 1, \dots, N.$$

Sparse Kernel Machines

Maximum Margin  
ClassifiersOverlapping Class  
DistributionsRelevance Vector  
Machines - Overview



# Maximum Margin Classifiers

- Kernel function

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$$

must imply a positive definite kernel (Gram matrix), otherwise the optimisation problem is not bounded from above.

- Solving the optimisation problem is of the order  $O(N^3)$  in the number of data points  $N$ . The original optimisation problem is of order  $O(M^3)$  where  $M$  was the number of basis functions.
- Again, kernel allows us to use large or infinite number of basis functions (feature maps), but disadvantageous for  $M < N$ .

Sparse Kernel Machines

Maximum Margin  
ClassifiersOverlapping Class  
DistributionsRelevance Vector  
Machines - Overview



# Maximum Margin Classifiers - Prediction

$$y(\mathbf{x}) = \sum_{n=1}^N a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b$$

- The Karush-Kuhn-Tucker (KKT) conditions are

$$a_n \geq 0$$

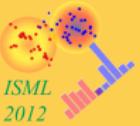
$$t_n y(\mathbf{x}_n) - 1 \geq 0$$

$$a_n \{t_n y(\mathbf{x}_n) - 1\} = 0.$$

- Therefore, either  $a_n = 0$  or  $t_n y(\mathbf{x}_n) - 1 = 0$ .
- If  $a_n = 0$ , no contribution to the prediction!
- After training, use only the set  $\mathcal{S}$  of points for which  $a_n > 0$  (and therefore  $t_n y(\mathbf{x}_n) = 1$ ) holds.
- $\mathcal{S}$  contains only the support vectors.

Sparse Kernel Machines

Maximum Margin  
ClassifiersOverlapping Class  
DistributionsRelevance Vector  
Machines - Overview

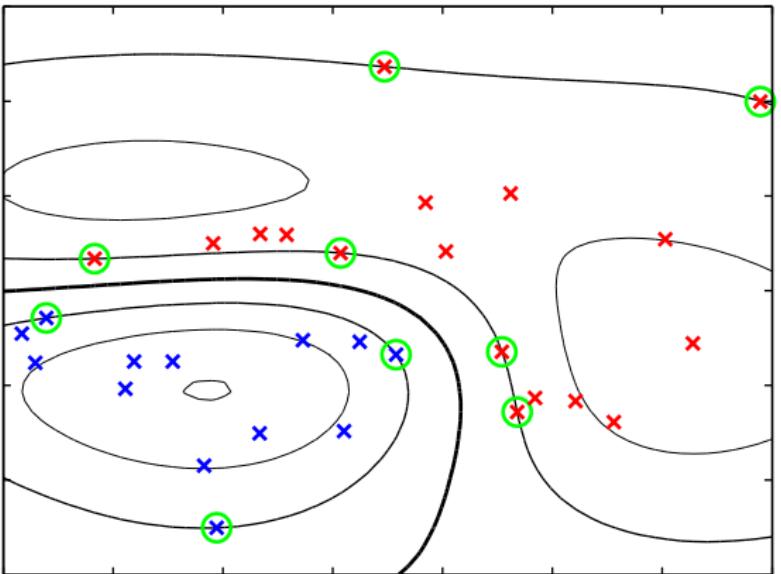


Sparse Kernel Machines

Maximum Margin  
Classifiers

Overlapping Class  
Distributions

Relevance Vector  
Machines - Overview



Decision and margin boundaries for a two-class problem using  
Gaussian kernel functions.



Sparse Kernel Machines

Maximum Margin  
ClassifiersOverlapping Class  
DistributionsRelevance Vector  
Machines - Overview

# Maximum Margin Classifiers - Support Vectors

- Now for the bias  $b$ .
- Observe that for the support vectors,  $t_n y(\mathbf{x}_n) = 1$ .
- Therefore, we find

$$t_n \left( \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) + b \right) = 1$$

and can use any support vector to calculate  $b$ .

- Numerically more stable: Multiply by  $t_n$ , observe  $t_n^2 = 1$ , and average over all support vectors.

$$b = \frac{1}{N_S} \sum_{n \in \mathcal{S}} \left( t_n - \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \right)$$



- Express the error for maximum margin classifiers with the help of a function

$$E_\infty(z) = \begin{cases} 0, & z \geq 0 \\ \infty, & z < 0 \end{cases}$$

enforcing the constraints as

$$\sum_{n=1}^N E_\infty(y(\mathbf{x}_n)t_n - 1)) + \lambda \|\mathbf{w}\|^2.$$

Sparse Kernel Machines

Maximum Margin  
ClassifiersOverlapping Class  
DistributionsRelevance Vector  
Machines - Overview



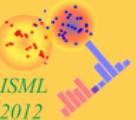
Sparse Kernel Machines

Maximum Margin  
ClassifiersOverlapping Class  
DistributionsRelevance Vector  
Machines - Overview

# Overlapping Class distributions

- If the training data in the feature space are not linearly separable?
- Allow some data points to be on the 'wrong side' of the decision boundary.
- Increase a penalty with distance from the decision boundary.
- Assume, the penalty increases linearly with distance.
- Introduce **slack** variable  $\xi_n \geq 0$  for each data point  $n$  .

$$\xi_n \begin{cases} = 0, & \text{data point is correctly classified and} \\ & \text{on margin boundary or beyond} \\ < 1, & \text{data point is in correct margin} \\ = 1, & \text{data point is on the decision boundary} \\ > 1, & \text{data point is misclassified} \end{cases}$$



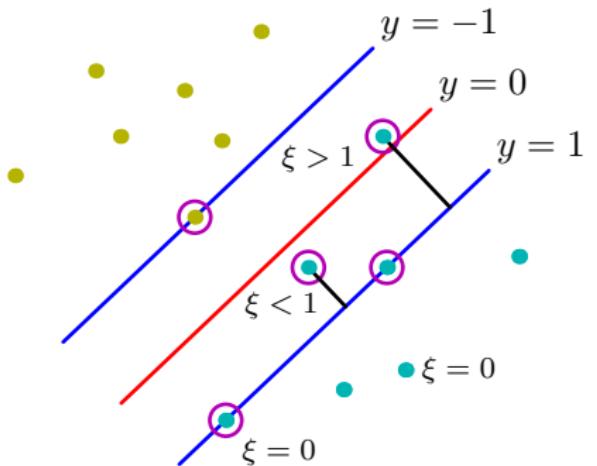
Sparse Kernel Machines

Maximum Margin  
ClassifiersOverlapping Class  
DistributionsRelevance Vector  
Machines - Overview

# Overlapping Class distributions

- Introduce **slack** variable  $\xi_n \geq 0$  for each data point  $n$ .

$$\xi_n = \begin{cases} 0, & \text{data point is correctly classified and} \\ & \text{on margin boundary or beyond} \\ |t_n - y(\mathbf{x})|, & \text{otherwise} \end{cases}$$





Sparse Kernel Machines

Maximum Margin  
ClassifiersOverlapping Class  
DistributionsRelevance Vector  
Machines - Overview

# Overlapping Class distributions

- Constraints of the separable classification

$$t_n y(\mathbf{x}_n) \geq 1, \quad n = 1, \dots, N$$

change now to

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n, \quad n = 1, \dots, N$$

- Minimise now

$$C \sum_{n=1}^N \xi_n + \frac{1}{2} \|\mathbf{w}\|^2$$

where  $C$  controls the trade-off between the slack variable penalty and the margin.

- Any misclassified point contributes  $\xi_n > 0$ , therefore  $\sum_{n=1}^N \xi_n$  is an upper bound on the number of misclassified points.



Sparse Kernel Machines

Maximum Margin  
ClassifiersOverlapping Class  
DistributionsRelevance Vector  
Machines - Overview

# Overlapping Class distributions

- The Lagrangian with the Lagrange multipliers  $\mathbf{a} = (a_1, \dots, a_N)^T$  and  $\boldsymbol{\mu} = (\mu_1, \dots, \mu_N)^T$  is now

$$\begin{aligned} L(\mathbf{w}, b, \boldsymbol{\xi}, \mathbf{a}, \boldsymbol{\mu}) = & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n \\ & - \sum_{n=1}^N a_n \{t_n y(\mathbf{x}_n) - 1 + \xi_n\} - \sum_{n=1}^N \mu_n \xi_n. \end{aligned}$$

- The KKT conditions for  $n = 1, \dots, N$  are

$$a_n \geq 0$$

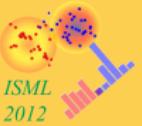
$$t_n y(\mathbf{x}_n) - 1 + \xi_n \geq 0$$

$$a_n (t_n y(\mathbf{x}_n) - 1 + \xi_n) = 0$$

$$\mu_n \geq 0$$

$$\xi_n \geq 0$$

$$\mu_n \xi_n = 0.$$



- The dual Lagrangian after eliminating  $\mathbf{w}$ ,  $b$ ,  $\xi$ , and  $\mu$  is then

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

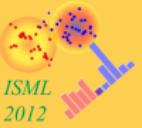
$$\sum_{n=1}^N a_n t_n = 0$$

$$0 \leq a_n \leq C \quad n = 1, \dots, N.$$

- The only change from the separable case, is the **box** constraint via the parameter  $C$ .

Sparse Kernel Machines

Maximum Margin  
ClassifiersOverlapping Class  
DistributionsRelevance Vector  
Machines - Overview



Sparse Kernel Machines

Maximum Margin  
ClassifiersOverlapping Class  
DistributionsRelevance Vector  
Machines - Overview

# Overlapping Class distributions

- Equivalent formulation by Schölkopf et. al. (2000)

$$\tilde{L}(\mathbf{a}) = -\frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

$$\sum_{n=1}^N a_n t_n = 0$$

$$0 \leq a_n \leq 1/N$$

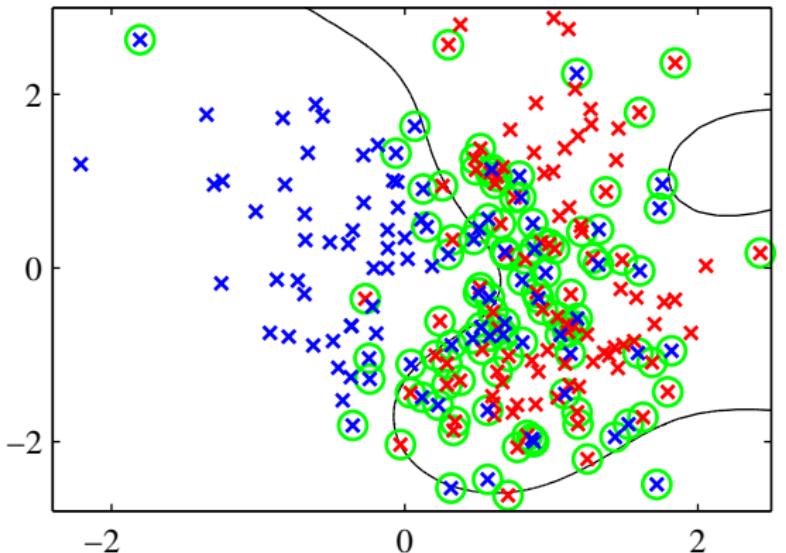
$$\sum_{n=1}^N a_n \geq \nu \quad n = 1, \dots, N.$$

where  $\nu$  is both

- ➊ an upper bound on the fraction of margin errors, and,
- ➋ a lower bound on the fraction of support vectors.



Sparse Kernel Machines

Maximum Margin  
ClassifiersOverlapping Class  
DistributionsRelevance Vector  
Machines - Overview

The  $\nu$ -SVM algorithm using Gaussian kernels  $\exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$  with  $\gamma = 0.45$  applied to a nonseparable data set in two dimensions. Support vectors are indicated by circles.



Sparse Kernel Machines

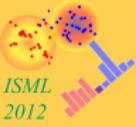
Maximum Margin  
Classifiers

Overlapping Class  
Distributions

Relevance Vector  
Machines - Overview

# Support Vector Machines - Limitations

- 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.



Sparse Kernel Machines

Maximum Margin  
ClassifiersOverlapping Class  
DistributionsRelevance Vector  
Machines - Overview

# Relevance Vector Machines

- Assume a Bayesian model as in Bayesian Linear Regression with the probability of the target  $t$  given by

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

and the linear model

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

with fixed nonlinear basis functions  $\phi(\mathbf{x})$  (including a constant term to accommodate for the bias).

- But now the prior over  $\mathbf{w}$  includes a different precision  $\alpha_i$  for each of the components of  $\mathbf{w}$

$$p(\mathbf{w} \mid \boldsymbol{\alpha}) = \prod_{i=1}^M \mathcal{N}(w_i \mid 0, \alpha_i^{-1}).$$



# Relevance Vector Machines

- When using this prior for the weights  $\mathbf{w}$

$$p(\mathbf{w} \mid \boldsymbol{\alpha}) = \prod_{i=1}^M \mathcal{N}(w_i \mid 0, \alpha_i^{-1})$$

and maximising the evidence with respect to the hyperparameters  $\alpha_i$ , a significant portion of the  $\alpha_i$  will go to infinity.

- The corresponding  $w_i$  will therefore be concentrated at zero, and play no role for the prediction.

Sparse Kernel Machines

Maximum Margin  
Classifiers

Overlapping Class  
Distributions

Relevance Vector  
Machines - Overview

# Relevance Vector Machines



- The Relevance Vector Machine iteratively calculates the  $\alpha_i$  and  $\beta$  from the training data by optimising a nonconvex function.
- Disadvantages
  - ➊ Nonconvex function means that the algorithm can get stuck in some local minimum.
  - ➋ Training times can be longer than for SVM.
- Advantages
  - ➌ As a Bayesian method, the parameters governing complexity and noise are determined from the input data. (Compare to SVM where we needed cross-validation).
  - ➍ Number of relevance vectors is much smaller than the number of support vectors in a SVM.
  - ➎ Therefore, prediction is much faster than with SVM.
  - ➏ Extends well to multiclass learning.

Sparse Kernel Machines

Maximum Margin  
Classifiers

Overlapping Class  
Distributions

Relevance Vector  
Machines - Overview

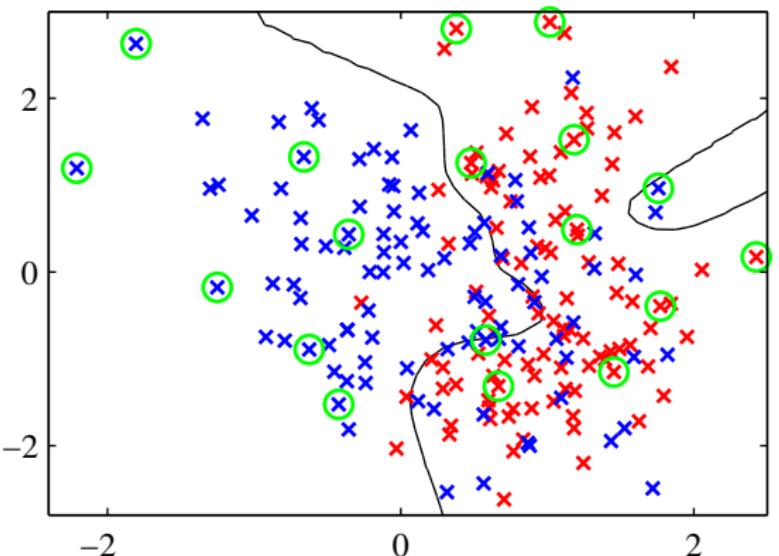


Sparse Kernel Machines

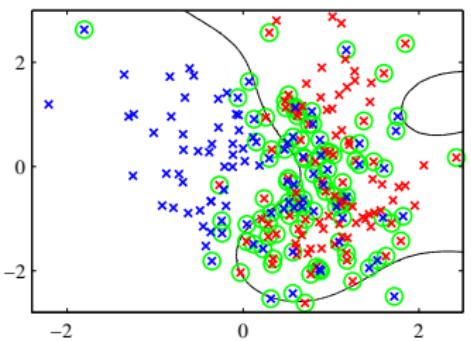
Maximum Margin  
Classifiers

Overlapping Class  
Distributions

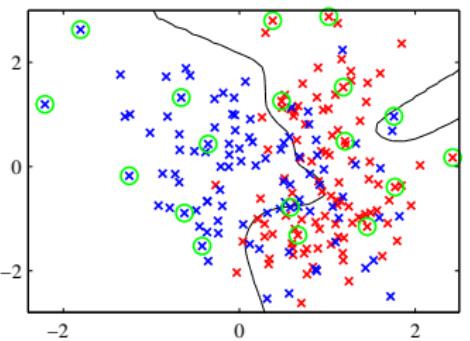
Relevance Vector  
Machines - Overview



Decision boundaries for a two-class problem found by a  
Relevance Vector Machine.



Support Vector Machine



Relevance Vector Machine

Sparse Kernel Machines

Maximum Margin  
Classifiers

Overlapping Class  
Distributions

Relevance Vector  
Machines - Overview

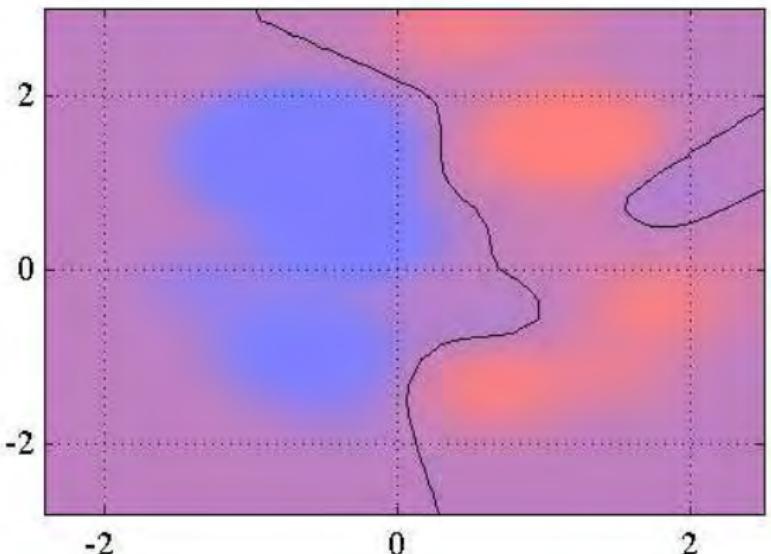


Sparse Kernel Machines

Maximum Margin  
Classifiers

Overlapping Class  
Distributions

Relevance Vector  
Machines - Overview



Decision boundaries and posterior probability for a two-class problem found by a Relevance Vector Machine.



*Motivation*

*Bayesian Network*

*Plate Notation*

*Conditional  
Independence*

## Part XIII

# *Probabilistic Graphical Models 1*

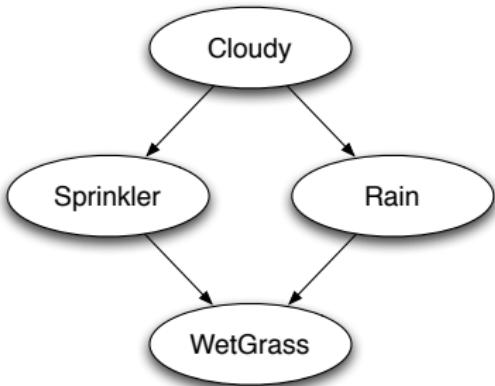


Motivation

Bayesian Network

Plate Notation

Conditional  
Independence



- 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**Bayesian Network**Plate Notation**Conditional  
Independence*

# *Motivation via Independence*

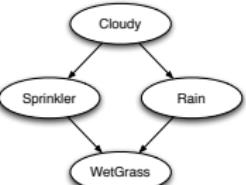
- Model the conditional probabilities

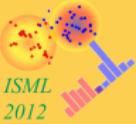
$p(C = F)$	$p(C = T)$
0.2	0.8

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

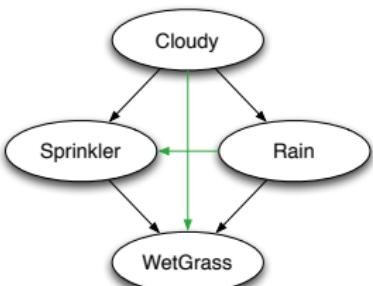


*Motivation**Bayesian Network**Plate Notation**Conditional  
Independence*

- 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

Bayesian Network

Plate Notation

Conditional  
Independence

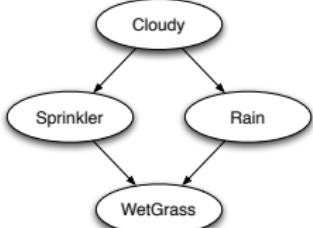
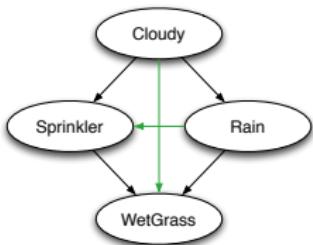
# Motivation via Independence

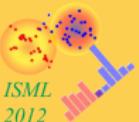
$$p(W) =$$

$$\sum_{S,R,C} p(W | S, R, C) p(S | R, C) p(R | C) p(C)$$

$$p(W) =$$

$$\sum_{S,R} p(W | S, R) \sum_C p(S | C) p(R | C) p(C)$$





## Motivation

Bayesian Network

Plate Notation

Conditional  
Independence

# Motivation via Distributive Law

- Two key observations when dealing with probabilities
  - Distributive Law can save operations

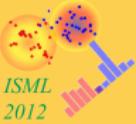
$$\underbrace{a(b + c)}_{\text{2 operations}} = \underbrace{ab + ac}_{\text{3 operations}}$$

- If some probabilities do not depend on all random variables, we might be able to factor them out. For example, assume

$$p(x_1, x_3 | x_2) = p(x_1 | x_2) p(x_3 | x_2),$$

then (using  $\sum_{x_3} p(x_3 | x_2) = 1$ )

$$\begin{aligned} p(x_1) &= \sum_{x_2, x_3} p(x_1, x_2, x_3) = \sum_{x_2, x_3} p(x_1, x_3 | x_2) p(x_2) \\ &= \underbrace{\sum_{x_2, x_3} p(x_1 | x_2) p(x_3 | x_2) p(x_2)}_{O(|\mathcal{X}_1||\mathcal{X}_2||\mathcal{X}_3|)} = \underbrace{\sum_{x_2} p(x_1 | x_2) p(x_2)}_{O(|\mathcal{X}_1||\mathcal{X}_2|)} \end{aligned}$$

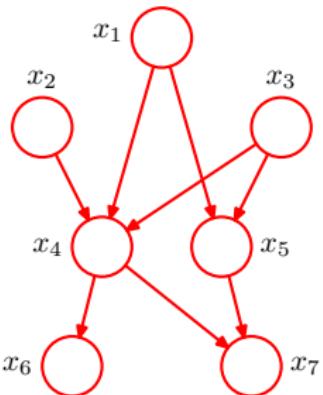


# Motivation via Complexity Reduction

- How to deal with more complex expression?

$$p(x_1) p(x_2) p(x_3) p(x_4|x_1, x_2, x_3) p(x_5|x_1, x_3) p(x_6|x_4) p(x_7|x_4, x_5)$$

- Graphical models



Motivation

Bayesian Network

Plate Notation

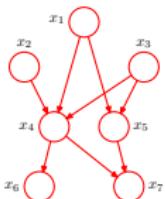
Conditional  
Independence

# Probabilistic Graphical Models

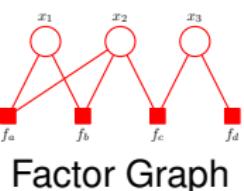


## Graphical models

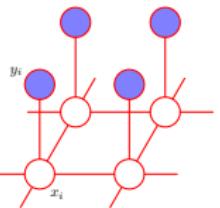
- Visualise the structure of a probabilistic model
- Complex computations with formulas → manipulations with graphs
- Obtain insights into model properties by inspection
- Develop and motivate new models



Bayesian Network



Factor Graph



Markov Random  
Field

Motivation

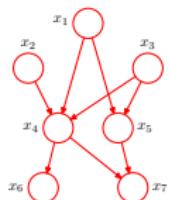
Bayesian Network

Plate Notation

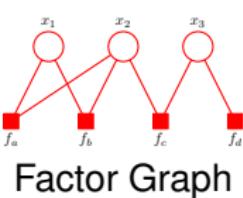
Conditional  
Independence

# Probabilistic Graphical Models

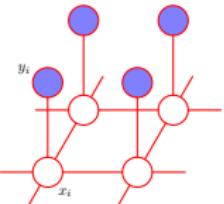
- Graph
  - ① Nodes (vertices) : a random variable
  - ② Edges (links, arcs; directed or undirected) : probabilistic relationship
- Directed Graph : **Bayesian Network** (also called Directed Graphical Model) expressing **causal** relationship between variables
- Undirected Graph : **Markov Random Field** expressing **soft constraints** between variables
- Factor Graph : convenient for solving **inference** problems (derived from Bayesian Networks or Markov Random Fields).



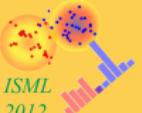
Bayesian Network



Factor Graph



Markov Random Field

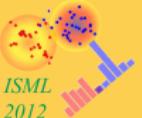


Motivation

Bayesian Network

Plate Notation

Conditional  
Independence



Motivation

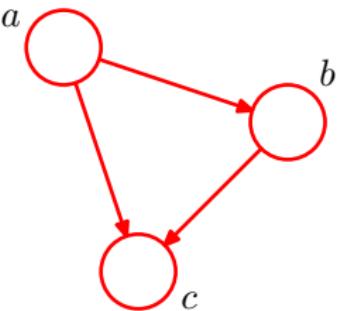
Bayesian Network

Plate Notation

Conditional  
Independence

$$p(a, b, c) = p(c | a, b) p(a, b) = p(c | a, b) p(b | a) p(a)$$

- 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.



- We have chosen a particular ordering of the variables !

# Bayesian Network



- General case for  $K$  variables

$$p(x_1, \dots, x_K) = p(x_K | x_1, \dots, x_{K-1}) \dots p(x_2 | x_1) p(x_1)$$

- The graph of this distribution is **fully** connected. (Prove it.)
- What happens if we deal with a distribution represented by a graph which is **not** fully connected?
- Can not be the most general distribution anymore.
- The **absence** of edges carries important information.

Motivation

Bayesian Network

Plate Notation

Conditional  
Independence

# Bayesian Network - Joint Distribution → Graph

$$p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5)$$



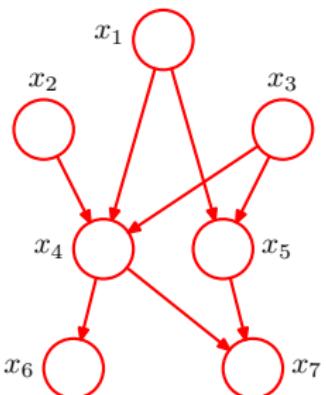
Motivation

Bayesian Network

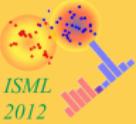
Plate Notation

Conditional  
Independence

- ① Draw a node for each conditional distribution associated with a random variable.
- ② Draw an edge **from** each conditional distribution associated with a random variable **to** all other conditional distribution which are conditioned on this variable.



# Bayesian Network - Graph → Joint Distribution

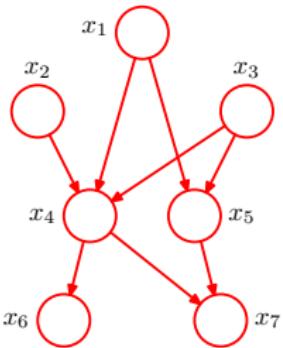


Motivation

Bayesian Network

Plate Notation

Conditional  
Independence



Can we get the expression from the graph?

- ① Write a product of probability distributions, one for each associated random variable. ↔ Draw a node for each conditional distribution associated with a random variable.
- ② 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.

$$p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5)$$



Motivation

Bayesian Network

Plate Notation

Conditional  
Independence

- The joint distribution defined by a graph is given by the product, over all of the nodes of the graph, of a conditional distribution for each node conditioned on the variables corresponding to the parents of the node in the graph.

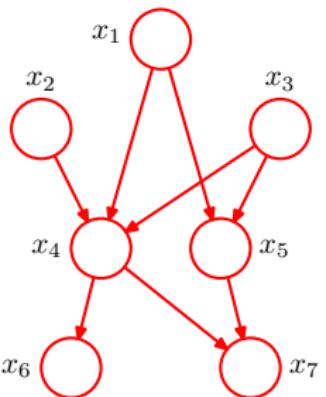
$$p(\mathbf{x}) = \prod_{k=1}^K p(x_k | \text{pa}(x_k))$$

where  $\text{pa}(x_k)$  denotes the set of parents of  $x_k$  and  $\mathbf{x} = (x_1, \dots, x_K)$ .

- Restriction : Graph must be a **directed acyclic graph** (DAG).

# Bayesian Network - Joint Distribution $\leftrightarrow$ Graph

- 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.



Motivation

Bayesian Network

Plate Notation

Conditional  
Independence



Motivation

Bayesian Network

Plate Notation

Conditional  
Independence

# Bayesian Network - Joint Distribution $\leftrightarrow$ Graph

- Given

$$p(\mathbf{x}) = \prod_{k=1}^K p(x_k | \text{pa}(x_k)).$$

- 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.

# Bayesian Network - Plate Notation



Motivation

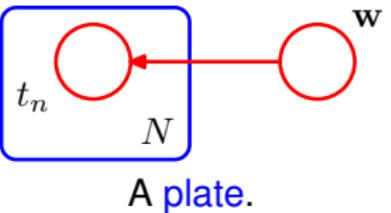
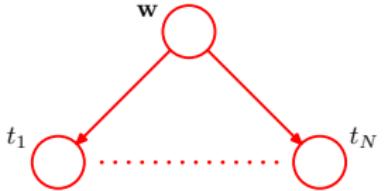
Bayesian Network

Plate Notation

Conditional  
Independence

- 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

$$p(\mathbf{t}, \mathbf{w}) = p(\mathbf{w}) \prod_{k=1}^N p(t_n | \mathbf{w})$$





Motivation

Bayesian Network

Plate Notation

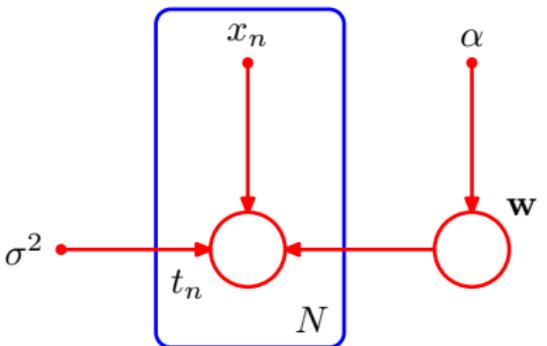
Conditional  
Independence

- Include also the parameters into the graphical model

$$p(\mathbf{t}, \mathbf{w} | \mathbf{x}, \alpha, \sigma^2) = p(\mathbf{w} | \alpha) \prod_{k=1}^N p(t_n | \mathbf{w}, x_n, \sigma^2)$$

Random variables = open circles

Deterministic variables = smaller solid circles



# Bayesian Network - Plate Notation



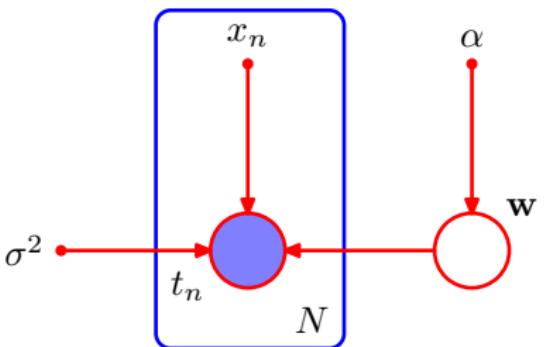
Motivation

Bayesian Network

Plate Notation

Conditional  
Independence

- Random variables
  - Observed random variables, e.g.  $t$
  - Unobserved random variables, e.g.  $w$ ,  
(latent random variables, hidden random variables)
- Shade the observed random variables in the graphical model.

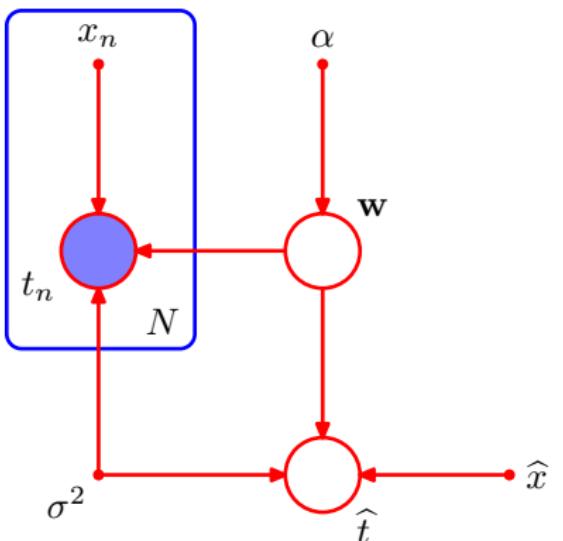


## *Bayesian Network - Plate Notation*



### *Plate Notation*

$$p(\hat{t}, \mathbf{t}, \mathbf{w} | \hat{x}, \mathbf{x}, \alpha, \sigma^2) = \left[ \prod_{k=1}^N p(t_n | x_n, \mathbf{w}, \sigma^2) \right] p(\mathbf{w} | \alpha) p(\hat{t} | \hat{x}, \mathbf{w}, \sigma^2)$$



Polynomial regression model including prediction.



Motivation

Bayesian Network

Plate Notation

Conditional  
Independence

# Bayesian Network - 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.

# Bayesian Network - Conditional Independence



Motivation

Bayesian Network

Plate Notation

Conditional  
Independence

## 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 :  $X \perp\!\!\!\perp W | Z, Y$   
and  $X \perp\!\!\!\perp Y | Z \implies X \perp\!\!\!\perp W, Y | Z$

Intersection :  $X \perp\!\!\!\perp Y | Z, W$   
and  $X \perp\!\!\!\perp W | Z, Y \implies X \perp\!\!\!\perp Y, W | Z$

Note: Intersection is only valid for  $p(X), p(Y), p(Z), p(W) > 0$ .



Motivation

Bayesian Network

Plate Notation

Conditional  
Independence

# Bayesian Network - Conditional Independence

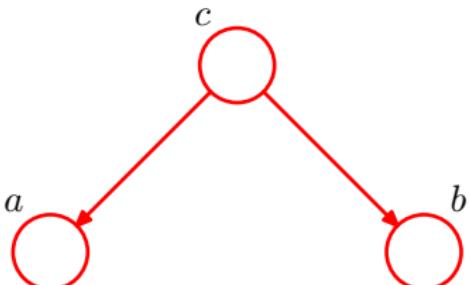
- Can we work with the graphical model directly?
- Check the simplest examples containing only three nodes.
- First example has joint distribution

$$p(a, b, c) = p(a | c) p(b | c) p(c)$$

- 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).





Motivation

Bayesian Network

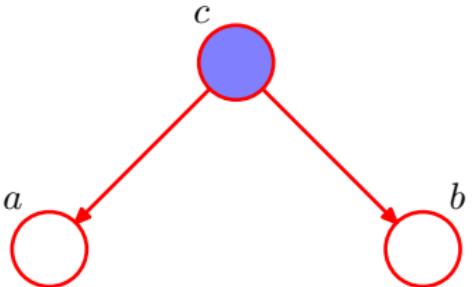
Plate Notation

Conditional  
Independence

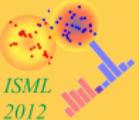
- Now condition on  $c$ .

$$p(a, b | c) = \frac{p(a, b, c)}{p(c)} = p(a | c)p(b | c)$$

- Therefore  $a \perp\!\!\!\perp b | c$ .



# Bayesian Network - Conditional Independence

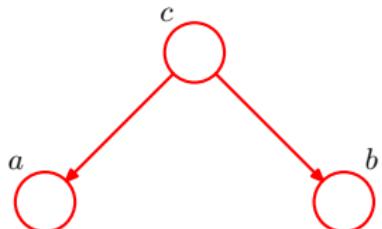


Motivation

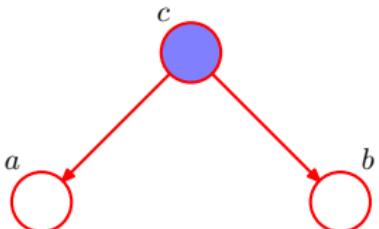
Bayesian Network

Plate Notation

Conditional  
Independence



Not  $a \perp\!\!\!\perp b | \emptyset$



$a \perp\!\!\!\perp b | c$

# Bayesian Network - Conditional Independence



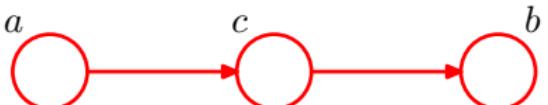
- Second example.

$$p(a, b, c) = p(a) p(c | a) p(b | c)$$

- 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 :  $a \perp\!\!\!\perp b | \emptyset$ .



Motivation

Bayesian Network

Plate Notation

Conditional  
Independence

# Bayesian Network - Conditional Independence

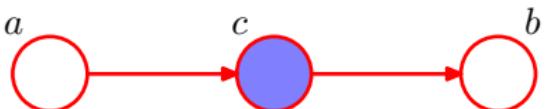


- Now condition on  $c$ .

$$p(a, b | c) = \frac{p(a, b, c)}{p(c)} = \frac{p(a)p(c|a)p(b|c)}{p(c)} = p(a|c)p(b|c)$$

where we used Bayes' theorem  $p(c|a) = p(a|c)p(c)/p(a)$ .

- Therefore  $a \perp\!\!\!\perp b | c$ .



Motivation

Bayesian Network

Plate Notation

Conditional  
Independence



Motivation

Bayesian Network

Plate Notation

Conditional  
Independence

# Bayesian Network - Conditional Independence

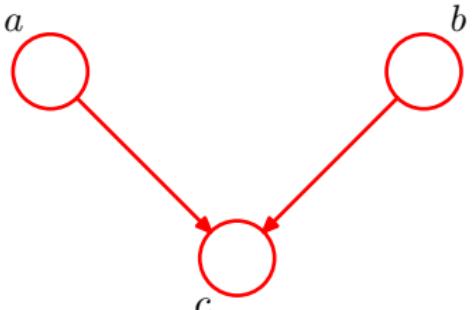
- Third example. (A little bit more subtle.)

$$p(a, b, c) = p(a)p(b)p(c | a, b)$$

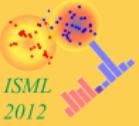
- Marginalise over  $c$  to test for independence.

$$\begin{aligned} p(a, b) &= \sum_c p(a)p(b)p(c | a, b) = p(a)p(b)\sum_c p(c | a, b) \\ &= p(a)p(b) \end{aligned}$$

- $a$  and  $b$  are independent if NO variable is observed:  
 $a \perp\!\!\!\perp b | \emptyset$ .



# Bayesian Network - Conditional Independence



Motivation

Bayesian Network

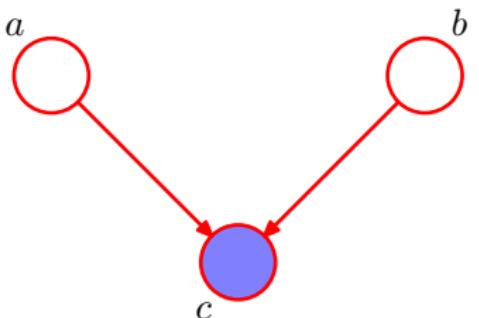
Plate Notation

Conditional  
Independence

- 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 :  $a \perp\!\!\!\perp b | c$ .



# Bayesian Network - Conditional Independence



Motivation

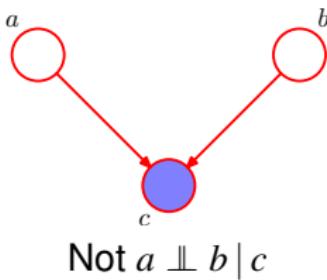
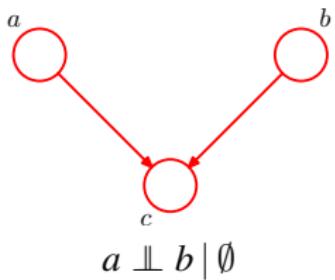
Bayesian Network

Plate Notation

Conditional  
Independence

## 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  $b$ .



# Bayesian Network - Conditional Independence

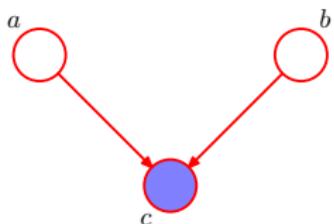


Motivation

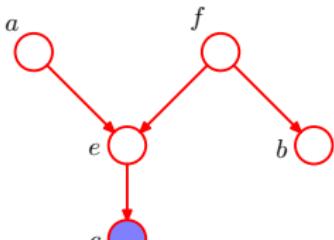
Bayesian Network

Plate Notation

Conditional  
Independence



Not  $a \perp\!\!\!\perp b | c$



Not  $a \perp\!\!\!\perp f | c$



Motivation

Bayesian Network

Plate Notation

Conditional  
Independence

- Conditional Independence and Factorisation have been shown to be equivalent for all possible configuration of three nodes.
- Are they equivalent for **any** Bayesian Networks?
- Characterise which conditional independence statements hold for an arbitrary factorisation and check whether a distribution satisfying those statements will have such a factorisation.

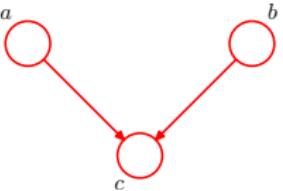
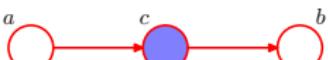
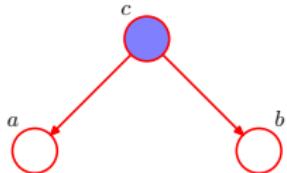
# Bayesian Network - D-Separation



## 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.



Motivation

Bayesian Network

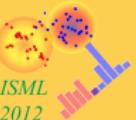
Plate Notation

Conditional  
Independence

# Bayesian Network - $D$ -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  $A \perp\!\!\!\perp B | C$ .

(Note:  $D$ -separation stands for 'directional' separation.)



Motivation

Bayesian Network

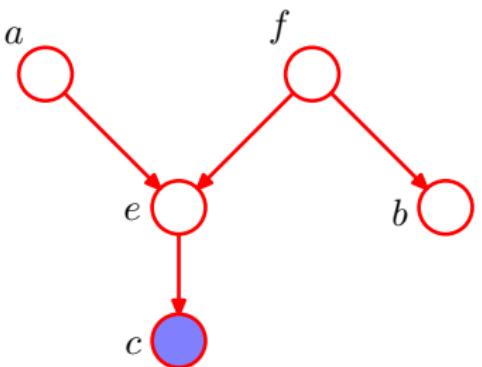
Plate Notation

Conditional  
Independence

# Bayesian Network - 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.

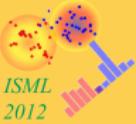


Motivation

Bayesian Network

Plate Notation

Conditional  
Independence



Motivation

Bayesian Network

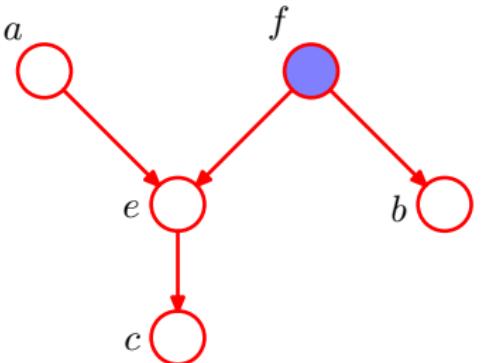
Plate Notation

Conditional  
Independence

# Bayesian Network - D-separation

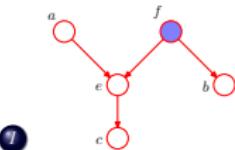
## Another example

- The path from  $a$  to  $b$  is blocked by  $f$  because  $f$  is a TT-node and observed. Therefore,  $a \perp\!\!\!\perp b | f$ .
- 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  $a \perp\!\!\!\perp b | f$ .





# Finding $p(a, b | f)$ - 'Analytical' method



- ② Conditional probability with the given observable(s):

$$p(a, b, c, e | f)$$

- ③ Rewrite it as a joint distribution over all variables

$$p(a, b, c, e | f) = \frac{p(a, b, c, e, f)}{p(f)}$$

- ④ Factorise the joint probability according to the graph

$$p(a, b, c, e | f) = \frac{p(a, b, c, e, f)}{p(f)} = \frac{p(a)p(f)p(e | a, f)p(b | f)p(c | e)}{p(f)}$$

- ⑤ Marginalise over all variables we don't care about

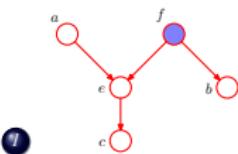
$$p(a, b | f) = \sum_{c,e} \frac{p(a)p(f)p(e | a, f)p(b | f)p(c | e)}{p(f)} = p(a)p(b | f)$$

Motivation

Bayesian Network

Plate Notation

Conditional  
Independence



- ② Check whether  $a \perp\!\!\!\perp b | f$  holds or not.

Result :  $a \perp\!\!\!\perp b | f$  holds.

- Reason : The path from  $a$  to  $b$  is blocked by  $f$  because  $f$  is a TT-node and observed. Therefore,  $a \perp\!\!\!\perp b | f$ .

- ③ Write down the factorisation

$$p(a, b | f) = p(a | f) p(b | f) = p(a) p(b | f)$$

Motivation

Bayesian Network

Plate Notation

Conditional  
Independence

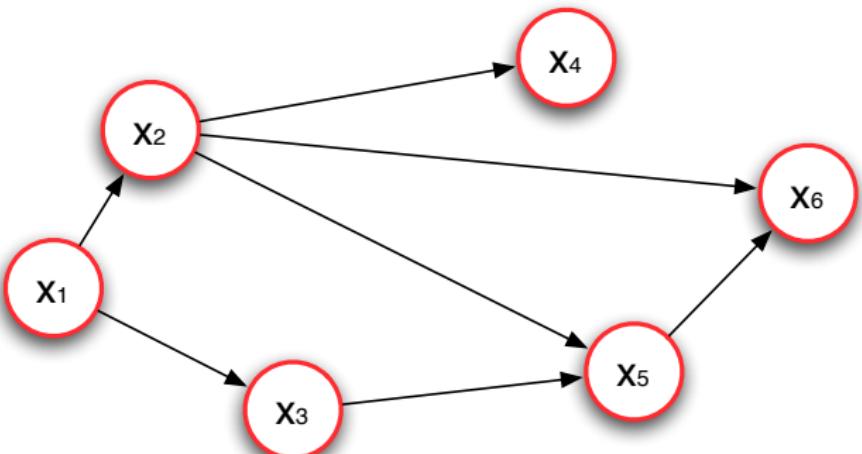


Motivation

Bayesian Network

Plate Notation

Conditional  
Independence



# Bayesian Network - D-separation

## A third example

- Is  $x_3$  d-separated from  $x_6$  given  $x_1$  and  $x_5$  ?
- Mark  $x_1$  and  $x_5$  as observed.



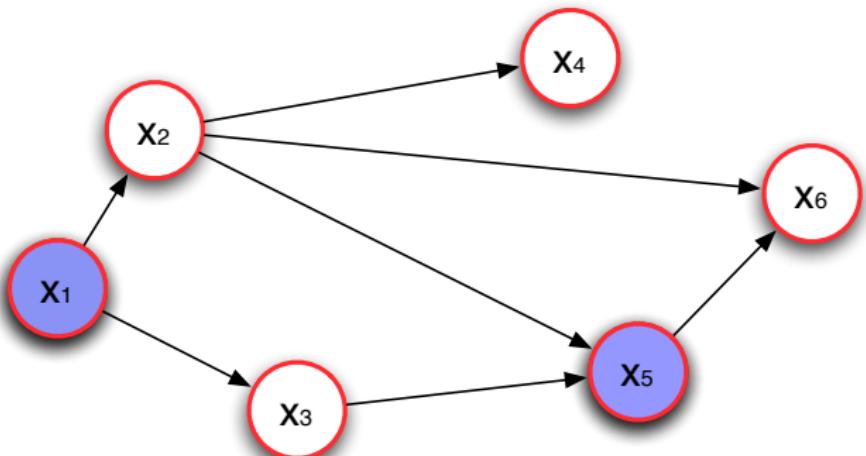
Motivation

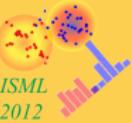
Bayesian Network

Plate Notation

Conditional  
Independence

- Is  $x_3$  *d*-separated from  $x_6$  given  $x_1$  and  $x_5$  ?





Motivation

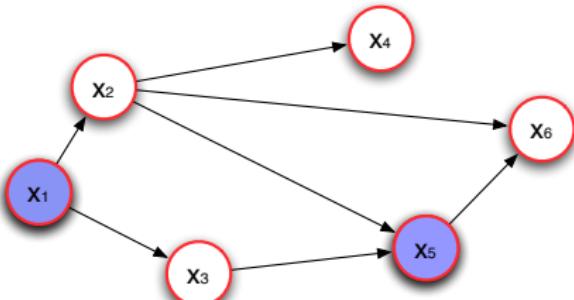
Bayesian Network

Plate Notation

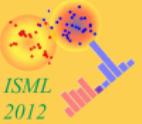
Conditional  
Independence

# Bayesian Network - D-separation

- Is  $x_3$  d-separated from  $x_6$  given  $x_1$  and  $x_5$  ?
- 4 paths between  $x_3$  and  $x_6$ .
- $\{x_3, x_1, x_2\}$  is blocked because  $x_1$  is TT-node and observed.
- $\{x_3, x_5, x_6\}$  is blocked because  $x_5$  is a HT-node and observed.
- $\{x_3, x_5, x_2\}$  is not blocked because  $x_5$  is a HH-node and observed.
- $\{x_5, x_2, x_6\}$  is not blocked because  $x_2$  is a TT-node and unobserved.
- Therefore,  $x_3$  is not d-separated from  $x_6$  given  $x_1$  and  $x_5$  as not all paths between  $x_3$  and  $x_6$  are blocked.



# *Conditional Independence $\Leftrightarrow$ Factorisation*



## *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  $A \perp\!\!\!\perp B | C$ .*

Motivation

Bayesian Network

Plate Notation

Conditional  
Independence

## *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.*



Motivation

Bayesian Network

Plate Notation

Conditional  
Independence

# Conditional Independence $\Leftrightarrow$ Factorisation

## 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.



ISML  
2012

Motivation

Bayesian Network

Plate Notation

Conditional  
Independence

- Given a set of Conditional Independence statements.
- Adding another statement will in general produce other statements.
- All statements can be read as  $d$ -separation in a DAG.
- However, there are sets of Conditional Independence statements which **cannot** be satisfied by **any** Bayesian Network.



Motivation

Bayesian Network

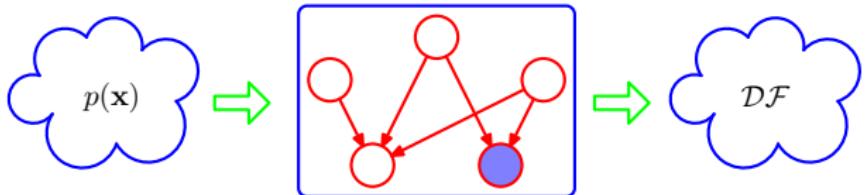
Plate Notation

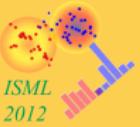
Conditional  
Independence

# Conditional Independence $\Leftrightarrow$ Factorisation

## The broader picture

- A directed graphical model can be viewed as a filter accepting probability distributions  $p(\mathbf{x})$  and only letting these through which satisfy the factorisation property. The set of all possible distribution  $p(\mathbf{x})$  which pass through the filter is denoted as  $\mathcal{DF}$ .
- Alternatively, only these probability distributions  $p(\mathbf{x})$  pass through the filter (graph), which respect the conditional independencies implied by the d-separation properties of the graph.
- The d-separation theorem says that the resulting set  $\mathcal{DF}$  is the same in both cases.





## Part XIV

# *Probabilistic Graphical Models 2*

*Markov Random Fields*

*Bayesian Networks vs.  
Markov Random Fields*



Markov Random Fields

Bayesian Networks vs.  
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.



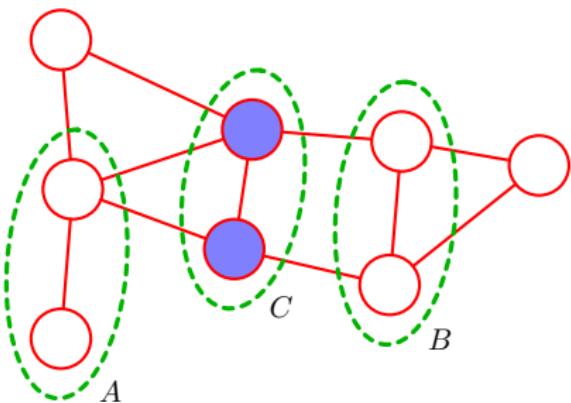
Markov Random Fields

Bayesian Networks vs.  
Markov Random Fields

# Graph Separation

## Definition (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  $G$ .





Markov Random Fields

Bayesian Networks vs.  
Markov Random Fields

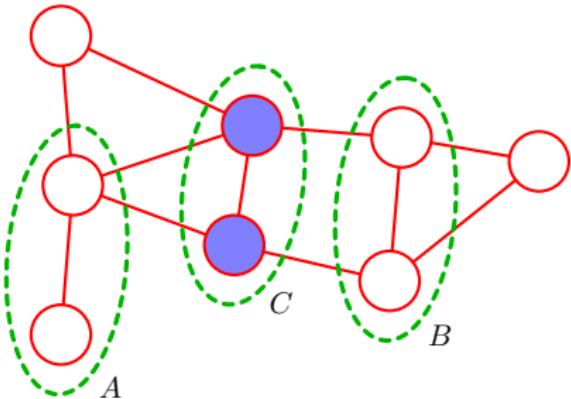
# Conditional Independence in MRFs

## Definition (Conditional Independence in Markov Random Field)

In an undirected graph  $G$ , having  $A$ ,  $B$  and  $C$  disjoint subsets of nodes,  $A$  is conditionally independent of  $B$  given  $C$

$$A \perp\!\!\!\perp B \mid C$$

iff  $C$  separates  $A$  from  $B$  in  $G$ .





Markov Random Fields

Bayesian Networks vs.  
Markov Random Fields

# Markov Random Field

## 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".

ISML  
2012

Markov Random Fields

Bayesian Networks vs.  
Markov Random Fields

# Factorisation in MRFs

- 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.

$$p(x_i, x_j | \mathbf{x}_{\setminus\{i,j\}}) = p(x_i | \mathbf{x}_{\setminus\{i,j\}}) p(x_j | \mathbf{x}_{\setminus\{i,j\}})$$

where  $\mathbf{x}_{\setminus\{i,j\}}$  denotes the set of all variables  $\mathbf{x}$  with  $x_i$  and  $x_j$  removed.

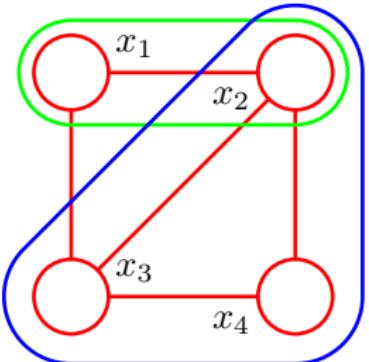
- Directly connected nodes are therefore not conditionally independent given all the others. Can we use this for the factorisation of the graph?



Markov Random Fields

Bayesian Networks vs.  
Markov Random Fields

- 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.)

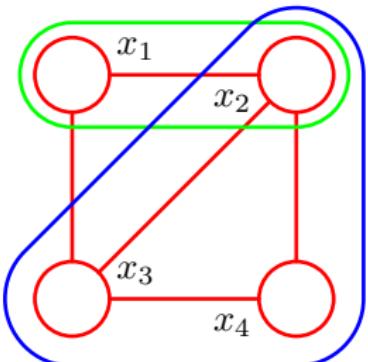




Markov Random Fields

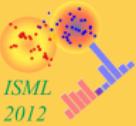
Bayesian Networks vs.  
Markov Random Fields

- We can express the factorisation with the help of the cliques.
- In fact, we only need the maximal cliques, as any function over the variables of a subset of a maximal clique can be expressed as a function of the members of the maximal clique.
- Denote by  $\mathcal{C}$  the set of maximal cliques of a graph.
- For a maximal clique  $C \in \mathcal{C}$ , denote by  $x_C$  the subset of the variables  $x$  which belong to  $C$ .



# Factorisation using Cliques

©2012

Christfried Webers  
NICTAThe Australian National  
University

- A probability distribution  $p(\mathbf{x})$  **factorises** with respect to a given undirected graph  $G$  if it can be written as

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(\mathbf{x}_C)$$

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  $p(\mathbf{x})$ .

Markov Random Fields

Bayesian Networks vs.  
Markov Random Fields



Markov Random Fields

Bayesian Networks vs.  
Markov Random Fields

# Conditional Independence $\Leftrightarrow$ 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))

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.



Markov Random Fields

Bayesian Networks vs.  
Markov Random Fields

# Factorisation with 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.

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(\mathbf{x}_C) = \frac{1}{Z} \exp\left\{-\sum_{C \in \mathcal{C}} E(\mathbf{x}_C)\right\}$$

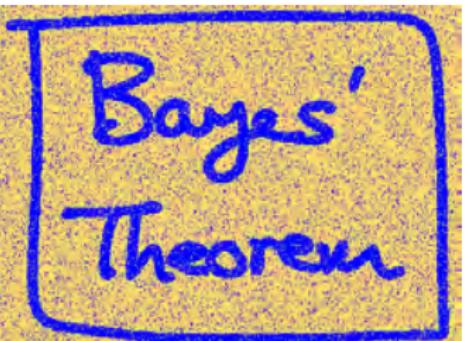


Markov Random Fields

Bayesian Networks vs.  
Markov Random Fields



Original image.



After randomly changing  
10% of the pixels.

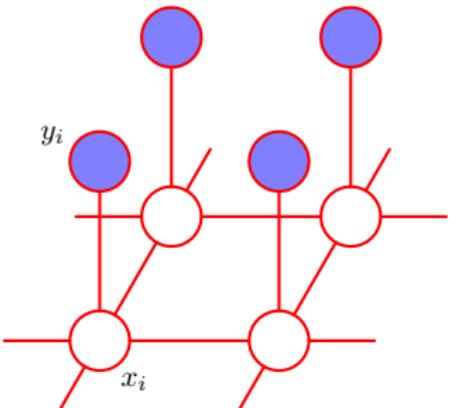


Markov Random Fields

Bayesian Networks vs.  
Markov Random Fields

# Example of Image Denoising

- Prior knowledge 1 : We know the noise level is small. Therefore: Strong correlation between original pixels  $x_i$  and noisy pixels  $y_i$ .
- Prior knowledge 2 : Neighbouring pixels  $x_i$  and  $x_j$  in an image are strongly correlated (given a decent resolution).
- Prior knowledge can be captured in a Markov Random Field.





Markov Random Fields

Bayesian Networks vs.  
Markov Random Fields

- Cliques  $\{x_i, y_i\}$  : Choose an energy function of the form  $-\eta x_i y_i$  for  $\eta > 0$  resulting in a potential function  $\exp\{\eta x_i y_i\}$ . This favours equal signs for  $x_i$  and  $y_i$ .
- Cliques  $\{x_i, x_j\}$  where  $i$  and  $j$  are indices of neighbouring pixels : Choose an energy function of the form  $-\beta x_i x_j$  for  $\beta > 0$  again favouring equal signs for  $x_i$  and  $x_j$ .
- We need to accomodate for the fact that one kind of pixels might exist more often than the other kind. This can be done by adding a term  $hx_i$  for each pixel in the noise-free image. (Potential function is an arbitrary, nonnegative function over maximal cliques, so we are allowed to multiply it by any nonnegative function of subsets of the clique.)



Markov Random Fields

Bayesian Networks vs.  
Markov Random Fields

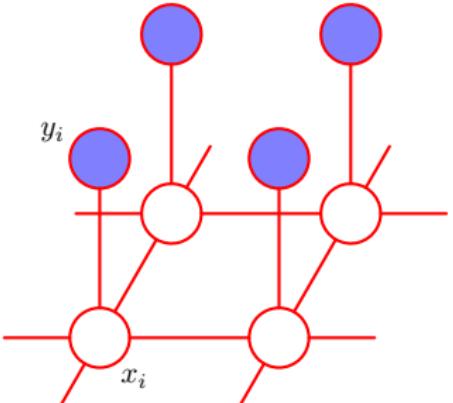
# Example of Image Denoising

- Energy function

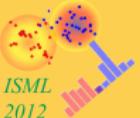
$$E(\mathbf{x}, \mathbf{y}) = h \sum_i x_i - \beta \sum_{i,j} x_i x_j - \eta \sum_i i x_i y_i$$

- which defines a joint distribution over  $\mathbf{x}$  and  $\mathbf{y}$

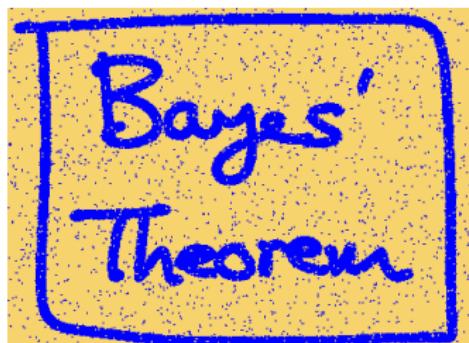
$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \exp\{-E(\mathbf{x}, \mathbf{y})\}.$$



# Example - Iterated Conditional Modes (ICM)



- ➊ Fix the elements for  $y$  as we have them observed.  
(Implicitly defines  $p(\mathbf{x} | \mathbf{y})$ ).
- ➋ Initialise  $x_i = y_i$  for  $i = 1, \dots, D$ .
- ➌ 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.
- ➍ Repeat for another node until stopping criterion is satisfied.



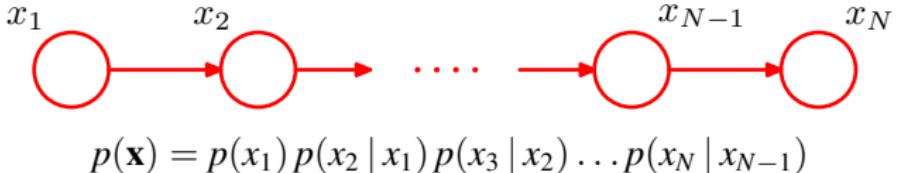
Local minimum (ICM).



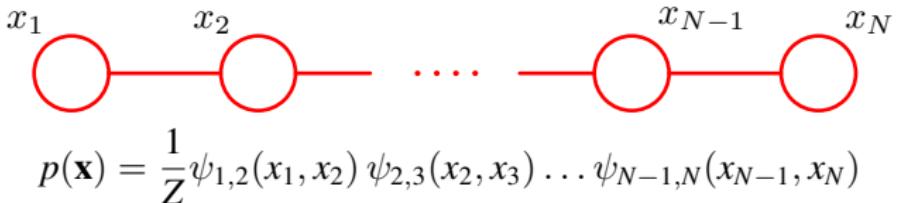
Global minimum (graph-cut).

# Bayesian Networks vs. Markov Random Fields

- Two frameworks for graphical models, can we convert between them?
- Bayesian Network



- Random Markov Field



Markov Random Fields

Bayesian Networks vs.  
Markov Random Fields



# Bayesian Networks vs. Markov Random Fields

- Bayesian Network

$$p(\mathbf{x}) = p(x_1) p(x_2 | x_1) p(x_3 | x_2) \dots p(x_N | x_{N-1})$$

- Random Markov Field

$$p(\mathbf{x}) = \frac{1}{Z} \psi_{1,2}(x_1, x_2) \psi_{2,3}(x_2, x_3) \dots \psi_{N-1,N}(x_{N-1}, x_N)$$

- Find corresponding terms

$$\psi_{1,2}(x_1, x_2) = p(x_1) p(x_2 | x_1)$$

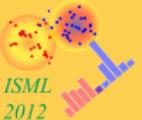
$$\psi_{2,3}(x_2, x_3) = p(x_3 | x_2)$$

$$\vdots$$

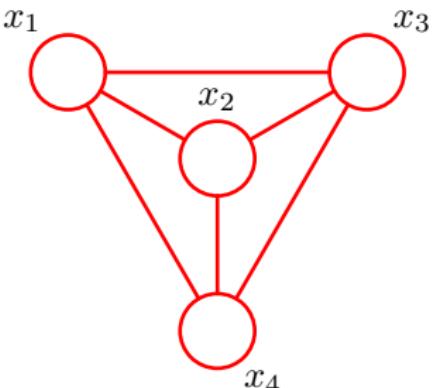
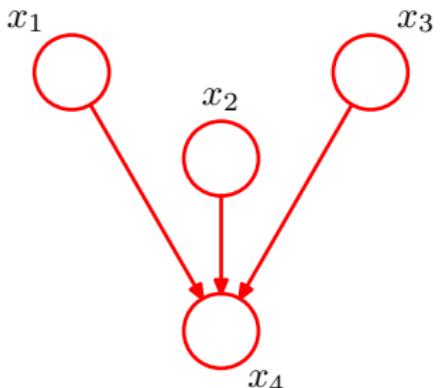
$$\psi_{N-1,N}(x_{N-1}, x_N) = p(x_N | x_{N-1})$$

and note that  $Z = 1$  in this case.

# Bayesian Networks vs. Markov Random Fields



- 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.





Markov Random Fields

Bayesian Networks vs.  
Markov Random Fields

# Bayesian Networks vs. Markov Random Fields

## 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.



Markov Random Fields

Bayesian Networks vs.  
Markov Random Fields

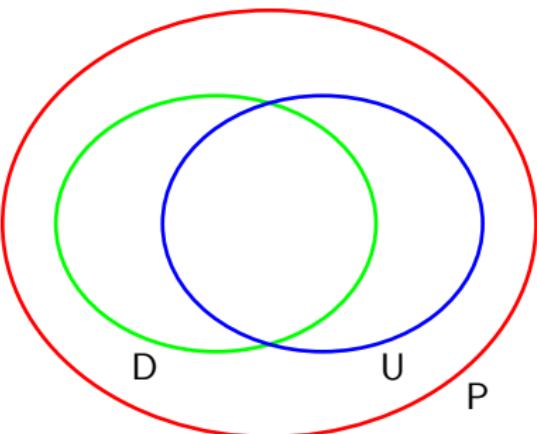
# Bayesian Networks vs. Markov Random Fields

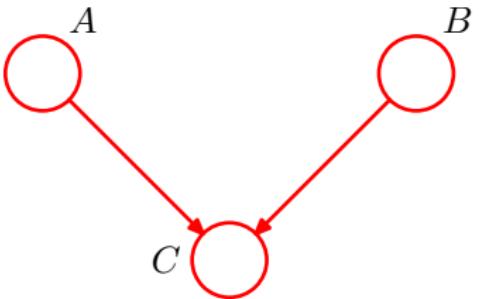
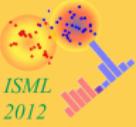
Consider probability distributions depending on  $N$  variables

$P$  set of all probability distributions

$D$  set of all probability distributions that can be represented  
as a perfect map by an **directed** graph

$U$  set of all probability distributions that can be represented  
as a perfect map by an **undirected** graph

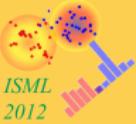




A directed graph whose conditional independence properties cannot be expressed using an undirected graph over the same three variables.

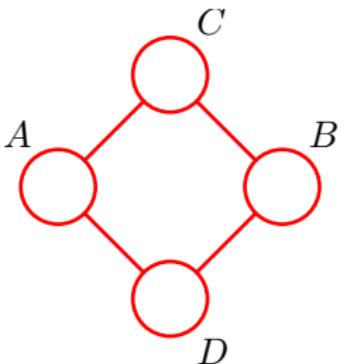
Markov Random Fields

Bayesian Networks vs.  
Markov Random Fields



Markov Random Fields

Bayesian Networks vs.  
Markov Random Fields



An undirected graph whose conditional independence properties cannot be expressed using a directed graph over the same four variables.

# *Comparison of BNs and RMFs*



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).

Markov Random Fields

Bayesian Networks vs.  
Markov Random Fields



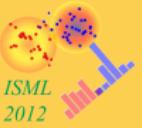
Markov Random Fields

Bayesian Networks vs.  
Markov Random Fields

# Comparison of BNs and RMFs

## 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  $Z = \sum_{\mathbf{x}} \prod_{C \in \mathcal{C}} \psi_C(\mathbf{x}_C)$  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).



## Part XV

# *Probabilistic Graphical Models 3*

*Factor Graphs*

*The Sum-Product  
Algorithm*

*Similar Algorithms*

*Learning the Graph  
Structure*



Factor Graphs

The Sum-Product  
Algorithm

Similar Algorithms

Learning the Graph  
Structure

# Factor Graphs

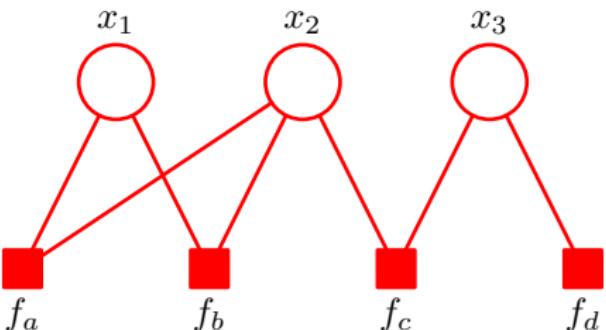
- Write  $p(\mathbf{x})$  in the form of a product of factors

$$p(\mathbf{x}) = \prod_s f_s(\mathbf{x}_s)$$

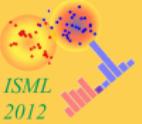
where  $\mathbf{x}_s$  denotes a subset of variables.

- Example

$$p(\mathbf{x}) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3).$$



- More information than in MRF, because there  $f_a(x_1, x_2) f_b(x_1, x_2)$  would be in one potential function.



Factor Graphs

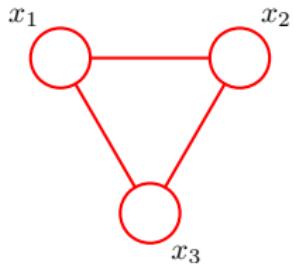
The Sum-Product  
Algorithm

Similar Algorithms

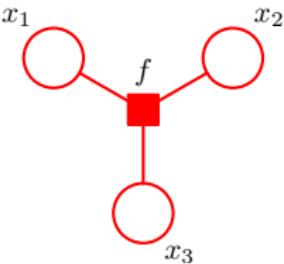
Learning the Graph  
Structure

# Factor Graphs - MRF example

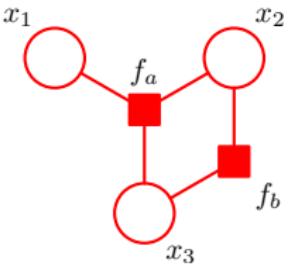
Example of factor graphs representing the same distribution



Undirected graph  
single clique  
potential  
 $\psi(x_1, x_2, x_3)$



Factor graph  
 $f(x_1, x_2, x_3)$   
=  
 $\psi(x_1, x_2, x_3)$



Factor graph  
factors satisfy  
 $f_a(x_1, x_2, x_3) f_b(x_2, x_3)$   
=  
 $\psi(x_1, x_2, x_3)$



Factor Graphs

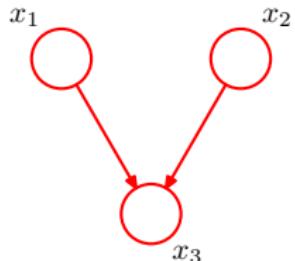
The Sum-Product  
Algorithm

Similar Algorithms

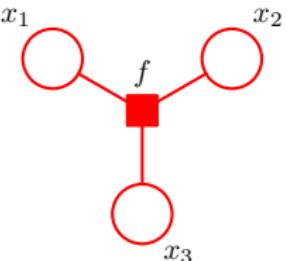
Learning the Graph  
Structure

# Factor Graphs - BN example

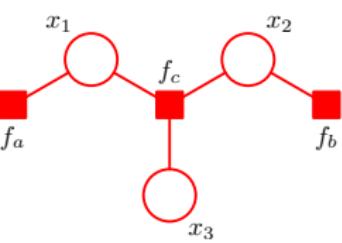
Example of factor graphs representing the same distribution



Directed graph  
 $p(x_1) p(x_2) p(x_3 | x_1, x_2)$



Factor graph  
 $f(x_1, x_2, x_3)$   
 $=$   
 $p(x_1) p(x_2) p(x_3 | x_1, x_2)$



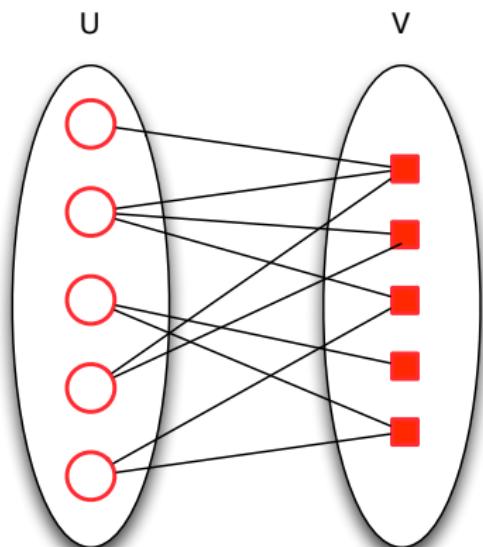
Factor graph  
factors satisfy  
 $f_a(x_1) = p(x_1)$   
 $f_b(x_2) = p(x_2)$   
 $f_c(x_1, x_2, x_3) =$   
 $p(x_3 | x_1, x_2)$

# Factor Graph - Bipartite Graph

- Factor Graphs are bipartite graphs.

## 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  $V$ .



Factor Graphs

The Sum-Product  
Algorithm

Similar Algorithms

Learning the Graph  
Structure



Factor Graphs

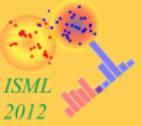
The Sum-Product  
Algorithm

Similar Algorithms

Learning the Graph  
Structure

- ➊ Create variable nodes for each node in the original graph.
- ➋ Create factor nodes corresponding to the maximal cliques  $\mathbf{x}_s$ .
- ➌ Set the factors  $f_s(\mathbf{x}_s)$  to the clique potentials.

Note: There may be several different factor graphs corresponding to the same undirected graph.



Factor Graphs

The Sum-Product  
Algorithm

Similar Algorithms

Learning the Graph  
Structure

# Bayesian Network → Factor Graph

- ➊ Create variable nodes for each node in the original graph.
- ➋ Create factor nodes corresponding to the conditional distributions.
- ➌ Add appropriate links.

Note: There may be several different factor graphs corresponding to the same directed graph.

# The Sum-Product Algorithm - Overview

- Assume a tree-structured factor graph.
- Try to find the marginal  $p(x)$  for a particular node  $x$ .  
(Assume here that all nodes are hidden.)

$$p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x})$$

- Key idea: Substitute for  $p(\mathbf{x})$  using the factor graph and then interchange summations and products in order to obtain an efficient algorithm.
- Partition the factors in the joint distribution into groups, with one group associated with each factor of the nodes that is a neighbour of the variable node  $x$ .

$$p(\mathbf{x}) = \prod_{s \in \text{ne}(x)} F_s(x, X_s)$$

where  $\text{ne}(x)$  denotes the set of factor nodes which are neighbours of  $x$ , and  $X_s$  denotes the set of all variables in the subtree connected to the variable node  $x$  via the factor node  $f_s$ , and  $F_s(x, X_s)$  represents the product of all the factors in the group associated with factor  $f_s$ .



Factor Graphs

The Sum-Product  
Algorithm

Similar Algorithms

Learning the Graph  
Structure



Factor Graphs

The Sum-Product  
Algorithm

Similar Algorithms

Learning the Graph  
Structure

# The Sum-Product Algorithm - In Detail

- Try to find the marginal  $p(x)$  for a particular node  $x$ .

$$p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x})$$

- Note, that  $x$  is an element of the set of variables  $\mathbf{x}$ ,  $x \in \mathbf{x}$ .



Factor Graphs

The Sum-Product  
Algorithm

Similar Algorithms

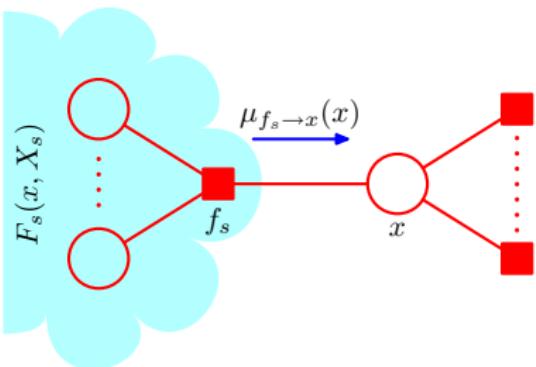
Learning the Graph  
Structure

# The Sum-Product Algorithm - In Detail

- The joint distribution  $p(\mathbf{x})$  can be written as a product

$$p(\mathbf{x}) = \prod_{s \in \text{ne}(x)} F_s(x, X_s)$$

- $\text{ne}(x)$  denotes the set of factor nodes that are neighbours of  $x$ .
- $X_s$  denotes the set of all variables in the subtree connected to the variable node  $x$  via the factor node.





Factor Graphs

The Sum-Product  
Algorithm

Similar Algorithms

Learning the Graph  
Structure

# The Sum-Product Algorithm - In Detail

- Goal: Marginal distribution  $p(x)$

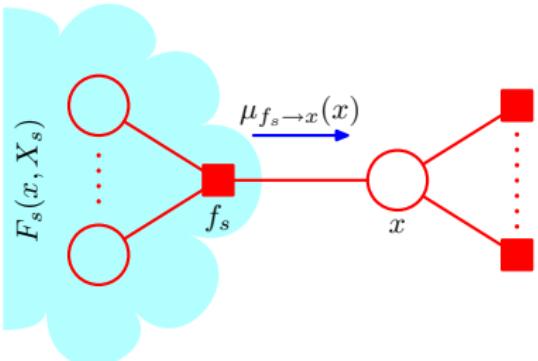
$$p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x})$$

- via joint distribution

$$p(\mathbf{x}) = \prod_{s \in \text{ne}(x)} F_s(x, X_s)$$

- resulting in

$$p(x) = \sum_{\mathbf{x} \setminus x} \prod_{s \in \text{ne}(x)} F_s(x, X_s) = \prod_{s \in \text{ne}(x)} \sum_{X_s} F_s(x, X_s)$$





Factor Graphs

The Sum-Product  
Algorithm

Similar Algorithms

Learning the Graph  
Structure

# Example for Exchanging Sum and Product

- Goal: Marginal distribution  $p(x)$

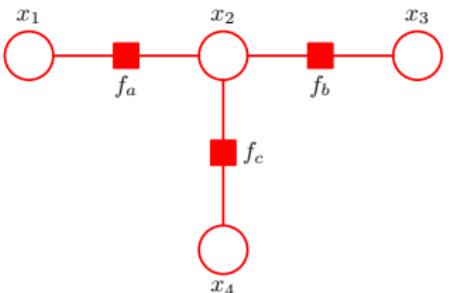
$$p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x})$$

$$p(x_2) = \sum_{\mathbf{x} \setminus x_2} p(\mathbf{x})$$

- via joint distribution

$$p(\mathbf{x}) = \prod_{s \in \text{ne}(x)} F_s(x, X_s)$$

$$p(x_1, x_2, x_3, x_4) = \prod_{s \in \text{ne}(x_2)} F_s(x_2, X_s) = f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4)$$





Factor Graphs

The Sum-Product  
Algorithm

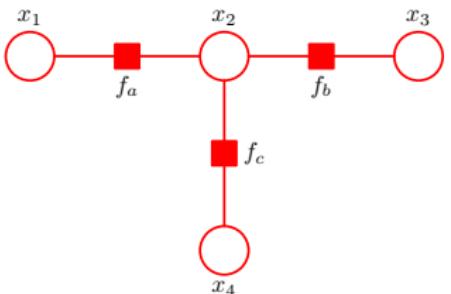
Similar Algorithms

Learning the Graph  
Structure

- Goal: Marginal distribution  $p(x)$   
(Note: Without normalisation  $Z = \sum_{x_2} p(x_2)$ )

$$p(x) = \sum_{\mathbf{x} \setminus x} \prod_{s \in \text{ne}(x)} F_s(x, X_s) = \prod_{s \in \text{ne}(x)} \sum_{X_s} F_s(x, X_s)$$

$$\begin{aligned} p(x_2) &= \sum_{x_1, x_3, x_4} f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4) \\ &= \left( \sum_{x_1} f_a(x_1, x_2) \right) \left( \sum_{x_3} f_b(x_2, x_3) \right) \left( \sum_{x_4} f_c(x_2, x_4) \right) \end{aligned}$$





Factor Graphs

The Sum-Product  
Algorithm

Similar Algorithms

Learning the Graph  
Structure

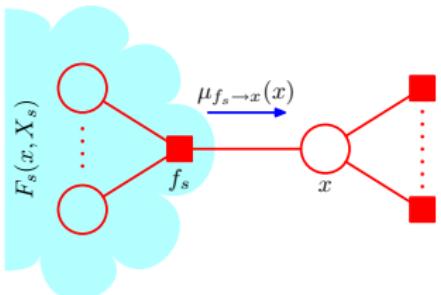
# The Sum-Product Algorithm - In Detail

- Goal: Marginal distribution  $p(x)$

$$\begin{aligned} p(x) &= \sum_{\mathbf{x} \setminus x} \prod_{s \in \text{ne}(x)} F_s(x, X_s) = \prod_{s \in \text{ne}(x)} \sum_{X_s} F_s(x, X_s) \\ &= \prod_{s \in \text{ne}(x)} \mu_{f_s \rightarrow x}(x) \end{aligned}$$

- with a set of functions which can be viewed as **messages**

$$\mu_{f_s \rightarrow x}(x) = \sum_{X_s} F_s(x, X_s).$$





Factor Graphs

The Sum-Product  
Algorithm

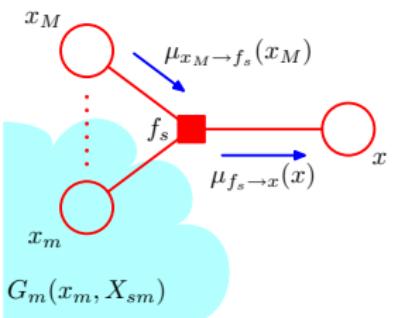
Similar Algorithms

Learning the Graph  
Structure

- Each factor  $F_s(x, X_s)$  consists of a subgraph and can therefore be written as

$$F_s(x, X_s) = f_s(x, x_1, \dots, x_M) G_1(x_1, X_{s1}) \dots G_M(x_M, X_{sM})$$

where  $\mathbf{x}_s = \{x, x_1, \dots, x_M\}$  is the set of variables on which the factor  $f_s$  depends.





Factor Graphs

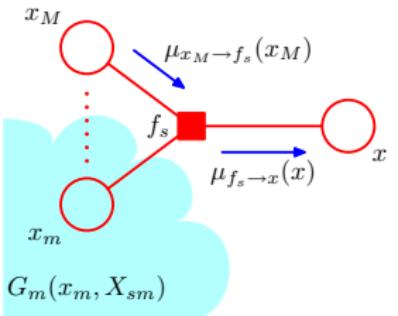
The Sum-Product  
Algorithm

Similar Algorithms

Learning the Graph  
Structure

- Using the factorisation of  $F_s(x, X_s)$ , the message  $\mu_{f_s \rightarrow x}(x)$  can be written as

$$\begin{aligned}\mu_{f_s \rightarrow x}(x) &= \sum_{X_s} F_s(x, X_s) \\ &= \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \dots, x_M) \prod_{m \in \text{ne}(f_s) \setminus x} \left[ \sum_{X_{sm}} G_m(x_m, X_{sm}) \right] \\ &= \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \dots, x_M) \prod_{m \in \text{ne}(f_s) \setminus x} \mu_{x_m \rightarrow f_s}(x_m).\end{aligned}$$





Factor Graphs

The Sum-Product  
Algorithm

Similar Algorithms

Learning the Graph  
Structure

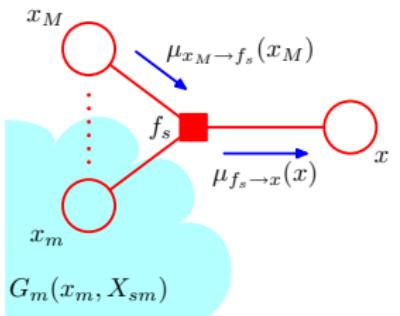
# Two kind of messages

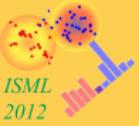
- Messages from factor nodes to variable nodes

$$\mu_{f_s \rightarrow x}(x) = \sum_{X_s} F_s(x, X_s)$$

- Messages from variable nodes to factor nodes

$$\mu_{x_m \rightarrow f_s}(x_m) = \sum_{X_{sm}} G_m(x_m, X_{sm})$$





Factor Graphs

The Sum-Product  
Algorithm

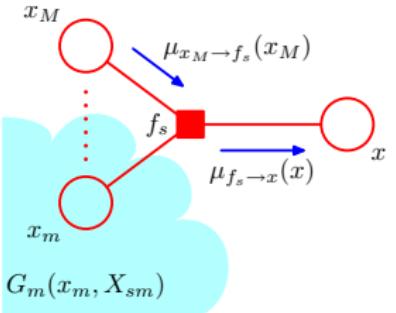
Similar Algorithms

Learning the Graph  
Structure

- We already have a formula to calculate messages from factor nodes to variable nodes

$$\mu_{f_s \rightarrow x}(x) = \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \dots, x_M) \prod_{m \in \text{ne}(f_s) \setminus x} \mu_{x_m \rightarrow f_s}(x_m)$$

- Take the product of all incoming messages, multiply with the factor associated with the node and marginalise over all variables associated with the incoming messages.



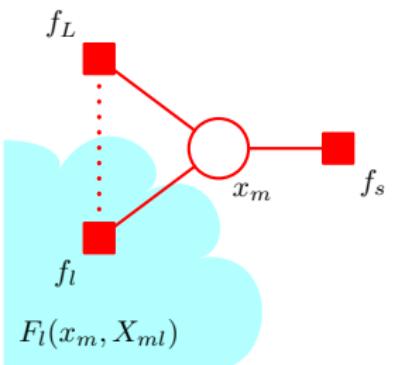


# Calculating messages variable → factor nodes

- $G_m(x_m, X_{sm})$  is a product of terms  $F_l(x_m, X_{ml})$

$$G_m(x_m, X_{sm}) = \prod_{l \in \text{ne}(x_m) \setminus f_s} F_l(x_m, X_{ml})$$

where the product is taken over all neighbours of node  $x_m$  except for node  $f_s$ .



*Factor Graphs*

*The Sum-Product  
Algorithm*

*Similar Algorithms*

*Learning the Graph  
Structure*



Factor Graphs

The Sum-Product  
Algorithm

Similar Algorithms

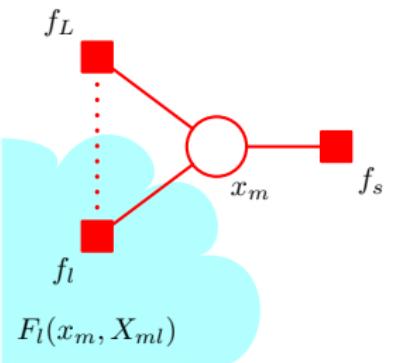
Learning the Graph  
Structure

# Calculating messages variable → factor nodes

- Therefore

$$\begin{aligned}\mu_{x_m \rightarrow f_s}(x_m) &= \prod_{l \in \text{ne}(x_m) \setminus f_s} \left[ \sum_{X_{ml}} F_l(x_m, X_{ml}) \right] \\ &= \prod_{l \in \text{ne}(x_m) \setminus f_s} \mu_{f_l \rightarrow x_m}(x_m).\end{aligned}$$

- Evaluate the message sent by a variable node to an adjacent factor node by taking the product of all incoming messages.





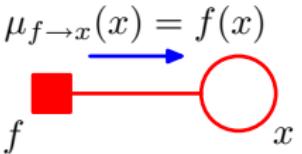
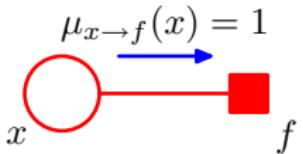
Factor Graphs

The Sum-Product  
Algorithm

Similar Algorithms

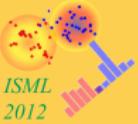
Learning the Graph  
Structure

- 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



- Normalisation : Calculate  $p(x)$  by message passing. Then

$$Z = \sum_x p(x)$$



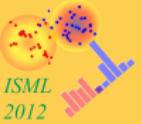
Factor Graphs

The Sum-Product  
Algorithm

Similar Algorithms

Learning the Graph  
Structure

- Brute force : Run the algorithm again for each node.
- More efficient
  - ➊ Arbitrarily choose one root in the graph.
  - ➋ Propagate all messages from leafs to root.
  - ➌ Now, root got all messages from its neighbours. Calculate marginal for root.
  - ➍ Root can now send messages to its neighbours.
  - ➎ Calculate their marginals and continue sending messages to the neighbours closer to the leafs.
- More efficient methods needs only twice as many computation to calculate marginals for all nodes than calculating marginal for one node.



Factor Graphs

The Sum-Product  
Algorithm

Similar Algorithms

Learning the Graph  
Structure

# Similar Algorithms

- **Max-Sum algorithm** : Find the values for the variables for which the probability has a maximum.
- **Junction Tree Algorithm** : Exact inference in general graphs. 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 Max-Sum algorithm 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.

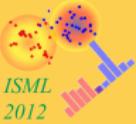
# Learning the Graph Structure

- Define a space of possible graph structures.
- Define a measure to score each of the graph structures.
- Bayesian viewpoint: Compute the posterior distribution over graph structures.
- If we have a prior  $p(m)$  over graphs indexed by  $m$ , the posterior is given via Bayes' theorem as

$$p(m | \mathcal{D}) \propto p(m) p(\mathcal{D} | m)$$

where  $\mathcal{D}$  is the observed data set, and the model evidence  $p(\mathcal{D} | m)$  provides the score for each model.

- Challenge 1: Evaluation of the model evidence involves marginalisation over the latent variables and is computationally very demanding.
- Challenge 2: The number of different graph structures grows exponentially with the number of nodes. Need to resort to heuristics to find good candidates.

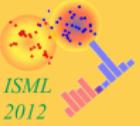


Factor Graphs

The Sum-Product  
Algorithm

Similar Algorithms

Learning the Graph  
Structure



## Part XVI

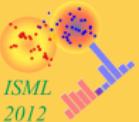
### *Mixture Models and EM 1*

*Motivation*

*K-means Clustering*

*K-means Applications*

*Mixture of Gaussians*



Motivation

K-means Clustering

K-means Applications

Mixture of Gaussians

- We studied how to define joint distributions over observed and latent variables (e.g. graphical models).
- Distribution over observed variables via marginalisation.
- Complex **marginal distributions** over observed variables can be expressed via more tractable **joint distributions** over the expanded space of observed and latent variables.
- Mixture Models can also be used to cluster data.
- General technique for finding maximum likelihood estimators in latent variable models:  
**expectation-maximisation (EM) algorithm.**
- Later: Variational Interference (Bayesian treatment).

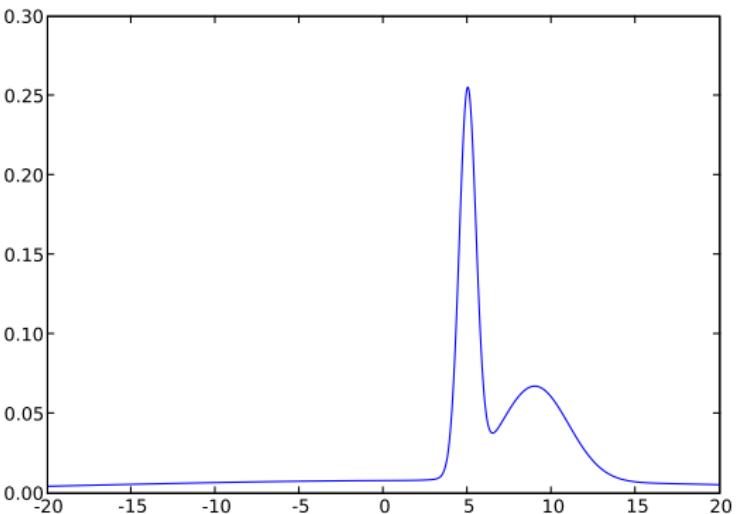


Motivation

K-means Clustering

K-means Applications

Mixture of Gaussians



# Example - Wallaby Distribution

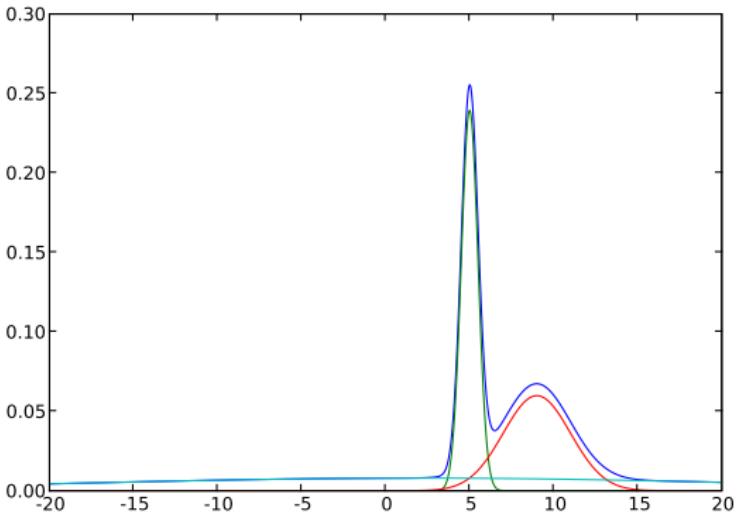
- Introduced very recently to show ...



# Example - 'Wallaby' Distribution

- ... that already a mixture of three Gaussian can be fun.

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

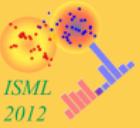


Motivation

K-means Clustering

K-means Applications

Mixture of Gaussians



## Motivation

## K-means Clustering

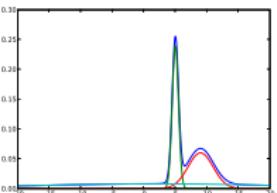
## K-means Applications

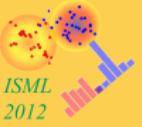
## Mixture of Gaussians

- Use  $\mu, \sigma$  as latent variables and define a distribution

$$p(\mu, \sigma) = \begin{cases} \frac{3}{10} & \text{if } (\mu, \sigma) = (5, 0.5) \\ \frac{3}{10} & \text{if } (\mu, \sigma) = (9, 2) \\ \frac{4}{10} & \text{if } (\mu, \sigma) = (2, 20) \\ 0 & \text{otherwise.} \end{cases}$$

$$\begin{aligned} p(x) &= \int_{-\infty}^{\infty} \int_0^{\infty} p(x, \mu, \sigma) d\mu d\sigma \\ &= \int_{-\infty}^{\infty} \int_0^{\infty} p(x | \mu, \sigma) p(\mu, \sigma) d\mu d\sigma \\ &= \frac{3}{10} \mathcal{N}(x | 5, 0.5) + \frac{3}{10} \mathcal{N}(x | 9, 2) + \frac{4}{10} \mathcal{N}(x | 2, 20) \end{aligned}$$





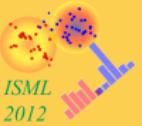
Motivation

K-means Clustering

K-means Applications

Mixture of Gaussians

- 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.
- 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.



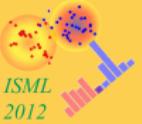
Motivation

K-means Clustering

K-means Applications

Mixture of Gaussians

- Start with arbitrary chosen prototypes  $\mu_k$ ,  $k = 1, \dots, K$ .
  - ➊ Assign each data point to the closest prototype.
  - ➋ Calculate new prototypes as the mean of all data points assigned to each of them.
- In the following, we will formalise this introducing a notation which will be useful later.



Motivation

K-means Clustering

K-means Applications

Mixture of Gaussians

# K-means Clustering - Notation

- 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**

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

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



Motivation

K-means Clustering

K-means Applications

Mixture of Gaussians

# K-means Clustering - Notation

- Find the values for  $\{r_{nk}\}$  and  $\{\mu_k\}$  so as to minimise  $J$ .
  - But  $\{r_{nk}\}$  depends on  $\{\mu_k\}$ , and  $\{\mu_k\}$  depends on  $\{r_{nk}\}$ .
  - Iterate until no further change
- ① Minimise  $J$  w.r.t.  $r_{nk}$  while keeping  $\{\mu_k\}$  fixed,

$$r_{nk} = \begin{cases} 1, & \text{if } k = \arg \min_j \|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2 \\ 0, & \text{otherwise.} \end{cases} \quad \forall n = 1, \dots, N$$

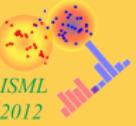
**Expectation step**

- ② Minimise  $J$  w.r.t.  $\{\mu_k\}$  while keeping  $r_{nk}$  fixed,

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

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^N r_{nk} \mathbf{x}_n}{\sum_{n=1}^N r_{nk}}$$

**Maximisation step**



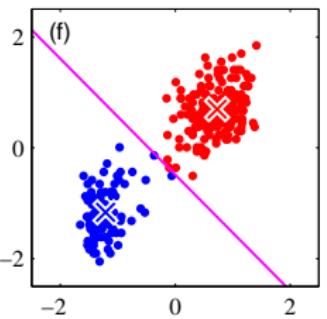
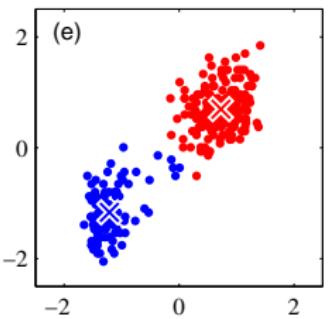
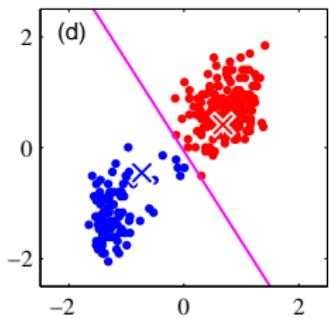
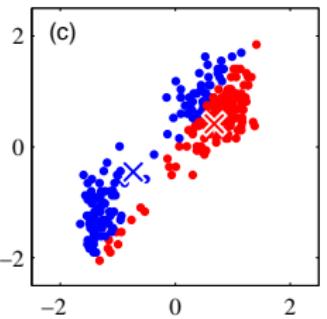
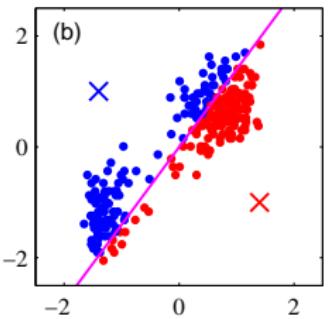
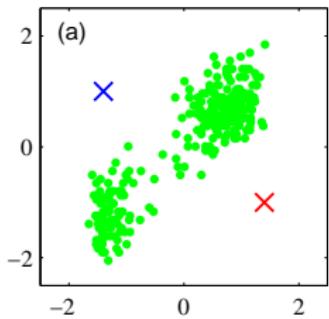
Motivation

K-means Clustering

K-means Applications

Mixture of Gaussians

# K-means Clustering - Example





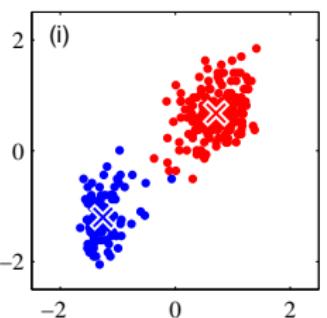
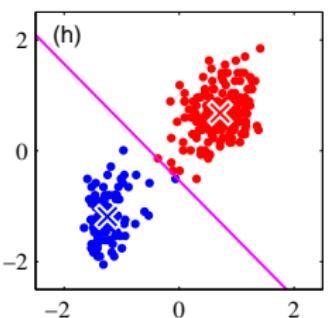
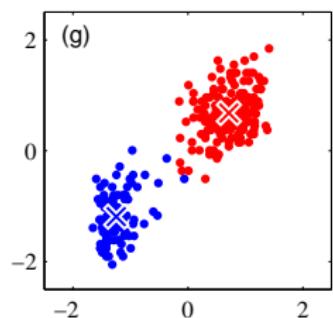
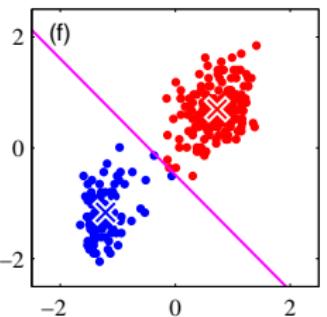
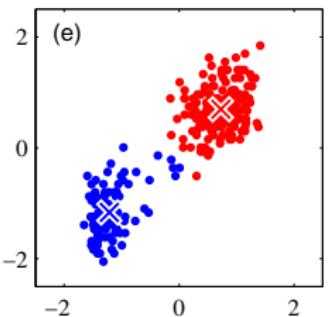
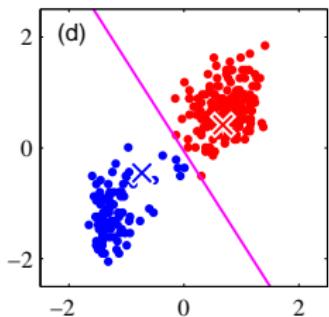
Motivation

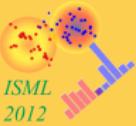
K-means Clustering

K-means Applications

Mixture of Gaussians

# K-means Clustering - Example



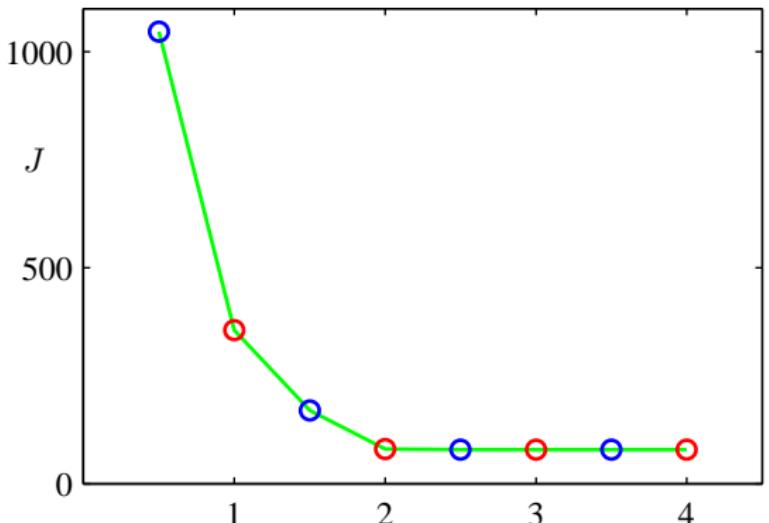


Motivation

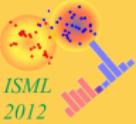
K-means Clustering

K-means Applications

Mixture of Gaussians



Cost function  $J$  after each E step (blue points)  
and M step (red points).



Motivation

K-means Clustering

K-means Applications

Mixture of Gaussians

# K-means Clustering - Notes

- 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; built K-D trees, ...)
- Generalise to non-Euclidian 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  $\eta_n$

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

# K-means Clustering - Image Segmentation



- 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 Euclidian).
- Run  $K$ -means on all points of the image until converge. 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.

Motivation

*K-means Clustering*

*K-means Applications*

*Mixture of Gaussians*

# Illustrating K-means Clustering - Segmentation



$K = 2$



$K = 10$



$K = 3$



Original image



Motivation

K-means Clustering

K-means Applications

Mixture of Gaussians

# Illustrating K-means Clustering - Segmentation

Introduction to Statistical  
Machine Learning

©2012

Christfried Webers  
NICTA

The Australian National  
University



Motivation

K-means Clustering

K-means Applications

Mixture of Gaussians

# K-means Clustering - Compression



Motivation

K-means Clustering

K-means Applications

Mixture of Gaussians

- Lossy data compression: accept some errors in the reconstruction as trade-off for higher compression.
- Apply  $K$ -means to the data.
- Store the code-book vectors  $\mu_k$ .
- Store the data in the form of references (labels) to the code-book. Each data points 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.



# Illustrating K-means Clustering - Compression

$K = 2$



4.2%

$K = 3$



8.3%

$K = 10$



16.7%

Original image



100 %

Motivation

K-means Clustering

K-means Applications

Mixture of Gaussians



Motivation

K-means Clustering

K-means Applications

Mixture of Gaussians

# Mixture of Gaussians

- A Gaussian mixture distribution is a linear superposition of Gaussians of the form

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

- As  $\int p(\mathbf{x}) d\mathbf{x} = 1$ , it follows  $\sum_{k=1}^K \pi_k = 1$ .
- Let us write this with the help of a latent variable  $\mathbf{z}$ .

## Definition (Latent variables)

Latent variables (as opposed to observable variables), are variables that are not directly observed but are rather inferred (through a mathematical model) from other variables that are observed and directly measured. They are also sometimes called hidden variables, model parameters, or hypothetical variables.



Motivation

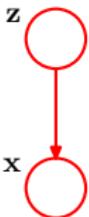
K-means Clustering

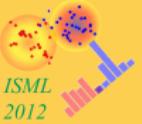
K-means Applications

Mixture of Gaussians

- Let  $\mathbf{z} \in \{0, 1\}^K$  and  $\sum_{k=1}^K z_k = 1$ . In words,  $\mathbf{z}$  is a  $K$ -dimensional vector in 1-of- $K$  representation.
- There are exactly  $K$  different possible vectors  $\mathbf{z}$  depending on which of the  $K$  entries is 1.
- Define the joint distribution  $p(\mathbf{x}, \mathbf{z})$  in terms of a marginal distribution  $p(\mathbf{z})$  and a conditional distribution  $p(\mathbf{x} | \mathbf{z})$  as

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z}) p(\mathbf{x} | \mathbf{z})$$





Motivation

K-means Clustering

K-means Applications

Mixture of Gaussians

# Mixture of Gaussians

- Set the marginal distribution to

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

where  $0 \leq \pi_k \leq 1$  together with  $\sum_{k=1}^K \pi_k = 1$ .

- Because  $\mathbf{z}$  uses 1-of- $K$  coding, we can also write

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

- Set the conditional distribution of  $\mathbf{x}$  given a particular  $\mathbf{z}$  to

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

or

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



Motivation

K-means Clustering

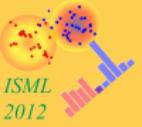
K-means Applications

Mixture of Gaussians

- The marginal distribution over  $\mathbf{x}$  is now found by summing the joint distribution over all possible states of  $\mathbf{z}$

$$\begin{aligned} p(\mathbf{x}) &= \sum_{\mathbf{z}} p(\mathbf{z}) p(\mathbf{x} | \mathbf{z}) = \sum_{\mathbf{z}} \prod_{k=1}^K \pi_k^{z_k} \prod_{k=1}^K \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k} \\ &= \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \end{aligned}$$

- The marginal distribution of  $\mathbf{x}$  is a Gaussian mixture.
- For several observations  $\mathbf{x}_1, \dots, \mathbf{x}_N$  we need one latent variable  $\mathbf{z}_n$  per observation.
- What have we gained? Can now work with the joint distribution  $p(\mathbf{x}, \mathbf{z})$ . Will lead to significant simplification later, especially for EM algorithm.



Motivation

K-means Clustering

K-means Applications

Mixture of Gaussians

# Mixture of Gaussians

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

$$\begin{aligned}\gamma(z_k) = p(z_k = 1 \mid \mathbf{x}) &= \frac{p(z_k = 1)p(\mathbf{x} \mid z_k = 1)}{\sum_{j=1}^K p(z_j = 1)p(\mathbf{x} \mid z_j = 1)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}\end{aligned}$$

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



Motivation

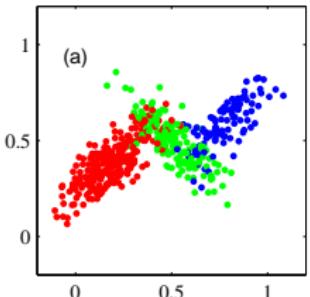
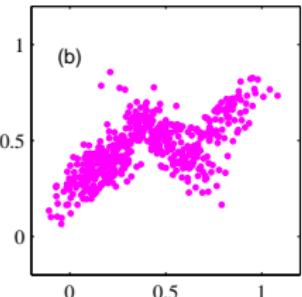
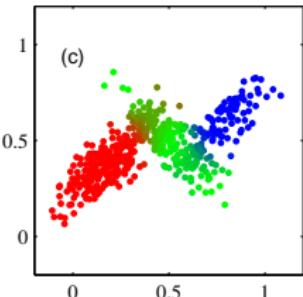
K-means Clustering

K-means Applications

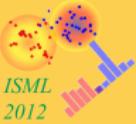
Mixture of Gaussians

# Mixture of Gaussians - Ancestral Sampling

- Goal: Generate random samples distributed according to the mixture model.
  - Generate a sample  $\hat{\mathbf{z}}$  from the distribution  $p(\mathbf{z})$ .
  - Generate a value  $\hat{\mathbf{x}}$  from the conditional distribution  $p(\mathbf{x} | \hat{\mathbf{z}})$ .
- Example: Mixture of 3 Gaussians, 500 points.

Original states of  $\mathbf{z}$ .Marginal  $p(\mathbf{x})$ .

(R, G, B) - colours  
mixed according to  
 $\gamma(z_{nk})$ .



Motivation

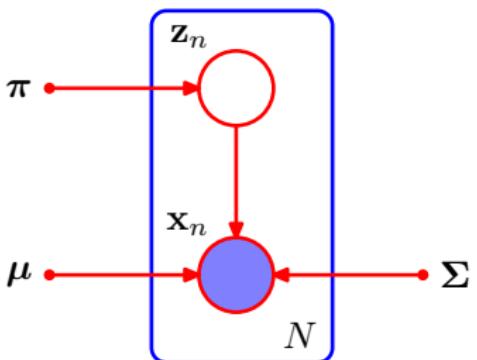
K-means Clustering

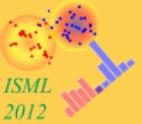
K-means Applications

Mixture of Gaussians

# Mixture of Gaussians - Maximum Likelihood

- Given  $N$  data points, each of dimension  $D$ , we have the data matrix  $\mathbf{X} \in \mathbb{R}^{N \times D}$  where each row contains one data point.
- Similarly, we have the matrix of latent variables  $\mathbf{Z} \in \mathbb{R}^{N \times K}$  with rows  $\mathbf{z}_n^T$ .
- Assume the data are drawn i.i.d., the distribution for the data can be represented by a graphical model.





Motivation

K-means Clustering

K-means Applications

Mixture of Gaussians

# Mixture of Gaussians - Maximum Likelihood

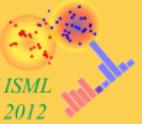
- The log of the likelihood function is then

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

- Significant problem: If a mean  $\boldsymbol{\mu}_j$  'sits' directly on a data point  $\mathbf{x}_n$  then

$$\mathcal{N}(\mathbf{x}_n | \mathbf{x}_n, \sigma_j^2 \mathbf{I}) = \frac{1}{(2\pi)^{1/2}} \frac{1}{\sigma_j}.$$

- Here we assumed  $\boldsymbol{\Sigma}_k = \sigma_k^2 \mathbf{I}$ . But problem is general, just think of a main axis transformation for  $\boldsymbol{\Sigma}_k$ .
- Overfitting (in disguise) occurring again with the maximum likelihood approach.
- Use heuristics to detect this situation and reset the mean of the corresponding component of the mixture.



Motivation

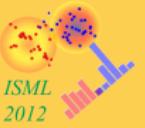
K-means Clustering

K-means Applications

Mixture of Gaussians

# Mixture of Gaussians - Maximum Likelihood

- 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.
- 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.



## Part XVII

### *Mixture Models and EM 2*

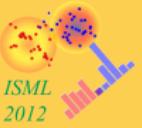
*EM for Gaussian  
Mixtures*

*EM for Gaussian  
Mixtures - Relation to  
K-Means*

*Mixture of Bernoulli  
Distributions*

*EM for Gaussian  
Mixtures - Latent  
Variables*

*Convergence of EM*



# EM for Gaussian Mixtures

- Starting point is the log of the likelihood function

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

- Critical point of  $\ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$  w.r.t.  $\boldsymbol{\mu}_k$

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

- Therefore

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

where the effective number of points assigned to Gaussian  $k$  is  $N_k = \sum_{n=1}^N \gamma(z_{nk})$ .

EM for Gaussian  
Mixtures

EM for Gaussian  
Mixtures - Relation to  
K-Means

Mixture of Bernoulli  
Distributions

EM for Gaussian  
Mixtures - Latent  
Variables

Convergence of EM

EM for Gaussian  
MixturesEM for Gaussian  
Mixtures - Relation to  
K-MeansMixture of Bernoulli  
DistributionsEM for Gaussian  
Mixtures - Latent  
Variables

Convergence of EM

# EM for Gaussian Mixtures

- Maximum of the log of the likelihood function for

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

- Similarly for the covariance matrix

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

- and for the mixing coefficients  $\pi_k$  (using a Lagrange multiplier as  $\sum_k \pi_k = 1$ )

$$\pi_k = \frac{N_k}{N}.$$

- This is not a closed form solution because the responsibilities  $\gamma(z_{nk})$  depend on  $\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}$ .



# EM for Gaussian Mixtures

- Given a Gaussian mixture and data  $\mathbf{X}$ , maximise the log likelihood w.r.t. the parameters  $(\pi, \mu, \Sigma)$ .
  - Initialise the means  $\mu_k$ , covariances  $\Sigma_k$  and mixing coefficients  $\pi_k$ . Evaluate the log likelihood function.
  - E step** : Evaluate the  $\gamma(z_k)$  using the current parameters

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

- M step** : Re-estimate the parameters using the current  $\gamma(z_k)$

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

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

- Evaluate the log likelihood, if not converged then goto 2.

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

EM for Gaussian  
MixturesEM for Gaussian  
Mixtures - Relation to  
K-MeansMixture of Bernoulli  
DistributionsEM for Gaussian  
Mixtures - Latent  
Variables

Convergence of EM



EM for Gaussian  
Mixtures

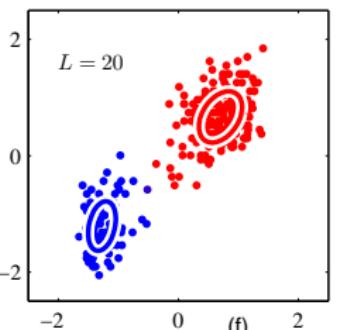
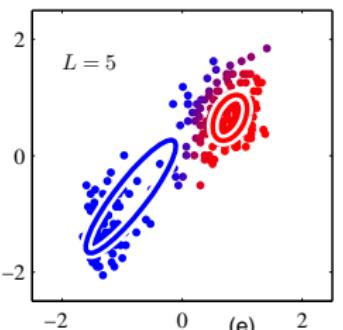
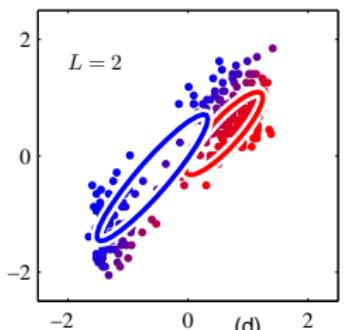
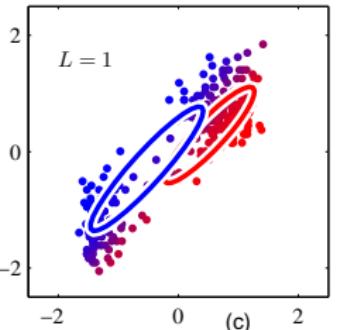
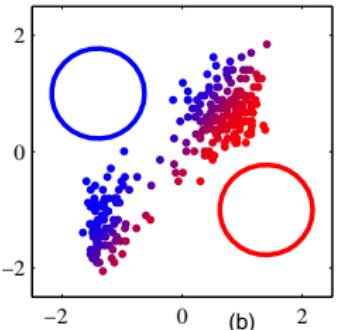
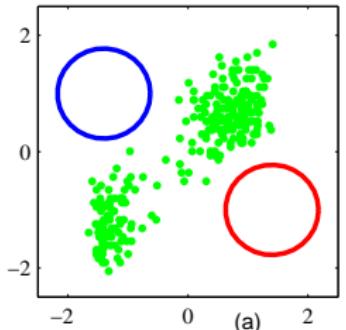
EM for Gaussian  
Mixtures - Relation to  
K-Means

Mixture of Bernoulli  
Distributions

EM for Gaussian  
Mixtures - Latent  
Variables

Convergence of EM

# EM for Gaussian Mixtures - Example



# EM for Gaussian Mixtures - Relation to K-Means



- Assume a Gaussian mixture model.
- Covariance matrices given by  $\epsilon \mathbf{I}$ , where  $\epsilon$  is shared by all components.
- Then

$$p(\mathbf{x} | \boldsymbol{\mu}_k, \Sigma_k) = \frac{1}{(2\pi\epsilon)^M/2} \exp \left\{ -\frac{1}{2\epsilon} \|\mathbf{x} - \boldsymbol{\mu}_k\|^2 \right\}.$$

- Keep  $\epsilon$  fixed, do not re-estimate.
- Responsibilities

$$\gamma(z_{nk}) = \frac{\pi_k \exp \left\{ -\|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2 / 2\epsilon \right\}}{\sum_j \pi_j \exp \left\{ -\|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2 / 2\epsilon \right\}}$$

- Taking the limit  $\epsilon \rightarrow 0$ , the term in the denominator for which  $\|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2$  is the smallest will go to zero most slowly.

EM for Gaussian  
Mixtures

EM for Gaussian  
Mixtures - Relation to  
K-Means

Mixture of Bernoulli  
Distributions

EM for Gaussian  
Mixtures - Latent  
Variables

Convergence of EM

EM for Gaussian  
MixturesEM for Gaussian  
Mixtures - Relation to  
K-MeansMixture of Bernoulli  
DistributionsEM for Gaussian  
Mixtures - Latent  
Variables

Convergence of EM

# EM for Gaussian Mixtures - Relation to K-Means

- Assume a Gaussian mixture model.

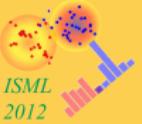
$$\gamma(z_{nk}) = \frac{\pi_k \exp\left\{-\|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2/2\epsilon\right\}}{\sum_j \pi_j \exp\left\{-\|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2/2\epsilon\right\}}$$

- Therefore

$$\gamma(z_{nk}) = \begin{cases} 1 & \text{if } \|\mathbf{x}_n - \boldsymbol{\mu}_k\| < \|\mathbf{x}_n - \boldsymbol{\mu}_j\| \quad \forall j \neq k \\ 0 & \text{otherwise} \end{cases}$$

- Holds independent of  $\pi_k$  as long as none is zero.
- Hard assignment to exactly one cluster : K-means.

$$\lim_{\epsilon \rightarrow 0} \gamma(z_{nk}) = r_{nk}$$

EM for Gaussian  
MixturesEM for Gaussian  
Mixtures - Relation to  
K-MeansMixture of Bernoulli  
DistributionsEM for Gaussian  
Mixtures - Latent  
Variables

Convergence of EM

# Mixture of Bernoulli Distributions

- Set of  $D$  binary variables  $x_i, i = 1, \dots, D$ .
- Each governed by a Bernoulli distribution with parameter  $\mu_i$ . Therefore

$$p(\mathbf{x} | \boldsymbol{\mu}) = \prod_{i=1}^D \mu_i^{x_i} (1 - \mu_i)^{1-x_i}$$

- Expectation and covariance

$$\mathbb{E} [\mathbf{x}] = \boldsymbol{\mu}$$

$$\text{cov}[\mathbf{x}] = \text{diag}\{\mu_i(1 - \mu_i)\}$$



# Mixture of Bernoulli Distributions

- Mixture

$$p(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\pi}) = \sum_{k=1}^K \pi_k p(\mathbf{x} \mid \boldsymbol{\mu}_k)$$

with

$$p(\mathbf{x} \mid \boldsymbol{\mu}_k) = \prod_{i=1}^D \mu_{ki}^{x_i} (1 - \mu_{ki})^{1-x_i}$$

- Similar calculation as with mixture of Gaussian

$$\gamma(z_{nk}) = \frac{\pi_k p(\mathbf{x}_n \mid \boldsymbol{\mu}_k)}{\sum_{j=1}^K \pi_j p(\mathbf{x}_n \mid \boldsymbol{\mu}_j)}$$

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

$$\bar{\mathbf{x}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n \quad \mu_k = \bar{\mathbf{x}}$$

$$\pi_k = \frac{N_k}{N}$$

EM for Gaussian  
Mixtures

EM for Gaussian  
Mixtures - Relation to  
K-Means

Mixture of Bernoulli  
Distributions

EM for Gaussian  
Mixtures - Latent  
Variables

Convergence of EM

# EM for Mixture of Bernoulli Distributions - Digits



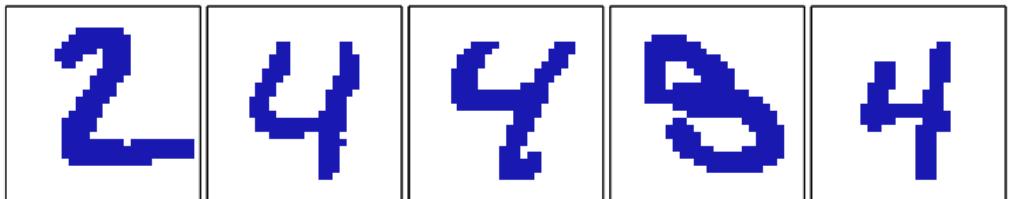
EM for Gaussian  
Mixtures

EM for Gaussian  
Mixtures - Relation to  
K-Means

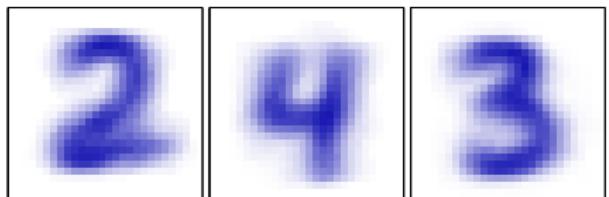
Mixture of Bernoulli  
Distributions

EM for Gaussian  
Mixtures - Latent  
Variables

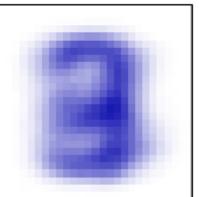
Convergence of EM



Examples from a digits data set, each pixel taken only binary  
values.



Parameters  $\mu_{ki}$  for each  
component in the mixture.



Fit to one multivariate  
Bernoulli distribution.

# The Role of Latent Variables

- EM finds the maximum likelihood solution for models with latent variables.
- Two kinds of variables
  - Observed variables  $\mathbf{X}$
  - Latent variables  $\mathbf{Z}$
- plus model parameters  $\theta$ .
- Log likelihood is then

$$\ln p(\mathbf{X} | \theta) = \ln \left\{ \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} | \theta) \right\}$$

- Optimisation problem due to the log-sum.
- Assume maximisation of the distribution  $p(\mathbf{X}, \mathbf{Z} | \theta)$  over the **complete data set**  $\{\mathbf{X}, \mathbf{Z}\}$  is straightforward.
- But we only have the **incomplete data set**  $\{\mathbf{X}\}$  and the posterior distribution  $p(\mathbf{Z} | \mathbf{X}, \theta)$ .

EM for Gaussian  
MixturesEM for Gaussian  
Mixtures - Relation to  
K-MeansMixture of Bernoulli  
DistributionsEM for Gaussian  
Mixtures - Latent  
Variables

Convergence of EM



EM for Gaussian  
Mixtures

EM for Gaussian  
Mixtures - Relation to  
K-Means

Mixture of Bernoulli  
Distributions

EM for Gaussian  
Mixtures - Latent  
Variables

Convergence of EM

# EM - Key Idea

- Key idea of EM: As  $\mathbf{Z}$  is not observed, work with an ‘averaged’ version  $Q(\theta, \theta^{\text{old}})$  of the complete log-likelihood  $\ln p(\mathbf{X}, \mathbf{Z} | \theta)$ , averaged over all states of  $\mathbf{Z}$ .

$$Q(\theta, \theta^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z} | \mathbf{X}, \theta^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z} | \theta)$$

EM for Gaussian  
MixturesEM for Gaussian  
Mixtures - Relation to  
K-MeansMixture of Bernoulli  
DistributionsEM for Gaussian  
Mixtures - Latent  
Variables

Convergence of EM

# EM Algorithm

- ➊ Choose an initial setting for the parameters  $\theta^{\text{old}}$ .
- ➋ E step Evaluate  $p(\mathbf{Z} | \mathbf{X}, \theta^{\text{old}})$ .
- ➌ M step Evaluate  $\theta^{\text{new}}$  given by

$$\theta^{\text{new}} = \arg \max_{\theta} Q(\theta, \theta^{\text{old}})$$

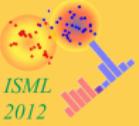
where

$$Q(\theta, \theta^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z} | \mathbf{X}, \theta^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z} | \theta)$$

- ➍ Check for convergence of log likelihood or parameter values. If not yet converged, then

$$\theta^{\text{old}} = \theta^{\text{new}}$$

and go to step 2.

EM for Gaussian  
MixturesEM for Gaussian  
Mixtures - Relation to  
K-MeansMixture of Bernoulli  
DistributionsEM for Gaussian  
Mixtures - Latent  
Variables

Convergence of EM

# EM Algorithm - Convergence

- Start with the product rule for the observed variables  $\mathbf{x}$ , the unobserved variables  $\mathbf{Z}$ , and the parameters  $\theta$

$$\ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta}) = \ln p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}) + \ln p(\mathbf{X} | \boldsymbol{\theta}).$$

- Apply  $\sum_{\mathbf{Z}} q(\mathbf{Z})$  with arbitrary  $q(\mathbf{Z})$  to the formula

$$\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta}) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}) + \ln p(\mathbf{X} | \boldsymbol{\theta}).$$

- Rewrite as

$$\ln p(\mathbf{X} | \boldsymbol{\theta}) = \underbrace{\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{q(\mathbf{Z})}}_{\mathcal{L}(q, \boldsymbol{\theta})} - \underbrace{\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})}}_{\text{KL}(q \| p)}$$

- $\text{KL}(q \| p)$  is the Kullback-Leibler divergence.

# Kullback-Leibler Divergence

©2012

Christfried Webers  
NICTAThe Australian National  
University

- ‘Distance’ between two distributions  $p(y)$  and  $q(y)$

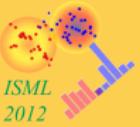
$$\text{KL}(q\|p) = \sum_y q(y) \ln \frac{q(y)}{p(y)} = - \sum_y q(y) \ln \frac{p(y)}{q(y)}$$

$$\text{KL}(q\|p) = \int q(y) \ln \frac{q(y)}{p(y)} \, dy = - \int q(y) \ln \frac{p(y)}{q(y)} \, dy$$

- $\text{KL}(q\|p) \geq 0$
- not symmetric:  $\text{KL}(q\|p) \neq \text{KL}(p\|q)$
- $\text{KL}(q\|p) = 0$  iff  $q = p$ .
- invariant under parameter transformations

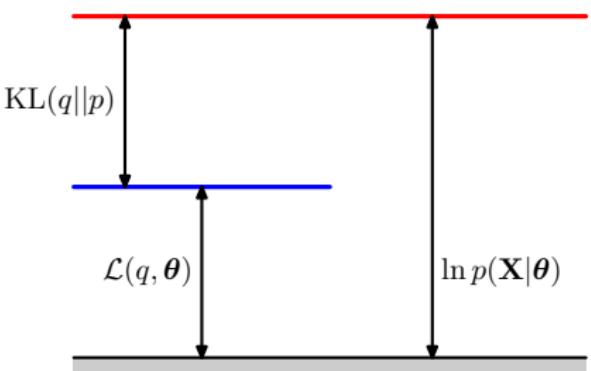
EM for Gaussian  
MixturesEM for Gaussian  
Mixtures - Relation to  
K-MeansMixture of Bernoulli  
DistributionsEM for Gaussian  
Mixtures - Latent  
Variables

Convergence of EM



- The two parts of  $\ln p(\mathbf{X} | \boldsymbol{\theta})$

$$\ln p(\mathbf{X} | \boldsymbol{\theta}) = \underbrace{\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{q(\mathbf{Z})}}_{\mathcal{L}(q, \boldsymbol{\theta})} - \underbrace{\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})}}_{\text{KL}(q \| p)}$$

EM for Gaussian  
MixturesEM for Gaussian  
Mixtures - Relation to  
K-MeansMixture of Bernoulli  
DistributionsEM for Gaussian  
Mixtures - Latent  
Variables

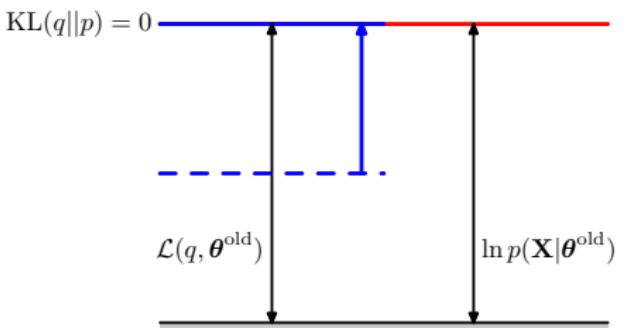
Convergence of EM



# EM Algorithm - E Step

- Hold  $\theta^{\text{old}}$  fixed. Maximise the lower bound  $\mathcal{L}(q, \theta^{\text{old}})$  with respect to  $q(\cdot)$ .
- $\mathcal{L}(q, \theta^{\text{old}})$  is a functional.
- $\ln p(\mathbf{X} | \theta)$  does NOT depend on  $q(\cdot)$ .
- Maximum for  $\mathcal{L}(q, \theta^{\text{old}})$  will occur when the Kullback-Leibler divergence vanishes.
- Therefore, choose  $q(\mathbf{Z}) = p(\mathbf{Z} | \mathbf{X}, \theta^{\text{old}})$

$$\ln p(\mathbf{X} | \theta) = \underbrace{\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{X}, \mathbf{Z} | \theta)}{q(\mathbf{Z})}}_{\mathcal{L}(q, \theta)} - \underbrace{\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{Z} | \mathbf{X}, \theta)}{q(\mathbf{Z})}}_{\text{KL}(q || p)}$$

EM for Gaussian  
MixturesEM for Gaussian  
Mixtures - Relation to  
K-MeansMixture of Bernoulli  
DistributionsEM for Gaussian  
Mixtures - Latent  
Variables

Convergence of EM



EM for Gaussian  
Mixtures

EM for Gaussian  
Mixtures - Relation to  
K-Means

Mixture of Bernoulli  
Distributions

EM for Gaussian  
Mixtures - Latent  
Variables

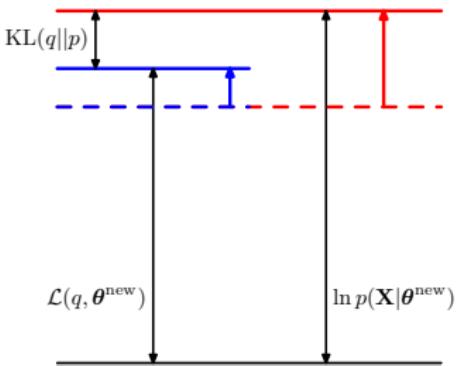
Convergence of EM

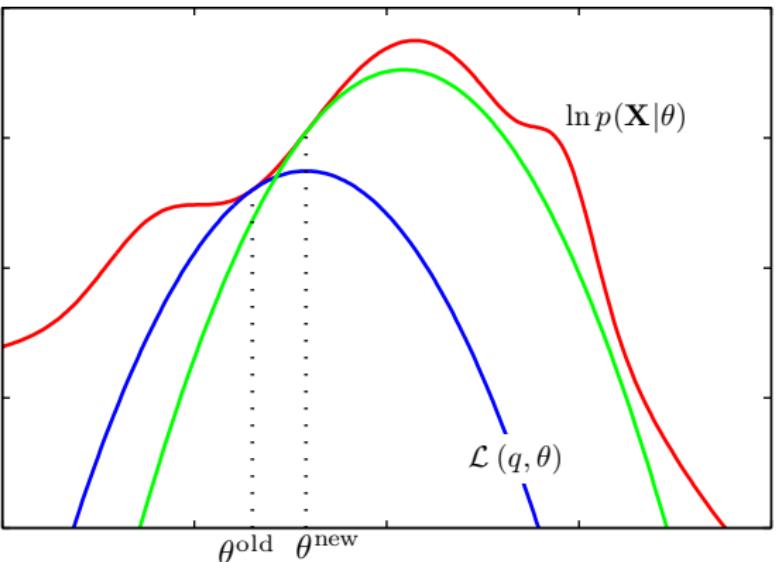
# EM Algorithm - M Step

- Hold  $q(\cdot) = p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}^{\text{old}})$  fixed. Maximise the lower bound  $\mathcal{L}(q, \boldsymbol{\theta})$  with respect to  $\boldsymbol{\theta}$  :  

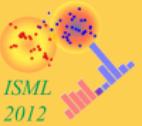
$$\boldsymbol{\theta}^{\text{new}} = \arg \max_{\boldsymbol{\theta}} \mathcal{L}(q, \boldsymbol{\theta}^{\text{old}}) = \arg \max_{\boldsymbol{\theta}} \sum_{\mathbf{Z}} q(\cdot) \ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})$$
- $\mathcal{L}(q, \boldsymbol{\theta}^{\text{new}}) > \mathcal{L}(q, \boldsymbol{\theta}^{\text{old}})$  unless maximum already reached.
- As  $q(\cdot) = p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}^{\text{old}})$  is fixed,  $p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}^{\text{new}})$  will not be equal to  $q(\cdot)$ , and therefore the Kullback-Leiber distance will be greater than zero (unless converged).

$$\ln p(\mathbf{X} | \boldsymbol{\theta}) = \underbrace{\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{q(\mathbf{Z})}}_{\mathcal{L}(q, \boldsymbol{\theta})} - \underbrace{\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})}}_{\text{KL}(q || p)}$$





Red curve : incomplete data likelihood.  
Blue curve : After E step. Green curve : After M step.



# Part XVIII

## *Approximate Inference*

*Approximate Inference*

*Approximation Schemes*

*Variational Optimisation*

*Calculus of Variation*

*Variational Optimisation  
applied to Inference*

*Exponential Family*

*Expectation Propagation*



Approximate Inference

Approximation Schemes

Variational Optimisation

Calculus of Variation

Variational Optimisation  
applied to Inference

Exponential Family

Expectation Propagation

- **Inference** : Drawing conclusions about a population from a random sample drawn from it, or, more generally, about a random process from its observed behavior during a finite period of time.
- **Descriptive Statistics** describes the main features of a data set in quantitative terms.
- **Inductive Statistics** : Hypothesis testing.
- Fitting a model to some observed data (finding the parameters of the model) is inference.



- Central task in the application of probabilistic models is the evaluation of the posterior distribution  $p(\mathbf{Z} | \mathbf{X})$  of the latent variables  $\mathbf{Z}$  given the observed variables  $\mathbf{X}$ .
- This may be infeasible because
  - Posterior distribution has a too complex form for which expectations are not tractable.
  - Integration (for continuous variables) may not have a closed form solution.
  - Numerical integration (continuous variables) or sums over all possible configuration (discrete variables). Impossible if too many variables.
- Need approximations.

Approximate Inference

Approximation Schemes

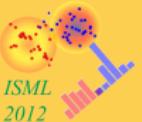
Variational Optimisation

Calculus of Variation

Variational Optimisation  
applied to Inference

Exponential Family

Expectation Propagation



Approximate Inference

Approximation Schemes

Variational Optimisation

Calculus of Variation

Variational Optimisation  
applied to Inference

Exponential Family

Expectation Propagation

# Approximation Schemes

## • Stochastic Approximation

- Produce exact results if given enough computational resources.
- However, sampling methods can be computational demanding.
- Example Markov Chain Monte Carlo (next lecture).

## • Deterministic Approximation:

- Based on analytical approximation of the posterior distribution  $p(\mathbf{Z} | \mathbf{X})$ .
- Example: posterior is assumed to factorise in a particular way.
- However, these simplifying assumptions can never produce exact results for the original problem.

# Variational Optimisation

- Originates from the **calculus of variations** (Euler, Lagrange, and others)
- Standard calculus uses a function to map one value to another value

$$y = f(x) \qquad x \xrightarrow{f} y$$

- A **functional** maps a function to a value.  
Example: Entropy  $H[p]$

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

- Functional derivative: How does the value change for infinitesimal changes of the input function.
- A large number of problems have the form: Find a function which optimises (maximises/minimises) a functional.
- Approximate solutions can be found by restricting the set of functions over which the functional will be optimised.



Approximate Inference

Approximation Schemes

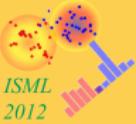
Variational Optimisation

Calculus of Variation

Variational Optimisation  
applied to Inference

Exponential Family

Expectation Propagation



Approximate Inference

Approximation Schemes

Variational Optimisation

Calculus of Variation

Variational Optimisation  
applied to Inference

Exponential Family

Expectation Propagation

# Calculus of Variation

- Given a functional of the form

$$F[y] = \int_a^b G(y(x), y'(x), x) dx$$

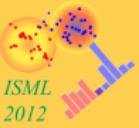
with  $f(a) = f_a$  and  $f(b) = f_b$  fixed, find a function  $y(\cdot)$  which optimises the functional under the given constraints.

- Consider an infinitesimal change of the function  $y(\cdot)$  by  $\eta(x)$

$$\begin{aligned} F[y + \epsilon\eta] &= F[y] + \epsilon \int_a^b \left\{ \frac{\partial G}{\partial y} \eta(x) + \frac{\partial G}{\partial y'} \eta'(x) \right\} dx + O(\epsilon^2) \\ &= F[y] + \epsilon \int_a^b \left\{ \frac{\partial G}{\partial y} - \frac{d}{dx} \left( \frac{\partial G}{\partial y'} \right) \right\} \eta(x) dx + O(\epsilon^2) \end{aligned}$$

- Euler-Lagrange equations

$$\frac{\partial G}{\partial y} - \frac{d}{dx} \left( \frac{\partial G}{\partial y'} \right) = 0$$



Approximate Inference

Approximation Schemes

Variational Optimisation

Calculus of Variation

Variational Optimisation  
applied to Inference

Exponential Family

Expectation Propagation

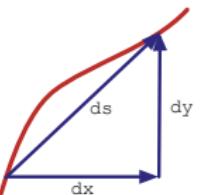
# Calculus of Variation - Shortest path

- Shortest path between two points in  $\mathbb{R}^N$  or on a manifold.
- In  $\mathbb{R}^N$ , the path length of a curve  $y(x)$  is

$$ds = \sqrt{dx^2 + dy^2} = dx \sqrt{1 + (dy/dx)^2}$$

$$L = \int_a^b \sqrt{1 + y'(x)^2} dx$$

and therefore  $G(y(x), y'(x), x) = \sqrt{1 + y'(x)^2}$



- Using the Euler-Lagrange equations

$$\frac{\partial G}{\partial y} - \frac{d}{dx} \left( \frac{\partial G}{\partial y'} \right) = 0$$

- we get

$$\frac{d}{dx} \sqrt{1 + y'(x)^2} = 0$$

and therefore  $y'(x) = \text{constant}$  : a straight line.



Approximate Inference

Approximation Schemes

Variational Optimisation

Calculus of Variation

Variational Optimisation  
applied to Inference

Exponential Family

Expectation Propagation

# Variational Optimisation applied to Inference

- Variational optimisation is exact, there is no approximation involved.
- But it can be used to find approximate solutions to a given problem by **restricting** the class of functions.
- Examples: consider only quadratic functions, or linear combinations of fixed basis functions with variable coefficients.
- Probabilistic inference: assume the probability functions **factorise** in a specific way.



Approximate Inference

Approximation Schemes

Variational Optimisation

Calculus of Variation

Variational Optimisation  
applied to Inference

Exponential Family

Expectation Propagation

# Variational Optimisation applied to Inference

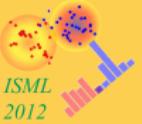
- From the lecture on EM

$$\ln p(\mathbf{X}) = \mathcal{L}(q) + \text{KL}(q\|p)$$

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right\} d\mathbf{Z}$$

$$\text{KL}(q\|p) = - \int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X})}{q(\mathbf{Z})} \right\} d\mathbf{Z}.$$

- Here  $\mathbf{Z}$  contains
  - latent variables (as before), and
  - parameters  $\theta$  which are now assumed to be stochastic (prior distribution over this variables provided).
- Given is the joint distribution  $p(\mathbf{X}, \mathbf{Z})$ .
- Find the posterior  $p(\mathbf{Z} | \mathbf{X})$  and the model evidence  $p(\mathbf{X})$ .



ISML  
2012

Approximate Inference

Approximation Schemes

Variational Optimisation

Calculus of Variation

Variational Optimisation  
applied to Inference

Exponential Family

Expectation Propagation

# Variational Optimisation applied to Inference

- From the lecture on EM

$$\ln p(\mathbf{X}) = \mathcal{L}(q) + \text{KL}(q\|p)$$

- Maximise the lower bound of  $\mathcal{L}(q)$  by optimisation.
- If all probability distributions  $p(\mathbf{Z})$  are allowed this will be achieved for  $\text{KL}(q\|p) = 0$ , or  $q(\mathbf{Z}) = p(\mathbf{Z} | \mathbf{X})$ .
- Assuming the true posterior is intractable, consider a restricted family  $q(\mathbf{Z})$  and find a member of this family which minimises the KL divergence.

# Factorised Distributions

©2012

Christfried Webers  
NICTAThe Australian National  
University

- Partition  $\mathbf{Z}$  into disjoint sets  $\mathbf{Z}_i$  where  $i = 1, \dots, M$ .
- Assume  $q(\mathbf{Z})$  factorises as

$$q(\mathbf{Z}) = \prod_{i=1}^M q_i(\mathbf{Z}_i)$$

- No restriction on the functional form of the individual  $q_i(\mathbf{Z}_i)$ .
- This factorisation for variational inference corresponds to **Mean Field Theory** in physics, replacing the interaction of  $n$  particles with a model of one particle and a mean field (created by all the other  $n - 1$  particles).

Approximate Inference

Approximation Schemes

Variational Optimisation

Calculus of Variation

Variational Optimisation  
applied to Inference

Exponential Family

Expectation Propagation



Approximate Inference

Approximation Schemes

Variational Optimisation

Calculus of Variation

Variational Optimisation  
applied to Inference

Exponential Family

Expectation Propagation

# Factorised Distributions

- Insert  $q(\mathbf{Z}) = \prod_{i=1}^M q_i(\mathbf{Z}_i)$  into

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right\} d\mathbf{Z}$$

- to get (using  $(q_j = q_j(\mathbf{Z}_j))$ )

$$\begin{aligned} \mathcal{L}(q) &= \int \prod_{i=1}^M q_i \left\{ \ln p(\mathbf{X}, \mathbf{Z}) - \sum_i \ln q_i \right\} d\mathbf{Z} \\ &= \int q_j \ln \tilde{p}(\mathbf{X}, \mathbf{Z}_j) d\mathbf{Z}_j - \int q_j \ln q_j d\mathbf{Z}_j - \text{const} \end{aligned}$$

where

$$\ln \tilde{p}(\mathbf{X}, \mathbf{Z}_j) - \text{const} = \int \ln p(\mathbf{X}, \mathbf{Z}) \prod_{i \neq j} q_i d\mathbf{Z}_i = \mathbb{E}_{i \neq j} [\ln p(\mathbf{X}, \mathbf{Z})]$$

- $\mathbb{E}_{i \neq j} [\ln p(\mathbf{X}, \mathbf{Z})]$  denotes an expectation with respect to the  $q$  distributions over all variables  $\mathbf{z}_i$  for  $i \neq j$ .



Approximate Inference

Approximation Schemes

Variational Optimisation

Calculus of Variation

Variational Optimisation  
applied to Inference

Exponential Family

Expectation Propagation

# Factorised Distributions

- Keep  $\{q_{i \neq j}\}$  fixed. Only  $q_j$  can vary.

$$\mathcal{L}(q) = \int q_j \ln \tilde{p}(\mathbf{X}, \mathbf{Z}_j) \, d\mathbf{Z}_j - \int q_j \ln q_j \, d\mathbf{Z}_j - \text{const}$$

- But  $\mathcal{L}(q) + \text{const}$  can now be written as a negative Kullback-Leibler divergence

$$\mathcal{L}(q) + \text{const} = \int q_j \ln \left\{ \frac{\ln \tilde{p}(\mathbf{X}, \mathbf{Z}_j)}{\ln q_j} \right\} = -\text{KL}(q_j \parallel \tilde{p}(\mathbf{X}, \mathbf{Z}_j))$$

- Maximising  $\mathcal{L}(q)$  is equivalent to minimising  $\text{KL}(q_j \parallel \tilde{p}(\mathbf{X}, \mathbf{Z}_j))$ .

# Factorised Distributions



- Optimal solution for

$$\ln q_j^*(\mathbf{Z}_j) = \mathbb{E}_{i \neq j} [\ln p(\mathbf{X}, \mathbf{Z})] - \text{const} \quad (2)$$

$$q_j^*(\mathbf{Z}_j) = \frac{\exp(\mathbb{E}_{i \neq j} [\ln p(\mathbf{X}, \mathbf{Z})])}{\int \exp(\mathbb{E}_{i \neq j} [\ln p(\mathbf{X}, \mathbf{Z})]) d\mathbf{Z}_j} \quad (3)$$

- The log of the optimal solution for factor  $q_j$  is obtained by considering the log of the joint distribution over all hidden and visible variables and taking expectations w.r.t. all other factors  $\{q_i\}$  for  $i \neq j$ .
- Don't use (3), better work with (2) and normalise (when required).

Approximate Inference

Approximation Schemes

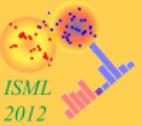
Variational Optimisation

Calculus of Variation

Variational Optimisation  
applied to Inference

Exponential Family

Expectation Propagation



Approximate Inference

Approximation Schemes

Variational Optimisation

Calculus of Variation

Variational Optimisation  
applied to Inference

Exponential Family

Expectation Propagation

# Factorised Distributions - Algorithm

$$\ln q_j^*(\mathbf{Z}_j) = \mathbb{E}_{i \neq j} [\ln p(\mathbf{X}, \mathbf{Z})] - const$$

- ➊ Initialise all factors.
- ➋ Cycle through the factors and replace each with the revised estimate evaluated using the current estimate.
- ➌ Convergence is guaranteed because the bound is convex w.r.t each of the factors.



Approximate Inference

Approximation Schemes

Variational Optimisation

Calculus of Variation

Variational Optimisation  
applied to Inference

Exponential Family

Expectation Propagation

# Exponential Family

- The exponential family of distributions over  $\mathbf{x}$ , given parameters  $\boldsymbol{\eta}$ , is defined to be the set of distributions of the form

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

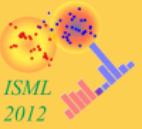
where  $\mathbf{x}$  may be scalar or vector, and may be discrete or continuous.

- Natural parameter  $\boldsymbol{\eta}$
- And  $\mathbf{u}$  is some function of  $\mathbf{x}$ .
- The function  $g(\boldsymbol{\eta})$  can be interpreted as the coefficient ensuring normalisation

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

- Other form with  $g(\boldsymbol{\eta}) = \exp\{-G(\boldsymbol{\eta})\}$  we get

$$p(\mathbf{x} | \boldsymbol{\eta}) = h(\mathbf{x}) \exp \{ \boldsymbol{\eta}^T \mathbf{u}(\mathbf{x}) - G(\boldsymbol{\eta}) \}$$



- Normal distribution with mean  $\mu$  and standard deviation  $\sigma$

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

$$\boldsymbol{\eta} = \left( \frac{\mu}{\sigma^2}, -\frac{1}{2\sigma^2} \right)^T$$

$$h(x) = \frac{1}{\sqrt{2\pi}}$$

$$\mathbf{u}(x) = (x, x^2)^T$$

$$g(\boldsymbol{\eta}) = \sqrt{-2\eta_2} \exp\left(\frac{\eta_1^2}{4\eta_2}\right)$$

$$G(\boldsymbol{\eta}) = -\frac{1}{2} \ln(-2\eta_2) - \frac{\eta_1^2}{4\eta_2}$$

Approximate Inference

Approximation Schemes

Variational Optimisation

Calculus of Variation

Variational Optimisation  
applied to Inference

Exponential Family

Expectation Propagation



Approximate Inference

Approximation Schemes

Variational Optimisation

Calculus of Variation

Variational Optimisation  
applied to Inference

Exponential Family

Expectation Propagation

# Exponential Family - Properties

- Differentiation of  $-\ln g(\boldsymbol{\eta})$  provides the moments

$$\frac{d}{d\boldsymbol{\eta}} - \ln g(\boldsymbol{\eta}) = \mathbb{E} [\mathbf{u}(\mathbf{x})]$$

$$\frac{d^2}{d\boldsymbol{\eta}^2} - \ln g(\boldsymbol{\eta}) = \text{cov}[\mathbf{u}(\mathbf{x})]$$

 $\dots$ 

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

$$\begin{aligned} \frac{d}{d\boldsymbol{\eta}} - \ln g(\boldsymbol{\eta}) &= \frac{d}{d\boldsymbol{\eta}} - \ln \left( \int h(\mathbf{x}) \exp \{ \boldsymbol{\eta}^T \mathbf{u}(\mathbf{x}) \} d\mathbf{x} \right)^{-1} \\ &= - \left( \int h(\mathbf{x}) e^{\boldsymbol{\eta}^T \mathbf{u}(\mathbf{x})} d\mathbf{x} \right) \frac{d}{d\boldsymbol{\eta}} \left( \int h(\mathbf{x}) e^{\boldsymbol{\eta}^T \mathbf{u}(\mathbf{x})} d\mathbf{x} \right)^{-1} \\ &= \left( \int h(\mathbf{x}) e^{\boldsymbol{\eta}^T \mathbf{u}(\mathbf{x})} d\mathbf{x} \right)^{-1} \frac{d}{d\boldsymbol{\eta}} \int h(\mathbf{x}) e^{\boldsymbol{\eta}^T \mathbf{u}(\mathbf{x})} d\mathbf{x} \\ &= \left( \int h(\mathbf{x}) e^{\boldsymbol{\eta}^T \mathbf{u}(\mathbf{x})} d\mathbf{x} \right)^{-1} \int \mathbf{u}(\mathbf{x}) h(\mathbf{x}) e^{\boldsymbol{\eta}^T \mathbf{u}(\mathbf{x})} d\mathbf{x} \end{aligned}$$



Approximate Inference

Approximation Schemes

Variational Optimisation

Calculus of Variation

Variational Optimisation  
applied to Inference

Exponential Family

Expectation Propagation

# Exponential Family - Properties

- Differentiation of  $-\ln g(\boldsymbol{\eta})$  provides the moments

$$\frac{d}{d\boldsymbol{\eta}} - \ln g(\boldsymbol{\eta}) = \mathbb{E} [\mathbf{u}(\mathbf{x})]$$

$$\frac{d^2}{d\boldsymbol{\eta}^2} - \ln g(\boldsymbol{\eta}) = \text{cov}[\mathbf{u}(\mathbf{x})]$$

...

or using  $G(\boldsymbol{\eta})$  defined by  $g(\boldsymbol{\eta}) = \exp\{-G(\boldsymbol{\eta})\}$ ,

$$\frac{d}{d\boldsymbol{\eta}} G(\boldsymbol{\eta}) = \mathbb{E} [\mathbf{u}(\mathbf{x})]$$

$$\frac{d^2}{d\boldsymbol{\eta}^2} G(\boldsymbol{\eta}) = \text{cov}[\mathbf{u}(\mathbf{x})]$$

...

where

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



# Expectation Propagation

- Given a joint distribution over observed data  $\mathcal{D}$  and stochastic variables  $\theta$  in the form of product of factors

$$p(\mathcal{D}, \theta) = \prod_i f_i(\theta)$$

- Goal: Approximate the posterior distribution  $p(\theta | \mathcal{D})$  by a distribution of the form

$$q(\theta) = \frac{1}{Z} \prod_i \tilde{f}_i(\theta).$$

- Also the model evidence  $p(\mathcal{D})$ .

Approximate Inference

Approximation Schemes

Variational Optimisation

Calculus of Variation

Variational Optimisation  
applied to Inference

Exponential Family

Expectation Propagation



Approximate Inference

Approximation Schemes

Variational Optimisation

Calculus of Variation

Variational Optimisation  
applied to Inference

Exponential Family

Expectation Propagation

# Expectation Propagation with Exponential Family

- ➊ Initialise all approximating factors  $\tilde{f}_i(\theta)$ .
- ➋ Initialise the posterior approximation by setting

$$q(\theta) \propto \prod_i \tilde{f}_i(\theta)$$

- ➌ Choose a factor  $\tilde{f}_j(\theta)$  to refine.
- ➍ Remove  $\tilde{f}_j(\theta)$  from the posterior by division

$$q^{\setminus j}(\theta) = \frac{q(\theta)}{\tilde{f}_j(\theta)}$$

- ➎ Evaluate the new posterior by setting the sufficient statistics (moments) of  $q^{\text{new}}$  equal to those of  $q^{\setminus j} \tilde{f}_j(\theta)$  including evaluation of the normalisation constant

$$Z_j = \int q^{\setminus j}(\theta) \tilde{f}_j(\theta) d\theta$$

- ➏ Evaluate and store the new factor, and goto 3.

$$\tilde{f}_j(\theta) = Z_j \frac{q^{\text{new}}(\theta)}{q^{\setminus j}(\theta)}$$



Approximate Inference

Approximation Schemes

Variational Optimisation

Calculus of Variation

Variational Optimisation  
applied to Inference

Exponential Family

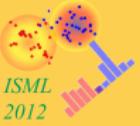
Expectation Propagation

# Expectation Propagation

- Finally, approximate the model evidence

$$p(\mathcal{D}) \approx \int \prod_i \tilde{f}_i(\boldsymbol{\theta}) \, d\boldsymbol{\theta}$$

- Expectation propagation is not guaranteed to converge.



# Part XIX

## *Sampling*

*Motivation*

*Sampling from the  
Uniform Distribution*

*Sampling from Standard  
Distributions*

*Rejection Sampling*

*Adaptive Rejection  
Sampling*

*Importance Sampling*

*Markov Chain Monte  
Carlo - The Idea*

*Gibbs Sampling*



## Motivation

Sampling from the  
Uniform DistributionSampling from Standard  
Distributions

Rejection Sampling

Adaptive Rejection  
Sampling

Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling

# Motivation

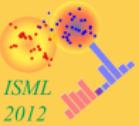
- For most probabilistic models of practical interest, exact inference is intractable. Need approximation.
- Last lecture: deterministic approximations (which can not be exact in principle).
- Now : Numerical sampling (**Monte Carlo** methods).
- Fundamental problem** : Find the expectation of some function  $f(\mathbf{z})$  w.r.t. a probability distribution  $p(\mathbf{z})$

$$\mathbb{E}[f] = \int f(\mathbf{z}) p(\mathbf{z}) d\mathbf{z}$$

- Key idea** : Draw  $\mathbf{z}^{(l)}$ ,  $l = 1, \dots, L$  independent samples from  $p(\mathbf{z})$  and approximate the expectation by

$$\hat{f} = \frac{1}{L} \sum_{l=1}^L f(\mathbf{z}^{(l)})$$

- Problem: How to obtain **independent** samples from  $p(\mathbf{z})$  ?



Motivation

Sampling from the  
Uniform Distribution

Sampling from Standard  
Distributions

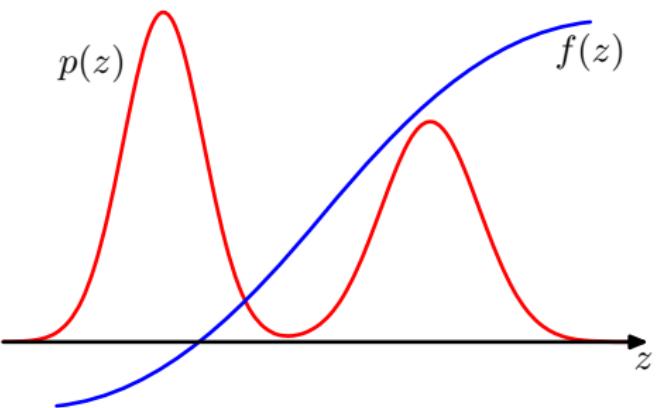
Rejection Sampling

Adaptive Rejection  
Sampling

Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling



# Approximating the Expectation off( $\mathbf{z}$ )

- Samples must be independent, otherwise the effective sample size is much smaller than the apparent sample size.
- If  $f(\mathbf{z})$  is small in regions where  $p(\mathbf{z})$  is large (or vice versa) : need large sample sizes to catch contributions from all regions.

# *Sampling from the Uniform Distribution*



Motivation

Sampling from the  
Uniform Distribution

Sampling from Standard  
Distributions

Rejection Sampling

Adaptive Rejection  
Sampling

Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling

- In a computer usually via **pseudorandom number generator** : an algorithm generating a sequence of numbers that approximates the properties of random numbers.
- Example : **linear congruential generators**

$$z^{(n+1)} = (az^{(n)} + c) \mod m$$

for modulus  $m > 0$ , multiplier  $0 < a < m$ , increment  $0 \leq c < m$ , and seed  $z_0$ .

- Other classes of pseudorandom number generators:
  - Lagged Fibonacci generators
  - Linear feedback shift registers
  - Generalised feedback shift registers

# Pseudorandom Number Generators - Problems



Motivation

Sampling from the  
Uniform Distribution

Sampling from Standard  
Distributions

Rejection Sampling

Adaptive Rejection  
Sampling

Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling

Careful mathematical analysis required to avoid problems like

- Shorter than expected periods for some seed states
- Lack of uniformity of distribution
- Correlation of successive values
- Poor dimensional distribution of the output sequence
- The distances between where certain values occur are distributed differently from those in a random sequence distribution.



Motivation

Sampling from the  
Uniform Distribution

Sampling from Standard  
Distributions

Rejection Sampling

Adaptive Rejection  
Sampling

Importance Sampling

Markov Chain Monte  
Carlo - The Idea

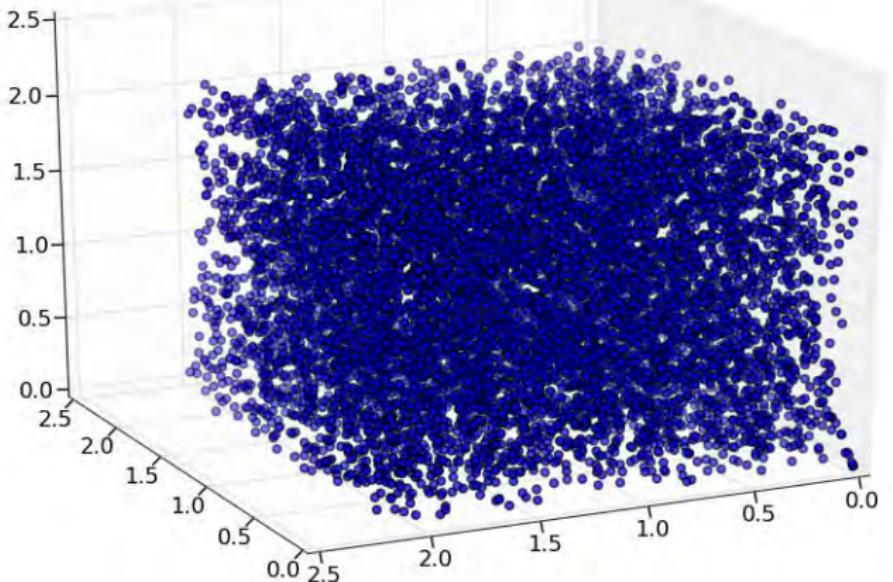
Gibbs Sampling

- Used since the 1960s on many machines
- Defined by the recurrence

$$z^{(n+1)} = (2^{16} + 3) z^{(n)} \mod 2^{31}$$

# A Bad Generator - RANDU

- Plotting  $(z^{(n+2)}, z^{(n+1)}, z^{(n)})^T$  in 3D ...



Motivation

Sampling from the  
Uniform Distribution

Sampling from Standard  
Distributions

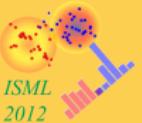
Rejection Sampling

Adaptive Rejection  
Sampling

Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling



Motivation

Sampling from the  
Uniform Distribution

Sampling from Standard  
Distributions

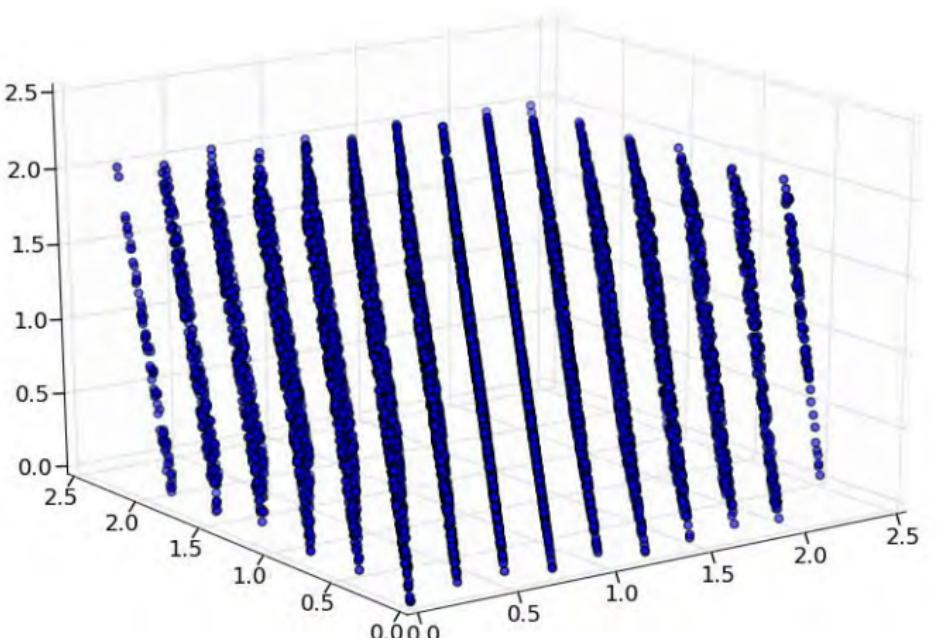
Rejection Sampling

Adaptive Rejection  
Sampling

Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling





Motivation

Sampling from the  
Uniform DistributionSampling from Standard  
Distributions

Rejection Sampling

Adaptive Rejection  
Sampling

Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling

- Analyse the recurrence
- Assuming every equation to be modulo  $2^{31}$ , we can correlate three samples
- Marsaglia, George "Random Numbers Fall Mainly In The Planes", Proc National Academy of Sciences 61, 25-28, 1968.

$$z^{(n+1)} = (2^{16} + 3) z^{(n)} \mod 2^{31}$$

$$\begin{aligned}z^{(n+2)} &= (2^{16} + 3)^2 z^{(n)} \\&= (2^{32} + 6 \cdot 2^{16} + 9) z^{(n)} \\&= (6(2^{16} + 3) - 9) z^{(n)} \\&= 6z^{(n+1)} - 9z^{(n)}\end{aligned}$$

# *Sampling from the Uniform Distribution*



Motivation

Sampling from the  
Uniform Distribution

Sampling from Standard  
Distributions

Rejection Sampling

Adaptive Rejection  
Sampling

Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling

- 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/>

# Sampling from Standard Distributions



Motivation

Sampling from the  
Uniform Distribution

Sampling from Standard  
Distributions

Rejection Sampling

Adaptive Rejection  
Sampling

Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling

- Goal: Sample from  $p(y)$  which is given in analytical form.
- Suppose uniformly distributed samples of  $z$  in the interval  $(0, 1)$  are available.
- Calculate the **cumulative distribution function**

$$h(y) = \int_{-\infty}^y p(x) \, dx$$

- Transform the samples from  $\mathcal{U}(z | 0, 1)$  by

$$y = h^{-1}(z)$$

to obtain samples  $y$  distributed according to  $p(y)$ .



Motivation

Sampling from the  
Uniform Distribution

Sampling from Standard  
Distributions

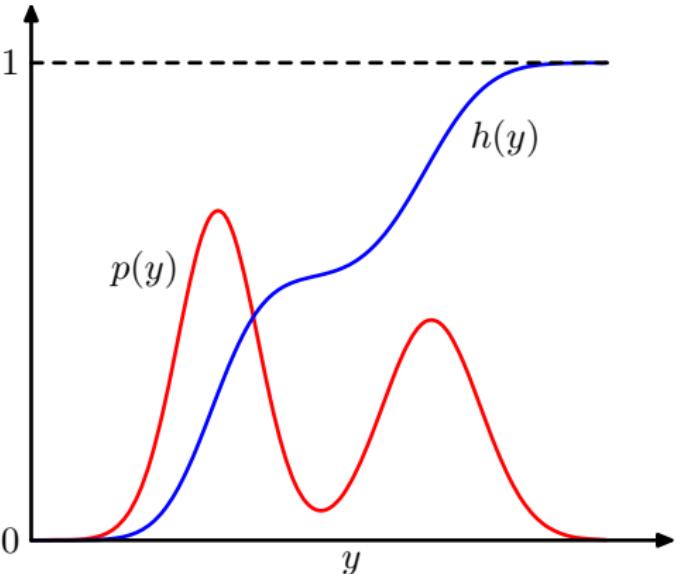
Rejection Sampling

Adaptive Rejection  
Sampling

Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling





Motivation

Sampling from the  
Uniform DistributionSampling from Standard  
Distributions

Rejection Sampling

Adaptive Rejection  
Sampling

Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling

# Sampling from the Exponential Distribution

- Goal: Sample from the exponential distribution

$$p(y) = \begin{cases} \lambda e^{-\lambda y} & 0 \leq y \\ 0 & y < 0 \end{cases}$$

with rate parameter  $\lambda > 0$ .

- Suppose uniformly distributed samples of  $z$  in the interval  $(0, 1)$  are available.
- Calculate the cumulative distribution function

$$h(y) = \int_{-\infty}^y p(x) dx = \int_0^y \lambda e^{-\lambda x} dx = 1 - e^{-\lambda y}$$

- Transform the samples from  $\mathcal{U}(z | 0, 1)$  by

$$y = h^{-1}(z) = -\frac{1}{\lambda} \ln(1 - z)$$

to obtain samples  $y$  distributed according to the exponential distribution.

# *Sampling from Multivariate Distributions*



- Generalisation to multiple variables is straightforward
- Consider change of variables via the Jacobian

$$p(y_1, \dots, y_M) = p(z_1, \dots, z_M) \left| \frac{\partial(z_1, \dots, z_M)}{\partial(y_1, \dots, y_M)} \right|$$

- Technical challenge: Multiple integrals; inverting nonlinear functions of multiple variables.

Motivation

Sampling from the  
Uniform Distribution

Sampling from Standard  
Distributions

Rejection Sampling

Adaptive Rejection  
Sampling

Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling



## Motivation

Sampling from the  
Uniform DistributionSampling from Standard  
Distributions

## Rejection Sampling

Adaptive Rejection  
Sampling

## Importance Sampling

Markov Chain Monte  
Carlo - The Idea

## Gibbs Sampling

## Sampling the Gaussian Distribution - Box-Muller

- ➊ Generate pairs of uniformly distributed random numbers  $z_1, z_2 \in (-1, 1)$  (e.g.  $z_i = 2z - 1$  for  $z$  from  $\mathcal{U}(z | 0, 1)$ )
- ➋ Discard any pair  $(z_1, z_2)$  unless  $z_1^2 + z_2^2 \leq 1$ . Results in a uniform distribution inside of the unit circle  $p(z_1, z_2) = 1/\pi$ .
- ➌ Evaluate  $r^2 = z_1^2 + z_2^2$  and

$$y_1 = z_1 \left( \frac{-2 \ln r^2}{r^2} \right)^{1/2}$$

$$y_2 = z_2 \left( \frac{-2 \ln r^2}{r^2} \right)^{1/2}$$

- ➍  $y_1$  and  $y_2$  are independent with joint distribution

$$p(y_1, y_2) = p(z_1, z_2) \left| \frac{\partial(z_1, z_2)}{\partial(y_1, y_2)} \right| = \frac{1}{\sqrt{2\pi}} e^{-y_1^2/2} \frac{1}{\sqrt{2\pi}} e^{-y_2^2/2}$$



## Motivation

Sampling from the  
Uniform DistributionSampling from Standard  
Distributions

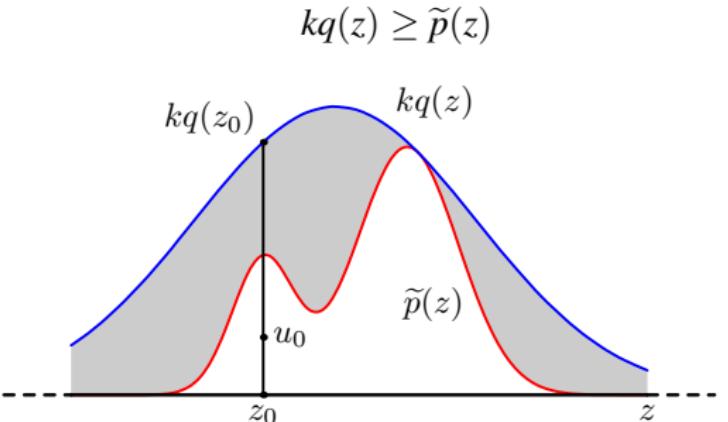
Rejection Sampling

Adaptive Rejection  
Sampling

Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling



# Rejection Sampling

- 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

$$kq(z) \geq \tilde{p}(z)$$



Motivation

Sampling from the  
Uniform DistributionSampling from Standard  
Distributions

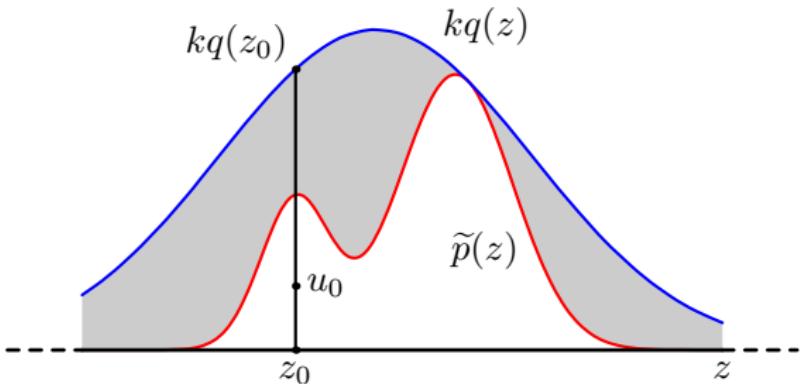
Rejection Sampling

Adaptive Rejection  
Sampling

Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling



# Rejection Sampling

- 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  $p(z)$ .



Motivation

Sampling from the  
Uniform DistributionSampling from Standard  
Distributions

Rejection Sampling

Adaptive Rejection  
Sampling

Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling

# Rejection Sampling - Example

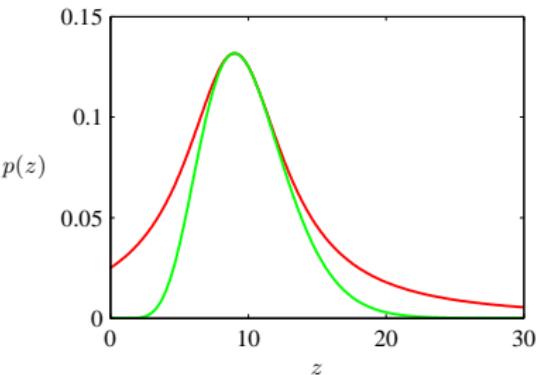
- Consider the Gamma Distribution for  $a > 1$

$$\text{Gam}(z | a, b) = \frac{b^a z^{a-1} \exp(-bz)}{\Gamma(a)}$$

- Suitable  $q(z)$  could be like the **Cauchy distribution**

$$q(z) = \frac{k}{1 + (z - c)^2/b^2}$$

- Samples  $z$  from  $q(z)$  by using uniformly distributed  $y$  and transformation  $z = b \tan y + c$  for  $c = a - 1$ ,  $b^2 = 2a - 1$  and  $k$  as small as possible for  $kq(z) \geq \tilde{p}(z)$ .





## Motivation

Sampling from the  
Uniform DistributionSampling from Standard  
Distributions

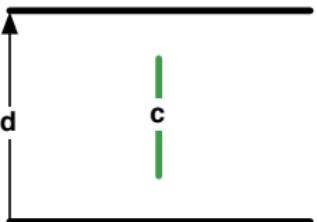
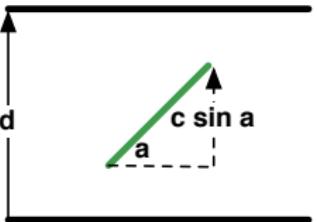
Rejection Sampling

Adaptive Rejection  
Sampling

Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling

(i) Needle falls  
perpendicular ( $a = \pi/2$ ).(ii) Needle falls at  
arbitrary angle  $a$ .



Motivation

Sampling from the  
Uniform DistributionSampling from Standard  
Distributions

Rejection Sampling

Adaptive Rejection  
Sampling

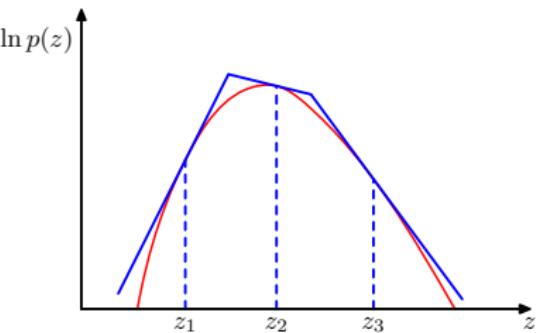
Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling

# Adaptive Rejection Sampling

- Suitable form for the proposal distribution  $q(z)$  might be difficult to find.
  - If  $p(z)$  is **log-concave** ( $\ln p(z)$  has nonincreasing derivatives), use the derivatives to construct an envelope.
- ① Start with an initial grid of points  $z_1, \dots, z_M$  and construct the envelope using the tangents at the  $p(z_i)$ ,  $i = 1, \dots, M$ .
- ② Draw a sample from the envelop function and if accepted use it to calculate  $p(z)$ . Otherwise, use it to refine the grid.



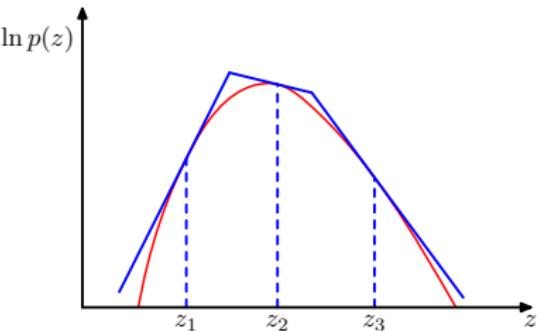
# Adaptive Rejection Sampling - Example



- The piecewise exponential distribution is defined as

$$p(z) = k_m \lambda_m e^{-\lambda_m(z-z_{m-1})} \quad \widehat{z}_{m-1,m} < z \leq \widehat{z}_{m,m+1}$$

where  $\widehat{z}_{m-1,m}$  is the point of intersection of the tangent lines at  $z_{m-1}$  and  $z_m$ ,  $\lambda_m$  is the slope of the tangent at  $z_m$  and  $k_m$  accounts for the corresponding offset.



Motivation

Sampling from the  
Uniform Distribution

Sampling from Standard  
Distributions

Rejection Sampling

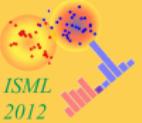
Adaptive Rejection  
Sampling

Importance Sampling

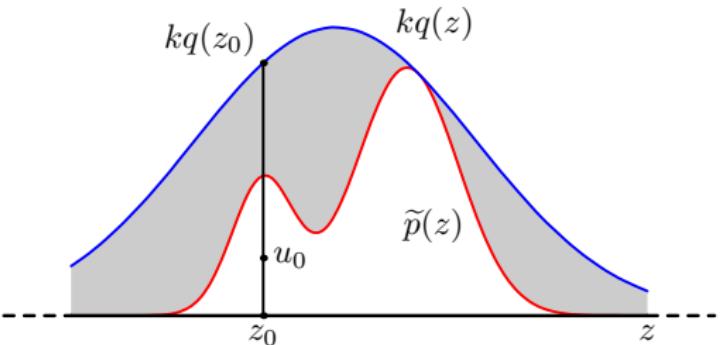
Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling

# Rejection Sampling - Problems



- 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.



Motivation

Sampling from the  
Uniform Distribution

Sampling from Standard  
Distributions

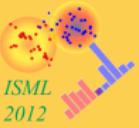
Rejection Sampling

Adaptive Rejection  
Sampling

Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling



## Motivation

Sampling from the  
Uniform DistributionSampling from Standard  
Distributions

## Rejection Sampling

Adaptive Rejection  
Sampling

## Importance Sampling

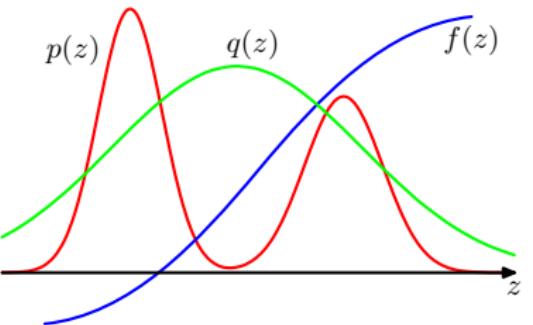
Markov Chain Monte  
Carlo - The Idea

## Gibbs Sampling

# Importance Sampling

- Provides a framework to directly calculate the expectation  $\mathbb{E}_p [f(z)]$  with respect to some distribution  $p(z)$ .
- Does NOT provide  $p(z)$ .
- Again use a proposal distribution  $q(z)$  and draw samples  $z$  from it.
- Then

$$\mathbb{E} [f] = \int f(z) p(z) dz = \int f(z) \frac{p(z)}{q(z)} q(z) dz \approx \frac{1}{L} \sum_{l=1}^L \frac{p(z^{(l)})}{q(z^{(l)})} f(z^{(l)})$$





## Motivation

Sampling from the  
Uniform DistributionSampling from Standard  
Distributions

Rejection Sampling

Adaptive Rejection  
Sampling

Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling

# Importance Sampling - Unnormalised

- Consider both  $\tilde{p}(z)$  and  $\tilde{q}(z)$  to be not normalised.

$$p(z) = \frac{\tilde{p}(z)}{Z_p} \quad q(z) = \frac{\tilde{q}(z)}{Z_q}.$$

- It follows then that

$$\mathbb{E}[f] \approx \frac{Z_q}{Z_p} \frac{1}{L} \sum_{l=1}^L \tilde{r}_l f(z^{(l)}) \quad \tilde{r}_l = \frac{\tilde{p}(z^{(l)})}{\tilde{q}(z^{(l)})}.$$

- Use the same set of samples to calculate

$$\frac{Z_p}{Z_q} \approx \frac{1}{L} \sum_{l=1}^L \tilde{r}_l,$$

- resulting in the formula for unnormalised distributions

$$\mathbb{E}[f] \approx \sum_{l=1}^L w_l f(z^{(l)}) \quad w_l = \frac{\tilde{r}_l}{\sum_{m=1}^L \tilde{r}_m}$$



Motivation

Sampling from the  
Uniform Distribution

Sampling from Standard  
Distributions

Rejection Sampling

Adaptive Rejection  
Sampling

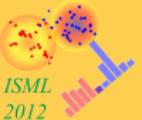
Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling

# Importance Sampling - Key Points

- Try to choose sample points in the input space where the product  $f(z) p(z)$  is large.
- Or at least where  $p(z)$  is large.
- 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.



## Motivation

Sampling from the  
Uniform DistributionSampling from Standard  
Distributions

## Rejection Sampling

Adaptive Rejection  
Sampling

## Importance Sampling

Markov Chain Monte  
Carlo - The Idea

## Gibbs Sampling

# Markov Chain Monte Carlo

- Goal : Generate samples from the distribution  $p(z)$ .
- Idea : Build a machine which uses the current sample to decide which next sample to produce in such a way that the overall distribution of the samples will be  $p(z)$  .

- ➊ Current sample  $z^{(r)}$  is known. Generate a new sample  $z^*$  from a proposal distribution  $q(z | z^{(r)})$  we know how to sample from.
- ➋ Accept or reject the new sample according to some appropriate criterion.

$$z^{(l+1)} = \begin{cases} z^* & \text{if accepted} \\ z^{(r)} & \text{if rejected} \end{cases}$$

- ➌ Proposal distribution depends on the current state.



## Motivation

Sampling from the  
Uniform DistributionSampling from Standard  
Distributions

Rejection Sampling

Adaptive Rejection  
Sampling

Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling

# Metropolis Algorithm

- ➊ Choose a symmetric proposal distribution  $q(z_A | z_B) = q(z_B | z_A)$ .
- ➋ Accept the new sample  $z^*$  with probability

$$A(z^*, z^{(r)}) = \min \left( 1, \frac{\tilde{p}(z^*)}{\tilde{p}(z^{(r)})} \right)$$

- ➌ How? Choose a random number  $u$  with uniform distribution in  $(0, 1)$ . Accept new sample if  $A(z^*, z^{(r)}) > u$ .

- ➍

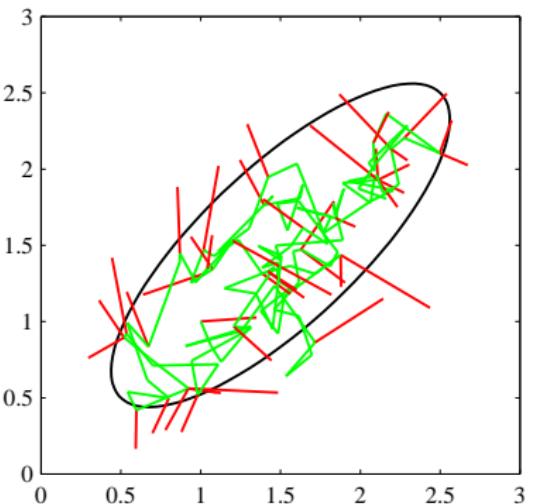
$$z^{(l+1)} = \begin{cases} z^* & \text{if accepted} \\ z^{(r)} & \text{if rejected} \end{cases}$$

Rejection of a point leads to inclusion of the previous sample.  
(Different from rejection sampling.)



# Metropolis Algorithm - Illustration

- Sampling from a Gaussian Distribution (black contour shows one standard deviation).
- Proposal distribution is isotropic Gaussian with standard deviation 0.2.
- 150 candidates generated; 43 rejected.



accepted steps, rejected steps.

Motivation

Sampling from the  
Uniform Distribution

Sampling from Standard  
Distributions

Rejection Sampling

Adaptive Rejection  
Sampling

Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling



# Markov Chain Monte Carlo - Why it works

- A **First Order Markov Chain** is a series of random variables  $z^{(1)}, \dots, z^{(M)}$  such that the following property holds

$$p(z^{(m+1)} | z^{(1)}, \dots, z^{(m)}) = p(z^{(m+1)} | z^{(m)})$$

- 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**.



Motivation

Sampling from the  
Uniform Distribution

Sampling from Standard  
Distributions

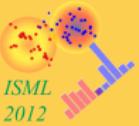
Rejection Sampling

Adaptive Rejection  
Sampling

Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling



## Motivation

Sampling from the  
Uniform DistributionSampling from Standard  
Distributions

Rejection Sampling

Adaptive Rejection  
Sampling

Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling

# Markov Chain Monte Carlo - Why it works

- Marginal probability

$$p(z^{(m+1)}) = \sum_{z^{(m)}} T_m(z^{(m)} | z^{(m+1)}) p(z^{(m)})$$

- 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

$$p^*(z) = \sum_{z'} T(z', z) p^*(z').$$

(Note: There can be many. If  $T$  is the identity matrix, every distribution is invariant.)

[Motivation](#)[Sampling from the  
Uniform Distribution](#)[Sampling from Standard  
Distributions](#)[Rejection Sampling](#)[Adaptive Rejection  
Sampling](#)[Importance Sampling](#)[Markov Chain Monte  
Carlo - The Idea](#)[Gibbs Sampling](#)

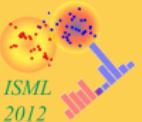
# Markov Chain Monte Carlo - Why it works

- Detailed balance

$$p^*(z) T(z, z') = p^*(z') T(z', z).$$

is sufficient (but not necessary) for  $p^*(z)$  to be invariant. (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  $p(z)$ .

*Motivation**Sampling from the  
Uniform Distribution**Sampling from Standard  
Distributions**Rejection Sampling**Adaptive Rejection  
Sampling**Importance Sampling**Markov Chain Monte  
Carlo - The Idea**Gibbs Sampling*

# *Markov Chain Monte Carlo - Metropolis-Hastings*

- 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

$$A_k^*(z, z^{(\tau)}) = \min \left( 1, \frac{\tilde{p}(z^*) q_k(z^{(\tau)} | z^*)}{\tilde{p}(z^{(\tau)}) q_k(z^* | z^{(\tau)})} \right)$$

- 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

*Motivation**Sampling from the  
Uniform Distribution**Sampling from Standard  
Distributions**Rejection Sampling**Adaptive Rejection  
Sampling**Importance Sampling**Markov Chain Monte  
Carlo - The Idea**Gibbs Sampling*

# Markov Chain Monte Carlo - Metropolis-Hastings

- 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

$$\begin{aligned} p(z) q_k(z' | z) A_k(z', z) &= \min(p(z) q_k(z' | z), p(z') q_k(z | z')) \\ &= \min(p(z') q_k(z | z'), p(z) q_k(z' | z)) \\ &= p(z') q_k(z | z') A_k(z, z'). \end{aligned}$$

# Markov Chain Monte Carlo - Metropolis-Hasting

Introduction to Statistical  
Machine Learning

©2012

Christfried Webers  
NICTA

The Australian National  
University



Motivation

Sampling from the  
Uniform Distribution

Sampling from Standard  
Distributions

Rejection Sampling

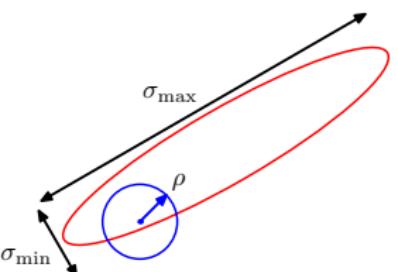
Adaptive Rejection  
Sampling

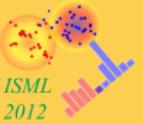
Importance Sampling

Markov Chain Monte  
Carlo - The Idea

Gibbs Sampling

- Isotropic Gaussian proposal distribution (blue)
- In order to keep the rejection rate low, use the smallest standard deviation  $\sigma_{min}$  of the multivariate Gaussian (red) for the proposal distribution.
- Leads to random walk behaviour → slow exploration of the state space.
- Number of steps separating states that are approximately independent is  $(\sigma_{max}/\sigma_{min})^2$ .





*Motivation*

*Principal Component  
Analysis (PCA)*

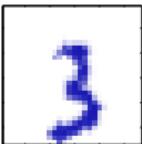
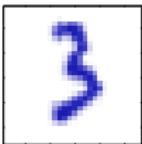
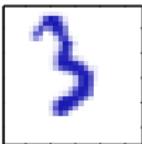
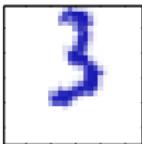
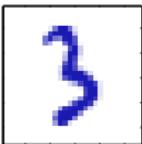
## *Principal Component Analysis*



## Motivation

Principal Component  
Analysis (PCA)

- We know already models with discrete latent variables (e.g. mixture of Gaussians), now look at continuous latent variables.
- Main goal : dimensionality reduction
- Many applications in visualisation, feature extraction, signal processing, data compression ...
- Example: Use hand-written digits (binary data) and place them into a larger frame ( $100 \times 100$ ) varying the position and the rotation angle.
- Data space size = 10 000.
- But data live on a three-dimensional manifold ( $x$ ,  $y$ , and the rotation angle).

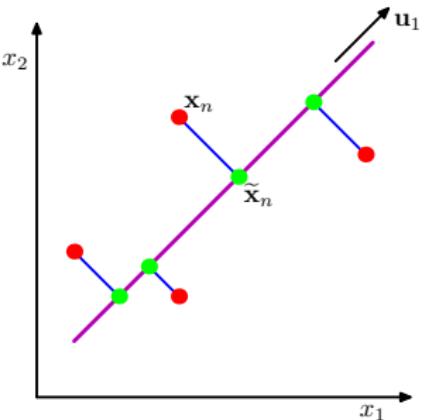




Motivation

Principal Component  
Analysis (PCA)

- Idea: Linearly project the data points onto a lower dimensional subspace such that
  - the variance of the projected data is maximised, or
  - the distortion error from the projection is minimised.
- Both formulation lead to the same result.
- Need to find the lower dimensional subspace, called the principal subspace.

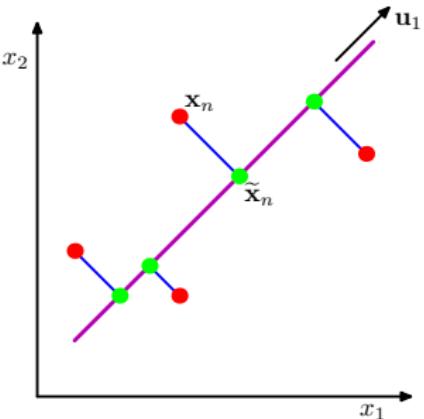




## Motivation

## Principal Component Analysis (PCA)

- Given  $N$  observations  $\mathbf{x}_n \in \mathbb{R}^D$ ,  $n = 1, \dots, N$ .
- Project onto a space with dimensionality  $M < D$  while maximising the variance.
- More advanced : How to calculate  $M$  from the data.  
Therefore here:  $M$  is fixed.
- Consider a 1-dimensional subspace spanned by some unit vector  $\mathbf{u}_1 \in \mathbb{R}^D$ ,  $\mathbf{u}_1^T \mathbf{u}_1 = 1$ .





Motivation

Principal Component  
Analysis (PCA)

# PCA - Maximise Variance

- Each data point  $\mathbf{x}_n$  is then projected onto a scalar value  $\mathbf{u}_1^T \mathbf{x}_n$ .
- The mean of the projected data is  $\mathbf{u}_1^T \bar{\mathbf{x}}$  where  $\bar{\mathbf{x}}$  is the sample mean

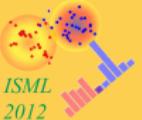
$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n.$$

- The variance of the projected data is then

$$\frac{1}{N} \sum_{n=1}^N \{ \mathbf{u}_1^T \mathbf{x}_n - \mathbf{u}_1^T \bar{\mathbf{x}} \}^2 = \mathbf{u}_1^T \mathbf{S} \mathbf{u}_1$$

with the covariance matrix

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T.$$



## Motivation

Principal Component  
Analysis (PCA)

- Maximising  $\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1$  under the constraint  $\mathbf{u}_1^T \mathbf{u}_1 = 1$  (why do we need to bound  $\mathbf{u}_1$ ?) leads to the Lagrange equation

$$L(\mathbf{u}_1, \lambda_1) = \mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^T \mathbf{u}_1)$$

which has a stationary point if  $\mathbf{u}_1$  is an eigenvector of  $\mathbf{S}$  with eigenvalue  $\lambda_1$ .

$$\mathbf{S} \mathbf{u}_1 = \lambda_1 \mathbf{u}_1.$$

- The variance is then  $\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 = \lambda_1$ .
- Variance is maximised if  $\mathbf{u}_1$  is the eigenvector of the covariance  $\mathbf{S}$  with the largest eigenvalue.



Motivation

Principal Component  
Analysis (PCA)

- Continue maximising the variance amongst all possible directions orthogonal to those already considered.
- The optimal linear projection onto a  $M$ -dimensional space for which the variance is maximised is defined by the  $M$  eigenvectors  $\mathbf{u}_1, \dots, \mathbf{u}_M$  of the covariance matrix  $\mathbf{S}$  corresponding to the  $M$  largest eigenvalues  $\lambda_1, \dots, \lambda_M$ .
- Is this subspace always uniquely defined?
- Not if  $\lambda_M = \lambda_{M+1}$ .



## Motivation

Principal Component  
Analysis (PCA)

# PCA - Minimise Distortion Error

- The distortion between data points  $\mathbf{x}_n$  and their projection  $\tilde{\mathbf{x}}_n$

$$J = \frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2$$

is minimised if the variance is maximised.

- The distortion error is then

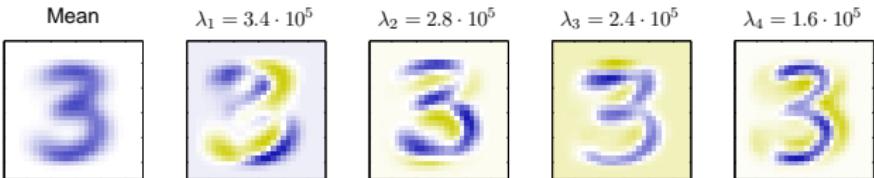
$$J = \sum_{i=M+1}^D \lambda_i$$

where  $\lambda_i, i = M + 1, \dots, D$  are the **smallest** eigenvalues of the covariance matrix  $\mathbf{S}$ .

- In signal processing we speak of the **signal space** (principal subspace) and the **noise space** (orthogonal to the principal subspace).



## Motivation

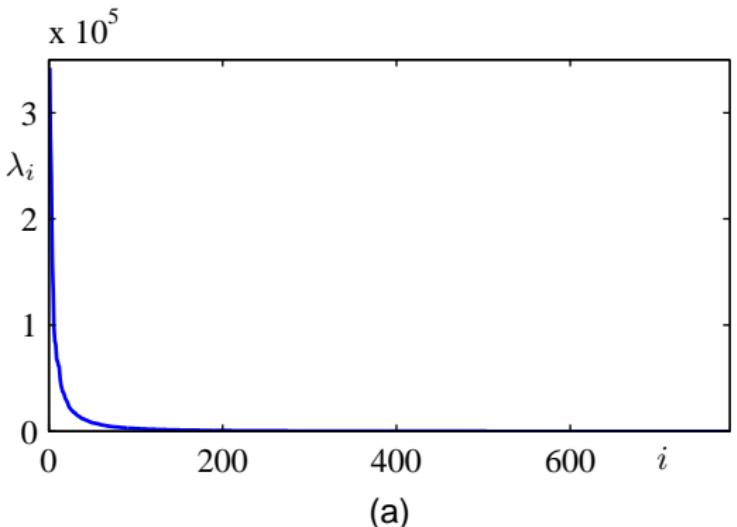
Principal Component  
Analysis (PCA)

The mean and the first four eigenvectors  $\mathbf{u}_1, \dots, \mathbf{u}_4$  of a set of handwritten digits of 'three'.

Blue corresponds to positive values, white is zero and yellow corresponds to negative values.



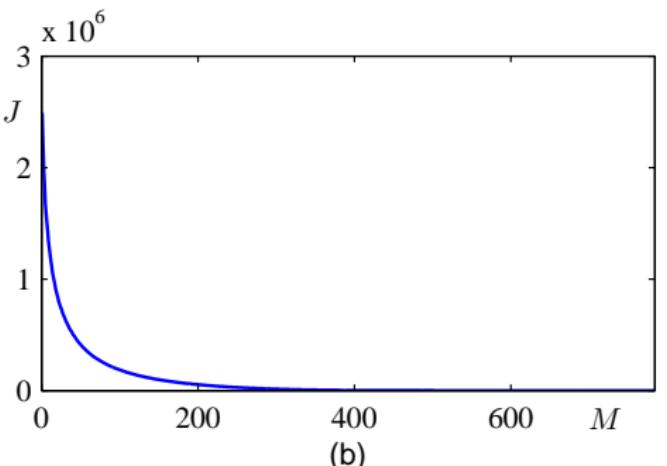
Motivation

Principal Component  
Analysis (PCA)

Plot of the eigenvalue spectrum for the digits of three data set.



Motivation

Principal Component  
Analysis (PCA)

Plot of the distortion error versus the number of dimension of the subspace considered for projection.



Motivation

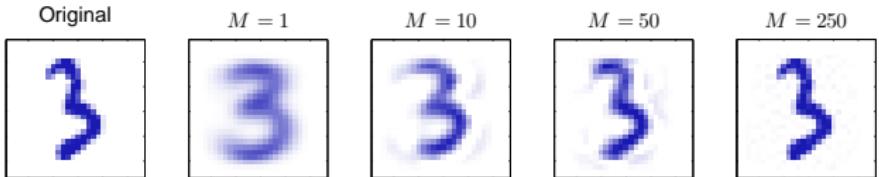
Principal Component  
Analysis (PCA)

# PCA - Compression

- The approximated data vector  $\tilde{\mathbf{x}}_n$  can be written in the form

$$\tilde{\mathbf{x}}_n = \bar{\mathbf{x}} + \sum_{i=1}^M (\mathbf{u}_i^T (\mathbf{x}_n - \bar{\mathbf{x}})) \mathbf{u}_i$$

- Codebook :  $M + 1$  vectors of dimension  $D$  ( $\bar{\mathbf{x}}$  and  $\mathbf{u}_i$ ).
- Compressed  $\mathbf{x}_n$  :  $M$  factors  $\mathbf{u}_i^T (\mathbf{x}_n - \bar{\mathbf{x}})$



Reconstruction of an image retaining  $M$  principal components.

# PCA - Data Preprocessing

- Standardise certain features of a data set (for instance as a preprocessing step to subsequent algorithms expecting these features).
- Usually, individual standardisation: each variable (dimension) has zero mean and unit variance. But variables are still correlated.
- PCA can do more: create **decorrelated** data (covariance is the identity; also called **whitening** or **sphering** of the data)
- Write the eigenvector equation for the covariance matrix  $\mathbf{S}$

$$\mathbf{S}\mathbf{U} = \mathbf{UL}$$

where  $\mathbf{L}$  is the diagonal matrix of (positive!) eigenvalues.

- Transform the original data by

$$\mathbf{y}_n = \mathbf{L}^{-1/2} \mathbf{U}^T (\mathbf{x}_n - \bar{\mathbf{x}})$$

- The set  $\{\mathbf{y}_n\}$  has mean zero and covariance given by the identity.





Motivation

Principal Component  
Analysis (PCA)

# PCA - Data Preprocessing

- Transform the original data by

$$\mathbf{y}_n = \mathbf{L}^{-1/2} \mathbf{U}^T (\mathbf{x}_n - \bar{\mathbf{x}})$$

- Mean of the set  $\{\mathbf{y}_n\}$

$$\frac{1}{N} \sum_{n=1}^N \mathbf{y}_n = \frac{1}{N} \sum_{n=1}^N \mathbf{L}^{-1/2} \mathbf{U}^T (\mathbf{x}_n - \bar{\mathbf{x}})$$

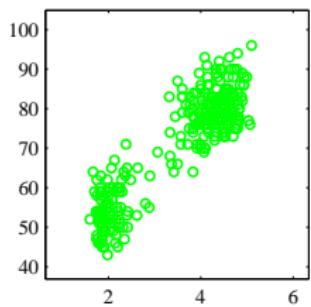
$$= \mathbf{L}^{-1/2} \mathbf{U}^T \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}}) = 0$$

- Covariance of the set  $\{\mathbf{y}_n\}$

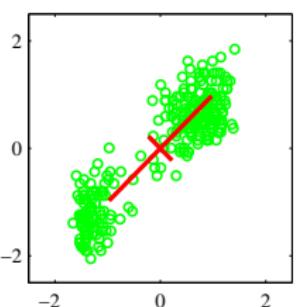
$$\begin{aligned}\frac{1}{N} \sum_{n=1}^N \mathbf{y}_n \mathbf{y}_n^T &= \frac{1}{N} \sum_{n=1}^N \mathbf{L}^{-1/2} \mathbf{U}^T (\mathbf{x}_n - \bar{\mathbf{x}}) (\mathbf{x}_n - \bar{\mathbf{x}})^T \mathbf{U} \mathbf{L}^{-1/2} \\ &= \mathbf{L}^{-1/2} \mathbf{U}^T \mathbf{S} \mathbf{U} \mathbf{L}^{-1/2} \\ &= \mathbf{L}^{-1/2} \mathbf{U}^T \mathbf{U} \mathbf{L} \mathbf{L}^{-1/2} \\ &= \mathbf{I}\end{aligned}$$

# PCA - The Effect of Whitenning

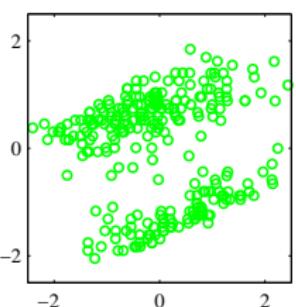
- Compare standardising and whitenning of a data set.
- (b) also shows the principal axis of the normalised data set plotted as red lines over the range  $\pm\lambda_i^{1/2}$ .



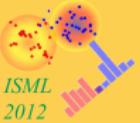
Original data  
(note the different  
axis).



Standardising to  
zero mean and unit  
variance.



Whitening to  
achieve unit  
covariance.



Motivation

Principal Component  
Analysis (PCA)



## Motivation

Principal Component  
Analysis (PCA)

# Independent Component Analysis - Overview

- Assume we have  $K$  signals and  $K$  recordings, each recording containing a mixture of the signals.
- ‘Cocktail party’ problem :  $K$  people speak at the same time in a room, and  $K$  microphones pickup a mixture of what they say.
- Given unknown source signals  $S \in \mathbb{R}^{N \times K}$  and an unknown mixing matrix  $A$ , producing the observed data  $X \in \mathbb{R}^{N \times K}$

$$X = S A$$

- Can we recover the original signals (Blind Source Separation)?
- Yes, under the assumption that
  - at most one of the signals is Gaussian distributed.
  - we don’t care for the amplitude (including the sign).
  - we don’t care for the order of the recovered signals.
  - we have at least as many observed mixtures as signals, the matrix  $A$  has full rank and can be inverted.



## Motivation

Principal Component  
Analysis (PCA)

# Independence versus Uncorrelatedness

- Independence

$$p(x_1, x_2) = p(x_1)p(x_2)$$

- Uncorrelatedness (defined via a zero covariance)

$$\mathbb{E}[x_1 x_2] - \mathbb{E}[x_1] \mathbb{E}[x_2] = 0$$

- Independence implies Uncorrelatedness (prove it!).
- BUT Uncorrelatedness does NOT imply Independence.
- Example: Draw the pair  $(x_1, x_2)$  with equal probability from the set  $\{(0, 1), (0, -1), (1, 0), (-1, 0)\}$ .
- Then  $x_1$  and  $x_2$  are uncorrelated because  $\mathbb{E}[x_1] = \mathbb{E}[x_2] = \mathbb{E}[x_1 x_2] = 0$ .
- But  $x_1$  and  $x_2$  are NOT independent

$$p(x_1 = 0, x_2 = -1) = \frac{1}{4}$$

$$p(x_1 = 0)p(x_2 = -1) = \frac{1}{2} \times \frac{1}{4}$$

# Independent Component Analysis - Overview



Motivation

Principal Component  
Analysis (PCA)

- Uncorrelated variables are not necessarily independent.
- ICA maximises the **statistical independence** of the estimated components.
- Find  $W$  (or  $W^{-1}$ ) in such a way that

$$S = XW^{-1}$$

the columns of  $S$  are maximally independent.

- Several definitions for statistical independence possible.
- Based on the concept of what is ‘nongaussian’.
- Central Limit Theorem: The distribution of a sum of independent random variables tends toward a Gaussian distribution (under certain conditions).
- FastICA algorithm.



# Part XXI

## *Sequential Data I*

*Motivation*

*Stationary versus  
Nonstationary*

*Markov Model*

*State Space Model*

*Hidden Markov Model*

*HMM - Generative View*

*HMM - Handwritten  
Digits*



Motivation

Stationary versus  
Nonstationary

Markov Model

State Space Model

Hidden Markov Model

HMM - Generative View

HMM - Handwritten  
Digits

# Sequential Data

- For many applications, the i.i.d. assumption is a poor one
  - Time series (currencies, rainfall, speech)
  - Sequence of nucleotide base pairs along a DNA strand
  - Sequence of characters in a sentence.
  - ...
- A Current data may not be independent of previous data.
- B The distribution of the data may change while the data of the sequence are drawn/produced/emitted.
- Both A and B.
- Note: Use 'past' and 'future' to describe an order on the observations, but the concept of sequence is not restricted to temporal sequences.



## Motivation

Stationary versus  
Nonstationary

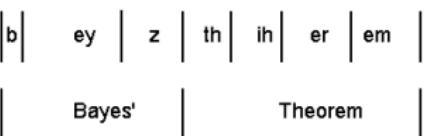
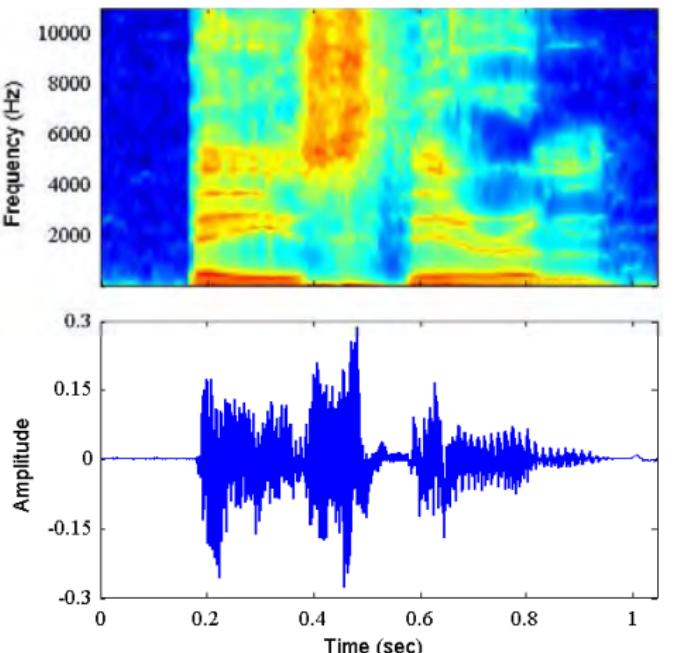
Markov Model

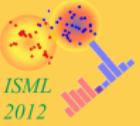
State Space Model

Hidden Markov Model

HMM - Generative View

HMM - Handwritten  
Digits





Motivation

Stationary versus  
Nonstationary

Markov Model

State Space Model

Hidden Markov Model

HMM - Generative View

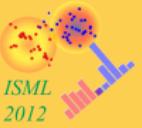
HMM - Handwritten  
Digits

# Stationary versus Nonstationary Sequential Distributions

- **Stationary case** : Data evolves in time, but the distribution from which it is drawn stays the same.
- **Nonstationary case** : The generative distribution changes itself with time.

We will focus on the stationary case.

# *How the future depends on the past*



Motivation

Stationary versus  
Nonstationary

Markov Model

State Space Model

Hidden Markov Model

HMM - Generative View

HMM - Handwritten  
Digits

- Goal: Predict the next value in a sequence.
- Assumption: Not all previous data are equally influencing the next value. (Technical problem: How to store an ever growing history of observations?)
- Assume that recent observations are more likely to be informative for the prediction of the next value than more historical observations.
- Is this always a good assumption?



Motivation

Stationary versus  
Nonstationary

Markov Model

State Space Model

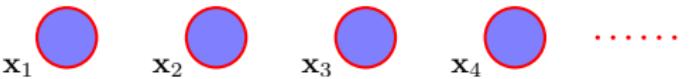
Hidden Markov Model

HMM - Generative View

HMM - Handwritten  
Digits

# Simplest Approach for Modelling a Sequence

- Treat all data as i.i.d.
- Example: Binary variable recording whether it rained or not on a day.
- Only information from such a model: frequency of rainfall.
- Can only use the frequency to predict whether it rains tomorrow or not. (Maybe not so bad for Canberra ;-)
- Usually: Observing whether it rained today helps to predict the weather for tomorrow.
- Need to relax the i.i.d. assumption to grasp this idea.





Motivation

Stationary versus  
Nonstationary

Markov Model

State Space Model

Hidden Markov Model

HMM - Generative View

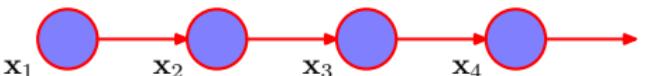
HMM - Handwritten  
Digits

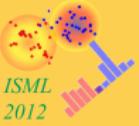
- One of the simplest ways to relax the i.i.d. assumption.
- Use the product rule to exactly express the joint distribution

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N) = p(\mathbf{x}_1) \prod_{n=2}^N p(\mathbf{x}_n | \mathbf{x}_1, \dots, \mathbf{x}_{n-1})$$

- Assume that each of the conditional expressions depends only on the most recent.
- First-order Markov chain

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N) = p(\mathbf{x}_1) \prod_{n=2}^N p(\mathbf{x}_n | \mathbf{x}_{n-1})$$





Motivation

Stationary versus  
Nonstationary

Markov Model

State Space Model

Hidden Markov Model

HMM - Generative View

HMM - Handwritten  
Digits

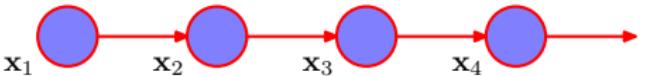
- Given the factorisation of the first-order Markov chain

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N) = p(\mathbf{x}_1) \prod_{n=2}^N p(\mathbf{x}_n | \mathbf{x}_{n-1})$$

- What is the conditional distribution for observation  $\mathbf{x}_n$  given the previous observations?

$$\begin{aligned} p(\mathbf{x}_n | \mathbf{x}_1, \dots, \mathbf{x}_{n-1}) &= \frac{p(\mathbf{x}_1, \dots, \mathbf{x}_n)}{p(\mathbf{x}_1, \dots, \mathbf{x}_{n-1})} \\ &= \frac{p(\mathbf{x}_1) \prod_{i=2}^n p(\mathbf{x}_i | \mathbf{x}_{i-1})}{p(\mathbf{x}_1) \prod_{j=2}^{n-1} p(\mathbf{x}_j | \mathbf{x}_{j-1})} \\ &= p(\mathbf{x}_n | \mathbf{x}_{n-1}) \end{aligned}$$

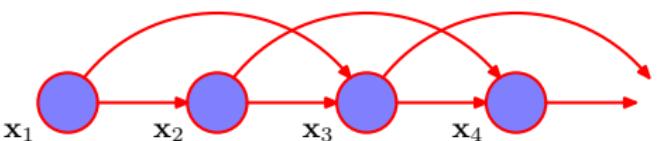
(Not surprisingly, this matches our assumption.)





- Assume that the trend in previous observations provides important information in predicting the next value.
- For a trend, we need at least two previous observations.

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N) = p(\mathbf{x}_1) p(\mathbf{x}_2 | \mathbf{x}_1) \prod_{n=3}^N p(\mathbf{x}_n | \mathbf{x}_{n-1}, \mathbf{x}_{n-2})$$



Motivation

Stationary versus  
Nonstationary

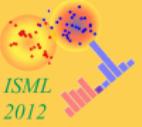
Markov Model

State Space Model

Hidden Markov Model

HMM - Generative View

HMM - Handwritten  
Digits



Motivation

Stationary versus  
Nonstationary

Markov Model

State Space Model

Hidden Markov Model

HMM - Generative View

HMM - Handwritten  
Digits

# $M^{\text{th}}$ -order Markov Chain

- Extend this idea to an  $M^{\text{th}}$ -order Markov chain in which the probability of each observation depends on the previous  $M$  observations

$$\begin{aligned} p(\mathbf{x}_1, \dots, \mathbf{x}_N) = & p(\mathbf{x}_1) p(\mathbf{x}_2 | \mathbf{x}_1) \dots p(\mathbf{x}_M | \mathbf{x}_1, \dots, \mathbf{x}_{M-1}) \\ & \times \prod_{n=M+1}^N p(\mathbf{x}_n | \mathbf{x}_{n-1}, \mathbf{x}_{n-2} \dots, \mathbf{x}_{n-M}) \end{aligned}$$

- What is a reasonable  $M$  ?



Motivation

Stationary versus  
Nonstationary

Markov Model

State Space Model

Hidden Markov Model

HMM - Generative View

HMM - Handwritten  
Digits

# $M^{\text{th}}$ -order Markov Chain

- Assuming  $K$  different states for each variable  $x$ , how many parameters does a  $M^{\text{th}}$ -order Markov chain have?
- $M = 0$  : no Markov parameter, i.i.d. data
- $M = 1$  : First-order Markov chain,  $K - 1$  parameters for each of the  $K$  states of the previous observation. Number of parameters:  $K(K - 1)$ .
- $M$  :  $M^{\text{th}}$ -order Markov Chain,  $K - 1$  parameters for each of the  $K$  states of the previous  $M$  observation. Number of parameters:  $K^M(K - 1)$ .
- Number of parameters grows exponentially with the order of the Markov chain. Impractical for larger  $M$ .

# Extending the Model



Motivation

Stationary versus  
Nonstationary

Markov Model

State Space Model

Hidden Markov Model

HMM - Generative View

HMM - Handwritten  
Digits

- Goal: We want a model which is NOT restricted to the Markov assumption to any order. BUT can be specified by a limited number of free parameters.
- Use the idea of **latent variables** to construct a rich class of models out of simple components.
- (Remember mixture of Gaussians.)



Motivation

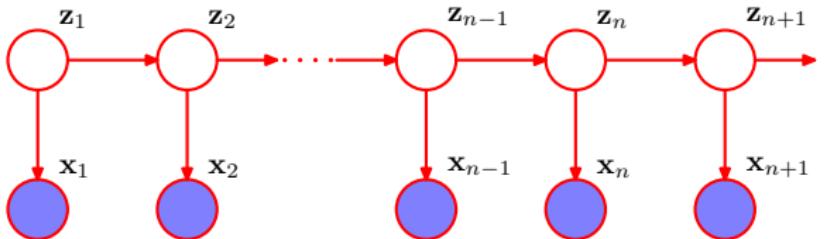
Stationary versus  
Nonstationary

Markov Model

State Space Model

Hidden Markov Model

HMM - Generative View

HMM - Handwritten  
Digits

# State Space Model

- For each observation  $\mathbf{x}_n$ , a latent variable  $\mathbf{z}_n$  is added.
- The type and dimensionality of  $\mathbf{z}_n$  can differ from  $\mathbf{x}_n$ .
- Assume that the latent variables form a Markov chain.
- Key property: conditional independence of the latent variables

$$\mathbf{z}_{n+1} \perp\!\!\!\perp \mathbf{z}_{n-1} \mid \mathbf{z}_n$$

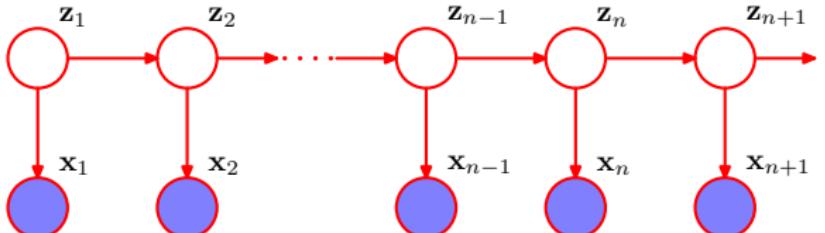


# State Space Model

- Joint distribution for the state space model

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{z}_1, \dots, \mathbf{z}_N) = p(\mathbf{z}_1) \left[ \prod_{n=2}^N p(\mathbf{z}_n | \mathbf{z}_{n-1}) \right] \prod_{n=1}^N p(\mathbf{x}_n | \mathbf{z}_n)$$

- $d$ -separation of the graphical model: no blocked path between two observed variables  $\mathbf{x}_i$  and  $\mathbf{x}_j$  (no HT/TH node on any  $\mathbf{x}_i$ , nor any HH node on the latent variables  $\mathbf{z}_i$ )
- Predictive distribution  $p(\mathbf{x}_{n+1} | \mathbf{x}_1, \dots, \mathbf{x}_n)$  depends on all previous observations.
- The observed variables  $\mathbf{x}_n$  do not satisfy the Markov property of any order.



Motivation

Stationary versus  
Nonstationary

Markov Model

State Space Model

Hidden Markov Model

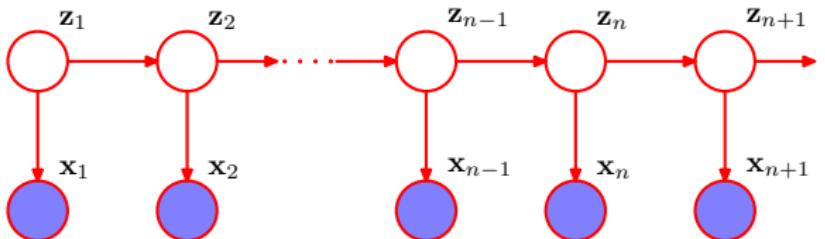
HMM - Generative View

HMM - Handwritten  
Digits

# *State Space Model*



- Two important models described by this graphical model.
- Hidden Markov Model or HMM : Latent variables  $z_n$  are discrete.
- Linear Dynamical System : Latent and observed variables are Gaussian with a linear-Gaussian dependence of the conditional distribution on their parents.



Motivation

Stationary versus  
Nonstationary

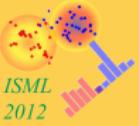
Markov Model

State Space Model

Hidden Markov Model

HMM - Generative View

HMM - Handwritten  
Digits



Motivation

Stationary versus  
Nonstationary

Markov Model

State Space Model

Hidden Markov Model

HMM - Generative View

HMM - Handwritten  
Digits

- State space model with discrete latent variables.

$$\begin{aligned} p(\mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{z}_1, \dots, \mathbf{z}_N) &= p(\mathbf{z}_1) \left[ \prod_{n=2}^N p(\mathbf{z}_n \mid \mathbf{z}_{n-1}) \right] \prod_{n=1}^N p(\mathbf{x}_n \mid \mathbf{z}_n) \\ &= p(\mathbf{z}_1) p(\mathbf{x}_1 \mid \mathbf{z}_1) \prod_{n=2}^N p(\mathbf{z}_n \mid \mathbf{z}_{n-1}) p(\mathbf{x}_n \mid \mathbf{z}_n) \end{aligned}$$

- Each step can be viewed as an extension of the mixture distribution model where each of the component densities is  $p(\mathbf{x} \mid \mathbf{z})$ . Choice of mixture component depends now on the previous state and is represented by  $p(\mathbf{z}_n \mid \mathbf{z}_{n-1})$ .
- Latent variables are discrete multinomial variables  $\mathbf{z}_n$  describing which component of the mixture is responsible for generating the corresponding observable  $\mathbf{x}_n$ .



Motivation

Stationary versus  
Nonstationary

Markov Model

State Space Model

Hidden Markov Model

HMM - Generative View

HMM - Handwritten  
Digits

# Hidden Markov Model

- Assume 1-in- $K$  coding scheme.
- Each latent variable  $\mathbf{z}_n$  has  $K$  different states.
- The conditional distribution  $p(\mathbf{z}_n \mid \mathbf{z}_{n-1})$  is a table (matrix) with  $K \times K$  entries, denoted by  $\mathbf{A} \in \{0, 1\}^{K \times K}$ .
- The elements of  $\mathbf{A}$  are called **transition probabilities**

$$A_{jk} = p(z_{n,k} = 1 \mid z_{n-1,j} = 1).$$

satisfying  $0 \leq A_{jk} \leq 1$  and  $\sum_{k=1}^K A_{jk} = 1$ .

- The number of independent parameters is  $K(K - 1)$  .



Motivation

Stationary versus  
Nonstationary

Markov Model

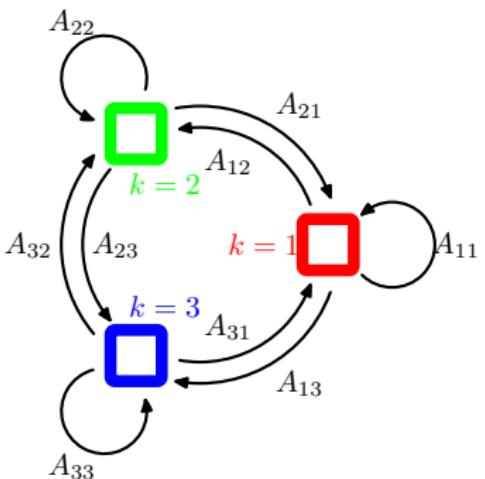
State Space Model

Hidden Markov Model

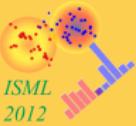
HMM - Generative View

HMM - Handwritten  
Digits

$$p(\mathbf{z}_n \mid \mathbf{z}_{n-1}, \mathbf{A}) = \prod_{k=1}^K \prod_{j=1}^K A_{jk}^{z_{n-1,j} z_{nk}}$$



Transition Diagram for a model with three possible states.  
Black lines denote the elements of the transition matrix  $A_{jk}$ .



Motivation

Stationary versus  
Nonstationary

Markov Model

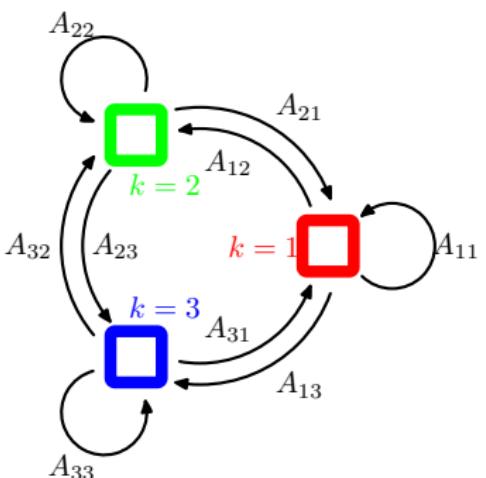
State Space Model

Hidden Markov Model

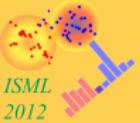
HMM - Generative View

HMM - Handwritten  
Digits

$$p(\mathbf{z}_1 \mid \boldsymbol{\pi}) = \prod_{k=1}^K \pi_k^{z_{1k}} \quad \sum_k \pi_k = 1$$



Transition Diagram for a model with three possible states.  
Black lines denote the elements of the transition matrix  $A_{jk}$ .



Motivation

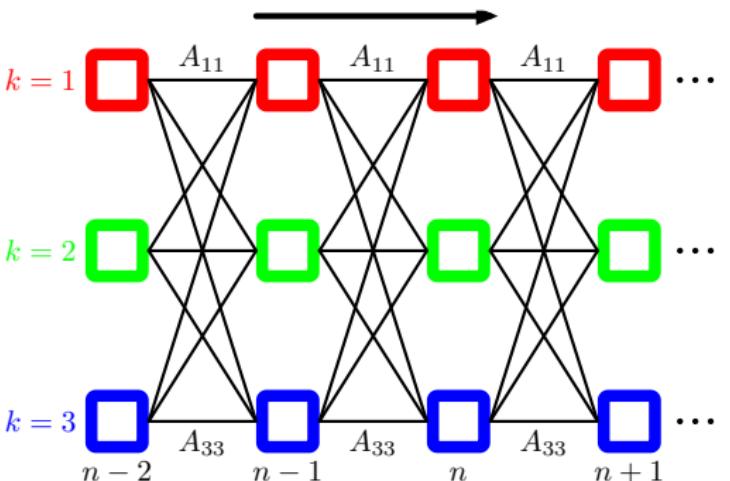
Stationary versus  
Nonstationary

Markov Model

State Space Model

Hidden Markov Model

HMM - Generative View

HMM - Handwritten  
Digits

Unfolded Transition Diagram for a model with 3 possible states.  
Each column corresponds to one latent variable  $\mathbf{z}_n$ .

# Hidden Markov Model - Emission Probabilities



Motivation

Stationary versus  
Nonstationary

Markov Model

State Space Model

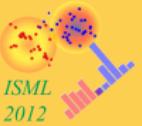
Hidden Markov Model

HMM - Generative View

HMM - Handwritten  
Digits

- Complete the HMM by defining the emission probabilities  $p(\mathbf{x}_n | \mathbf{z}_n, \phi)$  where  $\phi$  is a set of parameters governing the conditional distributions.
- As  $\mathbf{x}_n$  is observed, and  $\phi$  is given,  $p(\mathbf{x}_n | \mathbf{z}_n, \phi)$  is a  $K$ -dimensional vector corresponding to the  $K$  possible states of the binary vector  $\mathbf{z}_n$ .
- Emission probabilities can be represented as

$$p(\mathbf{x}_n | \mathbf{z}_n, \phi) = \prod_{k=1}^K p(\mathbf{x}_n | \phi_k)^{z_{nk}}$$



Motivation

Stationary versus  
Nonstationary

Markov Model

State Space Model

Hidden Markov Model

HMM - Generative View

HMM - Handwritten  
Digits

# Hidden Markov Model - Homogeneous

- Remember: Transition probability in a Markov chain  $T(\mathbf{z}_{n-1}, \mathbf{z}_n) = p(\mathbf{z}_n | \mathbf{z}_{n-1})$ .
- A Hidden Markov Model is called homogeneous if the transition probabilities are the same for all steps  $n$

$$p(\mathbf{z}_n | \mathbf{z}_{n-1}) = p(\mathbf{z}_{n-1} | \mathbf{z}_{n-2}) \quad \forall n = 3, \dots, N$$

- Assume a homogeneous HMM in the following.



- Joint probability distribution over both latent and observed variables

$$p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta}) = p(\mathbf{z}_1 | \boldsymbol{\pi}) \left[ \prod_{n=2}^N p(\mathbf{z}_n | \mathbf{z}_{n-1}, \mathbf{A}) \right] \prod_{m=1}^N p(\mathbf{x}_m | \mathbf{z}_m, \boldsymbol{\phi})$$

where  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ ,  $\mathbf{Z} = (\mathbf{z}_1, \dots, \mathbf{z}_N)$ , and  $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\phi}\}$ .

- Most of the discussion will be independent of the particular choice of emission probabilities (e.g. discrete tables, Gaussians, mixture of Gaussians).

Motivation

Stationary versus  
Nonstationary

Markov Model

State Space Model

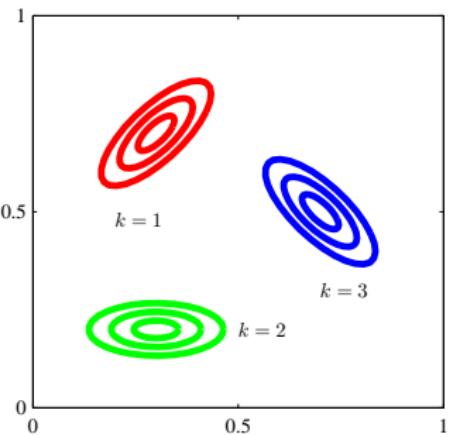
Hidden Markov Model

HMM - Generative View

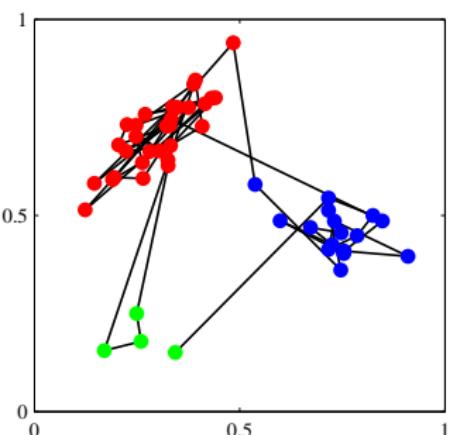
HMM - Handwritten  
Digits



- Sampling from a hidden Markov model having a 3-state latent variable  $\mathbf{z}$  and a Gaussian emission model  $p(\mathbf{x} | \mathbf{z})$  where  $\mathbf{x} \in \mathbb{R}^2$ .



Contours of constant probability for the emission probabilities.



Sampling with 5% probability of change to each of the other states.

Motivation

Stationary versus  
Nonstationary

Markov Model

State Space Model

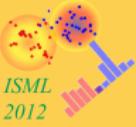
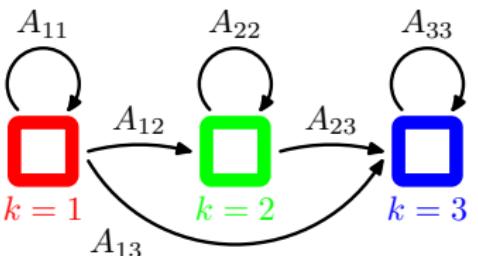
Hidden Markov Model

HMM - Generative View

HMM - Handwritten  
Digits

# Hidden Markov Model - Variants

- Create variants of HMM by imposing constraints on the transition matrix  $\mathbf{A}$ .
- **Left-to-right HMM** : set all elements of  $\mathbf{A}$  above the diagonal to zero,  $A_{jk} = 0$  for  $k < j$ .
- Set the initial state probability  $p(z_{11}) = 1$  and  $p(z_{1j}) = 0$  for  $j \neq 1$ .
- Then, every sequence is constrained to start in state  $k = 1$  and every state left can not be revisited again.



Motivation

Stationary versus  
Nonstationary

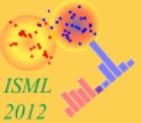
Markov Model

State Space Model

Hidden Markov Model

HMM - Generative View

HMM - Handwritten  
Digits



Motivation

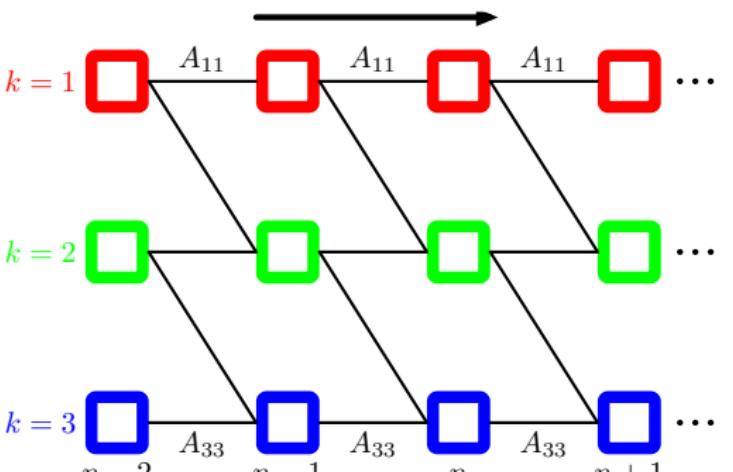
Stationary versus  
Nonstationary

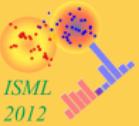
Markov Model

State Space Model

Hidden Markov Model

HMM - Generative View

HMM - Handwritten  
DigitsExample of a HMM with  $\Delta = 1$ .



Motivation

Stationary versus  
Nonstationary

Markov Model

State Space Model

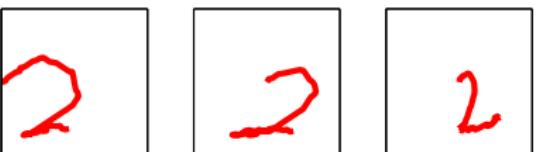
Hidden Markov Model

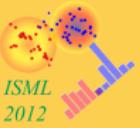
HMM - Generative View

HMM - Handwritten  
Digits

# Hidden Markov Model - Handwritten Digits

- Digitize the trajectory online.
- $K = 16$  states representing a line segment (of fixed width) at 16 different angles.
- Trained with 45 examples of the digit '2'.
- Left-to-right transmission probability with  $\Delta = 1$ .
- Upper row: Training samples used in 25 iterations of EM.
- Lower row: Sampled from the trained algorithm.





## Part XXII

### *Sequential Data 2*

*A Simple Example*

*Maximum Likelihood for  
HMM*

*Forward-Backward  
HMM*

*Conditional  
Independence*

*Alpha-Beta HMM*

*How to train a HMM  
using EM?*

*HMM - Viterbi Algorithm*



*A Simple Example*

*Maximum Likelihood for  
HMM*

*Forward-Backward  
HMM*

*Conditional  
Independence*

*Alpha-Beta HMM*

*How to train a HMM  
using EM?*

*HMM - Viterbi Algorithm*

# *Example of a Hidden Markov Model*

Assume Peter and Mary are students in Canberra and Sydney, respectively. Peter is a computer science student and only interested in riding his bicycle, shopping for new computer gadgets, and studying. (Well, he also does other things but because these other activities don't depend on the weather we neglect them here.)

Mary does not know the current weather in Canberra, but knows the general trends of the weather in Canberra. She also knows Peter well enough to know what he does on average every day.

She believes that the weather follows a given discrete Markov chain. She tries to guess the sequence of weather patterns for a number of days after Bob tells her on the phone what he did in the last days.

[A Simple Example](#)[Maximum Likelihood for  
HMM](#)[Forward-Backward  
HMM](#)[Conditional  
Independence](#)[Alpha-Beta HMM](#)[How to train a HMM  
using EM?](#)[HMM - Viterbi Algorithm](#)

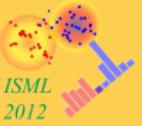
# Example of a Hidden Markov Model

Mary uses the following HMM

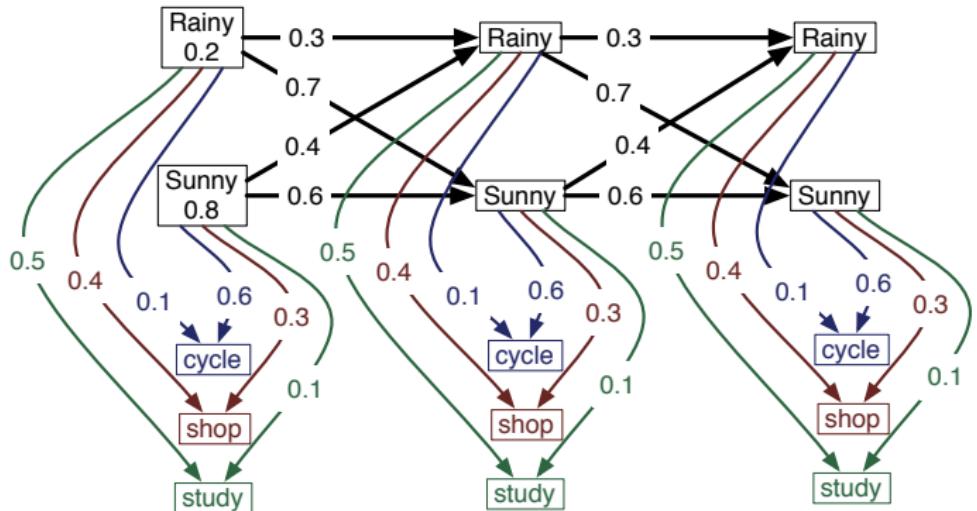
		rainy	sunny
initial probability		0.2	0.8
transition probability	rainy	0.3	0.7
	sunny	0.4	0.6
emission probability	cycle	0.1	0.6
	shop	0.4	0.3
	study	0.5	0.1

Assume, Peter tells Mary that the list of his activities in the last days was ['cycle', 'shop', 'study']

- Calculate the probability of this observation sequence.
- Calculate the most probable sequence of hidden states for these observations.



- Most probable hidden states for 'cycle', 'shop', 'study'



A Simple Example

Maximum Likelihood for HMM

Forward-Backward HMM

Conditional Independence

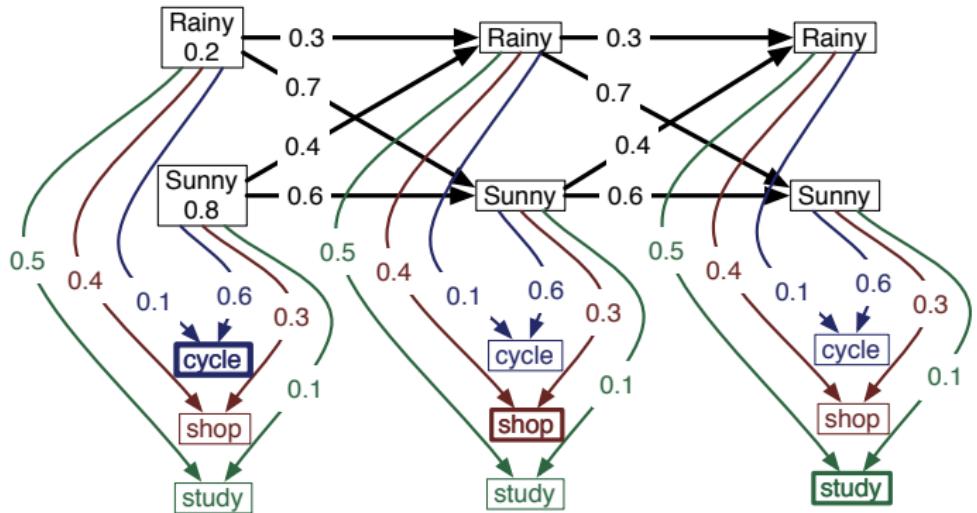
Alpha-Beta HMM

How to train a HMM using EM?

HMM - Viterbi Algorithm



- Most probable hidden states for 'cycle', 'shop', 'study'



A Simple Example

Maximum Likelihood for HMM

Forward-Backward HMM

Conditional Independence

Alpha-Beta HMM

How to train a HMM using EM?

HMM - Viterbi Algorithm



# Maximum Likelihood for HMM

- We have observed a data set  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$ .
- Assume it came from a HMM with a given structure (number of nodes, form of emission probabilities).
- The likelihood of the data is

$$p(\mathbf{X} | \boldsymbol{\theta}) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})$$

- This joint distribution does not factorise over  $n$  (as with the mixture distribution).
- We have  $N$  variables, each with  $K$  states :  $K^N$  terms.  
Number of terms grows exponentially with the length of the chain.
- But we may use the conditional independence of the latent variables to reorder their calculation later.
- Further obstacle to find a closed loop maximum likelihood solution: calculating the emission probabilities for different states  $\mathbf{z}_n$ .

A Simple Example

Maximum Likelihood for  
HMM

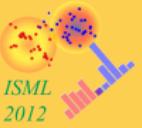
Forward-Backward  
HMM

Conditional  
Independence

Alpha-Beta HMM

How to train a HMM  
using EM?

HMM - Viterbi Algorithm



- Employ the EM algorithm to find the Maximum Likelihood for HMM.
- Start with some initial parameter settings  $\theta^{\text{old}}$ .
- E-step: Find the posterior distribution of the latent variables  $p(\mathbf{Z} | \mathbf{X}, \theta^{\text{old}})$ .
- M-step: Maximise

$$Q(\theta, \theta^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z} | \mathbf{X}, \theta^{\text{old}}) \ln p(\mathbf{Z}, \mathbf{X} | \theta)$$

with respect to the parameters  $\theta = \{\pi, \mathbf{A}, \phi\}$ .

A Simple Example

Maximum Likelihood for  
HMM

Forward-Backward  
HMM

Conditional  
Independence

Alpha-Beta HMM

How to train a HMM  
using EM?

HMM - Viterbi Algorithm



## A Simple Example

Maximum Likelihood for  
HMMForward-Backward  
HMMConditional  
Independence

## Alpha-Beta HMM

How to train a HMM  
using EM?

## HMM - Viterbi Algorithm

## Maximum Likelihood for HMM - EM

- Denote the marginal posterior distribution of  $\mathbf{z}_n$  by  $\gamma(\mathbf{z}_n)$ , and
- the joint posterior distribution of two successive latent variables by  $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$

$$\gamma(\mathbf{z}_n) = p(\mathbf{z}_n | \mathbf{X}, \boldsymbol{\theta}^{\text{old}})$$

$$\xi(\mathbf{z}_{n-1}, \mathbf{z}_n) = p(\mathbf{z}_{n-1}, \mathbf{z}_n | \mathbf{X}, \boldsymbol{\theta}^{\text{old}}).$$

- For each step  $n$ ,  $\gamma(\mathbf{z}_n)$  has  $K$  nonnegative values which sum to 1.
- For each step  $n$ ,  $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$  has  $K \times K$  nonnegative values which sum to 1.
- Elements of these vectors are denoted by  $\gamma(z_{nk})$  and  $\xi(z_{n-1,j}, z_{nk})$  respectively.



A Simple Example

Maximum Likelihood for  
HMMForward-Backward  
HMMConditional  
Independence

Alpha-Beta HMM

How to train a HMM  
using EM?

HMM - Viterbi Algorithm

# Maximum Likelihood for HMM - EM

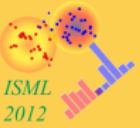
- Because the expectation of a binary random variable is the probability that it is one, we get with this notation

$$\gamma(z_{nk}) = \mathbb{E}[z_{nk}] = \sum_{\mathbf{z}_n} \gamma(\mathbf{z}_n) z_{nk}$$

$$\xi(z_{n-1,j}, z_{nk}) = \mathbb{E}[z_{n-1,j}, z_{nk}] = \sum_{\mathbf{z}_{n-1}, \mathbf{z}_n} \gamma(\mathbf{z}) z_{n-1,j} z_{nk}.$$

- Putting all together we get

$$\begin{aligned} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) &= \sum_{k=1}^K \gamma(z_{1k}) \ln \pi_k + \sum_{n=2}^N \sum_{j=1}^K \sum_{k=1}^K \xi(z_{n-1,j}, z_{nk}) \ln A_{jk} \\ &\quad + \sum_{n=1}^N \sum_{k=1}^K \gamma(z_{nk}) \ln p(\mathbf{x}_n | \phi_k). \end{aligned}$$



A Simple Example

Maximum Likelihood for  
HMMForward-Backward  
HMMConditional  
Independence

Alpha-Beta HMM

How to train a HMM  
using EM?

HMM - Viterbi Algorithm

# Maximum Likelihood for HMM - EM

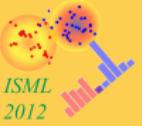
- M-step: Maximising

$$\begin{aligned} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) &= \sum_{k=1}^K \gamma(z_{1k}) \ln \pi_k + \sum_{n=2}^N \sum_{j=1}^K \sum_{k=1}^K \xi(z_{n-1,j}, z_{nk}) \ln A_{jk} \\ &\quad + \sum_{n=1}^N \sum_{k=1}^K \gamma(z_{nk}) \ln p(\mathbf{x}_n \mid \phi_k). \end{aligned}$$

- results in

$$\pi_k = \frac{\gamma(z_{1k})}{\sum_{j=1}^K \gamma(z_{1j})}$$

$$A_{jk} = \frac{\sum_{n=2}^N \xi(z_{n-1,j}, z_{nk})}{\sum_{l=1}^K \sum_{n=2}^N \xi(z_{n-1,j}, z_{nl})}$$



A Simple Example

Maximum Likelihood for  
HMMForward-Backward  
HMMConditional  
Independence

Alpha-Beta HMM

How to train a HMM  
using EM?

HMM - Viterbi Algorithm

# Maximum Likelihood for HMM - EM

- Still left: Maximising with respect to  $\phi$ .

$$\begin{aligned} Q(\theta, \theta^{\text{old}}) &= \sum_{k=1}^K \gamma(z_{1k}) \ln \pi_k + \sum_{n=2}^N \sum_{j=1}^K \sum_{k=1}^K \xi(z_{n-1,j}, z_{nk}) \ln A_{jk} \\ &\quad + \sum_{n=1}^N \sum_{k=1}^K \gamma(z_{nk}) \ln p(\mathbf{x}_n | \phi_k). \end{aligned}$$

- But  $\phi$  only appears in the last term, and under the assumption that all the  $\phi_k$  are independent of each other, this term decouples into a sum.
- Then maximise each contribution  $\sum_{n=1}^N \gamma(z_{nk}) \ln p(\mathbf{x}_n | \phi_k)$  individually.



- In the case of Gaussian emission densities

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

we get for the maximising parameters for the emission densities

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

A Simple Example

Maximum Likelihood for  
HMM

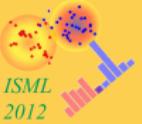
Forward-Backward  
HMM

Conditional  
Independence

Alpha-Beta HMM

How to train a HMM  
using EM?

HMM - Viterbi Algorithm



*A Simple Example*

*Maximum Likelihood for  
HMM*

*Forward-Backward  
HMM*

*Conditional  
Independence*

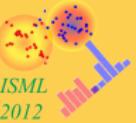
*Alpha-Beta HMM*

*How to train a HMM  
using EM?*

*HMM - Viterbi Algorithm*

# Forward-Backward HMM

- Need to efficiently evaluate the  $\gamma(z_{nk})$  and  $\xi(z_{n-1,j}, z_{nk})$ .
- The graphical model for the HMM is a tree!
- We know we can use a two-stage message passing algorithm to calculate the posterior distribution of the latent variables.
- For HMM this is called the **forward-backward** algorithm (Rabiner, 1989), or **Baum-Welch** algorithm (Baum, 1972).
- Other variants exist, different only in the form of the messages propagated.
- We look at the most widely known, the **alpha-beta** algorithm.

*A Simple Example**Maximum Likelihood for  
HMM**Forward-Backward  
HMM**Conditional  
Independence**Alpha-Beta HMM**How to train a HMM  
using EM?**HMM - Viterbi Algorithm*

# Conditional Independence for HMM

Given the data  $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  the following independence relations hold:

$$p(\mathbf{X} | \mathbf{z}_n) = p(\mathbf{x}_1, \dots, \mathbf{x}_n | \mathbf{z}_n) p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N | \mathbf{z}_n)$$

$$p(\mathbf{x}_1, \dots, \mathbf{x}_{n-1} | \mathbf{x}_n, \mathbf{z}_n) = p(\mathbf{x}_1, \dots, \mathbf{x}_{n-1} | \mathbf{z}_n)$$

$$p(\mathbf{x}_1, \dots, \mathbf{x}_{n-1} | \mathbf{z}_{n-1}, \mathbf{z}_n) = p(\mathbf{x}_1, \dots, \mathbf{x}_{n-1} | \mathbf{z}_{n-1})$$

$$p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N | \mathbf{z}_n, \mathbf{z}_{n+1}) = p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N | \mathbf{z}_{n+1})$$

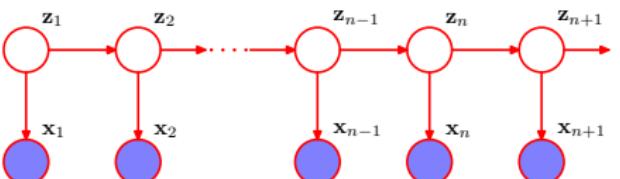
$$p(\mathbf{x}_{n+2}, \dots, \mathbf{x}_N | \mathbf{x}_{n+1}, \mathbf{z}_{n+1}) = p(\mathbf{x}_{n+2}, \dots, \mathbf{x}_N | \mathbf{z}_{n+1})$$

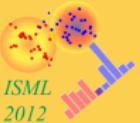
$$p(\mathbf{X} | \mathbf{z}_{n-1}, \mathbf{z}_n) = p(\mathbf{x}_1, \dots, \mathbf{x}_{n-1} | \mathbf{z}_{n-1}) p(\mathbf{x}_n | \mathbf{x}_n)$$

$$p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N | \mathbf{z}_n)$$

$$p(\mathbf{x}_{N+1} | \mathbf{X}, \mathbf{z}_{N+1}) = p(\mathbf{x}_{N+1} | \mathbf{z}_{N+1})$$

$$p(\mathbf{z}_{n+1} | \mathbf{X}, \mathbf{z}_N) = p(\mathbf{z}_{N+1} | \mathbf{z}_N)$$



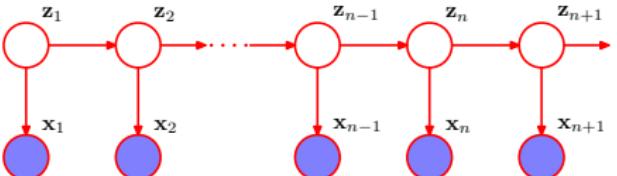
*A Simple Example**Maximum Likelihood for  
HMM**Forward-Backward  
HMM**Conditional  
Independence**Alpha-Beta HMM**How to train a HMM  
using EM?**HMM - Viterbi Algorithm*

# Conditional Independence for HMM - Example

- Let's look at the following independence relation:

$$p(\mathbf{X} | \mathbf{z}_n) = p(\mathbf{x}_1, \dots, \mathbf{x}_n | \mathbf{z}_n) p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N | \mathbf{z}_n)$$

- Any path from the set  $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$  to the set  $\{\mathbf{x}_{n+1}, \dots, \mathbf{x}_N\}$  has to go through  $\mathbf{z}_n$ .
- In  $p(\mathbf{X} | \mathbf{z}_n)$  the node  $\mathbf{z}_n$  is conditioned on (= observed).
- All paths through  $\mathbf{z}_n$  are head-tail.
- Therefore all paths through  $\mathbf{z}_n$  are blocked.





# Alpha-Beta HMM

- Define the joint probability of observing all data up to step  $n$  and having  $\mathbf{z}_n$  as latent variable to be

$$\alpha(\mathbf{z}_n) = p(\mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{z}_n).$$

- Define the probability of all future data given  $\mathbf{z}_n$  to be

$$\beta(\mathbf{z}_n) = p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N \mid \mathbf{z}_n).$$

- Then it can be shown the following recursions hold

$$\alpha(\mathbf{z}_n) = p(\mathbf{x}_n \mid \mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \alpha(\mathbf{z}_{n-1}) p(\mathbf{z}_n \mid \mathbf{z}_{n-1})$$

$$\alpha(\mathbf{z}_1) = \prod_{k=1}^K \{\pi_k p(\mathbf{x}_1 \mid \phi_k)\}^{z_{1k}}$$

A Simple Example

Maximum Likelihood for  
HMM

Forward-Backward  
HMM

Conditional  
Independence

Alpha-Beta HMM

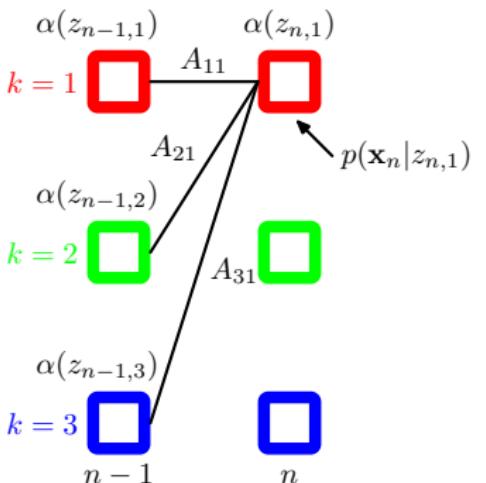
How to train a HMM  
using EM?

HMM - Viterbi Algorithm



- At step  $n$  we can efficiently calculate  $\alpha(\mathbf{z}_n)$  given  $\alpha(\mathbf{z}_{n-1})$

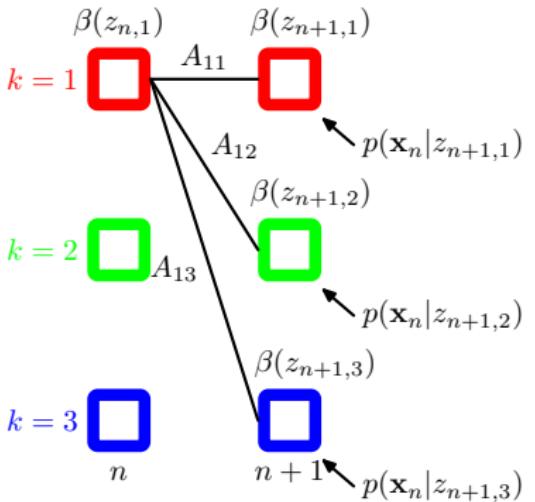
$$\alpha(\mathbf{z}_n) = p(\mathbf{x}_n | \mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \alpha(\mathbf{z}_{n-1}) p(\mathbf{z}_n | \mathbf{z}_{n-1})$$

*A Simple Example**Maximum Likelihood for HMM**Forward-Backward HMM**Conditional Independence**Alpha-Beta HMM**How to train a HMM using EM?**HMM - Viterbi Algorithm*



- And for  $\beta(\mathbf{z}_n)$  we get the recursion

$$\beta(\mathbf{z}_n) = \sum_{\mathbf{z}_{n+1}} \beta(\mathbf{z}_{n+1}) p(\mathbf{x}_{n+1} | \mathbf{z}_{n+1}) p(\mathbf{z}_{n+1} | \mathbf{z}_n)$$



A Simple Example

Maximum Likelihood for  
HMMForward-Backward  
HMMConditional  
Independence

Alpha-Beta HMM

How to train a HMM  
using EM?

HMM - Viterbi Algorithm



A Simple Example

Maximum Likelihood for  
HMMForward-Backward  
HMMConditional  
Independence

Alpha-Beta HMM

How to train a HMM  
using EM?

HMM - Viterbi Algorithm

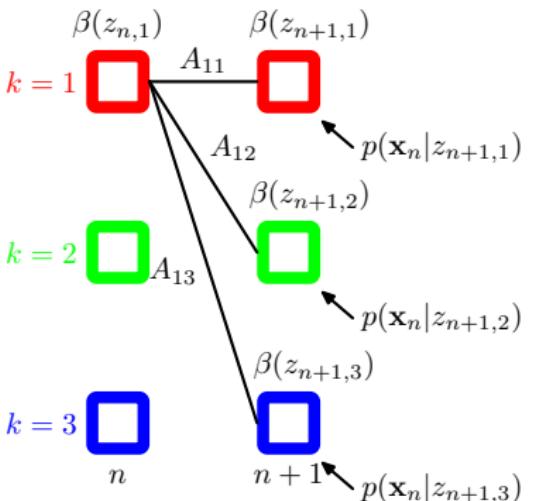
# Alpha-Beta HMM

- How do we start the  $\beta$  recursion? What is  $\beta(\mathbf{z}_N)$  ?

$$\beta(\mathbf{z}_N) = p(\mathbf{x}_{N+1}, \dots, \mathbf{x}_N \mid \mathbf{z}_n).$$

- Can be shown the following is consistent with the approach

$$\beta(\mathbf{z}_N) = 1.$$



*A Simple Example**Maximum Likelihood for  
HMM**Forward-Backward  
HMM**Conditional  
Independence**Alpha-Beta HMM**How to train a HMM  
using EM?**HMM - Viterbi Algorithm*

# Alpha-Beta HMM

- Now we know how to calculate  $\alpha(\mathbf{z}_n)$  and  $\beta(\mathbf{z}_n)$  for each step.
- What is the probability of the data  $p(\mathbf{X})$  ?
- Use the definition of  $\gamma(\mathbf{z}_n)$  and Bayes

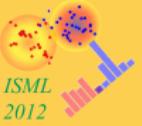
$$\gamma(\mathbf{z}_n) = p(\mathbf{z}_n \mid \mathbf{X}) = \frac{p(\mathbf{X} \mid \mathbf{z}_n) p(\mathbf{z}_n)}{p(\mathbf{X})} = \frac{p(\mathbf{X}, \mathbf{z}_n)}{p(\mathbf{X})}$$

- and the following conditional independence statement from the graphical model of the HMM

$$p(\mathbf{X} \mid \mathbf{z}_n) = \underbrace{p(\mathbf{x}_1, \dots, \mathbf{x}_n \mid \mathbf{z}_n)}_{\alpha(\mathbf{z}_n)/p(\mathbf{z}_n)} \underbrace{p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N \mid \mathbf{z}_n)}_{\beta(\mathbf{z}_n)}$$

- and therefore

$$\gamma(\mathbf{z}_n) = \frac{\alpha(\mathbf{z}_n)\beta(\mathbf{z}_n)}{p(\mathbf{X})}$$



A Simple Example

Maximum Likelihood for  
HMMForward-Backward  
HMMConditional  
Independence

Alpha-Beta HMM

How to train a HMM  
using EM?

HMM - Viterbi Algorithm

# Alpha-Beta HMM

- Marginalising over  $\mathbf{z}_n$  results in

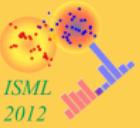
$$1 = \sum_{\mathbf{z}_n} \gamma(\mathbf{z}_n) = \frac{\sum_{\mathbf{z}_n} \alpha(\mathbf{z}_n)\beta(\mathbf{z}_n)}{p(\mathbf{X})}$$

and therefore at each step  $n$

$$p(\mathbf{X}) = \sum_{\mathbf{z}_n} \alpha(\mathbf{z}_n)\beta(\mathbf{z}_n)$$

- Most conveniently evaluated at step  $N$  where  $\beta(\mathbf{z}_N) = 1$  as

$$p(\mathbf{X}) = \sum_{\mathbf{z}_N} \alpha(\mathbf{z}_N).$$

ISML  
2012

A Simple Example

Maximum Likelihood for  
HMMForward-Backward  
HMMConditional  
Independence

Alpha-Beta HMM

How to train a HMM  
using EM?

HMM - Viterbi Algorithm

# Alpha-Beta HMM

- Finally, we need to calculate the joint posterior distribution of two successive latent variables by  $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$  defined by

$$\xi(\mathbf{z}_{n-1}, \mathbf{z}_n) = p(\mathbf{z}_{n-1}, \mathbf{z}_n \mid \mathbf{X}).$$

- This can be done directly from the  $\alpha$  and  $\beta$  values in the form

$$\xi(\mathbf{z}_{n-1}, \mathbf{z}_n) = \frac{\alpha(\mathbf{z}_{n-1}) p(\mathbf{x}_n \mid \mathbf{z}_n) p(\mathbf{z}_n \mid \mathbf{z}_{n-1}) \beta(\mathbf{z}_n)}{p(\mathbf{X})}$$

[A Simple Example](#)[Maximum Likelihood for  
HMM](#)[Forward-Backward  
HMM](#)[Conditional  
Independence](#)[Alpha-Beta HMM](#)[How to train a HMM  
using EM?](#)[HMM - Viterbi Algorithm](#)

# How to train a HMM using EM?

- 1 Make an initial selection for the parameters  $\theta^{\text{old}}$  where  $\theta = \{\pi, \mathbf{A}, \phi\}$ . (Often,  $\mathbf{A}$ ,  $\pi$  initialised uniformly or randomly. The  $\phi_k$ -initialisation depends on the emission distribution; for Gaussians run  $K$ -means first and get  $\mu_k$  and  $\Sigma_k$  from there.)
- 2 (Start of E-step) Run forward recursion to calculate  $\alpha(\mathbf{z}_n)$ .
- 3 Run backward recursion to calculate  $\beta(\mathbf{z}_n)$ s.
- 4 Calculate  $\gamma(\mathbf{z})$  and  $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$  from  $\alpha(\mathbf{z}_n)$  and  $\beta(\mathbf{z}_n)$ .
- 5 Evaluate the likelihood  $p(\mathbf{Z})$ .
- 6 (Start of M-step) Find a  $\theta^{\text{new}}$  maximising  $Q(\theta, \theta^{\text{old}})$ . This results in new settings for the parameters  $\pi_k, A_{jk}$  and  $\phi_k$  as described before.
- 7 Iterate until convergence is detected.

# Alpha-Beta HMM - Notes



- In order to calculate the likelihood, we need to use the joint probability  $p(\mathbf{X}, \mathbf{Z})$  and sum over all possible values of  $\mathbf{Z}$ .
- That means, every particular choice of  $\mathbf{Z}$  corresponds to one path through the lattice diagram. There are exponentially many !
- Using the alpha-beta algorithm, the exponential cost has been reduced to a linear cost in the length of the model.
- How did we do that?
- Swapping the order of multiplication and summation.

*A Simple Example*

*Maximum Likelihood for  
HMM*

*Forward-Backward  
HMM*

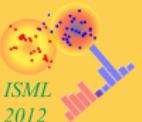
*Conditional  
Independence*

*Alpha-Beta HMM*

*How to train a HMM  
using EM?*

*HMM - Viterbi Algorithm*

# HMM - Viterbi Algorithm



- Motivation: The latent states can have some meaningful interpretation, e.g. phonems in a speech recognition system where the observed variables are the acoustic signals.
- Goal: After the system has been trained, find the most probable states of the latent variables for a given sequence of observations.
- Warning: Finding the set of states which are each individually the most probable does NOT solve this problem.

*A Simple Example*

*Maximum Likelihood for HMM*

*Forward-Backward HMM*

*Conditional  
Independence*

*Alpha-Beta HMM*

*How to train a HMM  
using EM?*

*HMM - Viterbi Algorithm*



A Simple Example

Maximum Likelihood for  
HMMForward-Backward  
HMMConditional  
Independence

Alpha-Beta HMM

How to train a HMM  
using EM?

HMM - Viterbi Algorithm

# HMM - Viterbi Algorithm

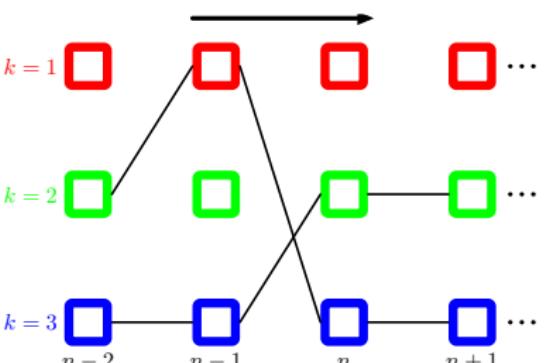
- Define  $\omega(\mathbf{z}_n) = \max_{\mathbf{z}_1, \dots, \mathbf{z}_{n-1}} \ln p(\mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{z}_1, \dots, \mathbf{z}_n)$
- From the joint distribution of the HMM given by

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{z}_1, \dots, \mathbf{z}_N) = p(\mathbf{z}_1) \left[ \prod_{n=2}^N p(\mathbf{z}_n | \mathbf{z}_{n-1}) \right] \prod_{n=1}^N p(\mathbf{x}_n | \mathbf{z}_n)$$

the following recursion can be derived

$$\omega(\mathbf{z}_n) = \ln p(\mathbf{x}_n | \mathbf{z}_n) + \max_{\mathbf{z}_{n-1}} \{\ln p(\mathbf{z}_n | \mathbf{z}_{n-1}) + \omega(\mathbf{z}_{n-1})\}$$

$$\omega(\mathbf{z}_1) = \ln p(\mathbf{z}_1) + \ln p(\mathbf{x}_1 | \mathbf{z}_1) = \ln p(\mathbf{x}_1, \mathbf{z}_1)$$



*A Simple Example**Maximum Likelihood for  
HMM**Forward-Backward  
HMM**Conditional  
Independence**Alpha-Beta HMM**How to train a HMM  
using EM?**HMM - Viterbi Algorithm*

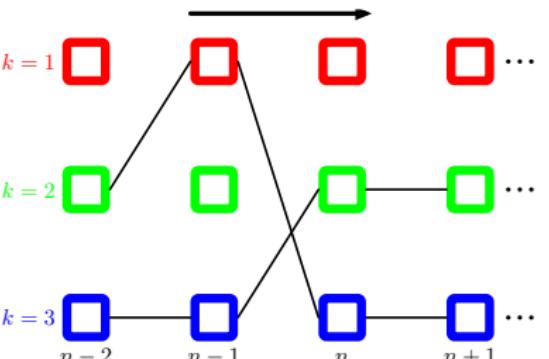
# HMM - Viterbi Algorithm

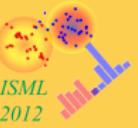
- Calculate

$$\omega(\mathbf{z}_n) = \max_{\mathbf{z}_1, \dots, \mathbf{z}_{n-1}} \ln p(\mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{z}_1, \dots, \mathbf{z}_n)$$

for  $n = 1, \dots, N$ .

- For each step  $n$  remember which is the best transition to go into each state at the next step.
- At step  $N$  : Find the state with the highest probability.
- For  $n = 1, \dots, N - 1$ : Backtrace which transition led to the most probable state and identify from which state it came.





## Part XXIII

### *Combining Models*

*Motivation*

*Model Combination vs.  
Bayesian Model  
Averaging*

*Committees*

*Boosting*

*Tree-based Models*

*Conditional Mixture  
Models*



Motivation

Model Combination vs.  
Bayesian Model  
Averaging

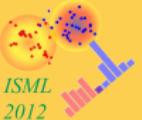
Committees

Boosting

Tree-based Models

Conditional Mixture  
Models

- Coming up with a very precise prediction rule can be very hard.
- It may be easier to come up with a number of not so precise prediction rules.
- Can combine them to make better predictions?
- Example: A team of people often performs better than an individual.
- Bayesian Averaging versus Model Combination.
- Tree based methods.



Motivation

Model Combination vs.  
Bayesian Model  
Averaging

Committees

Boosting

Tree-based Models

Conditional Mixture  
Models

# Model Combination

- Review density estimation via a mixture of Gaussian.
- Model specified via the joint distribution

$$p(\mathbf{x}, \mathbf{z})$$

with  $\mathbf{x}$  the observed variable, and  $\mathbf{z}$  the unobserved variable.

- Probability over the observed variable is found by marginalisation

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}).$$



- For the Gaussian mixture we get for each observed data point  $p(\mathbf{x})$

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

- If all the data are i.i.d, we can find the probability of all data  $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$

$$p(\mathbf{X}) = \prod_{n=1}^N p(\mathbf{x}_n) = \prod_{n=1}^N \left[ \sum_{\mathbf{z}_n} p(\mathbf{x}_n, \mathbf{z}_n) \right].$$

Motivation

Model Combination vs.  
Bayesian Model  
Averaging

Committees

Boosting

Tree-based Models

Conditional Mixture  
Models



Motivation

Model Combination vs.  
Bayesian Model  
Averaging

Committees

Boosting

Tree-based Models

Conditional Mixture  
Models

# Bayesian Model Averaging

- Now suppose different models index by  $h = 1, \dots, H$  with prior probabilities  $p(h)$ . (One model might be a mixture of Gaussian, another model a mixture of Cauchy distributions.)
- Now the marginal distribution over the data set  $\mathbf{X}$  can be expressed as **Bayesian Model Averaging**

$$p(\mathbf{X}) = \sum_{h=1}^H p(\mathbf{X} | h) p(h).$$

- Interpretation: Just **one** model is responsible for creating the data  $\mathbf{X}$ , but we do not know which one. Therefore  $p(h)$  reflects the uncertainty of which model is responsible.
- If more and more data become available, the posterior probability  $p(h | \mathbf{X})$  will become increasingly focussed on just one model.



Motivation

Model Combination vs.  
Bayesian Model  
Averaging

Committees

Boosting

Tree-based Models

Conditional Mixture  
Models

# Model Combination vs. Bayesian Model Averaging

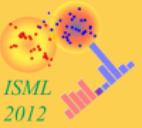
- **Bayesian Model Averaging** : One model is responsible for generating all data.

$$p(\mathbf{X}) = \sum_{h=1}^H p(\mathbf{X} | h) p(h).$$

- **Combine Multiple Models** : Different data points can be created from different values of the latent variable, and therefore by different components.

$$p(\mathbf{X}) = \prod_{n=1}^N p(\mathbf{x}_n) = \prod_{n=1}^N \left[ \sum_{\mathbf{z}_n} p(\mathbf{x}_n, \mathbf{z}_n) \right].$$

- Same applies to predictive density  $p(\mathbf{x} | \mathbf{X})$  or conditional distributions such as  $p(\mathbf{t} | \mathbf{x}, \mathbf{X}, \mathbf{T})$  instead of  $p(\mathbf{x})$ .



Motivation

Model Combination vs.  
Bayesian Model  
Averaging

Committees

Boosting

Tree-based Models

Conditional Mixture  
Models

# Committees

- Construct a **committee** by averaging the predictions of a set of individual models.
- Remember training multiple polynomials using the sinusoidal data, and then averaging?
- Using low-bias models (higher order polynomials) and averaging resulted in better prediction.



Motivation

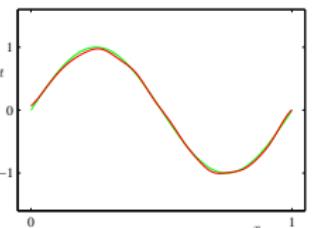
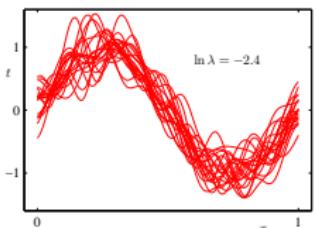
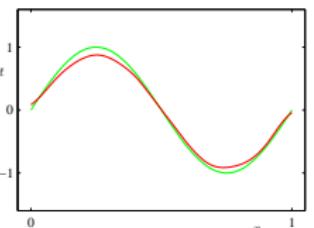
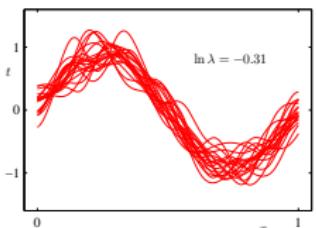
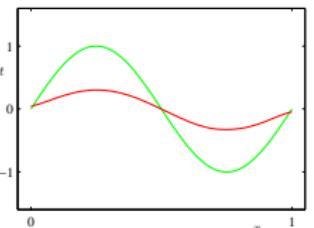
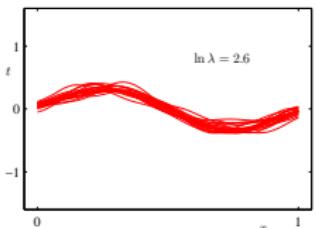
Model Combination vs.  
Bayesian Model  
Averaging

Committees

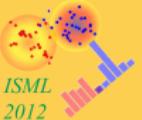
Boosting

Tree-based Models

Conditional Mixture  
Models



Left: Result of fitting the model to 100 data sets, only 25 shown.  
Right: Average of the 100 fits in red, the sinusoidal function



Motivation

Model Combination vs.  
Bayesian Model  
Averaging

Committees

Boosting

Tree-based Models

Conditional Mixture  
Models

# Where does the variability come from?

- Only one data set available.
- How to create new data sets?
- Use for instance **bootstrap** approach: Given  $N$  data points  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ , draw a set of  $N$  points from this data set with replacement. The new set  $\mathbf{X}'$  may contain some of the data  $\mathbf{x}_n$  multiple times, others may be absent.
- Created  $M$  'artificial' data sets and train several predictive models  $y_m(\mathbf{x})$  where  $m = 1, \dots, M$ .
- The **committee prediction**  $y_{\text{COM}}$  is then given by

$$y_{\text{COM}} = \frac{1}{M} \sum_{m=1}^M y_m(\mathbf{x})$$

- Also called **bootstrap aggregation** or **bagging**.



Motivation

Model Combination vs.  
Bayesian Model  
Averaging

Committees

Boosting

Tree-based Models

Conditional Mixture  
Models

# How big is the error?

- Suppose true regression function is given by  $h(\mathbf{x})$ .
- Write the output of each model as sum of  $h(\mathbf{x})$  and some model dependend error  $\epsilon_m(\mathbf{x})$ .

$$y_m(\mathbf{x}) = h(\mathbf{x}) + \epsilon_m(\mathbf{x})$$

- Average sum-of-squares error by each model individually is

$$\mathbb{E}_{\mathbf{x}} [\{y_m(\mathbf{x}) - h(\mathbf{x})\}^2] = \mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x})^2]$$

where  $\mathbb{E}_{\mathbf{x}} [\cdot]$  is the expectation with respect to the distribution of the data  $\mathbf{x}$ .

- Average by all models acting individually

$$E_{AV} = \frac{1}{M} \sum_{m=1}^M \mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x})^2]$$



Motivation

Model Combination vs.  
Bayesian Model  
Averaging

Committees

Boosting

Tree-based Models

Conditional Mixture  
Models

# How big is the error?

- Average by all models acting individually

$$E_{AV} = \frac{1}{M} \sum_{m=1}^M \mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x})^2]$$

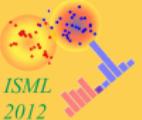
- But the **expected error from the committee** is

$$E_{COM} = \mathbb{E}_{\mathbf{x}} \left[ \left\{ \frac{1}{M} \sum_{m=1}^M y_m(\mathbf{x}) - h(\mathbf{x}) \right\}^2 \right] = \mathbb{E}_{\mathbf{x}} \left[ \left\{ \frac{1}{M} \sum_{m=1}^M \epsilon_m(\mathbf{x}) \right\}^2 \right]$$

- Assume errors have zero mean  $\mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x})] = 0$  and are uncorrelated  $\mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x}) \epsilon_l(\mathbf{x})] = 0$  for  $m \neq l$ .
- 'Dramatic' error reduction by using a committee.

$$E_{COM} = \frac{1}{M} E_{AV}$$

- Unfortunately, not true.** Why?



## Motivation

Model Combination vs.  
Bayesian Model  
Averaging

## Committees

## Boosting

## Tree-based Models

Conditional Mixture  
Models

# How big is the error?

- ‘Dramatic’ error reduction by using a committee.

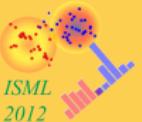
$$E_{\text{COM}} = \frac{1}{M} E_{\text{AV}}$$

- Unfortunately, not true. Why?
- The errors are typically highly correlated → the reduction in overall error is generally small.
- However, can show that

$$E_{\text{COM}} \leq E_{\text{AV}}$$

- Expected committee error does not exceed the expected average error of the individual models.

# *Learning from Learners*



Motivation

Model Combination vs.  
Bayesian Model  
Averaging

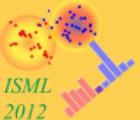
Committees

Boosting

Tree-based Models

Conditional Mixture  
Models

- In the committee method, the individual models did not know of each other. They do not learn from each other.
- How can models learn from each other?
- For example: Learn the models in sequence and use some information from learning one model in the learning process for the next model.
- How?
- For example: Tell the next model learning process which data points the current model did not learn well. These data points can then be focused on in the next learning process.



Motivation

Model Combination vs.  
Bayesian Model  
Averaging

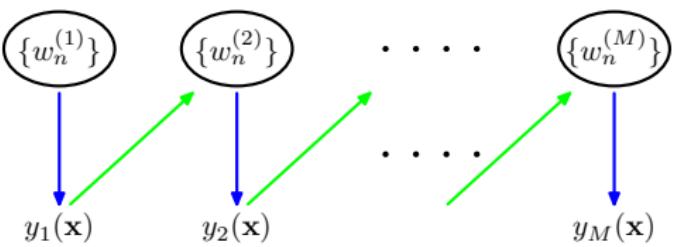
Committees

Boosting

Tree-based Models

Conditional Mixture  
Models

- Assume ‘base’ classifiers (also called **weak learners**).
- AdaBoost** (adaptive boosting): Points that are misclassified by one of the base classifiers are given greater weight when used to train the next classifier in the sequence.
- Combine the predictions of all the classifiers through a majority voting scheme.



$$Y_M(\mathbf{x}) = \text{sign} \left( \sum_m^M \alpha_m y_m(\mathbf{x}) \right)$$



Motivation

Model Combination vs.  
Bayesian Model  
Averaging

Committees

Boosting

Tree-based Models

Conditional Mixture  
Models

# Boosting - AdaBoost (for classification)

- ➊ Initialise the data weighting coefficients  $\{w_n\}$  to  $w_n^{(1)} = 1/N$ .
- ➋ For each model  $m = 1, \dots, M$ :
  - ➌ Fit classifier  $y_m(\mathbf{x})$ .
  - ➍ Compute the model weights  $\alpha_m$ .
  - ➎ Evaluate weighting coefficients  $w_n^{(n+1)}$ .
- ➏ Make predictions using the final model

$$Y_M(\mathbf{x}) = \text{sign} \left( \sum_{m=1}^M \alpha_m y_m(\mathbf{x}) \right).$$



Motivation

Model Combination vs.  
Bayesian Model  
Averaging

Committees

Boosting

Tree-based Models

Conditional Mixture  
Models

# Boosting - AdaBoost (for classification)

For each model  $m = 1, \dots, M$ :

- ➊ Fit classifier  $y_m(\mathbf{x})$  by minimising the weighted error function

$$J_m = \sum_{n=1}^N w_n^{(n)} \mathbb{I}(y_m(\mathbf{x}) \neq t_n)$$

where  $\mathbb{I}(y_m(\mathbf{x}) \neq t_n)$  is the indicator function which equals 1 if  $y_m(\mathbf{x}) \neq t_n$  and is 0 otherwise.

- ➋ Compute the model weights  $\alpha_m$  by calculating

$$\epsilon_m = \frac{\sum_{n=1}^N w_n^{(n)} \mathbb{I}(y_m(\mathbf{x}) \neq t_n)}{\sum_{n=1}^N w_n^{(n)}} \quad \text{and} \quad \alpha_m = \ln \left\{ \frac{1 - \epsilon_m}{\epsilon_m} \right\}.$$

- ➌ Evaluate weighting coefficients  $w_n^{(n+1)}$  as

$$w_n^{(n+1)} = w_n^{(n)} \exp \{ \alpha_m \mathbb{I}(y_m(\mathbf{x}) \neq t_n) \}.$$



Motivation

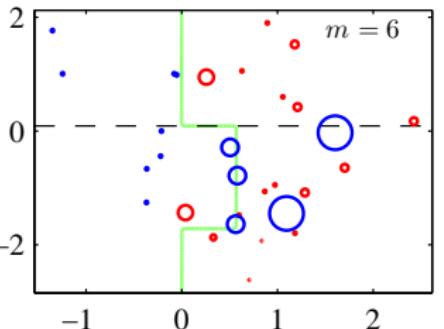
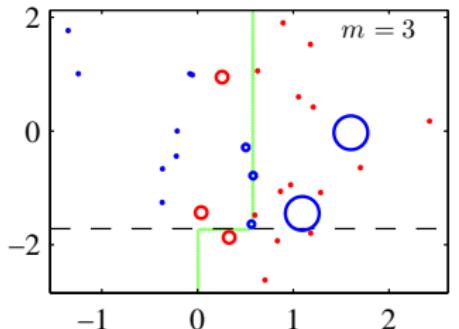
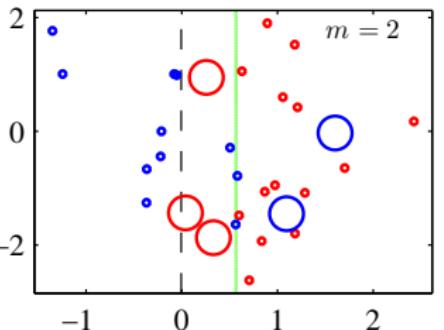
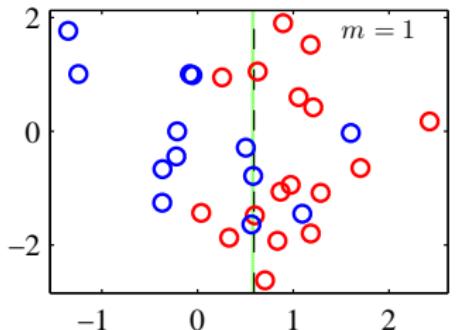
Model Combination vs.  
Bayesian Model  
Averaging

Committees

Boosting

Tree-based Models

Conditional Mixture  
Models



Decision boundaries: last learner (black), ensemble (green).  
Data point weights of most recent learner (circle sizes).



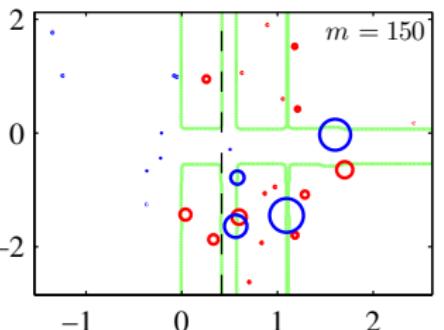
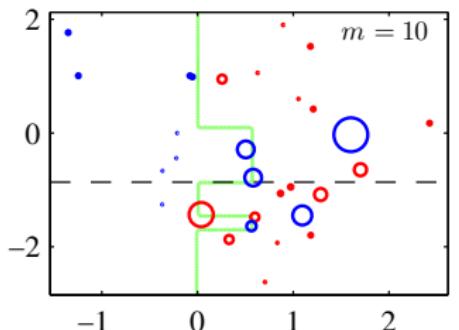
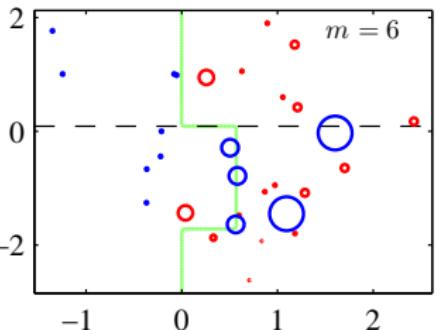
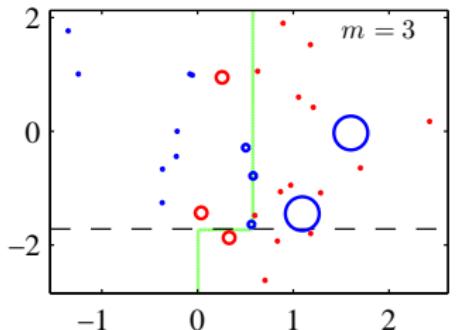
Motivation

Model Combination vs.  
Bayesian Model  
Averaging

Committees

Boosting

Tree-based Models

Conditional Mixture  
Models

Decision boundaries: last learner (black), ensemble (green).  
Data point weights of most recent learner (circle sizes).



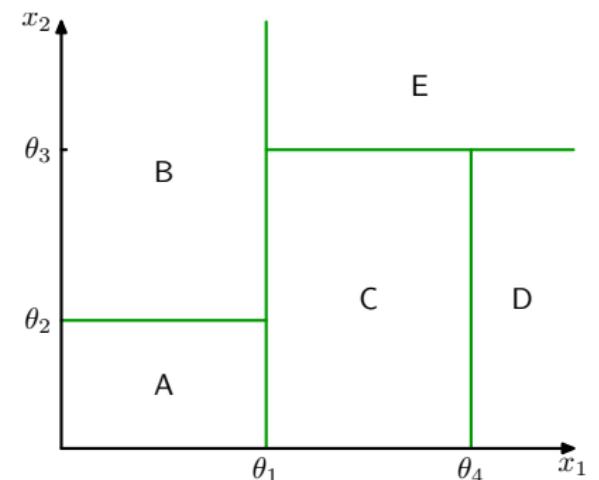
Motivation

Model Combination vs.  
Bayesian Model  
Averaging

Committees

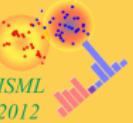
Boosting

Tree-based Models

Conditional Mixture  
Models

# Tree-based Models

- Idea: Partition the input space into cuboid regions (edges aligned with the axes).
- Assign a model to each of the regions.
- Can be viewed as model combination where only one model is responsible for making predictions at each point in input space.
- Which model is responsible for the prediction given a new data point? Traverse the binary tree.



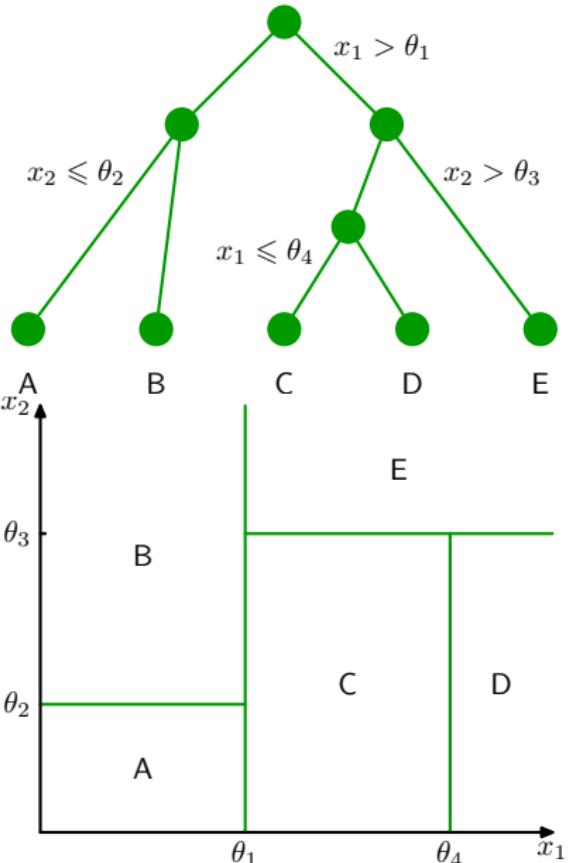
Motivation

Model Combination vs.  
Bayesian Model  
Averaging

Committees

Boosting

Tree-based Models

Conditional Mixture  
Models



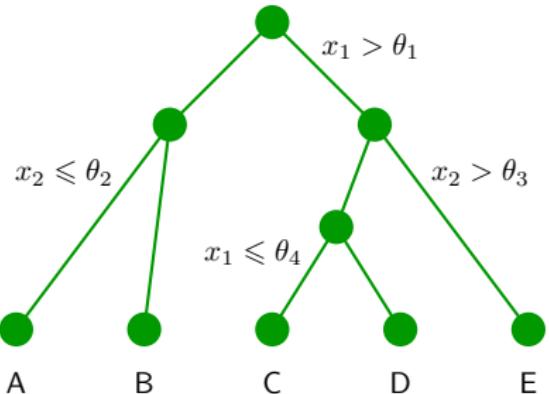
Motivation

Model Combination vs.  
Bayesian Model  
Averaging

Committees

Boosting

Tree-based Models

Conditional Mixture  
Models

Note: Decision trees are NOT probabilistic graphical models!



## Motivation

Model Combination vs.  
Bayesian Model  
Averaging

Committees

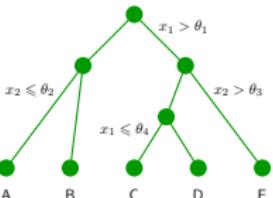
Boosting

Tree-based Models

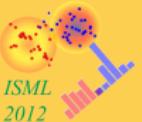
Conditional Mixture  
Models

# Classification and Regression Trees (CART)

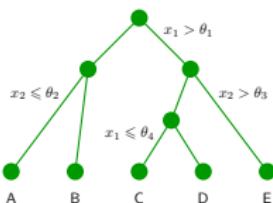
- How to learn the structure of the tree?
- Given training data  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  with each  $\mathbf{x}_n \in \mathbb{R}^D$ , and training targets  $\{t_1, \dots, t_N\}$ .
- Goal: Predict the target  $t$  for a new data point  $\mathbf{x}$ .
- If the partitioning of the input space is given, and we minimise the sum-of-squares error function, then the optimal prediction value in each region is the average of the training targets  $t_n$  in this regions.
- Training: For each node in the tree, we have to choose
  - an input variable to split on, and
  - a corresponding threshold
 such that the sum-of-squares error is minimised.
- **Combinatorial explosion!** Infeasible!



# Classification and Regression Trees (CART)



- Use greedy algorithm.
- Start with a single root node.
- For each new training data point, consider only regions which could be potentially split.
- Choose the split and threshold which minimises the sum-of-squares error.
- Local optimisation problem.
- When to stop?



Motivation

Model Combination vs.  
Bayesian Model  
Averaging

Committees

Boosting

Tree-based Models

Conditional Mixture  
Models

# Classification and Regression Trees (CART)



Motivation

Model Combination vs.  
Bayesian Model  
Averaging

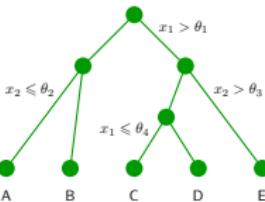
Committees

Boosting

Tree-based Models

Conditional Mixture  
Models

- Stop when there is no more reduction in residual error.
- However : empirically found that although none of the current splits reduces the error anymore, later splits can produce much smaller errors.
- Therefore: Grow a large tree (stopping criterion based only on the number of points associated with each leaf node). Then prune the tree.



*Motivation**Model Combination vs.  
Bayesian Model  
Averaging**Committees**Boosting**Tree-based Models**Conditional Mixture  
Models*

# Classification and Regression Trees (CART)

- Start pruning with tree  $T_0$ .
- Prune into tree  $T \subset T_0$  by collapsing internal nodes.
- Suppose the leaf nodes of  $T$  are indexed by  $\tau = 1, \dots, |T|$ .
- Optimal prediction for region  $\mathcal{R}$  is given by

$$y_\tau = \frac{1}{N_\tau} \sum_{\mathbf{x} \in \mathcal{R}} t_n.$$

- Corresponding contribution to the residual sum-of-squares

$$Q_\tau(T) = \sum_{\mathbf{x}_n \in \mathcal{R}} \{t_n - y_\tau\}^2.$$

- Pruning criterion (regulariser  $\lambda$  from cross-validation)

$$C(T) = \sum_{\tau=1}^{|T|} Q_\tau(T) + \lambda|T|.$$



Motivation

Model Combination vs.  
Bayesian Model  
Averaging

Committees

Boosting

Tree-based Models

Conditional Mixture  
Models

# Classification and Regression Trees (CART)

- Similar growing and pruning for classification problems.
- Replace the sum-of-squares error with an appropriate measure of performance.
- Define  $p_{\tau k}$  to be the proportion of data points in region  $\mathcal{R}_\tau$  assigned to class  $k$ , where  $k = 1, \dots, K$ .
- For growing the tree use the differentiable
  - Cross-Entropy

$$Q_\tau(T) = \sum_{k=1}^K p_{\tau k} \ln p_{\tau k}$$

or

- Gini Index

$$Q_\tau(T) = \sum_{k=1}^K p_{\tau k} (1 - p_{\tau k}).$$

- Both vanish for  $p_{\tau k} = 0$  and  $p_{\tau k} = 1$ , and have a maximum at  $p_{\tau k} = 0.5$ .
- For pruning the tree, use the misclassification rate.

# *Classification and Regression Trees (CART)*



Motivation

Model Combination vs.  
Bayesian Model  
Averaging

Committees

Boosting

Tree-based Models

Conditional Mixture  
Models

- Human interpretability seen as major strength.
- However, structure is can be very sensitive to the details of the data set.(Small changes lead to very different splits.  
Order of the data presentation matters.)
- Splits are aligned with the axes. What happens if the decision boundary between two classes runs at 45 degrees to the axes? (Large number of splits necessary.)
- Each region in input space is associated with exactly one leaf. (Hard splits)
- Especially problematic for regression, where one often tries to model smooth functions. The CART only provides piecewise constant predictions with discontinuities at the split boundaries.

# Conditional Mixture Models

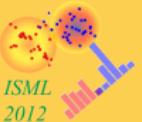
Overcoming the restrictions of Classification and Regression Trees (and loosing interpretability on the way :-)

(A) **Hierarchical Mixture of Experts** : A fully probabilistic tree-based model.

- Allow soft, probabilistic splits that can be functions of all the input variables, not just one at a time.
- Give leaf nodes a probabilistic interpretation.

(B) **Hierarchical Mixture of Experts** :

- Start with mixture of Gaussians (or other unconditional distributions).
- Replace the unconditional distribution by a conditional distribution conditioned on the input data.
- Create a **Mixture of Experts Model** by allowing the mixing coefficients to depend on the input data  $\pi_k \rightarrow \pi_k(\mathbf{x})$ .
- Allow each component in the mixture model to be itself a mixture model. (Recurse definition → tree.)



Motivation

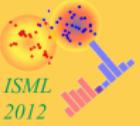
Model Combination vs.  
Bayesian Model  
Averaging

Committees

Boosting

Tree-based Models

Conditional Mixture  
Models



## Part XXIV

### *Selected Topics*

*Selected Topics*

*Occam's Razor*

*Reinforcement Learning*

*PageRank*

*Envelope Paradox*



- Occam's Razor
- Reinforcement Learning
- PageRank
- Envelope Paradox

*Selected Topics*

*Occam's Razor*

*Reinforcement Learning*

*PageRank*

*Envelope Paradox*



Selected Topics

Occam's Razor

Reinforcement Learning

PageRank

Envelope Paradox

# Occam's Razor

- Is there a unique principle which allows to formally arrive at predictions which
  - coincides (always?) with intuitive guesses or better?
  - which is (in some sense) most likely the best or correct answer?
- **Occam's Razor** : Use the simplest explanation which is consistent with the past data (and apply it for prediction).

William of Ockham (1288 - 1348):

"entities should not be multiplied unnecessarily."

He also produced significant works on logic, physics, and theology, e.g. tertiary logic.





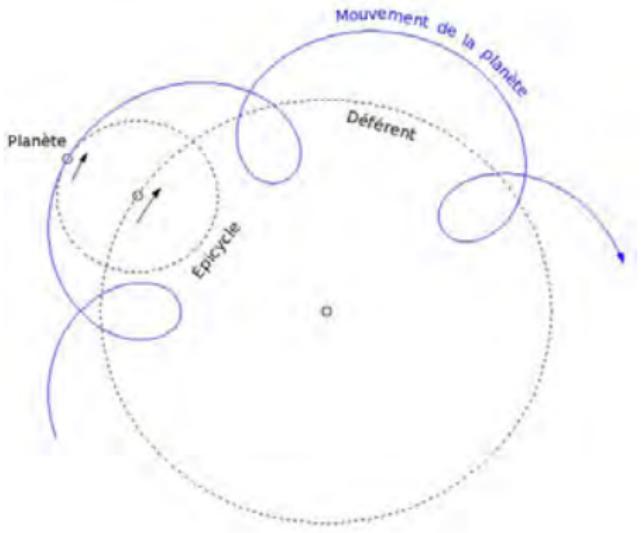
Selected Topics

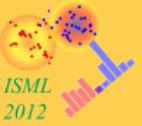
Occam's Razor

Reinforcement Learning

PageRank

Envelope Paradox





*Selected Topics*

*Occam's Razor*

*Reinforcement Learning*

*PageRank*

*Envelope Paradox*

# *Occam's Razor*

- **Occam's Razor** : Use the simplest explanation which is consistent with the past data (and apply it for prediction).
- Occam's razor can serve as a foundation of machine learning in general, and is even a fundamental principle (or maybe even the mere definition) of science.
- Karl Popper: a hypothesis, proposition, or theory is scientific only if it is falsifiable.
- A simple theory applies to more cases than a more complex one, and is thus more easily falsifiable.

# What is Simple?

- Occam's razor is not a formal/mathematical objective principle. What is simple for one may be complicated for someone else.
- Idea: Use a computer running programs of some language (say C or Python).
- Encode the data/theory etc. as a computer program.
- The complexity is the length of the shortest program encoding the data/theory.
- “**ababababababababab**” can be produced by the Python program ‘ab’ \* 10 (with length of 7 bytes); but to encode “**aidjendkwasuwemnduen**” we need more bytes (as I tried to hit the keyboard [almost] randomly when creating it).



Selected Topics

Occam's Razor

Reinforcement Learning

PageRank

Envelope Paradox



Selected Topics

Occam's Razor

Reinforcement Learning

PageRank

Envelope Paradox

# Kolmogorov Complexity

- Use a Turing machine  $M$ , and encode data  $x$  by the bitstring  $\langle M \rangle$  for the Turing machine itself and the bitstring  $w$  which is the program reproducing the data  $x$  when run with machine  $M$ .  $\langle M \rangle$  concatenated with  $w$  is called a **description** of the data  $x$ .
- The **Kolmogorov complexity** is the length of the **shortest** description of  $x$ .
- Powerful concept to describe the complexity of data in Algorithmic Information Theory.
- Uncomputable.
- 'Kolmogorov Complexity' was invented by Ray Solomonoff (1960) (Matthew effect: 'For to all those who have, more will be given, and they will have an abundance; but from those who have nothing, even what they have will be taken away.')



Selected Topics

Occam's Razor

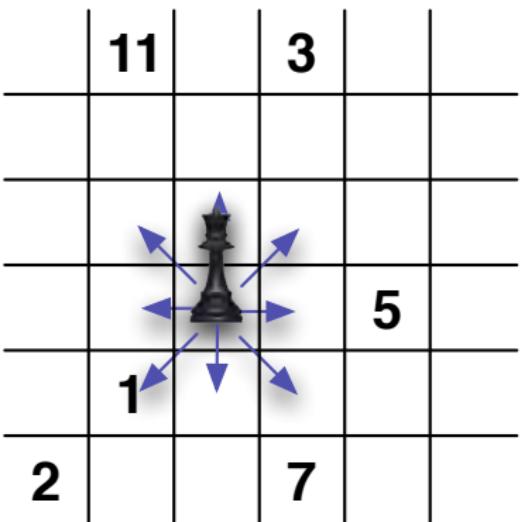
Reinforcement Learning

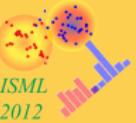
PageRank

Envelope Paradox

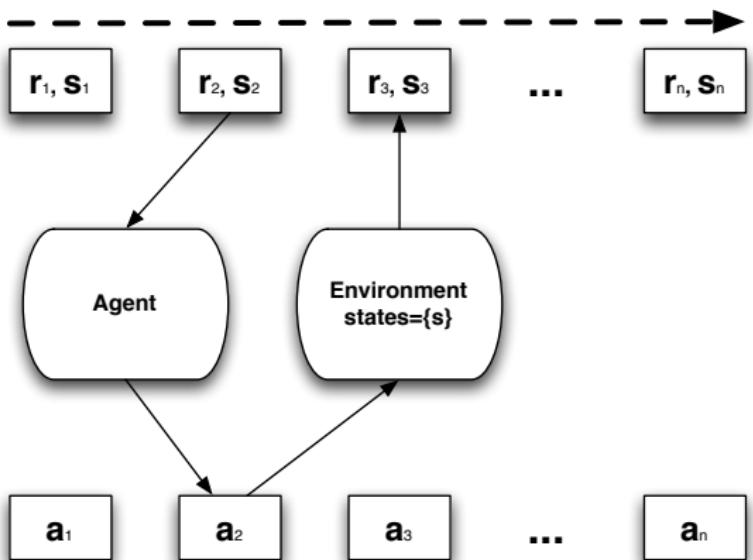
# Reinforcement Learning

- Reinforcement Learning is learning from interaction with an environment, from the consequences of action rather than from explicit teaching.
- Agent acts in an environment which is in state  $s$  by choosing an action  $a$  which leads to a change of state in the environment and a scalar reward  $r$  from the environment.





- Agent acts in an environment which is in state  $s$  by choosing an action  $a$  which leads to a change of state in the environment and a scalar reward  $r$  from the environment.



Selected Topics

Occam's Razor

Reinforcement Learning

PageRank

Envelope Paradox



Selected Topics

Occam's Razor

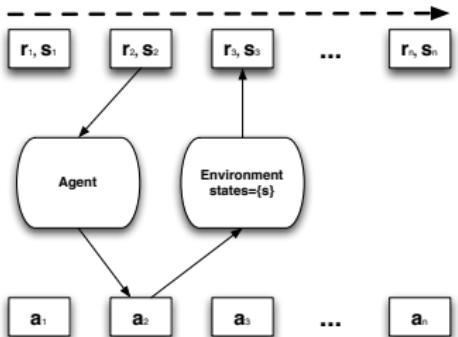
Reinforcement Learning

PageRank

Envelope Paradox

# Reinforcement Learning - Fully observable

- Agent receives some indication of the current state of the environment.
- Chooses action  $a$  as output.
- The action changes the state of the environment and the value of this state transition is communicated to the agent via a reward  $r$ .
- Goal: Maximise the sum of rewards  $r$ .
- How? Learn a policy maximising the sum of rewards.
- **Policy** : Mapping from states to actions.



# Reinforcement Learning

- Generally, the environment is **non-deterministic**: Taking the same action in the same state on two different occasions may result in different next states and rewards. This results in **transition probabilities** from one state to another state.
- No supervised output but delayed reward.
- Agent needs to gather information about the states, actions, transitions and rewards.
- Often **on-line** performance is important (robotics).
- **Exploration** versus **Exploitation**
- **Temporal Credit Assignment Problem**: How do we know whether the action taken now was good? (Wait until the 'end' if an 'end' exists?)
- Game playing
- Multiple agents



Selected Topics

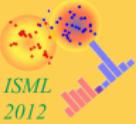
Occam's Razor

Reinforcement Learning

PageRank

Envelope Paradox

# *Reinforcement Learning - Partially observable*



- The agent does not know the state of the environment, but receives observations about it.
- Observations may be noisy.
- Simple strategy: Treat observations as states.
- Leads to **State free Stochastic Policies**: Mappings from observations to probability distributions over actions.
- POMDP: **Partially Observable Markov Decision Process**: Use a HMM to learn the hidden states of the environment from the observed data. (**Belief states**: Probability distributions over the states of the environment.)
- **Policy**: Now mapping from belief states into actions.

*Selected Topics*

*Occam's Razor*

*Reinforcement Learning*

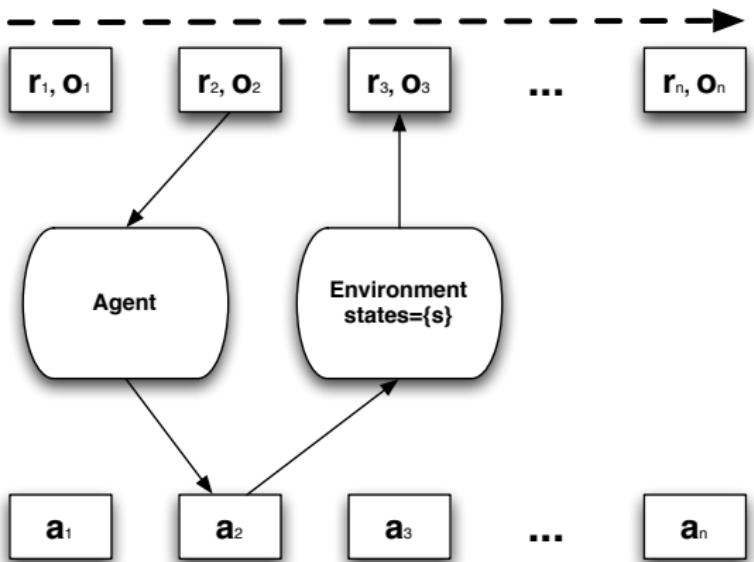
*PageRank*

*Envelope Paradox*

# Reinforcement Learning - Partially observable



- The agent does not know the state of the environment, but receives observations about it.



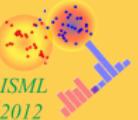
Selected Topics

Occam's Razor

Reinforcement Learning

PageRank

Envelope Paradox



Selected Topics

Occam's Razor

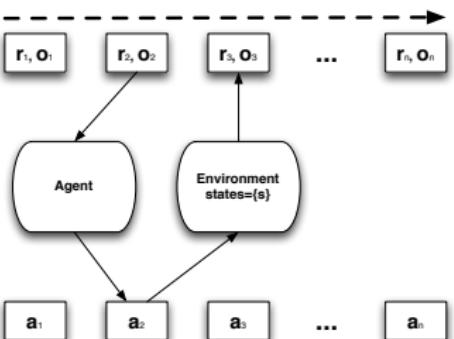
Reinforcement Learning

PageRank

Envelope Paradox

# Reinforcement Learning - Partially observable

- A Partially Observable Markov Decision Process is a generalization of a Markov Decision Process.
- A POMDP models an agent decision process in which it is assumed that the system dynamics are determined by an MDP, but the agent cannot directly observe the underlying state.
- Instead, it must infer a distribution over the state based on a model of the world and some local observations.
- Applications include robot navigation problems, machine maintenance, and planning under uncertainty in general.



# PageRank

- The problem : How to mechanically rank web pages by importance?
- Page, Lawrence and Brin, Sergey and Motwani, Rajeev and Winograd, Terry *"The PageRank Citation Ranking: Bringing Order to the Web."* Technical Report. Stanford InfoLab, 1999 (All graphics on the following slides taken from this report.)
- Citation from the paper: “To test the utility of PageRank for search, we built a web search engine called Google”
- Size of the Web

year	pages	comment
1998	150,000,000	from the paper
2008	1,000,000,000,000	one trillion

(2008 according to the official Google blog <http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html>)

- Not all pages are indexed in search engines. In fact, many are not.



Selected Topics

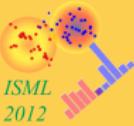
Occam's Razor

Reinforcement Learning

PageRank

Envelope Paradox

# Musing about magnitudes of numbers



- What a difference do 3 digits make?

1,000,000 seconds = 11.57 days

1,000,000,000 seconds = 31.71 years

1,000,000,000,000 seconds = 31,709.79 years

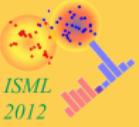
Selected Topics

Occam's Razor

Reinforcement Learning

PageRank

Envelope Paradox



Selected Topics

Occam's Razor

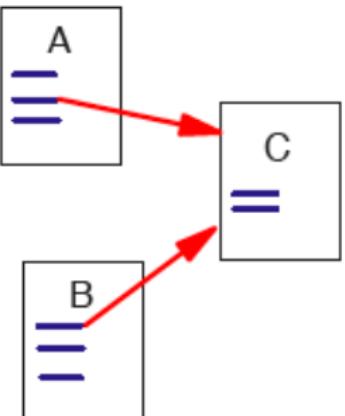
Reinforcement Learning

PageRank

Envelope Paradox

# PageRank

- Forward links: A and B are **forward linked** to C. (out-edges)
- Backlinks : A and B are **backlinks** of C. (in-edges)
- The World-Wide-Web can be seen as a graph with nodes (pages) and edges (links).
- The graph is not acyclic.
- Note: Can only collect forward links, not backlinks.
- Goal: Calculate the importance of a page from only the link structure. (Don't look at the contents of a page except for the links in it.)





ISML  
2012

Selected Topics

Occam's Razor

Reinforcement Learning

PageRank

Envelope Paradox

# PageRank - Assumptions

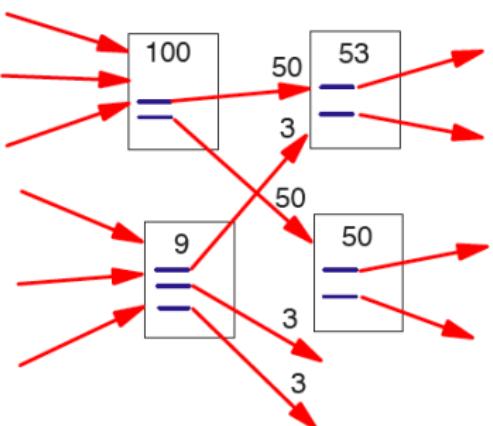
- ➊ Many links to a page signal a high importance (rank) for this page.
- ➋ A backlink from an important site counts more than a backlink from a less important site.
- Combine both of this assumptions: A page has high rank if the sum of the ranks of its backlinks is high.

# Simple PageRank - Definition

- Let  $u$  be a page. Denote by  $F_u$  the set of pages  $u$  points to, and by  $B_u$  the set of pages which point to  $u$ . Denote by  $N_u = |F_u|$  the number of out-going links from page  $u$ .
- Simple ranking  $R(u)$  is then defined as

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v}$$

where  $c$  is a normalisation such that the total rank of all web pages is constant.



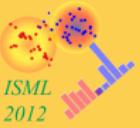
Selected Topics

Occam's Razor

Reinforcement Learning

PageRank

Envelope Paradox



Selected Topics

Occam's Razor

Reinforcement Learning

PageRank

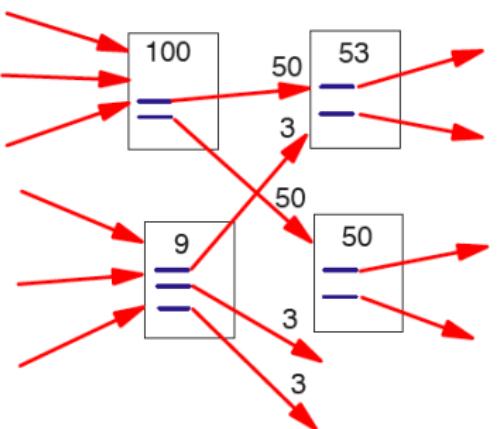
Envelope Paradox

# Simple PageRank - Calculating

- Recursive formula

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v} \quad \forall u$$

- Can be iterated.





Selected Topics

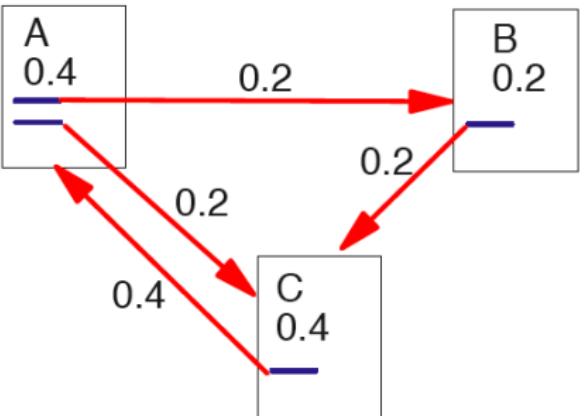
Occam's Razor

Reinforcement Learning

PageRank

Envelope Paradox

- A simple example after the PageRank calculation converged.





Selected Topics

Occam's Razor

Reinforcement Learning

PageRank

Envelope Paradox

# Simple PageRank - Eigenvector Problem

- Create a matrix  $\mathbf{A} \in \mathbb{R}^{N \times N}$ , where  $N$  is the number of web pages.

$$A_{u,v} = \begin{cases} 1/N_u & \text{page } u \text{ links to page } v \\ 0 & \text{otherwise.} \end{cases}$$

- Introduce the vector with rankings for all web pages  $\mathbf{R} \in \mathbb{R}^N$ .
- Then the simple PageRank

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v}$$

can be written as

$$\mathbf{R} = c \mathbf{A} \mathbf{R}$$

- $\mathbf{R}$  is an eigenvector of  $\mathbf{A}$  with eigenvalue  $1/c$ . (Paper says 'eigenvalue  $c$ ', but that seems to be wrong.)
- In fact, we are looking for the dominant eigenvector of  $\mathbf{A}$ .



Selected Topics

Occam's Razor

Reinforcement Learning

PageRank

Envelope Paradox

# Simple PageRank - Solved ?

- Nice result: Solve the eigenvector equation

$$\mathbf{R} = c \mathbf{A} \mathbf{R} \dots$$

- ... for a matrix with 1,000,000,000,000,000,000 entries.
- Need to iterate.



Selected Topics

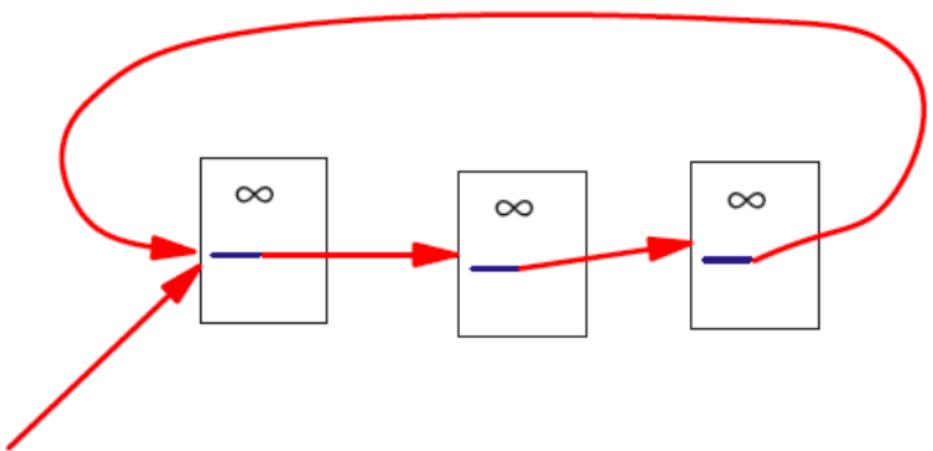
Occam's Razor

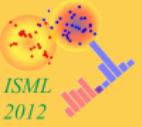
Reinforcement Learning

PageRank

Envelope Paradox

- Suppose you have two pages,  $u$  and  $v$ , which link to each other, and have incoming but NO outgoing links.
- During the iterative calculation, these pages will accumulate page rank, but never distribute it. They form a **page rank sink**.





Selected Topics

Occam's Razor

Reinforcement Learning

PageRank

Envelope Paradox

# PageRank - Add a Rank Source

- Introduce a page rank source.
- Let  $E(u)$  be some vector over the web pages that corresponds to a source of rank.
- The ranking  $R'(u)$  is then defined as

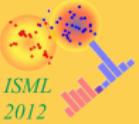
$$R'(u) = c \sum_{v \in B_u} \frac{R'(v)}{N_v} + c E(u) \quad \forall u$$

such that  $c$  is maximised and for the  $L_1$ -norm of  $R'(u)$  we have  $\|R'(u)\|_1 = 1$ .

- In matrix form

$$\mathbf{R}' = c(\mathbf{A} + \mathbf{E} \mathbf{1}^T) \mathbf{R}'$$

# *PageRank - Random Surfer Interpretation*



- The process described by the equation for PageRank

$$\mathbf{R}' = c(\mathbf{A} + \mathbf{E} \mathbf{1}^T) \mathbf{R}'$$

can also be interpreted as a random walk on graphs.

- A random surfer clicks randomly on page links.
- But a real surfer will periodically get ‘bored’ and jump randomly to some other page chosen based on the distribution in  $E$ .
- Mostly  $E$  will be chosen uniformly. But it can also be used to create ‘customised’ page ranks.

Selected Topics

Occam's Razor

Reinforcement Learning

PageRank

Envelope Paradox



Selected Topics

Occam's Razor

Reinforcement Learning

PageRank

Envelope Paradox

# PageRank - Computation

- ‘The computation of PageRank is fairly straightforward if we ignore the issues of scale.’ (citation from the paper)
- Let  $S$  be almost any vector over web pages (for instance  $E$ ).

$$R_0 \leftarrow S$$

loop :

$$R_{i+1} \leftarrow \mathbf{A}R_i$$

$$d \leftarrow \|R_i\|_1 - \|R_{i+1}\|_1$$

$$R_{i+1} \leftarrow R_{i+1} + dE$$

$$\delta \leftarrow \|R_{i+1} - R_i\|_1$$

while  $\delta > \epsilon$



Selected Topics

Occam's Razor

Reinforcement Learning

PageRank

Envelope Paradox

# PageRank - Searching

- Given a search item, find all pages which contain the search item.
- Calculate the PageRank for these selected pages.
- Don't need all ranks to provide the user with the first few ranks. But need to find the highest page ranks fast.
- 'Rank merging is known to be a very difficult problem, and we need to spend considerable additional effort before we will be able to do a reasonable evaluation of these types of queries. However, we do believe that using PageRank as a factor in these queries is quite beneficial.' (paper citation)

# *PageRank - Questions*



- How to deal with the size of the World Wide Web?
- How to update the search engine database?
- How to protect against 'Google bombs'?
- How to protect against manipulation by commercial interests?
- Search engines are an important source of information.  
Should the exact form of PageRank or any other algorithm added/applied by search engine companies be more transparent to the public?

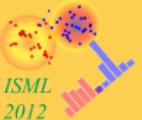
*Selected Topics*

*Occam's Razor*

*Reinforcement Learning*

*PageRank*

*Envelope Paradox*



Selected Topics

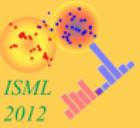
Occam's Razor

Reinforcement Learning

PageRank

Envelope Paradox

- You are given two closed envelopes, one of them contains twice the amount of money than the other. You are allowed to open one of them. Then you can choose to keep the money you find, or switch to the other envelope (which could double or half your gain).
- Symmetry: It doesn't matter whether you switch. The expected gain is the same.
- Refutation: With probability  $p = \frac{1}{2}$  the other envelope contains double or half the amount. So if you switch, the gain increases by  $\frac{1}{2} \times 2 + \frac{1}{2} \times \frac{1}{2} = \frac{5}{4}$ .
- Hint: Are the probabilities for the amounts of money equally distributed? In which interval?



## Part XXV

### *Discussion and Summary*

*Tools*

*Models for Decision  
Problems*

*Learning*

*Linear Regression and  
Classification*

*Density Estimation*

*Kernels*

*Factorising Distributions*

*Non-Factorising  
Distributions*

*Dimensionality  
Reduction*

*Sequential Data*

*Combining Models*

*What we did not cover*

*From Bayes to HMM*

# Tools - Probability Theory



- Frequentist vs. Bayes approach
- Conditional Probability
- Bayes Theorem
- Discrete vs. continuous random variables
- Distributions (Gaussian, Bernoulli, Binomial, Beta, . . .)
- Multivariate Distributions
- Change of Variables
- Conjugate Priors

Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
Distributions

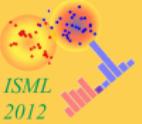
Dimensionality  
Reduction

Sequential Data

Combining Models

What we did not cover

From Bayes to HMM



- Vector Space
- Matrix-Vector Multiplication = Linear Combination
- Projection
- Positive (Semi)-definite Matrix
- Rank, Determinant, Trace, Inverse
- Eigenvectors, Eigenvalues
- Eigenvector Decomposition
- Singular Value Decomposition
- Directional Derivative
- Gradient Calculation

Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
Distributions

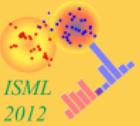
Dimensionality  
Reduction

Sequential Data

Combining Models

What we did not cover

From Bayes to HMM



Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
Distributions

Dimensionality  
Reduction

Sequential Data

Combining Models

What we did not cover

From Bayes to HMM

# Tools - Optimisation

- **Linear programming** (linear objective function, linear constraints)
- **Quadratic programming** (quadratic objective function, linear constraints)
- **Nonlinear programming** (nonlinear objective function, nonlinear constraints)
- **Convex programming** (objective function is convex, constraints, if any, form a convex set)
- **Stochastic programming** (some of the constraints or parameters depend on random variables)
- **Calculus of Variations** (find a function which optimises another objective function)
- **Dynamic programming** (e.g. Hidden Markov Model)



# Tools - Optimisation

- Gradient Descent
- Stochastic (On-line) Gradient Descent
- Gauss-Newton Method (Gradient + inverse Hessian)
- Quasi-Newton (e.g. BFGS: Gradient + iterative approximation of inverse Hessian)
- Global vs. Local extremum
- Minimum/Maximum vs. Extremum
- Lagrange Multipliers

Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
Distributions

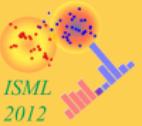
Dimensionality  
Reduction

Sequential Data

Combining Models

What we did not cover

From Bayes to HMM



Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
Distributions

Dimensionality  
Reduction

Sequential Data

Combining Models

What we did not cover

From Bayes to HMM

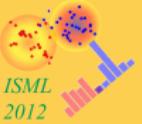
# Models for Decision Problems

Given the input space  $\mathbf{V}$ , input data  $\mathbf{x} \in \mathbf{V}$ , and a set of classes  $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_K\}$ .

- Discriminant Function  $f(\mathbf{x})$

$$f : \mathbf{V} \rightarrow \mathcal{C}$$

- Discriminant Model  $p(\mathcal{C}_k | \mathbf{x})$ , then use decision theory
- Generative Model  $p(\mathbf{x}, \mathcal{C}_k)$ , then use decision theory



- Examples
- General Setup
- Inductive Bias
- Restricted Hypothesis Space
- Importance of understanding the restrictions and whether they are appropriate

*Tools*

*Models for Decision  
Problems*

*Learning*

*Linear Regression and  
Classification*

*Density Estimation*

*Kernels*

*Factorising Distributions*

*Non-Factorising  
Distributions*

*Dimensionality  
Reduction*

*Sequential Data*

*Combining Models*

*What we did not cover*

*From Bayes to HMM*



Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
Distributions

Dimensionality  
Reduction

Sequential Data

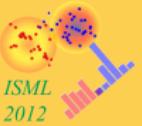
Combining Models

What we did not cover

From Bayes to HMM

# Linear Regression

- General Regression Setup
- Closed-form solution
- Maximum Likelihood and Least Squares
- Geometry of Least Squares
- Sequential Learning (on-line)
- Choice of basis function
- Regularisation
- Powerful with nonlinear feature mappings
- Bias-Variance Decomposition



Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
Distributions

Dimensionality  
Reduction

Sequential Data

Combining Models

What we did not cover

From Bayes to HMM

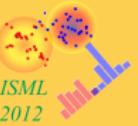
# Bayesian Linear Regression

- Closed-form solution
- Predictive Distribution

$$p(t \mid x, \mathbf{x}, \mathbf{t}) = \int p(t, \mathbf{w} \mid x, \mathbf{x}, \mathbf{t}) \, d\mathbf{w} = \int p(t \mid \mathbf{w}, x) p(\mathbf{w} \mid \mathbf{x}, \mathbf{t}) \, d\mathbf{w}$$

- Conjugate Prior
- Limitations of Linear Basis Function Models
- Curse of dimensionality

# Linear Classification



- General Classification Setup
- Input space versus Feature space
- Binary and Multiclass Labels
- Fisher's Linear Discriminant
- Perceptron Algorithm (Discontinuous activation function)
- Maximum Likelihood solution

$$\boldsymbol{\theta}^* = \arg \max_{\boldsymbol{\theta}} p(\mathbf{X}, \mathbf{t} | \boldsymbol{\theta})$$

- Naive Bayes : all features conditioned on the class are independent of each other

Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
Distributions

Dimensionality  
Reduction

Sequential Data

Combining Models

What we did not cover

From Bayes to HMM



Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
Distributions

Dimensionality  
Reduction

Sequential Data

Combining Models

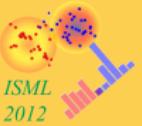
What we did not cover

From Bayes to HMM

# Logistic Regression

- Smooth logistic sigmoid acting on a linear feature vector
- Compare to perceptron
- Error as negative log likelihood (**cross-entropy** error)
- Gradient of error is target deviation times basis function (linear)
- Laplace approximation

# Flexible Basis Functions



- Neural Networks
- Multilayer Perceptron with differentiable activation function
- The basis functions can now adopt to the data.
- Weight space symmetries.
- Error Backpropagation.
- Regularisation in Neural Networks.

Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
Distributions

Dimensionality  
Reduction

Sequential Data

Combining Models

What we did not cover

From Bayes to HMM

# The Role of Training Data



- Parametric Methods : Learn the model parameter from the training data, then discard training data.
- Nonparametric methods: Use training data for prediction
  - Histogram method
  - $k$ -nearest neighbours
  - Parzen probability density model: set of function centered on the data
- Kernel methods: Use linear combination of functions evaluated at the training data.

Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
Distributions

Dimensionality  
Reduction

Sequential Data

Combining Models

What we did not cover

From Bayes to HMM



Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
DistributionsDimensionality  
Reduction

Sequential Data

Combining Models

What we did not cover

From Bayes to HMM

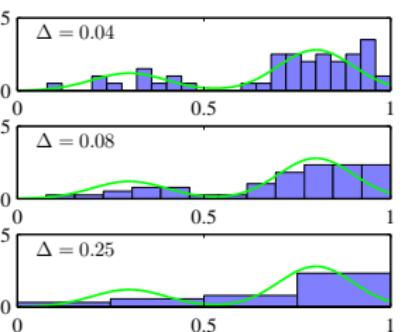
# Histogram

- Use bins  $\Delta_i$  (not necessarily of same size), count the number of data points in each bin, denoted by  $n_i$ .

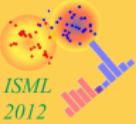
$$p(\mathbf{x} \in \text{bin}_i) = \frac{n_i}{N\Delta_i}$$

- Piecewise constant
- Normalisation

$$\sum_i p(\mathbf{x} \in \text{bin}_i)\Delta_i = \sum_i \frac{n_i}{N} = 1$$



Data set of 50 points drawn from a distribution (green curve).



## Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
DistributionsDimensionality  
Reduction

Sequential Data

Combining Models

What we did not cover

From Bayes to HMM

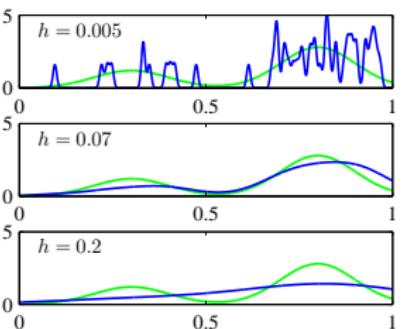
# Parzen Window

- Parzen Window density estimation

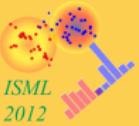
$$p(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^N \frac{1}{h^D} k\left(\frac{\mathbf{x} - \mathbf{x}_n}{h}\right)$$

where the indicator function over the unit cube centered at  $\mathbf{u}$  is defined as

$$k(\mathbf{u}) = \begin{cases} 1, & |u_i| \leq 1/2, \quad i = 1, \dots, D \\ 0, & \text{otherwise.} \end{cases}$$



Data set of 50 points drawn from a distribution (green curve).



Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
DistributionsDimensionality  
Reduction

Sequential Data

Combining Models

What we did not cover

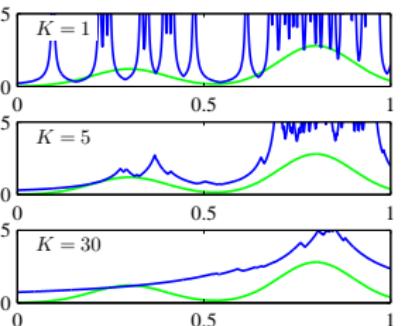
From Bayes to HMM

# Nearest-Neighbour Density Estimation

- Fix the number of data points per bin/volume; keep bin size/volume variable. (instead of fixing the bin/volume and counting the number of data points as in a histogram or Parzen window)

$$p(\mathbf{x}) = \frac{K}{NV}$$

where  $V$  is the volume enclosing the  $K$  nearest neighbours.



Data set of 50 points drawn from a distribution (green curve).



Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
Distributions

Dimensionality  
Reduction

Sequential Data

Combining Models

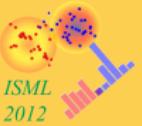
What we did not cover

From Bayes to HMM

# Kernels

- Inner Product → Kernel
- Kernels are a kind of similarity measure
- Sparse Kernel Machines
- Support Vector Machines
- How do we get to the relevant data points?
- Overlapping class distribution
- Output are decisions, not posterior probabilities.
- Relevance Vector Machines: Bayesian Sparse Kernel technique for classification and regression.

# Probabilistic Graphical Models



- Joint Probability factorises.
- Independence Structure or "the absence of edges"
- Directed, Undirected and Factor Graphs
- Bayesian Network, Blocked Path and  $d$ -separation
- Markov Random Field, (maximal) Cliques
- Factor Graphs are Bipartite Graphs
- Sum-Product Algorithm
- Messages

Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
Distributions

Dimensionality  
Reduction

Sequential Data

Combining Models

What we did not cover

From Bayes to HMM

# Mixture Models



- Joint Probability over observed variables does not longer factorise.
- Introduce discrete latent variables to model complex marginal distributions over the observed variables by simpler distributions over observed and latent variables.
- $K$ -means clustering
- Data compression
- Mixture of Gaussians

$$\begin{aligned} p(\mathbf{x}) &= \sum_{\mathbf{z}} p(\mathbf{z}) p(\mathbf{x} | \mathbf{z}) = \sum_{\mathbf{z}} \prod_{k=1}^K \pi_k^{z_k} \prod_{k=1}^K \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k} \\ &= \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \end{aligned}$$

Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
Distributions

Dimensionality  
Reduction

Sequential Data

Combining Models

What we did not cover

From Bayes to HMM



## Tools

Models for Decision  
Problems

## Learning

Linear Regression and  
Classification

Density Estimation

## Kernels

Factorising Distributions

Non-Factorising  
DistributionsDimensionality  
Reduction

Sequential Data

Combining Models

What we did not cover

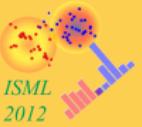
From Bayes to HMM

# Expectation Maximisation (EM)

- Evaluate the responsibilities, then maximise the parameters.
- E step: Find  $p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}^{\text{old}})$ .
- M step: Find  $\boldsymbol{\theta}^{\text{new}} = \arg \max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$  where

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})$$

- Kullback-Leibler Divergence



Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
Distributions

Dimensionality  
Reduction

Sequential Data

Combining Models

What we did not cover

From Bayes to HMM

# Approximate Inference

- Central task: Find  $p(\mathbf{Z} \mid \mathbf{X}, \theta)$ .
- Deterministic Approximation based on analysis.
- Partition  $\mathbf{Z}$  and apply mean field theory.
- Exponential Family



Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
Distributions

Dimensionality  
Reduction

Sequential Data

Combining Models

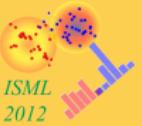
What we did not cover

From Bayes to HMM

# Sampling

- Central task: Find  $p(\mathbf{Z} | \mathbf{X}, \theta)$ .
- Stochastic Approximation
- Sampling from the uniform distribution.
- Sampling from standard distributions via the inversion of the cumulative distribution function
- Rejection Sampling
- Adaptive Rejection Sampling
- Importance Sampling : calculate the expectation of function  $f(z)$  under distribution  $p(z)$  using some simpler  $q(z)$ .
- Markov Chain Monte Carlo
- Metropolis Algorithm (using a symmetric proposal distribution)

# Dimensionality Reduction - PCA



- Maximise Variance
- Find the eigenvectors of the covariance corresponding to the largest eigenvalues.
- PCA and Compression
- Data Standardisation
- Data Whitening

Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
Distributions

Dimensionality  
Reduction

Sequential Data

Combining Models

What we did not cover

From Bayes to HMM



Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
Distributions

Dimensionality  
Reduction

Sequential Data

Combining Models

What we did not cover

From Bayes to HMM

# Sequential Data

- Stationary vs. Nonstationary Sequential Distributions
- Markov Model of order  $M = 0, 1, \dots$
- State Space Model using latent variables
- Hidden Markov Model (HMM): Latent variables are discrete.
- Linear Dynamical System: Latent variables are continuous.
- Homogeneous HMM
- Left-to-right HMM
- Viterbi algorithm



- Committee
- Boosting - AdaBoost
- Tree-based Models
- Classification and Regression Trees (CART)

Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
Distributions

Dimensionality  
Reduction

Sequential Data

Combining Models

What we did not cover

From Bayes to HMM



# Probability/Density Estimation

- Maximum Likelihood (ML)

$$\boldsymbol{\theta}^* = \arg \max_{\boldsymbol{\theta}} p(\mathcal{D} | \boldsymbol{\theta})$$

- Bayesian

$$p(\boldsymbol{\theta} | \mathcal{D}^{(n)}) \propto p(\mathbf{x}^{(n)} | \boldsymbol{\theta}) p(\boldsymbol{\theta} | \mathcal{D}^{(n-1)})$$

- Maximum a Posteriori (MAP)

$$\boldsymbol{\theta}^* = \arg \max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} | \mathcal{D}) \propto p(\mathcal{D} | \boldsymbol{\theta}) p(\boldsymbol{\theta})$$

Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
DistributionsDimensionality  
Reduction

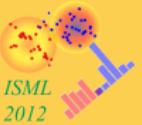
Sequential Data

Combining Models

What we did not cover

From Bayes to HMM

# Being Bayesian



- Bayesian = maintaining a distribution
  - for any quantity of interest (e.g. position)
  - Bayesian parameter estimation
- Key idea: robust to overfitting
  - maintains varying strength of belief in multiple hypothesis
- In the limit of infinite data: ML = Bayesian
  - Data ‘swamps’ prior
  - Can you explain why?

Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
Distributions

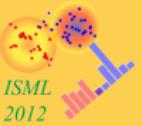
Dimensionality  
Reduction

Sequential Data

Combining Models

What we did not cover

From Bayes to HMM



ISML  
2012

Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
Distributions

Dimensionality  
Reduction

Sequential Data

Combining Models

What we did not cover

From Bayes to HMM

- Linear Dynamical System
- Natural Language Processing
- Reinforcement Learning
- Learning Sets of Rules
- Information Theory
- Evolutionary Methods (e.g. Genetic Algorithms)
- Simulated annealing



Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
Distributions

Dimensionality  
Reduction

Sequential Data

Combining Models

What we did not cover

From Bayes to HMM

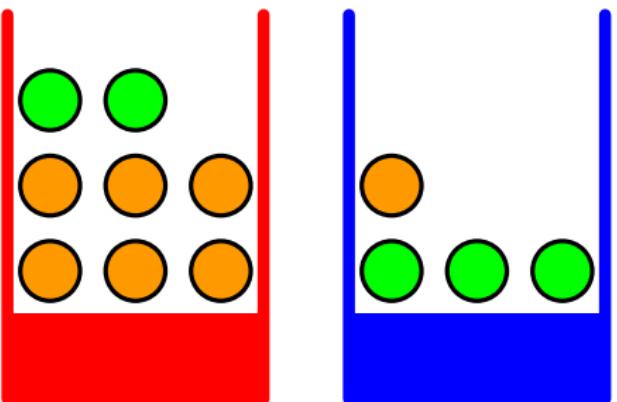
# Where did we start? - Simple Experiment

## 1 Choose a box.

- Red box  $p(B = r) = 4/10$
- Blue box  $p(B = b) = 6/10$

## 2 Choose any item of the selected box with equal probability.

- Given that we have chosen an orange, what is the probability that the box we chose was the blue one?





Tools

Models for Decision  
Problems

Learning

Linear Regression and  
Classification

Density Estimation

Kernels

Factorising Distributions

Non-Factorising  
DistributionsDimensionality  
Reduction

Sequential Data

Combining Models

What we did not cover

From Bayes to HMM

# Where did we finish? - State Space Model

- For each observation  $\mathbf{x}_n$ , a latent variable  $\mathbf{z}_n$  is added.
- The type and dimensionality of  $\mathbf{z}_n$  can differ from  $\mathbf{x}_n$ .
- Assume that the latent variables form a Markov chain.
- Key property: conditional independence of the latent variables

$$\mathbf{z}_{n+1} \perp\!\!\!\perp \mathbf{z}_{n-1} \mid \mathbf{z}_n$$

