Bayesian Networks II
Gaussian Mixture Models

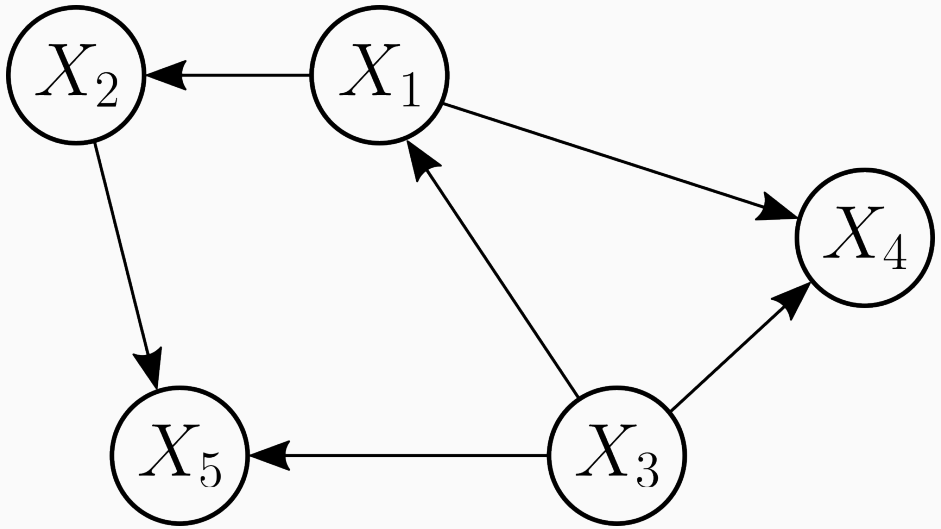
Probabilistic Decision Making — Lecture 7

12th November 2025

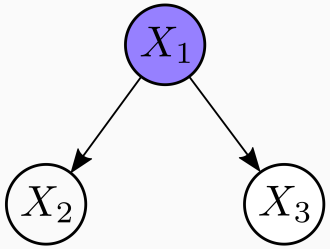
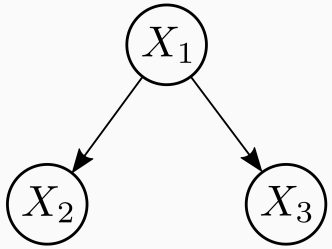
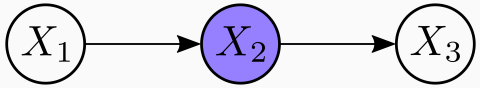
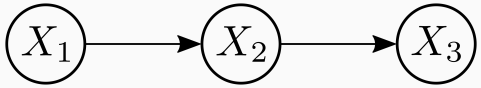
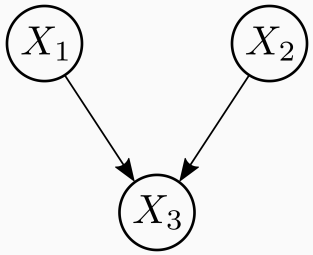
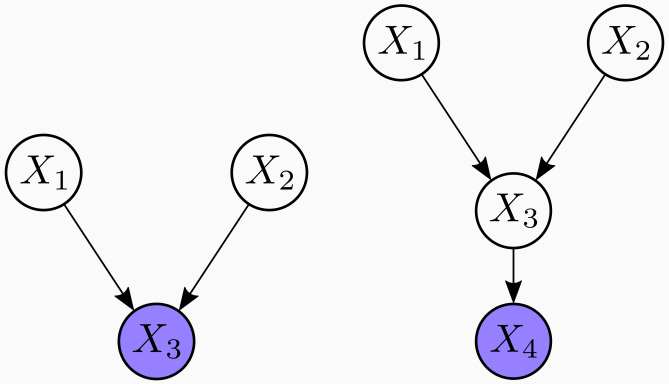
Robert Peharz
Institute of Machine Learning and Neural Computation
Graz University of Technology

A **Bayesian network** (BN) over RVs $\boldsymbol{X}$ is a pair $(\mathcal{G}, \mathcal{P})$, where $\mathcal{G}$ is a **directed acyclic graph (DAG)** that has RVs $\boldsymbol{X}$ as nodes and $\mathcal{P}$ is a collection of distributions $p(x_i \mid \mathbf{pa}_i)$, one for each $X_i \in \boldsymbol{X}$, and conditional on X_i's **parents** in $\mathcal{G}$. The BN represents the joint distribution

$$p(\boldsymbol{x}) = \prod_{i=1}^{D} p(x_i \mid \mathbf{pa}_i)$$



Independencies

blocked	unblocked
fork	
	
chain	
	
collider	
	

d-separation

Let $\mathcal{G}$ be a DAG over RVs $\boldsymbol{X}$. Consider arbitrary two RVs $X, Y \in \boldsymbol{X}$ and let $\boldsymbol{Z} \subseteq \boldsymbol{X} \setminus \{X, Y\}$. We say that X and Y are **d-separated by $\boldsymbol{Z}$** if **all** paths from X to Y are blocked by $\boldsymbol{Z}$.

Theorem **(Geiger, Verma, and Pearl, 1990)**

Let $(\mathcal{G}, \mathcal{P})$ be a Bayesian network. If X and Y are d-separated by $\boldsymbol{Z}$ in $\mathcal{G}$, then $X \perp\!\!\!\perp Y \mid \boldsymbol{Z}$ in the BN distribution.

Learning Bayesian Networks

Learning Bayesian Networks

A Bayesian network consists of the DAG $\mathcal{G}$ and the CPDs $\mathcal{P}$, describing a **model joint distribution**. This allows us to address learning from two angles:

- learning the **structure**, i.e. the DAG $\mathcal{G}$
- by assuming **parametric** CPDs $p(x_i \mid \mathbf{pa}_i, \boldsymbol{\theta}_i)$, learning the **parameters** $\boldsymbol{\theta} = \{\boldsymbol{\theta}_i\}_{i=1}^{D}$

Learning Parameters

- assume the structure DAG $\mathcal{G}$ fixed for now
- we have an iid dataset $\mathcal{D} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)}\}$
- the model (joint distribution) for parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_i\}_{i=1}^{D}$ is

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \prod_{i=1}^{D} p(x_i \mid \mathbf{pa}_i, \boldsymbol{\theta}_i)$$

- how shall we learn $\boldsymbol{\theta}$?

Learning Parameters

- assume the structure DAG $\mathcal{G}$ fixed for now
- we have an iid dataset $\mathcal{D} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)}\}$
- the model (joint distribution) for parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_i\}_{i=1}^{D}$ is

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \prod_{i=1}^{D} p(x_i \mid \mathbf{pa}_i, \boldsymbol{\theta}_i)$$

- how shall we learn $\boldsymbol{\theta}$?
- **maximum (log-)likelihood** is the first go-to solution

Log-likelihood Decomposes in Bayesian Networks

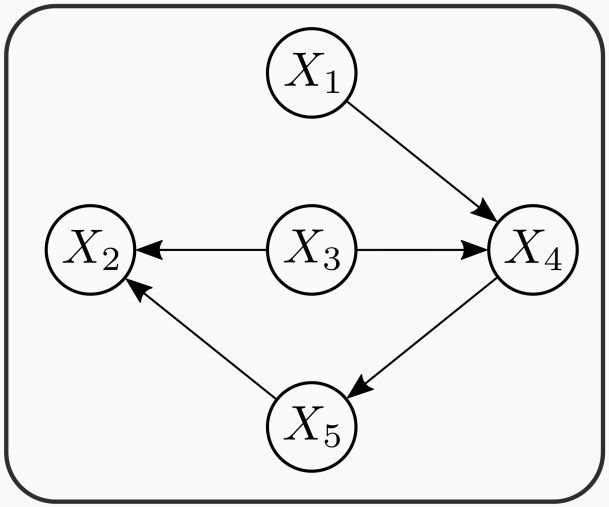
Log-likelihood for BN:

$$\mathcal{L} = \sum_{n=1}^{N} \log \prod_{i=1}^{D} p(x_i^{(n)} \mid \boldsymbol{x}_{\mathbf{pa}_i}^{(n)}, \boldsymbol{\theta}_i) = \sum_{n=1}^{N} \sum_{i=1}^{D} \log p(x_i^{(n)} \mid \boldsymbol{x}_{\mathbf{pa}_i}^{(n)}, \boldsymbol{\theta}_i)$$

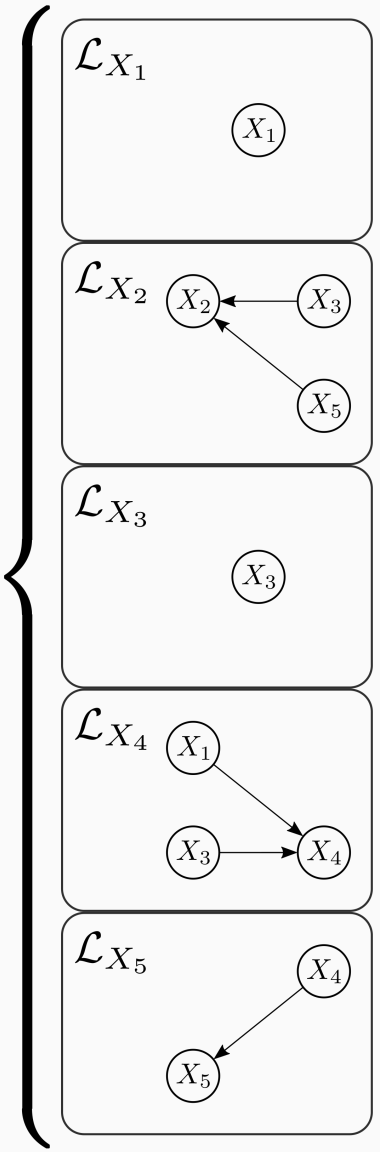
$$= \sum_{i=1}^{D} \underbrace{\sum_{n=1}^{N} \log p(x_i^{(n)} \mid \boldsymbol{x}_{\mathbf{pa}_i}^{(n)}, \boldsymbol{\theta}_i)}_{\mathcal{L}_i\text{: local log-likelihood for } X_i}$$

- $\mathcal{L}_i = \sum_{n=1}^{N} \log p(x_i^{(n)} \mid \boldsymbol{x}_{\mathbf{pa}_i}^{(n)}, \boldsymbol{\theta}_i)$ — just involves CPD for X_i
- since $\mathcal{L}_i$ depends only on $\boldsymbol{\theta}_i$, $\mathcal{L}$ is maximal if each $\mathcal{L}_i$ is maximal individually w.r.t. $\boldsymbol{\theta}_i$
- **independent and local optimization problems!**

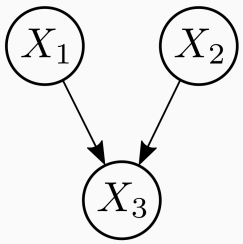
Log-likelihood Decomposes in Bayesian Networks — Example



$$\mathcal{L} = \sum$$



Maximum likelihood parameters for the following Bayesian network over 3 RVs, with 2, 3, and 2 states, respectively :



x_1	x_2	x_3
1	0	0
0	0	1
1	1	1
1	0	0
0	1	1
1	2	0
1	0	0
0	2	0
1	0	1
0	0	0
1	0	1
1	1	0
1	0	0
0	2	0

x_1	$\hat{p}(x_1)$
0	0.357
1	0.643

x_2	$\hat{p}(x_2)$
0	0.572
1	0.214
2	0.214

x_3	$\hat{p}(x_3 \mid x_1, x_2)$	x_1 x_2
0	0.5	0 0
1	0.5	
0	0	0 1
1	1	
0	1	0 2
1	0	
0	0.66	1 0
1	0.33	
0	0.5	1 1
1	0.5	
0	1	1 2
1	0	

E.g. there are 6 samples where $x_1 = 1$ and $x_2 = 0$. Out of these, in 4 samples $x_3 = 0$ and in 2 samples $x_3 = 1$. To avoid division by zero, one often adds a **pseudo-count** $\alpha > 0$ to the actual counts. This is called **Laplace smoothing** (in the example above $\alpha = 0$).

- what if we are dealing with continuous data?
- a simple choice for the CPDs is **linear Gaussian** distributions

$$p(x_i \mid \mathbf{pa}_i) = \mathcal{N}(x \mid \boldsymbol{\alpha}_i^T \mathbf{pa}_i + \beta_i, \sigma_i)$$

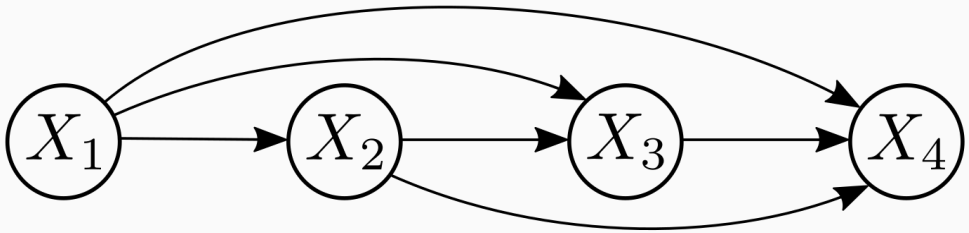
- the mean of X_i is an affine function of the parents' values, with weights $\boldsymbol{\alpha}_i$ and bias β_i; σ_i might be assumed independent of parents' values (**homoscedastic noise model**)
- closed form ML parameters are given as

$$\hat{\boldsymbol{\alpha}}_i = \hat{\mathbf{C}}^{-1}\hat{\mathbf{c}} \quad \hat{\beta}_i = \hat{\mu} - \hat{\boldsymbol{\mu}}^\top \hat{\boldsymbol{\alpha}}_i \quad \hat{\sigma}_i = \sqrt{\left(\sum_{k=1}^{N} x^{(k)} - \hat{\boldsymbol{\alpha}}_i^T \mathbf{pa}_i^{(k)} - \hat{\beta}_i\right)^2}$$

  - $\hat{\mu}$ is the empirical mean of X_i
  - $\hat{\boldsymbol{\mu}}$ is the empirical mean of $\mathbf{pa}_i$
  - $\hat{\mathbf{C}}$ is the empirical covariance matrix of $\boldsymbol{Pa}_i$
  - $\hat{\mathbf{c}}$ is the empirical covariance vector between $\boldsymbol{Pa}_i$ and X_i.

Structure Learning

What is a "suitable" BN structure? Can't we simply take a fully connected graph?



Recall, a fully connected BN corresponds to some chain rule:

$$p(x_1, x_2, x_3, x_4) = p(x_4 \mid x_1, x_2, x_3)\, p(x_3 \mid x_2, x_1)\, p(x_2 \mid x_1)\, p(x_1)$$

It surely contains the true data distribution p^*! But, is this a good idea?

$\boldsymbol{x}$	$p(\boldsymbol{x})$
$(0, 0, \ldots, 0, 0)$	π_1
$(0, 0, \ldots, 0, 1)$	π_2
$(0, 0, \ldots, 1, 0)$	π_3
$\ldots$	$\ldots$
$(1, 1, \ldots, 1, 1)$	$\pi_{\mathbf{1099511627776}}$

- **fully connected BN equivalent to the unconstrained joint**
- infeasible to **store**
- infeasible to **learn**
- even if feasible: prone to overfitting
- infeasible to **perform inference**

Approaches to Structure Learning

Structure Learning via CI-Tests

- structure leads to conditional independencies (CIs) in the joint
- conversely, testing for CI using data-driven approaches allows us to find connections and partially directions

Structure Learning as Score Maximization

- there exist several data-driven score functions, such that structure learning is phrased as discrete optimization problem:

$$\mathcal{G}^* = \arg\max_{\mathcal{G}} \quad \text{score}(\mathcal{G}; \mathcal{D})$$

Generally, structure learning is NP-hard, and the approaches above have worst-case exponential runtime. Furthermore, the DAG is usually not completely identifiable—the direction of several (or even all) arrows might not be determined.

Parameters:

- log-likelihood in BNs decomposes into families for each $X_i \in \boldsymbol{X}$
- thus, overall MLE problem decomposes in many independent small ML problems
- MLE parameters for two special cases of CPDs with closed form solution:
  - categorical distributions
  - linear Gaussian

Structure:

- structure can be learned as well—NP-hard problem in general
- we just scratched the tip of the iceberg—there is much more to learning BNs (e.g. Bayesian approaches)

Inference

Remember: Inference is hard! (curse of dimensionality)

$\boldsymbol{x}$	$p(\boldsymbol{x})$
$(0, 0, \ldots, 0, 0)$	π_1
$(0, 0, \ldots, 0, 1)$	π_2
$(0, 0, \ldots, 1, 0)$	π_3
$\ldots$	$\ldots$
$(1, 1, \ldots, 1, 1)$	$\pi_{\mathbf{1099511627776}}$

A Simple Inference Algorithm

6

Inference by Variable Elimination

We present in this chapter one of the simplest methods for general inference in Bayesian networks, which is based on the principle of variable elimination: A process by which we successively remove variables from a Bayesian network while maintaining its ability to answer queries of interest.

Adnan Darwiche, Modeling and Reasoning with Bayesian Networks, 2009.

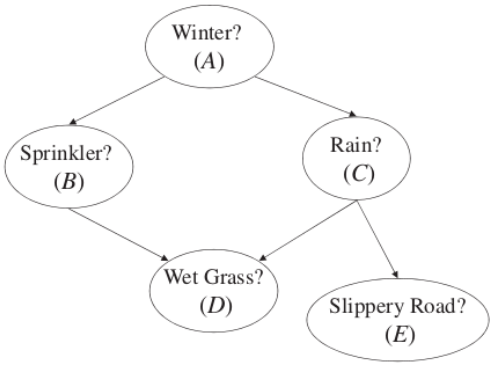
Exploit Structure for Inference

- sparse Bayesian network structure $\Rightarrow$ compact representation
- structure also helps with inference
- **variable elimination (VE)**: one of the simplest exact algorithms
- many other algorithms exist, e.g. **junction tree algorithm**
- however, these are essentially just variations of VE
- if you understand VE, you basically also understand junction trees

Factors

Definition 6.1. A *factor* f over variables $\mathbf{X}$ is a function that maps each instantiation $\mathbf{x}$ of variables $\mathbf{X}$ to a non-negative number, denoted $f(\mathbf{x})$.[1] ■

We will use vars(f) to denote the variables over which the factor f is defined. We will also write $f(X_1, \ldots, X_n)$ to indicate that $X_1, \ldots, X_n$ are the variables over which factor f is defined. Finally, we will allow factors over an empty set of variables. Such factors are called *trivial* as they assign a single number to the trivial instantiation $\top$.



A	Θ_A
true	.6
false	.4

A	B	$\Theta_{B\|A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25

A	C	$\Theta_{C\|A}$
true	true	.8
true	false	.2
false	true	.1
false	false	.9

B	C	D	$\Theta_{D\|BC}$
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

C	E	$\Theta_{E\|C}$
true	true	.7
true	false	.3
false	true	0
false	false	1

Figure 6.1: A Bayesian network.

- in BNs, factors are CPDs
- however, variable elimination can be applied to other graphical models as well, such as **Markov networks** and **factor graphs**
- here, we restrict to BNs

Inference = Crunching Factors

Recall the two core inference routines:

- **marginalization**

$$p(\boldsymbol{y}) = \int p(\boldsymbol{x}, \boldsymbol{y}) \, \mathrm{d}\boldsymbol{x} \qquad \text{(sum for discrete)}$$

- **conditioning**

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{p(\boldsymbol{y}, \boldsymbol{x})}{p(\boldsymbol{x})}$$

Conditioning uses a marginal as a sub-routine. Hence the computational hard part is marginalization!

Two operations are enough to perform marginalization in BNs:

- **factor summation**
- **factor multiplication**

Definition 6.2. Let f be a factor over variables $\mathbf{X}$ and let X be a variable in $\mathbf{X}$. The result of *summing out* variable X from factor f is another factor over variables $\mathbf{Y} = \mathbf{X} \setminus \{X\}$, which is denoted by $\sum_X f$ and defined as

$$\left(\sum_X f\right)(\mathbf{y}) \stackrel{def}{=} \sum_x f(x, \mathbf{y}). \quad \blacksquare$$

B	C	D	f_{BCD}
true	true	true	0.95
true	true	false	0.05
true	false	true	0.9
true	false	false	0.1
false	true	true	0.8
false	true	false	0.2
false	false	true	0
false	false	false	1

Please compute $f_{BC} = \sum_D f_{BCD}$ **and** $f_{BD} = \sum_C f_{BCD}$

B	C	D	f_{BCD}
true	true	true	0.95
true	true	false	0.05
true	false	true	0.9
true	false	false	0.1
false	true	true	0.8
false	true	false	0.2
false	false	true	0
false	false	false	1

B	C	f_{BC}
true	true	1
true	false	1
false	true	1
false	false	1

B	D	f_{BD}
true	true	1.85
true	false	0.15
false	true	0.8
false	false	1.2

Definition 6.3. The result of *multiplying* factors $f_1(\mathbf{X})$ and $f_2(\mathbf{Y})$ is another factor over variables $\mathbf{Z} = \mathbf{X} \cup \mathbf{Y}$, which is denoted by $f_1 f_2$ and defined as

$$(f_1 f_2)(\mathbf{z}) \stackrel{def}{=} f_1(\mathbf{x}) f_2(\mathbf{y}),$$

where $\mathbf{x}$ and $\mathbf{y}$ are compatible with $\mathbf{z}$; that is, $\mathbf{x} \sim \mathbf{z}$ and $\mathbf{y} \sim \mathbf{z}$. ■

A	C	f_{AC}
true	true	0.8
true	false	0.2
false	true	0.1
false	false	0.9

C	E	f_{CE}
true	true	0.7
true	false	0.3
false	true	0
false	false	1

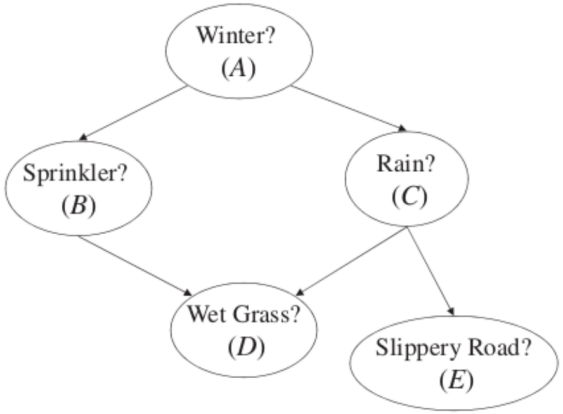
Please compute $f_{ACE} = f_{AC} f_{CE}$

A	C	f_{AC}
true	true	0.8
true	false	0.2
false	true	0.1
false	false	0.9

C	E	f_{CE}
true	true	0.7
true	false	0.3
false	true	0
false	false	1

A	C	E	f_{ACE}
true	true	true	0.56
true	true	false	0.24
true	false	true	0
true	false	false	0.2
false	true	true	0.07
false	true	false	0.03
false	false	true	0
false	false	false	0.9

Joint — Just a Big Factor Multiplication



A	Θ_A
true	.6
false	.4

A	B	$\Theta_{B\|A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25

A	C	$\Theta_{C\|A}$
true	true	.8
true	false	.2
false	true	.1
false	false	.9

B	C	D	$\Theta_{D\|BC}$
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

C	E	$\Theta_{E\|C}$
true	true	.7
true	false	.3
false	true	0
false	false	1

Figure 6.1: A Bayesian network.

$$p(\boldsymbol{x}) = \prod_{i=1}^{D} p(x_i \mid \boldsymbol{pa}_i)$$

A	B	C	D	E	Pr(.)
true	true	true	true	true	.06384
true	true	true	true	false	.02736
true	true	true	false	true	.00336
true	true	true	false	false	.00144
true	true	false	true	true	0
true	true	false	true	false	.02160
true	true	false	false	true	0
true	true	false	false	false	.00240
true	false	true	true	true	.21504
true	false	true	true	false	.09216
true	false	true	false	true	.05376
true	false	true	false	false	.02304
true	false	false	true	true	0
true	false	false	true	false	0
true	false	false	false	true	0
true	false	false	false	false	.09600
false	true	true	true	true	.01995
false	true	true	true	false	.00855
false	true	true	false	true	.00105
false	true	true	false	false	.00045
false	true	false	true	true	0
false	true	false	true	false	.24300
false	true	false	false	true	0
false	true	false	false	false	.02700
false	false	true	true	true	.00560
false	false	true	true	false	.00240
false	false	true	false	true	.00140
false	false	true	false	false	.00060
false	false	false	true	true	0
false	false	false	true	false	0
false	false	false	false	true	0
false	false	false	false	false	.0900

Variable Elimination

Algorithm 3 VE_PR1($\mathcal{N}$, $\mathbf{Q}$, π)

input:

$\mathcal{N}$: Bayesian network

$\mathbf{Q}$: variables in network $\mathcal{N}$

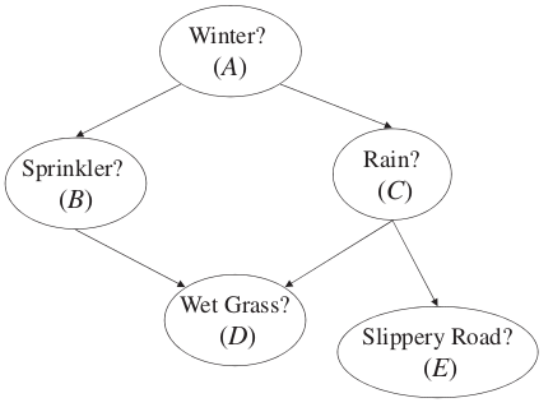
π: ordering of network variables not in $\mathbf{Q}$

output: the prior marginal Pr($\mathbf{Q}$)

main:

1: $\mathcal{S} \leftarrow$ CPTs of network $\mathcal{N}$
2: **for** $i = 1$ to length of order π **do**
3: $f \leftarrow \prod_k f_k$, where f_k belongs to $\mathcal{S}$ and mentions variable $\pi(i)$
4: $f_i \leftarrow \sum_{\pi(i)} f$
5: replace all factors f_k in $\mathcal{S}$ by factor f_i
6: **end for**
7: **return** $\prod_{f \in \mathcal{S}} f$

Simple algorithm, efficiency depends on variable order π



A	Θ_A
true	.6
false	.4

A	B	$\Theta_{B\|A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25

A	C	$\Theta_{C\|A}$
true	true	.8
true	false	.2
false	true	.1
false	false	.9

B	C	D	$\Theta_{D\|BC}$
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

C	E	$\Theta_{E\|C}$
true	true	.7
true	false	.3
false	true	0
false	false	1

Figure 6.1: A Bayesian network.

Let's compute $p(B, C)$ with variable order $\pi = E \prec D \prec A$

$$E \prec D \prec A$$

A	f_A
true	0.6
false	0.4

A	B	f_{AB}
true	true	0.2
true	false	0.8
false	true	0.75
false	false	0.25

A	C	f_{AC}
true	true	0.8
true	false	0.2
false	true	0.1
false	false	0.9

C	E	f_{CE}
true	true	0.7
true	false	0.3
false	true	0
false	false	1

B	C	D	f_{BCD}
true	true	true	0.95
true	true	false	0.05
true	false	true	0.9
true	false	false	0.1
false	true	true	0.8
false	true	false	0.2
false	false	true	0
false	false	false	1

Compute $\sum_E \prod_k f_k = \sum_E f_{CE} =: f_C$

$$E \prec D \prec A$$

A	f_A
true	0.6
false	0.4

A	B	f_{AB}
true	true	0.2
true	false	0.8
false	true	0.75
false	false	0.25

A	C	f_{AC}
true	true	0.8
true	false	0.2
false	true	0.1
false	false	0.9

C	E	f_{CE}
true	true	0.7
true	false	0.3
false	true	0
false	false	1

B	C	D	f_{BCD}
true	true	true	0.95
true	true	false	0.05
true	false	true	0.9
true	false	false	0.1
false	true	true	0.8
false	true	false	0.2
false	false	true	0
false	false	false	1

C	f_C
true	1
false	1

$$E \prec D \prec A$$

A	f_A
true	0.6
false	0.4

A	B	f_{AB}
true	true	0.2
true	false	0.8
false	true	0.75
false	false	0.25

A	C	f_{AC}
true	true	0.8
true	false	0.2
false	true	0.1
false	false	0.9

C	E	f_{CE}
true	true	0.7
true	false	0.3
false	true	0
false	false	1

B	C	D	f_{BCD}
true	true	true	0.95
true	true	false	0.05
true	false	true	0.9
true	false	false	0.1
false	true	true	0.8
false	true	false	0.2
false	false	true	0
false	false	false	1

C	f_C
true	1
false	1

$$E \prec D \prec A$$

A	f_A
true	0.6
false	0.4

A	B	f_{AB}
true	true	0.2
true	false	0.8
false	true	0.75
false	false	0.25

A	C	f_{AC}
true	true	0.8
true	false	0.2
false	true	0.1
false	false	0.9

C	f_C
true	1
false	1

B	C	D	f_{BCD}
true	true	true	0.95
true	true	false	0.05
true	false	true	0.9
true	false	false	0.1
false	true	true	0.8
false	true	false	0.2
false	false	true	0
false	false	false	1

$$E \prec D \prec A$$

A	f_A
true	0.6
false	0.4

A	B	f_{AB}
true	true	0.2
true	false	0.8
false	true	0.75
false	false	0.25

A	C	f_{AC}
true	true	0.8
true	false	0.2
false	true	0.1
false	false	0.9

C	f_C
true	1
false	1

B	C	D	f_{BCD}
true	true	true	0.95
true	true	false	0.05
true	false	true	0.9
true	false	false	0.1
false	true	true	0.8
false	true	false	0.2
false	false	true	0
false	false	false	1

Compute $\sum_D \prod_k f_k = \sum_D f_{BCD} =: f_{BC}$

$$E \prec D \prec A$$

A	f_A
true	0.6
false	0.4

A	B	f_{AB}
true	true	0.2
true	false	0.8
false	true	0.75
false	false	0.25

A	C	f_{AC}
true	true	0.8
true	false	0.2
false	true	0.1
false	false	0.9

C	f_C
true	1
false	1

B	C	D	f_{BCD}
true	true	true	0.95
true	true	false	0.05
true	false	true	0.9
true	false	false	0.1
false	true	true	0.8
false	true	false	0.2
false	false	true	0
false	false	false	1

B	C	f_{BC}
true	true	1
true	false	1
false	true	1
false	false	1

$$E \prec D \prec A$$

A	f_A
true	0.6
false	0.4

A	B	f_{AB}
true	true	0.2
true	false	0.8
false	true	0.75
false	false	0.25

A	C	f_{AC}
true	true	0.8
true	false	0.2
false	true	0.1
false	false	0.9

C	f_C
true	1
false	1

B	C	D	f_{BCD}
true	true	true	0.95
true	true	false	0.05
true	false	true	0.9
true	false	false	0.1
false	true	true	0.8
false	true	false	0.2
false	false	true	0
false	false	false	1

B	C	f_{BC}
true	true	1
true	false	1
false	true	1
false	false	1

$$E \prec D \prec A$$

A	f_A
true	0.6
false	0.4

A	B	f_{AB}
true	true	0.2
true	false	0.8
false	true	0.75
false	false	0.25

A	C	f_{AC}
true	true	0.8
true	false	0.2
false	true	0.1
false	false	0.9

C	f_C
true	1
false	1

B	C	f_{BC}
true	true	1
true	false	1
false	true	1
false	false	1

$$E \prec D \prec A$$

A	f_A
true	0.6
false	0.4

A	B	f_{AB}
true	true	0.2
true	false	0.8
false	true	0.75
false	false	0.25

A	C	f_{AC}
true	true	0.8
true	false	0.2
false	true	0.1
false	false	0.9

C	f_C
true	1
false	1

B	C	f_{BC}
true	true	1
true	false	1
false	true	1
false	false	1

Compute $\sum_A \prod_k f_k = \sum_A f_A f_{AB} f_{AC} =: f'_{BC}$

$$E \prec D \prec A$$

A	f_A
true	0.6
false	0.4

A	B	f_{AB}
true	true	0.2
true	false	0.8
false	true	0.75
false	false	0.25

A	C	f_{AC}
true	true	0.8
true	false	0.2
false	true	0.1
false	false	0.9

C	f_C
true	1
false	1

B	C	f_{BC}
true	true	1
true	false	1
false	true	1
false	false	1

A	B	C	$f_A f_{AB} f_{AC}$
true	true	true	0.096
true	true	false	0.024
true	false	true	0.384
true	false	false	0.096
false	true	true	0.03
false	true	false	0.27
false	false	true	0.01
false	false	false	0.09

$\Rightarrow$

B	C	f'_{BC}
true	true	0.126
true	false	0.294
false	true	0.394
false	false	0.186

Variable Elimination

Example

$$E \prec D \prec A$$

A	f_A
true	0.6
false	0.4

A	B	f_{AB}
true	true	0.2
true	false	0.8
false	true	0.75
false	false	0.25

A	C	f_{AC}
true	true	0.8
true	false	0.2
false	true	0.1
false	false	0.9

C	f_C
true	1
false	1

B	C	f_{BC}
true	true	1
true	false	1
false	true	1
false	false	1

A	B	C	$f_A f_{AB} f_{AC}$
true	true	true	0.096
true	true	false	0.024
true	false	true	0.384
true	false	false	0.096
false	true	true	0.03
false	true	false	0.27
false	false	true	0.01
false	false	false	0.09

$\Rightarrow$

B	C	f'_{BC}
true	true	0.126
true	false	0.294
false	true	0.394
false	false	0.186

$$E \prec D \prec A$$

B	C	f'_{BC}
true	true	0.126
true	false	0.294
false	true	0.394
false	false	0.186

C	f_C
true	1
false	1

B	C	f_{BC}
true	true	1
true	false	1
false	true	1
false	false	1

$E \prec D \prec A$

B	C	f'_{BC}
true	true	0.126
true	false	0.294
false	true	0.394
false	false	0.186

C	f_C
true	1
false	1

B	C	f_{BC}
true	true	1
true	false	1
false	true	1
false	false	1

$p(B, C)$ is the product of the remaining factors

Complexity of VE

Theorem 6.2. *If the largest factor constructed on Line 4 of Algorithm 3,* `VE_PR1`*, has w variables, the complexity of Lines 3–5 is then $O(n \exp(w))$, where n is the number of variables in the Bayesian network.* ■

Algorithm 3 `VE_PR1`$(\mathcal{N}, \mathbf{Q}, \pi)$

---

input:

$\mathcal{N}$: Bayesian network
$\mathbf{Q}$: variables in network $\mathcal{N}$
π: ordering of network variables not in $\mathbf{Q}$

output: the prior marginal $\Pr(\mathbf{Q})$

main:

1: $\mathcal{S} \leftarrow$ CPTs of network $\mathcal{N}$
2: **for** $i = 1$ to length of order π **do**
3: $f \leftarrow \prod_k f_k$, where f_k belongs to $\mathcal{S}$ and mentions variable $\pi(i)$
4: $f_i \leftarrow \sum_{\pi(i)} f$
5: replace all factors f_k in $\mathcal{S}$ by factor f_i
6: **end for**
7: **return** $\prod_{f \in \mathcal{S}} f$

Complexity of VE con't

- complexity of VE depends only on variable ordering
- at least exponential in the so-called **tree-width** of the graph
- there exists some optimal variable ordering for VE, such that complexity becomes optimal
- bad news: finding the optimal ordering is NP-hard
- good news: there are good heuristics, e.g. **MinFill**, **MinDegree** (see Darwiche)

Ancestral Sampling

How to sample from a Bayesian Network?

Ancestral Sampling

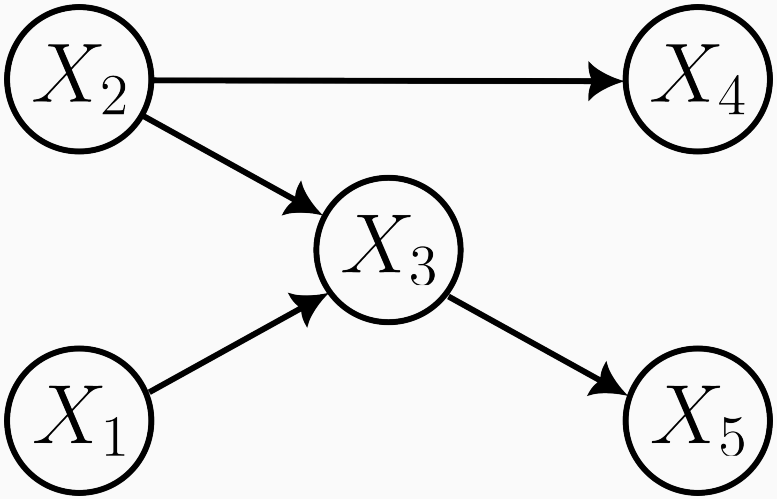
How to sample from a Bayesian Network?

Ancestral Sampling

- let $\boldsymbol{X}$ be set of RVs in the BN
- let $\boldsymbol{x} = \emptyset$
- repeat until $\boldsymbol{X}$ is empty:
  - for all $X \subseteq \boldsymbol{X}$ without parents in $\boldsymbol{X}$:
    - sample $x \sim p(x \mid \mathbf{pa}_X)$ where $\mathbf{pa}_X \subseteq \boldsymbol{x}$
    - include x in $\boldsymbol{x}$
    - remove X from $\boldsymbol{X}$

Ancestral Sampling

Example



x_1	$p(x_1)$
0	0.9
1	0.1

x_2	$p(x_2)$
0	0.3
1	0.7

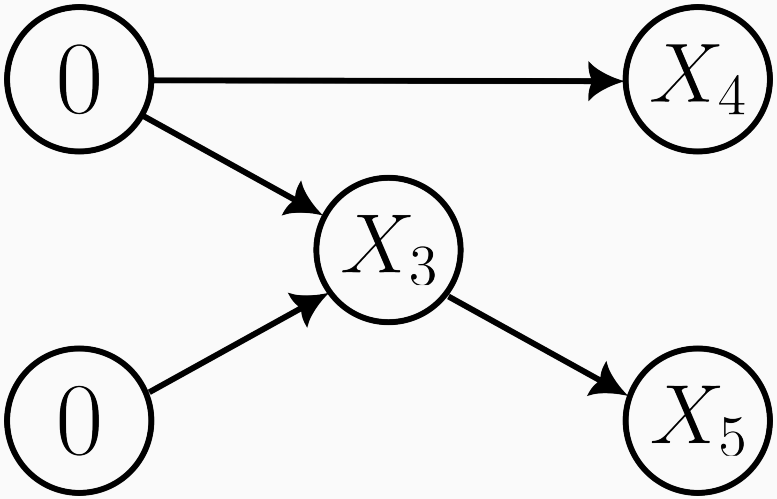
x_3	$p(x_3 \mid x_1, x_2)$	x_1	x_2
0	0.1	0	0
1	0.9	0	0
0	0.8	0	1
1	0.2	0	1
0	0.5	1	0
1	0.5	1	0
0	0.3	1	1
1	0.7	1	1

x_4	$p(x_4 \mid x_2)$	x_2
0	0.2	0
1	0.8	0
0	0.5	1
1	0.5	1

x_5	$p(x_5 \mid x_3)$	x_3
0	0.4	0
1	0.6	0
0	0.8	1
1	0.2	1

Ancestral Sampling

Example



x_1	$p(x_1)$
0	**0.9**
1	**0.1**

x_2	$p(x_2)$
0	**0.3**
1	**0.7**

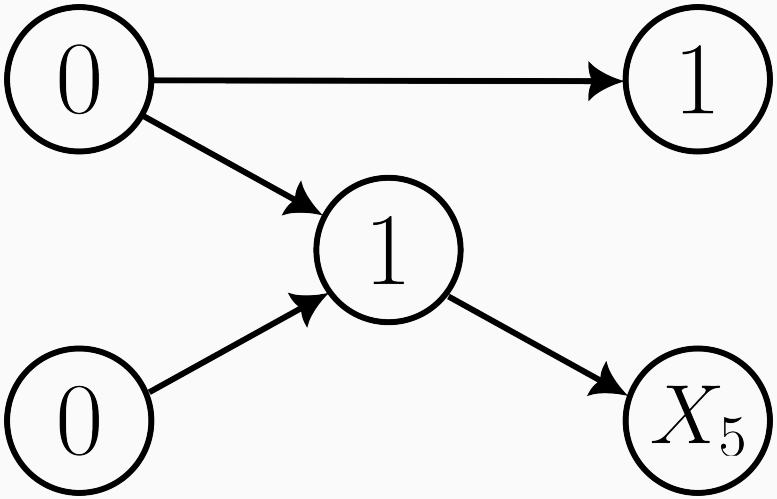
x_3	$p(x_3 \mid x_1, x_2)$	x_1	x_2
0	0.1	0	0
1	0.9	0	0
0	0.8	0	1
1	0.2	0	1
0	0.5	1	0
1	0.5	1	0
0	0.3	1	1
1	0.7	1	1

x_4	$p(x_4 \mid x_2)$	x_2
0	0.2	0
1	0.8	0
0	0.5	1
1	0.5	1

x_5	$p(x_5 \mid x_3)$	x_3
0	0.4	0
1	0.6	0
0	0.8	1
1	0.2	1

Ancestral Sampling

Example



x_1	$p(x_1)$
0	0.9
1	0.1

x_2	$p(x_2)$
0	0.3
1	0.7

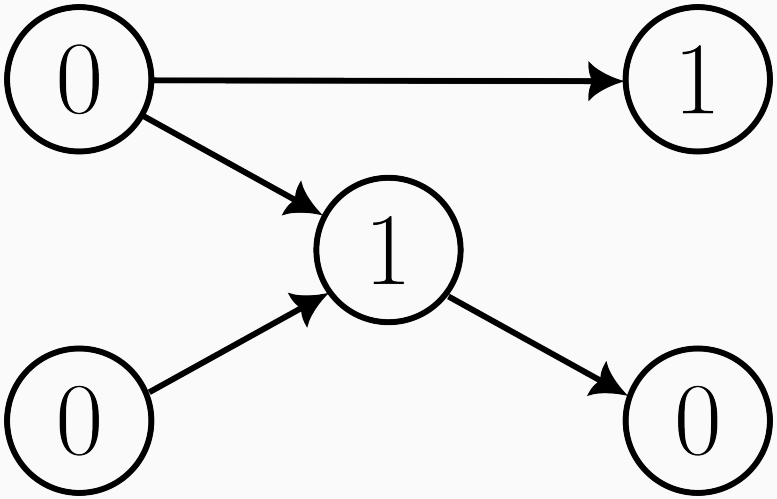
x_3	$p(x_3 \mid x_1, x_2)$	x_1	x_2
0	**0.1**	**0**	**0**
1	**0.9**	**0**	**0**
0	0.8	0	1
1	0.2	0	1
0	0.5	1	0
1	0.5	1	0
0	0.3	1	1
1	0.7	1	1

x_4	$p(x_4 \mid x_2)$	x_2
0	**0.2**	**0**
1	**0.8**	**0**
0	0.5	1
1	0.5	1

x_5	$p(x_5 \mid x_3)$	x_3
0	0.4	0
1	0.6	0
0	0.8	1
1	0.2	1

Ancestral Sampling

Example



x_1	$p(x_1)$
0	0.9
1	0.1

x_2	$p(x_2)$
0	0.3
1	0.7

x_3	$p(x_3 \mid x_1, x_2)$	x_1	x_2
0	0.1	0	0
1	0.9	0	0
0	0.8	0	1
1	0.2	0	1
0	0.5	1	0
1	0.5	1	0
0	0.3	1	1
1	0.7	1	1

x_4	$p(x_4 \mid x_2)$	x_2
0	0.2	0
1	0.8	0
0	0.5	1
1	0.5	1

x_5	$p(x_5 \mid x_3)$	x_3
0	0.4	0
1	0.6	0
0	**0.8**	**1**
1	**0.2**	**1**

Bayesian Networks as Lingua Franca

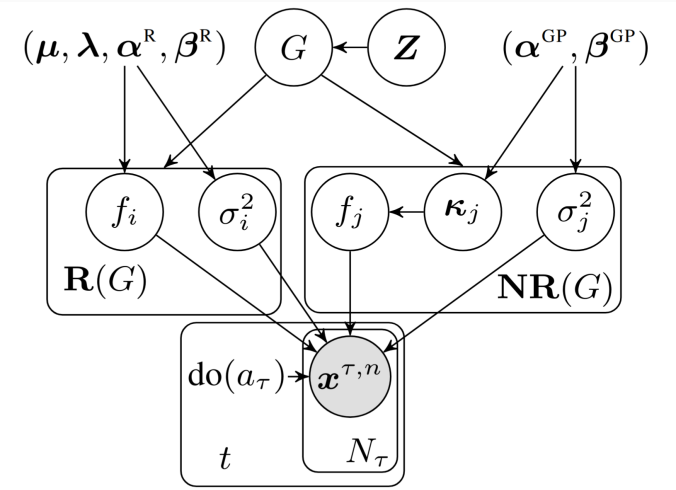


Figure 2: Graphical model of GP-DiBS-ABCI.

Toth et al., NeurIPS, 2022.

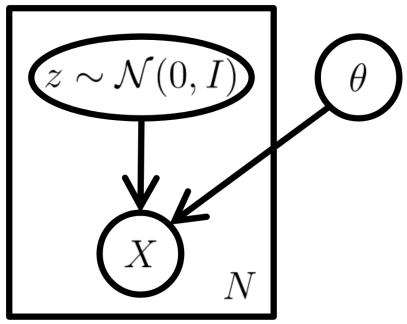


Figure 1: The standard VAE model represented as a graphical model. Note the conspicuous lack of any structure or even an "encoder" pathway: it is possible to sample from the model without any input. Here, the rectangle is "plate notation" meaning that we can sample from z and X N times while the model parameters θ remain fixed.

Doersch, `https://arxiv.org/pdf/1606.05908`, 2016.

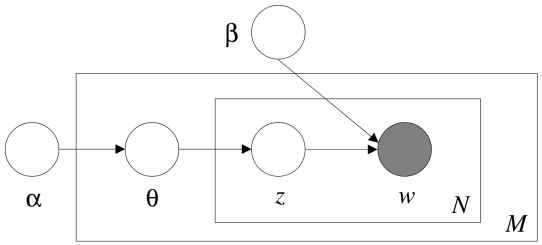
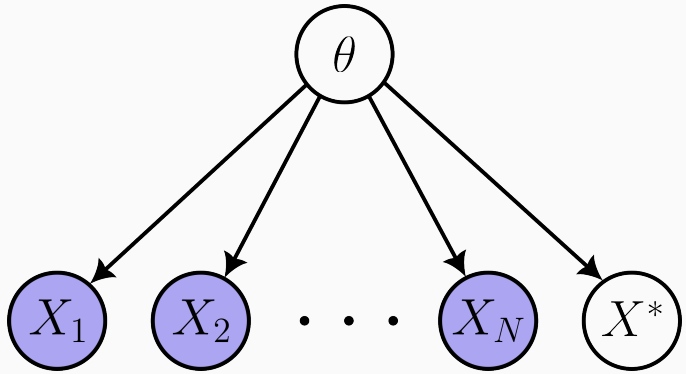


Figure 1: Graphical model representation of LDA. The boxes are "plates" representing replicates. The outer plate represents documents, while the inner plate represents the repeated choice of topics and words within a document.

Blei et al., JMLR, 2003.

Bayesian Bernoulli Model as Bayesian Network

- $\theta \sim \text{Beta}(\alpha, \beta)$
- $X_i \sim \text{Bernoulli}(\theta)$ (train data)
- $X^* \sim \text{Bernoulli}(\theta)$ (test data)



We computed the posterior:

$$p(\theta \mid x_1, \ldots, x_N) \propto p(\theta) \prod_i p(x_i \mid \theta)$$

And the Bayes predictive:

$$p(x^* \mid x_1, \ldots, x_N) = \int p(x^* \mid \theta)\, p(\theta \mid x_1, \ldots, x_N)\, \mathrm{d}\theta$$

These could also be implemented as VE on the BN above.

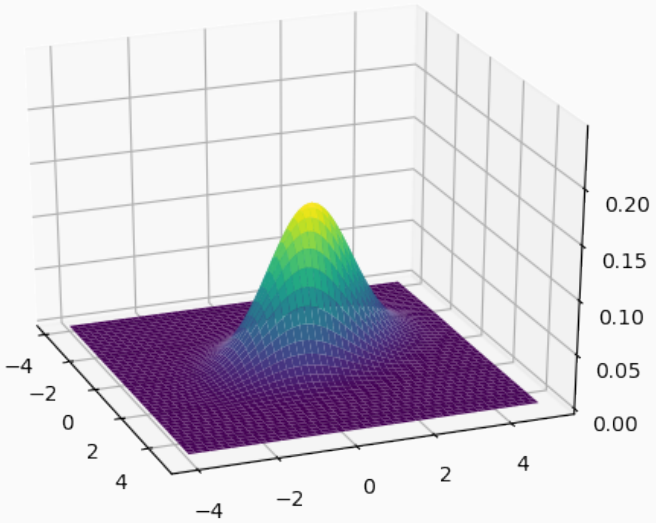
Gaussian Mixture Models

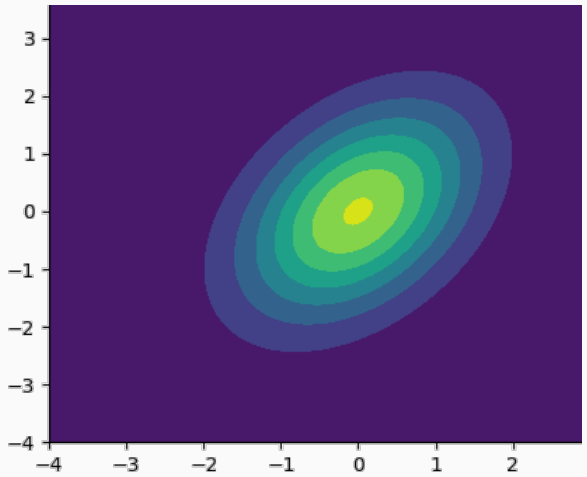
Why Gaussian Mixture Models?

Gaussian mixture models (GMMs) are simple yet expressive models for joint distributions. Many (advanced) topics such as **latent variables** are introduced in GMMs and can be studied there. They also allow a wide range of tractable inference routines.

$$p_{\boldsymbol{X}}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^D \, |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}$$

- D-dimensional **mean vector** $\boldsymbol{\mu}$
- $D \times D$ **positive definite covariance matrix** $\boldsymbol{\Sigma}$

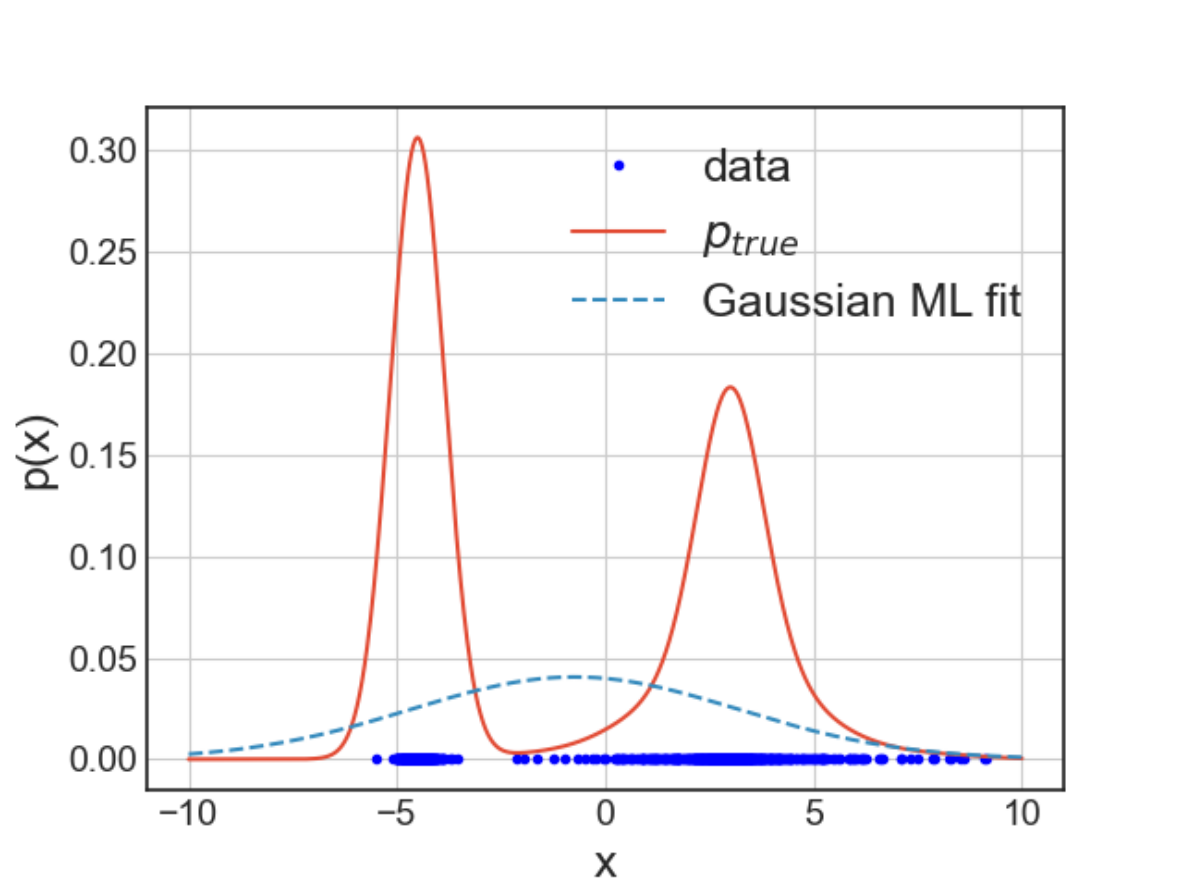




A Gaussian (Mis)fit

Example

Data is usually not Gaussian. Hence, using a Gaussian assumption might lead to severe misfit, in particular for **multimodal** distributions (**mode**: local maximum of density).



Let $K \geq 1$ and

- $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$ be K mean parameters
- $\Sigma_1, \Sigma_2, \ldots, \Sigma_K$ be K covariance matrices
- $w_1, w_2, \ldots, w_K$ be K **mixture weights** with $w_k \geq 0$ and $\sum_k w_k = 1$

The **Gaussian mixture model (GMM)** parametrized by $\boldsymbol{\theta} = \{w_k, \boldsymbol{\mu}_k, \Sigma_k\}_{k=1}^K$ is given as

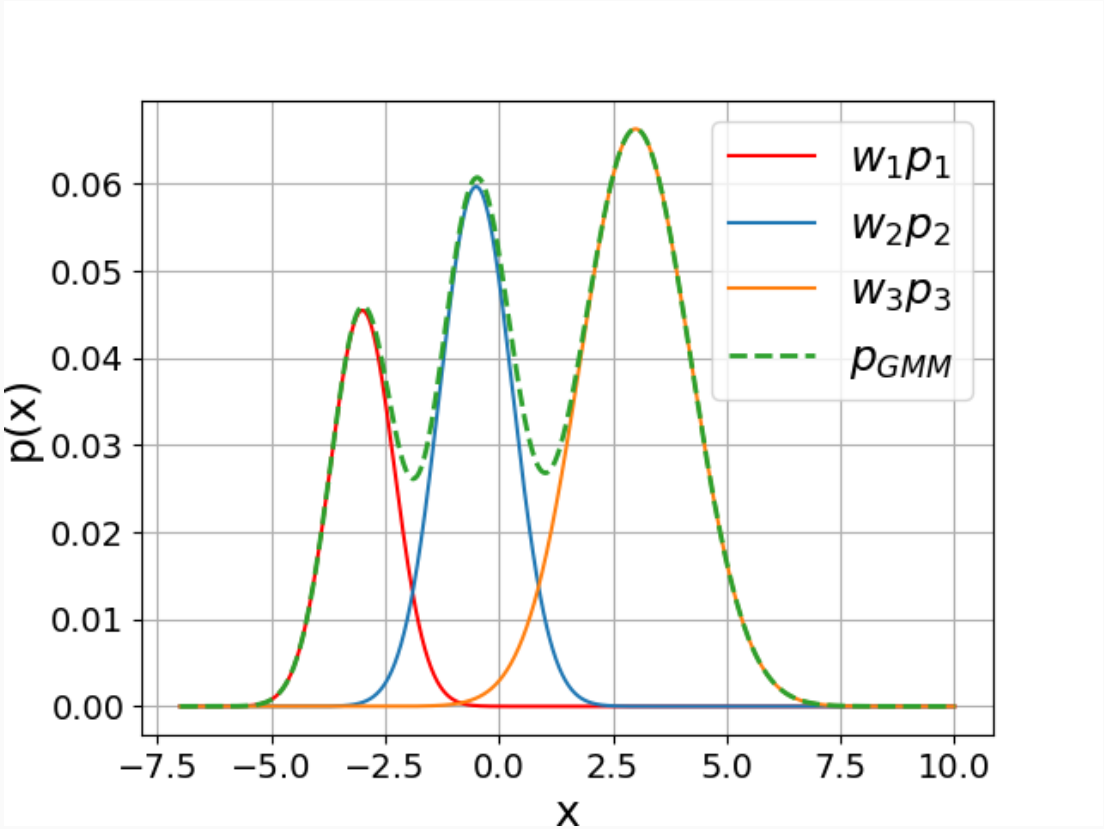
$$p(\boldsymbol{x} \mid \boldsymbol{\theta}) = \sum_{k=1}^{K} w_k \, p(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \Sigma_k)$$

where $p(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \Sigma_k)$ is the Gaussian density with parameters $\boldsymbol{\mu}_k, \Sigma_k$. Hence a GMM is a **convex combination*** of K Gaussian densities, often called **Gaussian components** in this context.

*A linear combination with non-negative weights which sum up to one is called a convex combination.

Gaussian Mixture Model, 1 Dimension

Example



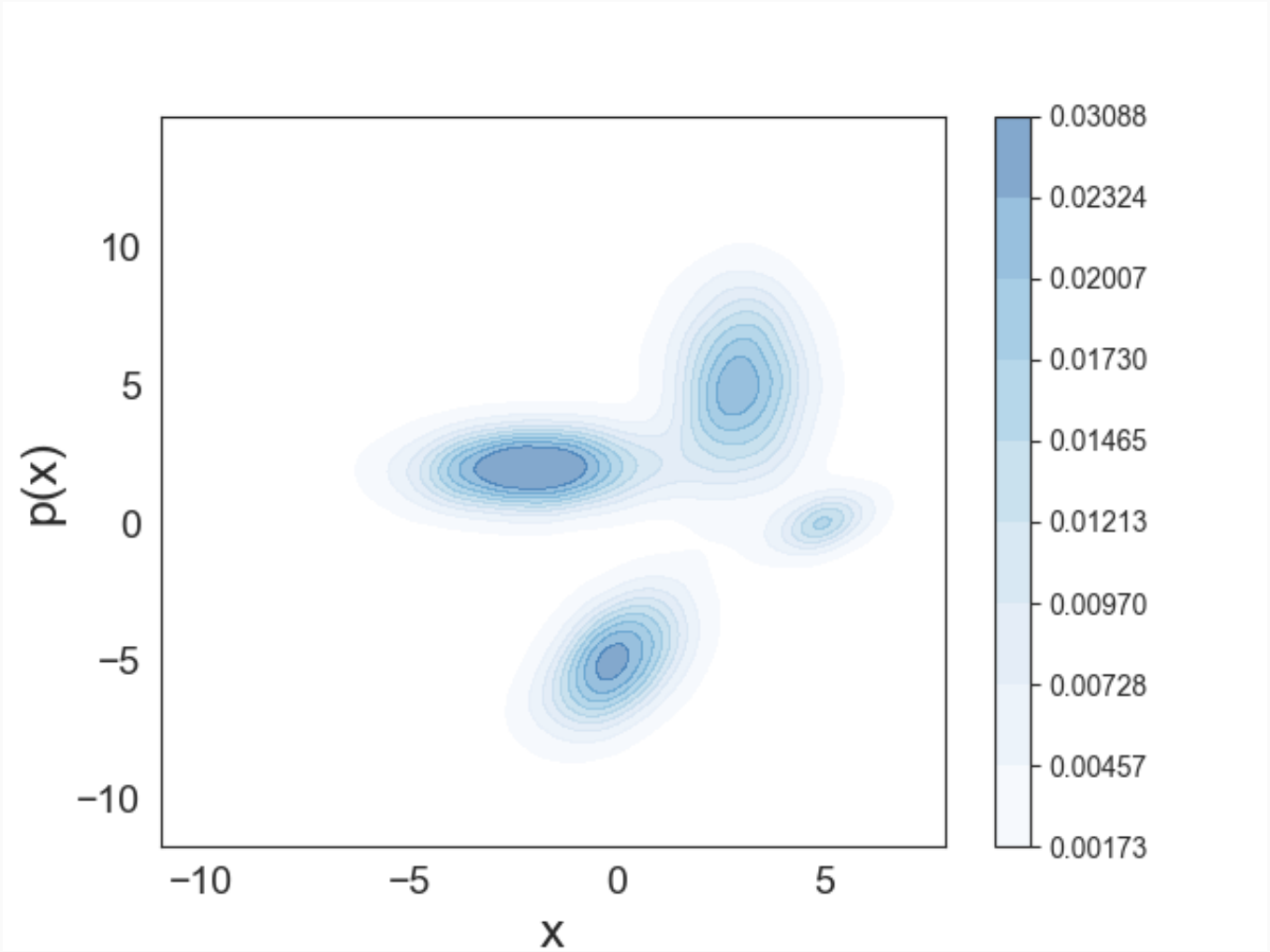
3 components:

$w_1 = 0.2, w_2 = 0.3, w_3 = 0.5,$

$\mu_1 = -3, \mu_2 = -0.5, \mu_3 = 3,$

$\sigma_1 = 0.7, \sigma_2 = 0.8, \sigma_3 = 1.2$

A GMM with 4 components in 2 dimensions:



Properness of GMMs

Any GMM specifies a proper probability density, since

- $p(\boldsymbol{x} \mid \boldsymbol{\theta})$ is **non-negative**:
  - $w_k \geq 0$ and
  - $p(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \Sigma_k) \geq 0$
  - Hence, $p(\boldsymbol{x} \mid \boldsymbol{\theta}) = \sum_{k=1}^{K} w_k \, p(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \Sigma_k) \geq 0$
- $p(\boldsymbol{x} \mid \boldsymbol{\theta})$ is **normalized**:

$$\int p(\boldsymbol{x} \mid \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{x} = \int \sum_{k=1}^{K} w_k \, p(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \Sigma_k) \, \mathrm{d}\boldsymbol{x}$$

$$= \sum_{k=1}^{K} w_k \int p(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \Sigma_k) \, \mathrm{d}\boldsymbol{x} = \sum_{k=1}^{K} w_k \, 1 = 1$$

Note: nowhere we used the fact that the components are Gaussian. Hence we can mix **any** densities, also different ones!