Advanced probabilistic methods

Lecture 1

Pekka Marttinen

Aalto University

January, 2021

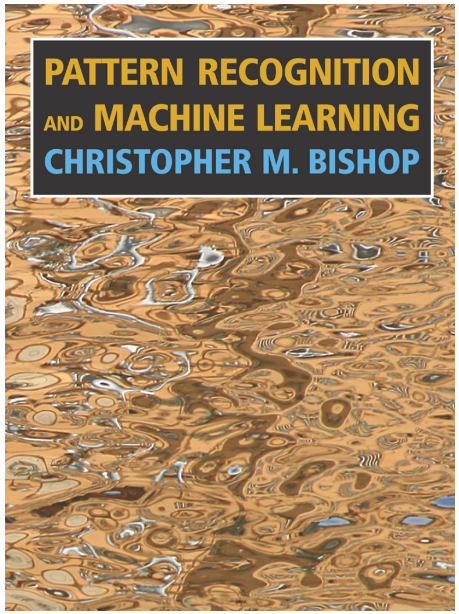
Lecture 1 overview

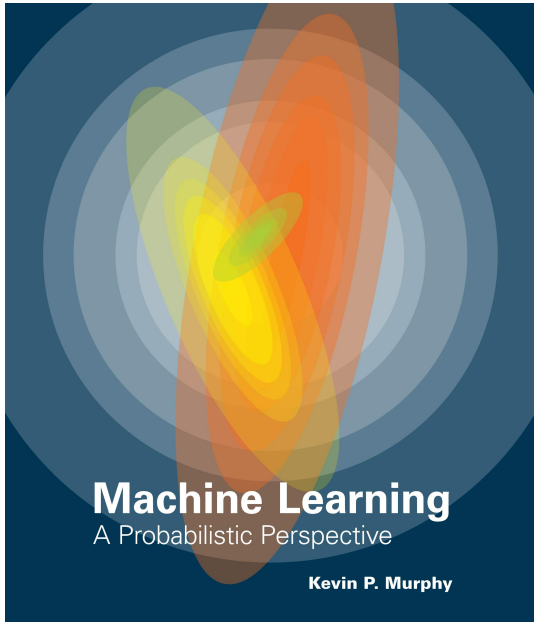
- Practical matters
  - Structure, grading, workload, online implementation
  - Exercise format
  - Student feedback from previous years
- Course overview
- Basic probability calculus (Barber, Ch. 1)
- Basic graph concepts

Course structure

- Structure
  - Lectures 9×2 hours
  - Exercise sessions 9×2 hours
  - See *Timetable* in *myCourses/Materials*
- Grading based on
  - Exam: 55% of the total weight
  - Each assignment: 5% of the weight (sum 45%)
  - *Preliminary* boundaries: 1:50%, 2:60%, 3:70%, 4:80%, 5:90%

Course books





- *Bayesian Reasoning and Machine Learning* available at *www.cs.ucl.ac.uk/staff/D.Barber/brml*
- *Pattern Recognition and Machine Learning* available at *www.microsoft.com/en-us/research/people/cmbishop/*

Estimated workload

- Lectures: 9×2h
- Preparation for lectures, reading the book (~ 200 pages): 9×4h
- Exercise sessions: 9×2h
- Doing the exercises: 9×5h
- Preparing for the exam: 14h
- Exam: 4h
- Total 135h. As credits $135/27 = 5$cr.

Online implementation in 2021

- The lectures are recorded and released in advance
- Students can post questions about lectures in **Slack**.
- Each lecture is followed by an online Q&A session in Zoom. The lecturer will go through questions related to the lectures posted in the Slack and the students can also ask additional questions.
- Links to Slack and Zoom will be posted in MyCourses.

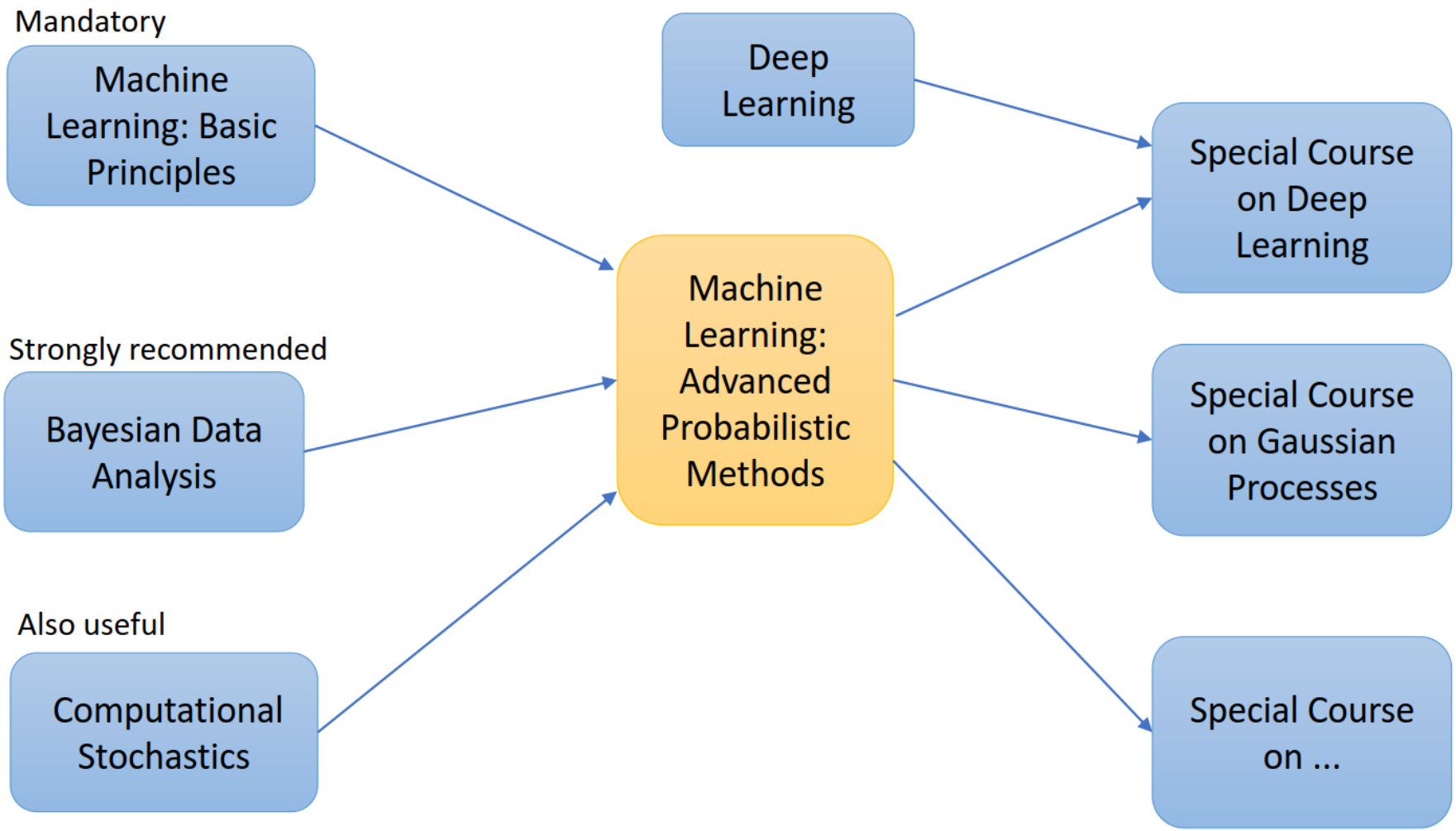
Exercises

- Exercises must be returned by the deadline using JupyterHub.
  - See instructions in MyCourses.
  - Grading principle: **2p**→done, almost correct; **1p**→done, but something clearly missing/incorrect; **0p**→not done or completely incorrect (may be modified case-by-case).
  - Exercises are graded by the TAs, not corrected → Make sure you know the correct answer afterwards by attending the exercise sessions or checking the model solutions.
- Exercise session format
  - Help for getting started with next week's exercises
  - Possibility to ask about next week's exercises or previous week's solutions
  - Two assistants present

Exercises

- Slack
  - Write a question in a dedicated Slack channel. The TAs will answer them at the times of the exercise sessions (possibly at other times, see details in MyCourses)
  - Students are welcome to comment and give hints to each other's questions in the Slack; however, do not reveal the full answer.
- Other policies
  - Collaborating with your colleagues to solve the exercises is allowed and encouraged, but in the end you are expected to write your own solution (copy-paste from someone else is not allowed and will be checked for).
  - Sharing solutions online or use of any such material is profibited. Suspected violations will be reported for further investigation.

Relation to other courses



Student feedback from previous years

- About prerequisites
  - *'Bayesian Data Analysis should be listed as a prerequisite'*
    $\longrightarrow$See previous slide.
- About lectures
  - '*One of the courses where lecture attendance is really beneficial, I would definitely point this out in the intro lecture as students are trying to optimize their time between other lectures and exercise sessions*'
  - '*All lectures could be recorded*'
    $\longrightarrow$This will be done.

Student feedback from previous years

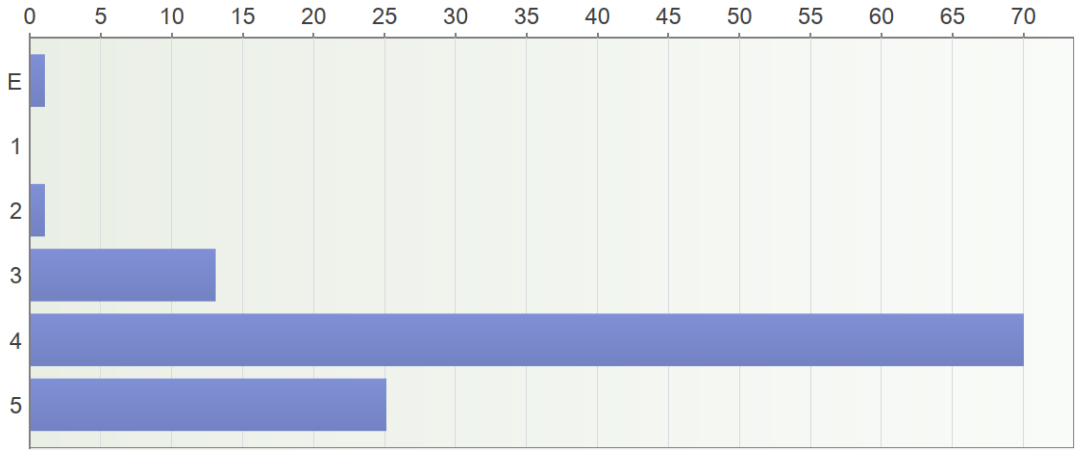
- About exercises (positive)
  - *'Participating in the exercise sessions was extremely useful and the course assistants were great'*
  - *'The exercises really helped me to understand the concepts related to this course, and I think that they played a huge role in my learning'*
  - *'The exercise sessions were very good. Without them I would never have been able to achieve that far.'*
- About exercises (things to improve)
  - *'A slack group would have been useful'*
    $\longrightarrow$This will be implemented.

Student feedback from previous years

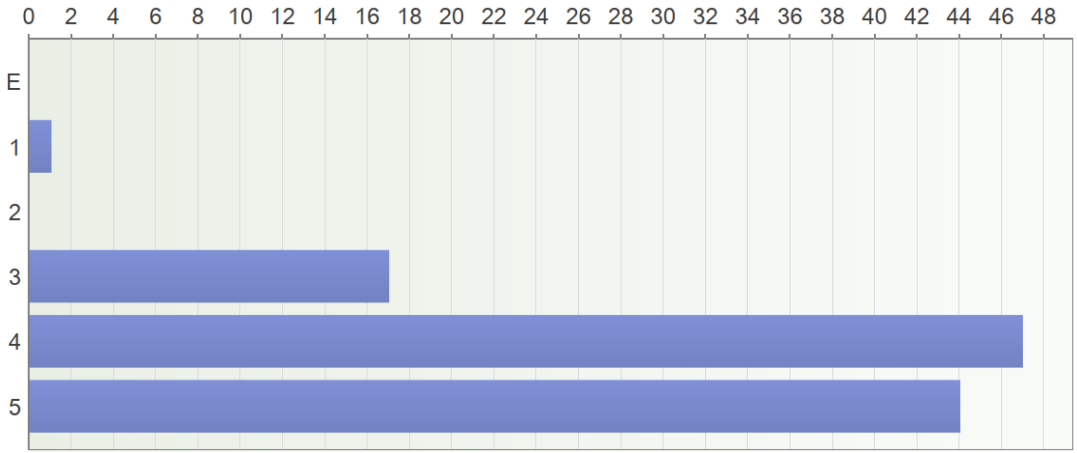
- About difficulty in general
  - *'At least to me there was steep curve in the topic difficulty starting from lecture 5. Compress lectures 1-4 and spend more time on the "new stuff"*
    $\rightarrow$Some changes in the material to this direction.
    $\rightarrow$Take a full advantage of the Exercise sessions (also and especially on the 2nd half).
    $\rightarrow$Ask clarifications in Slack and the Q&A session.
- About contents
  - '*Add a small project*'
    $\rightarrow$We used to have one, but dropped that to keep the workload manageable.
  - '*How were these topics are chosen? Does the course aim to practical work life skills, or prepare for research career or is it just general background knowledge?*'
    $\rightarrow$See slides *Course contents*, *Course outline* and *What to expect* below.

Student feedback from 2020, overview

- Overall assessment

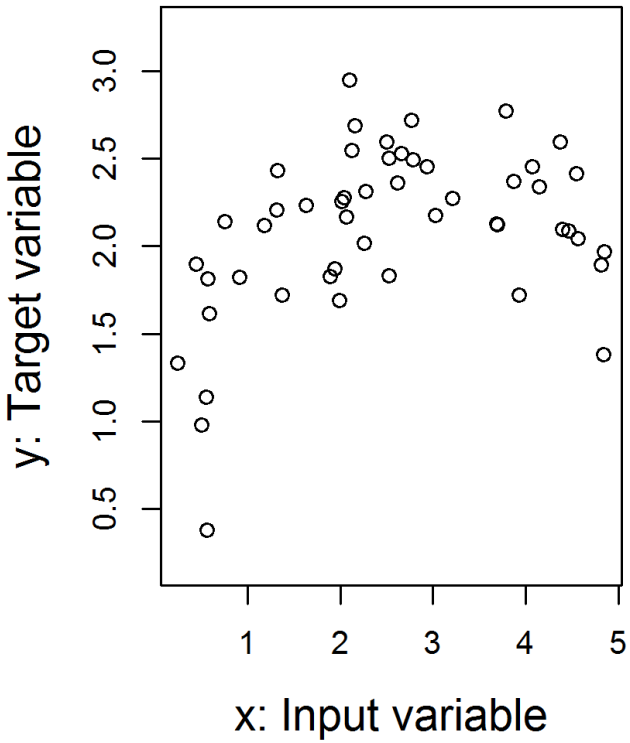


- I will benefit from things learnt on the course



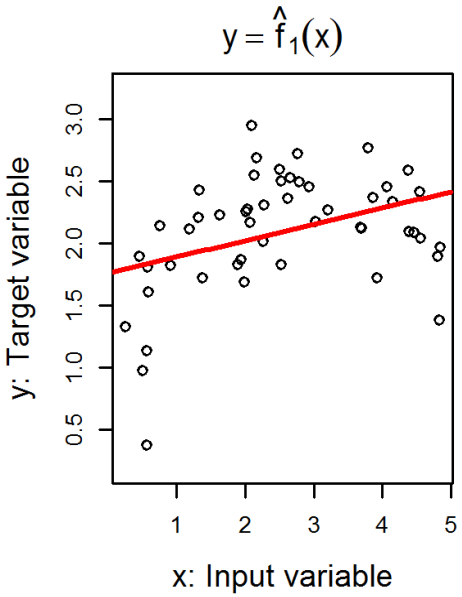
Probabilistic modeling overview (1/2)

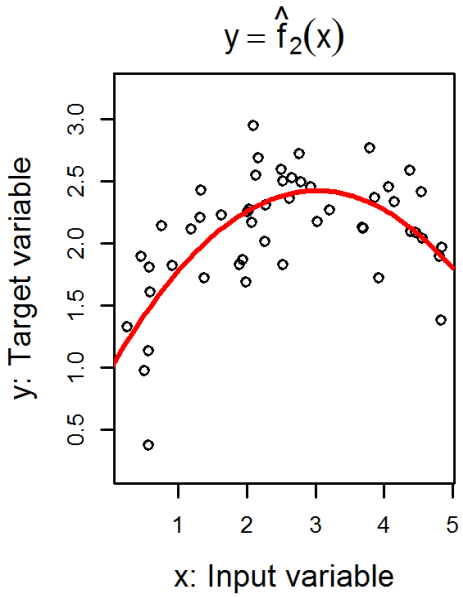
- The goal of **probabilistic modeling** is to answer a question about the data:
  - Classify the samples into groups
  - Create prediction for future observations
  - Select between competing hypotheses
  - Estimate a parameter, such as the mean, of the population
  - ...
- and **quantify uncertainty** using probabilities.

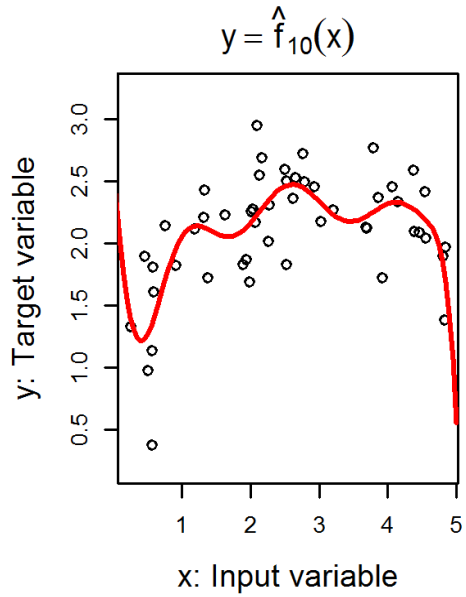


Probabilistic modeling overview (2/2)

- Probabilistic modeling in a nutshell
  1. Select a **model**
  2. **Infer** the parameters of the model (train/fit the model)
  3. Use the fitted model to answer the question of interest
- Usually several models are considered, requiring **model selection**.
- For example: $f_n(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$

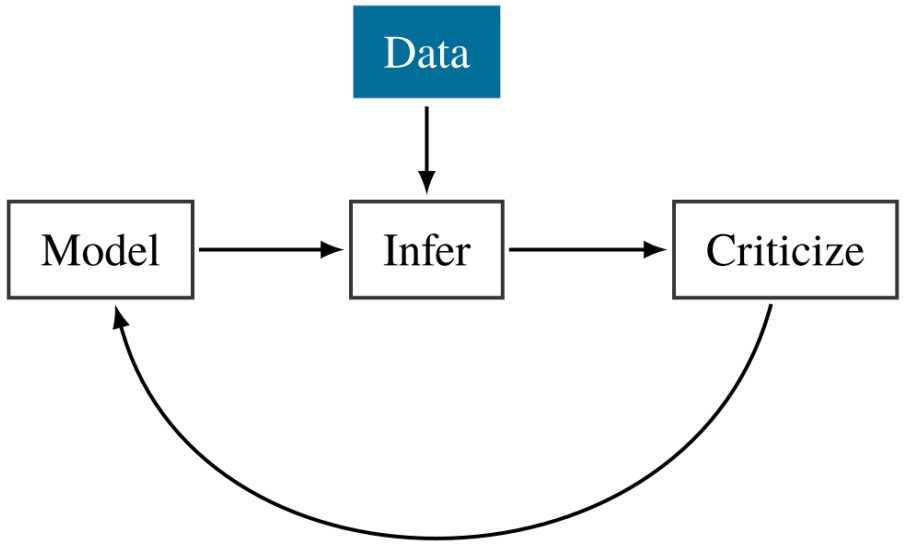






Course contents

- Ingredients of probabilistic modeling
  - **Models**: Bayesian networks, Sparse Bayesian linear regression, Gaussian mixture models, latent linear models
  - **Methods for inference**: maximum likelihood, maximum a posteriori (MAP), Laplace approximation, expectation maximization (EM), Variational Bayes (VB), Stochastic variational inference (SVI)
  - **Ways to select between models**



Box's loop (Blei, 2014)

Course outline

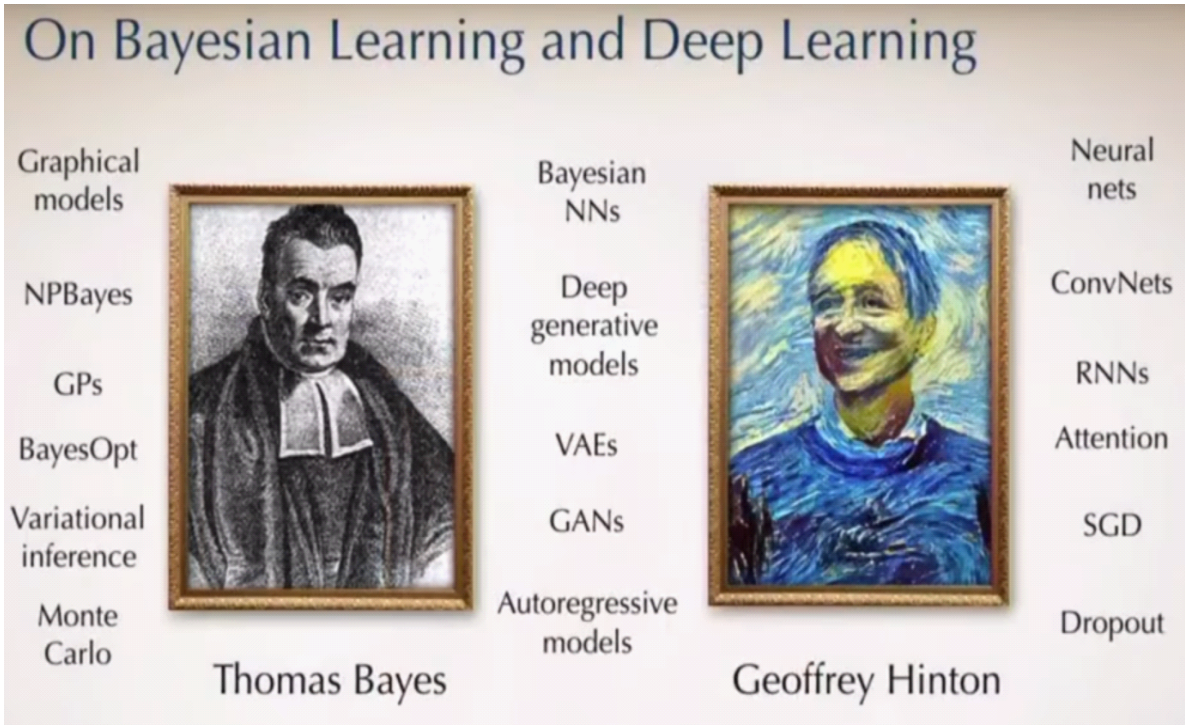
- "Probabilistic Modelling Course" (the last column) from the preface of Barber's book is used as the backbone. Deterministic approximate inference techniques (28) have been added to this, and correspondingly less weight is given to some other topics.

What to expect on the course

- More emphasis on
  - fundamental principles of probabilistic modeling and machine learning
  - introducing simple models that serve as building blocks for more complex (or realistic) models
  - building a detailed understanding of the algorithms through simple examples, which allows us to focus on the principles
  - enhancing mathematical skills to operate with probabilities
- Less emphasis on
  - hands-on learning on how to use a specific software package to fit 'advanced' models to a real-world data set (though we'll use *PyTorch* in some exercises to show how things are *nowadays* done 'in real life')
- Goal
  - knowledge needed to start understanding and applying probabilistic modeling using existing software packages, and those to come

Role of probabilistic machine learning today

- Keynote at *NeurIPS 2017* by Yee Whye Teh



https://www.youtube.com/watch?v=9saauSBgmcQ

- *NeurIPS 2020*: articles with a keyword in the title: *Bayes* (n=63), *Uncertainty* (n=18), *Variational* (n=36). Many of these coming from the major IT companies.

Probability theory, basics

- Marginalization
- Independence
- Conditional distribution
- Conditional independence
- Continuous random variables

(To recap these, see *Additional Reading* in *myCourses/Materials*)

Notation (1/2)

- Random variables: $X, Y, Z, \ldots$
- Values these random variables can take: $x, y, z, \ldots$
- Probability
  - The following notations are used interchangeably

$$p(X = x) = p_X(x) = p(x)$$

  - All are interpreted as the probability that variable X is in state x

Notation (2/2)

- Domain
  - $dom(X)$ denotes all possible states for variable X.
- Distribution of a variable X consists of
  - its domain $dom(X)$
  - and full specification of probability values $p_X(x)$, for all possible $x \in dom(X)$
- Normalization
  - The summation over all the states

$$\sum_{x \in dom(X)} p(X = x) = 1$$

  - The sum can be written as: $\sum_x p(x) = 1$

Example - probability table

B	M	K	$p(b, m, k)$
1	1	1	0.012
1	1	0	0.108
1	0	1	0.288
1	0	0	0.192
0	1	1	0.016
0	1	0	0.064
0	0	1	0.096
0	0	0	0.224

- The probability table lists the probabilities of all possible combinations of the random variables.
- The *joint* distribution of B, M and K
- For example

$$p_{B,M,K}(1,1,0) = p(B=1, M=1, K=0) = 0.108$$

- Modified from Example 1.3 "Inspector Clouseau"
  M ='Maid is the murderer'
  B ='Butler is the murderer'
  K ='Knife is the murder weapon'

Marginalization

- Given a joint dist $p_{X,Y}(x, y)$, the marginal dist of X is defined by

$$p_X(x) = \sum_y p_{X,Y}(x, y)$$

- More generally,

$$p(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = \sum_{x_i} p(x_1, \ldots, x_n)$$

Example - marginalization (1/2)

B	M	K	$p(b, m, k)$
1	1	1	0.012
1	1	0	0.108
1	0	1	0.288
1	0	0	0.192
0	1	1	0.016
0	1	0	0.064
0	0	1	0.096
0	0	0	0.224

- What is the marginal distribution of B and M?
- We need to compute $p_{B,M}(b, m)$, for all possible b and m.

Example - marginalization (2/2)

- Use

$$p_{B,M}(b,m) = \sum_{k=0}^{1} p_{B,M,K}(b,m,k)$$

- For example:

$$\begin{aligned} p_{B,M}(0,0) &= p_{B,M,K}(0,0,0) + p_{B,M,K}(0,0,1) \\ &= 0.096 + 0.224 = 0.32 \end{aligned}$$

- Doing this for all B, M combinations, we get the marginal probability table

B	M	$p(b,m)$
1	1	0.12
1	0	0.48
0	1	0.08
0	0	0.32

Independence

- Random variables X and Y are independent if

$$p_{X,Y}(x, y) = p_X(x)p_Y(y)$$

  for all x and y.
- Intuitively, this means that knowing the value of X does not provide any information about the value of Y.
- Notation: $X \perp\!\!\!\perp Y$
- More generally: $\mathcal{A} = \{A_1, \ldots, A_k\}$ and $\mathcal{B} = \{B_1, \ldots, B_l\}$ are independent if

$$\begin{aligned} &p_{A_1,\ldots,A_k,B_1,\ldots,B_l}(a_1, \ldots, a_k, b_1, \ldots, b_l) \\ &= p_{A_1,\ldots,A_k}(a_1, \ldots, a_k)p_{B_1,\ldots,B_l}(b_1, \ldots, b_l) \end{aligned}$$

Example - Independence (1/2)

B	M	$p(b, m)$
1	1	0.12
1	0	0.48
0	1	0.08
0	0	0.32

- Are B and M independent?

Example - Independence (2/2)

- Marginal distributions

B	$p(b)$
1	0.6
0	0.4

and

M	$p(m)$
1	0.2
0	0.8

- Direct computation gives

B	M	$p(b)p(m)$	$p(b,m)$
1	1	0.12	0.12
1	0	0.48	0.48
0	1	0.08	0.08
0	0	0.32	0.32

- Hence, B and M are (marginally) independent

Statistical vs. causal independence

- D='number of people drowned', A='amount of ice-cream sold'
  - Are D and A independent?
  - Are D and A causally dependent?

Conditional distribution

- Conditional distribution

$$p_{X|Y}(x|y) = \frac{p_{X,Y}(x,y)}{p_Y(y)}$$

specifies the probability of each possible value x of X given that we have observed variable Y in state y.

Example - Conditional distribution

- For example:

$$p(K=1|B=1,M=1) = \frac{p(B=1,M=1,K=1)}{p(B=1,M=1)} = 0.1$$

$$p(K=0|B=1,M=1) = 0.9$$

- All conditional probabilities in the last column

B	M	K	$p(b,m,k)$	$p(k\|b,m)$
1	1	1	0.012	0.1
1	1	0	0.108	0.9
1	0	1	0.288	0.6
1	0	0	0.192	0.4
0	1	1	0.016	0.2
0	1	0	0.064	0.8
0	0	1	0.096	0.3
0	0	0	0.224	0.7

Conditional independence

- $X \perp\!\!\!\perp Y|Z$ denotes that variables X and Y are conditionally independent of each other, given the state of variable Z. This is formally defined by condition

$$p_{X,Y|Z}(x, y|z) = p_{X|Z}(x|z)p_{Y|Z}(y|z)$$

  for all states x, y, z of variables X, Y, Z.
- Intuitively, this means that if we know the value of Z, knowing in addition the value of Y does not provide any information about the value of X. Indeed, provided $p(y, z) > 0$, we have

$$X \perp\!\!\!\perp Y|Z \Longrightarrow p_{X|Y,Z}(x|y, z) = p_{X|Z}(x|z)$$

Conditional independence

$$X \perp\!\!\!\perp Y|Z \implies p_{X|Y,Z}(x|y,z) = p_{X|Z}(x|z)$$

- Proof

$$\begin{aligned} p(x|y,z) &= \frac{p(x,y,z)}{p(y,z)} = \frac{p(x,y|z)p(z)}{p(z)p(y|z)} \\ &= \frac{p(x|z)p(y|z)p(z)}{p(z)p(y|z)} = p(x|z) \end{aligned}$$

- The general *chain rule of probability*

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

follows from iterative use of the definition of conditional probability.

Example - Conditional independence (1/3)

B	M	K	$p(b, m, k)$
1	1	1	0.012
1	1	0	0.108
1	0	1	0.288
1	0	0	0.192
0	1	1	0.016
0	1	0	0.064
0	0	1	0.096
0	0	0	0.224

- Are M and B conditionally independent, given K?
- We need to compare
  - $p_{M|K}(m|k)p_{B|K}(b|k)$
  - $p_{B,M|K}(b, m|k)$

  for all m, b, k.

Example - Conditional independence (2/3)

- For example,

$$p(B=1, M=1|K=1) = \frac{p(B=1, M=1, K=1)}{p(K=1)}$$
$$= \frac{0.012}{0.012+0.288+0.016+0.096} \approx 0.0291$$

- Similarly,

$$p(M=1|K=1) = \frac{p(M=1, K=1)}{p(K=1)}$$
$$= \frac{0.012+0.016}{0.012+0.288+0.016+0.096} \approx 0.0508$$

and

$$p(B=1|K=1) = \ldots \approx 0.7110$$

Example - Conditional independence (3/3)

B	M	K	$p(b,m\|k)$	$p(b\|k)$	$p(m\|k)$	$p(b\|k)p(m\|k)$
1	1	1	**0.029**	**0.711**	**0.051**	**0.036**
0	1	1	...	...	...	...
1	0	1				
0	0	1				
1	1	0				
0	1	0				
1	0	0				
0	0	0				

- Because $0.029 \neq 0.036$, it follows that B and M are not conditionally independent given K.

Intuition for independence and conditional independence (1/2)

- Let $X_1, X_2, \ldots, X_n$ denote the cumulative sum of n dice throws, such that $dom(X_1) = \{1, \ldots, 6\}$, $dom(X_2) = \{2, \ldots, 12\}$, etc.
  - Is X_{n+1} independent of X_{n-1}?
  - Is X_{n+1} conditionally independent of X_{n-1} given X_n?
- X='Location of an airplane now', Y='Location of the plane 15s ago', Z='Location 15s from now'
  - Is Y independent of Z?
  - Is Y conditionally independent of Z given X?

Intuition for independence and conditional independence (2/2)

- S='sunshine', D='number of people drowned', A='amount of ice-cream sold'
  - Are D and A independent?
  - Are D and A conditionally independent given S?
- A='The alarm is on', B=There is a burglar in the house", T='A truck passes the house'
  - Suppose that the alarm can be triggered either by a burglar or by a passing truck
  - Are B and T independent?
  - Are B and T conditionally independent given A

Continuous random variables (1/3)

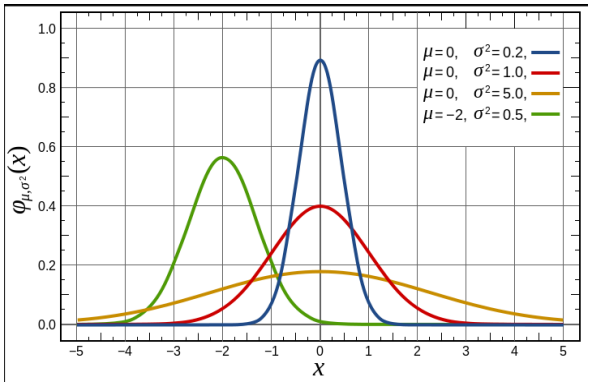
- Probability density function (pdf) for a continuous variable X, $f_X()$

$$\int_{x \in \mathcal{R}} f_X(x)dx = 1$$

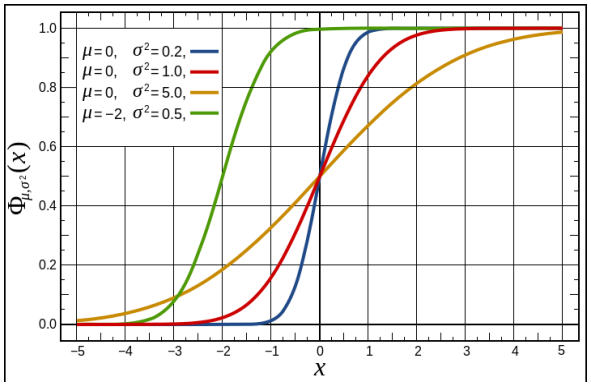
$$p(X \in [a, b]) = \int_{x=a}^{b} f_X(x)dx$$

- Cumulative distribution function (cdf)

$$F_X(x) = p(X \leq x) = \int_{t=-\infty}^{x} f_X(t)dt$$



$N(\mu, \sigma^2)$ pdf (Wikip.)



$N(\mu, \sigma^2)$ cdf (Wikip.)

Continuous random variables (2/3)

- Concepts presented can be generalized to continuous random variables
- Marginalization
  - Discrete: $p_X(x) = \sum_y p_{X,Y}(x,y)$
  - Continuous: $f_X(x) = \int_y f_{X,Y}(x,y)dy$
- Expected value
  - Discrete: $E(X) = \sum_x x\, p_X(x)$
  - Continuous: $E(X) = \int_x x\, f_X(x)\, dx$

Continuous random variables (3/3)

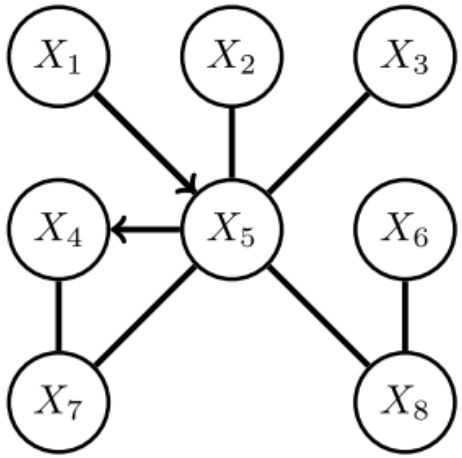
- Conditional distribution

$$f_{Y|X}(y|x) = \frac{f_{X,Y}(x,y)}{f_X(x)}$$

- (conditional) independence: $X \perp\!\!\!\perp Y|Z$, if

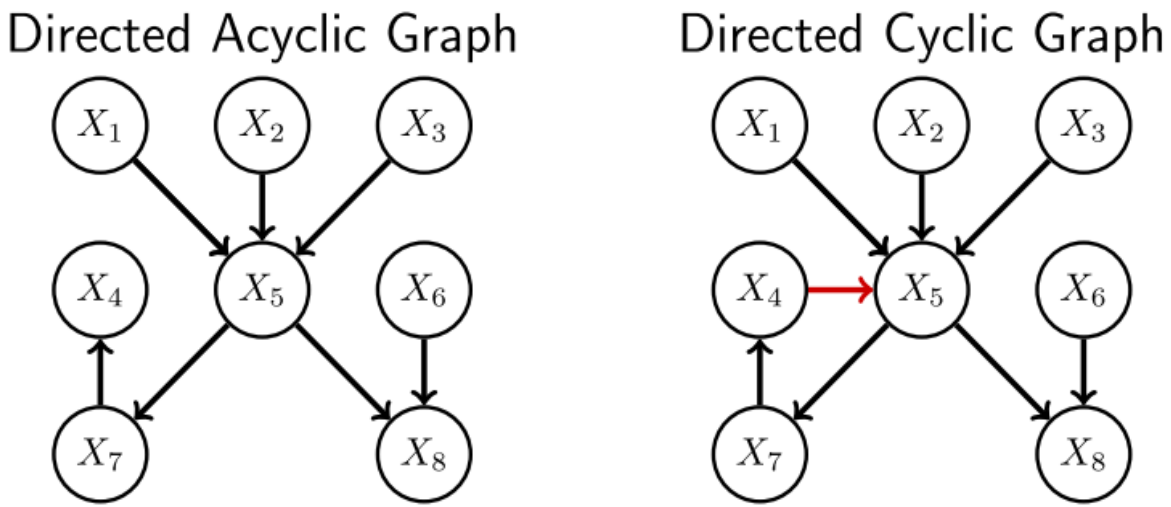
$$f_{X,Y|Z}(x,y|z) = f_{X|Z}(x|z) f_{Y|Z}(y|z)$$

Basic graph definitions



- A **graph** consists of **nodes** (vertices) and undirected of directed **edges** (links) between nodes.
- A path from X_i to X_j is a sequence of connected nodes starting at X_i and ending at X_j.

Directed graphs



- A Directed Acyclic Graph (**DAG**) is a directed graph without cycles
- **Parents, Children, Ancestors, Descendants,...** (see Ch. 2)

Important points

- marginalization
- conditional distribution
- conditional/marginal independence
- probability density function, cumulative distribution function
- Basic graph concepts

Advanced probabilistic methods

Lecture 2

Pekka Marttinen

Aalto University

January, 2021

Lecture 2 overview

- Bayesian networks (also called 'belief networks')
  - Definition
  - Motivation
- Independence in Bayesian networks
  - d-separation
  - Markov equivalence
- Computation using Bayesian networks
- Ch. 3 in Barber

Bayesian networks

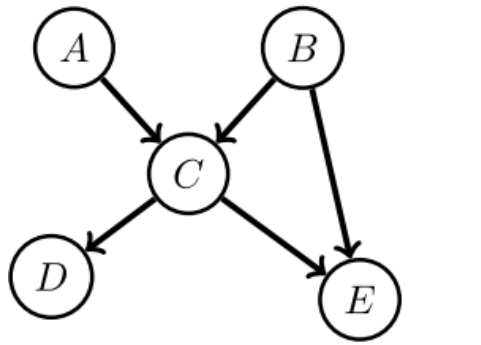
- A Bayesian network is a directed acyclic graph (DAG) in which nodes represent random variables, whose joint distribution can be written as

$$p(x_1, \ldots, x_D) = \prod_{i=1}^{D} p(x_i | \mathrm{pa}(x_i)),$$

where $\mathrm{pa}(x_i)$ represent the parents of x_i.

- Example:

$$p(a, b, c, d, e) = p(a)p(b)p(c|a, b)p(d|c)p(e|b, c)$$



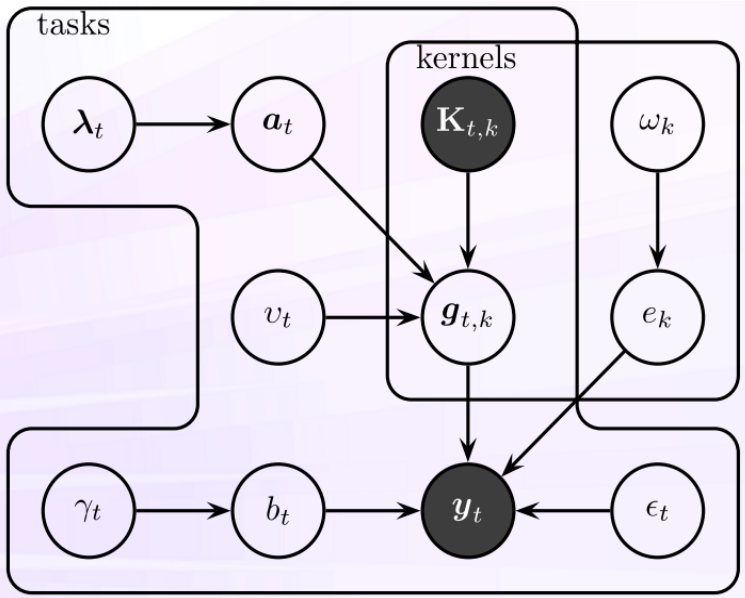
$p(E|B, C)$

Bayesian networks in machine learning

- **An important conceptual tool**
  - BNs are a concise way to represent and communicate the structure and assumptions of a model
- **Computational efficiency**
  - Compact representation of the joint distribution
  - Efficient algorithms exist to compute conditional distributions

Example 1: Prediction of drug sensitivity

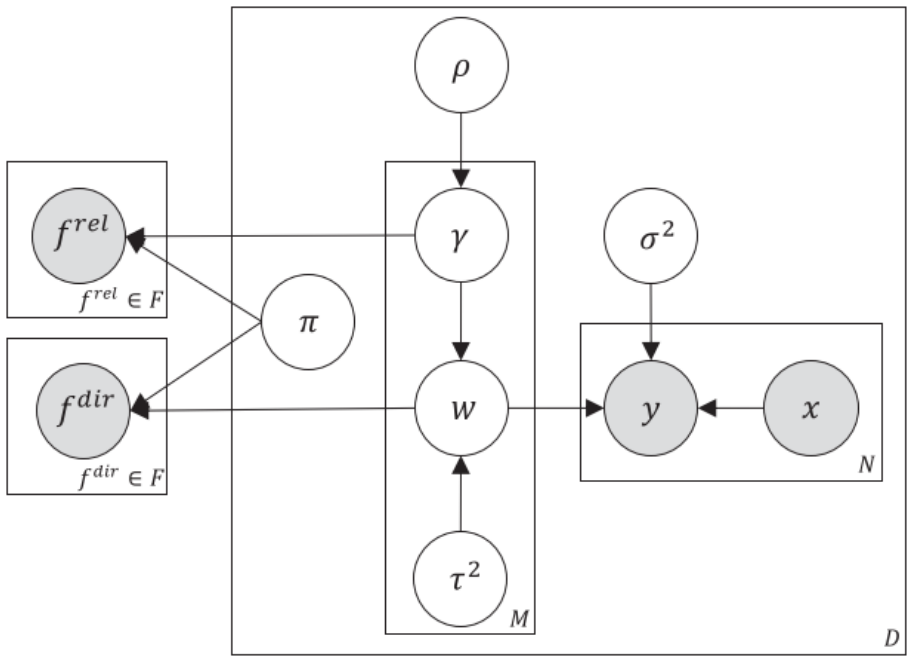
- Solution combines: multiple kernel learning, multiview learning, multitask learning, Bayesian inference
- Team Aalto rank 1/47. (Costello et al. *Nature Biotechnology*, 2014)
- Bayesian wetworks were used to communicate the model structure in a compact way



$$
\begin{aligned}
\lambda_{t,i} &\sim \mathcal{G}(\lambda_{t,i}; \alpha_\lambda, \beta_\lambda) \\
a_{t,i}|\lambda_{t,i} &\sim \mathcal{N}(a_{t,i}; 0, \lambda_{t,i}^{-1}) \\
\upsilon_t &\sim \mathcal{G}(\upsilon_t; \alpha_\upsilon, \beta_\upsilon) \\
\boldsymbol{g}_{t,k}|\boldsymbol{a}_t, \mathbf{K}_{t,k}, \upsilon_t &\sim \mathcal{N}(\boldsymbol{g}_{t,k}; \mathbf{K}_{t,k}\boldsymbol{a}_t, \upsilon_t^{-1}\mathbf{I}) \\
\omega_k &\sim \mathcal{G}(\omega_k; \alpha_\omega, \beta_\omega) \\
e_k|\omega_k &\sim \mathcal{N}(e_k; 0, \omega_k^{-1}) \\
\gamma_t &\sim \mathcal{G}(\gamma_t; \alpha_\gamma, \beta_\gamma) \\
b_t|\gamma_t &\sim \mathcal{N}(b_t; 0, \gamma_t^{-1}) \\
\epsilon_t &\sim \mathcal{G}(\epsilon_t; \alpha_\epsilon, \beta_\epsilon) \\
\boldsymbol{y}_t|b_t, e_\cdot, \boldsymbol{g}_{t,\cdot}, \epsilon_t &\sim \mathcal{N}\left(\boldsymbol{y}_t; \sum_{k=1}^{K} e_k \boldsymbol{g}_{t,k} + b_t \mathbf{1}, \epsilon_t^{-1}\mathbf{I}\right)
\end{aligned}
$$

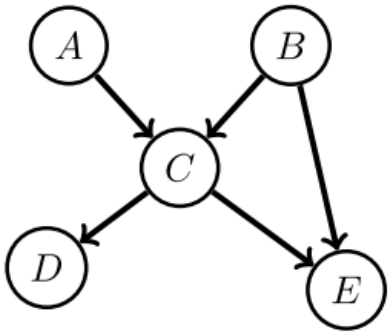
Example 2: Combining expert feedback with observed data

- A probabilistic model for treating expert feedback as additional data
- Helps to improve prediction accuracy when data are limited, as in precision medicine (Sundin et al. *Bioinformatics*, 2018)



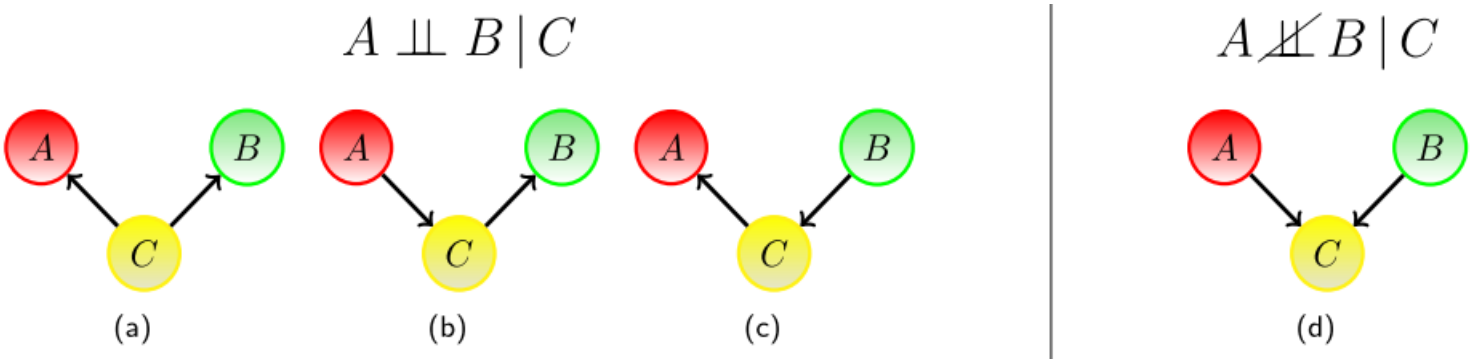
Reading independence statements from the DAG

- Motivating example: do the following independence statements hold:
  - $A \perp\!\!\!\perp B$
  - $A \perp\!\!\!\perp B | E$
  - $D \perp\!\!\!\perp E | C$?



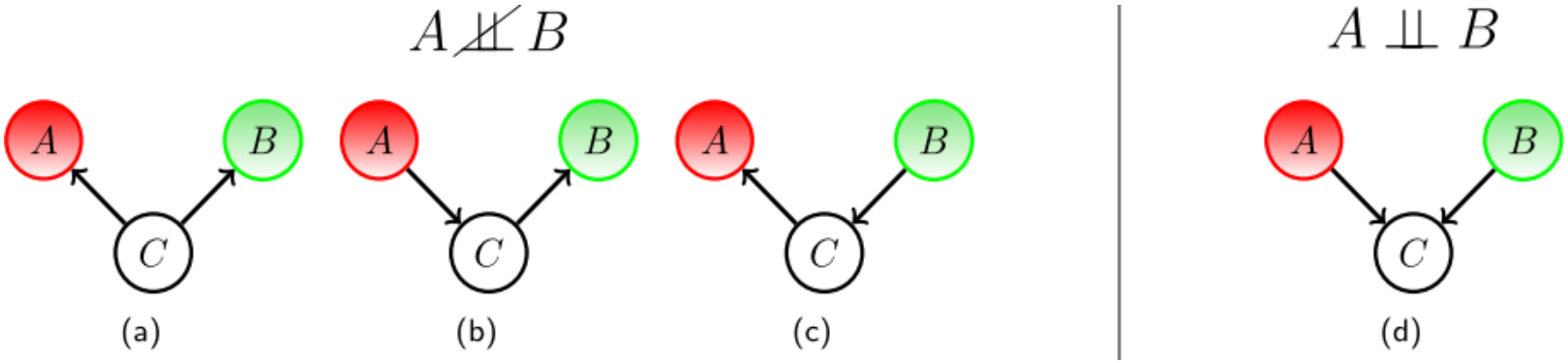
Independence in Bayesian networks (1/2)

- Possible BNs with three nodes and two links



- In (a), (b), and (c), A and B are **conditionally independent** given C.
  - $p(a, b|c) = p(a|c)p(b|c)$
- In (d), A and B are not conditionally independent given C
  - $p(a, b|c) \propto p(a)p(b)p(c|a, b)$

Independence in Bayesian networks (2/2)

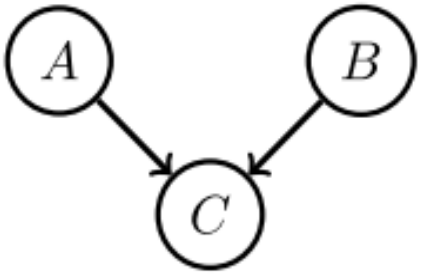


- In (a), (b), and (c), A and B are marginally dependent
- In (d) the variables A and B are **marginally independent**

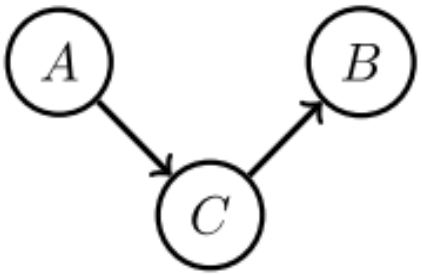
$$p(a,b) = \sum_c p(a,b,c) = \sum_c p(a)p(b)p(c|a,b) = p(a)p(b)$$

Collider

- A collider (v-structure, head-to-head meeting) has two incoming arrows along a chosen path



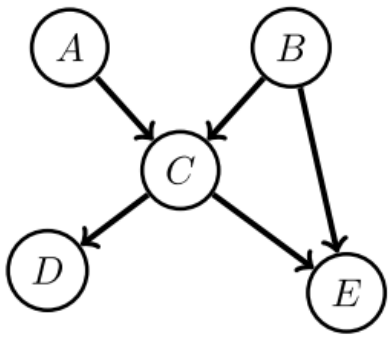
- C a collider
  - $A \perp\!\!\!\perp B$
  - $A \not\perp\!\!\!\perp B | C$
  - e.g. $A =$'Talented in sports', $B =$'Talented in maths', $C =$'Admitted to school'



- C a non-collider
  - $A \not\perp\!\!\!\perp B$
  - $A \perp\!\!\!\perp B | C$
  - e.g. $A =$'Cumulative sum of $n-1$ dice throws', $C =$'Cumsum of n throws', $B =$'Cumsum of $n+1$ throws'

Blocked paths

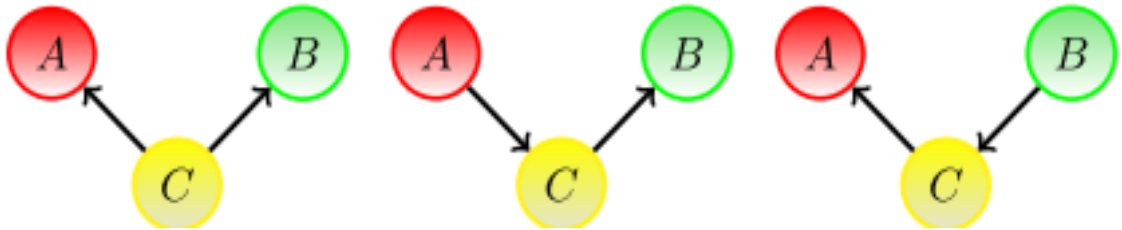
- **A path** between variables A and B **is blocked** by a set of variables $\mathcal{C}$, if
  - there is a collider in the path s.t. neither the collider nor any of its descendants is in the conditioning set $\mathcal{C}$
  - there is a non-collider in the path that is in the conditioning set $\mathcal{C}$.
- Sets of variables $\mathcal{A}$ and $\mathcal{B}$ are **d-separated** by $\mathcal{C}$ if all paths between $\mathcal{A}$ and $\mathcal{B}$ are blocked by $\mathcal{C}$.
  - d-separation implies: $\mathcal{A} \perp\!\!\!\perp \mathcal{B} | \mathcal{C}$



$A \perp\!\!\!\perp B$?
$A \perp\!\!\!\perp B | E$?
$D \perp\!\!\!\perp E | C$?

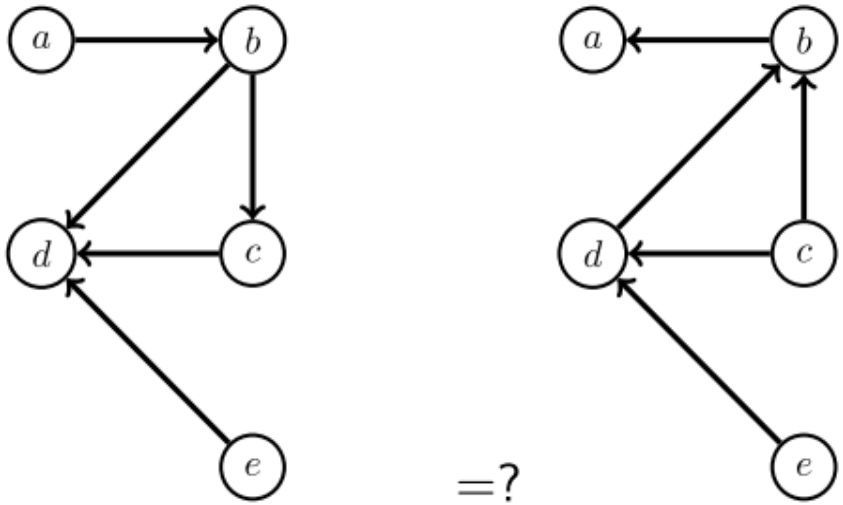
Markov equivalence

- Two graphs are **Markov equivalent**, if they
  - entail the same conditional independencies
  - equivalently: have the same d-separations
- For example:

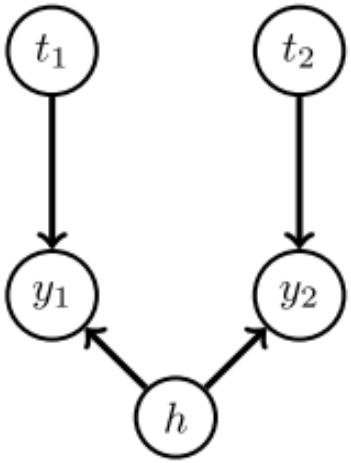


Determining Markov equivalence

- **skeleton:** undirected graph obtained by removing directions
- **immorality**: a collider structure $A \rightarrow C \leftarrow B$, such that there is no direct edge between A and B
- Two graphs are Markov equivalent if and only if they have the same skeleton and the same set of immoralities

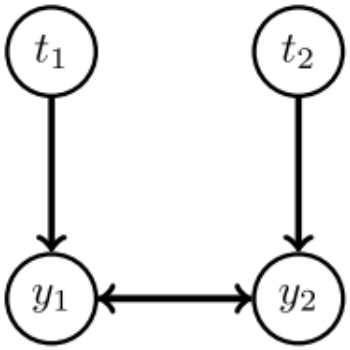


Limitations of expressibility*



- $t_1 \perp\!\!\!\perp t_2, y_2$ and $t_2 \perp\!\!\!\perp t_1, y_1$
- No Bayesian network for t_1, t_2, y_1, y_2 exists that could capture these independence statements (why?)

- A generalized class of models with bi-directed edges



How to select a DAG to model the system?

- Full graph can represent any distribution:

$$p(x_1, x_2, x_3, x_4) = p(x_1|x_2, x_3, x_4)p(x_2|x_3, x_4)p(x_3|x_4)p(x_4),$$

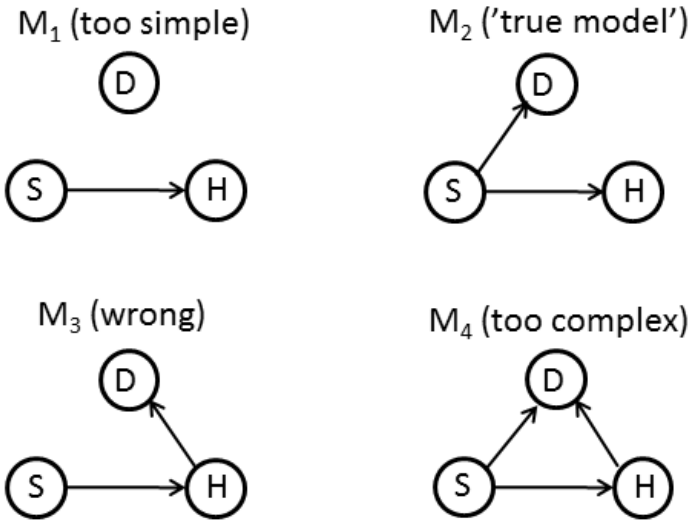
or equally valid

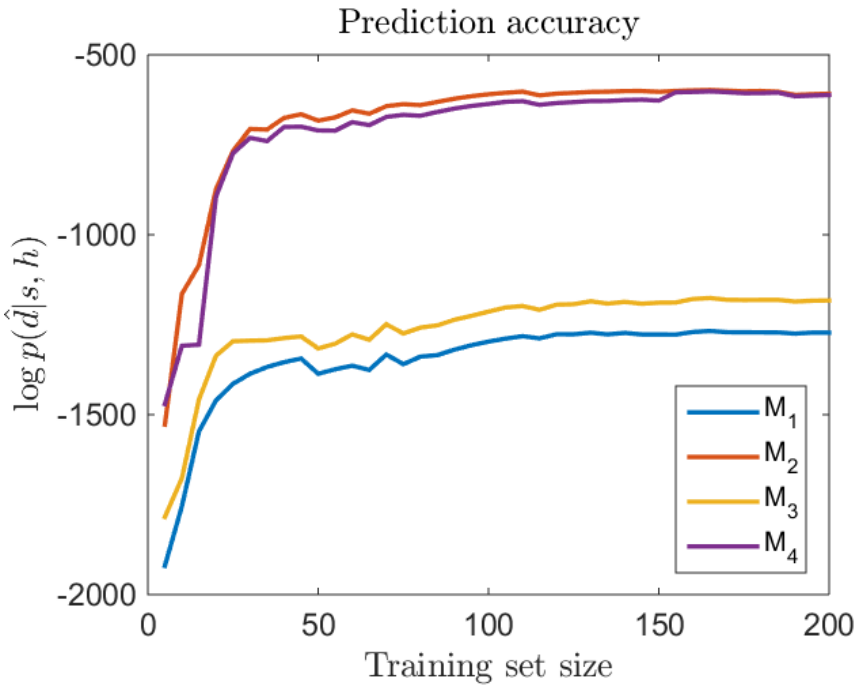
$$p(x_1, x_2, x_3, x_4) = p(x_3|x_4, x_1, x_2)p(x_4|x_1, x_2)p(x_1|x_2)p(x_2).$$

  - Misses all benefits of structure!
- Graph can only be determined up to Markov equivalence class

Example on using too simple or too complex a DAG (1/2)

- Simulate data from the 'true model'
- Train a model with training data, try to predict D given S and H in the test data
- S ='Smoking', H ='Hypertension', D ='(Some) Disease'





Example on using too simple or too complex DAG (2/2)

- Having a bit too simple model is much worse for prediction accuracy than having a bit too complex model
  - Advisable to include all components that may be useful in the prediction model
  - However: way too complex models waste data to learn redundant parameters
- Other interesting points:
  - M_3 seems better than M_1. Why?
  - The negative impact of having too complex a model is most severe when the amount of training data is limited (overfitting).

Possible ways to specify the graph

1. Construct the graph using assumptions about the system
   - add edges based on perceived **direct causalities**
   - Details in the following slides [1]
2. Learn structure from data
   - Ch. 9.5

- Before use, the model should always be checked
  - cross-validation
  - inspection or residuals
  - ...

[1] The slides about causal DAGs follow the derivations of Richard E. Neapolitan (2004) *Learning Bayesian Networks*, Ch. 1.5

Definition of causality (1/2)*

- Let S ='Sprinkler on', G ='Grass wet'
- By observing the values of S and G, we would surely find them dependent, so $p(s, g) \neq p(s)p(g)$
- Non-symmetric:
  - Turning the sprinkler on makes grass wet
  - Watering the grass (by some means other than the sprinkler) does not turn the sprinkler on
  - Interpretation: S is a cause of G

www.123rf.com

Definition of causality (2/2)*

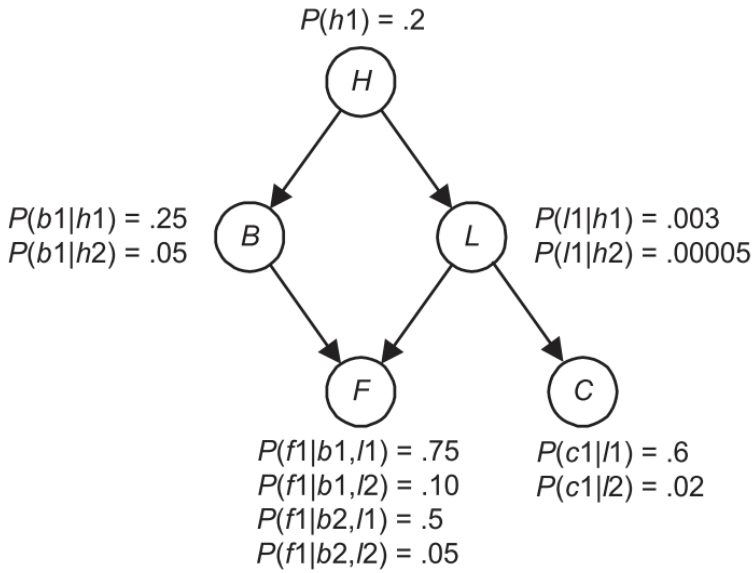
- The causality can be defined by intervetions (manipulations of variables)
  - Set the value of a putative cause to a certain value.
  - Investigate if the distribution of the putative effect changes
- For example (S ='Sprinkler on', G ='Grass wet'):

$$p(G = 1|do(S = 1)) \neq p(G = 1|do(S = 0))$$
$$p(S = 1|do(G = 1)) = p(S = 1|do(G = 0))$$

  Therefore: S is a cause of G
- Interventions form the basis of *randomized controlled experiments* that are used in clinical trials, for example to assess the effectiveness of a drug.

Causal DAG

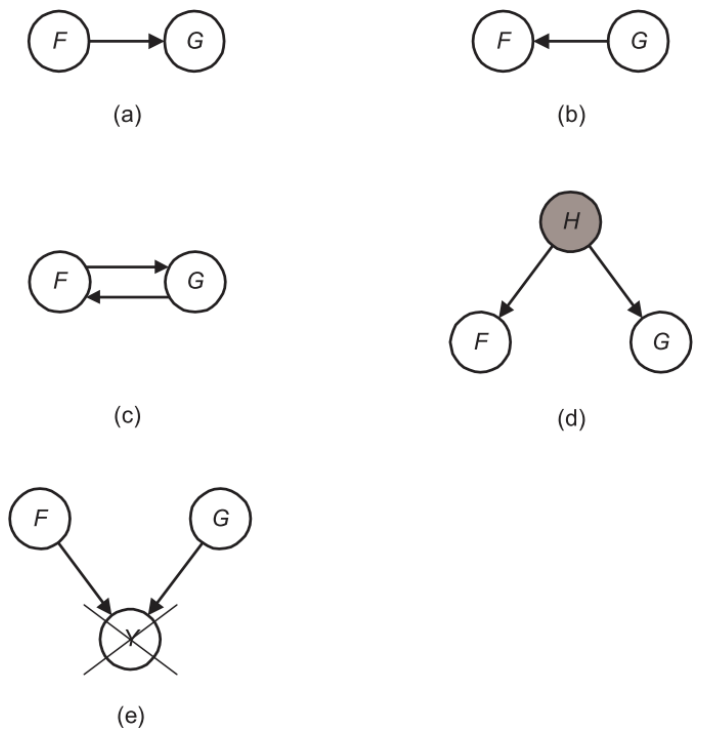
- Suppose our understanding of the causal influences is the following:
  - A history of smoking (H) causes bronchitis (B) and lung cancer (L).
  - Bronchitis (B) and lung cancer (L) can cause fatigue (F).
  - Lung cancer can cause a positive chest X-ray (C)
- A few additional assumptions aside (see next slide), the following DAG with empirically determined conditional distributions could be used to represent our knowledge



Neapolitan (2004, Fig 10)

Underlying assumptions in causal DAGs

- Correlation between F and G could be explained by the following causal structures:
- When constructing DAGs using causal edges, we must assume
  - No **feedback loops** (c)
  - No **hidden common causes** (d)
  - No **selection bias** (e)

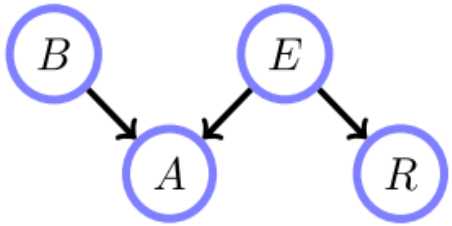


Neapolitan (2014, Fig 1.12)

Inference

- Inference corresponds to using the distribution to answer a question about the environment.
- Examples
  - What is the probability $p(x = 4|y = 1, z = 2)$?
  - What is the most likely joint state of the distribution $p(x, y)$?
  - What is the probability the stock market will do down tomorrow?
- Computational Efficiency
  - For singly-connected graps (e.g. trees), there exist efficient algorithms based on the concept of message passing.
  - In general, the case of multiply-connected models is computationally inefficient.
  - Ch 5 & 6

Computation - example (1/3)



- A: 'Alarm is on', B: 'There's a burglar in the house', E:'There's an earthquake', R:'Radio reports of an earthquake'
- Compute $p(B = 1|A = 1)$,the probability that there's a burglar, given the alarm is on.
- Conditional probabilities:

$p(A = 1\|B, E)$	B	E
0.9999	1	1
0.99	1	0
0.99	0	1
0.0001	0	0

$p(R = 1\|E)$	E
1	1
0	0

$$p(E = 1) = 0.000001$$

$$p(B = 1) = 0.01$$

Computation - example (2/3)

$$p(B=1|A=1) \overset{1}{=} \frac{p(B=1,A=1)}{p(A=1)}$$
$$\overset{2}{=} \frac{\sum_e \sum_r p(B=1,A=1,E=e,R=r)}{\sum_b \sum_e \sum_r p(B=b,A=1,E=e,R=r)}$$
$$\overset{3}{=} \frac{\sum_e \sum_r p(A=1|B=1,E=e)p(B=1)p(E=e)p(R=r|E=e)}{\sum_b \sum_e \sum_r p(A=1|B=b,E=e)p(B=b)p(E=e)p(R=r|E=e)}$$

1: definition of conditional probability, 2: marginalization, 3: factorization of the joint distribution according to the BN

Computation - example (3/3)

By reordering to simplify computations, we get further that

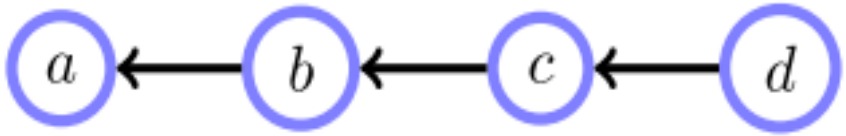
$$\ldots = \frac{\sum_e p(A=1|B=1,E=e)p(B=1)p(E=e)\sum_r p(R=r|E=e)}{\sum_b \sum_e p(A=1|B=b,E=e)p(B=b)p(E=e)\sum_r p(R=r|E=e)},$$

and, because $\sum_r p(R=r|E=e) = 1$, we finally get

$$\ldots = \frac{\sum_e p(A=1|B=1,E=e)p(B=1)p(E=e)}{\sum_b \sum_e p(A=1|B=b,E=e)p(B=b)p(E=e)} \approx 0.99.$$

- Note: even further re-ordering would have been possible.
- Note 2: People are not in general very good at estimating this kind of probabilities. Could you have come up with the result approximately from the given probabilities, without actually doing the computations?

Message passing - a simple example (1/2)*



- Compute marginal $p(a = 0)$ in the given graph

$$p(a = 0) = \sum_{b \in \{0,1\}} \sum_{c \in \{0,1\}} \sum_{d \in \{0,1\}} p(a = 0, b, c, d)$$

$$= \sum_{b \in \{0,1\}} \sum_{c \in \{0,1\}} \sum_{d \in \{0,1\}} p(a = 0|b)p(b|c)p(c|d)p(d)$$

Naive computation: summation of $2^{T-1} = 8$ terms

Message passing - a simple example (2/2)*

- A more efficient approach is to eliminate one variable at a time:

$$p(a=0) = \sum_{b\in\{0,1\}} \sum_{c\in\{0,1\}} p(a=0|b)p(b|c) \underbrace{\sum_{d\in\{0,1\}} p(c|d)p(d)}_{\gamma_d(c)}$$

$$= \sum_{b\in\{0,1\}} p(a=0|b) \underbrace{\sum_{c\in\{0,1\}} p(b|c)\gamma_d(c)}_{\gamma_c(b)}$$

$$= \sum_{b\in\{0,1\}} p(a=0|b)\gamma_c(b)$$

Computational cost: $2 \times (T-1) = 6$ summations

- *Variable elimination*: eliminate variables starting from the end of the chain (or a leaf of a tree)
- Pass a message (information) from the eliminated variable to its neighbor in the chain.

Main points

- The definition of Bayesian networks
- Reading (conditional) independendies using d-separation
- Understanding why it is important to select the model (network) structure appropriately
- Computation of marginal and conditional distributions using a BN

Advanced probabilistic methods

Lecture 3: Multivariate Gaussian, Bayesian linear models

Pekka Marttinen

Aalto University

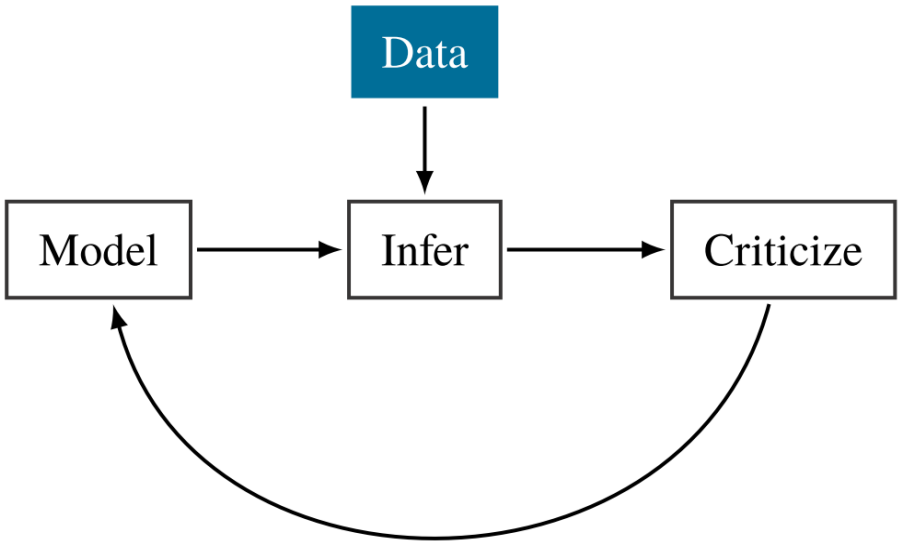
January, 2021

Lecture 3 overview

- Gaussian distribution
  - Bayesian parameter learning
- Multivariate Gaussian distribution
  - Characterization
  - Useful identities
- Bayesian Linear Parameter Models (LPMs)
  - Posterior computation (given fixed hyperparameters)
- Ch. 8 & 18 (until the end of Section 18.1.1) in Barber's book

Recall from lecture 1

- Tools for probabilistic modeling
  - **Models**: Bayesian networks, sparse Bayesian linear regression, Gaussian mixture models, latent linear models
  - **Methods for inference**: maximum likelihood, maximum a posteriori (MAP), analytical, Laplace approximation, expectation maximization (EM), Variational Bayes (VB), stochastic variational inference (SVI)
  - **Ways to select between models**



Box's loop (Blei, 2014)

What is a model?

- A model specifies a probability distribution for a random variable Y, and it is often affected by some parameter θ. The model can be denoted as $p(y|\theta)$.
- Fitting the model (i.e. inference) corresponds to learning the value (or the distribution) of θ, after some data y have been observed.

Prior, Likelihood, Posterior

- *Bayes' rule* tells us how to update our prior beliefs about variable θ in light of the data y to a posterior belief:

$$\underbrace{p(\theta|y)}_{\text{posterior}} = \frac{\underbrace{p(y|\theta)}_{\text{likelihood}}\underbrace{p(\theta)}_{\text{prior}}}{\underbrace{p(y)}_{\text{evidence}}}.$$

  The evidence is also called the marginal likelihood.

- $p(y|\theta)$ is the probability that the model generates the observed data y when using parameter θ
  - $L(\theta) \equiv p(y|\theta)$, with y held fixed, is called the *likelihood*
  - $f(y) \equiv p(y|\theta)$, with θ held fixed, is called the *observation model*
- "*Methods for inference*" = Bayes' rule + some algorithm to do the actual computations (on this course)

Point estimates for parameters

- The *Maximum A Posteriori (MAP)* parameter value, which maximizes the posterior

$$\theta_* = \arg\max_\theta p(\theta|y)$$

- The Maximum likelihood assignment (ML)

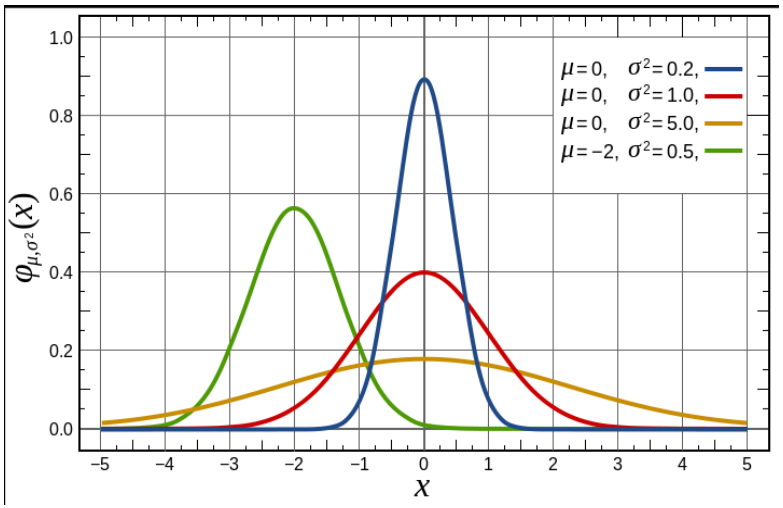
$$\theta_* = \arg\max_\theta p(y|\theta)$$

- The full posterior distribution $p(\theta|y)$ tells also of the uncertainty about the value of θ.

Gaussian distribution

- $X \sim N(\mu, \sigma^2)$
- Parameters: μ: mean, σ^2: variance
- Inverse of the variance, $\lambda = 1/\sigma^2$, is called the precision
- Standard deviation σ
- 95% credible interval equals approximately $[\mu - 2\sigma, \mu + 2\sigma]$
- PDF:

$$N(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$



Gaussian (or normal) distribution (wikip.)

Bayesian estimation of the mean of a Gaussian (1/2)

- Suppose we have observations $x = (x_1, \ldots, x_n)$ from $N(\mu, \sigma^2)$, where σ^2 is known.
- To learn μ, we specify a prior

$$\mu \sim N(\mu_0, \tau_0^2)$$

- Posterior

$$\begin{aligned} p(\mu|x) &= \frac{p(x|\mu)p(\mu)}{p(x)} \propto p(\mu)p(x|\mu) \\ &= \frac{1}{\sqrt{2\pi}\tau_0} e^{-\frac{1}{2\tau_0^2}(\mu-\mu_0)^2} \times \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(x_i-\mu)^2} \\ &\propto e^{-\frac{1}{2\tau_0^2}(\mu-\mu_0) - \frac{1}{2\sigma^2}\sum_i (x_i-\mu)^2} \\ &= \ldots \text{(details in BDA course)} \end{aligned}$$

Bayesian estimation of the mean of a Gaussian (2/2)

- Posterior

$$p(\mu|x) \propto e^{-\frac{1}{2\tau_n^2}(\mu-\mu_n)^2}$$
$$\propto N(\mu|\mu_n, \tau_n^2)$$

where

$$\mu_n = \frac{\frac{1}{\tau_0^2}\mu_0 + \frac{n}{\sigma^2}\overline{x}}{\frac{n}{\sigma^2} + \frac{1}{\tau_0^2}} \quad \text{and} \quad \frac{1}{\tau_n^2} = \frac{n}{\sigma^2} + \frac{1}{\tau_0^2}.$$

- Posterior precision $1/\tau_n^2$: sum of prior precision $1/\tau_0^2$ and data precision n/σ^2
- Posterior mean μ_n: precision weighted average of prior mean μ_0 and data mean $\overline{x}$.

Conjugate prior distributions (1/2)

- In the previous example

$$\text{Prior: } \mu \sim N(\mu_0, \tau_0^2)$$
$$\text{Posterior: } \mu \sim N(\mu_n, \tau_n^2).$$

  If the prior and posterior belong to the same family of distributions, we say that the prior is conjugate to the likelihood used.

  - For example, normal prior $\mu \sim N(\mu_0, \tau_0^2)$ is conjugate to the normal likelihood $N(x|\mu, \sigma^2)$.

- Conjugacy is useful, because it makes computations easy.

Conjugate prior distributions (2/2)

- With conjugate prior, the posterior is available in a closed form

$$p(\theta|x) \propto p(x|\theta)p(\theta)$$

  - Drop all terms not depending on θ
  - Recognize the result as a density function belonging to the same family of distributions as the prior $p(\theta)$, but with different parameters.
- Examples (likelihood - conjugate prior):
  - Likelihood for normal mean - Normal prior
  - Likelihood for normal variance - Inverse-Gamma prior
  - Bernoulli - Beta
  - Binomial - Beta
  - Exponential - Gamma
  - Poisson - Gamma

Gaussian distribution, unknown mean and precision (1/2)

- Suppose we have observations $x = (x_1, \ldots, x_n)$ from $N(\mu, \lambda^{-1})$, where both the mean μ and the precision λ are unknown.
- The conjugate prior distribution is the normal-gamma distribution

$$\begin{aligned} p(\mu, \lambda | \mu_0, \beta, a, b) &= N(\mu | \mu_0, (\beta\lambda)^{-1}) \mathrm{Gam}(\lambda | a, b) \\ &\equiv \text{Normal-Gamma}(\mu, \lambda | \mu_0, \beta, a, b) \end{aligned}$$

Note the dependency of the prior of μ on the value of λ.

Gaussian distribution, unknown mean and precision (2/2)

- The conjugate prior distribution is the normal-gamma distribution

$$p(\mu, \lambda|\mu_0, \beta, a, b) = \text{Normal-Gamma}(\mu, \lambda|\mu_0, \beta, a, b)$$

- Posterior

$$p(\mu, \lambda|x) = \text{Normal-Gamma}(\mu, \lambda|\mu_n, \beta_n, a_n, b_n),$$

with

$$\begin{aligned}
\mu_n &= \frac{\beta\mu_0 + n\overline{x}}{\beta + n} \\
\beta_n &= \beta + n \\
a_n &= a + \frac{n}{2} \\
b_n &= b + \frac{1}{2}\left(ns + \frac{\beta n(\overline{x} - \mu_0)^2}{\beta + n}\right)
\end{aligned}$$

Gaussian distribution, unknown mean and precision, example (1/2)

- Simulate samples from $N(\mu = 2, \sigma^2 = 0.25)$
  - precision $\lambda = 4$
- Try to learn μ and λ
- Specify prior

$$p(\mu, \lambda | \mu_0, \beta, a, b) = \text{Normal-Gamma}(\mu, \lambda | \mu_0, \beta, a, b)$$

with

$$\mu_0 = 0, \quad \beta = 0.001, \quad a = 0.01, \quad b = 0.01$$

- See: *normal_example.m*

Gaussian distribution, unknown mean and precision, example (2/2)

- When μ and λ have distribution

  $$\text{Normal-Gamma}(\mu, \lambda | \mu_n, \beta_n, a_n, b_n) = N(\mu | \mu_n, (\beta_n \lambda)^{-1}) \text{Gam}(\lambda | a_n, b_n),$$

  marginal distribution of λ can be plotted using the PDF of $\text{Gam}(\lambda | a_n, b_n)$
- To plot the marginal distribution of μ, we need to take the dependence on λ into account.
  - we compute the marginal distribution of μ by averaging over $N(\mu | \mu_n, (\beta_n \lambda_i)^{-1})$, for multiple λ_i simulated from $\text{Gam}(\lambda | a_n, b_n)$
  - (could also be done analytically...)

Consistency

- If $p(x|\theta_t)$ is the true data generating mechanism, and A is a neighborhood of θ_t, then

$$p(\theta \in A|x) \overset{n\to\infty}{\to} 1.$$

- The posterior distribution concentrates around the true value (if such a value exists!). See the *normal_example.m*
- It follows that

$$\overline{\theta}_{MAP} \overset{n\to\infty}{\to} \theta_t \quad \text{and} \quad \overline{\theta}_{ML} \overset{n\to\infty}{\to} \theta_t$$

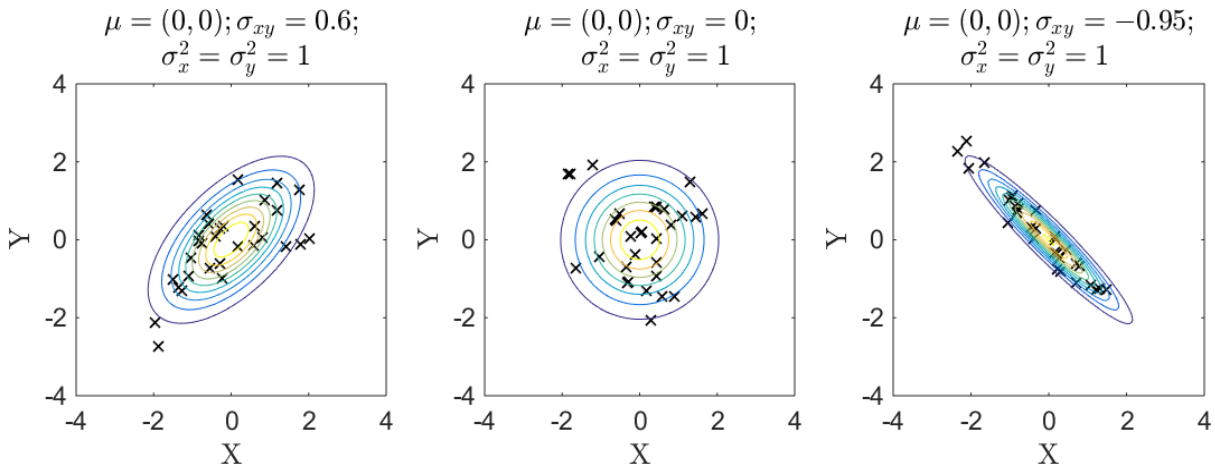
Multivariate Gaussian distribution

$$N_D(x|\mu,\Sigma) \equiv (2\pi)^{-\frac{D}{2}}|\Sigma|^{-\frac{1}{2}}e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

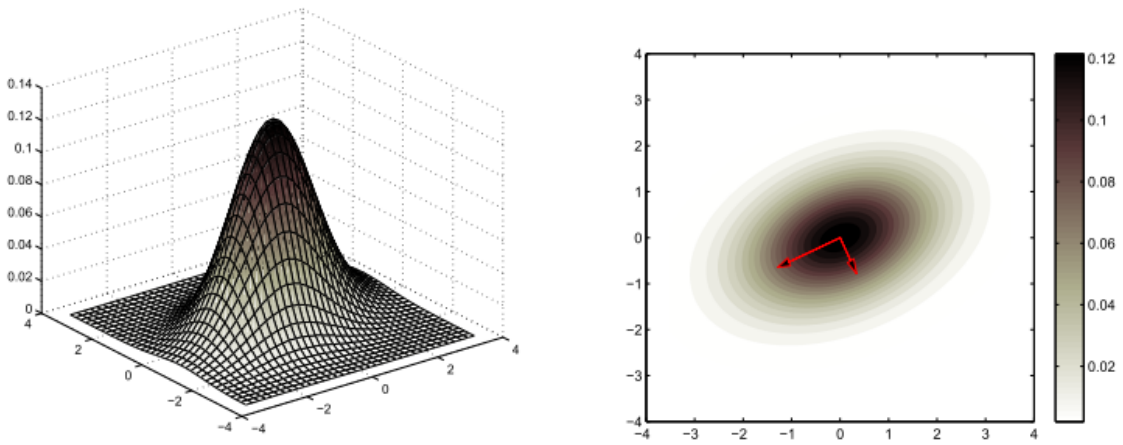
- D: dimension, μ: mean, Σ: covariance matrix. With $D = 2$:

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{bmatrix}$$

- $\sigma_{12} = \sigma_{21}$: covariance between x_1 and x_2. (tells direction of dependency)
- $\rho_{12} = \sigma_{12}/(\sigma_1\sigma_2)$:correlation between x_1 and x_2. (direction and strength)



Multivariate Gaussian - characterization (1/2)



- Eigendecomposition

$$\Sigma = E\Lambda E^T,$$

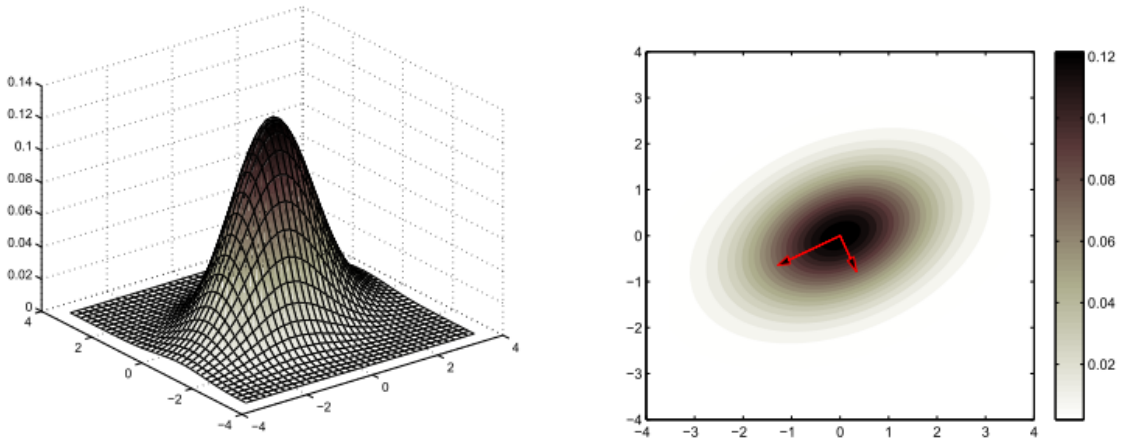
where $E^T E = I$ and $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_D)$.

- Now the transformation

$$y = \Lambda^{-\frac{1}{2}} E^T (x - \mu)$$

can be shown to have the distribution $N_D(0, I)$ (product of D independent standard Gaussians)

Multivariate Gaussian - characterization (2/2)



- Thus, $x = E\Lambda^{\frac{1}{2}} y + \mu$ with distribution $N_D(\mu, \Sigma)$ is obtained from standard independent Gaussians y by
  - *scaling* by the square roots of eigenvalues
  - *rotating* by the eigenvectors
  - *shifting* by adding the mean

Marginalization and conditioning (1/2)

- Let $z \sim N(\mu, \Sigma)$ and consider partitioning it as:

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

with

$$\mu = \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix} \quad \text{and} \quad \Sigma = \begin{pmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{pmatrix}.$$

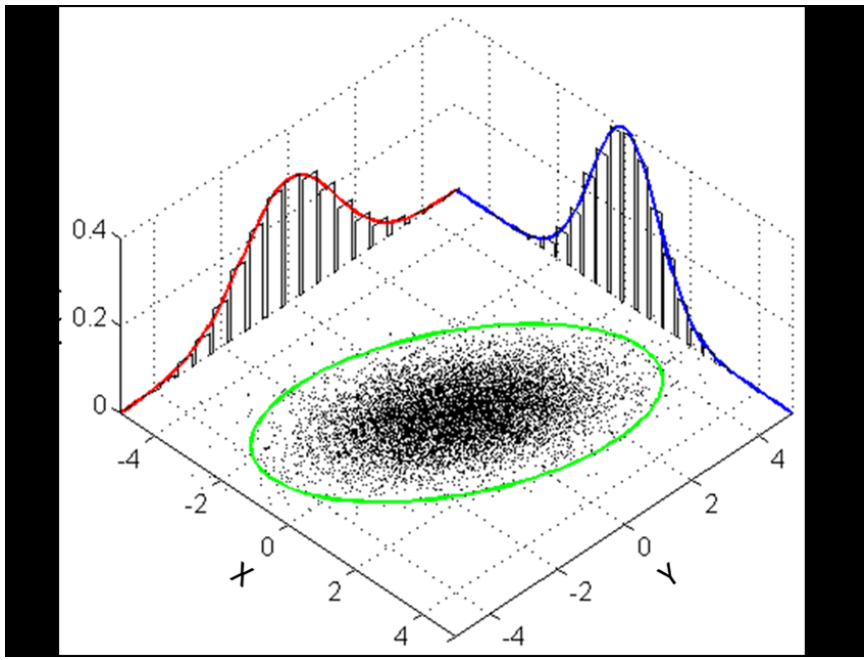
Marginalization and conditioning (2/2)

- Then

$$p(x) \sim N(\mu_x, \Sigma_{xx}) \quad \textbf{(marginalization)}$$

$$p(x|y) = N(\mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}(y - \mu_y), \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}) \quad \textbf{(conditioning)}$$

$\Longrightarrow$Marginals and conditionals of M-V Gaussians are still M-V Gaussian.



Important identities related to the multivariate Gaussian

- **Linear transformation**: if

$$y = Mx + \eta,$$

where $x \sim N(\mu_x, \Sigma_x)$ and $\eta \sim N(\mu, \Sigma)$,then

$$p(y) = N(y|M\mu_x + \mu, M\Sigma_x M^T + \Sigma)$$

- **Completing the square**:

$$\frac{1}{2}x^T Ax - b^T x = \frac{1}{2}(x - A^{-1}b)^T A(x - A^{-1}b) - \frac{1}{2}b^T A^{-1}b$$

From which one can derive, for example

$$\int \exp(-\frac{1}{2}x^T Ax + b^T x)dx = \sqrt{\det(2\pi A^{-1})}\exp(\frac{1}{2}b^T A^{-1}b)$$

Multivariate Gaussian - ML fitting

- Let $x = (x_1, \ldots, x_n)$ be from $N(\mu, \Sigma)$ with unknown μ and Σ. Log-likelihood, assuming data are $i.i.d.$:

$$L(\mu, \Sigma) = \sum_{i=1}^{N} \log p(x_i|\mu, \Sigma)$$

$$= -\frac{1}{2} \sum_{i=1}^{N} (x_i - \mu)^T \Sigma^{-1} (x_i - \mu) - \frac{N}{2} \log \det(2\pi\Sigma)$$

Multivariate Gaussian - ML fitting

- Differentiate $L(\mu, \Sigma)$ w.r.t. the vector μ:

$$\nabla_\mu L(\mu, \Sigma) = \sum_{i=1}^{N} \Sigma^{-1}(x_i - \mu)$$

Equating to zero gives

$$\sum_{i=1}^{N} \Sigma^{-1} x_i = N\Sigma^{-1}\mu.$$

Thus we get

$$\widehat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

- Similarly one can derive:

$$\widehat{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})(x_i - \overline{x})^T$$

Multivariate Gaussian - Bayesian learning*

- Gaussian-Wishart is the conjugate prior, when $X_i \sim N(\mu, \Lambda)$ and both mean μ and precision Λ are unknown:

$$p(\mu, \Lambda | \mu_0, \beta, W, \nu) = N(\mu | \mu_0, (\beta\Lambda)^{-1}) \mathcal{W}(\Lambda | W, \nu)$$

- If X_i are scalar, this is equivalent to the Gaussian-Gamma distribution.
- Posterior

$$p(\mu, \Lambda | x) = N(\mu | \mu_n, (\beta_n\Lambda)^{-1}) \mathcal{W}(\Lambda | W_n, \nu_n)$$

Wishart distribution*

- Wishart distribution is a distribution for nonnegative-definite matrix-valued random variables

$$\Lambda \sim \mathcal{W}(\Lambda|W,\nu)$$

$$E(\Lambda) = \nu W$$
$$\mathrm{Var}(\Lambda_{ij}) = n(w_{ij}^2 + w_{ii}w_{jj})$$

- Further: exercises...

Linear models with Gaussian noise

- **Data** $\mathcal{D} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, N\}$
  - $\mathbf{x}_i$: the input
  - y_i: the output
- **Model**:

$$y = \underbrace{f(\mathbf{w}, \mathbf{x})}_{\text{clean output}} + \underbrace{\eta}_{\text{noise}}, \quad \eta \sim N(0, \beta^{-1})$$

- In the simplest case

$$\begin{aligned} f(\mathbf{w}, \mathbf{x}) &= \mathbf{w}^T \mathbf{x} \\ &= w_1 x_1 + \ldots + w_D x_D \end{aligned}$$

- The *parameters* w_i are also called the *weights*

Bayesian linear parameter models

- A prior distribution $p(\mathbf{w}|\alpha)$ is placed on the weights $\mathbf{w}$.
- The posterior distribution $p(\mathbf{w}|\mathcal{D}, \Gamma)$ can be computed, and reflects the uncertainty of the parameters.

Prior distribution

- A Gaussian prior distribution may placed on $\mathbf{w}$:

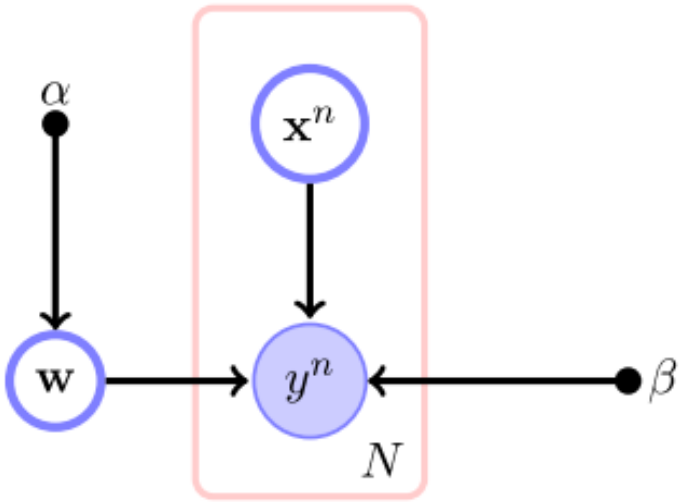
$$p(\mathbf{w}|\alpha) = N(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$$

$$= \prod_{i=1}^{D} N(w_i|0, \alpha^{-1}) = \left(\frac{\alpha}{2\pi}\right)^{\frac{D}{2}} e^{-\frac{\alpha}{2}\sum_i w_i^2}$$

- Posterior

$$\log p(\mathbf{w}|\Gamma, \mathcal{D}) = -\frac{\beta}{2}\sum_{i=1}^{N}\left[y_i - \mathbf{w}^T\mathbf{x}_i\right]^2 - \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} + \text{const}$$

Hyperparameters



- α: *precision* of the *regression weights*
  - determines the amount of regularization
  - large precision $\longrightarrow$ small variance $\longrightarrow$ weights are close to zero
- β: *precision* of the noise
- $\Gamma = \{\alpha, \beta\}$ are called the **hyperparameters** (in the course book...)

Posterior distribution

- Posterior distribution is obtained by completing the square (left as an exercise):

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

where

$$S = \left(\alpha I + \beta \sum_{i=1}^{N} \mathbf{x}_i {\mathbf{x}_i}^T\right)^{-1}, \qquad \mathbf{m} = \beta S \sum_{i=1}^{N} y_i \mathbf{x}_i$$

- Mean prediction

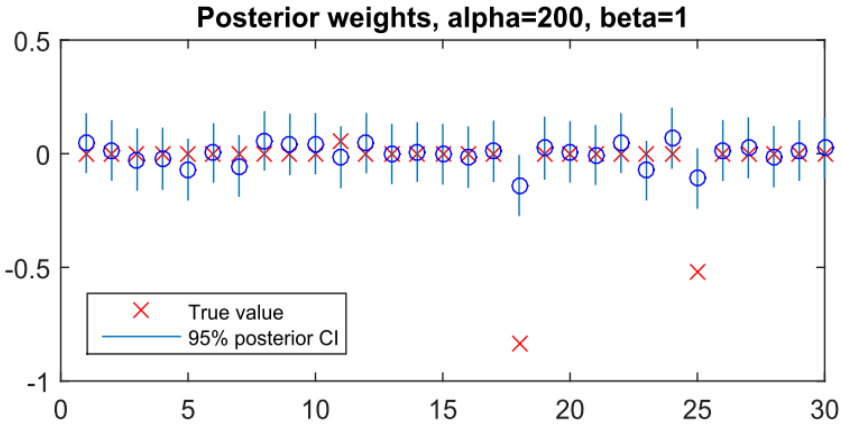
$$\widetilde{y} = \int \mathbf{w}^T \mathbf{x} \times p(\mathbf{w}|\Gamma, \mathcal{D}) d\mathbf{w} = \mathbf{m}^T \mathbf{x}$$

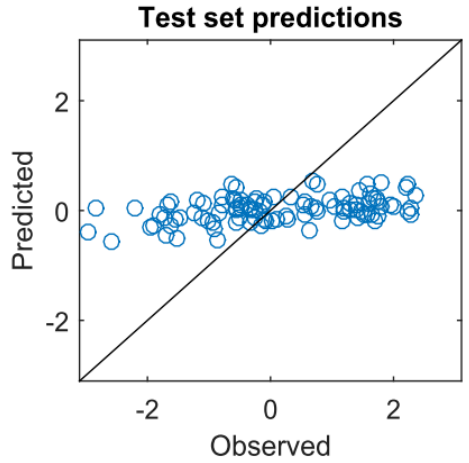
Example, impact of hyperparameters (1/3)

- Setup: simulate $y = \mathbf{w}_{true}^T \mathbf{x} + \epsilon$, where $\epsilon \sim N(0, \beta^{-1})$ and $\beta = 1$
- The goal is to investigate how hyperparameter α affects the posterior distribution of the parameters $\mathbf{w}$

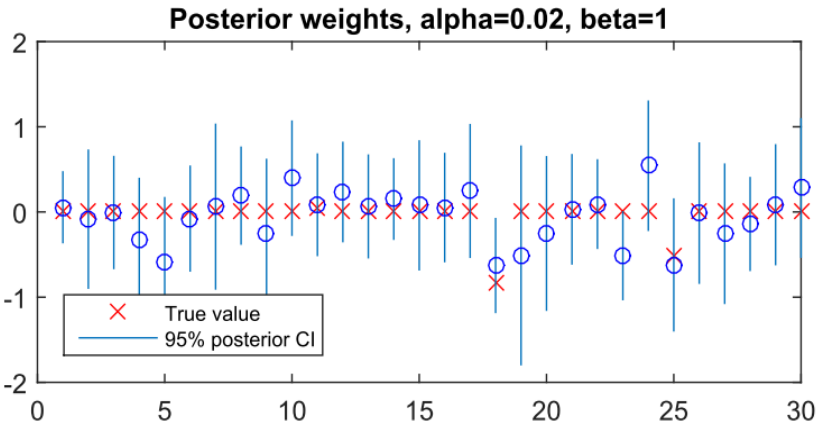
Example, impact of hyperparameters (2/3)

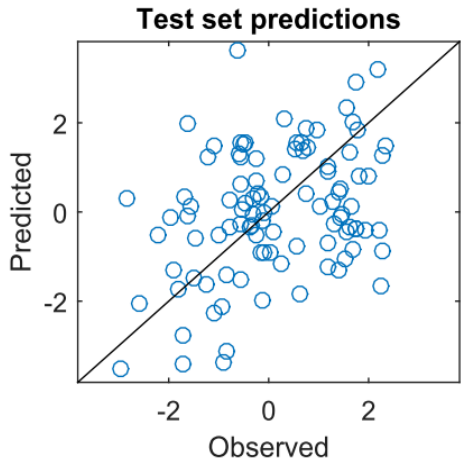
- Too large α, $Var(y - \widetilde{y}) = 1.54$ (Original $Var(y) = 1.75$)





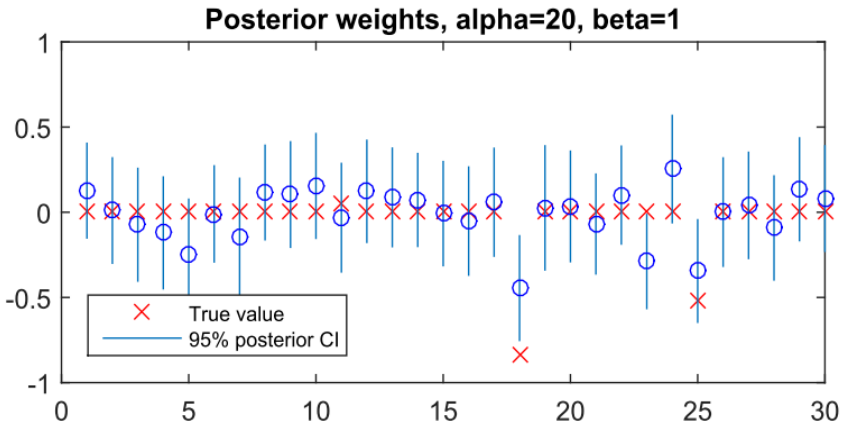
- Too small α, $Var(y - \widetilde{y}) = 2.48$

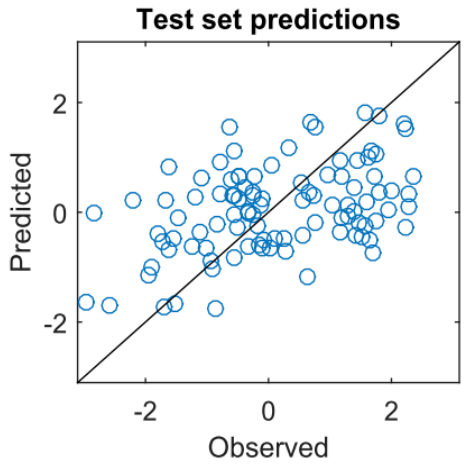




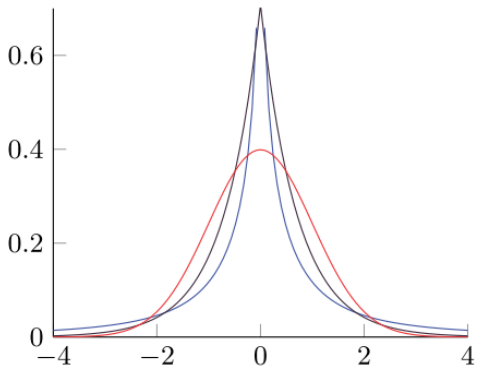
Example, impact of hyperparameters (3/3)

- About good α, $Var(y - \widetilde{y}) = 1.46$
- A compromise between bias and variance



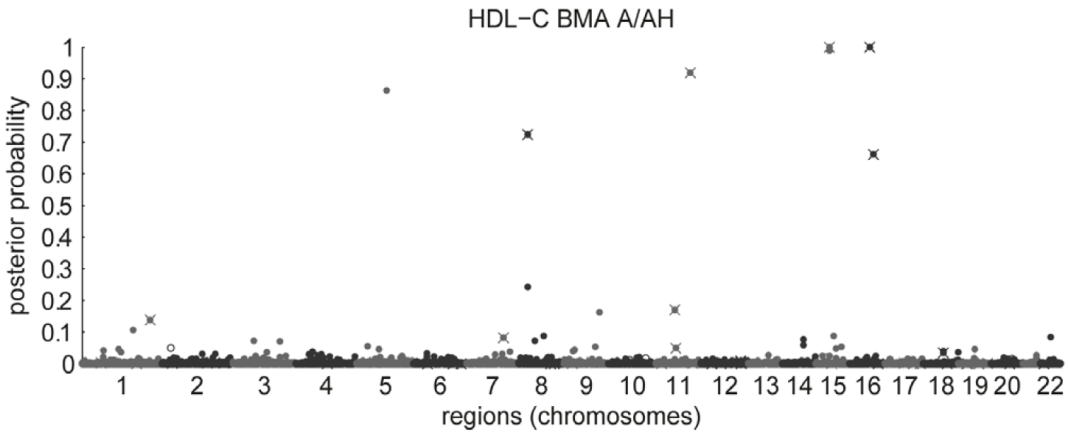


- Other sparse priors (e.g., Laplace, horse-shoe, spike-and-slab):



Example: genetic association studies

- Analysis of $\sim 1,000,000$ genetic polymorphisms in $\sim 50,000$ genomic regions (Peltola et al., 2012, *PLoS ONE*).
- *Spike-and-slab* prior on regression weights



Important points

- Bayesian learning of the Gaussian distribution using conjugate priors
- Multivariate Gaussian
  - Characterization
  - Marginal & conditional distributions
  - Linear transformation & completing the square
- By placing a Gaussian prior on the parameters of linear regression, the posterior is also Gaussian.
- Meaning and impact of hyperparameters in Bayesian linear regression.

Advanced probabilistic methods

Lecture 4: ML-II, Laplace approximation, and Gaussian mixtures

Pekka Marttinen

Aalto University

February, 2021

Lecture 4 overview

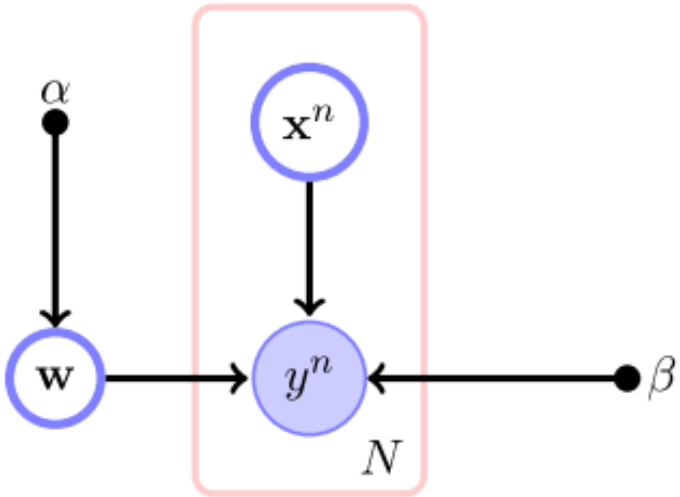
- Bayesian Linear Parameter Models (LPMs), continued
  - Lecture 3: Posterior computation given fixed hyperparameters
  - ML-II: Determining hyperparameters
  - Example using radial basis functions
- Logistic regression for classification
  - Laplace approximation
- Gaussian mixture models (GMMs)
- Suggested reading:
  - Barber, Ch. 18
  - Bishop, *Pattern Recognition and Machine Learning*, p. 110-113 (2.3.9): Mixtures of Gaussians

Recap: Bayesian linear regression

- **Data:** $\mathcal{D} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, N\}$
- **Model:**

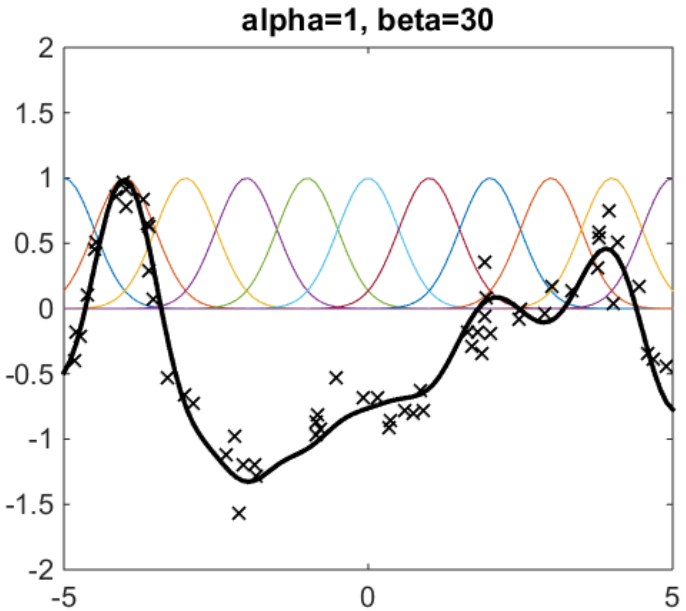
$$y_i = \mathbf{w}^T \mathbf{x}_i + \eta_i, \quad i = 1, \ldots, N$$
$$\eta_i \sim N(0, \beta^{-1}), \quad \mathbf{w} \sim N(\mathbf{0}, \alpha^{-1}\mathbf{I})$$

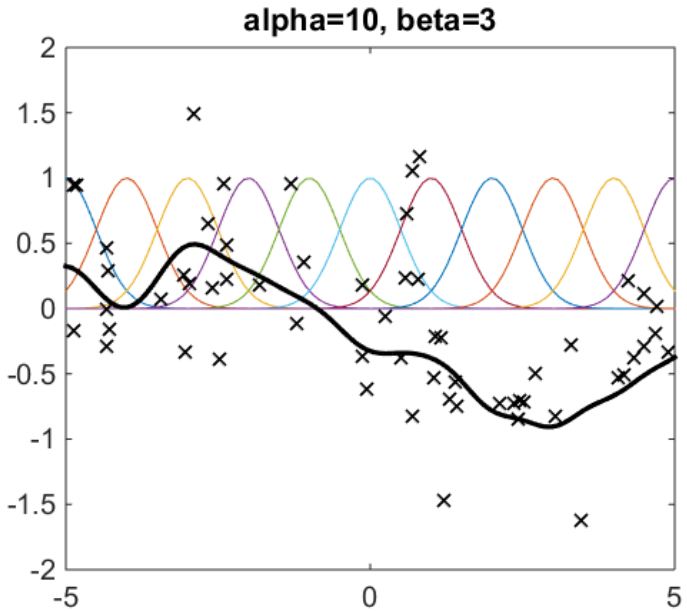
- **Parameters:** $\mathbf{w}$ called *weights* or *regression coefficients*
- **Hyperparameters:** $\Gamma = (\alpha, \beta)$



Non-linear transformation of the inputs

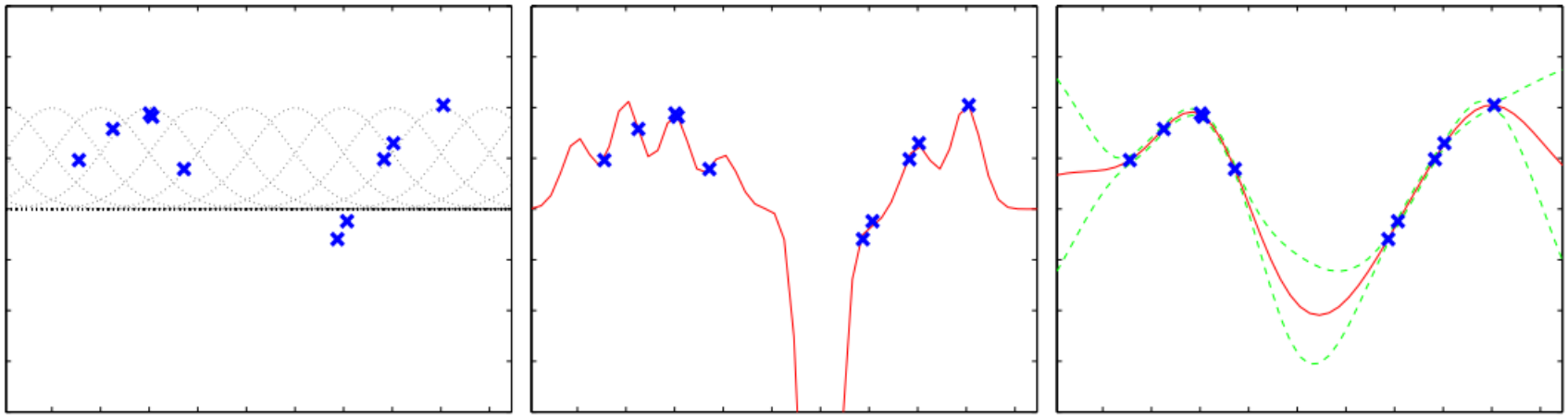
- Assume model $y_i = \mathbf{w}^T \phi(\mathbf{x}_i) + \eta_i$
- $\phi(\mathbf{x}_i)$ represent some transformation of $\mathbf{x}_i$ and are called *basis functions*
- Example
  - weights drawn from $N(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$; β is the noise precision.
  - $\mathbf{w} = (-0.7, 1.1, -0.8, -1.1, -0.8, -0.6, -0.6, 0.2, -0.2, 0.6, -0.9)$ for **radial basis functions** ordered from left to right (left panel)





Importance of learning hyperparameters

- (a): raw data and 15 radial basis functions $\phi_i(x) = \exp\left(-0.5(x - c_i)^2/\lambda^2\right)$ with $\lambda = 0.03^2$ and c_i spread evenly over the input space
- (b): predictions with $\beta = 100$ and $\alpha = 1$ (severe overfitting)
- (c): predictions with ML-II fitted hyperparameter values



(a) (b) (c)

Determining hyperparameters

- The hyperparameter posterior distribution is

$$p(\Gamma|\mathcal{D}) \propto p(\mathcal{D}|\Gamma)p(\Gamma)$$

- If $p(\Gamma) \approx const$ the optimal hyperparameter Γ^* is given by

$$\Gamma^* = \arg\max_{\Gamma} p(\mathcal{D}|\Gamma),$$

  where the **marginal likelihood**

$$p(\mathcal{D}|\Gamma) = \int p(\mathcal{D}|\Gamma, \mathbf{w})p(\mathbf{w}|\Gamma)d\mathbf{w}$$

- Selecting hyperparameters that maximize the marginal likelihood is called *ML-II* (a.k.a. *evidence maximization, empirical Bayes, maximum marginal likelihood*)

ML vs. ML-II

- In **maximum likelihood**, we select parameter values $\mathbf{w}$ that maximize the log-likelihood

$$\log p(y|\mathbf{w}, \mathbf{x}) = \sum_{i=1}^{N} \log N(y_i|\mathbf{w}^T \phi(\mathbf{x}_i), \beta^{-1})$$

$$\widehat{\mathbf{w}} = \arg\max_{\mathbf{w}} \{\log p(y|\mathbf{w}, \mathbf{x})\} \quad \text{(does not depend on } \beta)$$

- In **ML-II**, we select hyperparameter values α and β that *maximize the (log-)marginal likelihood* (parameters $\mathbf{w}$ integrated out)

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

$$\Gamma^* = \arg\max_{\Gamma} \{\log p(y|\Gamma, x)\}$$

Hyperparameter optimization in practice

- EM-algorithm
- using the gradient
- compute log-marginal likelihood over a grid of values and choose the best value
- use some standard optimization routine

Alternative to ML-II: validation data (1/2)*

- Set the hyperparameters Γ to the value that minimizes the prediction error in the validation data

$$\{\mathcal{X}_{val}, \mathcal{Y}_{val}\} = \left\{(\mathbf{x}_j^{val}, y_j^{val}), j = 1, \ldots, M\right\}.$$

- Mean squared error (MSE)

$$\text{MSE}(\Gamma) = \frac{1}{M}\sum_{j=1}^{M}(y_j^{val} - \widetilde{y}_j^{val})^2,$$

where

$$\widetilde{y}_j^{val} = \mathbf{m}^T\phi(\mathbf{x}_j^{val}), \qquad \mathbf{m} = E(\mathbf{w}|\Gamma, \mathcal{X}_{train}, \mathcal{Y}_{train})$$

Alternative to ML-II: validation data (2/2)*

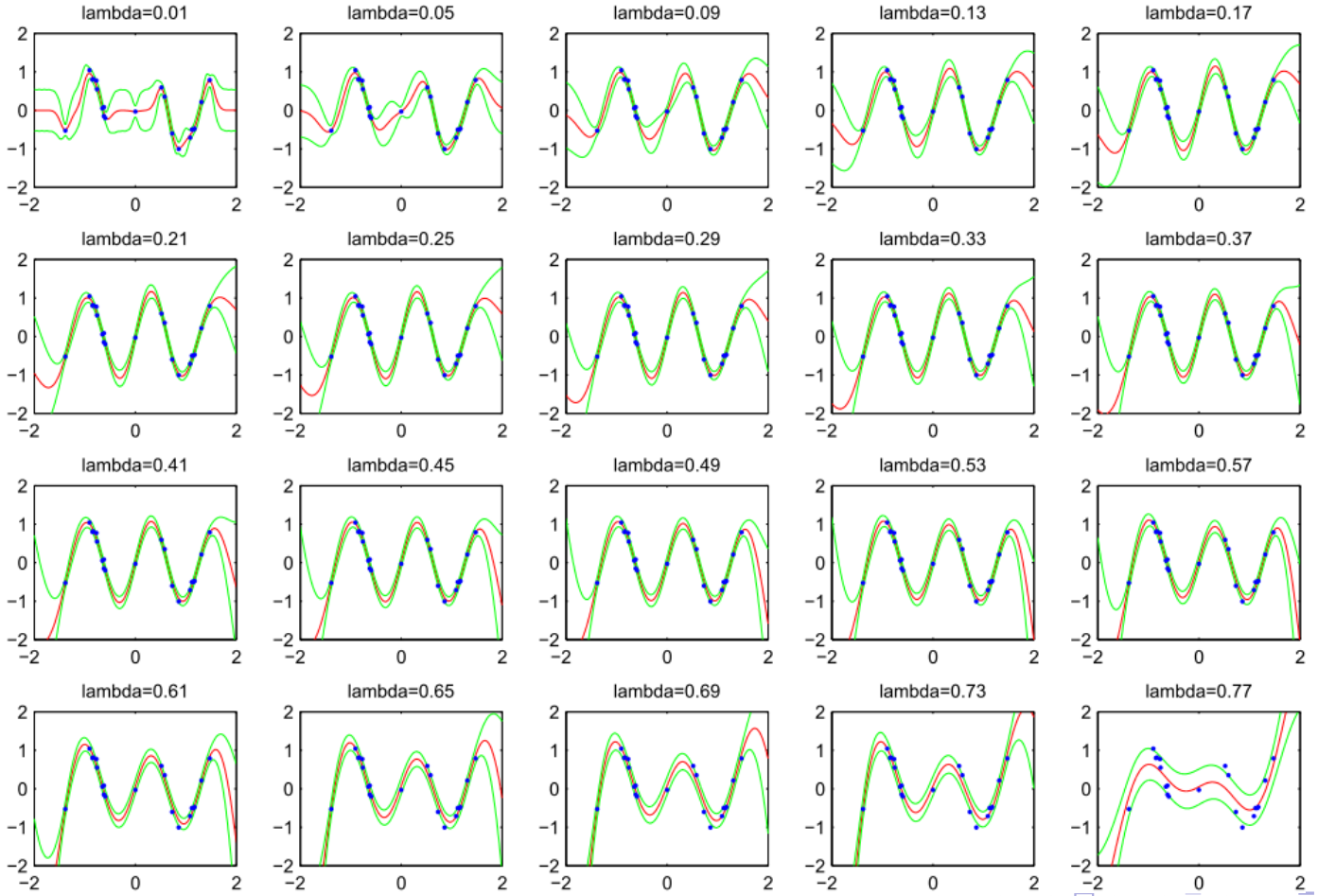
- Or by maximizing the validation data marginal likelihood

$$p(\mathcal{Y}_{val}|\Gamma, \mathcal{D}_{train}, \mathcal{X}_{val}) = \int_{\mathbf{w}} p(\mathcal{Y}_{val}|\mathbf{w}, \mathcal{X}_{val}, \Gamma) p(\mathbf{w}|\Gamma, \mathcal{X}_{train}, \mathcal{Y}_{train}) d\mathbf{w}$$

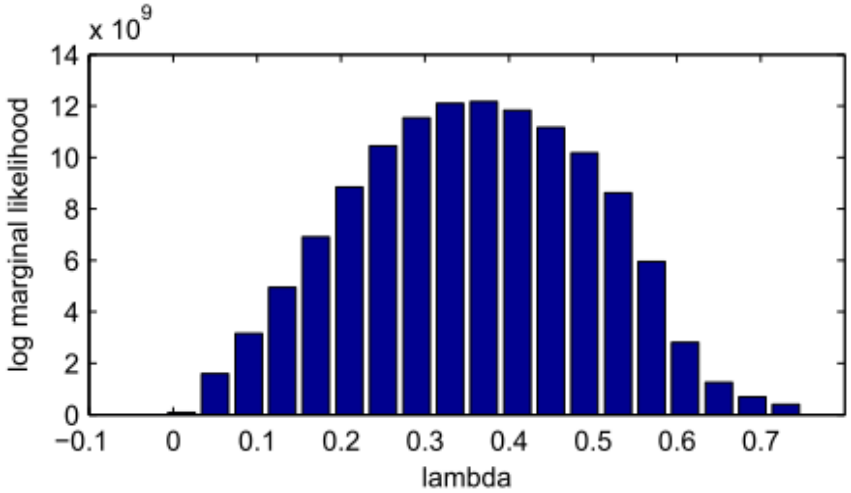
- Possible extension: *cross-validation*

Learning radial basis function width (1/2)

- A set of 10 evenly spaced radial basis functions is used
  $\phi_i(x) = \exp\left(-0.5(x - c_i)^2 / \lambda^2\right)$
- $\Gamma = (\alpha, \beta)$ optimized for different width parameters λ



Learning radial basis function width (2/2)



- The log marginal likelihood

$$\log p(\mathcal{D}|\lambda, \alpha^*(\lambda), \beta^*(\lambda))$$

  having optimized α and β using ML-II. These values depend on λ.
- The best model corresponds to $\lambda = 0.37$.

Logistic regression for classification

- Binary classification problem: $\mathcal{D} = \{(\mathbf{x}_i, c_i), i = 1, \ldots, N\}$, where the output $c \in \{0, 1\}$.
- Let p denote the probability that $p(c = 1|\mathbf{x})$
- Logistic (linear) regression

$$\log \frac{p}{1-p} = \mathbf{w}^T \mathbf{x}$$

- Or, equivalently

$$p(c = 1|\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x}),$$

where $\sigma(\cdot)$ is the so-called *logistic sigmoid*

$$\sigma(x) = \frac{e^x}{1 + e^x} = \frac{1}{1 + e^{-x}}$$

Logistic regression for classification

- When used for classification, the decision boundary is defined by $p(c = 1|\mathbf{x}) = p(c = 0|\mathbf{x}) = 0.5$. This corresponds to a hyperplane

$$\mathbf{w}^T\mathbf{x} = 0.$$

Classification rule

$$\mathbf{w}^T\mathbf{x} > 0 \rightarrow c = 1$$
$$\mathbf{w}^T\mathbf{x} < 0 \rightarrow c = 0$$

- Note: $\mathbf{x}$ can include a constant term, $\mathbf{x} = (1, x_1, \ldots, x_D)$, such that the *intercept* is automatically included

$$\mathbf{w}^T\mathbf{x} = w_0 + w_1 x_1 + \ldots + w_D x_D$$

Logistic regression, interpretation of parameters*

$$\log\left(\frac{p}{1-p}\right) = w_0 + w_1 x$$
$$\Leftrightarrow \frac{p}{1-p} = \exp(w_0 + w_1 x)$$

- Interpretation: when x increases by one unit, the **odds** $\frac{p}{1-p}$ of belonging in class 1 increases by a factor equal to e^{w_1}.
- If x is binary itself, $x \in \{0, 1\}$, then e^{w_1} is the **odds ratio** between classes $x = 1$ and $x = 0$.
  - a common term in medical literature, e.g., X='smoking', C='cancer'.

Prior for logistic regression

- Gaussian prior

$$p(\mathbf{w}|\alpha) = N_D(\mathbf{w}|\mathbf{0},\alpha^{-1}\mathbf{I}) = \alpha^{\frac{D}{2}}(2\pi)^{-\frac{D}{2}} e^{-\frac{\alpha}{2}\mathbf{w}^T\mathbf{w}}$$

  where α is the precision.

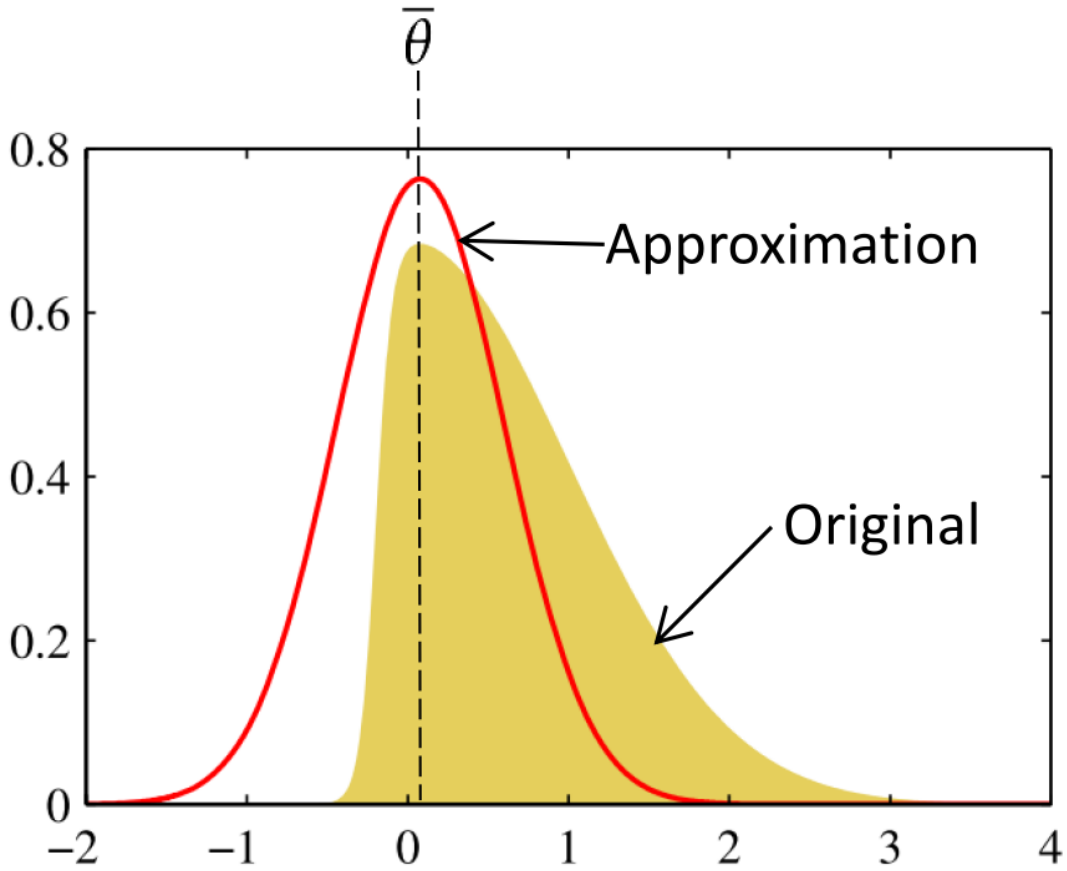
- Given $\mathcal{D} = \{(\mathbf{x}_i, c_i), i = 1, \ldots, N\}$ the posterior equals

$$p(\mathbf{w}|\alpha, \mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w}, \alpha)p(\mathbf{w}|\alpha)}{p(\mathcal{D}|\alpha)} = \frac{1}{p(\mathcal{D}|\alpha)} p(\mathbf{w}|\alpha) \prod_{i=1}^{N} p(c_i|\mathbf{x}_i, \mathbf{w})$$

  (not of standard form, Laplace approximation is feasible to compute).

Laplace approximation

- Gaussian approximation at the mode



modified from Bishop, Fig. 4.14

Laplace approximation of posterior distribution

- In general, for any posterior $p(\mathbf{w}|\alpha, \mathcal{D})$ it holds that

$$p(\mathbf{w}|\alpha, \mathcal{D}) \propto \exp(-E(\mathbf{w})), \quad E(\mathbf{w}) = -\log p(\mathbf{w}|\alpha, \mathcal{D}).$$

1. Approximate $E(\mathbf{w})$ by a 2nd order Taylor polynomial $\widetilde{E}(\mathbf{w})$ at the minimum $\overline{\mathbf{w}}$

$$\widetilde{E}(\mathbf{w}) = E(\overline{\mathbf{w}}) + \frac{1}{2}(\mathbf{w}-\overline{\mathbf{w}})^T H_{\overline{\mathbf{w}}}(\mathbf{w}-\overline{\mathbf{w}})$$

   (Note, this is quadratic in $\mathbf{w}$.)
2. Obtain a Gaussian approximation $q(\mathbf{w}|\alpha, \mathcal{D})$:

$$p(\mathbf{w}|\alpha, \mathcal{D}) \approx q(\mathbf{w}|\alpha, \mathcal{D}) \propto \exp(-\widetilde{E}(\mathbf{w}))$$

- For logistic regression,

$$E(\mathbf{w}) = \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \log \sigma(\mathbf{w}^T\mathbf{h}_i), \quad \mathbf{h}_i \equiv (2c_i - 1)\mathbf{x}_i.$$

Laplace approximation in practice

- In practice:
  - Find the minimum $\overline{\mathbf{w}}$ of $E(\mathbf{w})$ analytically (root of the derivative) or by numerical optimization, e.g. Newton's method:

$$\mathbf{w}^{new} = \mathbf{w} - \mathbf{H}_w^{-1} \nabla E$$

  - When converged, compute the Hessian $H_{\overline{\mathbf{w}}}$ of $E(\mathbf{w})$ at $\overline{\mathbf{w}}$.
  - The posterior approximation is

$$q(\mathbf{w}|\alpha, \mathcal{D}) = N(\mathbf{w}|\mathbf{m}, \mathbf{S}), \quad \mathbf{m} = \overline{\mathbf{w}}, \quad \mathbf{S} = \mathbf{H}_{\overline{\mathbf{w}}}^{-1}.$$

- Reminder: if $f \equiv f(x_1, \ldots, x_n)$

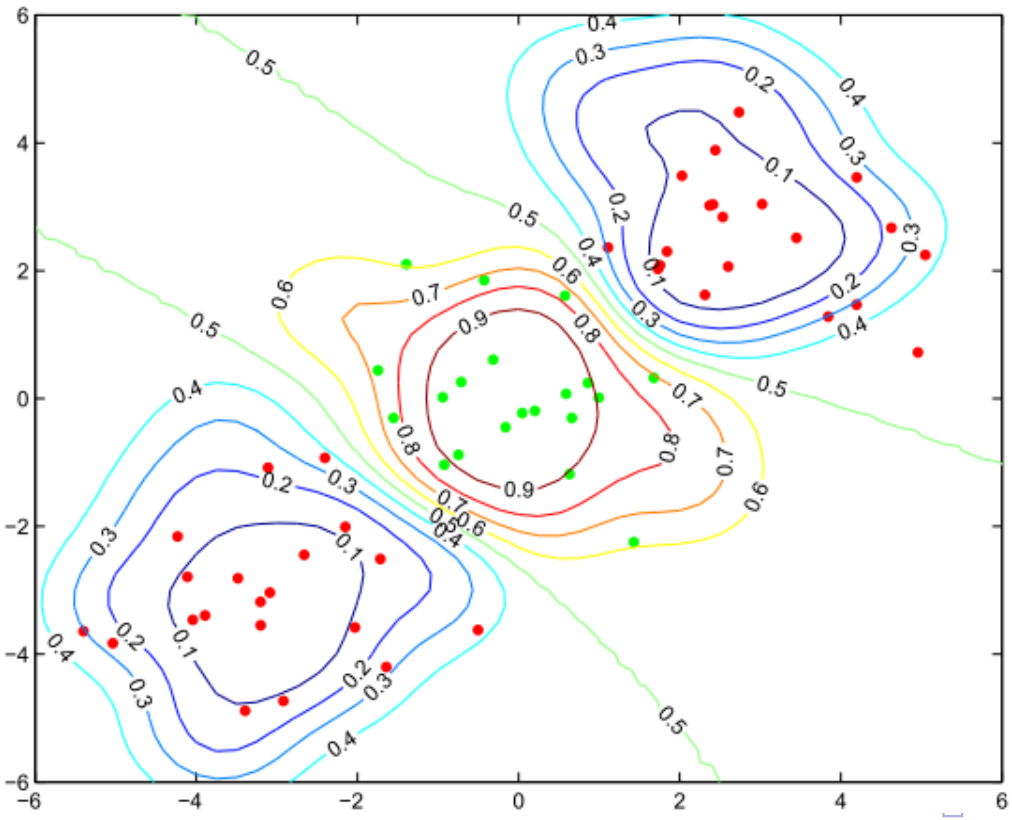
$$H_f = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

Laplace approximation for a univariate posterior distribution

- For some univariate parameter θ, you are given a prior $p(\theta)$ and the likelihood $p(\mathbf{x}|\theta)$.
- How do you calculate the Laplace approximation $q(\theta|\mathbf{x})$ of the posterior $p(\theta|\mathbf{x})$?

Laplace approximation for logistic regression

- Bayesian logistic regression with RBF functions $\phi_i(\mathbf{x}) = \exp(-\lambda(\mathbf{x} - \mathbf{m}_i)^2)$.
- $\mathbf{m}_i$ placed on a subset of training points, λ set to 2
- Hyperparameter α optimized as with the Bayesian linear regression by maximizing the approximated marginal likelihood ($\rightarrow \alpha = 0.45$).

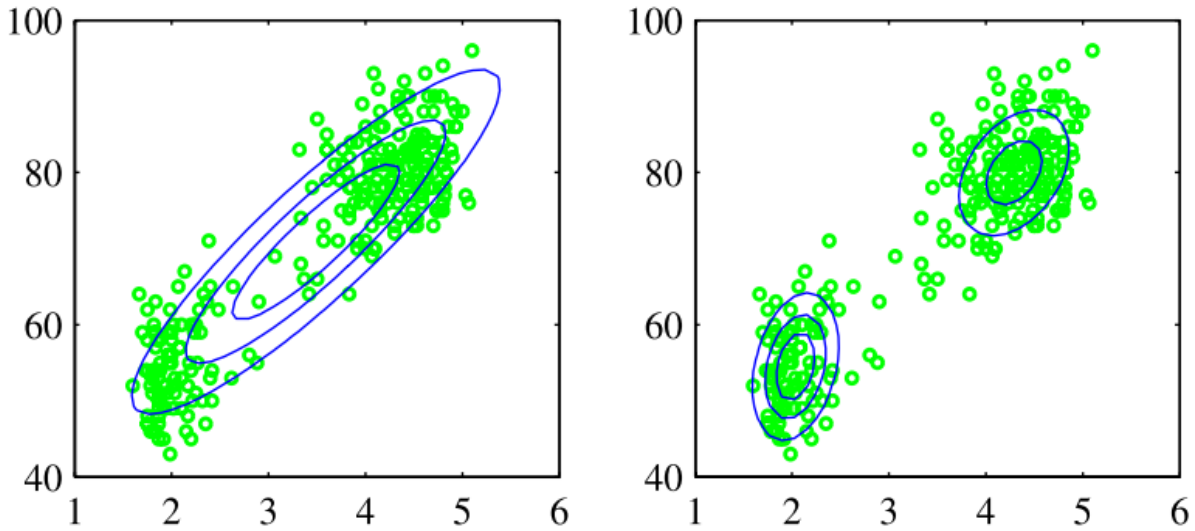


General comments on usage*

- Curse of dimensionality limits the use of RBFs to low-dimensional cases
  - Number of required basis functions grows exponentially w.r.t. the dimension D
  - Possible remedy: place basis functions on observations
  - Alternatives: kernel methods, Gaussian processes
- With sparse priors, standard linear models can be used with very large D
  - $y = \sum_{i=1}^{D} w_i x_i + \epsilon$

Gaussian mixture models (motivation)

- Standard Gaussian model (left) gives bad fit to data with clusters
- Combination of two Gaussians (right) is much better

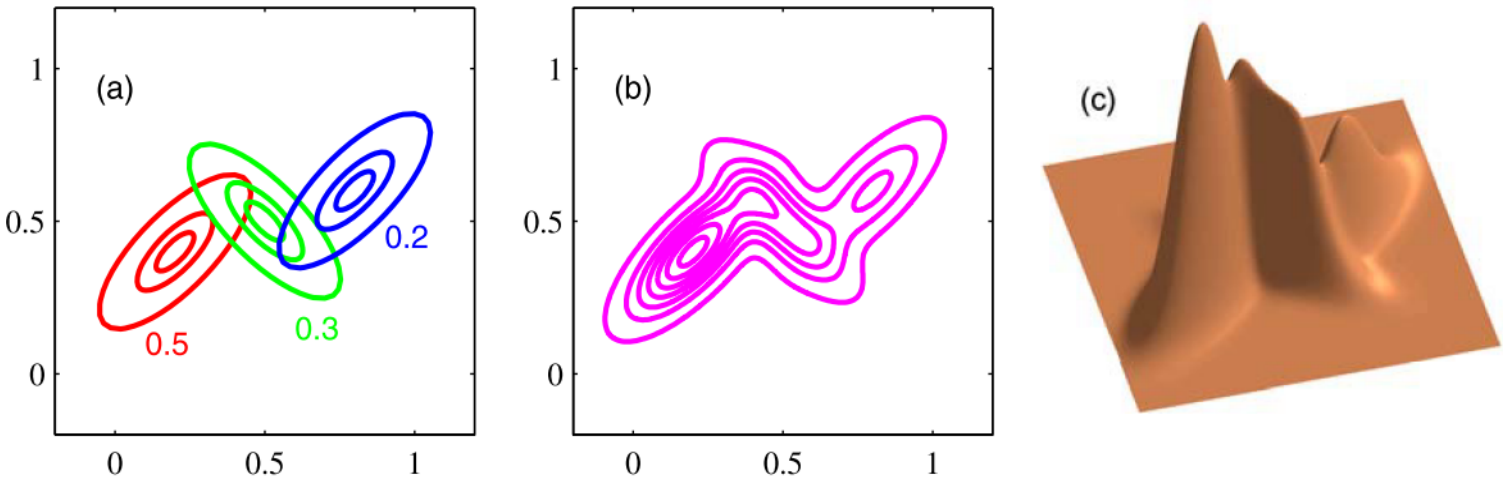


Gaussian mixture models

- Gaussian mixture model with K components has density

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k N(\mathbf{x}|\mu_k, \Sigma_k).$$

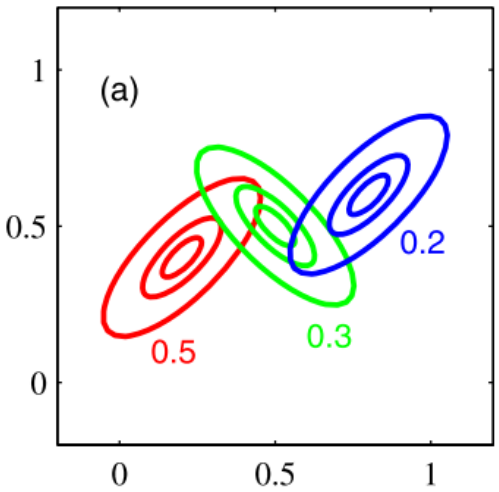
- $N(x|\mu_k, \Sigma_k)$ is a **component** with its own mean μ_k and covariance Σ_k.
- π_k are the **mixing coefficients**, which satisfy $\sum_k \pi_k = 1$, $0 \leq \pi_k \leq 1$.

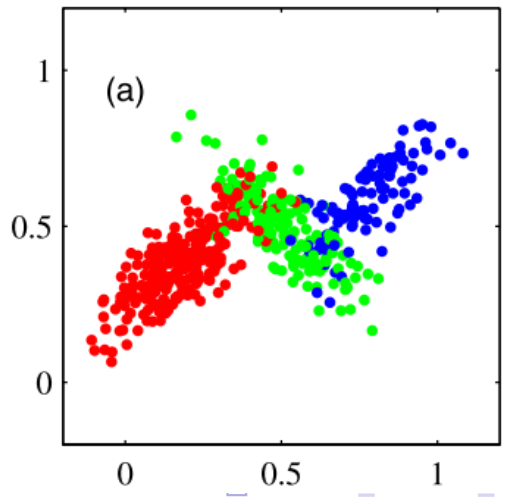


GMMs, latent variable representation (1/2)

- Equivalent formulation is obtained by defining **latent variables** $\mathbf{z}_n = (z_{n1}, \ldots, z_{nK})$ which tell the component for observation $\mathbf{x}_n$
- In detail $\mathbf{z}_n$ is a vector with exactly one element equal to 1 and other elements equal to 0. $z_{nk} = 1$ means that the observation $\mathbf{x}_n$ belongs to component k.

$$\mathbf{z}_n = (0, \ldots, 0, \underbrace{1}_{k^{th}\ \text{elem.}}, 0, \ldots, 0)^T$$





GMMs, latent variable representation (2/2)

- Define

$$p(z_{nk} = 1) = \pi_k \qquad \text{and} \qquad p(\mathbf{x}_n | z_{nk} = 1) = N(\mathbf{x}_n | \mu_k, \Sigma_k),$$

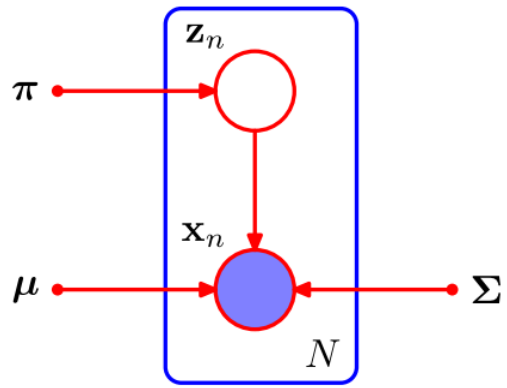
or equivalently

$$p(\mathbf{z}_n) = \prod_{k=1}^{K} \pi_k^{z_{nk}} \quad \text{and} \quad p(\mathbf{x}_n | \mathbf{z}_n) = \prod_{k=1}^{K} N(\mathbf{x}_n | \mu_k, \Sigma_k)^{z_{nk}}$$

- Then

$$p(\mathbf{x}_n) = \sum_{z_n} p(\mathbf{z}_n) p(\mathbf{x}_n | \mathbf{z}_n) = \sum_k \pi_k N(\mathbf{x}_n | \mu_k, \Sigma_k)$$

$\rightarrow \mathbf{x}_n$ has marginally the Gaussian mixture model distribution.



GMM: responsibilities (1/2)

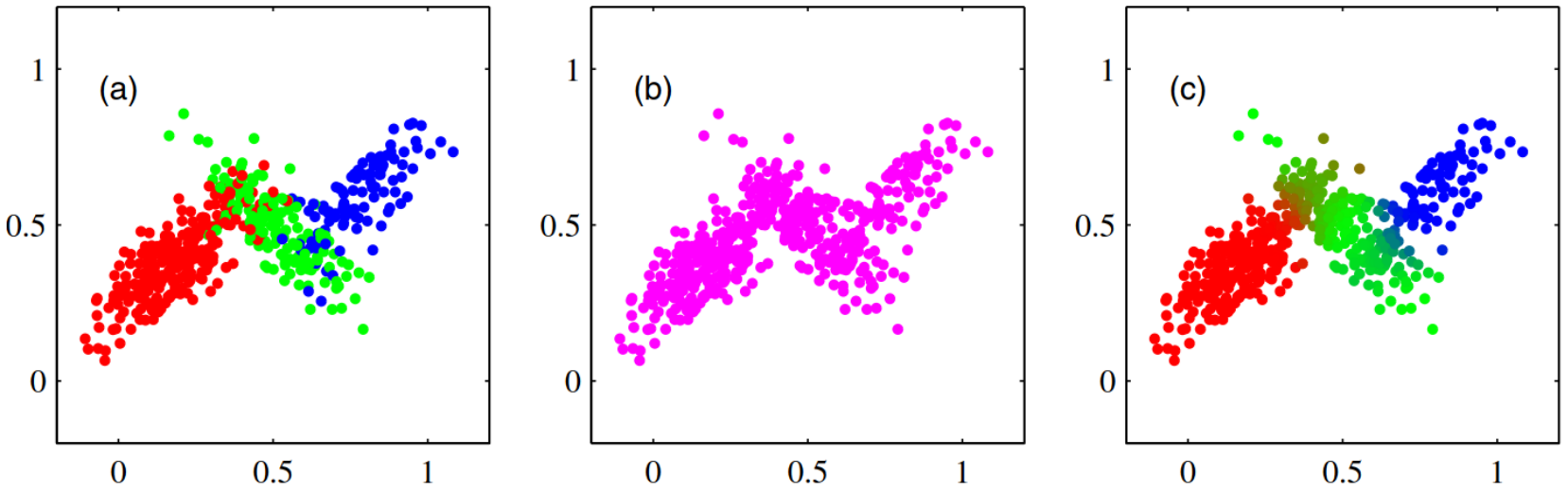
- Posterior probability $p(z_{nk} = 1|\mathbf{x}_n)$ that observation $\mathbf{x}_n$ was generated by component k

$$\gamma(z_{nk}) \equiv p(z_{nk} = 1|\mathbf{x}_n) = \frac{p(z_{nk} = 1)p(\mathbf{x}_n|z_{nk} = 1)}{\sum_{j=1}^{K} p(z_{nj} = 1)p(\mathbf{x}_n|z_{nj} = 1)}$$
$$= \frac{\pi_k N(\mathbf{x}_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(\mathbf{x}_n|\mu_j, \Sigma_j)}$$

- $\gamma(z_{nk})$ can be viewed as the **responsibility** that component k takes for explaining the observation $\mathbf{x}_n$

GMM: responsibilities (2/2)

- (left) samples from a joint distribution $p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$, showing both cluster labels $\mathbf{z}$ and observations $\mathbf{x}$ (**complete** data)
- (center) samples from the marginal distribution $p(\mathbf{x})$ (**incomplete** data)
- (right) **responsibilities** of the data points, computed using *known* parameters $\pi = (\pi_1, \ldots, \pi_K)$, $\mu = \mu_1, \ldots, \mu_K$, $\Sigma = (\Sigma_1, \ldots, \Sigma_K)$.
- Problem: in practice π, μ, and Σ are usually *unknown*.



Important points

- In classification, no closed form solution is available for logistic regression and approximations, e.g., the Laplace approximation, are needed.
- Hyperparameters can be set by maximizing the marginal likelihood (either exact or approximate).
- Definition of the Gaussian mixture model.
- Representing the GMM using discrete latent variables, which specify the components (or clusters) of the observations.

Advanced probabilistic methods

Lecture 5: Expectation maximization

Pekka Marttinen

Aalto University

February, 2021

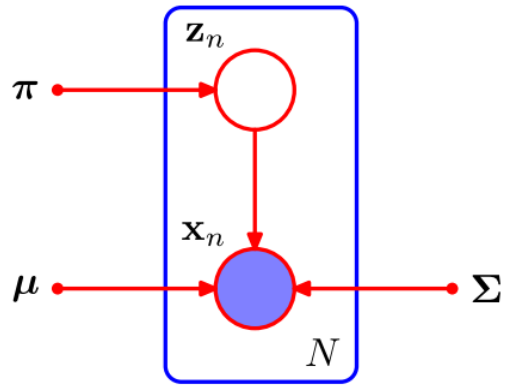
Lecture 5 overview

- Gaussian mixture models (GMMs), recap
- EM algorithm
- EM for Gaussian mixture models
- Suggested reading: Bishop: *Pattern Recognition and Machine Learning*
  - p. 110-113 (2.3.9): Mixtures of Gaussians
  - *simple_example.pdf*
  - p. 430-443: EM for Gaussian mixtures

GMMs, latent variable representation

- Introduce **latent variables** $\mathbf{z}_n = (z_{n1}, \ldots, z_{nK})$ which spcifies the component k of observation $\mathbf{x}_n$

$$\mathbf{z}_n = (0, \ldots, 0, \underbrace{1}_{k^{th}\ \text{elem.}}, 0, \ldots, 0)^T$$



- Define

$$p(\mathbf{z}_n) = \prod_{k=1}^{K} \pi_k^{z_{nk}} \quad \text{and} \quad p(\mathbf{x}_n|\mathbf{z}_n) = \prod_{k=1}^{K} N(\mathbf{x}_n|\mu_k, \Sigma_k)^{z_{nk}}$$

Then the marginal distribution $p(\mathbf{x}_n)$ is a GMM:

$$p(\mathbf{x}_n) = \sum_{k=1}^{K} \pi_k N(\mathbf{x}_n|\mu_k, \Sigma_k)$$

GMM: responsibilities, complete data

- Posterior probability (**responsibility**) $p(z_{nk} = 1|\mathbf{x}_n)$ that observation $\mathbf{x}_n$ was generated by component k

$$\gamma(z_{nk}) \equiv p(z_{nk} = 1|\mathbf{x}_n) = \frac{\pi_k N(\mathbf{x}_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(\mathbf{x}_n|\mu_j, \Sigma_j)}$$

- **Complete data**: latent variables $\mathbf{z}$ and data $\mathbf{x}$ together: $(\mathbf{x}, \mathbf{z})$

Idea of the EM algorithm (1/2)

- Let X denote the observed data, and θ model parameters. The goal in maximum likelihood is to find $\widehat{\theta}$:

$$\widehat{\theta} = \arg\max_{\theta} \{\log p(X|\theta)\}$$

- If model contains latent variables Z, the log-likelihood is given by

$$\log p(X|\theta) = \log \left\{ \sum_{Z} p(X, Z|\theta) \right\},$$

  which may be difficult to maximize analytically
- Possible solutions: 1) numerical optimization, 2) the EM algorithm (expectation-maximization)

Idea of the EM algorithm (2/2)

- X: **observed** data, Z: **unobserved** latent variables
- $\{X, Z\}$: **complete** data, X: **incomplete** data
- In EM algorithm, we assume that the complete data log-likelihood:

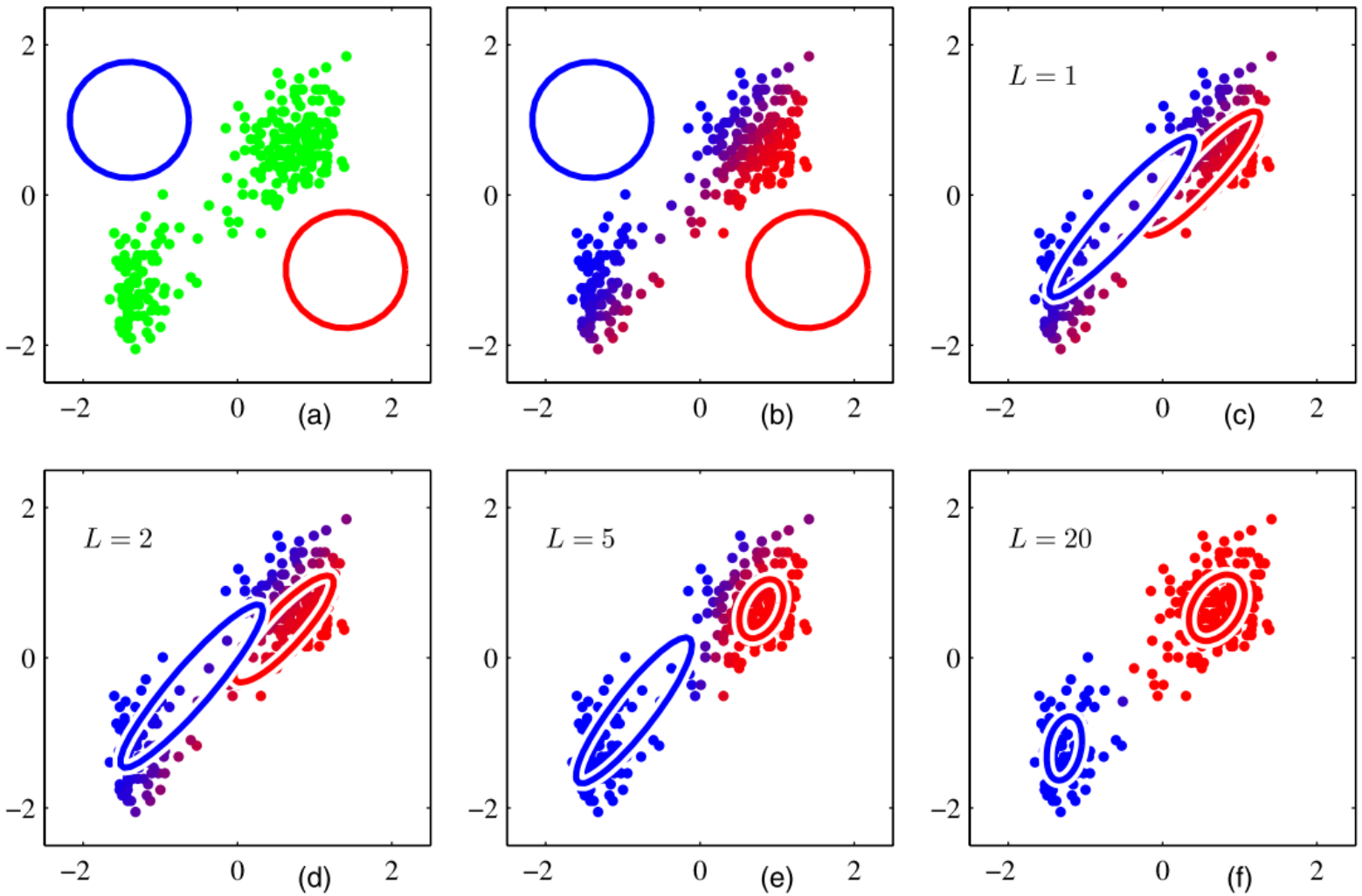
$$\log p(X, Z|\theta)$$

  is easy to maximize.
- Problem: Z is not observed
- Solution: maximize

$$\begin{aligned} Q(\theta, \theta_0) &\equiv E_{Z|X,\theta_0}\left[\log p(X, Z|\theta)\right] \\ &= \sum_Z p(Z|X, \theta_0) \log p(X, Z|\theta) \end{aligned}$$

  where $p(Z|X, \theta_0)$ is the posterior distribution of the latent variables computed using the current parameter estimate θ_0

Illustration of the EM algorithm for GMMs



EM algorithm in detail

Goal: maximize $\log p(X|\theta)$ w.r.t. θ

1. Initialize θ_0
2. **E-step** Evaluate $p(Z|X, \theta_0)$, and then compute

$$Q(\theta, \theta_0) = E_{Z|X,\theta_0}\left[\log p(X, Z|\theta)\right] = \sum_Z p(Z|X, \theta_0) \log p(X, Z|\theta)$$

3. **M-step** Evaluate θ^{new} using

$$\theta^{new} = \arg\max_\theta Q(\theta, \theta_0).$$

   Set $\theta_0 \leftarrow \theta^{new}$
4. Repeat **E** and **M** steps until convergence

Why EM works

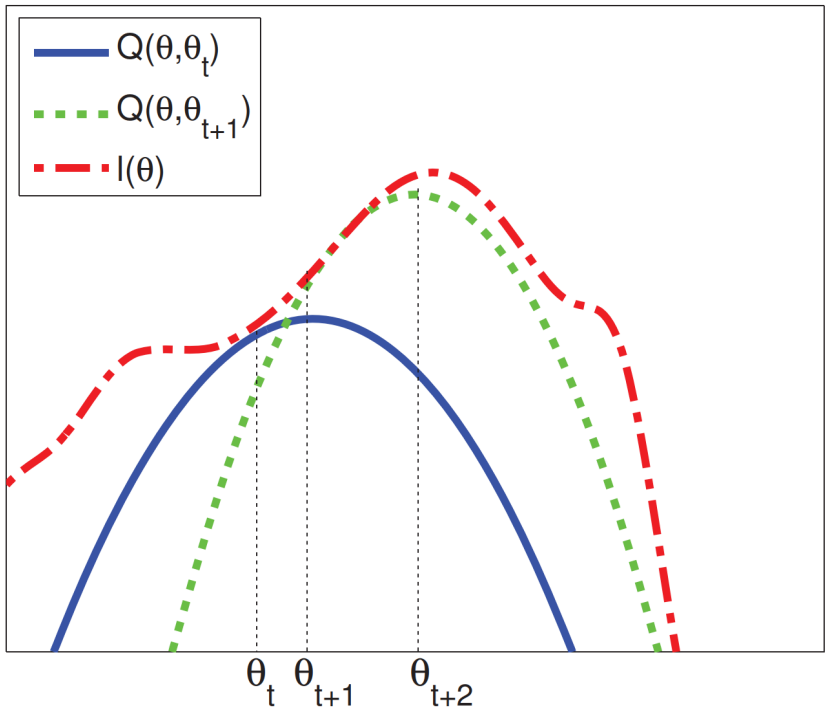


Figure: 11.16 in Murphy (2012)

- As a function of θ, $Q(\theta, \theta_0)$ is a lower bound of the log-likelihood $\log p(x|\theta)$ (plus a constant, see Bishop, Ch. 9.4).
- EM iterates between 1) updating the lower bound (E-step), 2) maximizing the lower bound (M-step).

EM algorithm, comments

- In general, Z does not have to be discrete, just replace the summation in $Q(\theta, \theta_0)$ by integration.
- EM-algorithm can be used to compute the MAP (*maximum a posteriori*) estimate by maximizing in the M-step $Q(\theta, \theta_0) + \log p(\theta)$.
- In general, EM-algorithm is applicable when the observed data X can be **augmented** into complete data $\{X, Z\}$ such that $\log p(X, Z|\theta)$ is easy to maximize; Z does not have to be latent variables but can represent, for example, unobserved values of missing or censored observations.

EM algorithm, simple example

- Consider N independent observations $\mathbf{x} = (x_1, \ldots, x_N)$ from a two-component mixture of univariate Gaussians

$$p(x_n|\theta) = \frac{1}{2}N(x_n|0,1) + \frac{1}{2}N(x_n|\theta,1). \quad (1)$$

- One unknown parameter, θ, the mean of the second component.
- **Goal**: estimate

$$\widehat{\theta} = \arg\max_{\theta} \{\log p(\mathbf{x}|\theta)\}.$$

- *simple_example.pdf*

EM algorithm for GMMs

$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k N(\mathbf{x}|\mu_k, \Sigma_k)$

1. Initialize parameter μ_k, Σ_k and mizing coefficients π_k. Repeat until convergence:
2. **E-step**: Evaluate the responsibilities using current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k N(\mathbf{x}_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_k N(\mathbf{x}_n|\mu_k, \Sigma_j)}$$

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

$$\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n$$

$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) (\mathbf{x}_n - \mu_k^{new})(\mathbf{x}_n - \mu_k^{new})^T$$

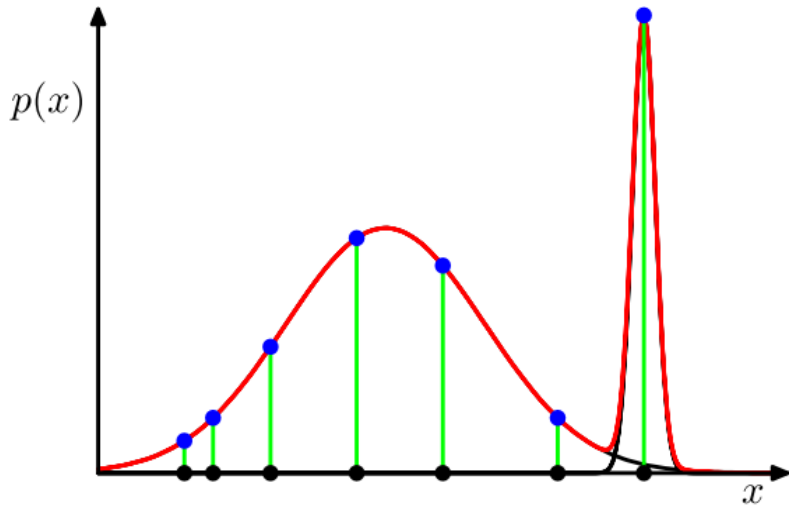
$$\pi_k^{new} = \frac{N_k}{N}$$

Derivation of the EM algorithm for GMMs

- In the **M-step** the formulas for μ_k^{new} and Σ_k^{new} are obtained by differentiating the expected complete data log-likelihood $Q(\theta, \theta_0)$ with respect to the particular parameters, and setting the derivatives to zero.
- The formula for π_k^{new} can be derived by maximizing $Q(\theta, \theta_0)$ under the constraint $\sum_{k=}^{K} \pi_k = 1$. This can be done using the *Lagrange multipliers*.

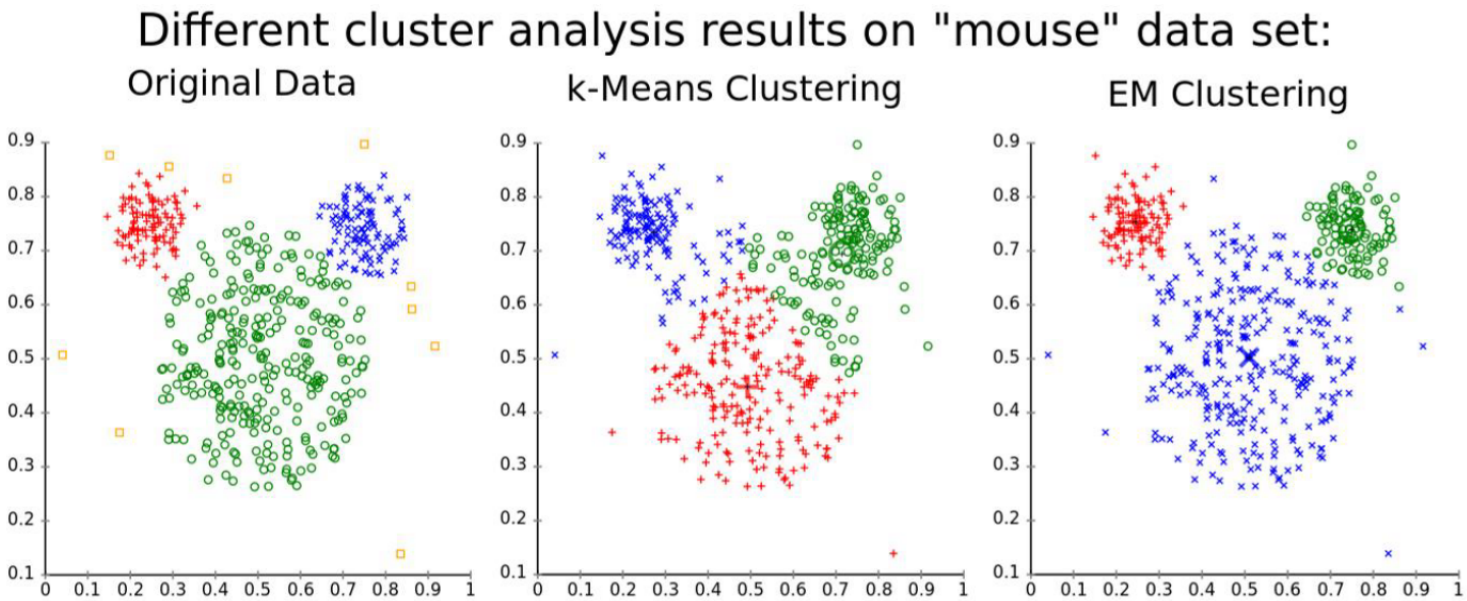
EM for GMM, caveats

- EM converges to a local optimum. In fact, the ML estimation for GMMs is not well-defined due to **singularities**: if $\sigma_k \longrightarrow 0$ for components k with a single data point, likelihood goes to infinity (fig). Remedy: prior on σ_k.
- **Label-switching**: non-identifiability due to the fact that cluster labels can be switched and likelihood remains the same.
- In practice it is recommended to initialize the EM for the GMM by k-means.



GMM vs. k-means

- "Why use GMMs and not just k-means?"



from Wikipedia

1. Clusters can be of different sizes and shapes
2. Probabilistic assignment of data items to clusters
3. Possibility to include prior knowledge (structure of the model/prior distributions on the parameters)

Important points

- ML-estimation of GMMs can be done using numerical optimization or the EM algorithm.
- The main idea of the EM algorithm is to maximize the expectation of the complete data log-likelihood, where the expectation is computed with respect to the current posterior distributions (responsibilites) of the latent variables.

Advanced probabilistic methods

Lecture 9: Variational Bayes by backpropagation

Pekka Marttinen

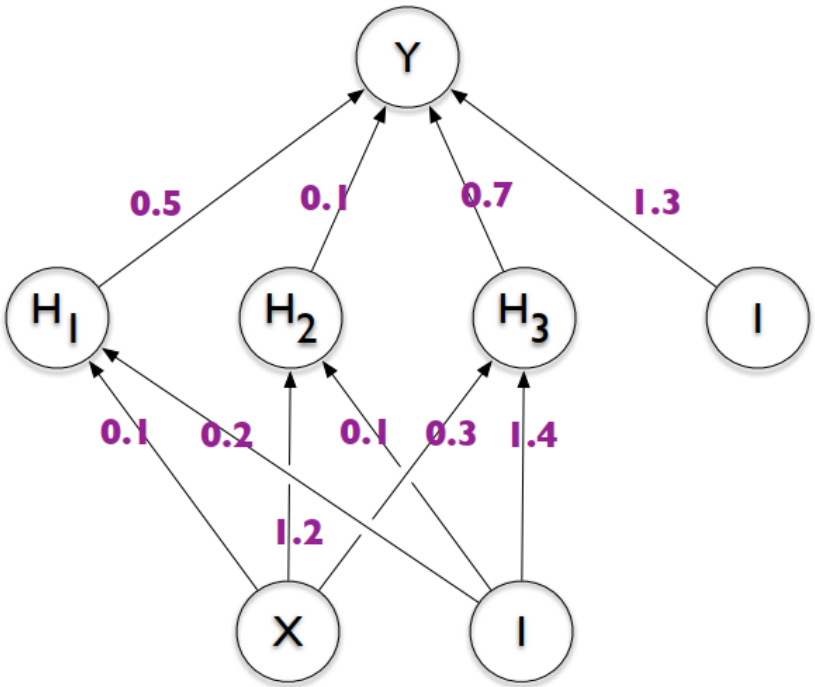
Aalto University

March, 2021

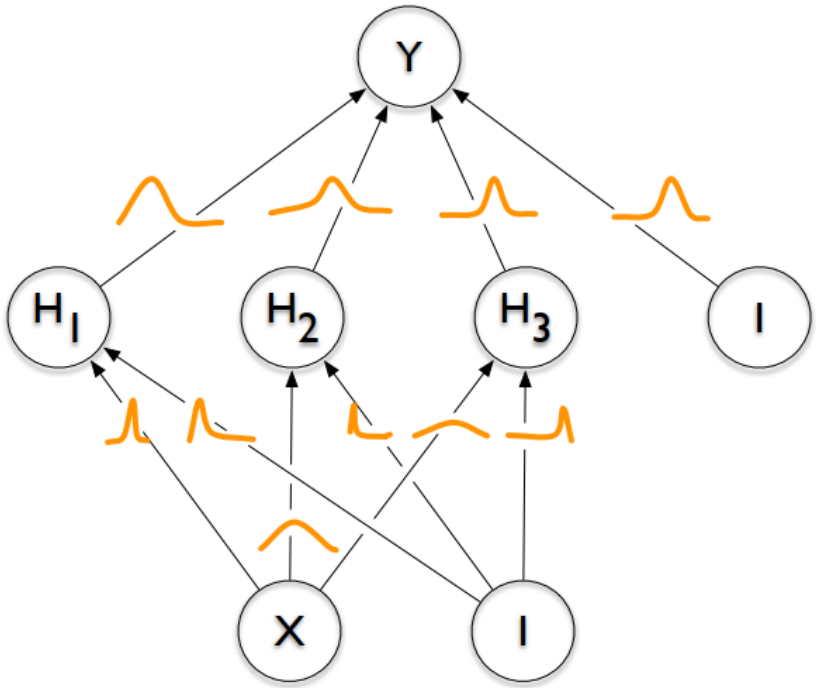
Lecture 9 overview

- Recap and caveats of VB
- Idea of variational Bayes by backpropagation
- Gradient of the ELBO
  - Monte Carlo sampling and backpropagation
- Computation using a mini-batch
- Lecture based on:
  - Blundell et al. (2015). Weight uncertainty in neural networks. *ICML*. https://arxiv.org/pdf/1505.05424.pdf
- Also relevant, for example:
  - Hoffman et al. (2013). Stochastic Variational Inference.
  - Kingma, Welling (2014). Auto-encoding variational Bayes.
  - Wilson, Izmailov (2020). Bayesian Deep Learning and a Probabilistic Perspective of Generalization

Motivation: Bayesian Neural Networks (BNNs)

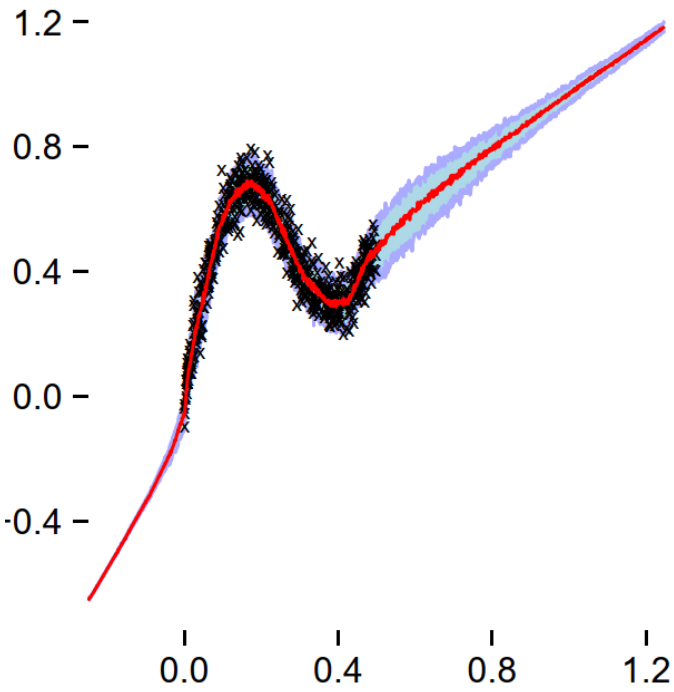


Classical neural network: each weight has a fixed value. (Blundell, Fig. 1)

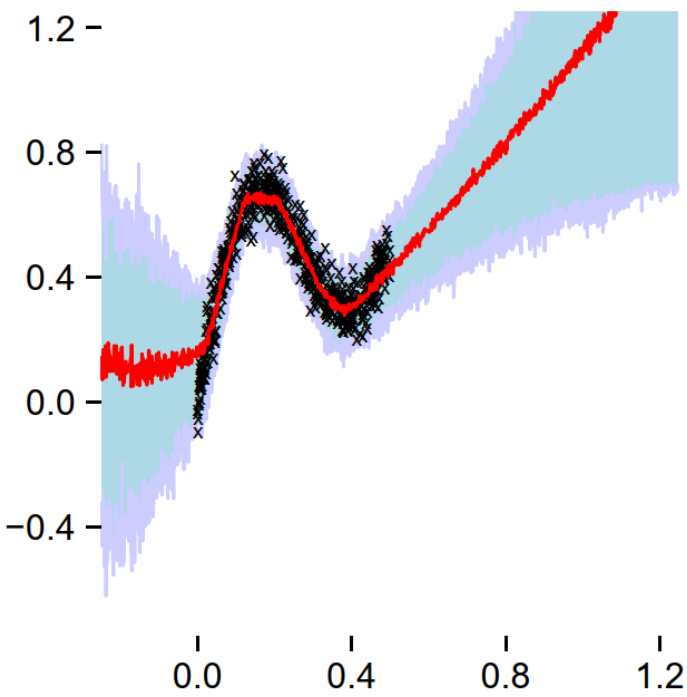


Bayesian neural network: each weight is assigned a probability distribution.

Benefits of being Bayesian (1/2)



Classical NN severely underestimates uncertainty in out-of-data regions.



Bayesian NN captures uncertainty better. (Blundell, Fig. 5)

Benefits of being Bayesian (2/2)

- Uncertainty properly quantified
  - Important in decision making
  - Critical in: medical applications, autonomous driving, ...
  - Active learning, reinforcement learning, ...
- Improved generalization (prediction accuracy)
  - Cheap model averaging over the posterior uncertainty
  - Automated complexity cost: regularization, robustness to small perturbations

Notation

- Model:

$$y_i = f(x_i, \mathbf{w}) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma_l^2), \quad i = 1, \ldots, N.$$

- The log-likelihood:

$$\log p(\mathcal{D}|\mathbf{w}) = \sum_{i=1}^{N} \log p(y_i|x_i, \mathbf{w}) = \sum_{i=1}^{N} \log \mathcal{N}(y_i|f(x_i, \mathbf{w}), \sigma_l^2)$$

- Prior: $\mathbf{w} \sim \mathcal{N}(0, \alpha^2 I)$
- Hyperparameters α^2 and σ_l^2 are assumed known constants.
- f can be a NN or linear regression (exercise)

Predictive uncertainty

- Classical NN:

$$p(y^*|x^*, \mathcal{D}) = \mathcal{N}(y^*|f(x^*, \mathbf{w}^{\text{MLE}}), \sigma_I^2), \text{ where}$$
$$\mathbf{w}^{\text{MLE}} = \arg\max_{\mathbf{w}} \log p(\mathcal{D}|\mathbf{w}).$$

- Bayesian NN:

$$p(y^*|x^*, \mathcal{D}) = \int_{\mathbf{w}} p(y^*|x^*, \mathbf{w}) p(\mathbf{w}|\mathcal{D}) d\mathbf{w}$$
$$= \int_{\mathbf{w}} \mathcal{N}(y^*|f(x^*, \mathbf{w}), \sigma_I^2) p(\mathbf{w}|\mathcal{D}) d\mathbf{w}$$

- Both models include noise uncertainty σ_I^2, but only the BNN accounts for the uncertainty in $\mathbf{w}$.

Classical way of training neural networks

- ML-estimate

$$\begin{aligned}\mathbf{w}^{\text{MLE}} &= \arg\max_{\mathbf{w}} \log p(\mathcal{D}|\mathbf{w}) \\ &= \arg\min_{\mathbf{w}} \underbrace{-\log p(\mathcal{D}|\mathbf{w})}_{\text{Loss (MSE)}}\end{aligned}$$

- (Stochastic) gradient descent:
  - Calculate loss (for a mini-batch m): $-\log p(D_m|\mathbf{w})$
  - Backpropagate to get the gradient: $-\nabla_{\mathbf{w}} \log p(D_m|\mathbf{w})$
  - Update $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} \log p(D_m|\mathbf{w})$
  - Repeat
- Very simple compared to the lengthy VB derivations!

Simple example: VB for linear regression

- Set $f(x, \mathbf{w}) = w_1 x + w_0$ such that

$$y_i = w_0 + w_1 x_i + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma_I^2), \quad i = 1, \ldots, N,$$

  where $\mathbf{w} = (w_0, w_1)$.
- Mean field assumption:

$$p(w_0, w_1 | \mathcal{D}) \approx q(w_0) q(w_1),$$

  where

$$q(w_0) = \mathcal{N}(w_0 | \mu_0, \sigma_0^2),$$
$$q(w_1) = \mathcal{N}(w_1 | \mu_1, \sigma_1^2).$$

- Parameters $\lambda = \{\mu_0, \sigma_0, \mu_1, \sigma_1\}$ are the **variational parameters**.

VB for linear regression

- The goal of VB is to learn the values of $\lambda = \{\mu_0, \sigma_0, \mu_1, \sigma_1\}$.
- Previously, we derived factor updates using formulas:

$$\log q^*(w_0) = \mathbb{E}_{q(w_1)}\left[\log p(\mathbf{x}, \mathbf{y}, w_0, w_1)\right] + \text{const.}$$
$$\log q^*(w_1) = \mathbb{E}_{q(w_0)}\left[\log p(\mathbf{x}, \mathbf{y}, w_0, w_1)\right] + \text{const.}$$

- And: exponentiate, normalize, figure out the values of the respective variational parameters.

Problems in VB

- **Problem 1**: Closed form update for:

$$\log q^*(w_0) = \mathbb{E}_{q(w_1)}\left[\log p(\mathbf{x}, \mathbf{y}, w_0, w_1)\right]$$

  available only when conjugate priors are assumed.
- **Problem 2**: Computing a single update slow when N large:

$$\log q^*(w_0) = \mathbb{E}_{q(w_1)} \underbrace{\left[\sum_{i=1}^{N} \log p(y_i|x_i, w_0, w_1)\right]}_{O(N)} + \log p(w_0).$$

- **Problem 3**: Lengthy model-specific derivations needed $\longrightarrow$ developing models slow.

VB by backpropagation, ideas

- **Idea 1**: Use Monte Carlo integration to calculate the required expectations.
  - No need for conjugate priors.
- **Idea 2**: Calculate updates using a minibatch.
  - Speed-up when N large.
- **Idea 3**: Use SGD and backpropagation to calculate the gradient of the ELBO
  - Avoids lengthy manual model-specific derivations.

Terminology

- Many methods have been introduced for VB which use SGD to optimize the ELBO.
  - Stochastic variational inference
  - Black-box variational inference
  - Stochastic gradient variational Bayes
  - Doubly-stochastic variational inference
  - Bayes by backprop
- Details of these methods may differ.
- The method presented here is called *Bayes by backprop* in Blundell *et al.* (2015).

A closer look at the variational objective (1/2)

- The ELBO for the linear regression model:

$$\mathcal{L}(\lambda) = \int q(\mathbf{w}|\lambda) \log \frac{p(\mathbf{x}, \mathbf{y}, \mathbf{w})}{q(\mathbf{w}|\lambda)} d\mathbf{w},$$

  which can be written as

$$\mathcal{L}(\lambda) = \mathbb{E}_{q(\mathbf{w}|\lambda)} \left[\log p(\mathbf{y}|\mathbf{x}, \mathbf{w})\right] - KL(q(\mathbf{w}|\lambda)||p(\mathbf{w})) + \text{const}$$

A closer look at the variational objective (2/2)

- Instead of maximizing the ELBO, in SGD we minimize the negative ELBO:

$$\text{Loss}(\lambda) = -\mathcal{L}(\lambda) = \underbrace{\mathbb{E}_{q(\mathbf{w}|\lambda)}\left[-\log p(\mathbf{y}|\mathbf{x},\mathbf{w})\right]}_{\text{Likelihood cost}} + \underbrace{KL(q(\mathbf{w}|\lambda)||p(\mathbf{w}))}_{\text{Complexity cost}}$$

- Gradient of the loss:

$$\nabla_\lambda \text{Loss}(\lambda) = \nabla_\lambda \mathbb{E}_{q(\mathbf{w}|\lambda)}\left[-\log p(\mathbf{y}|\mathbf{x},\mathbf{w})\right] + \nabla_\lambda KL(q(\mathbf{w}|\lambda)||p(\mathbf{w}))$$

- In general both the loss and its gradient are intractable.

Gradient of the complexity cost

- In the special case considered here (q factorized, distributions Gaussian), $KL(q(\mathbf{w}|\lambda)||p(\mathbf{w}))$ has a closed form.
  - Can be relaxed (details skipped).
- Hence, $KL(q(\mathbf{w}|\lambda)||p(\mathbf{w}))$ is a deterministic function of λ and can be computed in a forward pass.
- The gradient $\nabla_\lambda KL(q(\mathbf{w}|\lambda)||p(\mathbf{w}))$ can be calculated simply by using backpropagation with the chain rule.

Gradient of the likelihood cost (1/3)

- In principle, any expectation w.r.t. $q(\mathbf{w}|\lambda)$ could be approximated using Monte Carlo sampling, e.g.,

$$\mathbb{E}_{q(\mathbf{w}|\lambda)}\left[-\log p(\mathbf{y}|\mathbf{x},\mathbf{w})\right] \approx -\frac{1}{S}\sum_{s=1}^{S}\log p(\mathbf{y}|\mathbf{x},\mathbf{w}^{(s)}),$$

  where $\mathbf{w}^{(s)} \sim q(\mathbf{w}|\lambda)$.

- However, this can't be applied to compute the gradient because:

$$\begin{aligned}\nabla_\lambda \mathbb{E}_{q(\mathbf{w}|\lambda)}\left[-\log p(\mathbf{y}|\mathbf{x},\mathbf{w})\right] &= -\nabla_\lambda \int q(\mathbf{w}|\lambda)\log p(\mathbf{y}|\mathbf{x},\mathbf{w})d\mathbf{w} \\ &\overset{1}{=} -\int \log p(\mathbf{y}|\mathbf{x},\mathbf{w})\nabla_\lambda q(\mathbf{w}|\lambda)d\mathbf{w}\end{aligned}$$

  is not an expectation w.r.t. $q(\mathbf{w}|\lambda)$.

[1] Exchanging gradient and integration is ok if the variable w.r.t. which we integrate is different from the variable w.r.t. which we differentiate (assuming regularity conditions).

Gradient of the likelihood cost (2/3)

- Reparameterization trick: instead of sampling $\mathbf{w}^{(s)} \sim q(\mathbf{w}|\lambda)$ directly, do as follows:
  1. Sample $\mathbf{e}^{(s)} \sim N(0, I)$.
  2. Transform $\mathbf{w}^{(s)} = g_\lambda(\mathbf{e})$.
- To sample from $w_i \sim q(w_i|\lambda_i) = N(w_i|\mu_i, \sigma_i^2)$, where $\lambda_i = (\mu_i, \sigma_i)$, we need to select

$$g_{\lambda_i}(e^{(s)}) = \mu_i + e^{(s)}\sigma_i.$$

- Then, if $e^{(s)} \sim N(0, 1)$, $w_i^{(s)}$ has the correct distribution:

$$w_i^{(s)} = g_{\lambda_i}(e^{(s)}) = \mu_i + e^{(s)}\sigma_i \sim N(\mu_i, \sigma_i^2).$$

Gradient of the likelihood cost (3/3)

- After reparameterization, the gradient can be approximated using Monte Carlo sampling:

$$\begin{aligned}\nabla_\lambda \mathbb{E}_{q(\mathbf{w}|\lambda)}\left[-\log p(\mathbf{y}|\mathbf{x},\mathbf{w})\right] &= \nabla_\lambda \mathbb{E}_{q(\mathbf{e})}\left[-\log p(\mathbf{y}|\mathbf{x},g_\lambda(\mathbf{e}))\right] \\ &= -\mathbb{E}_{q(\mathbf{e})}\left[\nabla_\lambda \log p(\mathbf{y}|\mathbf{x},g_\lambda(\mathbf{e}))\right] \\ &\approx -\frac{1}{S}\sum_{s=1}^{S} \nabla_\lambda \log p(\mathbf{y}|\mathbf{x},g_\lambda(\mathbf{e}^{(s)})),\end{aligned}$$

  where $\mathbf{e}^{(s)} \sim N(0, I)$, $s = 1, \ldots, S$.
- In practice it's common to use $S = 1$.
- The gradient $\nabla_\lambda \log p(\mathbf{y}|\mathbf{x},g_\lambda(\mathbf{e}^{(s)}))$ can be obtained by backpropagation.

Using minibatches

- Suppose data $\mathcal{D}$ is divided into M minibatches: $\mathcal{D}_1, \ldots, \mathcal{D}_M$.
- Objective with the full data:

$$-\mathcal{L}(\lambda) = \mathbb{E}_{q(\mathbf{w}|\lambda)}\left[-\log p(\mathbf{y}|\mathbf{x}, \mathbf{w})\right] + KL(q(\mathbf{w}|\lambda)||p(\mathbf{w}))$$

- Objective for a mini-batch:

$$-\mathcal{L}_m(\lambda) = \mathbb{E}_{q(\mathbf{w}|\lambda)}\left[-\log p(\mathbf{y}_m|\mathbf{x}_m, \mathbf{w})\right] + \frac{1}{M} KL(q(\mathbf{w}|\lambda)||p(\mathbf{w}))$$

- Or, averaged per individual:

$$-\mathcal{L}_m(\lambda) = -\frac{1}{|\mathcal{D}_m|}\mathbb{E}_{q(\mathbf{w}|\lambda)}\left[\sum_{i\in\mathcal{D}_m} \log p(y_i|x_i, \mathbf{w})\right] + \frac{1}{N} KL(q||p)$$

- Scaling the two terms to correspond to the same number of individuals ensures that the expectation of the stochastic gradient for the mini-batch is aligned with the gradient of the full cost.

Putting it all together

- One iteration of the *Bayes-by-backprop* for linear regression and mini-batch $\mathcal{D}_m$
  1. Sample $\mathbf{e}^{(s)} \sim N(0, I)$
  2. Transform $w_i^{(s)} = \mu_i + e_i^{(s)}\sigma_i$ for $i = 0, 1$, where $\lambda = (\mu_0, \sigma_0, \mu_1, \sigma_1)$[1]
  3. Forward pass to calculate the noisy objective:

$$\text{Loss}(\lambda) = -\frac{1}{|\mathcal{D}_m|}\sum_{i\in\mathcal{D}_m} \log p(y_i|x_i, \mathbf{w}^{(s)}) + \frac{1}{N} KL(q(\mathbf{w}|\lambda)||p(\mathbf{w}))$$

  4. Backward pass to get the stochastic gradient: $\nabla_\lambda \text{Loss}(\lambda)$.
  5. Update the variational parameters

$$\lambda \leftarrow \lambda - \eta \nabla_\lambda \text{Loss}(\lambda).$$

[1] The exercise uses a slightly different parameterization to ensure std stays positive.

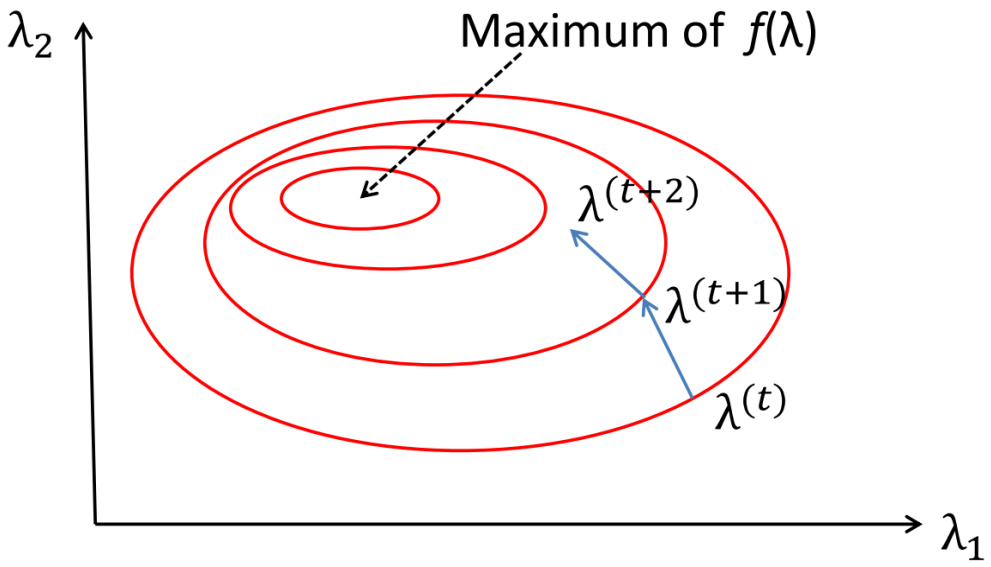
Summary

- Mean-field VB can be seen as an optimization problem: the variational parameters for each factor are updated in turn to maximize the ELBO $\mathcal{L}(q)$.
- In stochastic variational inference the negative ELBO is minimized directly using SGD.
- Stochastic gradient of the ELBO is obtained by
  - Monte Carlo sampling to approximate the loss during the forward pass.
  - Samples from $\mathbf{w}^{(s)} \sim q(\mathbf{w}|\lambda)$ are obtained using the reparameterization.
  - Backpropagation to calculate the gradient.
- Scaling up to massive data sets can be achieved using a mini-batch.

Reminder: gradient ascent algorithm*

- Gradient ascent algorithm maximizes a given function f by taking steps of length ρ to the direction of the gradient ∇f.

$$\lambda^{(t+1)} = \lambda^{(t)} + \rho \nabla_\lambda f(\lambda^{(t)}), \text{ where } \nabla_\lambda f = \left(\frac{\partial f}{\partial \lambda_1}, \ldots, \frac{\partial f}{\partial \lambda_D} \right)$$



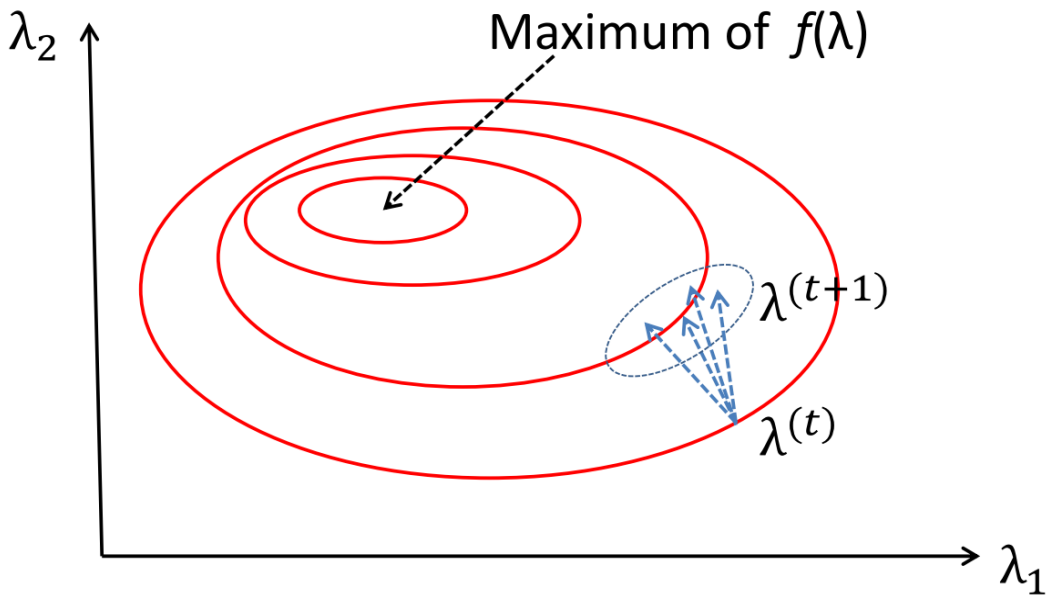
- $\lambda^{(t+1)} = \lambda^{(t)} - \rho \nabla_\lambda f(\lambda^{(t)})$ gives gradient descent.

Reminder: stochastic gradient ascent*

- Stochastic gradient ascent takes **random steps**, that are **on average to the correct direction**:

$$\lambda^{(t+1)} = \lambda^{(t)} + \rho b_t(\lambda^{(t)}),$$

- $b_t(\lambda)$ is a random variable s.t. $E(b_t(\lambda)) = \nabla_\lambda f(\lambda)$.



cartoonstock.com

Reminder: SGA with a mini-batch*

- To find a maximum likelihood estimate $\widehat{\lambda}$,

$$f(\lambda) = \frac{1}{N}\sum_{n=1}^{N} \log p(x_n|\lambda), \text{ and } \nabla_\lambda f(\lambda) = \frac{1}{N}\sum_{n=1}^{N} \nabla_\lambda \log p(x_n|\lambda)$$

  and we have to differentiate $\log p(x_n|\lambda)$ for all n.
- It is cheaper to sample a **minibatch** of S data points x_s and compute a noisy gradient

$$b(\lambda) = \frac{1}{S}\sum\nolimits_s \nabla_\lambda \log p(x_s|\lambda),$$

  which points approximately to the direction of $\nabla_\lambda f(\lambda)$.

Advanced probabilistic methods

Lecture 6: Variational inference

Pekka Marttinen

Aalto University

March, 2021

Lecture 6 overview

- Variational inference overview
- KL-divergence
- Mean-field variational inference
- Simple example using variational inference
- Suggested reading: Bishop: *Pattern Recognition and Machine Learning*
  - p. 461-474
  - *simple_vb_example.pdf* for the derivation of the VB updates for a simple GMM.
  - The general VB formulation for GMMs p. 474-486 (optional)

Approximate inference

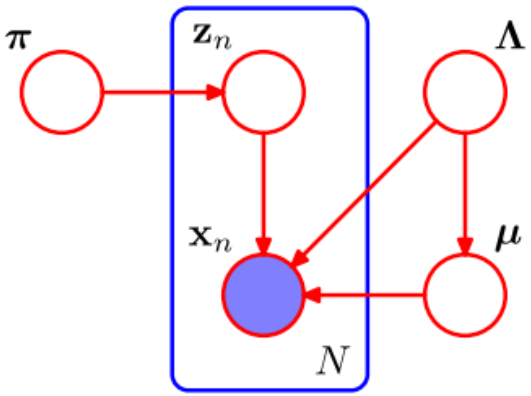
- A central task in probabilistic modeling is to evaluate the posterior distribution

$$p(Z|X)$$

of latent variables Z given the observed variables X.

- In a fully Bayesian model, model parameters θ may be given priors and included as part of Z (unlike in the EM).

  - Often, computation of $p(Z|X)$ may not be possible in a closed form, and approximations are needed
    - variational inference (today)
    - stochastic variational inference (later)
    - sampling ($\rightarrow$Bayesian data analysis)



Variational inference

- **Idea:** Approximate the posterior distribution of unknowns $p(Z|X)$ with a tractable distribution $q(Z)$.
- For example, $q(Z)$ may be assumed to have a simple form, e.g., Gaussian, or to factorize in a certain way.
- For the GMM, it would be sufficient to assume

$$q(\mathbf{z}, \pi, \Lambda, \mu) = q(\mathbf{z})q(\pi, \Lambda, \mu)$$

Basis of variational inference

- When $q(\mathbf{z})$ is an approximation for $p(\mathbf{z}|\mathbf{x})$, it is always true that

$$\log p(\mathbf{x}) = \mathcal{L}(q) + KL(q||p),$$

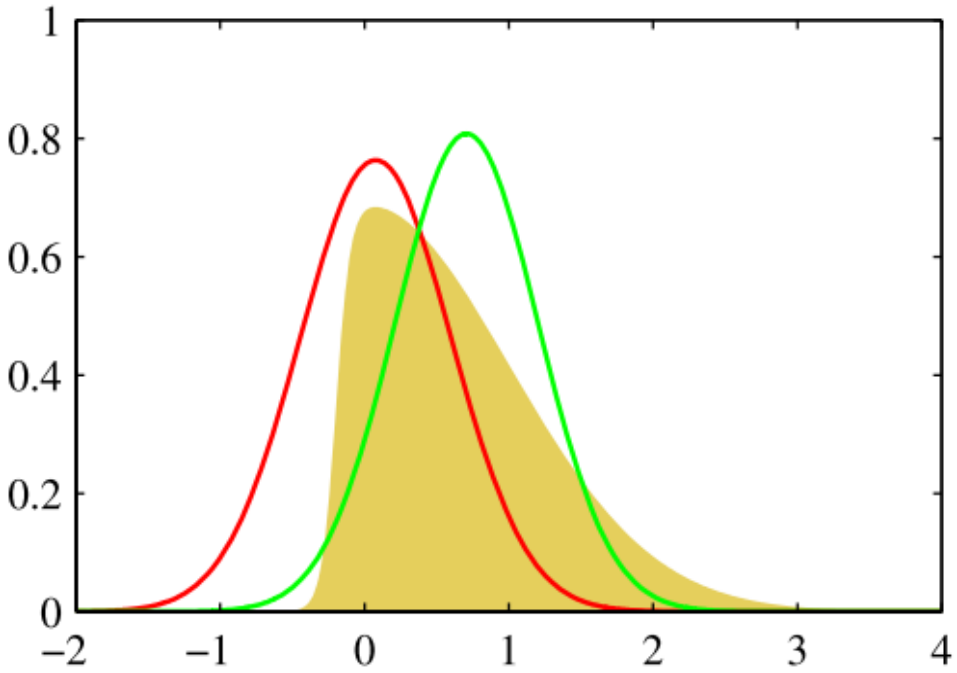
where

$$\mathcal{L}(q) = \int q(\mathbf{z}) \log \left\{ \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})} \right\} d\mathbf{z} \quad \text{(lower bound for } \log p(\mathbf{x})\text{)}$$

$$KL(q||p) = -\int q(\mathbf{z}) \log \left\{ \frac{p(\mathbf{z}|\mathbf{x})}{q(\mathbf{z})} \right\} d\mathbf{z} \quad \text{(KL-divergence btw } q \text{ and } p\text{)}.$$

- **Goal**: to maximize $\mathcal{L}(q)$ or, equivalently, to minimize the $KL(q||p)$.
- Note: $\mathcal{L}(q)$ is also called the 'ELBO' (evidence lower bound)

Variational Gaussian approximation



- Figure shows approximation of the original distribution (yellow) with a Gaussian at the mode (red, Laplace) or with a Gaussian that minimizes the KL-divergence (green).

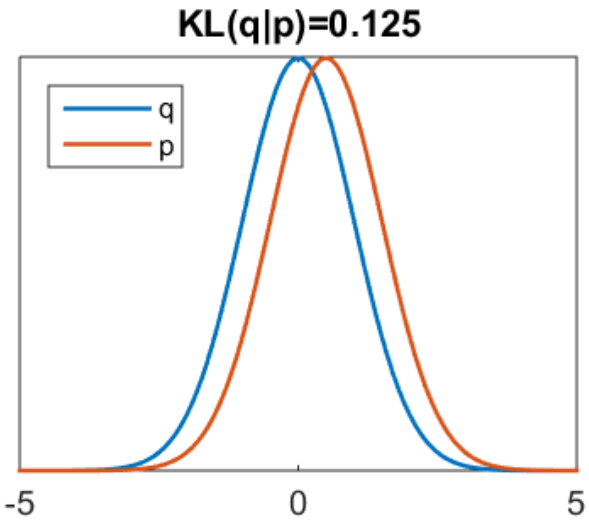
Kullback-Leibler divergence

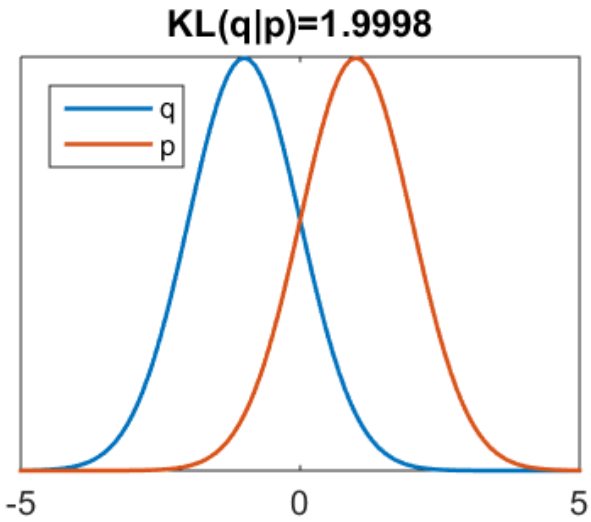
- **KL-divergence**. For two distributions $q(x)$ and $p(x)$

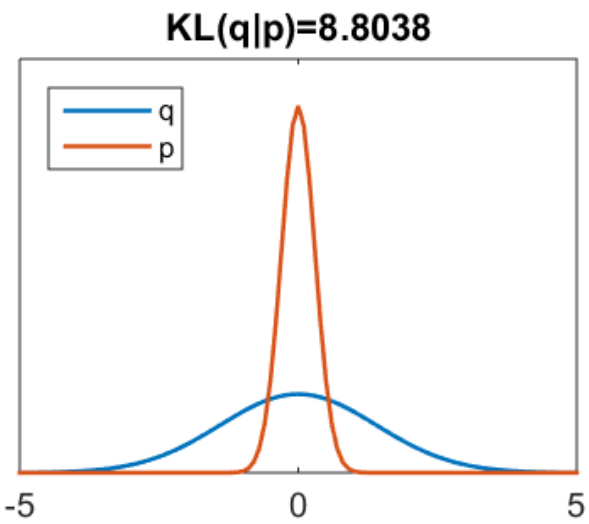
$$KL(q|p) \equiv \int_x q(x) \log \frac{q(x)}{p(x)} dx$$

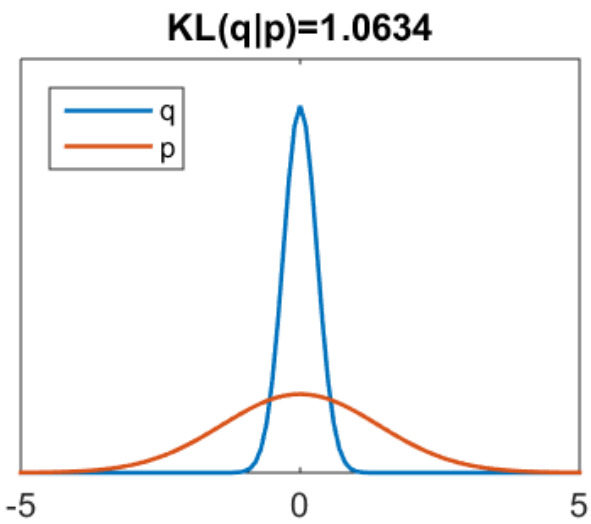
  - $KL(q|p) \geq 0$ (follows from Jensen's inequality)
  - $KL(q|p) = 0$ if and only if $q = p$
  - KL-divergence between q and p can be thought of as a 'distance' of p from q. However, $KL(q|p) \neq KL(p|q)$. Hence it's rather called 'divergence'.

Kullback-Leibler divergence - Example









Mean-field variational Bayes

- **Mean-field variational Bayes**: assume that the approximating distribution q factorizes according to M disjoint groups of $\mathbf{z}$

$$q(\mathbf{z}) = \prod_{i=1}^{M} q_i(\mathbf{z}_i)$$

- Distributions $q(\mathbf{z}_i)$ are called **factors**
- NB: above $\mathbf{z}$ is a generic notation for all unobserved variables in the model, and comprises both parameters (e.g. π, Λ, μ in a GMM) and latent variables (e.g. cluster labels $\mathbf{z}$ in a GMM!)
- For example, assuming:

$$q(\mathbf{z}, \pi, \Lambda, \mu) = q(\mathbf{z})q(\pi, \Lambda, \mu)$$

leads to a tractable solution for the posterior $p(\mathbf{z}, \pi, \Lambda, \mu|\mathbf{x})$ of a GMM.

Mean-field variational Bayes updates

- Assume some current values for all factors $q_i(\mathbf{z}_i)$
- It can be shown (p. 465-466) that by keeping other factors $q_i(z_i)$ fixed for $i \neq j$, the lower bound $\mathcal{L}(q)$ of $\log p(\mathbf{x})$ can be mazimized (or $KL(q||p)$ minimized) by updating factor $q_j(\mathbf{z}_j)$ using

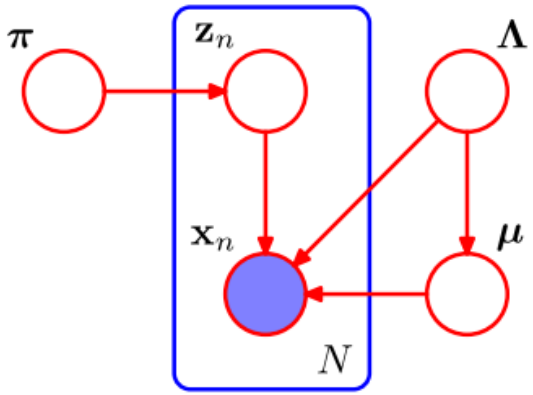
$$\log q_j^*(\mathbf{z}_j) = E_{q(\mathbf{z}_{\setminus j})}\left[\log p(\mathbf{x}, \mathbf{z})\right] + \text{const.}$$

- Here $q(\mathbf{z}_{\setminus j})$ is a short-hand for $\prod_{i \neq j} q_i(\mathbf{z}_i)$
- **Important formula**, as it forms the basis of deriving VB algorithms using factorized distributions
- **Algorithm**: update each factor in turn until convergence

Mean-field VB in practice (1/2)

- Assume a factorization, e.g., $q(\mathbf{z}, \pi, \Lambda, \mu) = q(\mathbf{z})q(\pi)q(\Lambda, \mu)$
- Write the log of the joint distribution

$$\log p(\mathbf{x}, \mathbf{z}, \mu, \Lambda, \pi) = \log p(\mathbf{x}|\mathbf{z}, \Lambda, \mu) + \log p(\mu|\Lambda) + \log p(z|\pi) + \log p(\Lambda) + \log p(\pi)$$



Mean-field VB in practice (2/2)

- When updating a certain factor, for example $q(\mathbf{z})$, we identify terms in the log of the joint distribution that depend on $\mathbf{z}$, and compute their expectation over other unobserved variables

$$\begin{aligned}\log q^*(\mathbf{z}) &= E_{q(\pi)q(\Lambda,\mu)}\left[\log p(\mathbf{x},\mathbf{z},\mu,\Lambda,\pi)\right] + \text{const} \\ &= E_{q(\Lambda,\mu)}\left[\log p(\mathbf{x}|\mathbf{z},\Lambda,\mu)\right] + E_{q(\pi)}\left[\log p(\mathbf{z}|\pi)\right] + \text{const}\end{aligned}$$

- Finally, we exponentiate and normalize to give the updated $q^*(\mathbf{z})$

$$q^*(\mathbf{z}) = \frac{\exp\left(E_{\pi,\Lambda,\mu}\left[\log p(\mathbf{x},\mathbf{z},\mu,\Lambda,\pi)\right]\right)}{\int \exp\left(E_{\pi,\Lambda,\mu}\left[\log p(\mathbf{x},\mathbf{z},\mu,\Lambda,\pi)\right]\right) d\mathbf{z}}$$

  If conjugate priors are used, this belongs to the same family as the prior.

- Notation: instead of $E_{q(\pi,\Lambda,\mu)}$ we may simply use $E_{\pi,\Lambda,\mu}$ or just E.

Idea of derivation of the mean-field VB update*

- Assume just two hidden variables z_1 and z_2 and $q(z_1, z_2) = q_1(z_1)q_2(z_2)$. Then

$$\mathcal{L}(q) = \int q(\mathbf{z}) \log \frac{p(x, \mathbf{z})}{q(\mathbf{z})} d\mathbf{z} = \int q_1(z_1)q_2(z_2) \log \frac{p(x, z_1, z_2)}{q_1(z_1)q_2(z_2)} dz_1 dz_2$$

$$= \cdots = \int q_1(z_1) \log \frac{\widetilde{p}(x, z_1)}{q_1(z_1)} dz_1 + \text{const} = -KL(q_1, \widetilde{p}) + \text{const},$$

where $\widetilde{p}(x, z_1)$ is a distribution defined by

$$\log \widetilde{p}(x, z_1) = E_{q_2(z_2)}[\log p(x, z_1, z_2)] + \text{const}.$$

- We see that $\mathcal{L}(q)$ is maximized w.r.t. to q_1 when $KL(q_1, \widetilde{p})$ is minimized, i.e. when

$$q_1(z_1) = \widetilde{p}(x, z_1).$$

Simple example

- Model: assume that we have observations $\mathbf{x} = (x_1, \ldots, x_N)$ s.t.

$$p(x_n|\theta, \tau) = (1 - \tau)N(x_n|0, 1) + \tau N(x_n|\theta, 1)$$

Prior:

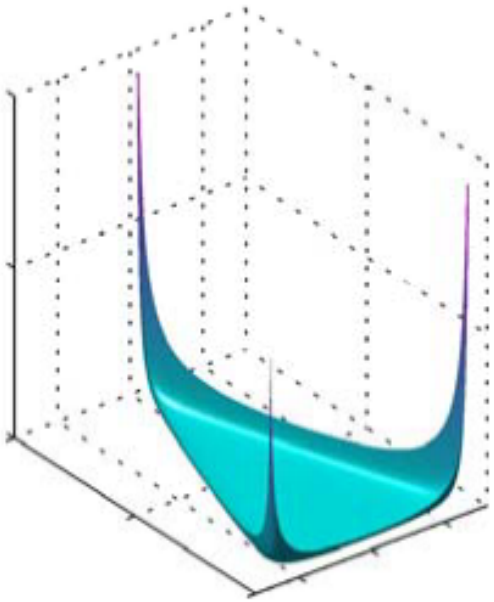
$$\tau \sim Beta(\alpha_0, \alpha_0) \qquad \theta \sim N(0, \beta_0^{-1})$$

Formulation using latent variables $\mathbf{z} = (z_1, \ldots, z_n)$:

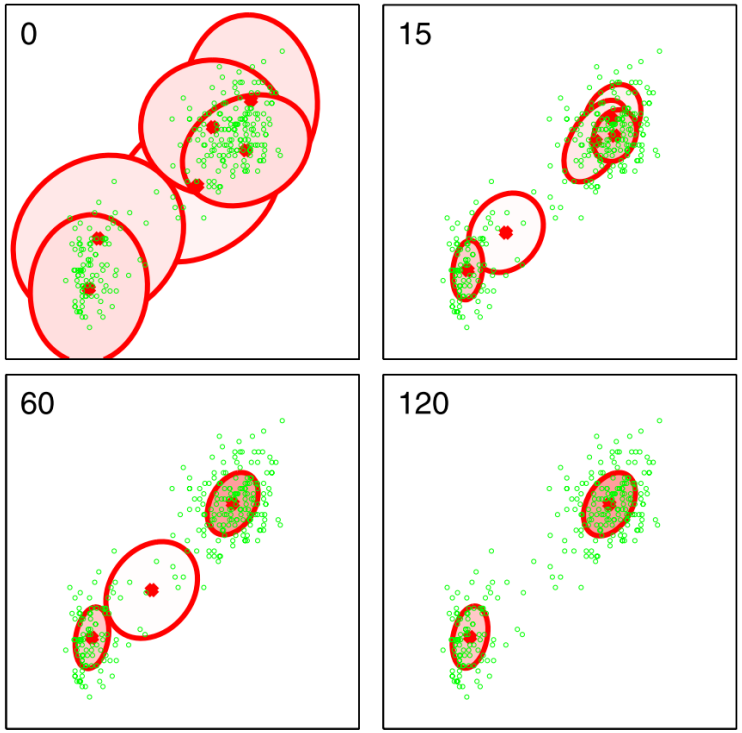
$$p(\mathbf{z}|\tau) = \prod_{n=1}^{N} \tau^{z_{n2}} (1 - \tau)^{z_{n1}}$$
$$p(\mathbf{x}|\mathbf{z}, \theta) = \prod_{n=1}^{N} N(x_n|0, 1)^{z_{n1}} N(x_n|\theta, 1)^{z_{n2}}$$

- *simple_vb_example.pdf*, and the next exercise.

Mean-field VB for the general GMM*

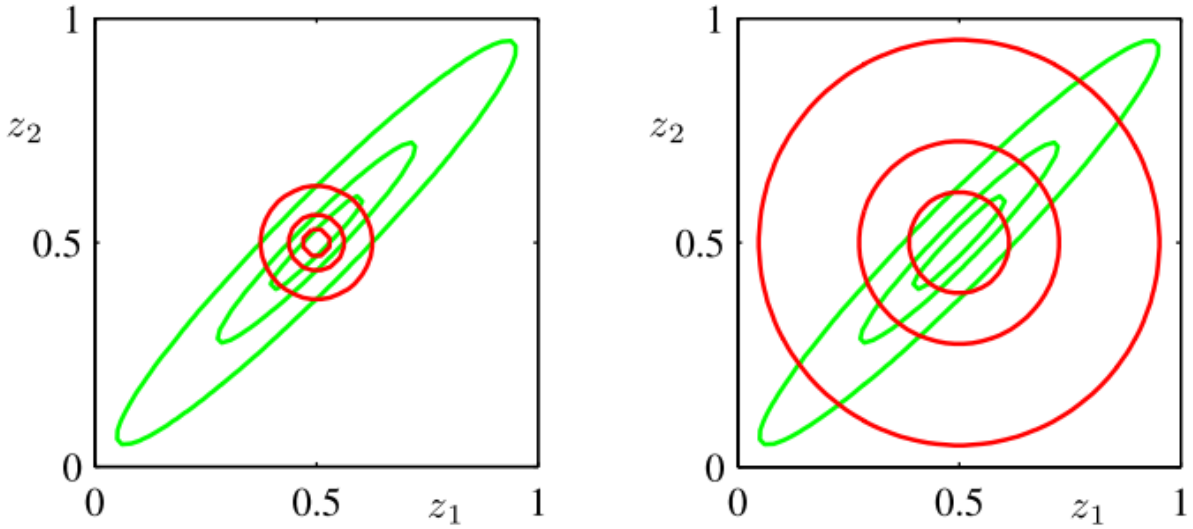


Bishop, Fig 2.5



- *Dirichlet*$(\pi|\alpha_0)$ prior on mixture coefficients with $\alpha_0 < 1$ favors **sparse solutions** $\rightarrow$some components remain empty, with corresponding parameters μ_k, Λ_k following prior distributions
- Avoids overfitting and singularities present in the EM algorithm.

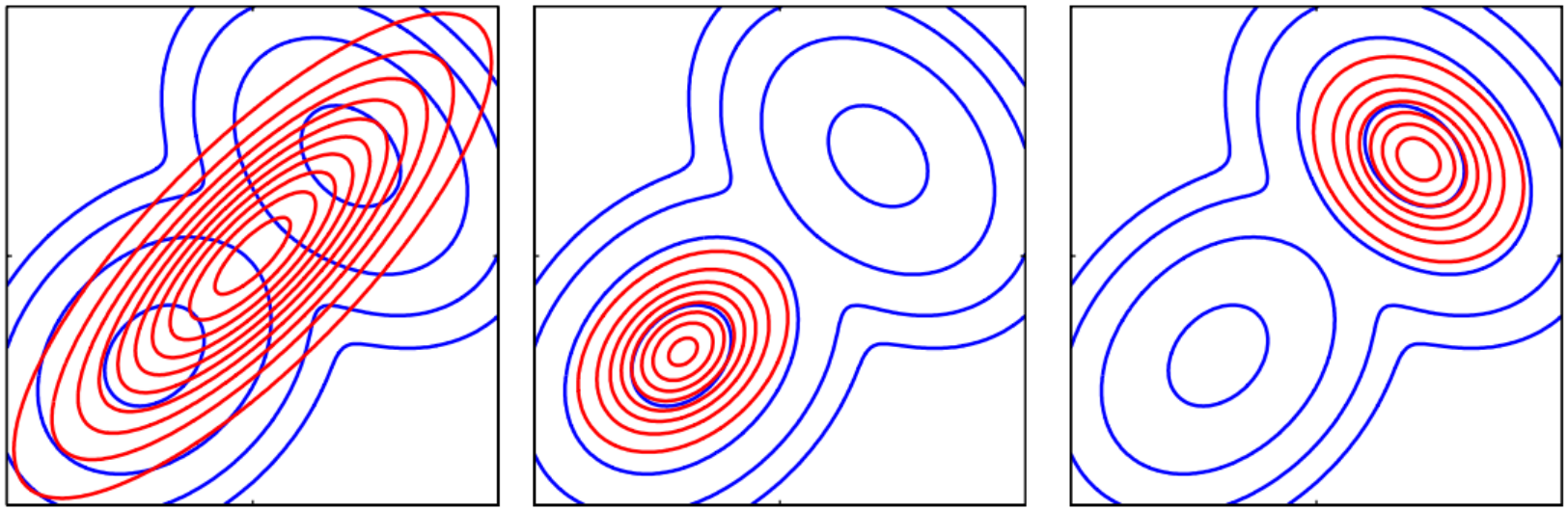
Properties of factorized approximations (1/2)



- Green: $p(\mathbf{z}|\mathbf{x})$, red: $q(\mathbf{z})$
- **Left**: q that minimizes $KL(q||p)$
- **Right**: q that minimizes $KL(p||q)$

$\longrightarrow$variational approximation (left) **underestimates uncertainty**.

Properties of factorized approximations (2/2)



- Blue: $p(\mathbf{z}|\mathbf{x})$, red: $q(\mathbf{z})$
- **Left**: q that minimizes $KL(p||q)$
- **Center**: q represents a local minimum of $KL(q||p)$
- **Right**: q represents another local minimum of $KL(q||p)$

$\longrightarrow$variational approximation usually captures only a single mode.

Important points

- Variational Bayes aims to find a tractable approximation $q(\mathbf{z})$ for the posterior distribution $p(\mathbf{z}|\mathbf{x})$.
- $q(\mathbf{z})$ is found by maximizing the ELBO $\mathcal{L}(q)$ or, equivalently, by minimizing $KL(q||p)$.
- Mean-field VB: if $q(\mathbf{z}) = \prod_{i=1}^{M} q_i(\mathbf{z}_i)$, factor $q_j(\mathbf{z}_j)$ can be updated using

$$\log q_j^*(\mathbf{z}_j) = E_{q(\mathbf{z}_{\setminus j})}\left[\log p(\mathbf{x}, \mathbf{z})\right] + \text{const.}$$

- Variational approximation for a fully Bayesian model with prior distributions avoids some of the problems related to the ML estimation of the GMM (overfitting, singularities).

Advanced probabilistic methods

Lecture 7: Model selection

Pekka Marttinen

Aalto University

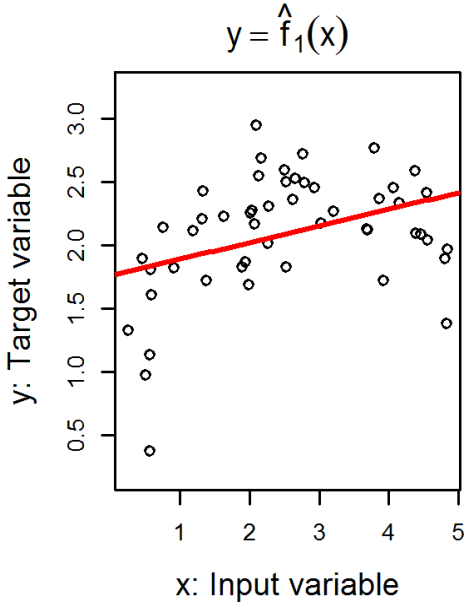
March, 2021

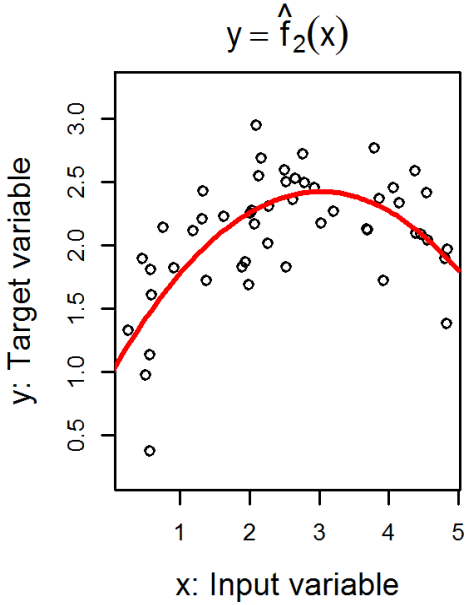
Lecture 7 overview

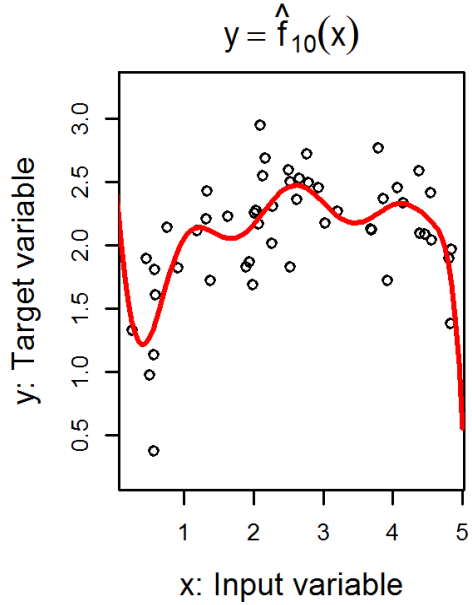
- Bayesian model selection
  - marginal likelihood
  - BIC, Laplace approximation
  - VB lower bound (ELBO)
- Predictive model selection
  - AIC, (DIC, WAIC, etc.)
  - Cross-validation
- Lecture based on (suggested reading):
  - Barber: Ch. 12 (Bayesian model selection)
  - *simple_elbo.pdf* (how to derive the ELBO for the simple model analytically)
  - For predictive model selection: Hastie et al. *The Elements of Statistical Learning*, (available at *http://statweb.stanford.edu/~tibs/ElemStatLearn/*): Ch. 7.1, 7.2, 7.4, 7.5, 7.10 (for AIC and CV)

Model selection

- Possible goal may be to learn
  - **the most useful model**, for example the one that best predicts future observations
  - **the most probable model**, for example when comparing between scientific hypotheses and different hypotheses correspond to different models







Bayesian model selection

- Consider m models M_i with parameters θ_i and associated priors,

$$p(x, \theta_i | M_i) = p(x | \theta_i, M_i) p(\theta_i | M_i), \quad i \in 1, \ldots, m,$$

- We can compute the **model posterior probabilities**

$$p(M_i | x) = \frac{p(x | M_i) p(M_i)}{p(x)},$$

where

$$p(x | M_i) = \int p(x | \theta_i, M_i) p(\theta_i | M_i) d\theta_i \quad \text{and}$$

$$p(x) = \sum_{i=1}^{m} p(x | M_i) p(M_i)$$

Bayes factors

- For comparing two models, we compute the **Bayes' factor**

$$\underbrace{\frac{p(M_i|x)}{p(M_j|x)}}_{\text{Posterior odds}} = \underbrace{\frac{p(x|M_i)}{p(x|M_j)}}_{\text{Bayes' factor}} \times \underbrace{\frac{p(M_i)}{p(M_j)}}_{\text{Prior odds}},$$

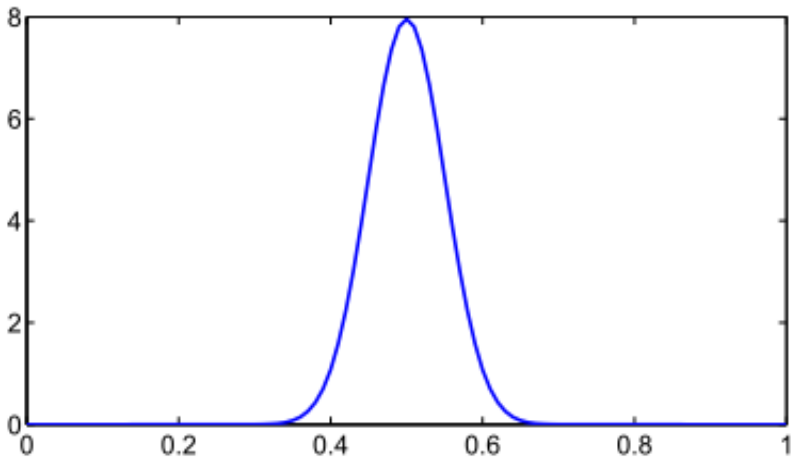
- Bayes factor is the ratio of **marginal likelihoods** $p(D|M_i)$ and it tells how much more seeing the data D has increased the probability of model M_i as opposed to model M_j.

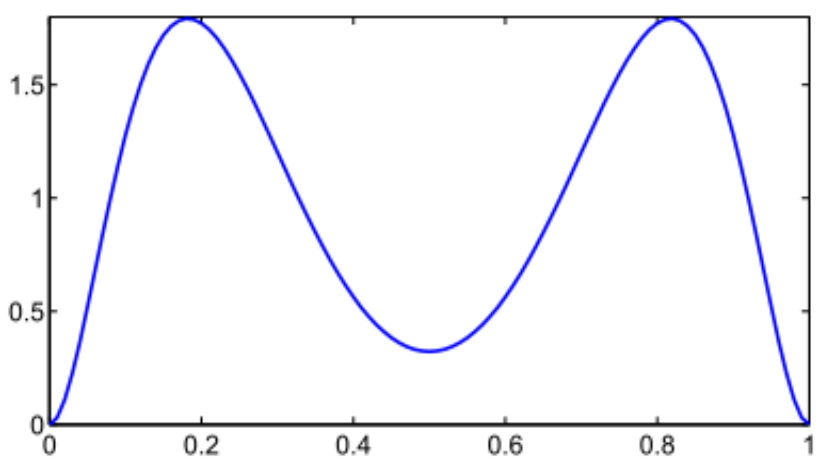
Bayes factor example (1/3)

- **Problem:** given N throws of a coin, determine whether a coin is biased or unbiased.
- Let θ denote the probability of heads. The probability of observing h heads and $N - h$ tails in a sequence of N throws is

$$p(H = h) = \binom{N}{h} \theta^h (1 - \theta)^{N-h}$$

- The difference between models is encoded in the prior distribution of θ (**Left**: fair coin, **Right**: biased coin)





Bayes factor example (2/3)

- M_{fair} ('Coin is fair') corresponds to prior

$$\begin{aligned} p(\theta|M_{fair}) &= Beta(\theta|a,b) \\ &= B(a,b)^{-1}\theta^{a-1}(1-\theta)^{b-1} \end{aligned}$$

  where $a = b = 50$.

- Probability of h heads in N throws of the coin is given by

$$\begin{aligned} p(x|M_{fair}) &= \int p(x|\theta, M_{fair})p(\theta|M_{fair})d\theta \\ &= \binom{N}{h} B(a,b)^{-1} \int \theta^h(1-\theta)^{N-h}\theta^{a-1}(1-\theta)^{b-1}d\theta \\ &= \binom{N}{h} B(a,b)^{-1} \int \theta^{h+a-1}(1-\theta)^{N-h+b-1}d\theta \\ &= \binom{N}{h} B(a,b)^{-1} B(h+a, N-h+b) \end{aligned}$$

Bayes factor example (3/3)

- M_{biased} ('Coin is biased') corresponds to assuming

$$p(\theta|M_2) = 0.5 \times Beta(\theta|3, 10) + 0.5 \times Beta(\theta|10, 3)$$

- We get

$$p(x|M_2) = \frac{1}{2}\binom{N}{h}\left\{\frac{B(h+3, N-h+10)}{B(3,10)} + \frac{B(h+10, N-h+3)}{B(10,3)}\right\}$$

- For example with $h = 50$ and $N = 70$, we get

$$BF_{fair,biased} = \frac{p(x|M_{fair})}{p(x|M_{biased})} = 0.087.$$

This indicates that if the models are *a priori* equally likely, after seeing the data, M_{biased} is about 11 times more probable than M_{fair}.

Laplace approximation for marginal likelihood*

- Laplace approximation for $p(x|M)$

$$\log p(x|M) \approx \log p(x|\widehat{\theta}, M) + \log p(\widehat{\theta}|M) + \frac{D}{2}\log(2\pi) - \frac{1}{2}\log|H_{\widehat{\theta}}|,$$

where

$$\widehat{\theta} = \arg\max_{\theta} p(x|\theta, M)p(\theta|M)$$

is the MAP estimate and $H_{\widehat{\theta}}$ is the Hessian (second derivative for univariate θ) of

$$f(\theta) = -\log\left[p(x|\theta, M)p(\theta|M)\right]$$

at $\widehat{\theta}$.

BIC approximation for marginal likelihood*

- BIC approximation[1]

$$\mathrm{BIC}(M) = \log p(x|\widehat{\theta}, M) - \frac{D}{2} \log N$$

  is obtained from the Laplace approximation by assuming $p(\theta) = const$, $H \approx NI_D$, and $N \to \infty$.

- Note that we can compute the approximate Bayes factor using

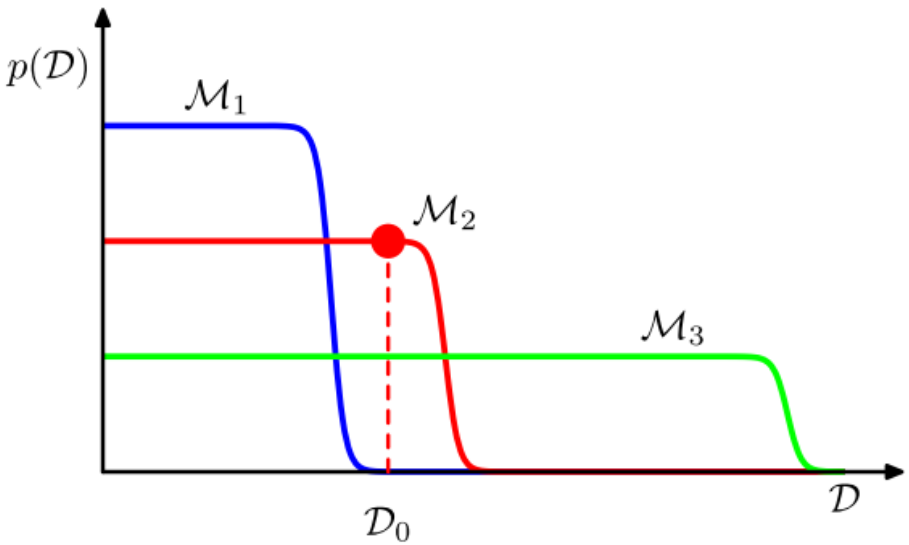
$$\mathrm{BF}_{12} = \frac{\exp(\mathrm{BIC}(M_1))}{\exp(\mathrm{BIC}(M_2))},$$

  or similarly by plugging in exponentiated Laplace approximation (Laplace is better, both to be used with caution, especially with small N).

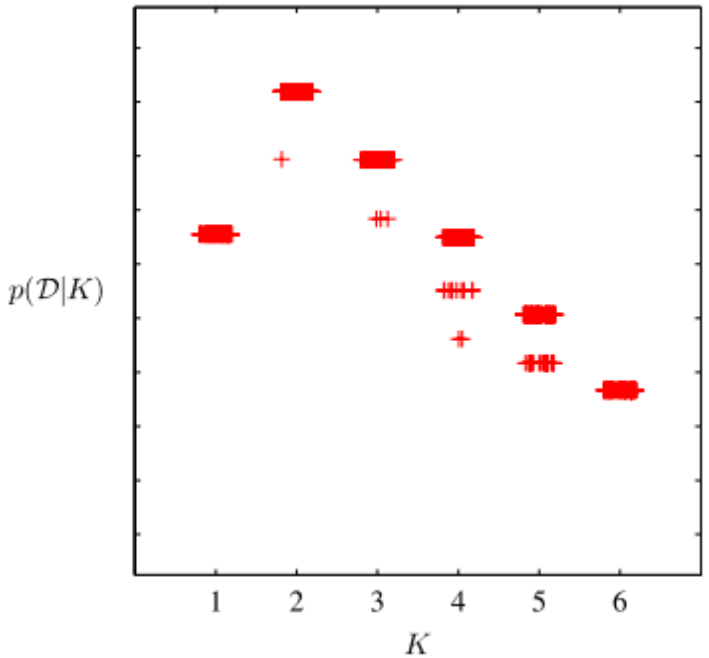
[1] Sometimes there is -2 in the front.

Bayesian model selection and Occam's razor

- When complexity of M increases, $p(x|\widehat{\theta}, M)$ always increases
- On the other hand, $p(x|M)$ **is the highest for the simplest model that can explain the data** (=Occam's razor principle)
- **Left:** illustration with model complexity increasing from M_1 to M_3
- **Right:** $p(x|K)$ for the number K of GMM components for the 'Old Faithful' data (approximated using the ELBO, see the next slides)



Bishop, Fig. 3.13



Bishop, Fig. 10.7

Variational lower bound (ELBO)

- The derivation of the VB algorithm was based on minimizing $KL(q||p)$ in

$$\log p(\mathbf{x}) = \mathcal{L}(q) + KL(q||p)$$

- When conjugate priors and exponential family distributions are used, we can compute the variational lower bound $\mathcal{L}(q)$ directly

$$\mathcal{L}(q) = \int q(\mathbf{z}) \log \left\{ \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})} \right\} d\mathbf{z}$$

- Computing $\mathcal{L}(q)$ gives:
  1. alternative way to define the factor updates by maximizing $\mathcal{L}(q)$.
  2. simple check of the VB algorithm - $\mathcal{L}(q)$ should never decrease.
  3. criterion to monitor convergence.
  4. an estimate of $\log p(x)$ to be used in **model selection**

Simple example: computing the ELBO

- The model:

$$p(x_n|\theta,\tau) = (1-\tau)N(x_n|0,1) + \tau N(x_n|\theta,1), \;\; n = 1,\ldots,N.$$

  Prior:

$$\tau \sim Beta(\alpha_0,\alpha_0) \qquad \theta \sim N(0,\beta_0^{-1})$$

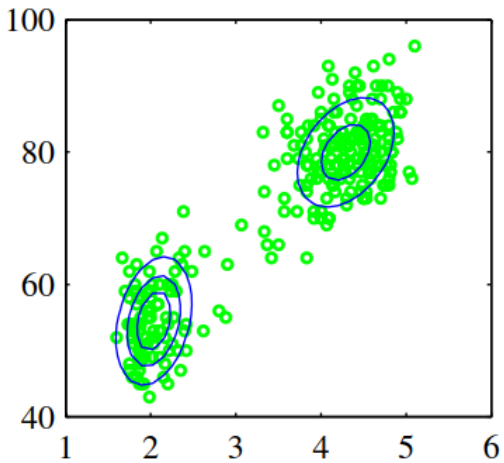
- After factorizing $\log p(\mathbf{x},\mathbf{z},\tau,\theta)$, ELBO can be written as:

$$\begin{aligned}\mathcal{L}(q) = {} & E_{q(\tau)}[\log p(\tau)] + E_{q(\theta)}[\log p(\theta)] + E_{q(\mathbf{z})q(\tau)}[\log p(\mathbf{z}|\tau)] \\ & + E_{q(\mathbf{z})q(\theta)}[\log p(\mathbf{x}|\mathbf{z},\theta)] - E_{q(\mathbf{z})}[\log q(\mathbf{z})] - E_{q(\tau)}[\log q(\tau)] \\ & - E_{q(\theta)}[\log q(\theta)].\end{aligned}$$

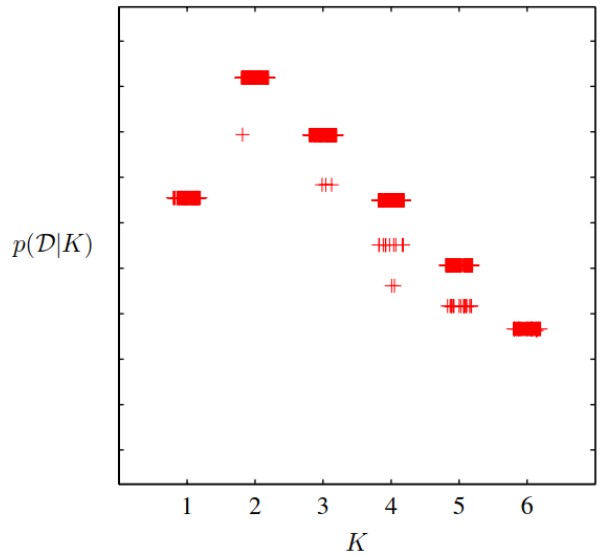
- All of the terms have analytic form (see *simple_elbo.pdf* and the next exercise).

Using the ELBO for model selection

- The ELBO $\mathcal{L}_K$ for a GMM with K components gives a lower bound of $\log p_K(x)$, where $p_K(x)$ is the marginal likelihood.
- However, VB approximates only a single mode and a GMM with K components has $K!$ equivalent modes (label switching). Hence, we add $\log(K!)$ to $\mathcal{L}_K$ when doing model selection (**right**).



Bishop, Fig. 2.21



Bishop, Fig 10.7

Selecting models for prediction, concepts (1/2)

- X: input variables, Y: target variable, $\widehat{f}(X)$: prediction model estimated from a training data $\mathcal{T}$.
- Loss function measures the (lack of) accuracy of prediction
- Squared loss

$$L(Y, \widehat{f}(X)) = (Y - \widehat{f}(X))^2$$

- Loss based on log-likelihood

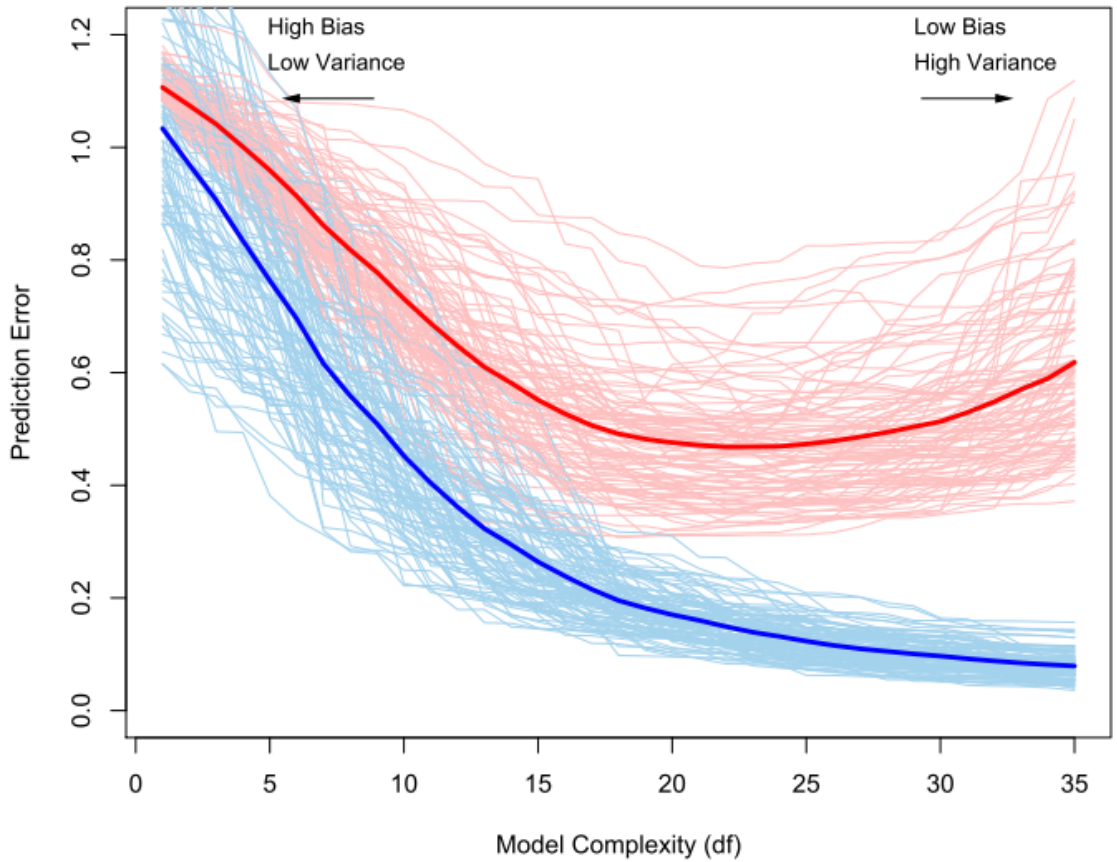
$$L(Y, \theta(X)) = -2 \log p(Y|\theta(X)),$$

where $\theta(X)$ is a parameter of the prediction model.

Selecting models for prediction, concepts (2/2)

$$\mathrm{Err}_{\mathcal{T}} = E\left[L(Y, \widehat{f}(X))|\mathcal{T}\right] \qquad \text{(test/prediction/generalization error)}$$

$$\overline{\mathrm{err}} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \widehat{f}(x_i)) \qquad \text{(training error)}$$

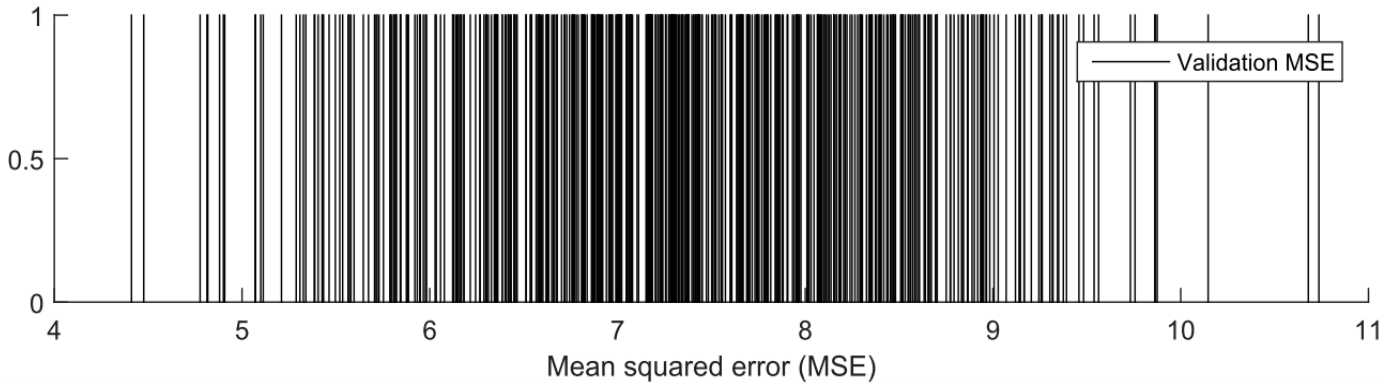


Predictive model selection criteria

- Predictive model selection criteria aim to approximate **expected prediction accuracy** in a new data set, either
  - analytically (e.g. AIC, DIC, WAIC), or
  - by efficient sample re-use (e.g. cross-validation)
- Hence, they aim to find a model that is **suitable for prediction**.
- Asymptotically, AIC and leave-one-out cross validation are equivalent.

Example (validation vs. test error)*

- Data $(\mathbf{x}_i, y_i)$ is simulated using $y_i = \sum_{i=1}^{30} w_i x_i + \epsilon_i$, where $w_i \sim U(-1, 1)$, and $\epsilon_i \sim N(0, 0.1^2)$.
- 500 candidate models created by randomly selecting 10 covariates out of 30, and fitting a linear model with the selected covariates.
- $n_{Train} = 300$ and $n_{Valid} = 60$. Validation MSEs for different models:



- **Question:** What is your best guess for the test set MSE for the best model?

AIC, basic idea*

- It can be shown that for large N

$$-2 \cdot E\left[\log p(\widetilde{y}|\widehat{\theta})\right] \approx -\frac{2}{N} \log p(y|\widehat{\theta}) + 2 \cdot \frac{d}{N},$$

  where $\widetilde{y}$ is an unobserved future observation and

$$\log p(y|\widehat{\theta}) = \sum_{i=1}^{N} \log p(y_i|\widehat{\theta})$$

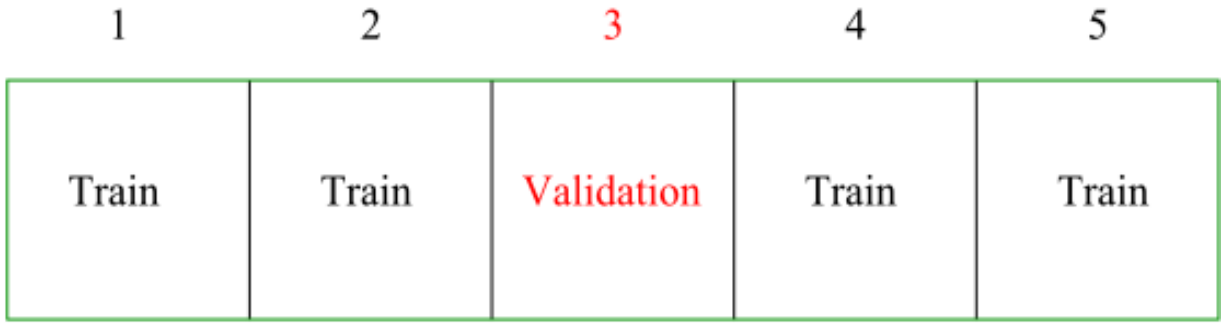
  is the log-likelihood.
- This gives rise to:

$$\text{AIC} = -\frac{2}{N} \log p(y|\widehat{\theta}) + 2 \cdot \frac{d}{N}$$

  (the smallest AIC is the best)
- **Main point**: AIC is one possible analytical approximation for the expected prediction accuracy measured using log probability of future data[2].

[2] For more Bayesian variants, see, e.g., Gelman *et al.* Stat. Comput. (2014)

Cross-Validation (CV)[3], basic idea*



- Let $\kappa : \{1, \ldots, N\} \longmapsto \{1, \ldots, K\}$ denotes the fold to which observation i belongs. Then

$$CV(\widehat{f}) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \widehat{f}^{-\kappa(i)}(x_i)),$$

  where $\widehat{f}^{-\kappa(i)}$ is the predictor model trained without fold $\kappa(i)$.
- CV yields an estimate of the expected prediction error $E\left[L(Y, \widehat{f}(X))\right]$.

[3] See, e.g., Vehtari *et al., Stat. Comput.* (2017).

A wrong way to do cross-validation*

- A (wrong!) strategy for building a classifier with a large number of predictors
  1. Pre-screening of the predictors: find a subset of predictors with strong univariate correlation with the class label
  2. Using the set of predictors from pre-screening, build a multivariate classifier
  3. Use cross-validation to estimate the unknown tuning parameter and to estimate the prediction error of the final model
- **Question:** what's the problem?

The correct way*

- The correct way for building a classifier with a large number of predictors
  1. Divide the samples into K folds
  2. For each fold $k = 1, \ldots, K$
     - Find a subset of predictors with strong univariate correlation with the class labels, using all samples except those in fold k.
     - Build a multivariate classifier using this set of predictors (excluding fold k)
     - Use the classifier to predict the class labels for the samples in fold k
- The class labels of the test fold should not be used at any point before predicting them in CV!

Remarks

- Bayesian model selection
  - asymptotically consistent
  - suitable when trying to find the "true" model from a set of distinct interpretable alternatives
  - heavy penalty on complexity $\longrightarrow$ may produce too sparse models for prediction
  - may be sensitive to the prior on the parameters
- Predictive model selection
  - no consistency guarantees
  - no need to assume a true model
  - less penalty for model complexity $\longrightarrow$ more complex models that may be suitable for prediction
- In practice people seem to use the two ways interchangeably for both goals: prediction and comparing hypotheses.

Model selection, summary

- There are two **different goals** for model selection: learning the correct model or selecting a model for prediction
- The **Bayesian model selection** gives probabilities of different models and may be more suitable if the goal is to learn the correct model.
- **Predictive model selection** criteria may be better if the goal is to use the model for prediction.
- BIC approximates Bayesian model selection, AIC and CV are related to predictive model selection.
- ELBO can be used to approximate the logarithm of the marginal likelihood $\log p_m(x)$ for model m, which can be used for Bayesian model selection.

Advanced probabilistic methods

Lecture 8: Factor analysis

Pekka Marttinen

Aalto University

March, 2021

Lecture 8 overview

- Factor analysis (FA)
- Probabilistic description of the model
- Some examples
- An extension: Group Factor Analysis
- Suggested reading: Ch. 21 of Barber

Two different views on classical multivariate analysis[1]

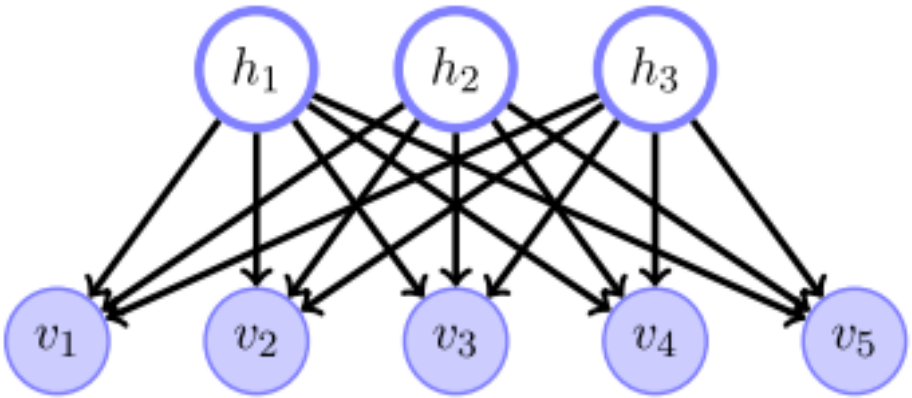
Given an $N \times D$ data matrix, we may be interested in comparing

1. rows of the data matrix (**individuals**)
   - starting point: similarities between individuals
   - techniques: **clustering**, multidimensional scaling, discriminant analysis
2. columns of the data matrix (**variables**)
   - starting point: correlation/covariance matrix between variables
   - techniques: **factor analysis**, principal component analysis, canonical correlation analysis

[1] From Mardia, K.V. (1980). Multivariate Analysis

Factor analysis - intuition

- Factor analysis attempts to explain correlation between a large set of visible variables (**v**) using a small number of hidden factors (**h**).
- It is not possible to observe the factors directly. The visible variables depend on the factors but are also subject to random error.
- A central tool in statistics, a simple example of **representation learning**, and a building block for more complex (deep) models.



Factor analysis, probabilistic description

- FA model generates a D-dimensional observation $\mathbf{v}$ from the H-dimensional vector $\mathbf{h}$ according to

$$\mathbf{v} = F\mathbf{h} + \mathbf{c} + \epsilon,$$

where

$$\epsilon \sim N(0, \Psi), \qquad \Psi = \mathrm{diag}(\psi_1, \ldots, \psi_D).$$

- The $D \times H$ *factor loading matrix* F tells how the factors affect the observations: f_{ij} is the effect of factor h_j on variable v_i.

Factor analysis (example) (1/3)

- Data matrix contains results of 5 exams for 120 students (see *factorandemo* in Matlab)
  - Exams 1 and 2 are about mathematics, exams 3 and 4 about literature, and exam 5 is comprehensive.
- Goal of analysis: to investigate if the results could be understood using a smaller number of characteristics (or, factors) of students, e.g., 'quantitative' and 'qualitative' skills.

$$\text{Data} = \begin{bmatrix} 65 & 77 & 69 & 75 & 69 \\ 61 & 74 & 70 & 66 & 68 \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix}$$

- The n^{th} row of the data matrix is $v_n^T = (v_{n1}, \dots, v_{n5})$

Factor analysis (example) (2/3)

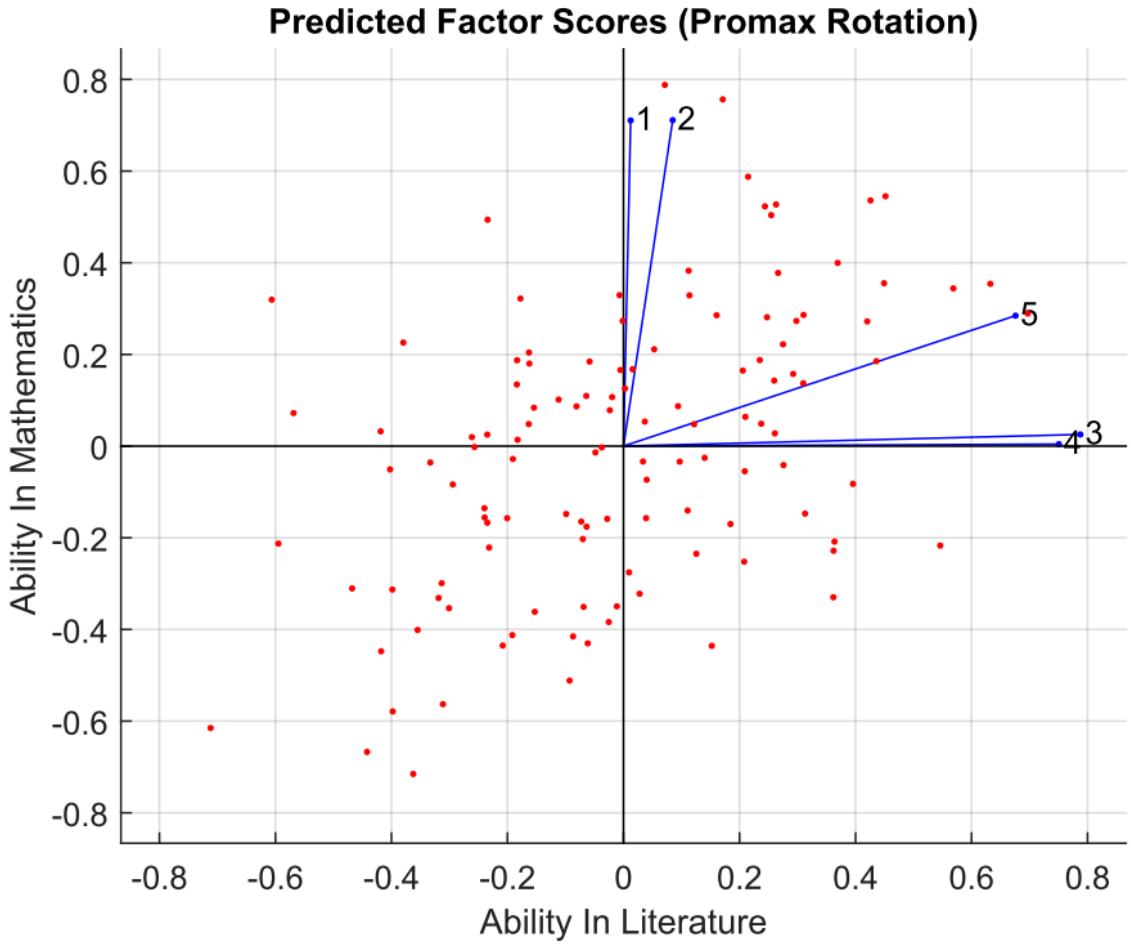
- Underlying model in detail

$$\begin{bmatrix} v_{n1} \\ v_{n2} \\ v_{n3} \\ v_{n4} \\ v_{n5} \end{bmatrix} = \begin{bmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \\ f_{31} & f_{32} \\ f_{41} & f_{42} \\ f_{51} & f_{52} \end{bmatrix} \times \begin{bmatrix} h_{n1} \\ h_{n2} \end{bmatrix} + \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} + \epsilon_n$$

$$\epsilon_n \sim N_5(0, \Psi), \; \Psi = \begin{bmatrix} \psi_1 & 0 & 0 & 0 & 0 \\ 0 & \psi_2 & 0 & 0 & 0 \\ 0 & 0 & \psi_3 & 0 & 0 \\ 0 & 0 & 0 & \psi_4 & 0 \\ 0 & 0 & 0 & 0 & \psi_5 \end{bmatrix}$$

Factor analysis (example) (3/3)

- Results



$$\widehat{F} = \begin{bmatrix} 0.01 & 0.71 \\ 0.08 & 0.71 \\ 0.79 & 0.03 \\ 0.75 & 0.00 \\ 0.68 & 0.28 \end{bmatrix}$$

Equivalent model without latent factors

- Given

$$p(\mathbf{v}|\mathbf{h}) = N_D(\mathbf{v}|F\mathbf{h} + \mathbf{c}, \Psi)$$

and assuming a prior on $\mathbf{h}$:

$$p(\mathbf{h}) = N_H(\mathbf{h}|0, I),$$

integrating out $\mathbf{h}$ yields

$$p(\mathbf{v}) = \int p(\mathbf{v}|\mathbf{h})p(\mathbf{h})d\mathbf{h} = N(\mathbf{v}|\mathbf{c}, FF^T + \Psi)$$

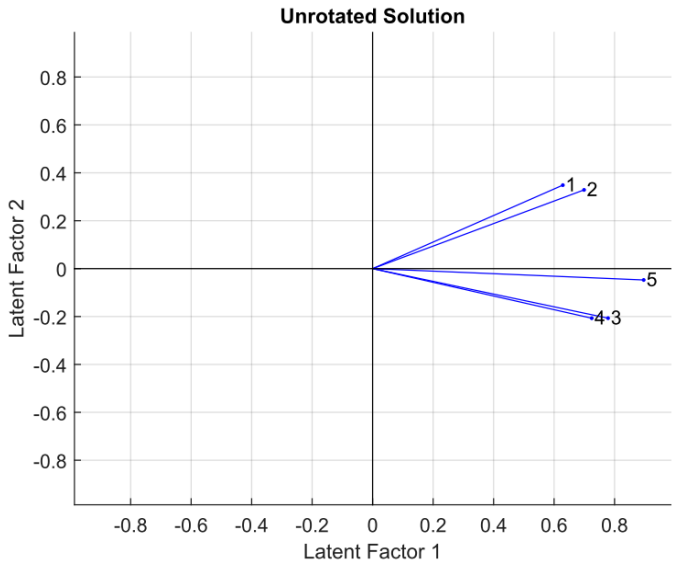
- The result follows from the *Linear transformation* of a Gaussian (see Lecture 3).

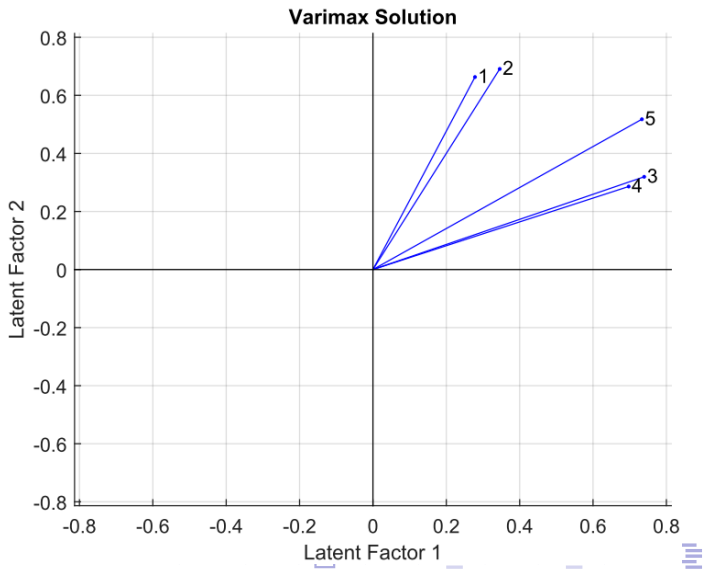
Rotation invariance

- The likelihood is unchanged if we rotate F using FR, with $RR^T = I$:

$$FR(FR)^T + \Psi = FRR^T F^T + \Psi = FF^T + \Psi.$$

- R is often selected to produce interpretable factors. *Varimax* rotation makes each column of F to have only a small number of large values.
- Note: rotation invariance does not matter if the goal is to fit the model in order to use it for prediction. For interpreting the factors, it does.





Probabilistic principal component analysis

- Probabilistic PCA has almost same the model as FA

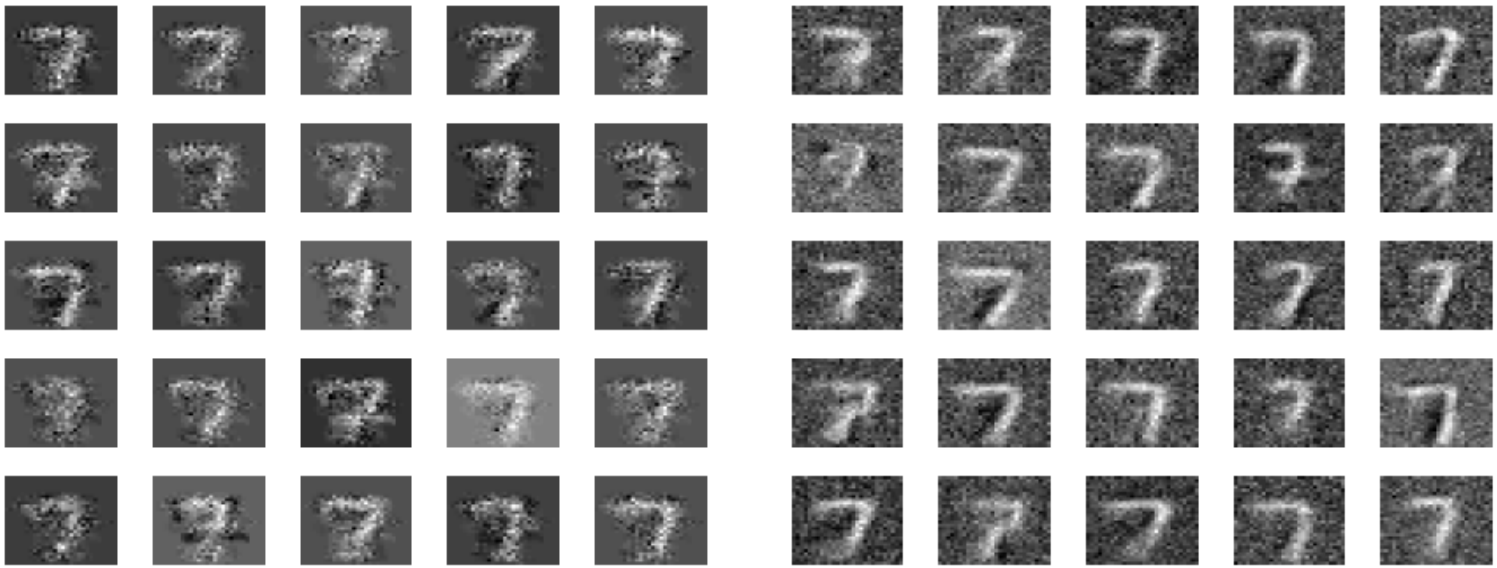
$$\mathbf{v} = F\mathbf{h} + \mathbf{c} + \epsilon,$$

$$\epsilon \sim N(0, \Psi), \qquad \Psi = \sigma^2 I.$$

- In FA

$$\Psi = \text{diag}(\psi_1, \ldots, \psi_D).$$

Example PPCA and FA, digit modeling

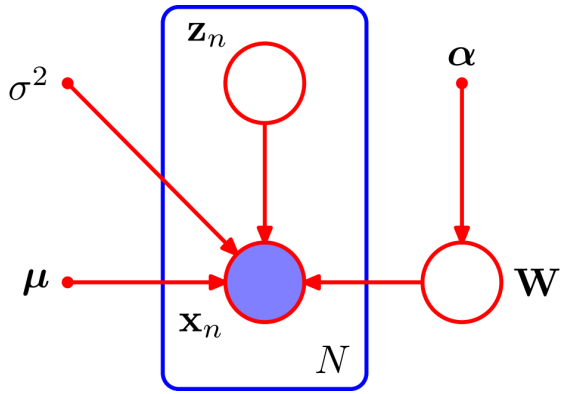


(a) Factor Analysis (b) PPCA

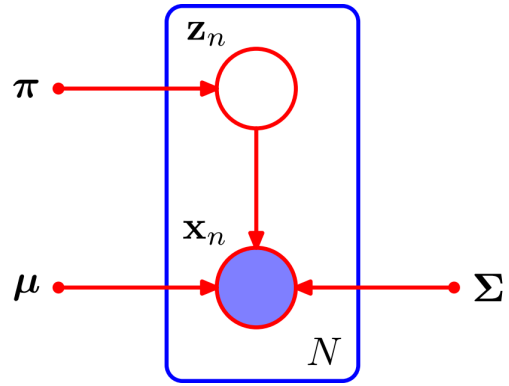
- Samples drawn from FA and PPCA models trained for digit 7.
- FA has different noise parameters for each pixel $\longrightarrow$ reduced noise in boundary regions.

FA vs. GMM

- How are FA and GMM similar? How are they different?



Bayesian PCA (Bishop, Fig. 12.13)



GMM (Bishop, Fig. 9.6)

FA, geometric intuition (1/2)

- FA assumes that the data lies close to a low-dimensional linear manifold
- For example, if $H = 1$ and $D = 2$:

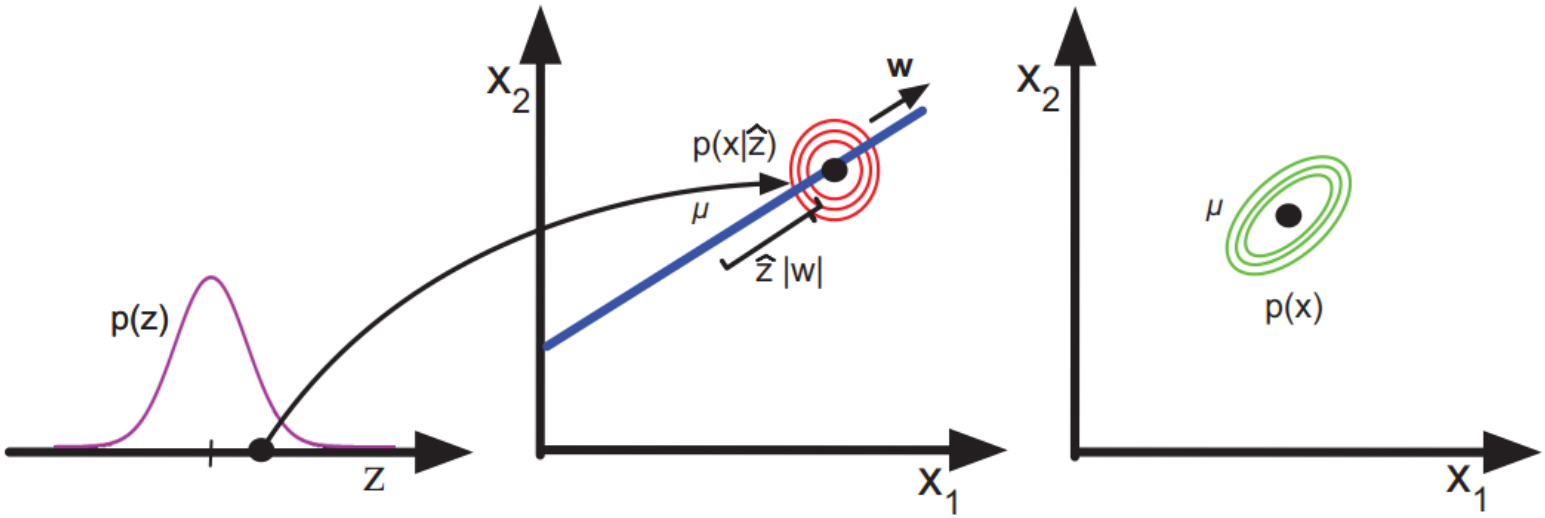


Figure: 12.1 in Murphy

FA, geometric intuition (2/2)

- If $H = 2$ and $D = 3$, the data points form a 'pancake'

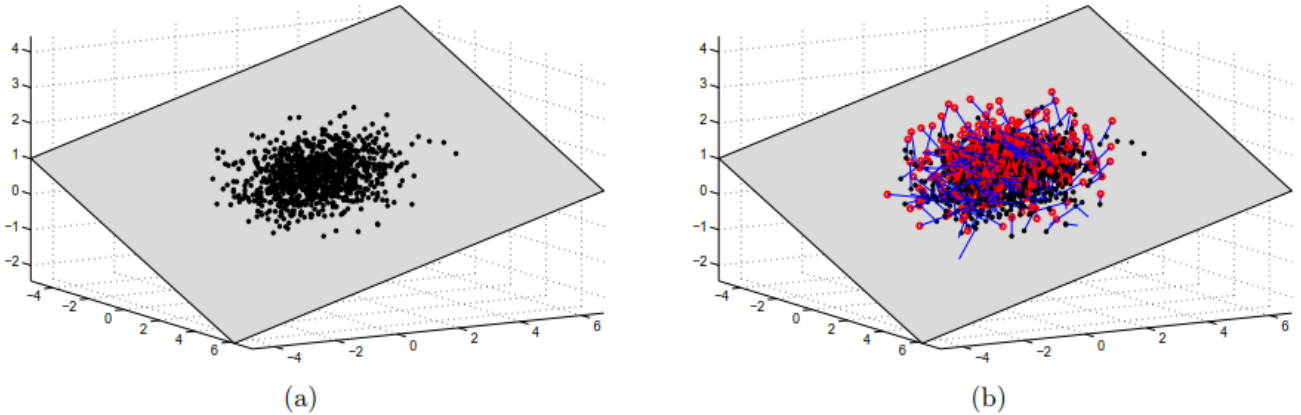


Figure: 21.2 in Barber

- Left: latent 2D points $\mathbf{h}_n$ sampled from $N(\mathbf{h}|\mathbf{0}, \mathbf{I})$ and mapped to the 3D plane by $\mathbf{v}_n^0 = F\mathbf{h}_n + \mathbf{c}$.
- Right: data points $\mathbf{v}_n$ are obtained by adding noise $\mathbf{v}_n = \mathbf{v}_n^0 + \epsilon_n$, where $\epsilon_n \sim N(\mathbf{0}, \Psi)$

Mixture of factor analysers

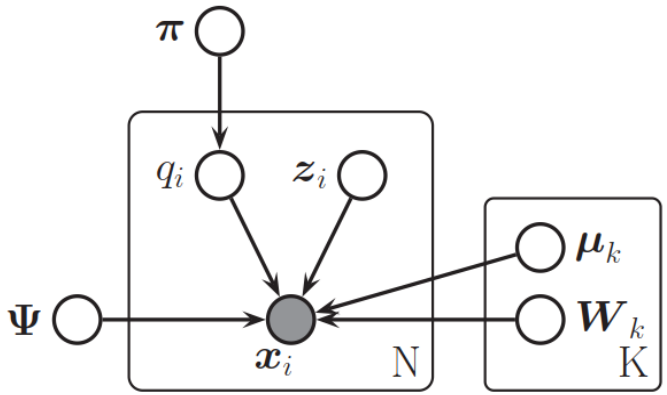


Figure: 12.3 in Murphy

Left: $K = 1$, right, $K = 10$:

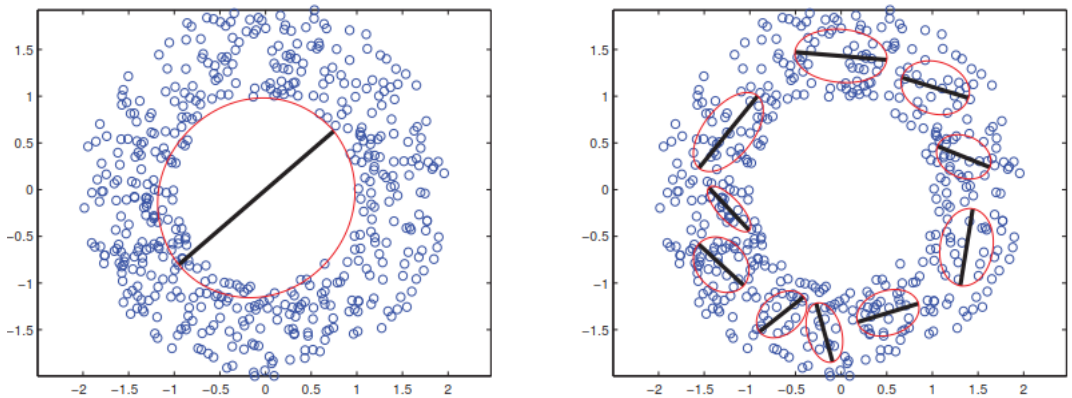


Figure: 12.4 in Murphy

Fitting the FA model

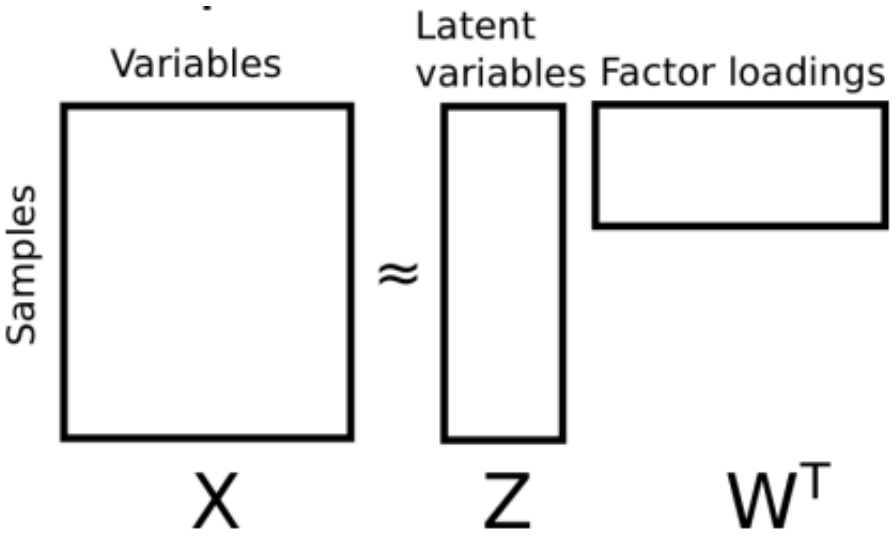
- EM algorithm
- Mean-field VB straightforward with conjugate priors (left as an exercise)
- Stochastic variational inference (next week)
- MCMC
- etc.

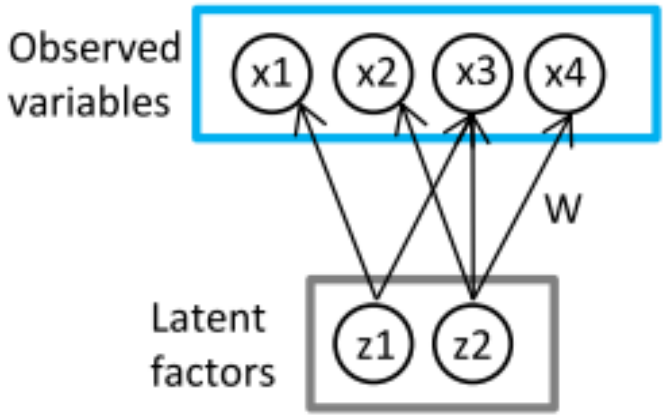
Determining the number of factors

- Same techniques as for determining the number of clusters in GMMs
  - Bayesian model selection
  - Cross-validation
  - ...
- Automated relevance determination (ARD)
  - **shrink unneeded aspects** of the model, such that they have no impact
  - empty clusters in GMM (corresponding to mixture weights driven to zero)
  - factors that don't have any effect (achieved by a shrinkate prior on the columns of the factor loading matrix, see below)
- Nonparameteric methods
  - Assume infinite number of dimensions with diminishing importance
  - Avoids the selection of any fixed dimension (in principle)
  - Dirichlet process prior for clustering, Beta process prior for factor analysis

Research at Aalto: Group factor analysis (GFA) (1/4)

In standard factor analysis the observed variables are modeled using unobserved latent factors

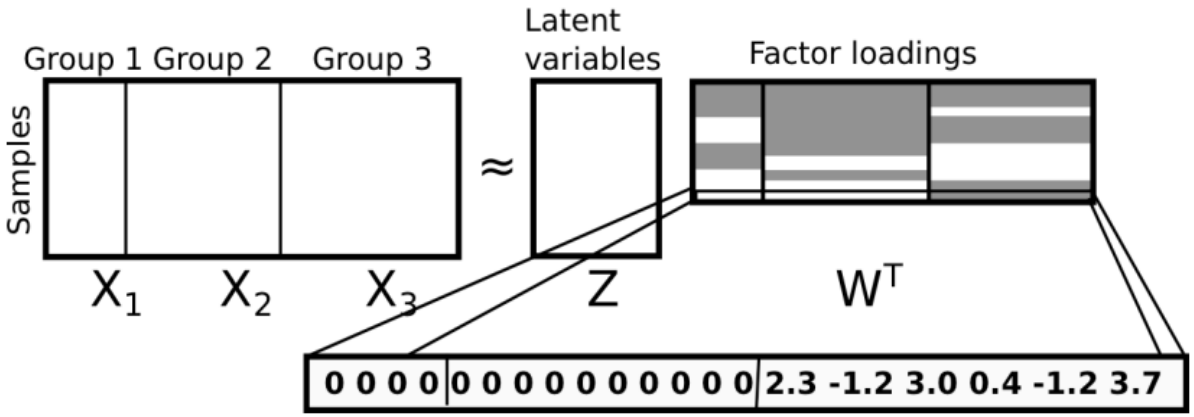




$$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \qquad \mathbf{x} \sim \mathcal{N}(\mathbf{z}\mathbf{W}^T, \mathbf{\Sigma})$$

GFA (2/4)

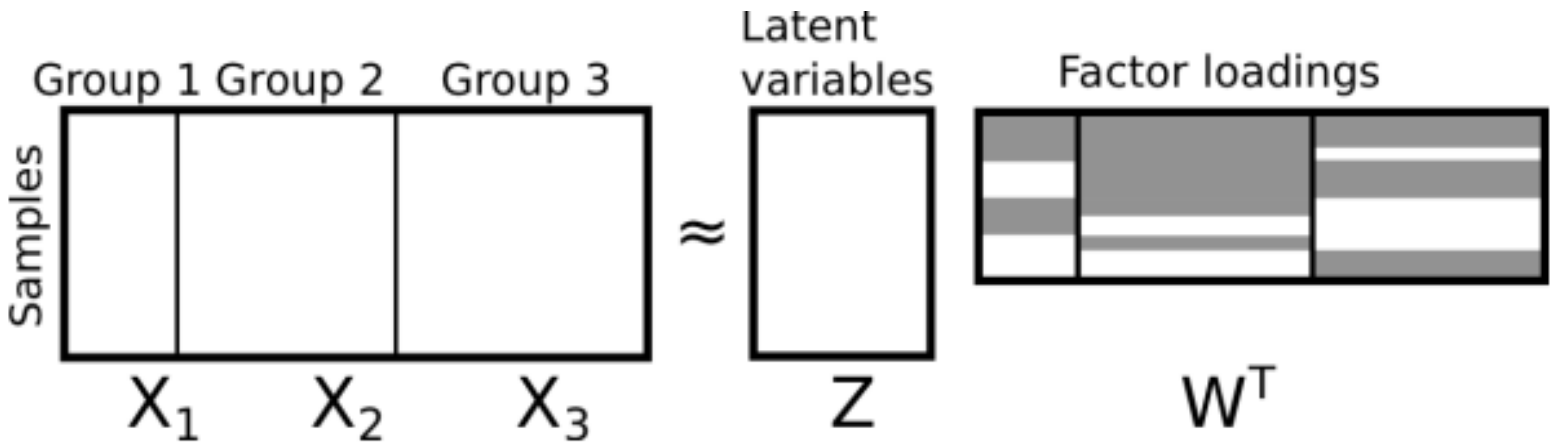
- Group factor analysis (GFA) = FA + specific sparsity structure in the model



Virtanen et al., AISTATS 2012; Klami et al., JMLR 2013

GFA (3/4)

- Group factor analysis (GFA) = FA + specific sparsity structure in the model



- Efficient variational approximation with group-sparsity prior

$$\alpha_{mk} \sim \text{Gamma}(\alpha_0, \beta_0) \qquad w_{mk} \sim N(0, \alpha_{mk}^{-1} I)$$

GFA (4/4)

- Summary of GFA
  - extends FA to model dependencies between groups of variables (as opposed to between individual variables)
  - Efficient inference of group-wise sparsity structure (important for modeling high-dimensional data)
  - basic modeling tool for unsupervised data integration of multiple data sources
  - *http://research.ics.aalto.fi/mi/software/CCAGFA*

GFA example (1/2)

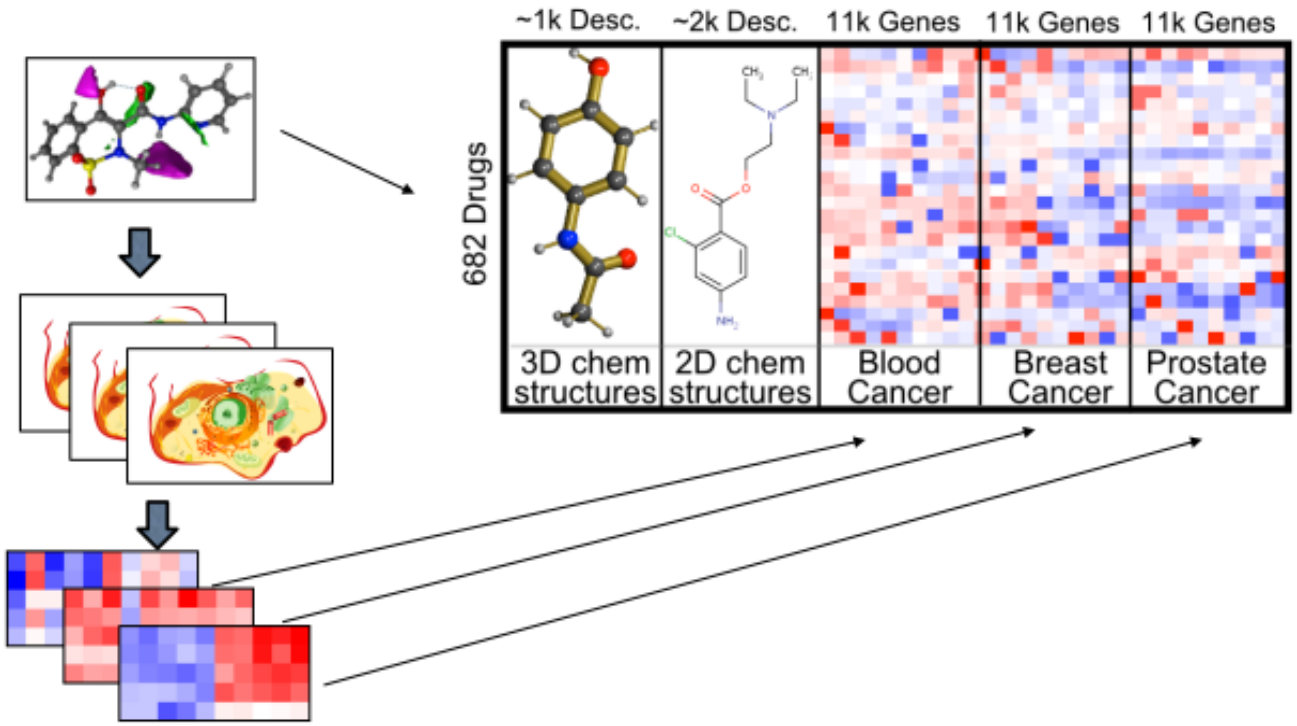
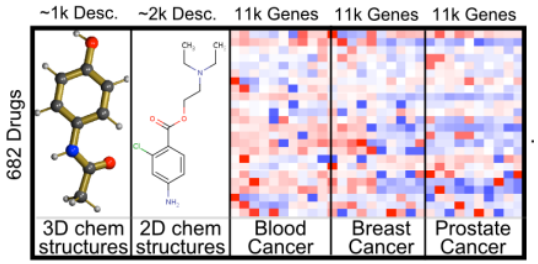
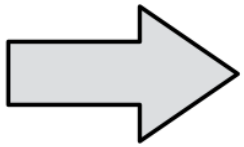


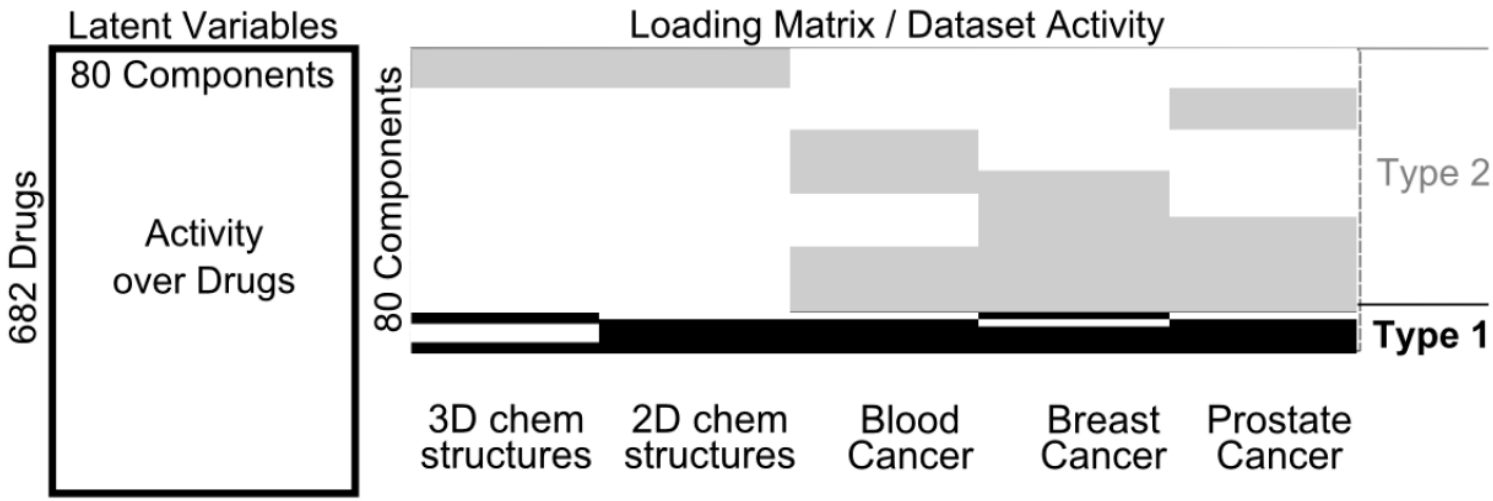
Figure: Khan et al., Bioinformatics (2014)

- GFA for studying associations between drug characteristics and cellular responses

GFA example (2/2)







- GFA for studying associations between drug characteristics and cellular responses

Remark

- FA-model is based on the Gaussian distribution, but often used with other data types as well.
- Pragmatic justification that FA often works well with other data types.
- Performance may not be good with highly non-Gaussian variables, for example binary 0-1 variables with a very small number of individuals with value 1.

Important points

- Factor analysis model explains correlations between variables using latent variables (the factors) that affect several observed variables simultaneously, thus explaining the observed correlations
- FA model can be represented both with and without latent variables
- Factor loading matrix can be rotated without changing the likelihood - this must be kept in mind when interpreting the factors, but does not matter for prediction.
- FA model can be extended in many ways, for example to model dependencies between groups of variables.