Probabilistic Graphical Models

Summary

Fabian Damken

February 20, 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 . . . 27
- 5.3 Join Trees . . . 27

5.4 Junction Trees . . . 27
5.4.1 Collecting Evidence . . . 27
5.4.2 Distributing Evidence . . . 27
5.5 Non-Triangulated Graphs . . . 27

6 Learning **28**
6.1 Complete and Incomplete Data Sets . . . 28
6.1.1 Hidden Variables . . . 28
6.2 Parameter Estimation . . . 28
6.2.1 Known Structure, Complete Data . . . 28
6.2.2 Known Structure, Incomplete Data (Expectation-Maximization) . . . 28
6.2.3 Gradient Ascent . . . 28
6.2.4 Bayesian Parameter Estimation . . . 28
6.2.5 Summary . . . 29
6.3 Structure Learning / Model Selection . . . 29
6.3.1 Minimal I-Maps . . . 29
6.3.2 Perfect Maps (P-Maps) . . . 29
6.3.3 I-Equivalence . . . 29
6.3.4 Obtaining a P-Map . . . 29
6.3.5 Accurate Structures . . . 29
6.3.6 Learning . . . 29
6.3.7 Structure Search as Optimization . . . 29
6.3.8 Structural EM . . . 29
6.3.9 Summary . . . 29

7 Dynamic Bayesian Networks **30**
7.1 Hidden Markov Models . . . 31
7.2 Inference . . . 31
7.2.1 Decoding . . . 31
7.2.2 Best State Sequence . . . 31
7.2.3 Parameter Estimation . . . 31
7.3 State Estimation (Kalman Filter) . . . 31
7.3.1 Recursive Bayesian Updating . . . 31
7.3.2 (Modeling) Actions . . . 31
7.3.3 Bayes Filter . . . 31
7.3.4 Discrete-Time Kalman Filter . . . 31
7.4 General Dynamic Bayesian Networks . . . 31
7.4.1 Exact Inference . . . 31
7.4.2 Tractable, Approximate Inference . . . 31

8 Approximate Inference **32**
8.1 Message Passing . . . 32
8.1.1 Sum-Product Belief Propagation . . . 32
8.1.2 (Acyclic) Belief Propagation as Dynamic Programming . . . 32
8.1.3 Loopy Belief Propagation . . . 32
8.2 Sampling . . . 32
8.2.1 Forward Sampling (Without Evidence) . . . 32
8.2.2 Forward Sampling (With Evidence) . . . 32

8.2.3 Gibbs Sampling . . . 32
8.2.4 Likelihood Weighting . . . 32

9 Tractable Probabilistic Models 33
9.1 Deep Learning . . . 33
9.2 Probabilistic Circuits . . . 33
9.3 Sum-Product Networks . . . 33
9.3.1 Inference . . . 33
9.3.2 Learning . . . 33
9.3.3 Inference on Devices . . . 33

10 Deep Generative Models 34
10.1 Likelihood-Based . . . 34
10.1.1 Autoregressive Generatie Models . . . 34
10.1.2 Variational Auto-Encoders . . . 34
10.1.3 Normalizing Flows . . . 34
10.2 Likelihood-Free . . . 34
10.2.1 Generative Adversarial Networks . . . 34
10.3 Applications in Scientific Discovery . . . 34

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

List of Tables

List of Algorithms

1 Variable Elimination 20
2 Most Probable Explanations via Variable Elimination 22

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

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

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.,

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

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 properties. A milder property is *conditional* independence, i.e., two random variables X and Y are independent if Z is given:

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

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

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

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

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

[1] Take 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*:

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

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

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

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

$$H_P(X, Y, Z) = H_P(X) + H_P(Y \mid X) + H_P(Z \mid X, Y).$$

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

$$I_P(X; Y) = H_P(X) - H_P(X \mid Y)$$

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 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., $\text{dom}\, f_{X,Y,Z} = \{X, Y, Z\}$. 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

$P(X = x \mid Y = y, Z = z)$	$x = \mathfrak{f}$	$x = \mathfrak{t}$
$(Y, Z) = (\mathfrak{t}, \mathfrak{t})$	0.8	0.2
$(Y, Z) = (\mathfrak{t}, \mathfrak{f})$	0.5	0.5
$(Y, Z) = (\mathfrak{t}, \mathfrak{t})$	0.2	0.8
$(Y, Z) = (\mathfrak{t}, \mathfrak{f})$	0.7	0.3

$\longleftrightarrow$

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

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 $P(X_1, X_2, \ldots, X_n)$ with $|\text{val}(X_i)| = k$ for simplicity, the CPT has $k^n - 1$ entries[1]! But lots of information is redundant in the joint if some of the variables exhibit independencies. Assume, for instance, that all subsets $\mathcal{X}, \mathcal{Y} \subseteq \{X_1, X_2, \ldots, X_n\}$ are independent ($\mathcal{X} \perp \mathcal{Y}$). Then the joint can be written as

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

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 Bayed 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 $\{X_1, X_2, \ldots, X_n\}$ are all conditionally independent given the class C; $\forall \mathcal{X}, \mathcal{Y} \subseteq \{X_1, X_2, \ldots, X_n\} : \mathcal{X} \perp \mathcal{Y} \,|\, C$. The joint distribution is therefore simply

$$P(X_1, X_2, \ldots, X_n, C) = P(X) \prod_{i=1}^{n} P(X_i). \tag{3.1}$$

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

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

$$c_{\text{MAP}} = \operatorname*{arg\,max}_{c \in \text{val}(C)} P(c \,|\, \boldsymbol{x}) = \operatorname*{arg\,max}_{c \in \text{val}(C)} \frac{P(\boldsymbol{x} \,|\, c)\, P(c)}{P(\boldsymbol{x})} = \operatorname*{arg\,max}_{c \in \text{val}(C)} P(\boldsymbol{x} \,|\, c)\, P(c) = \operatorname*{arg\,max}_{c \in \text{val}(C)} P(X) \prod_{i=1}^{n} P(X_i).$$

[1] The -1 comes from the requirement of the probability being normalized, hence the probability of the "last" configuration is implicit.

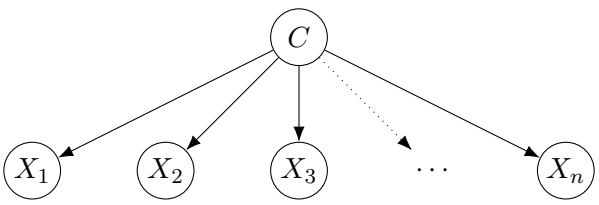


Figure 3.1: Bayesian network for the naive Bayes classifier.

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:

$$\hat{P}(C = c_j) = \frac{N(C = c_j)}{N} \qquad \hat{P}(X_i = x_{ik} \mid c_j) = \frac{N(X_i = x_{ik}, C = C_j)}{N(C = c_j)}.$$

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

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

where $\mathcal{D} \subseteq \text{val}(X_1) \times \text{val}(X_2) \times \cdots \times \text{val}(X_n) \times \text{val}(C)$ 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 reducing by *smoothing* the distributions to avoid overfitting,

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

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, is 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 $\{X_1, X_2, \ldots, X_n\}$.
- Directed acyclic graph (DAG) encoding the (in-) dependencies.

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

Then, the joint distribution is given by

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P\big(X_i \mid \text{Pa}(X_i)\big) \tag{3.2}$$

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 $\text{Pa}(X_i)$:

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

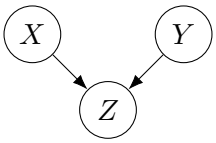
Let $\text{Ch}(X_i)$ be the children of X_i, then X_i's non-descendants are all random variables that are not in $\text{Des}(X_i)$, X_i's descendants:

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

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 $X \perp Y \mid Z$ 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:

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

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 (induces 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 if an I-map of P if and only if P factorizes according to G via (3.2).

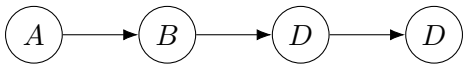


Figure 3.2: Illustration of a Bayesian network capturing more independencies than induced by the local Markov assumption. In this BN, the local Markov assumption encodes $I_\ell(G) = \{ B \perp A \,|\, A, C \perp (A, B) \,|\, B, D \perp (A, B, C) \,|\, C \}$. However, also $A \perp D \,|\, C$ is a captured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

Proof The proof is split into proofing "if" and "only if".

"if":

"only if": Assume, w.l.o.g., a topological ordering $X_1, X_2, \ldots, X_n$ where $j < i$ for all children X_j of some random variable X_i. Applying the chain rule to the joint yields $P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_1 \,|\, X_1, X_2, \ldots, X_{i-1})$. By the topological ordering, $\text{Pa}(X_1) \subseteq \{X_1, X_2, \ldots, X_{i-1}\}$ and hence, using the local Markov assumption, $P(X_i \,|\, X_1, X_2, \ldots, X_{i-1}) = P\big(X_1 \,|\, \text{Pa}(X_i)\big)$. 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 $Y \to X$ only removes local independencies, but does not induce new ones.

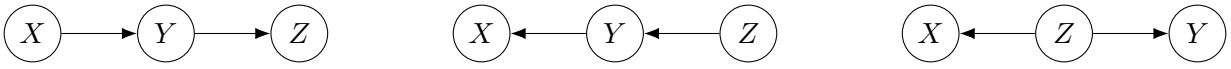
3.3 Encoded Independencies

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:



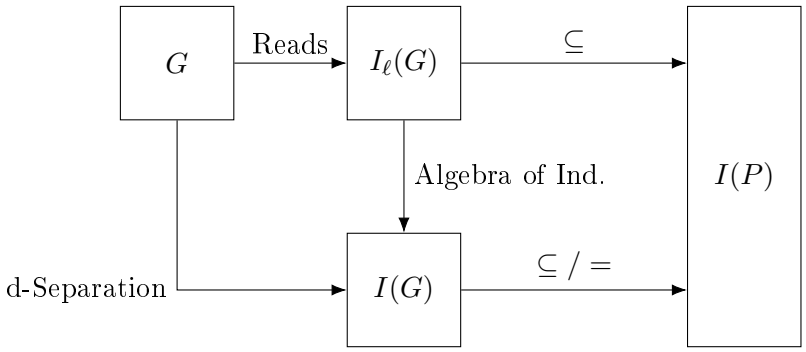
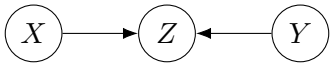


Figure 3.3: Illustration of different methods to capture the independencies of a distribution. Note that $I(G)$ and $I(P)$ are equal (i.e., P is faithful to G) for almost all P that factor over G.

In both cases, the conditional independence $X \perp Y \mid Z$ holds, but *not* the simple independence $X \perp Y$.

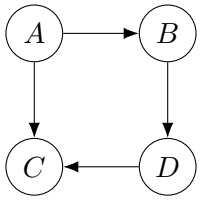
The last and most special structure is a v-structure, where two variables have a common effect:



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 $X'_1 \leftrightarrow X'_2 \leftrightarrow \cdots \leftrightarrow X'_k$ 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 note can only be visited once. For example, the network



has the following trails:

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

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 $X'_1 \leftrightarrow X'_2 \leftrightarrow \cdots \leftrightarrow X'_k$ 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 $X_{i-1} \to X_i \to X_{i+1}$, X_i is *not observed*.
- for $X_{i-1} \leftarrow X_i \leftarrow X_{i+1}$, X_i is *not observed*.
- for $X_{i-1} \leftarrow X_i \to X_{i+1}$, X_i is *not observed*.
- for $X_{i-1} \to X_i \leftarrow X_{i+1}$, 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

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

holds but context-specific independencies,

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

are not captured.

One option for representing CSIs are Tree CPDs. *Tree CPDs* encode a distribution $P\big(X \mid \text{Pa}(X)\big)$ 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 $P(X \mid e)$ computing some probability given some evidence e. The straightforward way to compute the conditional is to directly use its definition,

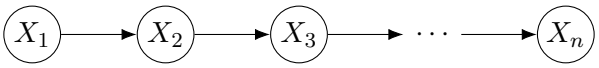
$$P(X \mid e) = \frac{P(X, e)}{P(e)}.$$

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

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

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

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

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

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

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 $Y_1, Y_2, \ldots, Y_k$ is named $f_X(Y_1, Y_2, \ldots, Y_k)$. 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 $\boldsymbol{X} \cup \boldsymbol{E}$ 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 polytrees[2], inference is linear in *representation* size (i.e., in the number of entries across all CPTs)! In general, however, variable elimination 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 $X_1, X_2, \ldots, X_n$ and query $P(\boldsymbol{X} \mid \boldsymbol{E} = \boldsymbol{e})$
Output: Probability $P(\boldsymbol{X} \mid \boldsymbol{E} = \boldsymbol{e})$

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

[1] `https://fabian.damken.net/summaries/cs/elective/iws/dmml/dmml-summary.pdf`
[2] A 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 $P(\boldsymbol{X} \mid \boldsymbol{E} = \boldsymbol{e})$ (where $\boldsymbol{E}$ 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*.

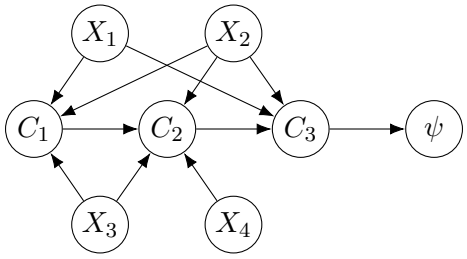
Finding (approximate) MPEs is quite straightforward by replacing the sums (marginalizations) in variable elimination by max's. In a second backward pass, the sums are replace by arg max-operators to find the configuration. A pseudocode version of this approach is given in algorithm 2.

4.4 Complexity of Conditional Queries

In general, inference on conditional query $P(\boldsymbol{X} \mid \boldsymbol{E} = \boldsymbol{e})$ 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 $X_1, X_2, \ldots, X_4$)?

$$\psi = (\neg X_1 \vee X_2 \vee X_3) \wedge (X_2 \vee \neg X_3 \vee X_4) \wedge (X_1 \vee \neg X_2) \tag{4.1}$$

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 $P(X_i = \mathrm{t}) = 0.5$ over all (boolean) RVs, satisfiability can be queried as $P(\psi = \mathrm{t}) > 0$. Hence, if conditional inference would be tractable, 3-SAT would be too. By contradiction, this shows that conditional inference is NP-hard. □

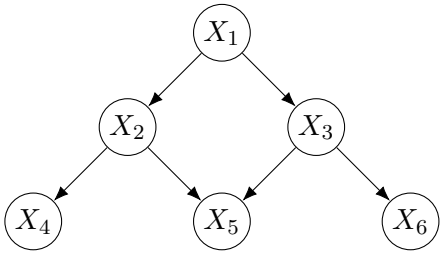
However, while NP-hard is already pretty hard, it can also be shown that inference is #P-complete involving counting all satisfying configurations.

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.

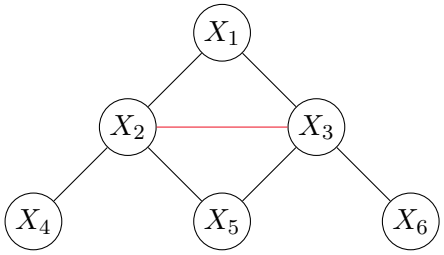
Algorithm 2: Most Probable Explanations via Variable Elimination

Input: Bayesian network with RVs $X_1, X_2, \ldots, X_n$ and evidence $\boldsymbol{E} = \boldsymbol{e}$
Output: Most probable explanation $\boldsymbol{x}^*$

Instantiate evidence $\boldsymbol{E} = \boldsymbol{e}$.
Prune non-active variables (w.r.t. $\boldsymbol{X} \cup \boldsymbol{E}$), giving $X_1, X_2, \ldots, X_{\tilde{n}}$, $\tilde{n} \leq n$.
Choose an ordering $X_1, X_2, \ldots, X_{\tilde{n}}$.
// Initialize factors:
$\mathcal{F}^{(0)} \leftarrow \{f_i = P(X_i \mid \mathsf{Pa}(X_i)) : i = 1, 2, \ldots, \tilde{n}\}$
for $i = 1, 2, \ldots, \tilde{n}$ do
    if $X_i \in \boldsymbol{E}$ then
        skip
    Collect factors $f_1, f_2, \ldots, f_k \in \mathcal{F}^{(i-1)}$ containing X_i.
    // Generate a new factor g by maximizing over X_i:
    $g = \mathsf{max}_{X_i} \prod_{i=1}^{k} f_j$
    // Replace factors in factor set:
    $\mathcal{F}^{(i)} \leftarrow (\mathcal{F}^{(i-1)} \setminus \{f_1, f_2, \ldots, f_k\}) \cup \{g\}$
// Initilize vector of MPE-values:
$\boldsymbol{x}^* \in \mathsf{val}(X_1) \times \mathsf{val}(X_2) \times \cdots \times \mathsf{val}(X_n)$
for $i = \tilde{n}, \tilde{n} - 1, \ldots, 1$ do
    if $X_i \in \boldsymbol{E}$ then
        // Copy evident value to result vector:
        $x_i^* \leftarrow e_i$
    else
        // Compute $\mathsf{arg\,max}$ over X_i, incorporating the evidence and already determined values:
        $x_i^* \leftarrow \mathsf{arg\,max}_{X_i} \prod_{f \in \mathcal{F}^{i-1}} f(\boldsymbol{x}_{1:i-1}^*, \boldsymbol{e})$
Set pruned variables $X_{\tilde{n}+1}, X_{\tilde{n}+2}, \ldots, X_n$ so arbitrary values.

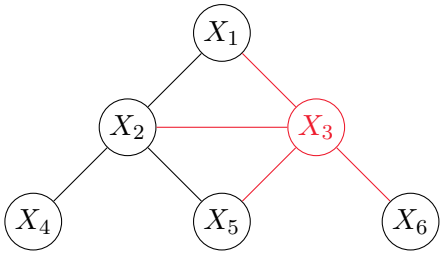


(a) Bayesian Network

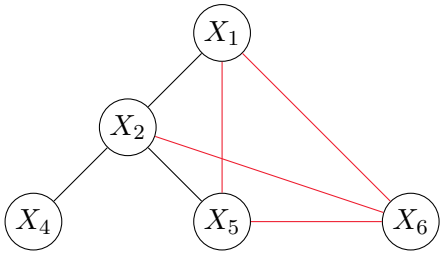


(b) Moral Graph

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.



(a) About to eliminate X_3. The affected node and edges are depicted in red.



(b) After eliminating X_3. The introduces fill-ins are depicted in red.

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

4.5 Variable Elimination in Moral Graphs

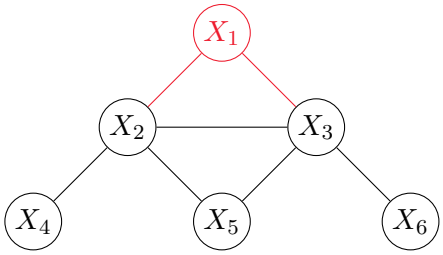
The induces moral graph of a Bayesian network is an undirected graph where the members of each clique correspond to the domain of a potential in the Bayesian network. To extract the moral graph from a Bayesian network, it might be necessary to introduce moral links to fulfill this (defining) criterion. A *moral link* is be introduced between unconnected parents of a node[3]. 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).

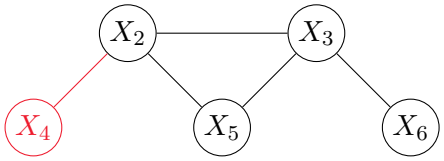
4.5.1 Complexity

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

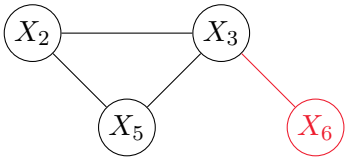
[3] The term "moralizing" has quite outdated origins and refers to the unmorality of "unmarried" parents.



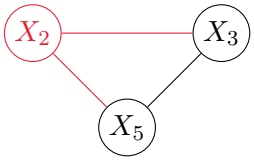
(a) About to eliminate X_1.



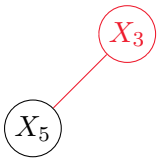
(b) After eliminating X_1, about to eliminate X_4.



(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.

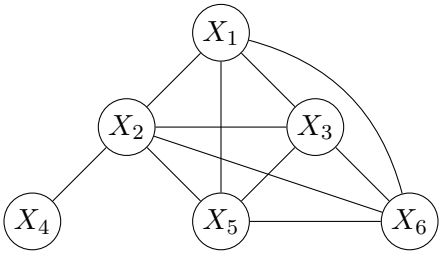


Figure 4.4: Induced graph for the imperfect elimination ordering $(X_3, X_4, X_6, X_1, X_2, X_5)$.

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:

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

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:

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

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.

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.

5 Markov Random Fields

5.1 Bayesian Networks as MRFs

5.2 Triangulated Graphs

5.3 Join Trees

5.4 Junction Trees

5.4.1 Collecting Evidence

5.4.2 Distributing Evidence

5.5 Non-Triangulated Graphs

6 Learning

6.1 Complete and Incomplete Data Sets

6.1.1 Hidden Variables

6.2 Parameter Estimation

6.2.1 Known Structure, Complete Data

Maximum Likelihood

Decomposability of the Likelihood

Likelihood for (Conditional) Bi- and Multinomials

6.2.2 Known Structure, Incomplete Data (Expectation-Maximization)

EM Idea

Complete-Data Likelihood

EM for (Conditional) Multinomials

Monotonicity

6.2.3 Gradient Ascent

6.2.4 Bayesian Parameter Estimation

Laplace Estimation

Bayesian Prediction

Conjugate Priors

Binomial Prior

Dirichlet Prior

Bayesian Networks and Bayesian Prediction

6.2.5 Summary

6.3 Structure Learning / Model Selection

6.3.1 Minimal I-Maps

6.3.2 Perfect Maps (P-Maps)

6.3.3 I-Equivalence

Skeleton and Immoralities

6.3.4 Obtaining a P-Map

Identifying the Skeleton

Identifying Immoralities

From Immoralities to Structures

6.3.5 Accurate Structures

6.3.6 Learning

Constrained-Based

Score-Based

Likelihood Score

Bayesian Score and Bayesian Information Criterion

6.3.7 Structure Search as Optimization

Learning Trees (Complete Data)

Heuristic (Local) Search

6.3.8 Structural EM

6.3.9 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 Generatie 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