

Probabilistic Graphical Models

Summary

Fabian Damken

February 21, 2022



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Contents

1	Introduction	8
1.1	Examples	8
1.2	Fundamental Questions	8
2	Foundations	9
2.1	Probability Theory	9
2.1.1	(Conditional) Independence	9
2.1.2	Inference	10
2.1.3	Potentials	10
2.2	Machine Learning	10
2.2.1	(Document) Classification	10
3	Bayesian Networks	12
3.1	The Naive Bayes Model	12
3.1.1	Maximum Likelihood Parameter Estimation	13
3.1.2	Application	13
3.2	Definition and Independence Assumptions	13
3.2.1	“Explaining Away” / Berkson’s Paradox	14
3.2.2	Representation Theorem	14
3.2.3	“Adding Edges Does Not Hurt”	15
3.3	Encoded Independencies	15
3.3.1	Dependency Structures	15
3.3.2	d-Separation	16
3.3.3	Context-Specific Independence (CSI)	17
3.3.4	The Bayes’ Ball Algorithm	18
3.4	SOTA Modal	18
4	Inference	19
4.1	Chain Models	19
4.2	Variable Elimination	20
4.3	Abductive Inference	21
4.4	Complexity of Conditional Queries	21
4.5	Variable Elimination in Moral Graphs	23
4.5.1	Complexity	23
5	Markov Random Fields	27
5.1	Bayesian Networks as MRFs	27
5.2	Triangulated Graphs and Simplicial Nodes	29
5.3	Join Trees	29
5.3.1	Markov Network as a Join Tree: Junction Tree	29

5.4	Junction Trees	31
5.4.1	Inference	31
6	Learning	33
6.1	Parameter Estimation	33
6.1.1	Known Structure, Complete Data	33
6.1.2	Known Structure, Incomplete Data (Expectation-Maximization)	34
6.1.3	Gradient Ascent	35
6.1.4	Bayesian Parameter Estimation	35
6.1.5	Summary	36
6.2	Structure Learning / Model Selection	36
6.2.1	Minimal I-Maps and Perfect Maps (P-Maps)	36
6.2.2	Perfect Maps (P-Maps) and I-Equivalence	37
6.2.3	Obtaining a P-Map	37
6.2.4	Accurate Structures	37
6.2.5	Learning	37
6.2.6	Structure Search as Optimization	37
6.2.7	Structural EM	37
6.2.8	Summary	37
7	Dynamic Bayesian Networks	38
7.1	Hidden Markov Models	39
7.2	Inference	39
7.2.1	Decoding	39
7.2.2	Best State Sequence	39
7.2.3	Parameter Estimation	39
7.3	State Estimation (Kalman Filter)	39
7.3.1	Recursive Bayesian Updating	39
7.3.2	(Modeling) Actions	39
7.3.3	Bayes Filter	39
7.3.4	Discrete-Time Kalman Filter	39
7.4	General Dynamic Bayesian Networks	39
7.4.1	Exact Inference	39
7.4.2	Tractable, Approximate Inference	39
8	Approximate Inference	40
8.1	Message Passing	40
8.1.1	Sum-Product Belief Propagation	40
8.1.2	(Acyclic) Belief Propagation as Dynamic Programming	40
8.1.3	Loopy Belief Propagation	40
8.2	Sampling	40
8.2.1	Forward Sampling (Without Evidence)	40
8.2.2	Forward Sampling (With Evidence)	40
8.2.3	Gibbs Sampling	40
8.2.4	Likelihood Weighting	40
9	Tractable Probabilistic Models	41
9.1	Deep Learning	41

9.2	Probabilistic Circuits	41
9.3	Sum-Product Networks	41
9.3.1	Inference	41
9.3.2	Learning	41
9.3.3	Inference on Devices	41
10	Deep Generative Models	42
10.1	Likelihood-Based	42
10.1.1	Autoregressive Generative Models	42
10.1.2	Variational Auto-Encoders	42
10.1.3	Normalizing Flows	42
10.2	Likelihood-Free	42
10.2.1	Generative Adversarial Networks	42
10.3	Applications in Scientific Discovery	42

List of Figures

2.1	Comparison of CPT and Potential	11
3.1	Naive Bayes Bayesian Network	12
3.2	A Bayesian Network Captures More than Local Independencies	15
3.3	Illustration of Independency Capturing	16
4.1	Example for Moralizing	23
4.2	Example for Elimination in Moral Graphs	23
4.3	Example for a Perfect Elimination Sequence	24
4.4	Induced Graph for Imperfect Elimination Ordering	25
5.1	Join Tree Illustration	30
5.2	Example of a Junction Tree	31
5.3	Messages Boxes used in Junction Tree Message Passing	32



List of Tables

4.1	Summary of Inference Complexities	26
6.1	Overview Over Different Learning Problems	33

List of Algorithms

1	Variable Elimination	20
2	Most Probable Explanations via Variable Elimination	22
3	Triangulated, Undirected Graph \rightarrow Join Tree	30



1 Introduction

1.1 Examples

1.2 Fundamental Questions

2 Foundations

This chapter covers fundamental concepts of probability theory and machine learning that are required for the later chapters. Note that not all relevant concepts of probability theory are covered.

2.1 Probability Theory

This section covers some very important concepts of probability theory, however, one should already be familiar with some basics like probability measures, density functions, joint distributions, marginalization, etc.¹

One note on notation: whenever a sum represents a marginalization over some random variable X , it is written as

$$P(Y) = \sum_X^{\text{marg.}} P(X, Y) := \sum_{x \in \text{val}(X)} P(X = x, Y)$$

for brevity.

2.1.1 (Conditional) Independence

The most important concept leveraged in probabilistic graphical models is (conditional) independence of random variables. Two random variables X and Y are *statistically independent* if knowing either does not change the belief/probability of the other, i.e.,

$$P(X | Y) = P(X) \quad \text{and} \quad P(Y | X) = P(Y).$$

This is equivalent to the definition of independence, $P(X, Y) = P(X) P(Y)$. Independence is denoted $X \perp Y$ and is a symmetric property. A milder property is *conditional* independence, i.e., two random variables X and Y are independent if Z is given:

$$P(X | Y, Z) = P(X | Z) \quad \text{and} \quad P(Y | X, Z) = P(Y | Z).$$

Again, this property can be written as $P(X, Y | Z) = P(X | Z) P(Y | Z)$ by the chain rule. Conditional independency is denoted $X \perp Y | Z$.

The following properties hold and can be useful for some proofs later on:

$$\begin{array}{llll} X \perp Y | Z & \iff & Y \perp X | Z & \text{(Symmetry)} \\ X \perp (Y, W) | Z & \implies & (X \perp Y | Z) \wedge (X \perp W | Z) & \text{(Decomposition)} \\ X \perp (Y, W) | Z & \implies & X \perp Y | (Z, W) & \text{(Weak Union)} \\ (X \perp W | (Y, Z)) \wedge (X \perp Y | Z) & \implies & X \perp (Y, W) | Z & \text{(Contraction)} \\ (X \perp Y | (W, Z)) \wedge (X \perp W | (Y, Z)) & \implies & X \perp (Y, W) | Z & \text{(Intersection)} \end{array}$$

Note that the intersection property only holds for positive distributions (i.e., $P(X = x) > 0$ for all x).

¹Take a look the chapter of statistics fundamentals of <https://fabian.damken.net/summaries/cs/elective/vc/statml/statml-summary.pdf>.

Monty Hall Problem

2.1.2 Inference

Information Theory

Information theory is trying to quantify how much information is encoded in some distribution $P(X)$. The central measure is *entropy*:

$$H_P(X) = \mathbb{E}[\log(1/P(X))] = \sum_{x \in \text{val}(X)} P(X) \log \frac{1}{P(X)} = - \sum_{x \in \text{val}(X)} P(X) \log P(X)$$

If the logarithm is of base two, the entropy encodes how much bits are required *on average* to encode X when X follows the distribution $P(X)$. Similarly, *conditional entropy* can be defined as

$$H_P(X | Y) = \mathbb{E}[\log(1/P(X | Y))] = H_P(X, Y) - H_P(X)$$

where $H_P(X, Y)$ is the joint entropy over X and Y . Like for probabilities, a *chain rule of entropies* is derivable:

$$H_P(X, Y, Z) = H_P(X) + H_P(Y | X) + H_P(Z | X, Y).$$

To quantify (in)dependency between two variables X and Y , the *mutual information*

$$I_P(X; Y) = H_P(X) - H_P(X | Y)$$

can be used. This quantity is symmetric and is zero if and only if X and Y are independent.

2.1.3 Potentials

A *potential* is an alternative way of representing (conditional) probabilities aside from conditional probability tables (CPTs). A potential $\phi_{X,Y,Z}$ is a function that maps each configuration $(x, y, z) \in \text{val}(X) \times \text{val}(Y) \times \text{val}(Z)$ to a non-negative real number. The set of random variables targeted by a potential is its *domain*, i.e., $\text{dom } \phi_{X,Y,Z} = \{X, Y, Z\}$. Note that a (conditional) probability distribution is a special case of potentials where the potential is normalized. Vice versa, a potential can always be normalized into a CPT. This is illustrated in Figure 2.1.

Similar to CPTs, potentials can be multiplied by pairing up the entries and marginalized by summing up the corresponding entries. Compared to CPTs, it is not necessary to normalize a potential into a probability distribution afterwards, easing some calculations.

Potentials will come in handy later on when covering inference in junction trees (section 5.4).

2.2 Machine Learning

2.2.1 (Document) Classification

$P(X = x \mid Y = y, Z = z) \mid x = \mathfrak{f} \quad x = \mathfrak{t}$				<table> <tr> <th>X</th> <th>Y</th> <th>Z</th> <th>$\phi_{X,Y,Z}$</th> <th>$\tilde{\phi}_{X,Y,Z}$</th> </tr> <tr><td>\mathfrak{t}</td><td>\mathfrak{t}</td><td>\mathfrak{t}</td><td>0.8</td><td>8</td></tr> <tr><td>\mathfrak{t}</td><td>\mathfrak{t}</td><td>\mathfrak{f}</td><td>0.2</td><td>2</td></tr> <tr><td>\mathfrak{t}</td><td>\mathfrak{f}</td><td>\mathfrak{t}</td><td>0.5</td><td>5</td></tr> <tr><td>\mathfrak{t}</td><td>\mathfrak{f}</td><td>\mathfrak{f}</td><td>0.5</td><td>5</td></tr> <tr><td>\mathfrak{f}</td><td>\mathfrak{t}</td><td>\mathfrak{t}</td><td>0.2</td><td>2</td></tr> <tr><td>\mathfrak{f}</td><td>\mathfrak{t}</td><td>\mathfrak{f}</td><td>0.8</td><td>8</td></tr> <tr><td>\mathfrak{f}</td><td>\mathfrak{f}</td><td>\mathfrak{t}</td><td>0.7</td><td>7</td></tr> <tr><td>\mathfrak{f}</td><td>\mathfrak{f}</td><td>\mathfrak{f}</td><td>0.3</td><td>3</td></tr> </table>	X	Y	Z	$\phi_{X,Y,Z}$	$\tilde{\phi}_{X,Y,Z}$	\mathfrak{t}	\mathfrak{t}	\mathfrak{t}	0.8	8	\mathfrak{t}	\mathfrak{t}	\mathfrak{f}	0.2	2	\mathfrak{t}	\mathfrak{f}	\mathfrak{t}	0.5	5	\mathfrak{t}	\mathfrak{f}	\mathfrak{f}	0.5	5	\mathfrak{f}	\mathfrak{t}	\mathfrak{t}	0.2	2	\mathfrak{f}	\mathfrak{t}	\mathfrak{f}	0.8	8	\mathfrak{f}	\mathfrak{f}	\mathfrak{t}	0.7	7	\mathfrak{f}	\mathfrak{f}	\mathfrak{f}	0.3	3
X	Y	Z	$\phi_{X,Y,Z}$	$\tilde{\phi}_{X,Y,Z}$																																													
\mathfrak{t}	\mathfrak{t}	\mathfrak{t}	0.8	8																																													
\mathfrak{t}	\mathfrak{t}	\mathfrak{f}	0.2	2																																													
\mathfrak{t}	\mathfrak{f}	\mathfrak{t}	0.5	5																																													
\mathfrak{t}	\mathfrak{f}	\mathfrak{f}	0.5	5																																													
\mathfrak{f}	\mathfrak{t}	\mathfrak{t}	0.2	2																																													
\mathfrak{f}	\mathfrak{t}	\mathfrak{f}	0.8	8																																													
\mathfrak{f}	\mathfrak{f}	\mathfrak{t}	0.7	7																																													
\mathfrak{f}	\mathfrak{f}	\mathfrak{f}	0.3	3																																													
$(Y, Z) = (\mathfrak{t}, \mathfrak{t})$	0.8	0.2	\longleftrightarrow																																														
$(Y, Z) = (\mathfrak{t}, \mathfrak{f})$	0.5	0.5																																															
$(Y, Z) = (\mathfrak{t}, \mathfrak{t})$	0.2	0.8																																															
$(Y, Z) = (\mathfrak{t}, \mathfrak{f})$	0.7	0.3																																															

Figure 2.1: Comparison of a CPT (left) and the corresponding potential (right). The rightmost column in the potential $\tilde{\phi}$ is equivalent to ϕ as it can be normalized accordingly.

3 Bayesian Networks

Bayesian networks provide a compact representation for exponentially-large distribution by exploiting conditional independencies. When representing a joint distribution $P(X_1, X_2, \dots, X_n)$ with $|\text{val}(X_i)| = k$ for simplicity, the CPT has $k^n - 1$ entries¹! But lots of information is redundant in the joint if some of the variables exhibit independencies. Assume, for instance, that all subsets $\mathcal{X}, \mathcal{Y} \subseteq \{X_1, X_2, \dots, X_n\}$ are independent ($\mathcal{X} \perp \mathcal{Y}$). Then the joint can be written as

$$P(X_1, X_2, \dots, X_n) = \prod_{i=1}^n P(X_i),$$

requiring only $n(k - 1) \in \mathcal{O}(n)$ parameters! This is linear in the number of random variables! While this assumption seems rather simplistic, it is used with success in practice: this is the assumption of the naive Bayes classifier covered in section 3.1.

3.1 The Naive Bayes Model

In the *naive Bayes model*, it is assumed that some feature random variables $\{X_1, X_2, \dots, X_n\}$ are all conditionally independent given the class C ; $\forall \mathcal{X}, \mathcal{Y} \subseteq \{X_1, X_2, \dots, X_n\} : \mathcal{X} \perp \mathcal{Y} | C$. The joint distribution is therefore simply

$$P(X_1, X_2, \dots, X_n, C) = P(C) \prod_{i=1}^n P(X_i | C). \quad (3.1)$$

Using Bayesian networks, the independencies are represented using a graph as shown in Figure 3.1.

To classify a new instance \mathbf{x} (a vector composed of the individual instances of X_1, X_2, \dots, X_n), the class $c \in \text{val}(C)$ with the highest posterior probability is selected:

$$c_{\text{MAP}} = \arg \max_{c \in \text{val}(C)} P(c | \mathbf{x}) = \arg \max_{c \in \text{val}(C)} \frac{P(\mathbf{x} | c) P(c)}{P(\mathbf{x})} = \arg \max_{c \in \text{val}(C)} P(\mathbf{x} | c) P(c) = \arg \max_{c \in \text{val}(C)} P(C) \prod_{i=1}^n P(X_i | C).$$

¹The -1 comes from the requirement of the probability being normalized, hence the probability of the “last” configuration is implicit.

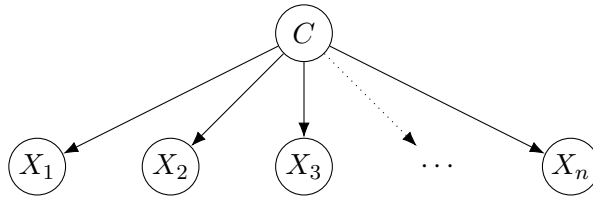


Figure 3.1: Bayesian network for the naive Bayes classifier.

3.1.1 Maximum Likelihood Parameter Estimation

To learn the probabilities required to perform classification using the naive Bayes model, the straightforward approach is to use maximum likelihood estimates, i.e., the frequencies in the data:

$$\hat{P}(C = c_j) = \frac{N(C = c_j)}{N} \quad \hat{P}(X_i = x_{ik} | c_j) = \frac{N(X_i = x_{ik}, C = c_j)}{N(C = c_j)}.$$

here, $N(C = c_j)$ is the number of data points where the class is c_j and $N(X_i = x_{ik}, C = c_j)$ is the number of data points with class c_j and value x_{ik} in the i -th feature. Using the identity function $\mathbb{1}[\psi]$ that is one iff ψ is true and zero otherwise, these quantities can be expressed as

$$N(C = c_j) = \sum_{((x_1, x_2, \dots, x_n), c) \in \mathcal{D}} \mathbb{1}[c = c_j] \quad N(X_i = x_{ik}, C = c_j) = \sum_{((x_1, x_2, \dots, x_n), c) \in \mathcal{D}} \mathbb{1}[x_i = x_{ik}, c = c_j]$$

where $\mathcal{D} \subseteq \text{val}(X_1) \times \text{val}(X_2) \times \dots \times \text{val}(X_n) \times \text{val}(C)$ is the data set. Note that the formality of the data set and values is quite rigorous in this section. From now on, the notation will be abused from time to time for brevity.

This estimation method poses a major challenge: if no instances have been observed of a given feature/class-configuration, the evidence is zero and no matter how high other evidence is in the joint (3.1), the joint will be zero. This problem can be reduced by *smoothing* the distributions to avoid overfitting,

$$\hat{P}(X_i = x_{ik} | c_j) = \frac{N(X_i = x_{ik}, C = c_j) + m(N(X_i = x_{ik})/N)}{N(C = c_j) + m},$$

where m controls the extend of smoothing and is a hyper-parameter.

3.1.2 Application

Even though the assumption of all features being conditionally independent is often false, the naive Bayes classifier turns out to be quite effective in practice. While not being the best classification method, it usually serves as a good and dependable baseline. If the assumption holds, the Bayes classifier is optimal (and can be tuned for different misclassification costs, for example).

Also, it is very fast: learning is done in a single pass over the data (by counting) and testing is linear in the number of attributes and data size. For the same reason (small CPTs), it requires very little storage, making it suitable for on-device classification.

3.2 Definition and Independence Assumptions

After some introductory treatment of the naive Bayes classifier and its associated Bayesian network, this section now focuses on a more rigorous treatment of the independencies that are actually encoded in a Bayesian network and a formal definition of what a Bayesian network is.

The key idea for Bayesian networks is the incorporation of independence assumptions into their structure. A formal Bayesian network consists of the following components:

- Set of random variables $\{X_1, X_2, \dots, X_n\}$.
- Directed acyclic graph (DAG) encoding the (in-) dependencies.

- Conditional probability table for each random variable, $P(X_i | \text{Pa}(X_i))$. $\text{Pa}(X_i)$ contains all parents of the random variable in the given DAG.

Then, the joint distribution is given by

$$P(X_1, X_2, \dots, X_n) = \prod_{i=1}^n P(X_i | \text{Pa}(X_i)) \quad (3.2)$$

and the local Markov assumption holds. The *local Markov assumption* states that a random variable X_i is independent of all its non-descendants $\text{ND}(X_i)$ given its parents $\text{Pa}(X_i)$:

$$X_i \perp \text{ND}(X_i) | \text{Pa}(X_i).$$

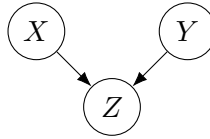
Let $\text{Ch}(X_i)$ be the children of X_i , then X_i 's non-descendants are all random variables that are not in $\text{Des}(X_i)$, X_i 's descendants:

$$\text{ND}(X_i) = \{X_1, X_2, \dots, X_n\} \setminus \text{Des}(X_i), \quad \text{Des}(X_i) = \text{Ch}(X_i) \cup \bigcup_{X_j \in \text{Ch}(X_i)} \text{Des}(X_j).$$

This is a recursive definition for which $\text{Des}(X_i) = \emptyset$ holds iff X_i does not have children.

3.2.1 “Explaining Away” / Berkson’s Paradox

Berkson’s paradox describe the phenomenon that, by the local Markov assumption, for a network like



the independence $X \perp Y$ holds but $X \perp Y | Z$ does not. This implies that information can flow from X to Y iff Z is given, i.e., having evidence on either X or Y is already sufficient to explain the evidence on Z :

$$P(X = x | Z = z, Y = y) \leq P(X = x | Z = z).$$

3.2.2 Representation Theorem

All this work on independencies raise the question of whether it is actually worth it: what distributions can be represented using a Bayesian network? And what networks are required to represent a distribution? Also, what other independencies apart from the local Markov assumption are encoded in a network? Some independencies can certainly be derived using the relations presented in subsection 2.1.1. Let P be the real distribution and let $I(P)$ be the independencies encoded in the true distribution. Similarly, let $I_\ell(G)$ be the local independencies (induced by the local Markov assumption) encoded in a graph G . Then the key assumption of Bayesian networks is that

$$I_\ell(G) \subseteq I(P).$$

That is, the local independencies only capture such that are actual present in the true distribution. If this relation holds, G is said to be an *I-map* (independency map) of P .

Theorem The *representation theorem* states that G is an I-map of P if and only if P factorizes according to G via (3.2).

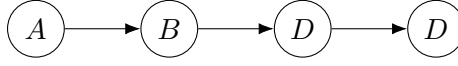


Figure 3.2: Illustration of a Bayesian network capturing more independencies than induced by the local Markov assumption. In this BN, the local Markov assumption encodes $I_\ell(G) = \{B \perp A \mid A, C \perp (A, B) \mid B, D \perp (A, B, C) \mid C\}$. However, also $A \perp D \mid C$ is a captured independency, for example.

Proof The proof is split into proving “if” and “only if”.

“if”:

“only if”: Assume, w.l.o.g., a topological ordering X_1, X_2, \dots, X_n where $j < i$ for all children X_j of some random variable X_i . Applying the chain rule to the joint yields $P(X_1, X_2, \dots, X_n) = \prod_{i=1}^n P(X_i \mid X_1, X_2, \dots, X_{i-1})$. By the topological ordering, $\text{Pa}(X_i) \subseteq \{X_1, X_2, \dots, X_{i-1}\}$ and hence, using the local Markov assumption, $P(X_i \mid X_1, X_2, \dots, X_{i-1}) = P(X_i \mid \text{Pa}(X_i))$. Therefore, P factorized according to (3.2).

3.2.3 “Adding Edges Does Not Hurt”

Theorem Let G be an I-map for P . Then any DAG G' having the same directed edges as G , then G' is also an I-map of P .

From this theorem it follows that

- G' is strictly more expressive than G by capturing more independencies, and
- adding edges to G still results in an I-map.

Proof Idea Let $I_\ell(G) \subseteq I(P)$. To show that $I_\ell(G') \subseteq I(P)$ holds, it is enough to show $I_\ell(G') \subsetneq I_\ell(G)$. Note that is enough to show that this property holds if a single edge is added as adding two edges can be understood as two modifications, resulting in a third graph G'' . It therefore follows from $I_\ell(G'') \subsetneq I_\ell(G') \subseteq I_\ell(G)$ that also $I_\ell(G'') \subsetneq I_\ell(G)$.

Let X and Y be some random variables of G such that $Y \notin \text{Pa}_G(X)$. Then adding an edge $Y \rightarrow X$ only removes local independencies, but does not induce new ones.

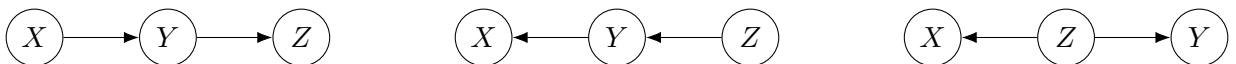
3.3 Encoded Independencies

So far, only local independencies have been considered. However, a Bayesian network encodes even more dependencies by applying the algebra of conditional independencies (subsection 2.1.1; see Figure 3.2 for an example).

The findings of this section are summarized in Figure 3.3 which gives an overview of the different variants to getting the set of independencies (read this section first before inspecting the figure).

3.3.1 Dependency Structures

In a three-node BN, four types of structures are possible. The first, second, and third are indirect causal and evidential effects and a common cause, respectively:



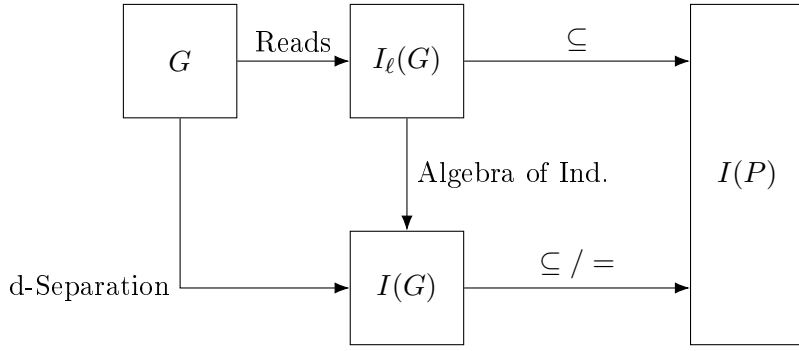
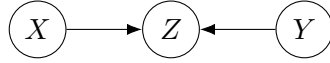


Figure 3.3: Illustration of different methods to capture the independencies of a distribution. Note that $I(G)$ and $I(P)$ are equal (i.e., P is faithful to G) for almost all P that factor over G .

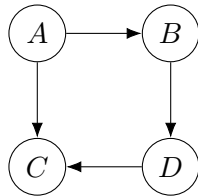
In both cases, the conditional independence $X \perp Y \mid Z$ holds, but *not* the simple independence $X \perp Y$. The last and most special structure is a v-structure, where two variables have a common effect:



In this case, the independencies are inverted: the simple independence $X \perp Y$ holds, but $X \perp Y \mid Z$ *does not*. This is the same effect as the “explain away” phenomenon / Berkson’s paradox (subsection 3.2.1).

3.3.2 d-Separation

d-Separation (dependence separation) is a principled way of finding all independencies modeled by a Bayesian network. The main component are trails. A *trail* is a non-cyclic path $X'_1 \leftrightarrow X'_2 \leftrightarrow \dots \leftrightarrow X'_k$ from one random variable X'_1 to another X'_k . Note that a trail can also be along an edge in opposite order (i.e., the trail itself is undirected) and that every edge and node can only be visited once. For example, the network



has the following trails:

- $A \rightarrow B, A \rightarrow C \leftarrow D \leftarrow B; A \rightarrow C, A \rightarrow B \rightarrow D \rightarrow C; A \rightarrow B \rightarrow D, A \rightarrow C \leftarrow D$
- $B \leftarrow A, B \rightarrow D \rightarrow C \leftarrow A; B \leftarrow A \rightarrow C, B \rightarrow D \rightarrow C; B \rightarrow D, B \leftarrow A \rightarrow C \leftarrow D$
- $C \leftarrow A, C \leftarrow D \leftarrow B \leftarrow A; C \leftarrow A \rightarrow B, C \leftarrow D \leftarrow B; C \leftarrow D, C \leftarrow A \rightarrow B \rightarrow D$
- $D \leftarrow B \leftarrow A, D \rightarrow C \leftarrow A; D \leftarrow B, D \rightarrow C \leftarrow A \rightarrow B; D \rightarrow C, D \leftarrow B \leftarrow A \rightarrow C$

Note that as trails are themselves undirected, the graph itself really has half as many trails, but they are kept for consistency.

d-Separation now uses these trails to extract independencies. Consider any trail $X'_1 \leftrightarrow X'_2 \leftrightarrow \dots \leftrightarrow X'_k$ between two variables X'_1 and X'_k this trail is *active* if for each triplet $X'_{i-1} \leftrightarrow X'_i \leftrightarrow X'_{i+1}$, the following holds:

- for $X_{i-1} \rightarrow X_i \rightarrow X_{i+1}$, X_i is not observed.
- for $X_{i-1} \leftarrow X_i \leftarrow X_{i+1}$, X_i is not observed.
- for $X_{i-1} \leftarrow X_i \rightarrow X_{i+1}$, X_i is not observed.
- for $X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$, X_i or one of its descendants is observed.

Theorem Two variables X and Y are independent given a set of variables \mathcal{Z} if there is no active trail between X and Y if \mathcal{Z} are observed. They are said to be *d-separated*. The set of independencies obtained via d-separation is called $I(G)$.

Soundness

Theorem d-Seperation is *sound*. That is, if P factorizes over G , then $I(G) \subseteq I(P)$.
This means that d-separation only captured independencies that the true distributions exhibits.

Proof

Completeness

A distribution P is said to be *faithful* if it does not have independencies that cannot be read from G .

Theorem For almost all distributions P that factorize over G , $I(G) = I(P)$. “Almost all” means that the possible distributions are a set of measure zero parametrizations of the CPTs.

Hence, the Bayesian network is usually sufficient for capturing all independence properties of the distribution! But this only holds for complete independence, but there might be context-specific independencies that are not captured.

Proof

3.3.3 Context-Specific Independence (CSI)

As already said, d-separation only captures complete independencies, i.e., once for which

$$\forall x, \in \text{val}(X), y \in \text{val}(Y), z \in (Z) : (X = x \perp Y = y \mid Z = z)$$

holds but context-specific independencies,

$$\exists x, \in \text{val}(X), y \in \text{val}(Y), z \in (Z) : (X = x \perp Y = y \mid Z = z),$$

are not captured.

One option for representing CSIs are Tree CPDs. *Tree CPDs* encode a distribution $P(X \mid \text{Pa}(X))$ using a decision-tree like structure, where the paths are an assignment of (a subset of) $\text{Pa}(X)$ and leaves represent the distributions given the assignments on the path. This way, representation size can be drastically reduced when CSIs are present.

Another variant where CSIs occur is determinism. While determinism already makes the CPT sparse, it has even greater influences on inference as propagating the zeros often leads to a bunch of new zeros.

3.3.4 The Bayes' Ball Algorithm

The *Bayes' ball algorithm* is a algorithmic approach to applying d-separation.

3.4 SOTA Modal

Current state-of-the-art (SOTA) models are often characterized by richness in their local structures (determinism, CSI), massive sizes (thousand of variables), and high connectivity (treewidth, see ??). These developments are enabled by high-level modeling tools (relational and first-order logic), the overall advancement in machine learning, and new application areas (e.g., bioinformatics and sensor networks).

In these big models, it is a must to exploit local and relational structure!

4 Inference

Inference is concerned with answering statistical queries in a probabilistic model. As this summary is on probabilistic model, it is concerned with inference in graphical models. However, inference in Bayesian networks is—in general—pretty hopeless aka. NP-hard (even approximate inference). In practice, however, special structures can be exploited producing many effective (approximate) inference algorithms. This chapter covers exact inference whereas approximate inference is covered later in chapter 8.

A general probabilistic query is $P(X | e)$ computing some probability given some evidence e . The straightforward way to compute the conditional is to directly use its definition,

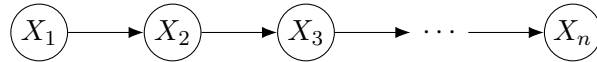
$$P(X | e) = \frac{P(X, e)}{P(e)}.$$

This requires computing the joint distribution and marginalizing out X to get the normalization factor. In general, such a computation has exponential complexity in the number of random variables!

For all complexity arguments in the following sections, let k be the number of configurations of a single random variable, $k = |\text{val}(X)|$, and assume that all random variables have the same number of values.

4.1 Chain Models

The simplest BN to perform efficient exact inference in is a chain model



that factors as

$$P(X_1, X_2, \dots, X_n) = P(X_1) \prod_{i=2}^n P(X_i | X_{i-1}).$$

However, to computing $P(X_n)$ still requires an exponential number of operations, $\mathcal{O}(k^n)$:

$$\begin{aligned} P(X_n) &= \sum_{X_1}^{\text{marg.}} \sum_{X_2}^{\text{marg.}} \cdots \sum_{X_{n-1}}^{\text{marg.}} P(X_1, X_2, \dots, X_{n-1}, X_n) \\ &= \sum_{X_1}^{\text{marg.}} \sum_{X_2}^{\text{marg.}} \cdots \sum_{X_{n-1}}^{\text{marg.}} P(X_1) P(X_2 | X_1) \cdots P(X_n | X_{n-1}). \end{aligned}$$

Instead, it is advisable to reverse the sums and to pull the independent factors out of the marginalization. This corresponds to iteratively computing $P(X_1)$, $P(X_2)$, ..., $P(X_n)$:

$$P(X_{i+1}) = \sum_{X_i}^{\text{marg.}} P(X_{i+1} | X_i) P(X_i).$$

This process is linear in the number of random variables and quadratic in the number of values of a random variable, $\mathcal{O}(nk^2)$! Hence, at least for simple chain models, inference is tractable.

4.2 Variable Elimination

Variable elimination extends the idea introduced in the chain models of pulling factors out of the marginalization sums. To incorporate evidence e , the corresponding RVs are simply set to the respective values and marginalization over these variables is removed. See section 8.2.1 in *Data Mining and Machine Learning*¹ for a more thorough treatment of this topic. By convention, the factor that appears by eliminating a variable X that still depends on other RVs Y_1, Y_2, \dots, Y_k is named $f_X(Y_1, Y_2, \dots, Y_k)$. Pseudocode for variable elimination is shown in algorithm 1.

One of the most important steps in variable elimination is pruning the non-active variables (line 2). This means removing all random variables that are independent of all variables in $\mathbf{X} \cup \mathbf{E}$ as the distributions over these random variables simply reduce to a multiplication with 1, not changing the results. However, this can reduce the number of required operations to a tractable amount.

The most important aspect determining the efficiency of variable elimination is, intuitively, the choice of the elimination ordering (this is covered in more detail in section 4.5). For relatively simple networks like polytrees², inference is linear in *representation* size (i.e., in the number of entries across all CPTs)! In general, however, variable elimination is exponential in the number of variables in the intermediate factors—hence its complexity is dominated by the largest intermediate factor. This again is discussed in more detail in section 4.5.

While variable elimination on a general Bayesian network is still exponential in time w.r.t. the number of variables in the intermediate factors, it is linear (in representation size!) in polytree networks.

Also note that variable elimination can also directly be applied to potentials. It is not required that the original factors actually represent CPTs.

Algorithm 1: Variable Elimination

```
Input: Bayesian network with RVs  $X_1, X_2, \dots, X_n$  and query  $P(\mathbf{X} \mid \mathbf{E} = e)$ 
Output: Probability  $P(\mathbf{X} \mid \mathbf{E} = e)$ 
1 Instantiate evidence  $\mathbf{E} = e$ .
2 Prune non-active variables (w.r.t.  $\mathbf{X} \cup \mathbf{E}$ ), giving  $X_1, X_2, \dots, X_{\tilde{n}}, \tilde{n} \leq n$ .
3 Choose an ordering  $X_1, X_2, \dots, X_{\tilde{n}}$ .
  // Initialize factors:
4  $\mathcal{F}^{(0)} \leftarrow \{f_i = P(X_i \mid \text{Pa}(X_i)) : i = 1, 2, \dots, \tilde{n}\}$ 
5 for  $i = 1, 2, \dots, \tilde{n}$  do
6   if  $X_i \in \mathbf{X} \cup \mathbf{E}$  then
7     skip
8   Collect factors  $f_1, f_2, \dots, f_k \in \mathcal{F}^{(i-1)}$  containing  $X_i$ .
  // Generate a new factor  $g$  by eliminating  $X_i$ :
9    $g = \sum_{X_i}^{\text{marg.}} \prod_{j=1}^k f_j$ 
  // Replace factors in factor set:
10   $\mathcal{F}^{(i)} \leftarrow (\mathcal{F}^{(i-1)} \setminus \{f_1, f_2, \dots, f_k\}) \cup \{g\}$ 
11 Read only remaining factor  $P(\mathbf{X}, \mathbf{E} = e)$  from  $\mathcal{F}^{(\tilde{n})}$ .
12 Normalize  $P(\mathbf{X}, \mathbf{E} = e)$  to get  $P(\mathbf{X} \mid \mathbf{E} = e)$ .
```

¹<https://fabian.damken.net/summaries/cs/elective/iws/dmml/dmml-summary.pdf>

²A polytree is the directed variant of a tree. That is, a DAG is a polytree if its underlying undirected graph is a tree.

4.3 Abductive Inference

So far, only the calculation of a-posterior beliefs/probabilities has been covered. That is, computing queries of the form $P(\mathbf{X} \mid \mathbf{E} = e)$ (where \mathbf{E} is a, possibly empty, set of evidence). While this type of inference is useful in many cases (e.g., to perform predictions by computing the probability of an outcome, diagnosing a disease by computing the probability given some symptoms, ...), it is also desirable to perform *abductive* inference. *Abductive inference* is concerned with finding a configuration of a set of random variables that “best” explains the evidence at hand. This can be done in two fashions:

- Most Probable Explanation (MPE): finding the most probable configuration of *all* variables in a Bayesian network given some evidence.
- Maximum A Posteriori (MAP): finding the most probable configuration of a *subset* of variables in a Bayesian network given some evidence.

Note that being able to perform either does not necessarily solve the other (i.e., the MPE cannot be found by individually maximizing all probabilities for all variables and the MAP cannot be found by taking the corresponding subset of the MPE). It is said that MPE and MAP are *not consistent*.

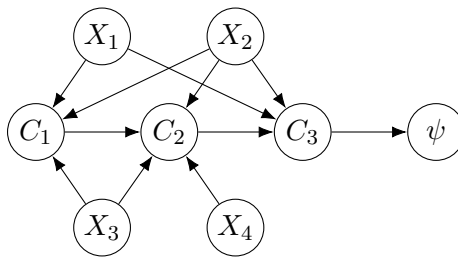
Finding (approximate) MPEs is quite straightforward by replacing the sums (marginalizations) in variable elimination by max’s. In a second backward pass, the sums are replaced by arg max-operators to find the configuration. A pseudocode version of this approach is given in algorithm 2.

4.4 Complexity of Conditional Queries

In general, inference on conditional query $P(\mathbf{X} \mid \mathbf{E} = e)$ is NP-hard. This can be shown by reducing 3-SAT to inference in a Bayesian network. 3-SAT is concerned with the question whether a propositional formula (in conjunctive normal form) with at most three literals per clause is satisfiable. For instance, is the following formula satisfiable (with boolean X_1, X_2, \dots, X_4)?

$$\psi = (\neg X_1 \vee X_2 \vee X_3) \wedge (X_2 \vee \neg X_3 \vee X_4) \wedge (X_1 \vee \neg X_2) \quad (4.1)$$

This problem is known to be NP-hard. This problem can be expressed using a Bayesian network:



The CPTs of C_1 , C_2 , and C_3 then encode the clauses of (4.1) and ψ summarizes them into a single random variable. While it seems possible to just direct all random variables into ψ , this would make the reduction non-polynomial, invalidating the proof of NP-hardness. By putting a prior $P(X_i = t) = 0.5$ over all (boolean) RVs, satisfiability can be queried as $P(\psi = t) > 0$. Hence, if conditional inference would be tractable, 3-SAT would be too. By contradiction, this shows that conditional inference is NP-hard. \square

However, while NP-hard is already pretty hard, it can also be shown that inference is #P-complete involving counting all satisfying configurations.

Even though this shows that inference is hard in the general case, it can be solved for problems with special structures (which are very common). For these, efficient (exact) inference algorithms exist.

Algorithm 2: Most Probable Explanations via Variable Elimination

Input: Bayesian network with RVs X_1, X_2, \dots, X_n and evidence $\mathbf{E} = \mathbf{e}$

Output: Most probable explanation \mathbf{x}^*

```

1 Instantiate evidence  $\mathbf{E} = \mathbf{e}$ .
2 Prune non-active variables (w.r.t.  $\mathbf{X} \cup \mathbf{E}$ ), giving  $X_1, X_2, \dots, X_{\tilde{n}}, \tilde{n} \leq n$ .
3 Choose an ordering  $X_1, X_2, \dots, X_{\tilde{n}}$ .
  // Initialize factors:
4  $\mathcal{F}^{(0)} \leftarrow \{f_i = P(X_i \mid \text{Pa}(X_i)) : i = 1, 2, \dots, \tilde{n}\}$ 
5 for  $i = 1, 2, \dots, \tilde{n}$  do
6   if  $X_i \in \mathbf{E}$  then
7     skip
8   Collect factors  $f_1, f_2, \dots, f_k \in \mathcal{F}^{(i-1)}$  containing  $X_i$ .
  // Generate a new factor  $g$  by maximizing over  $X_i$ :
9    $g = \max_{X_i} \prod_{j=1}^k f_j$ 
  // Replace factors in factor set:
10   $\mathcal{F}^{(i)} \leftarrow (\mathcal{F}^{(i-1)} \setminus \{f_1, f_2, \dots, f_k\}) \cup \{g\}$ 
  // Initialize vector of MPE-values:
11  $\mathbf{x}^* \in \text{val}(X_1) \times \text{val}(X_2) \times \dots \times \text{val}(X_n)$ 
12 for  $i = \tilde{n}, \tilde{n} - 1, \dots, 1$  do
13   if  $X_i \in \mathbf{E}$  then
14     // Copy evident value to result vector:
15      $x_i^* \leftarrow e_i$ 
16   else
17     // Compute  $\arg \max$  over  $X_i$ , incorporating the evidence and already determined values:
18      $x_i^* \leftarrow \arg \max_{X_i} \prod_{f \in \mathcal{F}^{(i-1)}} f(\mathbf{x}_{1:i-1}^*, \mathbf{e})$ 
19 Set pruned variables  $X_{\tilde{n}+1}, X_{\tilde{n}+2}, \dots, X_n$  so arbitrary values.
```



Figure 4.1: Example for creating a moral graph (right) from a Bayesian network (left) by moralizing. All edges are turned into undirected edges and a moral link (depicted in red) has to be introduced between the “unmarried” parents X_2 and X_3 or X_5 .

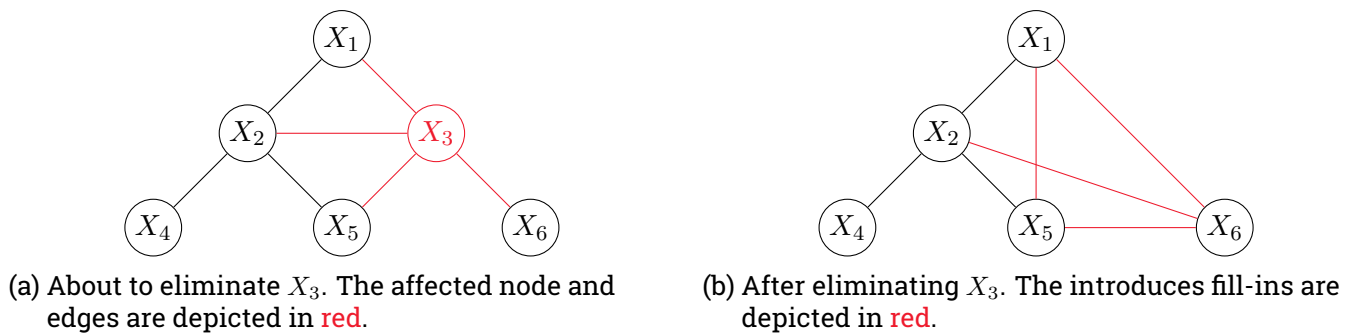


Figure 4.2: Example for eliminating a variable in a moral graph.

4.5 Variable Elimination in Moral Graphs

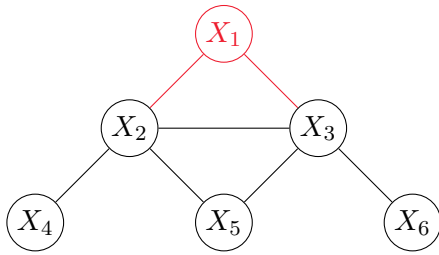
The induced moral graph of a Bayesian network is an undirected graph where the members of each clique correspond to the domain of a potential in the Bayesian network. To extract the moral graph from a Bayesian network, it might be necessary to introduce moral links to fulfill this (defining) criterion. A *moral link* is introduced between unconnected parents of a node³. The rest of the moral graph is then given as the skeleton (i.e., the underlying undirected graph of a DAG) of the network. An example is given in Figure 4.1. More details on why this moral link is necessary are covered in section 5.1.

As variable elimination introduces new factors/potentials into the joint probability, these changes have to be reflected in the moral graph after the elimination. If an elimination step introduces a potential with random variables that have not yet been together in a potential, a *fill-in* is introduced to the moral graph representing this new connection. An example of this process is shown in Figure 4.2. An elimination ordering is said to be *perfect* if it does not introduce any fill-ins (see Figure 4.3 for an example with the same graph used in Figure 4.2).

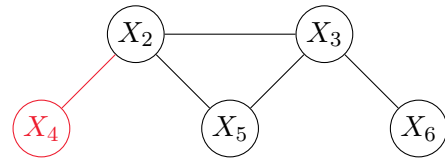
4.5.1 Complexity

As already pointed out in section 4.2, the driving factor of the complexity of VE is the number of variables in the intermediate factors. Hence, for VE on moral graphs, the complexity is determined by the number of variables in the domains of the potentials producing during the elimination. This leads to a set of domains

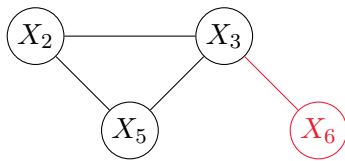
³The term “moralizing” has quite outdated origins and refers to the unmorality of “unmarried” parents.



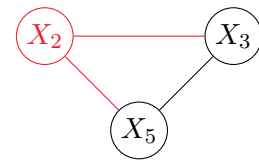
(a) About to eliminate X_1 .



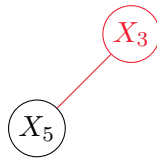
(b) After eliminating X_1 , about to eliminate X_4 .



(c) After eliminating X_4 , about to eliminate X_6 .



(d) After eliminating X_6 , about to eliminate X_2 .



(e) After eliminating X_2 , about to eliminate X_3 .



(f) After eliminating X_3 , elimination finished.

Figure 4.3: Example for a perfect elimination sequence executed in a moral graph. The variable that should be eliminated is always depicted in **red** and is eliminated in the upcoming graph.

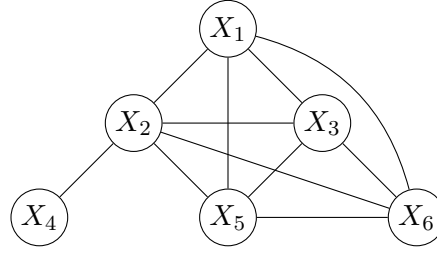


Figure 4.4: Induced graph for the imperfect elimination ordering $(X_3, X_4, X_6, X_1, X_2, X_5)$.

where all sets that are subsets of other sets can be removed (as they do not affect the complexity). For the elimination ordering shown in Figure 4.3, these are, in order of appearance, the following sets:

$$\begin{aligned} & \{\{X_1, X_2, X_3\}, \{X_2, X_4\}, \{X_3, X_6\}, \{X_2, X_3, X_5\}, \{X_3, X_5\}, \{X_5\}\} \\ \longrightarrow & \{\{X_1, X_2, X_3\}, \{X_2, X_4\}, \{X_3, X_6\}, \{X_2, X_3, X_5\}\} \end{aligned}$$

It turns out that all perfect elimination orderings (i.e., those that do not produce fill-ins), have the same set of potential domains. Also, all perfect elimination orderings ending some random variable X are *optimal* in terms of computing $P(X)$.

For the imperfect elimination ordering $(X_3, X_4, X_6, X_1, X_2, X_5)$ (for which the first elimination is shown in Figure 4.2), the following sets of potentials are created:

$$\begin{aligned} & \{\{X_1, X_2, X_3, X_5, X_6\}, \{X_2, X_4\}, \{X_1, X_2, X_5, X_6\}, \{X_1, X_2, X_5\}, \{X_2, X_5\}, \{X_5\}\} \\ \longrightarrow & \{\{X_1, X_2, X_3, X_5, X_6\}, \{X_2, X_4\}\} \end{aligned}$$

Intuitively, the number of items of this clique are closely related to the complexity of VE (a small clique set corresponds to large connectivity in the graph, hence large factors).

Induced Graph and Treewidth

To further formalize the aforementioned idea, the notion of an induced graph is introduced. The *induced graph* of some moral graph is the graph including all connections of the clique set, i.e., the moral graph with all fill-ins inserted. Hence, the moral graph is also a sub-graph of the induced graph. For the perfect elimination ordering of Figure 4.3, the induced graph is also the moral graph (as no fill-ins were added). However, for the imperfect elimination sequence $(X_3, X_4, X_6, X_1, X_2, X_5)$, the induced graph has more connections as shown in Figure 4.4.

It is clear that the size of the maximum clique differs for the perfect and imperfect elimination sequence. This quantity (minus one) is called the *treewidth* of a graph and is called *induced treewidth* for a Bayesian network combined with an elimination ordering (which highly influences the treewidth of the moral graph). Hence, finding a good elimination ordering is equivalent to minimizing the treewidth of the induced graph.

(Poly-) Trees

A direct consequence of the tight coupling between the complexity of VE and the treewidth is that inference on a “tree” Bayesian network (one in which every node has at most one parent) is linear in the number of variables (as the treewidth is 1 for a tree)! Finding an elimination ordering is also straightforward: by starting from the leaves, each factor only involves two RVs. It is also guaranteed that a tree has a leave, so the ordering cannot stall.

Query	Graph Type	
	General	Low Treewidth
Probabilistic Inference	#P-Complete	Easy
Most Probable Explanation (MPE)	NP-Complete	Easy
Maximum A Posteriori (MAP)	NP ^{PP} -Complete	NP-Hard

For approximate probabilistic inference, hardness depends on the error:

Absolute Error, $|\hat{P} - P| \leq \epsilon$: NP-Hard for $\epsilon < 0.5$

Relative Error, $|\hat{P} - P|/P \leq \epsilon$: NP-Hard for $\epsilon > 0$

Table 4.1: Summary of complexities for various inference queries.

For polytree Bayesian networks (ones of which the skeleton is a tree but a node may have more than one parent), the treewidth is 2 and inference is linear in CPT size (i.e., representation size of the network). Opposed to a tree BN, the treewidth *is not* 1 as moral links have to be added between “unmarried” parents.

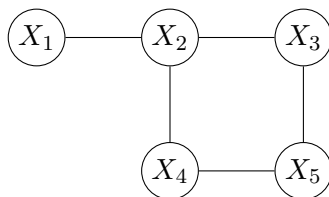
General Networks

For general networks, the treewidth is usually greater than 1. In fact, for any graph that has a cycle, the treewidth is at least 2. Form graph theory, it is known that finding an optimal ordering minimizing the treewidth is NP-hard, so again, in general, inference is intractable. However, there are some neat heuristics that can be applied for finding “good” (but not necessarily optimal) orderings. There also exist algorithm that—if the graph has low treewidth—find the optimal ordering in polynomial time. Also, incorporating CSIs can reduce the complexity to be tractable even for graphs with large treewidth.

A summary of the complexity results is given in Table 4.1.

5 Markov Random Fields

Bayesian networks are not allowed to have cycles as they would not make any sense. Another variant of PGMs, Markov Random Fields (MRFs), allow cycles and are undirected. Instead of encoding dependencies, the edges in a Markov network encode which random variables occur together in a potential. For example, the Markov network



corresponds to the distribution

$$P(X_1, X_2, X_3, X_4, X_5) \propto \phi(X_1, X_2) \phi(X_2, X_3) \phi(X_2, X_4) \phi(X_3, X_5) \phi(X_4, X_5)$$

up to normalization. In general, a distribution $P(X_1, X_2, \dots, X_n)$ specified by a MRF G factors as

$$P(X_1, X_2, \dots, X_n) = \frac{1}{Z} \prod_{C \in \text{Cliques}(G)} \phi_C(X_1, X_2, \dots, X_n)$$

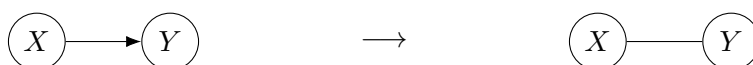
where $\text{Cliques}(G)$ is the set of maximal cliques in G and $1/Z$ is the normalization factor. In an MRF, the *Markov property* holds: two sets of random variables, \mathcal{X} and \mathcal{Y} are independent given a *separating subset* \mathcal{Z} ($\mathcal{X} \perp \mathcal{Y} \mid \mathcal{Z}$). This subset has the property that every path from any variable in \mathcal{X} to any variable in \mathcal{Y} contains a variable from \mathcal{Z} .

Proof

5.1 Bayesian Networks as MRFs

It is possible to convert Bayesian networks into MRFs for leveraging the theory that evolved around them. This process was already introduced as *moralizing* before (section 4.5). This section builds up a more formal view on how the addition of a moral link is necessary. The following paragraphs show to convert the fundamental building blocks (pair, chain, common cause, and common effect) of a Bayesian network into a Markov network.

Pair



This conversion makes sense as from the Bayesian model,

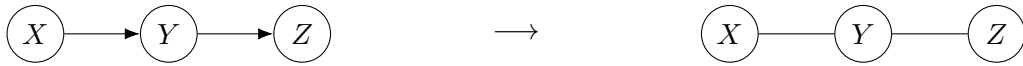
$$P(X, Y) = P(X) P(Y | X),$$

and from the Markov network,

$$P(X, Y) \propto \phi(X, Y)$$

with $\phi(X, Y) \propto P(X) P(Y | X)$.

Chain



Like for the pair of variables, the joint probability is

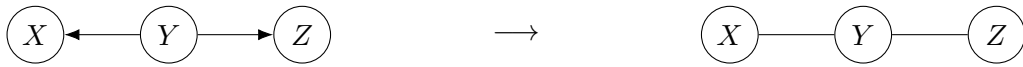
$$P(X, Y, Z) = P(X) P(Y | X) P(Z | Y)$$

by the Bayesian model and

$$P(X, Y, Z) \propto \phi(X, Y) \phi(Y, Z)$$

by the Markov network with $\phi(X, Y) \propto P(X) P(Y | X)$ and $\phi(Y, Z) \propto P(Z | Y)$. Note that the parametrization of the potentials is not unique.

Common Cause



For a common cause, the conversion is again similar to the chain model. By the Bayesian network,

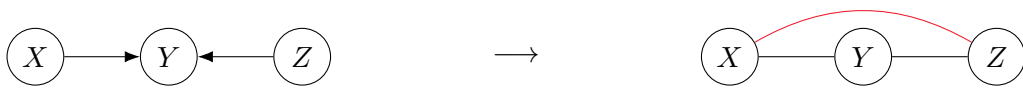
$$P(X, Y, Z) = P(X | Y) P(Y) P(Z | Y)$$

and by the Markov network,

$$P(X, Y, Z) \propto \phi(X, Y) \phi(Y, Z)$$

with $\phi(X, Y) \propto P(X | Y) P(Y)$ and $\phi(Y, Z) \propto P(Z | Y)$. Note again that the parametrization is not unique.

Common Effect



For the v-structure, the conversion is a tad more complicated. By the Bayesian network,

$$P(X, Y, Z) = P(X) P(Y | X, Z) P(Z).$$

But if the moral link (depicted in red) would not be present, the Markov network would model the distribution as

$$P(X, Y, Z) = \phi(X, Y) P(Y, Z).$$

However, this would entail $X \perp Y \mid Z$ which is—prominently as Berkson’s paradox—not true. Hence, a moral link has to be included to ensure that the said conditional independency does not hold. With the moral link, the Markov network entails

$$P(X, Y, Z) \propto \phi(X, Y) \phi(Y, Z) \phi(X, Z).$$

Note that this formulation *does not* encode the same independencies: in the Markov network formulation, $X \perp Y$ does not hold anymore. However, it is more important to not capture invalid independencies than missing out on some.

This observation depends on the fact that two variables X and Y are conditionally dependent given Z if and only if their distribution $P(X, Y, Z)$ can be written as $P(X, Y, Z) = \phi(X, Y) \phi(Y, Z)$ with some $\phi(\cdot)$.

5.2 Triangulated Graphs and Simplicial Nodes

For the upcoming sections, some nomenclature is needed:

- A graph is *triangulated* if it has a perfect elimination ordering (i.e., one that does not introduce any fill-ins).
- A node is *simplicial* if its family (itself in addition to its neighbors) is equivalent to its maximum clique.

It follows that in a triangulated graph, the elimination of a simplicial node again yields a triangulated graph. Also, every triangulated graph (with at least two nodes) has at least two simplicial nodes. Hence, it directly follows that a perfect elimination ordering exists for every random variable in the graph.

Note that it is possible to triangulate every graph by applying any elimination ordering and consider the induced graph instead. However, while this creates a triangulated graph, it does not induce high-performance inference as the produces graph might have very large treewidth!

5.3 Join Trees

A *join tree* is a special kind of tree where each node (also called *cluster* in join trees) holds a set of variables. Additionally, the *running intersection property* (RIP) holds: for each pair of nodes, \mathcal{X} and \mathcal{Y} , each node along each path between the two nodes contains the intersection $\mathcal{X} \cap \mathcal{Y}$. An example is shown in Figure 5.1. It can be shown that if the (maximum) cliques of a undirected graph can be organized into a join tree, then the graph is triangulated and vice versa.

5.3.1 Markov Network as a Join Tree: Junction Tree

Organizing a triangulated graph into a join tree is straightforward by exploiting the special properties of such a graph. Pseudocode for this process is given in algorithm 3 and. A join tree G_J built from a Markov network G' also models the joint distribution as

$$P(X_1, X_2, \dots, X_n) = \frac{\prod_{C \in \text{Cliques}(G_J)} \phi_C(X_1, X_2, \dots, X_n)}{\prod_{S \in \text{Sep}(G_J)} \phi_S(X_1, X_2, \dots, X_n)}$$

That is, the joint distribution is the product of the clique potentials divided by the product of the separator potentials.

Proof

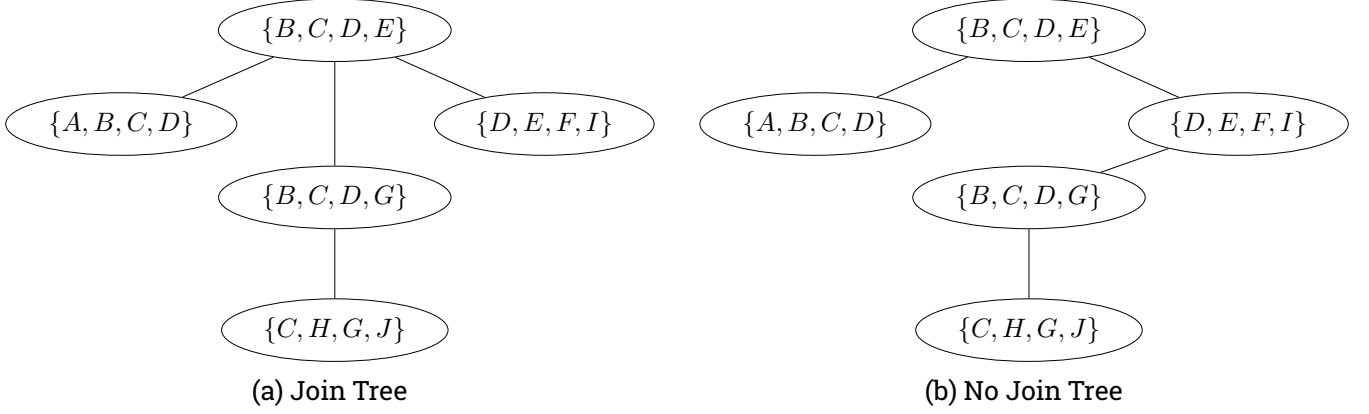


Figure 5.1: Illustration of a join tree. The left is a join tree, but the right is not as on the path between $\{A, B, C, D\}$ and $\{C, H, G, J\}$ has to contain $\{C\}$ in every node, which is not a subset of $\{D, E, F, I\}$.

Algorithm 3: Triangulated, Undirected Graph \rightarrow Join Tree

Input: Triangulated undirected graph G with random variables X_1, X_2, \dots, X_n

Output: Join tree with clique and separator nodes

- 1 Initialize set of nodes \mathcal{X} .
 - 2 Initialize count of eliminated nodes $n \leftarrow 0$
 - 3 while \mathcal{X} is not empty do
 - 4 Pop simplicial node X from \mathcal{X} .
 - 5 // Construct set of variables to eliminate:
 $\mathcal{E} \leftarrow \{Y \in \text{Family}(X) : \text{Family}(Y) \subseteq \text{Family}(X)\}$.
 - 6 Eliminate all variables in \mathcal{E} .
 - 7 // Increment counter for number of eliminated variables:
 $n \leftarrow n + |\mathcal{E}|$
 - 8 // Create clique and separator nodes for X :
 $V_n \leftarrow \text{Family}(X)$
 - 9 $S_n \leftarrow \text{Family}(X) \setminus \mathcal{E}$
 - 10 Connect V_n and S_n .
 - 11 Connect each separator S_i to exactly one clique V_j with $j > i$ and $S_i \subseteq V_j$.
-

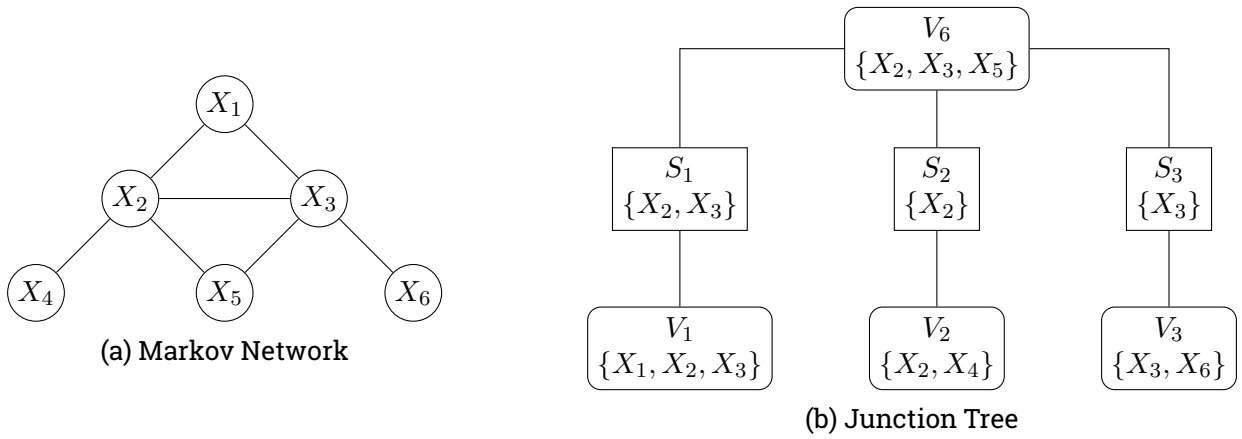


Figure 5.2: Example of the junction tree (right) produced using algorithm 3 with the elimination ordering $(X_1, X_4, X_6, X_2, X_3, X_5)$ showed in Figure 4.3.

5.4 Junction Trees

A *junction tree* of a Markov network is a join tree for which each factor ϕ in the Markov network is associated to a cluster V in the join tree such that $\text{dom } \phi \subseteq V$. This is called the *family preservation property* of junction trees. Such a join tree can, for example, be generated using algorithm 3 if an elimination ordering is given; see Figure 5.2 for an example. However, it is not necessary to build up a triangulated graph beforehand in general (it would be, for example, possible to create a junction tree with a node containing all nodes of the original tree; however, this would be useless as it does not leverage any theory of junction trees at all).

Ideally, the clusters are as small and modular as possible. Unfortunately, it is NP-hard to find the optimal tree...However, some useful heuristics and special cases exist. For example, tree-BNs can directly be converted to an efficient junction tree by converting each edge into a cluster.

5.4.1 Inference

To perform inference in a junction tree, each node sends exactly one *message* to exactly one of its neighbors once it has received messages from all its other neighbors. Hence, message passing must start at the leaves of the tree. To define a leaf, however, first an arbitrary root node must be picked. Messages are then collected at this node and afterwards distributed back across the tree to the leaves. After this process, all clusters have the relevant information to compute the potential for its variables.

To pass messages between the clusters, each separator S_i has two “mailboxes” ψ_i and ψ^i , one for the forward- and one for the backward-pass, respectively. That is, ψ_i is filled when the messages flow from the leaves to the root while ψ^i is filled when the beliefs flow back from the root to the leaves. Note that from the construction of the junction tree, each separator has exactly two connections. Each clique C_i is associated with the potentials ϕ_j of the Markov network for which $\text{dom } \phi_j \subseteq C_i$ holds. Let $\Phi(C_i)$ be the set of potentials associated with C_i . Additionally, each clique C_i is associated with a potential φ_i that is used to store the current belief over the variables of the clique. All of this is illustrated in Figure 5.3.

Input: Junction tree with separators $\{S_j\}$ and cliques $\{V_i\}$ with associated potentials $\Phi(C_i)$ from the Markov network

Output: Potentials $\{\varphi_j\}$ for each clique

1 Initialize separator potentials

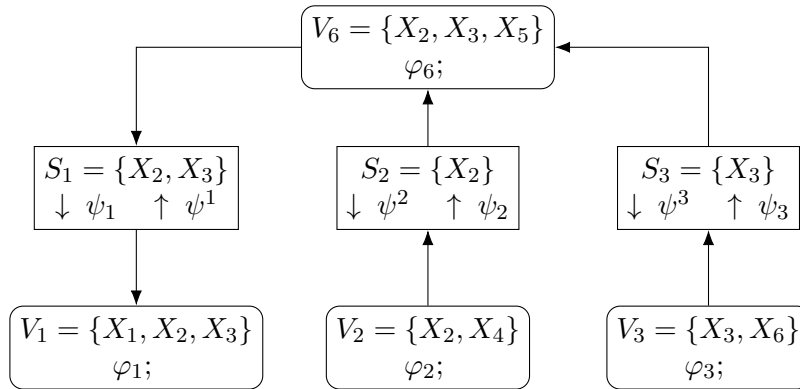


Figure 5.3: Illustration of the message boxes and potentials used for message passing in junction trees, based on the junction tree shown in Figure 5.2. For this tree, V_1 is chosen as the root node. The arrow directions depict the forward pass, the backward pass goes in reverse.

6 Learning

Until now, it was assumed that both the network structure and the CPTs are given somehow. This chapter is concerned with learning these from data. It is important to be able to learn structures from data as usually, no expert is available that can craft a network by hand or she is really expensive. And in the age of big data, data is cheap and huge amounts of data are available.

Learning Bayesian network is especially appealing as detecting and modeling conditional independencies captures the structure of many real-world distributions and can lead to knowledge discovery by capturing causal relationships. Also, the learned model can be used for handling missing data and hidden variables.

In a *complete* dataset, every data point has the values for all attributes that are getting investigated. On the other hand, *incomplete* datasets lack some values of random variables (this is usually the case in real-world data...). Also, a special kind of incompleteness, are hidden variables. *Hidden variables* never have assigned values in the whole dataset. One example for this is the cluster ID in an unsupervised clustering problem. They can also present a low-dimensional embedding that summarizes some other variables and separates them from a second set of RVs (dimensionality reduction).

The relation of the different problem types are summarized in Table 6.1.

6.1 Parameter Estimation

In *parameter estimation*, it is assumed that the structure of the Bayesian network (i.e., the random variables and dependencies between them) is fixed and only the entries of the CPTs have to be estimated. This field further subdivides into the cases of complete and incomplete data.

6.1.1 Known Structure, Complete Data

With known structure and complete data, the problem simply reduces to basically counting. The most basic method to do so are maximum likelihood estimators (where the likelihood can be computed using the Bayesian network). Some advantages of ML estimated are that they are (often) intuitive, consistent (the estimate converges to the best possible value with increasing number of data points), and asymptotic efficiency (given a dataset, the estimate is as close as possible to the true value).

Observed	Fixed Structure	Fixed Variables	Hidden Variables
Fully	Easy (Counting, MLE)	Selection of Edges	—
Partially	Numerical Optimization (difficult for large networks)	Encompasses for Difficult Subproblem (only structural EM known)	Scientific Discovery

Table 6.1: Overview over different learning problems and solution approaches.

Decomposability of the Likelihood

An important property of the likelihood for a Bayesian network is that it is *decomposable*. Let θ_i be the CPT of $P(X_i | \text{Pa}(X_i))$ and let $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)}$ be the data points. Then the log-likelihood is

$$\log P(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)} | \theta_1, \theta_2, \dots, \theta_n) = \sum_{j=1}^N \log P(\mathbf{x}^{(j)} | \theta_j) = \sum_{j=1}^N \sum_{i=1}^n \log P(X_i = x_i^{(j)} | \text{Pa}(X_i), \theta_j).$$

By swapping the sums, the log-likelihood can be rewritten as

$$\log P(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)} | \theta_1, \theta_2, \dots, \theta_n) = \sum_{i=1}^n \log P(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)} | \theta_i).$$

Hence, it is possible to maximize each likelihood function independently and subsequently combine the solutions to get an ML estimate. This allows efficient solutions by distributing the calculations.

Likelihood for (Conditional) and Multinomials

An important special case for the CPTs are (conditional) multinomials, i.e., distributions for which

$$P(X = k) = \theta_k$$

where $k = 1, 2, \dots, |\text{val}(X)|$ are indices for the values of X . Maximizing the likelihood of this distribution yields the solution

$$\theta_k^* = \frac{N_k}{N}$$

where N_k is the number of data points for which $X = k$, i.e., $N_k = \sum_x \mathbb{1}[x = k]$ and N is the total number of data points.

For *conditional* multinomials, the distribution has the form $P(X = k | Y = \ell) = \theta_{k|\ell}$. Similarly, the ML solution is

$$\theta_{k|\ell}^* = \frac{N_{k|\ell}}{N_\ell} \quad (6.1)$$

where $N_{k|\ell}$ is the number of data points for which $X = k$ and $Y = \ell$ and N_ℓ is the number of data points for which $Y = \ell$. Hence, conditional multinomials are completely analogous to non-conditional multinomials but the dataset is constrained to the respective conditional values.

Proof

6.1.2 Known Structure, Incomplete Data (Expectation-Maximization)

When the dataset is incomplete, the algorithm somehow has to deal with the incompleteness. One idea is to use an EM algorithm to complete the data in the E-step and to estimate the parameters in the M-step. For multinomial distributions and discrete data, this involves computing *expected counts* in the E-step and using the ML estimated (6.1) in the M-step to tune the model. Let $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)}$ be the data points, then the expected count $\mathbb{E}[N_{k|\ell}]$, where $N_{k|\ell}$ would be the number of data points for which $X = k$ and $Y = \ell$. The

expected count $\mathbb{E}[N_{k|\ell}]$, where $N_{k|\ell}$ would be the number of data points for which $X = k$ and $Y = \ell$ holds (assuming a distribution $P(X = k | Y = \ell) = \theta_{k|\ell}$), is

$$\mathbb{E}[N_{k|\ell}] = \sum_{i=1}^N P(k, \ell | \mathbf{x}^{(i)}). \quad (6.2)$$

If $\mathbf{x}^{(i)}$ is complete, the probability would reduce to $\mathbb{1}[x_x^{(i)} = k \wedge x_y^{(i)} = \ell]$, i.e., counting. However, if the data is not complete, inference has to be run to actually compute the probability. Using the expected counts, the parameters can easily be estimated in the M-step using

$$\theta_{k|\ell} = \frac{\mathbb{E}[N_{k|\ell}]}{\mathbb{E}[N_\ell]},$$

where $\mathbb{E}[N_\ell]$ is analogous to (6.2).

In practice, some more things are needed to apply EM. Firstly, the parameters have to be initialized somehow (e.g., randomly, or as a best guess from other sources, by estimating from just the complete data points, ...). Secondly, a stopping criterion has to be defined (only small changes in the likelihood or parameter values, for example). Thirdly, bad local maxima (which the EM algorithm is prone to converge to) have to be avoided (e.g., by running multiple times, pruning unpromising runs soon, ...). Also, various methods are available to speed up convergence.

6.1.3 Gradient Ascent

An alternative, rather bruteforcy method for dealing with incomplete data is to just ignore it and use that the gradient

$$\frac{\partial}{\partial \theta_{k|\ell}} \log P(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)} | \Theta) = \frac{1}{\theta_{k|\ell}} \sum_{j=1}^N \log P(k, \ell | \mathbf{x}^{(j)}, \Theta)$$

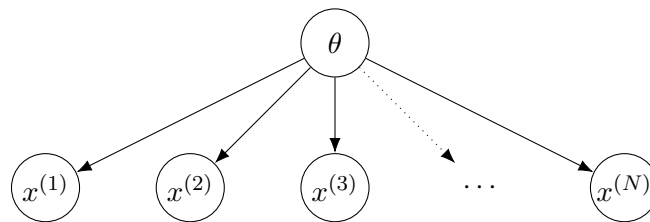
can be computed in closed form. Then, gradient ascent can be applied. This has the advantage that training is well-known from training neural networks, coming with the disadvantages that the parameters somehow have to be constrained to be legal and the requirement for smart optimization techniques to ensure convergence.

6.1.4 Bayesian Parameter Estimation

In Bayesian estimation, the parameters are assumed to be random variables as well. Hence, by putting a prior on the parameters, learning can be understood as a form of inference using Bayes rule:

$$P(\Theta | \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)}) = \frac{P(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)} | \Theta) P(\Theta)}{P(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)})}.$$

Assume (for now) that the model only has a single binomial random variable X with the parameter θ . Then, by the i.i.d. assumption, all data points are independent from each other and they can be represented easily by the following Bayesian network:



This network has the interesting property that $P(x^{(i)} | \theta) = \theta$. Hence, predicting θ directly corresponds to inference in this network.

Laplace Estimation

Bayesian Prediction

Conjugate Priors

Binomial Prior

Dirichlet Prior

Bayesian Networks and Bayesian Prediction

6.1.5 Summary

Two approaches for parameter estimation with a fixed structure have been considered: maximum likelihood and Bayesian estimation. While both are asymptotically equivalent and consistent (i.e., they converge to the same “true” value), Bayesian estimation provides smoother result regularized by the prior. And if the prior is chosen to be close to the optimal value, it converges faster.

6.2 Structure Learning / Model Selection

In the previous section, it was assumed that the structure of the Bayesian network (the random variables and dependencies between them) were given. However, this is often not the case. In essence, the goal is to find, given a P , a G which is an I-map of P . To do some, a notion of minimal I-maps and equivalence of I-maps is necessary.

6.2.1 Minimal I-Maps and Perfect Maps (P-Maps)

A straightforward approach for defining an I-map to be *minimal* is to call an I-map minimal if it is as simple as possible, i.e., the removal of an edge would make the graph not be an I-map anymore. However, this has the major caveat that even graphs with lots of edges might be “minimal” in this sense although other configuration have less edges. Consider, for instance, the following networks (left is the true model):



Although both models entail the independencies required to be an I-map, the left model is obviously simpler than the right. Hence, the above definition of a minimal I-map (removing an edge results in the graph not being an I-map anymore) is insufficient to capture simplicity and minimalism.

6.2.2 Perfect Maps (P-Maps) and I-Equivalence

An alternative approach is to not just try to find an I-map but to find a *perfect map* (P-map) with $I(G) = I(P)$. This eliminates the problem of defining minimalism and simplicity. Instead, a notion of equivalence between maps must be created. Straightforwardly, two models G_1 and G_2 are said to be *I-equivalent* if $I(G_1) = I(G_2)$. This definition leads to a theorem for I-equivalence of two Bayesian networks:

Theorem Two Bayesian networks G_1 and G_2 are I-equivalent if and only if they have the same skeleton and same immoralities (v-structures with disconnected parents).

6.2.3 Obtaining a P-Map

To obtain a P-map, in a first step the basic independencies of P (skeleton and immoralities) have to be extracted. This yields the equivalence class the Bayesian network has to live in. In a second step, every BN structure has to be obtained that lies in this equivalence class.

Identifying the Skeleton

Identifying Immoralities

From Immoralities to Structures

6.2.4 Accurate Structures

6.2.5 Learning

Constrained-Based

Score-Based

Likelihood Score

Bayesian Score and Bayesian Information Criterion

6.2.6 Structure Search as Optimization

Learning Trees (Complete Data)

Heuristic (Local) Search

6.2.7 Structural EM

6.2.8 Summary

7 Dynamic Bayesian Networks

7.1 Hidden Markov Models

7.2 Inference

7.2.1 Decoding

Forward Pass

Backward Pass

7.2.2 Best State Sequence

Viterbi Algorithm

7.2.3 Parameter Estimation

7.3 State Estimation (Kalman Filter)

7.3.1 Recursive Bayesian Updating

7.3.2 (Modeling) Actions

7.3.3 Bayes Filter

7.3.4 Discrete-Time Kalman Filter

Dynamics and Observations

Belief Update: Prediction

Belief Update: Correction

7.4 General Dynamic Bayesian Networks

7.4.1 Exact Inference

7.4.2 Tractable, Approximate Inference

Assumed Density Filtering

8 Approximate Inference

8.1 Message Passing

8.1.1 Sum-Product Belief Propagation

8.1.2 (Acyclic) Belief Propagation as Dynamic Programming

8.1.3 Loopy Belief Propagation

8.2 Sampling

8.2.1 Forward Sampling (Without Evidence)

8.2.2 Forward Sampling (With Evidence)

8.2.3 Gibbs Sampling

Burn-In

Irreducibility, Aperiodicity, and Ergodicity

Convergence

Performance

Speeding Convergence

Skipping Samples

Randomized Variable Order

Blocking

Rao-Blackwellization

Multiple Chains

8.2.4 Likelihood Weighting

9 Tractable Probabilistic Models

9.1 Deep Learning

9.2 Probabilistic Circuits

9.3 Sum-Product Networks

9.3.1 Inference

9.3.2 Learning

Directly Learning SPNs

9.3.3 Inference on Devices

10 Deep Generative Models

10.1 Likelihood-Based

10.1.1 Autoregressive Generative Models

Learning and Inference

Parametrization

10.1.2 Variational Auto-Encoders

Inference as Optimization

Variational Bayes

Learning and Inference

Open Questions

10.1.3 Normalizing Flows

Learning and Inference

10.2 Likelihood-Free

10.2.1 Generative Adversarial Networks

Inference

10.3 Applications in Scientific Discovery
