Introduction to Probabilistic Graphical Models

Franz Pernkopf*, Robert Peharz*, Sebastian Tschiatschek*

Graz University of Technology, Laboratory of Signal Processing and Speech Communication
Inffeldgasse 16c, 8010 Graz, Austria
{pernkopf,robert.peharz,tschiatschek}@tugraz.at

Abstract

Over the last decades, probabilistic graphical models have become the method of choice for representing uncertainty in machine learning. They are used in many research areas such as computer vision, speech processing, time-series and sequential data modelling, cognitive science, bioinformatics, probabilistic robotics, signal processing, communications and error-correcting coding theory, and in the area of artificial intelligence.

This tutorial provides an introduction to probabilistic graphical models. We review three representations of probabilistic graphical models, namely, Markov networks or undirected graphical models, Bayesian networks or directed graphical models, and factor graphs. Then, we provide an overview about structure and parameter learning techniques. In particular, we discuss maximum likelihood and Bayesian learning, as well as generative and discriminative learning. Subsequently, we overview exact inference methods and briefly cover approximate inference techniques. Finally, we present typical applications for each of the three representations, namely, Bayesian networks for expert systems, dynamic Bayesian networks for speech processing, Markov random fields for image processing, and factor graphs for decoding error-correcting codes.

Contents

*These authors contributed equally to this work. This work was supported by the Austrian Science Fund (Project number P22488-N23) and (Project number S10610).

1 Introduction

Machine learning and pattern recognition are elementary building blocks of *intelligent systems*. The aim is to capture relevant phenomena hidden in empirical data using computational and statistical methods. Hence, the task is to automatically recognize and identify complex patterns in data which are further cast into a model. A prerequisite is that the learned model generalizes well in order to provide useful information for new data samples. To account for this, machine learning relies on many fields in science such as statistics and probability calculus, optimization methods, cognitive science, computer science, et cetera.

Learning can be divided into supervised, unsupervised, and reinforcement learning problems. For supervised learning the training data comprises of feature vectors which contain an abstract description of the object and their corresponding target or class label. If the desired target value is continuous, then this is referred to as *regression*. In the case where the input vector is assigned to a finite number of categories this is called *classification*. In unsupervised learning, the training data consists of a set of feature vectors without corresponding target value. The aim is to discover groups/clusters within the data or to model the distribution of the data, i.e. representations of the data are modeled. In *reinforcement* learning, an agent should perform actions in an environment so that it maximizes a long-term reward. The learning algorithm cannot access a finite set of labeled samples for training as in supervised classification and it must discover which actions of the agent result in an optimal long-term reward. Typically, the actions do not only affect an immediate reward but also have impact on future actions, i.e. taking the best reward at each time step is insufficient for the global optimal solution.

Probabilistic graphical models (PGMs) [Koller and Friedman, 2009] are important in all three learning problems and have turned out to be the method of choice for modeling uncertainty in many areas such as computer vision, speech processing, time-series and sequential data modelling, cognitive science, bioinformatics, probabilistic robotics, signal processing, communications and error-correcting coding theory, and in the area of artificial intelligence in general. One reason is that this common modelling framework allows to transfer concepts and ideas among different application areas. Another

reason stated in Pearl [1988] is that *common-sense reasoning is naturally embedded within the syntax of probability calculus*. PGMs combine probability theory and graph theory and are, therefore, well suited. They offer a compact graph-based representation of joint probability distributions exploiting conditional independencies among the random variables. Conditional independence assumptions alleviate the computational burden. Graphical models provide techniques to deal with two inherent problems throughout applied mathematics and engineering, namely, uncertainty and complexity [Jordan, 1999]. Many well-known statistical models, e.g. (dynamic) Bayesian networks, mixture models, factor analysis, hidden Markov models (HMMs), factorial HMMs, Kalman filters, Boltzmann machines, the Ising model, amongst others, can be represented by graphical models. The framework of graphical models provides techniques for inference (e.g. *sum product algorithm*, also known as *belief propagation or message passing*) [Pearl, 1988] and learning [Koller and Friedman, 2009] [1]. In this article, three popular representations of graphical models are presented: Markov networks (MNs) (also known as undirected graphical models (UGMs) or Markov random fields (MRFs), Bayesian networks (BNs) (also known as directed graphical models (DGMs) or belief networks) and factor graphs (FGs).

The research in PGMs has focused on two major fields which both are tightly connected to advanced optimization methods:

1. Learning of graphical models: Recent advances in structure learning are in finding structures satisfying some optimality conditions. This is in contrast to search heuristics which in general do not provide any guarantees. Current research in learning the parameters goes beyond classical maximum likelihood parameter learning which belongs to *generative learning*. Popular objectives are the conditional likelihood or a probabilistic formulation of the margin. Optimizing these objectives can lead to better classification performance, particularly when the class conditional distributions poorly approximate the true distribution [Bishop, 2006]. This is often referred to as *discriminative learning* with the aim of optimizing the model for one inference scenario, namely classification.

2. Efficient inference: Inference means answering queries using the PGM. In more technical language, inference deals with determining the marginal or most likely configuration of random variables given observed variables using the joint distribution modeled as PGM. Generally, exact inference is not tractable in large and dense graphical models and, therefore, approximate inference techniques are needed. Over the past decades, tractable approximate inference has been one of the most challenging and active research fields.

This tutorial is organized as follows: In Section 3 we present three types of representations, namely BNs, MNs, and FGs. Then in Section 4 and Section 5 different learning approaches and inference methods are discussed. In Section 6, we present typical examples for each of the three graphical model representations. Section 7 and Section 8 provide references to implementations and data sets. Finally, Section 9 concludes the tutorial providing with a brief perspective on advanced methods and future trends.

In this tutorial article we presume that the reader is familiar with elementary probability calculus and fundamental linear algebra concepts. This introduction to probabilistic graphical models is necessarily incomplete due to the vast amount of methods developed over the last decades. Nevertheless, we hope to provide the reader with an easily accessible and interesting introduction to the field of PGMs. For additional details on any of the covered subjects the interested reader is referred to the cited literature.

2 Preliminaries

In this section we introduce parts of the notation used throughout this tutorial. Further, we review the basics of probability theory and present elementary definitions from graph theory.

[1] In Bayesian statistics, learning and inference are the same, i.e. learning is inference of the parameters and structure.

2.1 Probability Theory

In this section we present a short review of the most important basic concepts from probability theory. For a more detailed introduction we refer the reader to [Papoulis and Pillai, 2002], [Cover and Thomas, 1991] and [Koller and Friedman, 2009]. We start with some basic definitions.

Definition 1 (Outcome Space, Elementary Events)**.** *An* outcome space Ω *is a non-empty set. The elements of* Ω *are called* elementary events*.*

Example 1 (Outcome Space, Elementary Events)**.** *We can define an outcome space of a die game according to* $\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6\}$*. The elementary event* ω_i *denotes the outcome "die shows number* i*".*

We further define the notion of an event space.

Definition 2 (Event Space, Events)**.** *Given an outcome space* Ω*, an* event space Σ *over* Ω *is a set of subsets of* Ω*, satisfying the following properties:*

- $\emptyset \in \Sigma$ *and* $\Omega \in \Sigma$*.*
- *If* $\sigma_1 \in \Sigma$ *and* $\sigma_2 \in \Sigma$*, then also* $\sigma_1 \cup \sigma_2 \in \Sigma$*.*
- *If* $\sigma \in \Sigma$*, then also* $\Omega \setminus \sigma \in \Sigma$*.*

The elements of Σ *are called* events*.* Ω *is called the* certain event *and* $\emptyset$ *is the* impossible event*.*

Example 2 (Event Space, Events)**.** *Given the outcome space of the die game* $\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6\}$ *of Example 1, we give three examples of possible event spaces:*

- *The event space* $\Sigma_1 = \{\emptyset, \{\omega_1, \omega_3, \omega_5\}, \{\omega_2, \omega_4, \omega_6\}, \Omega\}$ *contains the impossible event, the certain event, and the two events "outcome is odd" and "outcome is even".*
- *The events space* $\Sigma_2 = \{\emptyset, \{\omega_1, \omega_2, \omega_3\}, \{\omega_4, \omega_5, \omega_6\}, \Omega\}$ *contains the impossible event, the certain event, and the two events "outcome is smaller or equal* 3*" and "outcome is larger than* 3*".*
- *The event space* $\Sigma_3 = 2^\Omega$*, where* 2^Ω *is the power set of* Ω*, contains all subsets of* Ω*, i.e. all possible combinations of elementary events.*

We are now ready to define probability distributions.

Definition 3 (Probability Distribution, Probability Space)**.** *Let* Ω *be an outcome space and* Σ *an event space over* Ω*. A* probability distribution *over* (Ω, Σ) *is a function* $P : \Sigma \mapsto \mathbb{R}$ *satisfying the following properties:*

- $P(\sigma) \geq 0, \forall \sigma \in \Sigma$*.*
- $P(\Omega) = 1$*.*
- *If* $\sigma_1 \cap \sigma_2 = \emptyset$*, then* $P(\sigma_1 \cup \sigma_2) = P(\sigma_1) + P(\sigma_2)$*,* $\forall \sigma_1, \sigma_2 \in \Sigma$*.*

The triplet (Ω, Σ, P) *is called a* probability space*.*

Example 3 (Probability Distribution)**.** *Let* $\Sigma_3 = 2^\Omega$ *be the event space from Example 2. We can define a probability distribution by setting* $P(\omega_1) = P(\omega_2) = P(\omega_3) = P(\omega_4) = P(\omega_5) = P(\omega_6) = \frac{1}{6}$*, i.e. we assume a fair die. Since* P *has to be a probability distribution, it follows that* $P(\sigma) = \frac{|\sigma|}{6}$*,* $\forall \sigma \in \Sigma$*.*

Throughout the article, we use events and probability distributions defined via random variables, which are characterized by their cumulative distribution functions.

Definition 4 (Random Variable, Cumulative Distribution Function (CDF))**.** *Let (Ω, Σ, P) be a probability space and $\mathbb{S}$ a measurable space (in this tutorial we assume $\mathbb{S} = \mathbb{R}$). A* random variable *(RV) X is a function $X : \Omega \mapsto \mathbb{S}$, where "$X \leq x$" $= \{\omega \in \Omega : X(\omega) \leq x\}$ is an event for every $x \in \mathbb{S}$, and for the events "$X = -\infty$" $= \{\omega \in \Omega : X(\omega) = -\infty\}$ and "$X = \infty$" $= \{\omega \in \Omega : X(\omega) = \infty\}$ it must hold that $P(X = -\infty) = P(X = \infty) = 0$. The probability of the event "$X \leq x$" is the cumulative distribution function (CDF)*

$$F_X(x) = P(X \leq x). \tag{1}$$

More generally, let $\mathbf{X} = \{X_1, X_2, \ldots, X_N\}$ be an ordered set of N RVs and $\mathbf{x}$ a corresponding set of N values from $\mathbb{S}$. The joint *CDF of $\mathbf{X}$ is defined as*

$$F_{\mathbf{X}}(\mathbf{x}) = P(X_1 \leq x_1 \cap X_2 \leq x_2 \cap \cdots \cap X_N \leq x_N). \tag{2}$$

The image of an RV X is denoted as $\mathbf{val}(X) = \{x \in \mathbb{S} \,|\, \exists\, \omega \in \Omega : X(\omega) = x\}$. We say that X takes values out of $\mathbf{val}(X)$. Similarly, the set of values which can be assumed by a set of random variables $\mathbf{X}$ is denoted as $\mathbf{val}(\mathbf{X})$.

When we are only interested in events connected with RVs, the CDF fully characterizes the underlying probability distribution. If the CDF of an RV X is continuous, then we say that X is a *continuous* RV. For continuous RVs we define the probability density function.

Definition 5 (Probability Density Function (PDF))**.** *Let $F_X(x)$ be the CDF of a continuous RV X. The* probability density function *(PDF) of X is defined as*

$$P_X(x) = \frac{\mathrm{d}F_X(x)}{\mathrm{d}x}. \tag{3}$$

If $P_X(x)$ exists, then the CDF of X is given as

$$F_X(x) = \int_{-\infty}^{x} P_X(x)\mathrm{d}x. \tag{4}$$

More generally, let $F_{\mathbf{X}}(\mathbf{x})$ be the CDF of a set of continuous RVs $\mathbf{X}$. The joint *PDF of $\mathbf{X}$ is defined as*

$$P_{\mathbf{X}}(\mathbf{x}) = \frac{\partial^N F_{\mathbf{X}}(\mathbf{x})}{\partial x_1 \ldots \partial x_N}. \tag{5}$$

If $P_{\mathbf{X}}(\mathbf{x})$ exists, then the CDF of $\mathbf{X}$ is given as

$$F_{\mathbf{X}}(\mathbf{x}) = \int_{-\infty}^{x_1} \ldots \int_{-\infty}^{x_N} P_{\mathbf{X}}(\mathbf{x})\mathrm{d}x_1 \ldots \mathrm{d}x_N. \tag{6}$$

The PDF, if it exists, is an equivalent representation of the CDF, i.e. the PDF also represents the underlying probability distribution. It follows from the definition of probability distributions, that a PDF is nonnegative and integrates to one, i.e. $\mathrm{P}_{\mathbf{X}}(\mathbf{x}) \geq 0, \forall \mathbf{x}$, and $\int \mathrm{P}_{\mathbf{X}}(\mathbf{x})\mathrm{d}\mathbf{x} = 1$.

A RV is called *discrete* when the CDF of X is piecewise constant and has a finite number of discontinuities. For discrete RVs we define the probability mass function.

Definition 6 (Probability Mass Function (PMF))**.** *Let X be a discrete RV. The probability mass function (PMF) of X is a function $P_X : \mathbf{val}(X) \mapsto \mathbb{R}$ and defined as*

$$P_X(x) = P(X = x),\ x \in \mathbf{val}(X). \tag{7}$$

More generally, let $\mathbf{X}$ be a set of discrete RVs. The joint *PMF of $\mathbf{X}$ is a function $P_{\mathbf{X}} : \mathbf{val}(\mathbf{X}) \mapsto \mathbb{R}$ and defined as*

$$P_{\mathbf{X}}(\mathbf{x}) = P(X_1 = x_1 \cap X_2 = x_2 \cap \cdots \cap X_N = x_N),\ \mathbf{x} = \{x_1, x_2, \ldots, x_N\} \in \mathbf{val}(\mathbf{X}). \tag{8}$$

It follows from the definition of probability distributions, that a PMF is nonnegative and sums to one, i.e. $\mathrm{P}_{\mathbf{X}}(\mathbf{x}) \geq 0, \forall \mathbf{x} \in \mathbf{val}(\mathbf{X})$, and $\sum_{\mathbf{x} \in \mathbf{val}(\mathbf{X})} \mathrm{P}_{\mathbf{X}}(\mathbf{x}) = 1$. Unlike the PDF, a PMF always exists in the case of discrete RVs.

We use the same symbol $\mathrm{P}_{\mathbf{X}}(\mathbf{x})$ to denote PDFs and PMFs throughout the manuscript, since most of the discussion holds for both objects. From now on, we assume that $\mathrm{P}_{\mathbf{X}}(\mathbf{x})$ exists and fully represents the underlying probability distribution, and with slight abuse of notation, we refer to $\mathrm{P}_{\mathbf{X}}(\mathbf{x})$ itself as distribution. Usually we omit the subscripts of PDFs/PMFs, i.e. we use the shorthand $\mathrm{P}(\mathbf{x}) = \mathrm{P}_{\mathbf{X}}(\mathbf{x})$. Furthermore, we use the notation $\mathrm{P}(\mathbf{X})$ for unspecific values and $\mathrm{P}(\mathbf{x})$ for instantiations $\mathbf{x}$ of $\mathbf{X}$. Throughout this tutorial, we assume that $\mathbf{X} = \{X_1, \ldots, X_N\}$. Unless stated otherwise, the variables $X_1, \ldots, X_N$ are assumed to be discrete and the number of possible states of variable X_i is denoted as $\mathbf{sp}(X_i) = |\mathbf{val}(X_i)|$. Similarly, for a set of RVs $\mathbf{X}$, the total number of possible assignments is given as

$$\mathbf{sp}(\mathbf{X}) = \prod_{i=1}^{N} \mathbf{sp}(X_i). \tag{9}$$

When $\mathbf{Y}$ is a subset of $\mathbf{X}$, and $\mathbf{x}$ is a set of values for $\mathbf{X}$, then $\mathbf{x}(\mathbf{Y})$ denotes the set of values corresponding to $\mathbf{Y}$.

An important quantity of RVs is the expected value.

Definition 7 (Expected Value)**.** *Let X be an RV with distribution $P(X)$. The expected value of X is defined as*

$$\mathbb{E}[X] = \begin{cases} \sum_{x \in \mathbf{val}(X)} x\, P(x) & \text{if } X \text{ is discrete} \\ \int_{\mathbf{val}(X)} x\, P(x) \mathrm{d}x & \text{if } X \text{ is continuous.} \end{cases} \tag{10}$$

More generally, the expected value of a function $f : \mathbf{val}(X) \mapsto \mathbb{R}$ is defined as

$$\mathbb{E}[f(X)] = \begin{cases} \sum_{x \in \mathbf{val}(X)} f(x)\, P(x) & \text{if } X \text{ is discrete} \\ \int_{\mathbf{val}(X)} f(x)\, P(x) \mathrm{d}x & \text{if } X \text{ is continuous.} \end{cases} \tag{11}$$

Similarly, we define expectations over sets of RVs.

When the joint distribution $\mathrm{P}(\mathbf{X})$ is given, one is often interested in the distribution $\mathrm{P}(\mathbf{Y})$ of a subset $\mathbf{Y} \subseteq \mathbf{X}$ of RVs. Also, one often considers the distributions of subsets $\mathbf{Y} \subseteq \mathbf{X}$, conditioned on the values of the remaining variables $\mathbf{Z} = \mathbf{X} \setminus \mathbf{Y}$.

Definition 8 (Marginal Distribution, Conditional Distribution)**.** *Let $P(\mathbf{X})$ be a distribution over a set of RVs $\mathbf{X}$. Further let $\mathbf{Y} \subseteq \mathbf{X}$, and $\mathbf{Z} = \mathbf{X} \setminus \mathbf{Y}$, i.e. $\mathbf{X} = \mathbf{Y} \cup \mathbf{Z}$ and $P(\mathbf{X}) = P(\mathbf{Y}, \mathbf{Z})$. Assuming discrete RVs, the* marginal distribution *$P(\mathbf{Y})$ over $\mathbf{Y}$ is given as*

$$P(\mathbf{Y}) = \sum_{\mathbf{z} \in \mathbf{val}(\mathbf{Z})} P(\mathbf{Y}, \mathbf{z}), \tag{12}$$

and the conditional distribution *$P(\mathbf{Y}|\mathbf{Z})$ over $\mathbf{Y}$, conditioned on $\mathbf{Z}$, is given as*

$$P(\mathbf{Y}|\mathbf{Z}) = \frac{P(\mathbf{Y}, \mathbf{Z})}{P(\mathbf{Z})} = \frac{P(\mathbf{Y}, \mathbf{Z})}{\sum_{\mathbf{y} \in \mathbf{val}(\mathbf{Y})} P(\mathbf{y}, \mathbf{Z})}. \tag{13}$$

In the case of continuous RVs, summations are replaced by integration in a straightforward manner.

The definition of the marginal and conditional distribution leads to an important rule, which allows us to invert the "direction of conditioning". Using (13), it is easily seen that for two sets of discrete RVs $\mathbf{Y}$ and $\mathbf{Z}$ the following holds:

$$\mathrm{P}(\mathbf{Y}, \mathbf{Z}) = \mathrm{P}(\mathbf{Y}, \mathbf{Z}) \tag{14}$$

$$\mathrm{P}(\mathbf{Y}|\mathbf{Z})\mathrm{P}(\mathbf{Z}) = \mathrm{P}(\mathbf{Z}|\mathbf{Y})\mathrm{P}(\mathbf{Y}) \tag{15}$$

$$\mathrm{P}(\mathbf{Y}|\mathbf{Z}) = \frac{\mathrm{P}(\mathbf{Z}|\mathbf{Y})\mathrm{P}(\mathbf{Y})}{\mathrm{P}(\mathbf{Z})} = \frac{\mathrm{P}(\mathbf{Z}|\mathbf{Y})\mathrm{P}(\mathbf{Y})}{\sum_{\mathbf{y} \in \mathbf{val}(\mathbf{Y})} \mathrm{P}(\mathbf{Z}|\mathbf{y})\mathrm{P}(\mathbf{y})}. \tag{16}$$

Equation (16) is known as the *Bayes' rule* or the rule of *inverse probability*. For continuous RVs, the summation is replaced by an integral.

From the conditional distribution follows the concept of statistical independence. Informally, two RVs X and Y are independent, if knowing the state of one variable, i.e. conditioning on this variable, does not change the distribution over the other variable. Formally, statistical independence and conditional statistical independence are defined as follows.

Definition 9 (Statistical Independence, Conditional Statistical Independence)**.** *Let* $\mathbf{X}$*,* $\mathbf{Y}$ *and* $\mathbf{Z}$ *be mutually disjoint sets of RVs.* $\mathbf{X}$ *and* $\mathbf{Y}$ *are* conditionally statistically independent *given* $\mathbf{Z}$*, annotated as* $\mathbf{X} \perp \mathbf{Y}|\mathbf{Z}$*, if and only if the following equivalent conditions hold:*

- $P(\mathbf{X}|\mathbf{Y},\mathbf{Z}) = P(\mathbf{X}|\mathbf{Z})$.
- $P(\mathbf{Y}|\mathbf{X},\mathbf{Z}) = P(\mathbf{Y}|\mathbf{Z})$.
- $P(\mathbf{X},\mathbf{Y}|\mathbf{Z}) = P(\mathbf{X}|\mathbf{Z})P(\mathbf{Y}|\mathbf{Z})$.

For the special case $\mathbf{Z} = \emptyset$*, we say that* $\mathbf{X}$ *and* $\mathbf{Y}$ *are* statistically independent*, annotated as* $\mathbf{X} \perp \mathbf{Y}$*.*

Furthermore, using conditional distributions, there are many ways to factorize an arbitrary joint distribution $\mathrm{P}(\mathbf{X})$, i.e. any joint probability over $\mathbf{X} = \{X_1, \ldots, X_N\}$ can be written as

$$\mathrm{P}(\mathbf{X}) = \mathrm{P}(X_N|X_{N-1}, X_{N-2}, \ldots, X_1)\,\mathrm{P}(X_{N-1}|X_{N-2}, \ldots, X_1)\,\ldots\,\mathrm{P}(X_2|X_1)\,\mathrm{P}(X_1) \tag{17}$$

$$= \mathrm{P}(X_1)\prod_{i=2}^{N}\mathrm{P}(X_i|X_i - 1, \ldots, X_1). \tag{18}$$

This is called the *chain rule of probability*. The chain rule holds for any variable order, i.e. for any re-labeling of the variables in $\mathbf{X}$.

2.2 Graph Theory

In the context of PGMs *graphs* are used to represent probability distributions in an intuitive and easily readable way. Formally, a graph is defined as follows:

Definition 10 (Graph, Vertex, Edge)**.** *A* graph $\mathcal{G} = (\mathbf{X}, \mathbf{E})$ *is a tuple consisting of a set of* vertices*, also denoted as* nodes*,* $\mathbf{X}$ *and a set of* edges $\mathbf{E}$*.*

In PGMs each RV corresponds to exactly one node, and vice versa. That is, there is a one-to-one relationship between RVs and nodes. Therefore, we use these terms equivalently. Throughout this tutorial, we assume that $\mathbf{X} = \{X_1, \ldots, X_N\}$.

The edges of the graph define specific relationships between these variables. Any two nodes X_i and X_j, $i \neq j$, in a graph can be connected by either a *directed* or an *undirected* edge. An undirected edge between node i and j is denoted as $(X_i - X_j)$, and a directed edge from node X_i to X_j as $(X_i \to X_j)$. While an undirected edge is a symmetric relationship, i.e. the edge $(X_i - X_j)$ is the same as the edge $(X_j - X_i)$, a directed edge represents an asymmetric relationship. Note that we do not allow these two types of edges to exist between any pair of nodes simultaneously and that there must not be an edge connecting some node with itself.

Depending on the types of edges in a graph we define the following three graph types:

Definition 11 (Directed, Undirected and Mixed Graph)**.** *A graph* $\mathcal{G} = (\mathbf{X}, \mathbf{E})$ *is* directed *(*undirected*) if all edges* $e \in \mathbf{E}$ *are directed (undirected). Further, a graph is said to be a* mixed graph *if the set of edges contains undirected and directed edges.*

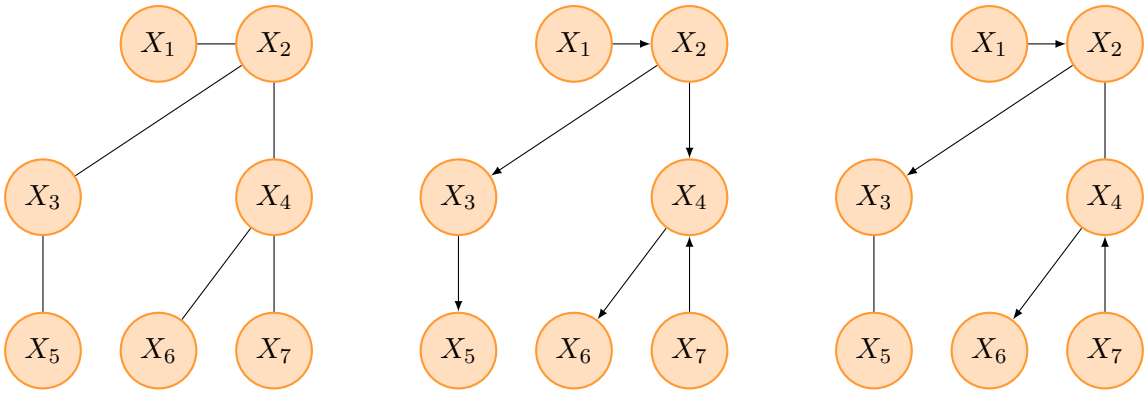
At this point we want to introduce a graphical representation of PGMs: every node is represented by a circle and edges are depicted as lines, in the case of undirected edges, or as arrows, in the case of directed edges, connecting the corresponding nodes.

Example 4 (Graphical representation of graphs)**.** *Consider Figure 1. For all shown graphs* $\mathcal{G} = (\mathbf{X}, \mathbf{E})$ *the set of nodes is* $\mathbf{X} = \{X_1, \ldots, X_7\}$*. The corresponding edge sets are*

(a) $\mathbf{E} = \{(X_1 - X_2), (X_2 - X_3), (X_2 - X_4), (X_3 - X_5), (X_4 - X_6), (X_4 - X_7)\}$,

(b) $\mathbf{E} = \{(X_1 \to X_2), (X_2 \to X_3), (X_2 \to X_4), (X_3 \to X_5), (X_4 \to X_6), (X_7 \to X_4)\}$*, and*

(c) $\mathbf{E} = \{(X_1 \to X_2), (X_2 \to X_3), (X_2 - X_4), (X_3 - X_5), (X_4 \to X_6), (X_7 \to X_4)\}$.



(a) Undirected graph. (b) Directed graph. (c) Mixed graph.

Figure 1: Graphical representation of different graph types.

In graphs we can define relationships between variables. In directed and mixed graphs we introduce a parent-child relationship:

Definition 12 (Parent and Child)**.** *Let $\mathcal{G} = (\mathbf{X}, \mathbf{E})$ be a directed or mixed graph and $X_i, X_j \in \mathbf{X}, i \neq j$. If $(X_i \to X_j) \in \mathbf{E}$, then X_i is a* parent *of X_j and X_j is a* child *of X_i. The set $\mathbf{Pa}_{\mathcal{G}}(X_j)$ consists of all parents of X_j and the set $\mathbf{Ch}_{\mathcal{G}}(X_i)$ contains all children of X_i.*

Example 5 (Parents and Children)**.** *In Figure 1(b) X_3 is a child of X_2 and X_2 is a parent of X_3. Further, the set $\mathbf{Pa}_{\mathcal{G}}(X_3) = \{X_2\}$, as the only parent of X_3 is X_2. The set of all children of X_2 is $\mathbf{Ch}_{\mathcal{G}}(X_2) = \{X_3, X_4\}$.*

For undirected and mixed graphs we define the notion of neighborhood:

Definition 13 (Neighbor)**.** *Let $\mathcal{G} = (\mathbf{X}, \mathbf{E})$ be an undirected or mixed graph and $X_i, X_j \in \mathbf{X}, i \neq j$. If $(X_i - X_j) \in \mathbf{E}$, then X_i is said to be a neighbor of X_j. The set of all neighbors of X_i is denoted as $\mathbf{Nb}_{\mathcal{G}}(X_i)$, i.e.*

$$\mathbf{Nb}_{\mathcal{G}}(X_i) = \{X_j : (X_i - X_j) \in \mathbf{E}, X_j \in \mathbf{X}\}. \tag{19}$$

Example 6 (Neighborhood)**.** *In Figure 1(a) X_1, X_3 and X_4 are neighbors of X_2. As these are all neighbors of X_2 in the graph,*

$$\mathbf{Nb}_{\mathcal{G}}(X_2) = \{X_1, X_3, X_4\}. \tag{20}$$

While edges represent *direct* relationships between two nodes, the notion of *paths* and *trails* describes *indirect* relationships across several nodes:

Definition 14 (Path, Directed Path, Trail)**.** *Let $\mathcal{G} = (\mathbf{X}, \mathbf{E})$ be a graph. A sequence of nodes $Q = (X_1, \ldots, X_n)$, with $X_1, \ldots, X_n \in \mathbf{X}$ is a* path *from X_1 to X_n if*

$$\forall\, i \in \{1, \ldots, n-1\} : (X_i \to X_{i+1}) \in \mathbf{E} \vee (X_i - X_{i+1}) \in \mathbf{E}. \tag{21}$$

A path is directed *if*

$$\exists\, i \in \{1, \ldots, n-1\} : (X_i \to X_{i+1}) \in \mathbf{E}, \tag{22}$$

i.e. if there is at least one directed edge on the path.
The sequence of nodes Q is a trail *if*

$$\forall\, i \in \{1, \ldots, n-1\} : (X_i \to X_{i+1}) \in \mathbf{E} \vee (X_{i+1} \to X_i) \in \mathbf{E} \vee (X_i - X_{i+1}) \in \mathbf{E}, \tag{23}$$

i.e. if Q is a path from X_1 to X_n replacing all directed edges by undirected edges.

Example 7 (Paths)**.** *Again, consider Figure 1.*

(a) In Figure 1(a), the sequence (X_2, X_4, X_6) is a path from X_2 to X_6, and there exists a path between all pairs of distinct nodes.

(b) In Figure 1(b), (X_1, X_2, X_4) is a directed path from X_1 to X_4, but there does not exist a path from X_6 to X_7 because of the directed edges. But, X_6 and X_7 are connected by the trail (X_6, X_4, X_7).

(c) In Figure 1(c), the sequence of nodes (X_7, X_4, X_2) is a directed path, but again, there does not exist a path from X_6 to X_7.

An important class of undirected graphs are *trees*:

Definition 15 (Tree)**.** *An undirected graph $\mathcal{G}$ is a* tree *if any two distinct nodes are connected by exactly one path*

Example 8 (Tree)**.** *Figure 1(a) represents a tree.*

In directed graphs we can identify parts of the graph that appear *before* and *after* a certain node:

Definition 16 (Ancestor, Descendant)**.** *In a directed graph $\mathcal{G} = (\mathbf{X}, \mathbf{E})$ with $X_i, X_j \in \mathbf{X}, i \neq j$, the node X_i is an* ancestor *of X_j if there exists a directed path from X_i to X_j. In this case, X_j is also denoted as a* descendant *of X_i.*
The set of all ancestors of some node $X_j \in \mathbf{X}$ is denoted as $\boldsymbol{Anc}_{\mathcal{G}}(X_j)$, i.e.

$$\boldsymbol{Anc}_{\mathcal{G}}(X_j) = \{X_i | X_i \text{ is an ancestor of } X_j\}. \tag{24}$$

Similarly, the set of all descendants of some node $X_i \in \mathbf{X}$ is denoted as $\boldsymbol{Desc}_{\mathcal{G}}(X_i)$, i.e.

$$\boldsymbol{Desc}_{\mathcal{G}}(X_i) = \{X_j | X_j \text{ is a descendant of } X_i\}. \tag{25}$$

We further define the nondescendants of X_i as

$$\boldsymbol{NonDesc}_{\mathcal{G}}(X_i) = \mathbf{X} \setminus (\boldsymbol{Desc}_{\mathcal{G}}(X_i) \cup \{X_i\} \cup \mathbf{Pa}_{\mathcal{G}}(X_i)), \tag{26}$$

i.e. the nondescendants of X_i are all nodes in the graph that are neither descendants of X_i, nor the node X_i itself or the parents $\mathbf{Pa}_{\mathcal{G}}(X_i)$ of this node.

A special type of paths are *directed cycles*:

Definition 17 (Directed cycle)**.** *Let $\mathcal{G} = (\mathbf{X}, \mathbf{E})$ be a graph, $X_i \in \mathbf{X}$, and let P be a directed path from X_i to X_i. Then P is denoted as a* directed cycle*.*

Using these notion, we can define an important type of directed graphs:

Definition 18 (Directed acyclic graphs)**.** *A graph $\mathcal{G} = (\mathbf{X}, \mathbf{E})$ is a* directed acyclic graph *(DAG) if it contains no directed cycles.*

Example 9 (DAG)**.** *An example of a DAG is given in Figure 1(b).*

Sometimes, we will consider graphs that are *contained* in a *larger* graph:

Definition 19 (Subgraph, Supergraph)**.** *The graph $\mathcal{G} = (\mathbf{X}, \mathbf{E})$ is a* subgraph *of $\mathcal{G}' = (\mathbf{X}', \mathbf{E}')$, if and only if $\mathbf{X} \subseteq \mathbf{X}'$ and $\mathbf{E} \subseteq \mathbf{E}'$. Further, $\mathcal{G}'$ is a* supergraph *of $\mathcal{G}$ if and only if $\mathcal{G}$ is a subgraph of $\mathcal{G}'$.*

For some of our considerations we need to identify special subsets of the nodes of an undirected graph:

Definition 20 (Clique, Maximal Clique)**.** *In an undirected graph $\mathcal{G} = (\mathbf{X}, \mathbf{E})$ a set of nodes $\mathbf{C} \subseteq \mathbf{X}$ is a* clique *if there exists an edge between all pairs of nodes in the subset $\mathbf{C}$, i.e. if*

$$\forall C_i, C_j \in \mathbf{C}, i \neq j : (C_i - C_j) \in \mathbf{E}. \tag{27}$$

The clique $\mathbf{C}$ is a maximal clique *if adding any node $X \in \mathbf{X} \setminus \mathbf{C}$ makes it no longer a clique.*

Example 10 (Cliques, Maximal Cliques)**.** *Consider Figure 2. The set of nodes* $\{X_2, X_3, X_4\}$ *is a maximal clique as* $\{X_1, X_2, X_3, X_4\}$ *is no clique. The set* $\{X_1, X_3\}$ *is a clique that is not maximal as* $\{X_1, X_2, X_3\}$ *is also a clique.*

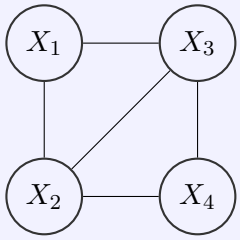


Figure 2: Clique and maximal clique.

The notion of cliques is closely related to that of *complete* (sub)graphs:

Definition 21 (Complete Graph)**.** *The undirected graph* $\mathcal{G} = (\mathbf{X}, \mathbf{E})$ *is* complete *if every pair of distinct nodes is connected by an edge, i.e. if*

$$\forall X_i, X_j \in \mathbf{X}, i \neq j : (X_i - X_j) \in \mathbf{E}. \tag{28}$$

That is, the set $\mathbf{C} \subseteq \mathbf{X}$ is a clique if the *induced* graph $\mathcal{G}' = (\mathbf{C}, \mathbf{E}')$ is complete, where

$$\mathbf{E}' = \{(X_i - X_j) \in \mathbf{E} : X_i, X_j \in \mathbf{C}, i \neq j\}. \tag{29}$$

At this end, we introduce the notion of a topological ordering of the nodes in a directed graph:

Definition 22 (Topological Ordering)**.** *Let* $\mathcal{G} = (\mathbf{X}, \mathbf{E})$ *be a directed graph and* $\mathbf{X} = \{X_1, \ldots, X_N\}$*. The nodes* $\mathbf{X}$ *are* topologically ordered *if for any edge* $(X_i \to X_j) \in \mathbf{E}$ *also* $i \leq j$*.*

Example 11. *The nodes of the graph in Figure 3 are topologically ordered, while the nodes of the graph in Figure 1(b) are not.*

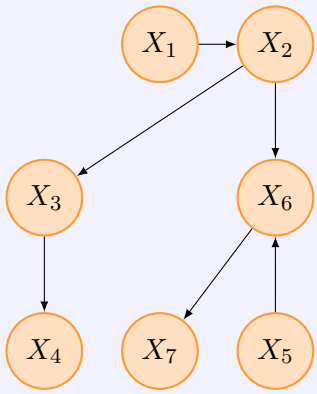


Figure 3: Topologically ordered graph.

3 Representations

In this section we introduce three types of PGMs, namely *Bayesian networks* in Section 3.1, *Markov networks* in Section 3.2 and *factor graphs* in Section 3.3. Each type has its specific advantages and disadvantages, captures different aspects of probabilistic models, and is preferred in a certain application domain. Finally, we conclude the section by an example demonstrating how Markov chains can be represented in each of the three types of PGMs.

3.1 Bayesian Networks (BNs)

Bayesian networks are directed graphical models. Furthermore, there is the notion of *dynamic BNs* which means that the BN is unrolled over time. This essentially means that at each time slice the BN has the same structure [Bilmes, 2000]. There are two equivalent ways to define BNs, namely (a) by *factorization properties*, or (b) by *conditional independencies*. We introduce BNs by their factorization properties and show that this is equivalent to a set of conditional independence assumptions in Section 3.1.2.

3.1.1 Definition and Factorization Properties

A BN is defined as follows:

Definition 23 (Bayesian Network)**.** *A* Bayesian network $\mathcal{B}$ *is a tuple* $(\mathcal{G}, \{P_{X_1}, \ldots, P_{X_N}\})$*, where* $\mathcal{G} = (\mathbf{X}, \mathbf{E})$ *is a DAG, each node* X_i *corresponds to an RV, and* $P_{X_1}, \ldots, P_{X_N}$ *are conditional probability distributions (CPDs) associated with the nodes of the graph. Each of these CPDs has the form*

$$P_{X_i}(X_i|\mathbf{Pa}_{\mathcal{G}}(X_i)). \tag{30}$$

The BN $\mathcal{B}$ *defines the joint probability distribution* $P_{\mathcal{B}}(X_1, \ldots, X_N)$ *according to*

$$P_{\mathcal{B}}(X_1, \ldots, X_N) = \prod_{i=1}^{N} P_{X_i}(X_i|\mathbf{Pa}_{\mathcal{G}}(X_i)). \tag{31}$$

We will use the shorthand notation $\mathrm{P}(X_i|\mathbf{Pa}_{\mathcal{G}}(X_i))$ for $\mathrm{P}_{X_i}(X_i|\mathbf{Pa}_{\mathcal{G}}(X_i))$ when the meaning is clear from the context. Usually each conditional probability distribution P_{X_i} is specified by a parameter set $\boldsymbol{\theta}^i$, and we collect all the parameters into the set $\boldsymbol{\Theta} = \{\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^N\}$ and use the terms $\mathcal{B} = (\mathcal{G}, \{\mathrm{P}_{X_1}, \ldots, \mathrm{P}_{X_N}\})$ and $\mathcal{B} = (\mathcal{G}, \boldsymbol{\Theta})$ equivalently. Similarly, in the case of parameterized distributions, we write $\mathrm{P}(X_i|\mathbf{Pa}_{\mathcal{G}}(X_i), \boldsymbol{\theta}^i)$ for $\mathrm{P}_{X_i}(X_i|\mathbf{Pa}_{\mathcal{G}}(X_i))$.

In the sequel, we give two commonly used examples of parametric BNs, one with discrete nodes, and one with conditional Gaussian probabilities.

Example 12 (Discrete Variables)**.** *Assume a BN* $\mathcal{B}$ *with discrete variables* $X_1, \ldots, X_N$*, where each* X_i *takes one out of a finite set of states. For simplicity, we identify the states of* X_i *with natural numbers in* $\{1, \ldots, \mathbf{sp}(X_i)\}$*. In this case, the conditional distributions* $P(X_i|\mathbf{Pa}_{\mathcal{G}}(X_i), \boldsymbol{\theta}^i)$ *are PMFs, parameterized by* $\boldsymbol{\theta}^i$ *such that*

$$P(X_i = j|\mathbf{Pa}_{\mathcal{G}}(X_i) = \mathbf{h}, \boldsymbol{\theta}^i) = \theta^i_{j|\mathbf{h}}, \tag{32}$$

where $\mathbf{h}$ *is the instantiation of the parent variables* $\mathbf{Pa}_{\mathcal{G}}(X_i)$ *and* j *the instantiation of* X_i*. Note that the parameters* $\boldsymbol{\theta}^i$ *must satisfy*

$$\theta^i_{j|\mathbf{h}} \geq 0, \quad \forall\, j \in \mathbf{val}\,(X_i)\,, \mathbf{h} \in \mathbf{val}\,(\mathbf{Pa}_{\mathcal{G}}(X_i)) \;\; \textit{and} \tag{33}$$

$$\sum_{j=1}^{\mathbf{sp}(X_i)} \theta^i_{j|\mathbf{h}} = 1, \quad \forall\, \mathbf{h} \in \mathbf{val}\,(\mathbf{Pa}_{\mathcal{G}}(X_i))\,, \tag{34}$$

in order to specify valid probability distributions. Therefore, the CPDs corresponding to the i^{th} *variable can be stored in a table with* $(\mathbf{sp}(X_i) - 1) \cdot \mathbf{sp}(\mathbf{Pa}_{\mathcal{G}}(X_i))$ *entries.*

The BN distribution in Equation (31) *becomes*

$$P_{\mathcal{B}}(\mathbf{X} = \mathbf{x}) = \prod_{i=1}^{N} \prod_{j=1}^{\mathbf{sp}(X_i)} \prod_{\mathbf{h} \in \mathbf{val}(\mathbf{Pa}_{\mathcal{G}}(X_i))} \left(\theta^i_{j|\mathbf{h}}\right)^{u^i_{j|\mathbf{h}}}, \tag{35}$$

where $u^i_{j|\mathbf{h}} = \mathbb{1}_{\{x_i = j \;\textit{and}\; \mathbf{x}(\mathbf{Pa}_{\mathcal{G}}(X_i)) = \mathbf{h}\}}$*. The symbol* $\mathbb{1}_{\{A\}}$ *denotes the indicator function which equals* 1 *if the statement* A *is true and* 0 *otherwise.*

Example 13 (Conditional Gaussian Variables)**.** *In this example we assume that the variables of a BN $\mathcal{B}$ are real-valued and that the CPDs are specified by Gaussian distributions, with the mean depending linearly on the value of the parents, and shared variance σ^2. That is, the CPD of node X_i, is given as*

$$P(X_i|\mathbf{Pa}_{\mathcal{G}}(X_i), \boldsymbol{\theta}^i) = \mathcal{N}\left(X_i \middle| \sum_{X_k \in \mathbf{Pa}_{\mathcal{G}}(X_i)} \alpha_{i,k}\, x_k + \beta_i; \sigma^2\right), \tag{36}$$

where

$$\mathcal{N}(X|\mu;\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(X-\mu)^2}{2\sigma^2}\right) \tag{37}$$

denotes the Gaussian distribution with mean μ and variance σ^2. Here, the parameter set $\boldsymbol{\theta}^i = \{\boldsymbol{\alpha}_i, \beta_i\}$ contains the linear weights $\boldsymbol{\alpha}_i = (\alpha_{i,1}, \ldots, \alpha_{i,|\mathbf{Pa}_{\mathcal{G}}(X_i)|})$ and the bias β_i. It can be shown that in this case the BN distribution in Equation (31) *is a multivariate Gaussian distribution, with mean vector $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_N)$ and covariance matrix $\boldsymbol{\Sigma}$, which are recursively given as [Bishop, 2006]:*

$$\mu_i = \mathbb{E}[X_i] = \sum_{X_k \in \mathbf{Pa}_{\mathcal{G}}(X_i)} \alpha_{i,k}\, \mu_k + \beta_i, \tag{38}$$

$$\begin{aligned}\Sigma_{i,j} &= \mathbb{E}\left[(X_i - \mathbb{E}[X_i])\,(X_j - \mathbb{E}[X_j])\right] \\ &= \sum_{X_k \in \mathbf{Pa}_{\mathcal{G}}(X_j)} \alpha_{j,k}\, \Sigma_{i,k} + I_{i,j}\, \sigma^2,\end{aligned} \tag{39}$$

where I denotes the identity matrix, $\Sigma_{i,j}$ the entry in $\boldsymbol{\Sigma}$ in the i^{th} row and j^{th} column.

3.1.2 Conditional Independence Properties

The factorization properties of the joint distribution of BNs as given in the last section imply a set of conditional independencies [Koller and Friedman, 2009]:

Theorem 1 (Local Conditional Independencies)**.** *Let $\mathcal{B} = (\mathcal{G}, \{P_{X_1}, \ldots, P_{X_N}\})$ be a BN and $P_{\mathcal{B}}$ the joint probability defined by $\mathcal{B}$. Then, $P_{\mathcal{B}}$ satisfies the* local independencies

$$X_i \perp \boldsymbol{NonDesc}_{\mathcal{G}}(X_i) | \mathbf{Pa}_{\mathcal{G}}(X_i) \quad \forall i, \tag{40}$$

i.e. any variable X_i is conditional independent of its nondescendants given its parents.

Proof. To prove the stated conditional independence property, it suffices to show that

$$\mathrm{P}_{\mathcal{B}}(X_i|\mathbf{NonDesc}_{\mathcal{G}}(X_i), \mathbf{Pa}_{\mathcal{G}}(X_i)) = \mathrm{P}_{\mathcal{B}}(X_i|\mathbf{Pa}_{\mathcal{G}}(X_i)). \tag{41}$$

The conditional distribution $\mathrm{P}(X_i|\mathbf{NonDesc}_{\mathcal{G}}(X_i), \mathbf{Pa}_{\mathcal{G}}(X_i))$ can be written as

$$\mathrm{P}_{\mathcal{B}}(X_i|\mathbf{NonDesc}_{\mathcal{G}}(X_i), \mathbf{Pa}_{\mathcal{G}}(X_i)) = \frac{\mathrm{P}_{\mathcal{B}}(X_i, \mathbf{NonDesc}_{\mathcal{G}}(X_i), \mathbf{Pa}_{\mathcal{G}}(X_i))}{\mathrm{P}_{\mathcal{B}}(\mathbf{NonDesc}_{\mathcal{G}}(X_i), \mathbf{Pa}_{\mathcal{G}}(X_i))}. \tag{42}$$

The numerator is

$$\mathrm{P}_{\mathcal{B}}(X_i, \mathbf{NonDesc}_{\mathcal{G}}(X_i), \mathbf{Pa}_{\mathcal{G}}(X_i)) = \sum_{\mathbf{Desc}_{\mathcal{G}}(X_i)} \mathrm{P}_{\mathcal{B}}(X_i, \mathbf{NonDesc}_{\mathcal{G}}(X_i), \mathbf{Pa}_{\mathcal{G}}(X_i), \mathbf{Desc}_{\mathcal{G}}(X_i)) \tag{43}$$

$$= \sum_{\mathbf{Desc}_{\mathcal{G}}(X_i)} \mathrm{P}_{\mathcal{B}}(X_i|\mathbf{Pa}_{\mathcal{G}}(X_i)) \prod_{X_j \in \mathbf{Desc}_{\mathcal{G}}(X_i)} \mathrm{P}_{\mathcal{B}}(X_j|\mathbf{Pa}_{\mathcal{G}}(X_j)) \quad \times$$

$$\prod_{X_k \in \mathbf{NonDesc}_{\mathcal{G}}(X_i) \cup \mathbf{Pa}_{\mathcal{G}}(X_i)} \mathrm{P}_{\mathcal{B}}(X_k|\mathbf{Pa}_{\mathcal{G}}(X_k)) \tag{44}$$

$$= \mathrm{P}_{\mathcal{B}}(X_i|\mathbf{Pa}_{\mathcal{G}}(X_i)) \underbrace{\prod_{X_k \in \mathbf{NonDesc}_{\mathcal{G}}(X_i) \cup \mathbf{Pa}_{\mathcal{G}}(X_i)} \mathrm{P}_{\mathcal{B}}(X_k|\mathbf{Pa}_{\mathcal{G}}(X_k))}_{=A} \quad \times$$

$$\underbrace{\sum_{\mathbf{Desc}_{\mathcal{G}}(X_i)} \prod_{X_j \in \mathbf{Desc}_{\mathcal{G}}(X_i)} \mathrm{P}_{\mathcal{B}}(X_j|\mathbf{Pa}_{\mathcal{G}}(X_j))}_{=1}. \tag{45}$$

$$\tag{46}$$

In the same way, the denominator is given as

$$\mathrm{P}_{\mathcal{B}}(\mathbf{NonDesc}_{\mathcal{G}}(X_i), \mathbf{Pa}_{\mathcal{G}}(X_i)) = \underbrace{\prod_{X_k \in \mathbf{NonDesc}_{\mathcal{G}}(X_i) \cup \mathbf{Pa}_{\mathcal{G}}(X_i)} \mathrm{P}_{\mathcal{B}}(X_k|\mathbf{Pa}_{\mathcal{G}}(X_k))}_{=A} \tag{47}$$

Consequently, we obtain

$$\mathrm{P}_{\mathcal{B}}(X_i|\mathbf{NonDesc}_{\mathcal{G}}(X_i), \mathbf{Pa}_{\mathcal{G}}(X_i)) = \mathrm{P}_{\mathcal{B}}(X_i|\mathbf{Pa}_{\mathcal{G}}(X_i)) \tag{48}$$

and the desired statement follows. □

Theorem 1 could have also been used as a basis for the definition of BNs, as, without proof, the local conditional independence assumptions from the Theorem imply the factorization properties of $\mathrm{P}_{\mathcal{B}}$ given in Equation (31).

Additionally to the local conditional independencies, several other conditional independencies hold in BNs. These can be read off the graph by the concept of *d*-separation:

Definition 24 (*d*-separation [Pearl, 1988])**.** *Let $\mathcal{B} = (\mathcal{G}, \{P_{X_1}, \ldots, P_{X_N}\})$ be a BN, $\mathcal{G} = (\mathbf{X}, \mathbf{E})$, and $\mathbf{X} = \{X_1, \ldots, X_N\}$. Two RVs X_i and X_j, $i \neq j$, are* d-separated *if for all trails between X_i and X_j there is an intermediate variable X_k, $i \neq k, j \neq k$, such that*

- *the connection is* serial *or* diverging *and the state of X_k is observed, or*
- *the connection is* converging *and neither the state of X_k nor the state of any descendant of X_k is observed,*

where the term connection means the path corresponding to the trail.

We now introduce three simple graphs illustrating the underlying concept of the definition, i.e. the different connection types and the thereby implied conditional independencies.

Serial connection. A BN $\mathcal{B}$ showing a so called *serial connection* is presented in Figure 4. According to Equation (31) the corresponding joint distribution is given as

$$\mathrm{P}_{\mathcal{B}}(X_i, X_k, X_j) = \mathrm{P}(X_i)\mathrm{P}(X_k|X_i)\mathrm{P}(X_j|X_k). \tag{49}$$

We now condition the joint probability on X_k, i.e. we assume X_k to be observed. Then, by probability calculus,

$$\mathrm{P}_{\mathcal{B}}(X_i, X_j|X_k) = \frac{\overbrace{\mathrm{P}(X_i)\mathrm{P}(X_k|X_i)}^{=\mathrm{P}_{\mathcal{B}}(X_i, X_k)}\mathrm{P}(X_j|X_k)}{\mathrm{P}_{\mathcal{B}}(X_k)} \tag{50}$$

$$= \mathrm{P}_{\mathcal{B}}(X_i|X_k)\mathrm{P}(X_j|X_k), \tag{51}$$

where $\mathrm{P}_{\mathcal{B}}(X_i, X_j|X_k)$, $\mathrm{P}_{\mathcal{B}}(X_i, X_k)$, and $\mathrm{P}_{\mathcal{B}}(X_i|X_k)$ are conditional and marginal distributions resulting from the joint distribution $\mathrm{P}_{\mathcal{B}}(X_i, X_k, X_j)$. That is, X_i and X_j are conditionally independent given X_k, i.e. $X_i \perp X_j|X_k$. However, if X_k is not observed, the RVs X_i and X_j are dependent in general.

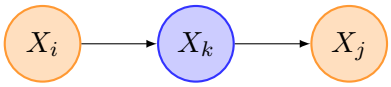


Figure 4: Serial connection.

Diverging connection. A diverging connection is shown in Figure 5. Again, by exploiting the factorization properties of the BN according to (31), we obtain the joint probability

$$\mathrm{P}_{\mathcal{B}}(X_i, X_k, X_j) = \mathrm{P}(X_k)\mathrm{P}(X_i|X_k)\mathrm{P}(X_j|X_k). \tag{52}$$

Conditioning this probability on X_k yields

$$\begin{aligned}\mathrm{P}_{\mathcal{B}}(X_i, X_j|X_k) &= \frac{\overbrace{\mathrm{P}(X_k)\mathrm{P}(X_i|X_k)}^{=\mathrm{P}_{\mathcal{B}}(X_i,X_k)}\mathrm{P}(X_j|X_k)}{\mathrm{P}_{\mathcal{B}}(X_k)} &(53)\\ &= \mathrm{P}_{\mathcal{B}}(X_i|X_k)\mathrm{P}(X_j|X_k). &(54)\end{aligned}$$

Hence, $X_i \perp X_j|X_k$. That is, when X_k is observed, then the variables X_i and X_j are conditionally independent given X_k, while, when X_k is not observed, they are dependent in general.

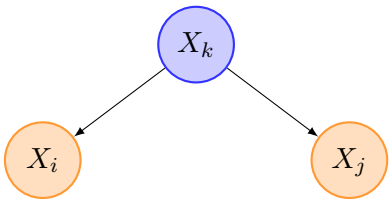


Figure 5: Diverging connection.

Converging connection. A converging connection is illustrated in Figure 6. The represented joint distribution is given as

$$\mathrm{P}_{\mathcal{B}}(X_i, X_k, X_j) = \mathrm{P}(X_i)\mathrm{P}(X_j)\mathrm{P}(X_k|X_i, X_j). \tag{55}$$

In contrast to the two cases before, the RVs X_i and X_j are a-priori independent since

$$\sum_{x_k \in \mathbf{val}(X_k)} \mathrm{P}(X_K = x_k|X_i, X_j) = 1. \tag{56}$$

That is

$$\mathrm{P}_{\mathcal{B}}(X_i, X_j) = \mathrm{P}(X_i)\mathrm{P}(X_j), \tag{57}$$

which can be stated as $X_i \perp X_j$. To this end, consider the probability $\mathrm{P}_{\mathcal{B}}(X_i, X_j|X_k)$ which can be computed as

$$\mathrm{P}_{\mathcal{B}}(X_i, X_j|X_k) = \frac{\mathrm{P}(X_i)\mathrm{P}(X_j)\mathrm{P}(X_k|X_i, X_j)}{\mathrm{P}_{\mathcal{B}}(X_k)}. \tag{58}$$

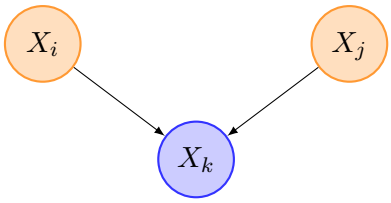


Figure 6: Converging connection.

In general, this does not factorize in probabilities over X_i and X_j. Hence, the RVs X_i and X_j are independent if X_k is not observed but become dependent upon observation of variable X_k (or any descendant of X_k). This is known as the *Berkson's paradox* [Berkson, 1946] or the *explaining away phenomenon*.

Without proof, the d-separation Theorem expresses the conditional independence properties that follow from two sets of variables being d-separated (a proof is provided in Lauritzen [1996]):

Theorem 2 (d-separation [Pearl, 1988])**.** *Let $\mathcal{B} = (\mathcal{G}, \{P_{X_1}, \ldots, P_{X_N}\})$ be a BN and $\mathcal{G} = (\mathbf{X}, \mathbf{E})$. Furthermore, let $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{D}$ be mutually disjoint subsets of $\mathbf{X}$, where the variables in $\mathbf{D}$ are observed. If every pair of nodes $A \in \mathbf{A}$ and $B \in \mathbf{B}$ are d-separated, then*

$$\mathbf{A} \perp \mathbf{B} | \mathbf{D}, \tag{59}$$

that is $\mathbf{A}$ and $\mathbf{B}$ are conditionally independent given $\mathbf{D}$.

3.2 Markov Networks (MNs)

Markov networks, also known as *Markov Random Fields* (MRFs) or *undirected graphical models*, are represented by undirected graphs. As in BNs, an MN encodes certain factorization and conditional independence properties of the joint probability distribution.

3.2.1 Definition and Factorization Properties

Definition 25 (Markov Network)**.** *A* Markov network *is a tuple $\mathcal{M} = (\mathcal{G}, \{\Psi_{\mathbf{C}_1}, \ldots, \Psi_{\mathbf{C}_L}\})$, where $\mathcal{G} = (\mathbf{X}, \mathbf{E})$ is an undirected graph with maximal cliques $\mathbf{C}_1, \ldots, \mathbf{C}_L$. The nodes $\mathbf{X}$ correspond to RVs and $\Psi_{\mathbf{C}_i} : \mathbf{val}(\mathbf{C}_i) \to \mathbb{R}_+$ are nonnegative functions, called* factors *or* potentials*, over the maximal cliques of the graph $\mathcal{G}$.*

The MN defines a probability distribution according to

$$P_{\mathcal{M}}(X_1, \ldots, X_N) = \frac{1}{Z} \prod_{l=1}^{L} \Psi_{\mathbf{C}_l}(\mathbf{C}_l), \tag{60}$$

where Z is a normalization constant given as

$$Z = \sum_{\mathbf{x} \in \mathbf{val}(\mathbf{X})} \prod_{l=1}^{L} \Psi_{\mathbf{C}_l}(\mathbf{x}(\mathbf{C}_l)). \tag{61}$$

Often, Z is also referred to as partition function.

Example 14 (Markov Network)**.** *Consider the MN shown in Figure 7. The maximal cliques in this graph are surrounded by dotted boxes and are given as*

$$\mathbf{C}_1 = \{X_1, X_2, X_3\}, \tag{62}$$
$$\mathbf{C}_2 = \{X_3, X_4\}, \text{ and} \tag{63}$$
$$\mathbf{C}_3 = \{X_3, X_5\}. \tag{64}$$

Hence, the joint probability distribution of the represented model is

$$P_{\mathcal{M}}(X_1, \ldots, X_5) = \frac{1}{Z}\Psi_{\mathbf{C}_1}(X_1, X_2, X_3)\Psi_{\mathbf{C}_2}(X_3, X_4)\Psi_{\mathbf{C}_3}(X_3, X_5). \tag{65}$$

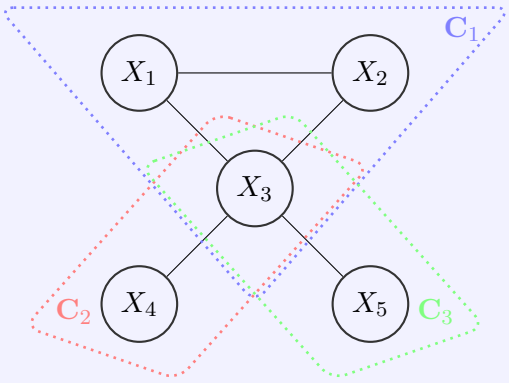


Figure 7: Example of an MN.

3.2.2 Conditional Independence Properties

Similarly as in BNs, MNs define a set of conditional independencies. In the following we present three conditional independence properties implied by MNs with strictly positive factors. If the factors are not strictly positive, the conditional independence properties stated below are not equivalent in general. Details regarding this are provided in [Koller and Friedman, 2009].

The joint probability distributions represented by MNs with strictly positive factors satisfy the following equivalent conditional independencies:

(i) **Local Markov property.** Any RV X_i is conditionally independent from all other RVs given its neighbors. Formally,

$$X_i \perp (\mathbf{X} \setminus (\mathbf{Nb}_{\mathcal{G}}(X_i) \cup \{X_i\}))|\mathbf{Nb}_{\mathcal{G}}(X_i). \tag{66}$$

(ii) **Pairwise Markov property.** Any two non-adjacent RVs are conditionally independent given all other RVs, i.e.

$$X_i \perp X_j|(\mathbf{X} \setminus \{X_i, X_j\}), \quad \text{if } (i-j) \notin \mathbf{E}. \tag{67}$$

(iii) **Global Markov property.** Let $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{D}$ be mutually disjoint subsets of $\mathbf{X}$, and the variables in $\mathbf{D}$ are observed. Then,

$$\mathbf{A} \perp \mathbf{B}|\mathbf{D}, \tag{68}$$

if every path from any node in $\mathbf{A}$ to any node in $\mathbf{B}$ passes through $\mathbf{D}$. In this case, the set $\mathbf{D}$ is also denoted as *separating subset*.

We illustrate these conditional independence statements in an example.

Example 15 (Conditional Independencies in an MN)**.** *Again, consider the MN shown in Figure 7. Then, for example,*

(a) by the local Markov property*, X_1 is conditionally independent from the RVs X_4 and X_5 given its neighbors X_2 and X_3, i.e.*

$$X_1 \perp \{X_4, X_5\} | \{X_2, X_3\}. \tag{69}$$

This is illustrated in Figure 8(a).

(b) by the pairwise Markov property*, X_4 and X_5 are conditionally independent given X_1, X_2 and X_3. Formally,*

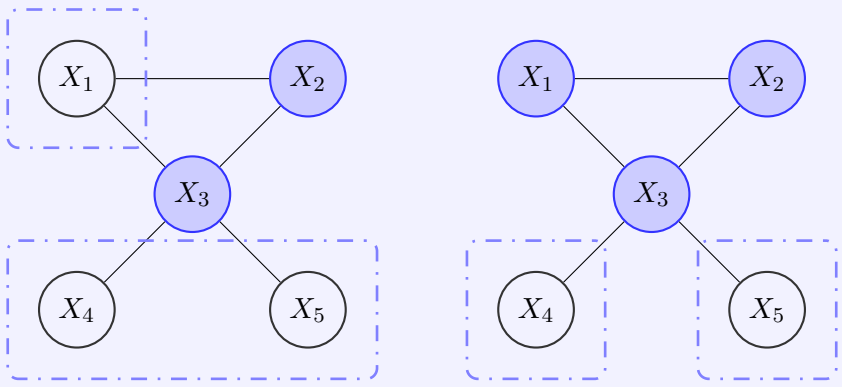
$$X_4 \perp X_5 | \{X_1, X_2, X_3\}. \tag{70}$$

This is illustrated in Figure 8(b).

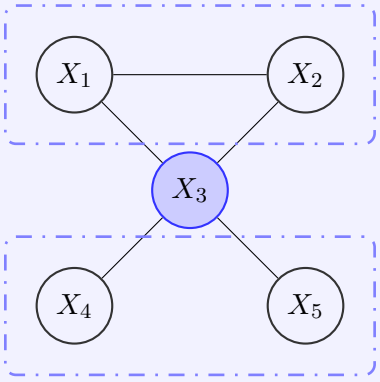
(c) by the global Markov property*, X_1 and X_2 are conditionally independent of X_4 and X_5 given X_3, i.e.*

$$\{X_1, X_2\} \perp \{X_4, X_5\} | X_3. \tag{71}$$

This is illustrated in Figure 8(c).



(a) Local Markov property. (b) Pairwise Markov property.



(c) Global Markov property.

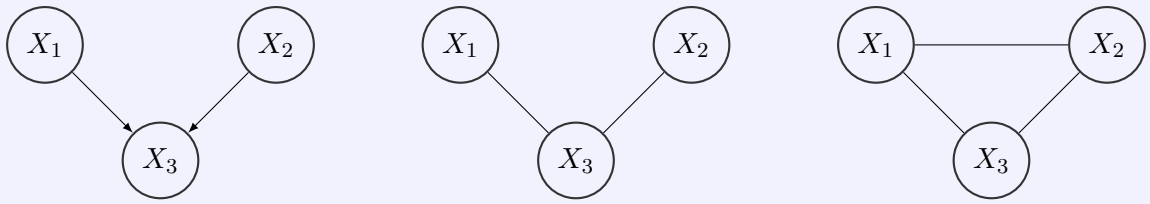
Figure 8: Conditional independencies in MNs.

Difference between BNs and MRFs. As introduced in the last sections, both BNs and MNs imply certain conditional independence statements. One might ask if any BN and its conditional independence properties can be represented by an MN and vice versa. Two simple examples demonstrate, that this is not the case:

Example 16 (Conditional Independencies in BNs and MNs)**.** *Consider the PGM with RVs X_1, X_2 and X_3 represented by the BN in Figure 9(a). We want to represent this BN by an MN that captures all the conditional independence properties of the BN:*

- *An appropriate MN must contain the edges $(X_1 - X_3)$ and $(X_2 - X_3)$. Otherwise, X_3 would not depend on the state of X_1 and X_2 as in the BN. Such an MN is shown in Figure 9(b). However, while in the BN in general $X_1 \not\perp X_2 | X_3$ the MN satisfies $X_1 \perp X_2 | X_3$ by the pairwise Markov property.*
- *Hence, we additionally need to add the edge $(X_1 - X_2)$ to the MN resulting in the MN represented in Figure 9(c). However, now the MN does in general not satisfy that $X_1 \perp X_2$ while the BN does.*

Consequently, the conditional independencies of the BN in Figure 9(a) cannot be represented by an MN.



(a) BN to be represented as an MN. (b) Candidate MN. (c) Another candidate MN.

Figure 9: Representation of conditional independencies in BNs by MNs.

Example 17 (Conditional Independencies in BNs and MNs)**.** *Consider the MN in Figure 10. We want to represent this MN by a BN capturing all the independence properties that hold in the MN. By the pairwise Markov property X_1 is conditionally independent from X_4 given X_2 and X_3, and X_2 is conditionally independent from X_3 given X_1 and X_4, i.e.*

$$X_1 \perp X_4 | \{X_2, X_3\}, \text{ and} \tag{72}$$
$$X_2 \perp X_3 | \{X_1, X_4\}. \tag{73}$$

However, these conditional independencies cannot be encoded in a BN. The interested reader can find a proof for this in Koller and Friedman [2009].

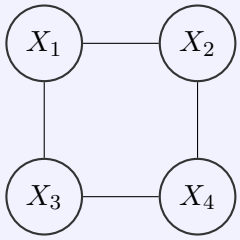


Figure 10: Representation of conditional independencies in MNs by BNs.

Nevertheless, there exist distributions which can be represented by both MNs and BNs. An example is the Markov chain, cf. Section 3.4. This can be illustrated graphically by the notion of *perfect maps*. A graph is a perfect map for a probability distribution if it represents all conditional independence properties of the distribution, and vice versa. Now, consider probability distributions over a given set of variables. Then, the set of probability distributions $\{\mathrm{P}_{\mathcal{B}}\}$ for which some BN is a perfect map is a subset of all probability distributions $\{\mathrm{P}\}$. Similarly, the set of probability distributions $\{\mathrm{P}_{\mathcal{M}}\}$ for which some MN is a perfect map is also a subset of all probability distributions. Additionally, there are probability distributions $\{\mathrm{P}_{\mathcal{B}}\} \cap \{\mathrm{P}_{\mathcal{M}}\}$ for which both BNs and MNs are perfect maps. This is illustrated by the Venn diagram in Figure 11 [Bishop, 2006].

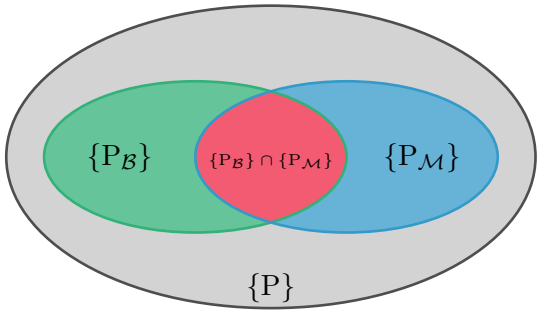


Figure 11: Venn diagram depicting the set of all probability distributions {P} over a given set of variables, and the sets of probability distributions {P$_\mathcal{B}$} and {P$_\mathcal{M}$} for which some BN or some MN is a perfect map, respectively.

3.3 Factor Graphs (FGs)

As last type of probabilistic model representation we introduce *factor graphs* (FGs) which explicitly represent the factorization properties of a function and have initially been used in the coding and image processing community.

3.3.1 Definition

FGs are defined using the notion of bipartite graphs. These are graphs $\mathcal{G} = (\mathbf{X}, \mathbf{E})$ for which the set of nodes can be divided into two disjoint subsets such that there are only edges between nodes from one subset to nodes of the other subset, as follows:

Definition 26 (Factor Graph)**.** *A* factor graph *is a tuple* $\mathcal{F} = (\mathcal{G}, \{f_1, \ldots, f_L\})$*, where*

- $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ *is an undirected bipartite graph such that* $\mathbf{V} = \mathbf{X} \cup \mathbf{F}$*, where* $\mathbf{X}$*, and* $\mathbf{F}$ *are disjoint, and the nodes* $\mathbf{X} = \{X_1, \ldots, X_N\}$ *correspond to RVs. The nodes* $\mathbf{X}$ *are* variable *nodes, while the nodes* $\mathbf{F}$ *are* factor *nodes.*
- *Further,* $f_1, \ldots, f_L$ *are positive functions and the number of functions* L *equals the number of nodes in* $\mathbf{F} = \{F_1, \ldots, F_L\}$*. Each node* F_i *is associated with the corresponding function* f_i *and each* f_i *is a function of the neighboring variables of* F_i*, i.e.* $f_i = f_i(\mathbf{Nb}_\mathcal{G}(F_i))$*.*

The factor graph $\mathcal{F}$ *defines the joint probability distribution*

$$P_\mathcal{F}(X_1, \ldots, X_N) = \frac{1}{Z} \prod_{l=1}^{L} f_l(\mathbf{Nb}_\mathcal{G}(F_l)), \tag{74}$$

where Z *is the normalization constant given as*

$$Z = \sum_{\mathbf{x} \in \mathbf{val}(\mathbf{X})} \prod_{l=1}^{L} f_l(\mathbf{x}(\mathbf{Nb}_\mathcal{G}(F_l))). \tag{75}$$

As each factor node corresponds to a factor we use these both terms equivalently. We illustrate the definition of FGs by the following example:

Example 18 (Factor Graph). *Consider Figure 12. The given factor graph $\mathcal{F}$ is $\mathcal{F} = (\mathcal{G}, \{f_1, \dots, f_4\})$, where the nodes of $\mathcal{G}$ are the union of the set of variable nodes $\{X_1, \dots, X_4\}$ and the set of factor nodes $\{F_1, \dots, F_4\}$. The functions corresponding to these factor nodes are $f_1(X_1, X_2)$, $f_2(X_1, X_3)$, $f_3(X_2, X_3)$, and $f_4(X_3, X_4)$. The represented joint probability density is*

$$P_{\mathcal{F}}(X_1, \dots, X_4) = \frac{1}{Z} f_1(X_1, X_2) f_2(X_1, X_3) f_3(X_2, X_3) f_4(X_3, X_4), \tag{76}$$

where Z is the normalization constant as given in the definition of FGs.

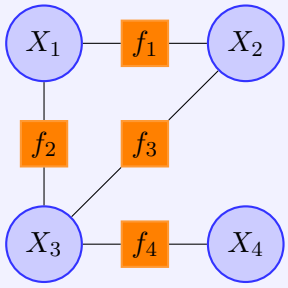


Figure 12: Example of an FG.

3.3.2 BNs and MNs as FGs

FGs allow the representation of both BNs and MNs. However, note that there is in general no unique transformation from a BN/MN to an FG. We illustrate the transformation of a BN or MN to an FG in the following two examples:

Example 19 (Conversion of a BN to an FG). *The BN shown in Figure 13(a) can be represented as the FG in Figure 13(b) where the factors f_1, f_2, f_3 are given as*

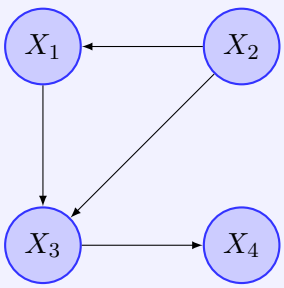
$$f_1(X_1, X_2) = P_{X_2}(X_2) P_{X_1}(X_1|X_2), \tag{77}$$
$$f_2(X_1, X_2, X_3) = P_{X_3}(X_3|X_1, X_2) \tag{78}$$
$$f_3(X_3, X_4) = P_{X_4}(X_4|X_3). \tag{79}$$

In this way, the factor graph represents the probability distribution

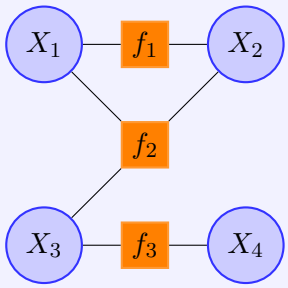
$$P_{\mathcal{F}}(X_1, \dots, X_4) = f_1(X_1, X_2) f_2(X_1, X_2, X_3) f_3(X_3, X_4) \tag{80}$$
$$= P_{X_2}(X_2) P_{X_1}(X_1|X_2) P_{X_3}(X_3|X_1, X_2) P_{X_4}(X_4|X_3), \tag{81}$$

which is exactly the one given by the BN. Alternatively, the BN could also be represented by the FG shown in Figure 13(c), where the factor f is given as

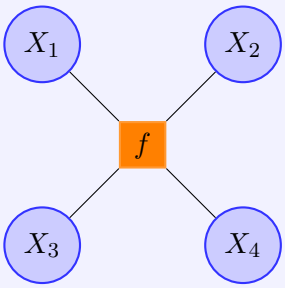
$$f(X_1, \dots, X_4) = P_{X_2}(X_2) P_{X_1}(X_1|X_2) P_{X_3}(X_3|X_1, X_2) P_{X_4}(X_4|X_3). \tag{82}$$



(a) BN to be converted to an FG.



(b) FG for representing the BN.



(c) Alternative FG for representing the BN.

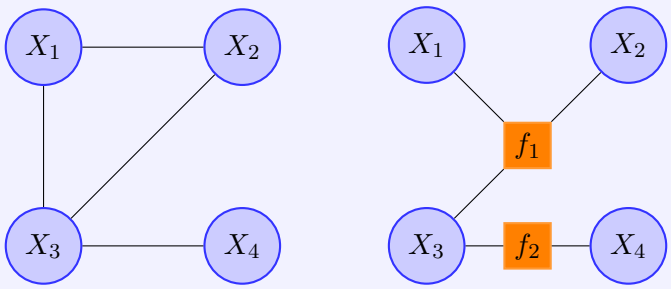
Figure 13: Conversion of a BN to an FG.

Example 20 (Conversion of an MN to an FG)**.** *The MN in Figure 14(a) represents the factorization*

$$P_{\mathcal{M}}(X_1, \ldots, X_4) = \Psi_{\mathbf{C}_1}(X_1, X_2, X_3)\Psi_{\mathbf{C}_2}(X_3, X_4), \tag{83}$$

where the maximal cliques are $\mathbf{C}_1 = \{X_1, X_2, X_3\}$ *and* $\mathbf{C}_2 = \{X_3, X_4\}$*, respectively. This factorization can be represented by the FG in Figure 14(b), where*

$$f_1(X_1, X_2, X_3) = \Psi_{\mathbf{C}_1}(X_1, X_2, X_3), \text{ and} \tag{84}$$
$$f_2(X_3, X_4) = \Psi_{\mathbf{C}_2}(X_3, X_4). \tag{85}$$



(a) MN to be converted to an FG. (b) FG for representing the MN.

Figure 14: Conversion of an MN to an FG.

3.4 Markov Chains

To conclude the section on PGM representations we consider Markov chains (MCs), and how they can be represented in each of the three types of PGMs. An MC is defined as follows:

Definition 27 (Markov chain)**.** *Let* $X_1, \ldots, X_N$ *be a sequence of RVs. These RVs form a* Markov chain *if they satisfy*

$$P(X_i|X_1, \ldots, X_{i-1}) = P(X_i|X_{i-1}), \quad \forall i. \tag{86}$$

This is called Markov property*.*

In an MC the RVs preceding some given RV X_i are conditionally independent of all RVs succeeding X_i, i.e.

$$\{X_1, \ldots, X_{i-1}\} \perp \{X_{i+1}, \ldots, X_N\}|X_i. \tag{87}$$

Furthermore, since any joint probability distribution $\mathrm{P}(X_1, \ldots, X_N)$ factorizes as

$$\mathrm{P}(X_1, \ldots, X_N) = \prod_{i=1}^{N} \mathrm{P}(X_i|X_{i-1}, \ldots, X_1), \tag{88}$$

this simplifies due to the Markov property to

$$\mathrm{P}(X_1, \ldots, X_N) = \prod_{i=1}^{N} \mathrm{P}(X_i|X_{i-1}). \tag{89}$$

We can represent MCs as BNs, MNs and FGs, as presented in the following:

- **Bayesian network.** A BN $\mathcal{B}$ modeling an MC is shown in Figure 15. The represented joint distribution is

$$\mathrm{P}_{\mathcal{B}}(X_1, \ldots, X_N) = \prod_{i=1}^{N} \mathrm{P}(X_i|\mathbf{Pa}_{\mathcal{G}}(X_i)) \tag{90}$$

$$= \mathrm{P}_{\mathcal{B}}(X_1, \ldots, X_N) = \mathrm{P}(X_1) \prod_{i=2}^{N} \mathrm{P}(X_i|X_{i-1}), \tag{91}$$

where the only parent of X_i is X_{i-1}.

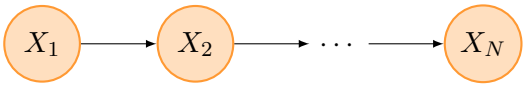


Figure 15: BN modeling an MC.

- **Markov network.** The MN $\mathcal{M}$ for representing an MC in Figure 16 has the maximal cliques $\mathbf{C}_i = \{X_i, X_{i+1}\}$ for $i = 1, \ldots, N-1$. If we define the factor $\Psi_{\mathbf{C}_i}$ to represent the function $\mathrm{P}(X_{i+1}|X_i)$ for $i = 2, \ldots, N-1$ and $\Psi_{\mathbf{C}_1} = \mathrm{P}(X_1)\mathrm{P}(X_2|X_1)$, then the MN specifies the joint probability distribution of an MC, i.e.

$$\mathrm{P}_{\mathcal{M}}(X_1, \ldots, X_N) = \prod_{l=1}^{N-1} \Psi_{\mathbf{C}_l}(\mathbf{C}_l) \tag{92}$$

$$= \mathrm{P}(X_1) \prod_{i=2}^{N} \mathrm{P}(X_i|X_{i-1}). \tag{93}$$

Figure 16: MN modeling an MC.

- **Factor graph.** Finally, the FG $\mathcal{F}$ in Figure 17 with factors $f_1 = \mathrm{P}(X_1)$ and $f_i = \mathrm{P}(X_i|X_{i-1})$ for $i = 2, \ldots, N$ specifies the distribution of an MC, i.e.

$$\mathrm{P}_{\mathcal{F}}(X_1, \ldots, X_N) = \prod_{l=1}^{N} f_l(\mathbf{Nb}_{\mathcal{G}}(F_l)) \tag{94}$$

$$= \mathrm{P}(X_1) \prod_{i=2}^{N} \mathrm{P}(X_i|X_{i-1}). \tag{95}$$

Figure 17: FG modeling an MC.

4 Learning

Our goal is now to determine a PGM which models the variables of interest. One possibility is to specify the PGM by hand. Especially BNs are suited to be specified by a domain expert, since the directed edges can be treated as causal directions and local probability tables can be elicited relatively easily. However, the more attractive approach might be to automatically determine a PGM from a set of training data. Assume we want to find a PGM for a set of random variables $\mathbf{X} = \{X_1, \ldots, X_N\}$ which are distributed according to an unknown joint distribution $\mathrm{P}^*(\mathbf{X})$. Further, assume that we have a collection of L independent and identically distributed (i.i.d.) samples $\mathcal{D} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(L)}\}$, drawn from distribution $\mathrm{P}^*(\mathbf{X})$. In this article we assume *complete* data, i.e. each data case $\mathbf{x}^{(i)}$ contains values of all variables in $\mathbf{X}$. For incomplete data, learning becomes much harder and is typically tackled using *expectation-maximization* methods [Dempster et al., 1977], or related techniques. The task of learning a PGM can be described as follows: given $\mathcal{D}$, return a PGM which approximates $\mathrm{P}^*(\mathbf{X})$

best. This type of learning is known as *generative learning*, since we aim to model the process which generated the data. Generative learning can be seen as a "multi-purpose" tool, since we directly strive for the primary object of interest: the joint distribution $\mathrm{P}^*(\mathbf{X})$. All other quantities of the model which might be interesting, such as marginal distributions of single variables/sets of variables, conditional distributions, or expectations, can be derived from the joint distribution. However, it is rarely possible to exactly retrieve $\mathrm{P}^*(\mathbf{X})$, especially when using a finite sample $\mathcal{D}$. This can be obstructive, when the ultimate goal is not to retrieve $\mathrm{P}^*(\mathbf{X})$, but to use the PGM in a specialized way. An example for this is when we want to use the PGM as classifier, i.e. we want to infer the value of a class variable, given the values of the remaining variables. In the classification context we are primarily interested in minimizing the loss resulting from erroneous decisions, not in modeling the overall generative process. It can be easily shown that knowledge of $\mathrm{P}^*(\mathbf{X})$ leads to optimal classification, and therefore generative learning seems to be suitable to learn classifiers. However, a somewhat surprising but consistently occurring phenomenon in machine learning literature is that so-called *discriminative* learning methods, which refrain from modeling $\mathrm{P}^*(\mathbf{X})$, but directly address the classification problem, often yield better classification results than the generative approach. The discriminative paradigm is discussed in Section 4.3. For now, let us focus on the generative approach.

4.1 Principles of Generative Learning

Let us assume a specific class of models $\mathcal{C}$, e.g. the class of all BNs, or the class of all MNs. We define a prior distribution $\mathrm{P}(\mathcal{C})$ over the model class, which represents our preference for certain models. For example, if $\mathcal{C}$ is the class of all BNs, we may prefer networks with fewer edges, and therefore define a prior which decreases with the number of edges. Furthermore, let $\mathcal{D}$ be a set of i.i.d. training samples drawn from the true distribution $\mathrm{P}^*(\mathbf{X})$. In the generative approach, we are interested in the posterior probability distribution over the model class, conditioned on the given data. Using Bayes' law, this distribution is

$$\mathrm{P}(\mathcal{C}|\mathcal{D}) = \frac{\mathrm{P}(\mathcal{D}|\mathcal{C})\,\mathrm{P}(\mathcal{C})}{\mathrm{P}(\mathcal{D})} = \frac{\mathrm{P}(\mathcal{D}|\mathcal{C})\,\mathrm{P}(\mathcal{C})}{\int_{\mathcal{C}} \mathrm{P}(\mathcal{D}|M')\;\mathrm{P}(M')\,\mathrm{d}M'} \propto \mathrm{P}(\mathcal{D}|\mathcal{C})\,\mathrm{P}(\mathcal{C}). \tag{96}$$

The data generation probability $\mathrm{P}(\mathcal{D}|\mathcal{C})$ is widely known as *likelihood* $\mathcal{L}(\mathcal{C};\mathcal{D})$. The only difference between $\mathrm{P}(\mathcal{D}|\mathcal{C})$ and $\mathcal{L}(M;\mathcal{D})$ is that the likelihood is interpreted as a function of $\mathcal{C}$, while the data generation probability is viewed as a probability of $\mathcal{D}$. In (96) we see, that $\mathrm{P}(\mathcal{C}|\mathcal{D})$ is proportional to the product of prior $\mathrm{P}(\mathcal{C})$ and likelihood $\mathcal{L}(\mathcal{C};\mathcal{D})$, i.e. the posterior $\mathrm{P}(\mathcal{C}|\mathcal{D})$ represents an "updated" version of the prior $\mathrm{P}(\mathcal{C})$, incorporating the likelihood $\mathcal{L}(\mathcal{C};\mathcal{D})$. There exist two general approaches to use the posterior distribution for learning:

(i) **Maximum a-posteriori (MAP) approach.** This approach aims to find the most probable model $M_{\mathrm{MAP}} \in \mathcal{C}$ for given data, i.e.

$$M_{\mathrm{MAP}} = \arg\max_{M\in\mathcal{C}} \mathrm{P}(M|\mathcal{D}) \tag{97}$$

$$= \arg\max_{M\in\mathcal{C}} \mathcal{L}(M;\mathcal{D})\,\mathrm{P}(M). \tag{98}$$

In the MAP approach we accept the single model M_{MAP} as best explanation for the data and consequently use it for further tasks.

(ii) **Bayesian approach.** In the Bayesian approach, we refrain to decide for a single model out of $\mathcal{C}$. Instead, we maintain the uncertainty about which model might be the "true" one and keep the whole model class $\mathcal{C}$, together with the posterior distribution $\mathrm{P}(\mathcal{C}|\mathcal{D})$.

To illustrate the difference of these two approaches, let us assume we are interested in the probability of an unseen sample $\mathbf{x}'$, after observing $\mathcal{D}$. The MAP approach yields

$$\mathrm{P}(\mathbf{X} = \mathbf{x}'|\mathcal{D}) = \mathrm{P}(\mathbf{x}'|M_{\mathrm{MAP}}), \tag{99}$$

while the Bayesian approach results in

$$\mathrm{P}(\mathbf{X} = \mathbf{x}'|\mathcal{D}) = \int_{\mathcal{C}} \mathrm{P}(\mathbf{x}'|M)\,\mathrm{P}(M|\mathcal{D})\,\mathrm{d}M. \tag{100}$$

The Bayesian approach proceeds more carefully and treats the uncertainty by marginalizing over all possible models. The MAP and the Bayesian approach often agree in the large sample limit, since for large L the mass of the posterior $\mathrm{P}(\mathcal{C}|\mathcal{D})$ is typically concentrated on a single model. For small sample sizes, the Bayesian approach typically surpasses the MAP approach. However, it is rarely possible to solve the integral in (100) exactly, which is the main drawback of the Bayesian approach. Typically, (100) has to be approximated, e.g. by integrating (or summing) over a small portion of all models.

In the special case when the prior distribution $\mathrm{P}(\mathcal{C})$ is flat, i.e. no model is preferred over the other, the posterior $\mathrm{P}(\mathcal{C}|\mathcal{D})$ is proportional to the likelihood $\mathcal{L}(M;\mathcal{D})$ and the MAP solution coincides with the *maximum-likelihood* (ML) solution, i.e.

$$M_{\mathrm{MAP}} = M_{\mathrm{ML}} = \arg\max_{M\in\mathcal{C}} \mathcal{L}(M;\mathcal{D}). \tag{101}$$

Since the data samples in $\mathcal{D}$ are i.i.d., the likelihood is given as

$$\mathcal{L}(\mathcal{C};\mathcal{D}) = \mathrm{P}(\mathcal{D}|\mathcal{C}) = \prod_{l=1}^{L} \mathrm{P}(\mathbf{x}^{(l)}|\mathcal{C}). \tag{102}$$

In practise, it is often convenient and numerically more stable to work with the *log-likelihood*

$$\log \mathcal{L}(\mathcal{C};\mathcal{D}) = \sum_{l=1}^{L} \log \mathrm{P}(\mathbf{x}^{(l)}|\mathcal{C}). \tag{103}$$

Since the logarithm is an increasing function, maximizing the log-likelihood is equivalent to maximizing the likelihood. Intuitively, (log-)likelihood is a measure of "goodness of fit", i.e. a model with maximum likelihood is considered as best explanation for the training data. Furthermore, there is an interesting connection between likelihood and information theory; For the sample $\mathcal{D}$, the empirical data distribution is defined as

$$\mathrm{P}_{\mathcal{D}}(\mathbf{x}) = \frac{1}{L}\sum_{l=1}^{L} \mathbb{1}_{\{\mathbf{x}=\mathbf{x}^{(l)}\}}, \tag{104}$$

where $\mathbb{1}_{\{\cdot\}}$ is the indicator function. It is easily shown that maximizing the likelihood is equivalent to minimize the Kullback-Leibler (KL) divergence between empirical distribution and model distribution, which is given as

$$\mathrm{KL}\left(\mathrm{P}_{\mathcal{D}}||\mathrm{P}(\cdot|\mathcal{C})\right) = \sum_{\mathbf{x}\in\mathbf{var}(\mathbf{X})} \mathrm{P}_{\mathcal{D}}(\mathbf{x}) \log \frac{\mathrm{P}_{\mathcal{D}}(\mathbf{x})}{\mathrm{P}(\mathbf{x}|\mathcal{C})}. \tag{105}$$

The KL divergence is a dissimilarity measure of the argument distributions. In particular, $\mathrm{KL}\left(\mathrm{P}||\mathrm{Q}\right) \geq 0$ and $\mathrm{KL}\left(\mathrm{P}||\mathrm{Q}\right) = 0$ if and only if $\mathrm{P}(\mathbf{X})$ and $\mathrm{Q}(\mathbf{X})$ are identical. Furthermore, $\mathrm{KL}\left(\mathrm{P}||\mathrm{Q}\right)$ measures the minimal average coding overhead, for encoding data using the distribution $\mathrm{Q}(\mathbf{X})$, while the data is actually distributed according to $\mathrm{P}(\mathbf{X})$ [Cover and Thomas, 1991].

4.2 Generative Learning of Bayesian Networks

We now specify the model class $\mathcal{C}$, and concentrate on the problem of learning BNs from data. As discussed in Section 3.1, a BN $\mathcal{B} = (\mathcal{G}, \boldsymbol{\Theta})$ defines a distribution over the model variables $\mathbf{X}$ according to

$$\mathrm{P}_{\mathcal{B}}(\mathbf{X}) = \prod_{i=1}^{N} \mathrm{P}(X_i|\mathbf{Pa}_{\mathcal{G}}(X_i), \boldsymbol{\theta}^i), \tag{106}$$

which is a product over local probability distributions, one for each variable X_i. Equation (106) highlights the two aspects of a BN: the structure $\mathcal{G}$, which determines the parents for each variable, and the parameters $\boldsymbol{\Theta} = \{\boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \ldots, \boldsymbol{\theta}^N\}$ which parameterize the local distributions. Although $\mathcal{G}$ and $\boldsymbol{\Theta}$ are coupled, e.g. the number of parameters in $\boldsymbol{\theta}^i$ depends on $\mathcal{G}$, an isolated consideration of $\mathcal{G}$

and $\boldsymbol{\Theta}$ is possible and reasonable. This leads to the two tasks of learning BNs: parameter learning and structure learning. For parameter learning, we assume that the network structure $\mathcal{G}$ is fixed, either because a domain expert specified the independence relationships among the model variables, or because a structure learning algorithm found an appropriate structure. For fixed structure $\mathcal{G}$, we introduce the shorthand $\mathbf{Pa}_i = \mathbf{Pa}_{\mathcal{G}}(X_i)$ and $\mathbf{x}_{\mathbf{Pa}_i} = \mathbf{x}(\mathbf{Pa}_{\mathcal{G}}(X_i))$. In the next section we discuss ML parameter learning, and in Section 4.2.2 we introduce a full Bayesian approach for BN parameter learning. In Section 4.2.3 we address the problem of learning also the structure $\mathcal{G}$ from data.

4.2.1 Maximum Likelihood Parameter Learning

For an i.i.d. sample $\mathcal{D}$, the log-likelihood of a BN model is given as

$$\log \mathcal{L}(\mathcal{B}; \mathcal{D}) = \sum_{l=1}^{L} \sum_{i=1}^{N} \log \mathrm{P}(x_i^{(l)} | \mathbf{x}_{\mathbf{Pa}_i}^{(l)}, \boldsymbol{\theta}^i) = \sum_{i=1}^{N} \ell(\boldsymbol{\theta}^i, \mathcal{D}). \tag{107}$$

Equation (107) shows that the log-likelihood decomposes, i.e. $\log \mathcal{L}(\mathcal{B}; \mathcal{D})$ equals to a sum over local terms, one for each variable, given as

$$\ell(\boldsymbol{\theta}^i, \mathcal{D}) = \sum_{l=1}^{L} \log \mathrm{P}(x_i^{(l)} | \mathbf{x}_{\mathbf{Pa}_i}^{(l)}, \boldsymbol{\theta}^i). \tag{108}$$

Thus the overall BN log-likelihood is maximized by maximizing the individual local terms w.r.t. the local parameters $\boldsymbol{\theta}^i$. This holds true as long as each local term $\ell(\boldsymbol{\theta}^i; \mathcal{D})$ only depends on $\boldsymbol{\theta}^i$. If there exists a coupling between the parameters, as for instance introduced by parameter tying, we loose this decomposition property.

Maximum Likelihood Parameter Learning for Discrete Variables. If the variables $\mathbf{X}$ are discrete, the most general CPDs are discrete distributions (cf. Example 12):

$$\mathrm{P}(X_i = j | \mathbf{Pa}_i = \mathbf{h}, \boldsymbol{\theta}^i) = \theta^i_{j|\mathbf{h}}. \tag{109}$$

In order to maximize the likelihood, we aim to individually maximize the local terms $\ell(\boldsymbol{\theta}^i; \mathcal{D})$ (108), which are given as

$$\ell(\boldsymbol{\theta}^i; \mathcal{D}) = \sum_{l=1}^{L} \log \left(\prod_{j=1}^{\mathbf{sp}(X_i)} \prod_{\mathbf{h} \in \mathbf{val}(\mathbf{Pa}_i)} {\theta^i_{j|\mathbf{h}}}^{u^{i,l}_{j|\mathbf{h}}} \right) \tag{110}$$

$$= \sum_{\mathbf{h}} \sum_{l: \mathbf{x}_l \in \mathcal{D}_{\mathbf{h}}} \sum_{j=1}^{\mathbf{sp}(X_i)} u^{i,l}_{j|\mathbf{h}} \log \theta^i_{j|\mathbf{h}} \tag{111}$$

$$= \sum_{\mathbf{h}} \log \mathcal{L}(\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}; \mathcal{D}_{\mathbf{h}}). \tag{112}$$

Here $u^{i,l}_{j|\mathbf{h}} = \mathbb{1}_{\{x_i^{(l)} = j \text{ and } \mathbf{x}^{(l)}_{\mathbf{Pa}_i} = \mathbf{h}\}}$. The data set $\mathcal{D}_{\mathbf{h}}$ is a subset of the training data $\mathcal{D}$, containing those samples $\mathbf{x}^{(l)}$ where $\mathbf{x}^{(l)}_{\mathbf{Pa}_i} = \mathbf{h}$. In (111) sums are interchanged and some terms which would be evaluated to zero, are discarded. In (112) the local terms $\ell(\boldsymbol{\theta}^i; \mathcal{D})$ decompose to a sum of local log-likelihoods

$$\log \mathcal{L}(\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}; \mathcal{D}_{\mathbf{h}}) = \sum_{l: \mathbf{x}_l \in \mathcal{D}_{\mathbf{h}}} \sum_{j=1}^{\mathbf{sp}(X_i)} u^{i,l}_{j|\mathbf{h}} \log \theta^i_{j|\mathbf{h}}. \tag{113}$$

Inserting this result into (107), we see that the BN log-likelihood is given as a sum of local log-likelihoods:

$$\log \mathcal{L}(\mathcal{B}; \mathcal{D}) = \sum_{i=1}^{N} \sum_{\mathbf{h} \in \mathbf{val}(\mathbf{Pa}_i)} \log \mathcal{L}(\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}; \mathcal{D}_{\mathbf{h}}) \tag{114}$$

Since each $\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}$ only occurs in a single summand in (114), the overall log-likelihood is maximized by individually maximizing each local log-likelihood. Note that $\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}$ are the parameters of X_i conditioned on a particular $\mathbf{h}$, i.e. they are the parameters of single dimensional discrete distributions. Finding these ML parameters is easy. For notational simplicity, we derive the ML estimate for a generic variable Y with data $\mathcal{D}_Y = \{y^{(1)}, \ldots, y^{(L)}\}$ and generic parameters $\boldsymbol{\theta}^Y = (\theta^Y_1, \ldots, \theta^Y_{\mathbf{sp}(Y)})$. The ML estimate $\hat{\boldsymbol{\theta}}^Y$ is found by solving the optimization problem:

$$\underset{\boldsymbol{\theta}^Y}{\text{maximize}} \quad \sum_{l=1}^{L} \sum_{j=1}^{\mathbf{sp}(Y)} u_j^l \, \log \theta_j^Y \tag{115}$$

$$\text{s.t.} \quad \sum_{j=1}^{\mathbf{sp}(Y)} \theta_j^Y = 1 \tag{116}$$

Here u_j^l is the indicator function $\mathbb{1}_{\{y^{(l)}=j\}}$. The Lagrangian function of this problem is given as

$$L(\boldsymbol{\theta}^Y, \nu) = \sum_{l=1}^{L} \sum_{j=1}^{\mathbf{sp}(Y)} u_j^l \, \log \theta_j^Y + \nu \left(1 - \sum_{j=1}^{\mathbf{sp}(Y)} \theta_j^Y\right). \tag{117}$$

Since the objective in (115) is concave and we have a linear equality constraint (116), stationarity of $L(\boldsymbol{\theta}^Y, \nu)$ is necessary and sufficient for optimality [Boyd and Vandenberghe, 2004], i.e. the constraint (116) has to hold and the partial derivatives w.r.t. θ_j^Y have to vanish:

$$\frac{\partial L(\boldsymbol{\theta}^Y, \nu)}{\partial \theta_j^Y} = \sum_{l=1}^{L} u_j^l \frac{1}{\theta_j^Y} - \nu \stackrel{!}{=} 0. \tag{118}$$

We see that the optimal parameters have the form

$$\hat{\theta}_j^Y = \frac{\sum_{l=1}^{L} u_j^l}{\nu} = \frac{n_j}{\nu}, \tag{119}$$

where $n_j = \sum_{l=1}^{L} u_j^l$ is the number of occurrences of $Y = j$ in the data set $\mathcal{D}_Y$. Since (116) has to hold, we see that ν has to be equal to $\sum_{j=1}^{\mathbf{sp}(Y)} n_j$, and therefore the ML parameters are simply the relative frequency counts according to $\mathcal{D}_Y$:

$$\hat{\theta}_j^Y = \frac{n_j}{\sum_{j'=1}^{\mathbf{sp}(Y)} n_{j'}}. \tag{120}$$

Applying this result to $\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}$, we obtain the ML parameters as

$$\hat{\theta}^i_{j|\mathbf{h}} = \frac{\sum_{l=1}^{L} u^{i,l}_{j|\mathbf{h}}}{\sum_{j'=1}^{\mathbf{sp}(X_j)} \sum_{l=1}^{L} u^{i,l}_{j'|\mathbf{h}}}. \tag{121}$$

This estimate is well-defined if there is at least one sample in the subset $\mathcal{D}_{\mathbf{h}}$. If there is no sample in $\mathcal{D}_{\mathbf{h}}$, an arbitrary distribution (e.g. the uniform distribution) can be assumed, since in this case $\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}$ does not contribute to the likelihood.

Maximum Likelihood Parameter Learning for Conditional Gaussian Variables. Now let us assume that the variables are real-valued, and that the CPDs are conditional Gaussians (cf. Example 13), i.e.

$$\mathrm{P}(X_i | \mathbf{Pa}_i, \boldsymbol{\theta}^i) = \mathcal{N}\left(X_i \,\middle|\, \boldsymbol{\alpha}_i^T \mathbf{x}_{\mathbf{Pa}_i} + \beta_i; \sigma^2\right), \tag{122}$$

where we assume a fixed shared variance σ^2. Again, in order to maximize the BN likelihood, we aim to individually maximize the local terms $\ell(\boldsymbol{\theta}^i; \mathcal{D})$ in (108). Inserting the logarithm of (122) in (108) yields

$$\ell(\boldsymbol{\theta}^i; \mathcal{D}) = \sum_{l=1}^{L} \left[-\frac{1}{2} \log(2\pi\sigma^2) - \frac{\left(x_i^{(l)} - \boldsymbol{\alpha}_i^T \mathbf{x}_{\mathbf{Pa}_i}^{(l)} - \beta_i \right)^2}{2\sigma^2} \right] \tag{123}$$

$$= -\frac{1}{2\sigma^2} \sum_{l=1}^{L} \left[x_i^{(l)^2} + \boldsymbol{\alpha}_i^T \mathbf{x}_{\mathbf{Pa}_i}^{(l)} \mathbf{x}_{\mathbf{Pa}_i}^{(l)T} \boldsymbol{\alpha}_i + \beta_i^2 - 2\, x_i^{(l)}\, \boldsymbol{\alpha}_i^T \mathbf{x}_{\mathbf{Pa}_i}^{(l)} - 2\, x_i^{(l)} \beta_i + 2\, \boldsymbol{\alpha}_i^T \mathbf{x}_{\mathbf{Pa}_i}^{(l)} \beta_i \right] + const., \tag{124}$$

where additive terms which are independent of $\boldsymbol{\alpha}_i$ and β_i are summarized in a constant. The overall log-likelihood of the BN is maximized by individually maximizing (123) w.r.t. $\boldsymbol{\theta}^i = \{\boldsymbol{\alpha}_i, \beta_i\}$. The function $\ell(\boldsymbol{\theta}^i; \mathcal{D})$ is the negative square of an affine function and therefore concave in $\boldsymbol{\alpha_i}$ and β_i [Boyd and Vandenberghe, 2004]. Thus, a maximum can be found be setting the partial derivatives to zero. For β_i we get:

$$\frac{\partial \ell(\boldsymbol{\theta}^i; \mathcal{D})}{\partial \beta_i} = -\frac{1}{2\sigma^2} \sum_{l=1}^{L} \left[2\,\beta_i - 2\, x_i^{(l)} + 2\, \mathbf{x}_{\mathbf{Pa}_i}^{(l)T} \boldsymbol{\alpha}_i \right] \stackrel{!}{=} 0. \tag{125}$$

Hence the ML parameter $\hat{\beta}_i$ is given as

$$\hat{\beta}_i = \frac{1}{L} \sum_{l=1}^{L} \left[x_i^{(l)} - \mathbf{x}_{\mathbf{Pa}_i}^{(l)T} \boldsymbol{\alpha}_i \right] = \hat{x}_i - \hat{\mathbf{x}}_{\mathbf{Pa}_i}^T \boldsymbol{\alpha}_i, \tag{126}$$

where $\hat{x}_i$ and $\hat{\mathbf{x}}_{\mathbf{Pa}_i}$ are the sample means of X_i and $\mathbf{Pa}_i$, respectively. For $\boldsymbol{\alpha}_i$ we have

$$\nabla_{\boldsymbol{\alpha_i}} \ell(\boldsymbol{\theta}^i; \mathcal{D}) = -\frac{1}{2\,\sigma^2} \sum_{l=1}^{L} \left[2\, \mathbf{x}_{\mathbf{Pa}_i}^{(l)} \mathbf{x}_{\mathbf{Pa}_i}^{(l)T} \boldsymbol{\alpha}_i - 2\, x_i^{(l)}\, \mathbf{x}_{\mathbf{Pa}_i}^{(l)} + 2\, \mathbf{x}_{\mathbf{Pa}_i}^{(l)} \beta_i \right] \stackrel{!}{=} \mathbf{0}. \tag{127}$$

Using $\hat{\beta}_i$ in (127), we get the ML parameters $\hat{\boldsymbol{\alpha}}_i$ by the following chain of equations:

$$\sum_{l=1}^{L} \left[\mathbf{x}_{\mathbf{Pa}_i}^{(l)} \left(\mathbf{x}_{\mathbf{Pa}_i}^{(l)T} - \hat{\mathbf{x}}_{\mathbf{Pa}_i}^T \right) \right] \hat{\boldsymbol{\alpha}}_i = \sum_{l=1}^{L} \left[\left(x_i^{(l)} - \hat{x}_i \right) \mathbf{x}_{\mathbf{Pa}_i}^{(l)} \right] \tag{128}$$

$$\sum_{l=1}^{L} \left[\left(\mathbf{x}_{\mathbf{Pa}_i}^{(l)} - \hat{\mathbf{x}}_{\mathbf{Pa}_i} + \hat{\mathbf{x}}_{\mathbf{Pa}_i} \right) \left(\mathbf{x}_{\mathbf{Pa}_i}^{(l)T} - \hat{\mathbf{x}}_{\mathbf{Pa}_i}^T \right) \right] \hat{\boldsymbol{\alpha}}_i = \sum_{l=1}^{L} \left[\left(x_i^{(l)} - \hat{x}_i \right) \left(\mathbf{x}_{\mathbf{Pa}_i}^{(l)} - \hat{\mathbf{x}}_{\mathbf{Pa}_i} + \hat{\mathbf{x}}_{\mathbf{Pa}_i} \right) \right] \tag{129}$$

$$\frac{1}{L-1} \sum_{l=1}^{L} \left[\left(\mathbf{x}_{\mathbf{Pa}_i}^{(l)} - \hat{\mathbf{x}}_{\mathbf{Pa}_i} \right) \left(\mathbf{x}_{\mathbf{Pa}_i}^{(l)T} - \hat{\mathbf{x}}_{\mathbf{Pa}_i}^T \right) \right] \hat{\boldsymbol{\alpha}}_i = \frac{1}{L-1} \sum_{l=1}^{L} \left[\left(x_i^{(l)} - \hat{x}_i \right) \left(\mathbf{x}_{\mathbf{Pa}_i}^{(l)} - \hat{\mathbf{x}}_{\mathbf{Pa}_i} \right) \right] \tag{130}$$

$$\hat{\mathbf{C}}_{\mathbf{Pa}}\, \hat{\boldsymbol{\alpha}}_i = \hat{\mathbf{c}} \tag{131}$$

$$\hat{\boldsymbol{\alpha}}_i = \hat{\mathbf{C}}_{\mathbf{Pa}}^{-1}\, \hat{\mathbf{c}}. \tag{132}$$

Here $\hat{\mathbf{C}}_{\mathbf{Pa}}$ is the estimated covariance matrix of the parent variables, and $\hat{\mathbf{c}}$ is the estimated covariance vector between parents and the child. In (130) we used the fact that $\sum_{l=1}^{L} \mathbf{v} \left(\mathbf{x}_{\mathbf{Pa}_i}^{(l)T} - \hat{\mathbf{x}}_{\mathbf{Pa}_i}^T \right) = \mathbf{0}$ and $\sum_{l=1}^{L} \left(x_i^{(l)} - \hat{x}_i \right) \mathbf{v} = \mathbf{0}$, for any vector $\mathbf{v}$ which does not depend on l.

4.2.2 Bayesian Parameter Learning

In a Bayesian approach, we do not assume that there exists a single set of "true" parameters, but maintain the uncertainty about the parameters. Therefore, we explicitly include the parameters into the model as in Figure 18, where a BN over variables $X_1, \ldots, X_4$ is augmented with their local CPD

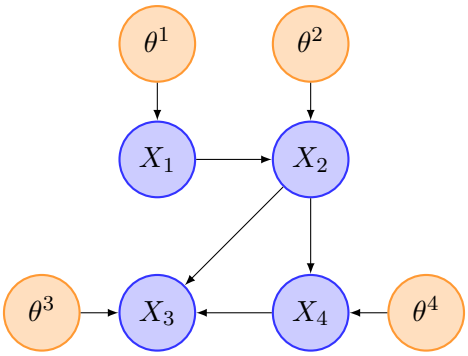


Figure 18: Bayesian network over variables $X_1, \ldots, X_4$, augmented with local CPD parameters $\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^4$.

parameters $\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^4$. Learning in the Bayesian setting means to perform inference over parameters; in the Bayesian view, there is actually no difference between learning and inference.

Note that there are no edges between any $\boldsymbol{\theta}^i$ and $\boldsymbol{\theta}^j$, which means that we assume parameter independence. This kind of independence is called *global parameter independence* (as opposed to *local parameter independence*, discussed below) [Heckerman et al., 1995, Koller and Friedman, 2009]. Global parameter independence, although reasonable in many cases, does not necessarily hold in every context. However, in the case of global parameter independence any prior over $\boldsymbol{\Theta} = \{\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^N\}$ factorizes according to

$$\mathrm{P}(\boldsymbol{\Theta}) = \prod_{i=1}^{N} \mathrm{P}(\boldsymbol{\theta}^i). \tag{133}$$

Furthermore, if all model variables $X_1, \ldots, X_N$ are observed, then the local parameters $\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^N$ are pairwise d-separated, and thus independent. Furthermore, the same is true for multiple i.i.d. samples, i.e. for complete data sets $\mathcal{D}$. In this case, if global (a-priori) parameter independence holds, also a-posteriori parameter independence holds, and the posterior over $\boldsymbol{\Theta}$ factorizes according to

$$\mathrm{P}(\boldsymbol{\Theta}|\mathcal{D}) = \prod_{i=1}^{N} \mathrm{P}(\boldsymbol{\theta}^i|\mathcal{D}). \tag{134}$$

Combining (134) with (106), we obtain the model posterior as

$$\begin{aligned}
\mathrm{P}_{\mathcal{B}}(\mathbf{X}, \boldsymbol{\Theta}|\mathcal{D}) &= \mathrm{P}_{\mathcal{B}}(\mathbf{X}|\boldsymbol{\Theta})\, \mathrm{P}(\boldsymbol{\Theta}|\mathcal{D}) && (135)\\
&= \left(\prod_{i=1}^{N} \mathrm{P}(X_i|\mathbf{Pa}_i, \boldsymbol{\theta}^i)\right) \left(\prod_{i=1}^{N} \mathrm{P}(\boldsymbol{\theta}^i|\mathcal{D})\right) && (136)\\
&= \prod_{i=1}^{N} \mathrm{P}(X_i|\mathbf{Pa}_i, \boldsymbol{\theta}^i)\, \mathrm{P}(\boldsymbol{\theta}^i|\mathcal{D}). && (137)
\end{aligned}$$

Since one is usually interested in the data posterior $\mathrm{P}(\mathbf{X}|\mathcal{D})$, one has to marginalize over $\boldsymbol{\Theta}$:

$$\begin{aligned}
\mathrm{P}(\mathbf{X}|\mathcal{D}) &= \int_{\boldsymbol{\Theta}} \mathrm{P}(\mathbf{X}, \boldsymbol{\Theta}|\mathcal{D})\, \mathrm{d}\boldsymbol{\Theta} && (138)\\
&= \int_{\boldsymbol{\theta}^1} \int_{\boldsymbol{\theta}^2} \cdots \int_{\boldsymbol{\theta}^N} \left(\prod_{i=1}^{N} \mathrm{P}(X_i|\mathbf{Pa}_i, \boldsymbol{\theta}^i)\, \mathrm{P}(\boldsymbol{\theta}^i|\mathcal{D})\right) \mathrm{d}\boldsymbol{\theta}^1\, \mathrm{d}\boldsymbol{\theta}^2 \ldots \mathrm{d}\boldsymbol{\theta}^N && (139)\\
&= \prod_{i=1}^{N} \left(\int_{\boldsymbol{\theta}^i} \mathrm{P}(X_i|\mathbf{Pa}_i, \boldsymbol{\theta}^i)\, \mathrm{P}(\boldsymbol{\theta}^i|\mathcal{D})\, \mathrm{d}\boldsymbol{\theta}^i\right) && (140)
\end{aligned}$$

The exchange of integrals and products in (140) is possible since each term in the product in (139) depends only on a single $\boldsymbol{\theta}^i$. Hence, under the assumption of global parameter independence and completely observed data, the integral over the entire parameter space in (138) factorizes into a product of several simpler integrals (140).

Bayesian Parameter Learning for Discrete Variables. We now return to the example, where all variables of the BN are discrete. As a basic example, and in order to introduce concepts which are required later, let us consider a simple BN with only a single variable, i.e. $N = 1$ and set $X = X_1$ and $J = \mathbf{sp}(X)$. The probability of X being in state j is

$$\mathrm{P}(X = j|\boldsymbol{\theta}) = \theta_j. \tag{141}$$

Figure 19 shows the corresponding BN, where $\boldsymbol{\theta}$ is explicitly included. The parameters can be rep-

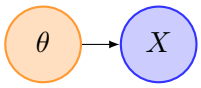


Figure 19: Simple Bayesian network containing a single variable X, augmented with parameters $\boldsymbol{\theta}$.

resented as a vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_J)^T$ of RVs, which has to be an element of the standard simplex $\mathcal{S} = \{\boldsymbol{\theta}|\theta_j \geq 0,\ \forall j, \sum_{j=1}^J \theta_j = 1\}$. We need to specify a prior distribution over $\boldsymbol{\theta}$, where a natural choice is the Dirichlet distribution, defined as

$$\mathrm{Dir}(\boldsymbol{\theta}; \boldsymbol{\alpha}) = \begin{cases} \frac{1}{B(\boldsymbol{\alpha})} \prod_{j=1}^J \theta_j^{\alpha_j - 1} & \text{if } \boldsymbol{\theta} \in \mathcal{S} \\ 0 & \text{otherwise.} \end{cases} \tag{142}$$

The Dirichlet distribution is parameterized by the vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_J)^T$ with $\alpha_j > 0$ for all j. The term $\frac{1}{B(\boldsymbol{\alpha})}$ is the normalization constant of the distribution, where the *beta-function* $B(\boldsymbol{\alpha})$ is given as

$$B(\boldsymbol{\alpha}) = \int_{\mathcal{S}} \prod_{j=1}^J \theta_j^{\alpha_j - 1} \,\mathrm{d}\boldsymbol{\theta} = \frac{\prod_{j=1}^J \Gamma(\alpha_j)}{\Gamma\left(\sum_{j=1}^J \alpha_j\right)}. \tag{143}$$

Here $\Gamma(\cdot)$ is the *gamma function*, i.e. a continuous generalization of the factorial function. In the special case that $\alpha_j = 1$, $\forall j$, the Dirichlet distribution is uniform over all $\boldsymbol{\theta} \in \mathcal{S}$.

Assuming a Dirichlet prior over $\boldsymbol{\theta}$, the distribution of the BN in Figure 19 is given as $\mathrm{P}(X, \boldsymbol{\theta}) = \mathrm{P}_{\mathcal{B}}(X|\boldsymbol{\theta}) \,\mathrm{Dir}(\boldsymbol{\theta}; \boldsymbol{\alpha})$. Usually one is interested in the posterior over X, which is obtained by marginalizing over $\boldsymbol{\theta}$:

$$\begin{aligned}
\mathrm{P}(X = j') &= \int_{\boldsymbol{\theta}} \mathrm{P}(X|\boldsymbol{\theta}) \,\mathrm{Dir}(\boldsymbol{\theta}; \boldsymbol{\alpha}) \,\mathrm{d}\boldsymbol{\theta} && (144)\\
&= \int_{\mathcal{S}} \theta_{j'} \,\frac{1}{B(\boldsymbol{\alpha})} \prod_{j=1}^J \theta_j^{\alpha_j - 1} \,\mathrm{d}\boldsymbol{\theta} && (145)\\
&= \frac{1}{B(\boldsymbol{\alpha})} \frac{\Gamma(\alpha_{j'} + 1) \prod_{j \neq j'} \Gamma(\alpha_j)}{\Gamma\left(1 + \sum_{j=1}^J \alpha_j\right)} && (146)\\
&= \frac{B(\boldsymbol{\alpha})}{B(\boldsymbol{\alpha})} \frac{\alpha_{j'}}{\sum_{j=1}^J \alpha_j} && (147)\\
&= \frac{\alpha_{j'}}{\sum_{j=1}^J \alpha_j}. && (148)
\end{aligned}$$

The integral in (145) can be solved using (143), and (147) follows from the properties of the gamma function. We see that (147) has the form of relative frequency counts, exactly as the ML estimator

in (121). Thus, the Dirichlet parameters $\boldsymbol{\alpha}$ gain the interpretation of a-priori pseudo-data-counts, which represent our belief about $\boldsymbol{\theta}$. Now assume that we have an i.i.d. sample $\mathcal{D} = (x^{(1)}, \ldots, x^{(L)})$ of variable X. Learning in the Bayesian framework means to incorporate the information given by $\mathcal{D}$, i.e. to modify our belief about $\boldsymbol{\theta}$. Formally, we want to obtain the posterior distribution over $\boldsymbol{\theta}$:

$$\mathrm{P}(\boldsymbol{\theta}|\mathcal{D}) \propto \mathrm{P}(\mathcal{D}|\boldsymbol{\theta})\,\mathrm{P}(\boldsymbol{\theta}) = \left(\prod_{l=1}^{L} \mathrm{P}(x^{(l)}|\boldsymbol{\theta})\right) \mathrm{Dir}(\boldsymbol{\theta};\boldsymbol{\alpha}) \tag{149}$$

$$= \left(\prod_{l=1}^{L}\prod_{j=1}^{J} \theta_j^{u_j^l}\right) \left(\frac{1}{B(\boldsymbol{\alpha})}\prod_{j=1}^{J} \theta_j^{\alpha_j-1}\right) \tag{150}$$

$$= \frac{1}{B(\boldsymbol{\alpha})}\prod_{j=1}^{J} \theta_j^{n_j+\alpha_j-1}, \tag{151}$$

where $u_j^l = \mathbb{1}_{\{x^{(l)}=j\}}$ and $n_j = \sum_{l=1}^{L} u_j^l$. Note that (151) has again the functional form of a Dirichlet distribution, although not correctly normalized. Hence, we can conclude that the posterior $\mathrm{P}(\boldsymbol{\theta}|\mathcal{D})$ is also a Dirichlet distribution. After normalizing, we obtain

$$\mathrm{P}(\boldsymbol{\theta}|\mathcal{D}) = \mathrm{Dir}(\boldsymbol{\theta};\boldsymbol{\alpha}+\mathbf{n}), \tag{152}$$

where $\mathbf{n} = (n_1, n_2, \ldots, n_J)^T$. The property that the posterior is from the same family of distributions as the prior, comes from the fact that the Dirichlet distribution is a so-called *conjugate prior* [Bishop, 2006] for the discrete distribution. The intuitive interpretation of (152) is that the distribution over $\boldsymbol{\theta}$ is updated by adding the real data counts $\mathbf{n}$ to our a-priori pseudo-counts $\boldsymbol{\alpha}$. Replacing the prior $\mathrm{P}(\boldsymbol{\theta})$ with the posterior $\mathrm{P}(\boldsymbol{\theta}|\mathcal{D})$ in (144)-(147) yields

$$\mathrm{P}(X = j'|\mathcal{D}) = \frac{\alpha_{j'} + n'_j}{\sum_{j=1}^{J}(\alpha_j + n_j)}. \tag{153}$$

Now we extend the discussion to general BNs. First, we interpret the CPD parameters $\boldsymbol{\Theta} = \{\boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \ldots, \boldsymbol{\theta}^N\}$ as RVs, i.e. we define a prior $\mathrm{P}(\boldsymbol{\Theta})$. We assume global parameter independence, such that the prior takes the form $\mathrm{P}(\boldsymbol{\Theta}) = \prod_{i=1}^{N} \mathrm{P}(\boldsymbol{\theta}^i)$. Note that $\boldsymbol{\theta}^i$ can be interpreted as a matrix of RVs, containing one column $\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}$ for each possible assignment $\mathbf{h}$ of the parent variables $\mathbf{Pa}_i$. We can additionally assume that the columns of $\boldsymbol{\theta}^i$ are a-priori independent, i.e. that the prior over $\boldsymbol{\theta}^i$ factorizes as

$$\mathrm{P}(\boldsymbol{\theta}^i) = \prod_{\mathbf{h}\in\mathbf{val}(\mathbf{Pa}_i)} \mathrm{P}(\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}). \tag{154}$$

This independence assumption is called *local parameter independence*, as in contrast to global parameter independence [Heckerman et al., 1995, Koller and Friedman, 2009]. Similar, as global parameter independence, local parameter independence is often a reasonable assumption, but has to be considered with care. For a specific assignment $\mathbf{h}$, the vector $\boldsymbol{\theta}^i_{\cdot|\mathbf{h}} = (\theta^i_{1|\mathbf{h}}, \ldots, \theta^i_{\mathbf{sp}(X_i)|\mathbf{h}})^T$ represents a single (conditional) discrete distribution over X_i. As in the basic example with only one variable, we can assume a Dirichlet prior for $\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}$, each with its own parameters $\boldsymbol{\alpha}^i_{\mathbf{h}} = (\alpha^i_{1|\mathbf{h}}, \ldots, \alpha^i_{\mathbf{sp}(X_i)|\mathbf{h}})$. Together with the assumption of global and local parameter independence, the prior over the whole parameter set $\boldsymbol{\Theta}$ is then given as

$$\mathrm{P}(\boldsymbol{\Theta}) = \prod_{i=1}^{N}\prod_{\mathbf{h}} \mathrm{Dir}\left(\boldsymbol{\theta}^i_{\cdot|\mathbf{h}};\boldsymbol{\alpha}^i_{\mathbf{h}}\right). \tag{155}$$

For global parameter independence, we observed that the parameters $\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^N$ remain independent, when conditioned on a completely observed data set (cf. (133) and (134)) since the parameters are d-separated by the observed nodes $X_1, \ldots, X_N$. For local parameter independence, it does not happen that any two sets of parameters $\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}$ and $\boldsymbol{\theta}^i_{\cdot|\mathbf{h}'}$, $\mathbf{h} \neq \mathbf{h}'$, are d-separated.

Example 21. *Consider Figure 20 and let* $q = \mathbf{sp}\,(X_1)\ \mathbf{sp}\,(X_2)$ *be the total number of parent configurations for node* X_3*. The parameter sets* $\boldsymbol{\theta}^3_{\cdot|\mathbf{h}_1}, \ldots, \boldsymbol{\theta}^3_{\cdot|\mathbf{h}_q}$ *are not d-separated, since* X_3 *is observed.*

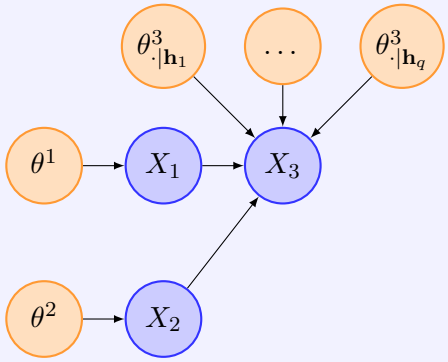


Figure 20: Bayesian network over variables X_1, X_2, X_3, X_4.

However, note that d-separation is sufficient for conditional independence, but not necessary. This means that even if two sets of nodes are not d-separated, they still can turn out to be independent. Fortunately, this is the case here. To see this, assume that local (a-priori) parameter independence holds, and that all nodes $X_1, \ldots, X_N$ are observed. For some arbitrary node X_i, the posterior parameter distribution is given as

$$\mathrm{P}(\boldsymbol{\theta}^i|\mathbf{X}) = \mathrm{P}(\boldsymbol{\theta}^i|X_i, \mathbf{Pa}_i) = \frac{\mathrm{P}(X_i|\boldsymbol{\theta}^i, \mathbf{Pa}_i)\,\mathrm{P}(\boldsymbol{\theta}^i|\mathbf{Pa}_i)}{\mathrm{P}(X_i|\mathbf{Pa}_i)} \tag{156}$$

$$= \frac{\mathrm{P}(X_i|\boldsymbol{\theta}^i_{\cdot|\mathbf{Pa}_i}, \mathbf{Pa}_i)\ \prod_{\mathbf{h}} \mathrm{P}(\boldsymbol{\theta}^i_{\cdot|\mathbf{h}})}{\mathrm{P}(X_i|\mathbf{Pa}_i)} \tag{157}$$

$$= \frac{\prod_{\mathbf{h}} f(\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}, \mathbf{X})}{\mathrm{P}(X_i|\mathbf{Pa}_i)}, \tag{158}$$

where

$$f(\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}, \mathbf{X}) = \begin{cases} \mathrm{P}(X_i|\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}, \mathbf{Pa}_i)\,\mathrm{P}(\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}) & \text{if } \mathbf{h} = \mathbf{Pa}_i, \text{and} \\ \mathrm{P}(\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}) & \text{otherwise.} \end{cases} \tag{159}$$

We see that the posterior parameter distribution factorizes over the parent configurations, i.e. that the parameters $\boldsymbol{\theta}^i_{\cdot|\mathbf{h}_1}, \ldots, \boldsymbol{\theta}^i_{\cdot|\mathbf{h}_q}$, $q = \mathbf{sp}\,(\mathbf{Pa}_i)$, remain independent when conditioned on $\mathbf{X}$. This conclusion comes from two observations in (157): (i) All $\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}$ except $\boldsymbol{\theta}^i_{\cdot|\mathbf{Pa}_i}$ are independent from X_i, when $\mathbf{Pa}_i$ is given, and (ii) $\boldsymbol{\theta}^i$ is a priori independent from $\mathbf{Pa}_i$.

Furthermore, local a-posteriori parameter independence also holds for multiple observed i.i.d. samples $\mathcal{D}$, i.e.

$$\mathrm{P}(\boldsymbol{\theta}^i|\mathcal{D}) = \prod_{\mathbf{h}} \mathrm{P}(\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}|\mathcal{D}). \tag{160}$$

Since we assumed Dirichlet priors, which are conjugate priors for discrete distributions, the posterior over $\boldsymbol{\theta}^i$ takes the form (cf. (152)),

$$\mathrm{P}(\boldsymbol{\theta}^i|\mathcal{D}) = \prod_{\mathbf{h}} \mathrm{Dir}(\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}|\boldsymbol{\alpha}^i_{\mathbf{h}} + \mathbf{n}^i_{\mathbf{h}}), \tag{161}$$

where $\mathbf{n}^i_{\mathbf{h}} = (n^i_{1|\mathbf{h}}, n^i_{2|\mathbf{h}}, \ldots, n^i_{\mathbf{sp}(X_i)|\mathbf{h}})^T$, and $n^i_{j|\mathbf{h}} = \sum_{l=1}^{L} u^{i,l}_{j|\mathbf{h}}$. This result can now be applied to (140), where we showed (under the assumption of global parameter independence), that

$$\mathrm{P}(\mathbf{X}|\mathcal{D}) = \prod_{i=1}^{N} \left(\int_{\boldsymbol{\theta}^i} \mathrm{P}(X_i|\mathbf{Pa}_i, \boldsymbol{\theta}^i)\,\mathrm{P}(\boldsymbol{\theta}^i|\mathcal{D})\,\mathrm{d}\boldsymbol{\theta}^i \right). \tag{162}$$

Inserting (161) in (162), we obtain

$$\mathrm{P}(\mathbf{X}=\mathbf{x}|\mathcal{D}) = \prod_{i=1}^{N}\left(\int_{\boldsymbol{\theta}^i}\left(\prod_{j=1}^{\mathbf{sp}(X_i)}\prod_{\mathbf{h}}{\theta^i_{j|\mathbf{h}}}^{u^i_{j|\mathbf{h}}}\right)\left(\prod_{\mathbf{h}}\mathrm{Dir}(\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}|\boldsymbol{\alpha}^i_{\mathbf{h}}+\mathbf{n}^i_{\mathbf{h}})\right)\mathrm{d}\boldsymbol{\theta}^i\right) \tag{163}$$

$$= \prod_{i=1}^{N}\left(\int_{\boldsymbol{\theta}^i}\prod_{\mathbf{h}}\left[\left(\prod_{j=1}^{\mathbf{sp}(X_i)}{\theta^i_{j|\mathbf{h}}}^{u^i_{j|\mathbf{h}}}\right)\mathrm{Dir}(\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}|\boldsymbol{\alpha}^i_{\mathbf{h}}+\mathbf{n}^i_{\mathbf{h}})\right]\mathrm{d}\boldsymbol{\theta}^i\right) \tag{164}$$

$$= \prod_{i=1}^{N}\left(\int_{\boldsymbol{\theta}^i_{\cdot|\mathbf{h}_1}}\cdots\int_{\boldsymbol{\theta}^i_{\cdot|\mathbf{h}_{q_i}}}\prod_{\mathbf{h}}\left[\left(\prod_{j=1}^{\mathbf{sp}(X_i)}{\theta^i_{j|\mathbf{h}}}^{u^i_{j|\mathbf{h}}}\right)\mathrm{Dir}(\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}|\boldsymbol{\alpha}^i_{\mathbf{h}}+\mathbf{n}^i_{\mathbf{h}})\right]\mathrm{d}\boldsymbol{\theta}^i_{\cdot|\mathbf{h}_1}\cdots\mathrm{d}\boldsymbol{\theta}^i_{\cdot|\mathbf{h}_{q_i}}\right) \tag{165}$$

$$= \prod_{i=1}^{N}\prod_{\mathbf{h}}\left(\int_{\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}}\left(\prod_{j=1}^{\mathbf{sp}(X_i)}{\theta^i_{j|\mathbf{h}}}^{u^i_{j|\mathbf{h}}}\right)\mathrm{Dir}(\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}|\boldsymbol{\alpha}^i_{\mathbf{h}}+\mathbf{n}^i_{\mathbf{h}})\,\mathrm{d}\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}\right), \tag{166}$$

where $q_i = \mathbf{sp}\left(\mathbf{Pa}_i\right)$. The integration over the entire parameter space factorizes into a product of many simpler integrals. Each integral in (166) has the same form as in (144) and is given as

$$\int_{\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}}\left(\prod_{j=1}^{\mathbf{sp}(X_i)}{\theta^i_{j|\mathbf{h}}}^{u^i_{j|\mathbf{h}}}\right)\mathrm{Dir}(\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}|\boldsymbol{\alpha}^i_{\mathbf{h}}+\mathbf{n}^i_{\mathbf{h}})\,\mathrm{d}\boldsymbol{\theta}^i_{\cdot|\mathbf{h}} = \begin{cases}\frac{\alpha^i_{x_i|\mathbf{h}}+n^i_{x_i|\mathbf{h}}}{\sum_{j=1}^{\mathbf{sp}(X_i)}(\alpha^i_{j|\mathbf{h}}+n^i_{j|\mathbf{h}})} & \text{if } \mathbf{h}=\mathbf{x}_{\mathbf{Pa}_i}\\ 1 & \text{otherwise.}\end{cases} \tag{167}$$

Therefore, under the assumptions of global and local parameter independence and using Dirichlet parameter priors, the Bayesian approach is feasible for BNs with discrete variables and reduces to a product over closed-form integrations. This result can be seen as the Bayesian analog to (114), where we showed that the BN likelihood is maximized by solving many independent closed-form ML problems.

4.2.3 Learning the Structure of Bayesian Networks

So far we have discussed parameter learning techniques for BNs, i.e. we assumed that the BN structure $\mathcal{G}$ is fixed and learned parameters the $\boldsymbol{\Theta}$. The goal is now to also learn the structure $\mathcal{G}$ from data, where we restrict the discussion to discrete variables from now on. Generally there are two main approaches to structure learning:

(i) **Constraint-based approach.** In this approach, we aim to find a structure $\mathcal{G}$ which represents the same conditional independencies as present in the data. While being theoretically sound, these methods heavily rely on statistical independence tests performed on the data, which might be inaccurate and usually render the approach unsuitable in practise. Therefore, we do not discuss constraint-based approaches here, but refer the interested reader to [Verma and Pearl, 1992, Spirtes et al., 2001, Koller and Friedman, 2009], and references therein.

(ii) **Scoring-based approach.** Here we aim to find a graph which represents a "suitable" distribution to represent the true distribution $\mathrm{P}^*(\mathbf{X})$, where "suitability" is measured by a scoring function $\mathrm{score}\left(\mathcal{G}\right)$. There are two key issues here: Defining an appropriate scoring function, and developing a search algorithm which finds a score-maximizing structure $\mathcal{G}$.

A natural choice for the scoring function would be the log-likelihood, i.e. the aim to maximize $\log\mathcal{L}(\mathcal{B};\mathcal{D}) = \log\mathcal{L}(\mathcal{G},\boldsymbol{\Theta};\mathcal{D})$, w.r.t. $\mathcal{G}$ and $\boldsymbol{\Theta}$. Using the ML parameter estimates $\hat{\boldsymbol{\Theta}}$ from Section 4.2.1, this problem reduces to maximize

$$\log\mathcal{L}(\mathcal{G},\hat{\boldsymbol{\Theta}};\mathcal{D}) = \sum_{l=1}^{L}\log\mathrm{P}(\mathbf{X}=\mathbf{x}^{(l)}|\mathcal{G},\hat{\boldsymbol{\Theta}}). \tag{168}$$

w.r.t. $\mathcal{G}$. However, the likelihood score in (168) is in general not suitable for structure learning. When $\mathcal{G}$ is a subgraph of $\mathcal{G}'$, it is easy to show that a BN defined over $\mathcal{G}'$ contains all distributions which can

also be represented by a BN defined over $\mathcal{G}$. Consequently, the supergraph $\mathcal{G}'$ will always have at least the same likelihood-score as $\mathcal{G}$, and furthermore, any fully connected graph will also be a maximizer of (168). We see that likelihood prefers more complex models and leads to overfitting. Note that overfitting stems from using a too flexible model class for a finite data set, and that the flexibility of a model is reflected by the number of free parameters. The number of parameters $\mathrm{T}(\mathcal{G})$ in a BN is given as

$$\mathrm{T}(\mathcal{G}) = \sum_{i=1}^{N} (\mathbf{sp}(X_i) - 1)(\mathbf{sp}(\mathbf{Pa}_{\mathcal{G}}(X_i))), \tag{169}$$

which grows exponentially with the maximal number of parents. The likelihood-score merely aims to fit the training data, but is blind to model complexity.

As a remedy, we can modify the pure likelihood-score in order to account for model complexity. One way delivers the minimum description length (MDL) principle [Rissanen, 1978], which aims to find the shortest, most compact representation of the data $\mathcal{D}$. As discussed in Section 4.1, maximizing likelihood is equivalent to minimizing $\mathrm{KL}(\mathrm{P}_{\mathcal{D}}||\mathrm{P}(\cdot|\mathcal{C}))$, i.e. the KL divergence between the empirical data distribution and the model distribution. The KL divergence measures the average coding overhead for encoding the data $\mathcal{D}$ using $\mathrm{P}(\mathbf{X}|\mathcal{C})$ instead of $\mathrm{P}_{\mathcal{D}}(\mathbf{X})$. The MDL principle additionally accounts for the description length of the model, which is measured by the number of parameters. For BNs with discrete variables, the following MDL score was derived [Lam and Bacchus, 1994, Suzuki, 1993]:

$$\mathrm{MDL}(\mathcal{G}) = -\sum_{l=1}^{L} \log \mathrm{P}(\mathbf{x}^{(l)}|\mathcal{G}, \boldsymbol{\Theta}_{\mathrm{ML}}) + \frac{\log L}{2} \mathrm{T}(\mathcal{G}), \tag{170}$$

which is a negative score, since we aim to find a graph $\mathcal{G}$ minimizing $\mathrm{MDL}(\mathcal{G})$. The MDL score is simply the negative likelihood score (168), plus a penalization term for the number of parameters. In typical applications, the likelihood decreases linearly with L, while the penalization term in (170) grows logarithmically. Therefore, for large L, the likelihood dominates the penalization term and the MDL score becomes equivalent to likelihood. However, when L is small, then the penalization term significantly influences (170), resulting in a preference for BNs with fewer parameters.

An alternative way to avoid overfitting is delivered by a Bayesian approach. The main problem with the likelihood score (168) is that it is not aware of model complexity, i.e. the number of free parameters. While with more densely connected graphs $\mathcal{G}$ the dimensionality of the parameter space $\boldsymbol{\Theta}$ increases exponentially, nevertheless only a single point estimate is used from this space, i.e. the ML parameters $\boldsymbol{\Theta}_{\mathrm{ML}}$. Therefore, the ML approach can be described as overly optimistic, trusting in the single "true" parameters $\boldsymbol{\Theta}_{\mathrm{ML}}$. In contrast to the ML approach, the Bayesian approach (cf. Section 4.2.2) uses the parameter space in its entirety. We introduce a prior distribution $\mathrm{P}(\boldsymbol{\Theta})$ over the parameters, and marginalize over $\boldsymbol{\Theta}$, leading to the Bayesian score

$$\mathrm{Bayes}(\mathcal{G}) = \mathrm{P}(\mathcal{D}|\mathcal{G}) = \int_{\boldsymbol{\Theta}} \mathrm{P}(\mathcal{D}|\boldsymbol{\Theta}, \mathcal{G}) \mathrm{P}(\boldsymbol{\Theta}) \, \mathrm{d}\boldsymbol{\Theta} \tag{171}$$

$$= \int_{\boldsymbol{\Theta}} \prod_{l=1}^{L} \mathrm{P}(\mathbf{x}^{(l)}|\boldsymbol{\Theta}, \mathcal{G}) \mathrm{P}(\boldsymbol{\Theta}) \, \mathrm{d}\boldsymbol{\Theta}. \tag{172}$$

We see that this score actually represents a semi-Bayesian approach: While being Bayesian about $\boldsymbol{\Theta}$, we still follow the ML principle for the structure $\mathcal{G}$. A Bayesian approach, which requires approximation techniques, is provided in [Friedmann and Koller, 2000]. Assuming global and local parameter independence, and using Dirichlet priors for the local parameters (cf. Section 4.2.2) leads to the

Bayesian-Dirichlet score (BD) [Buntine, 1991, Cooper and Herskovits, 1992]

$$\mathrm{BD}(\mathcal{G}) = \int_{\boldsymbol{\Theta}} \left(\prod_{l=1}^{L} \prod_{i=1}^{N} \prod_{j=1}^{\mathbf{sp}(X_i)} \prod_{\mathbf{h}} {\theta^i_{j|\mathbf{h}}}^{u^{i,l}_{j|\mathbf{h}}} \right) \left(\prod_{i=1}^{N} \prod_{\mathbf{h}} \frac{1}{B(\boldsymbol{\alpha}^i_{\mathbf{h}})} \prod_{j=1}^{\mathbf{sp}(X_i)} {\theta^i_{j|\mathbf{h}}}^{\alpha^i_{j|\mathbf{h}}-1} \right) \mathrm{d}\boldsymbol{\Theta} \tag{173}$$

$$= \int_{\boldsymbol{\Theta}} \prod_{i=1}^{N} \prod_{\mathbf{h}} \frac{1}{B(\boldsymbol{\alpha}^i_{\mathbf{h}})} \prod_{j=1}^{\mathbf{sp}(X_i)} {\theta^i_{j|\mathbf{h}}}^{\alpha^i_{j|\mathbf{h}}+n^i_{j|\mathbf{h}}-1} \mathrm{d}\boldsymbol{\Theta} \tag{174}$$

$$= \prod_{i=1}^{N} \prod_{\mathbf{h}} \frac{1}{B(\boldsymbol{\alpha}^i_{\mathbf{h}})} \left(\int_{\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}} \prod_{j=1}^{\mathbf{sp}(X_i)} {\theta^i_{j|\mathbf{h}}}^{\alpha^i_{j|\mathbf{h}}+n^i_{j|\mathbf{h}}-1} \mathrm{d}\boldsymbol{\theta}^i_{\cdot|\mathbf{h}} \right) \tag{175}$$

$$= \prod_{i=1}^{N} \prod_{\mathbf{h}} \frac{B(\boldsymbol{\alpha}^i_{\mathbf{h}} + \mathbf{n}^i_{\mathbf{h}})}{B(\boldsymbol{\alpha}^i_{\mathbf{h}})} \tag{176}$$

$$= \prod_{i=1}^{N} \prod_{\mathbf{h}} \frac{\Gamma\left(\sum_{j=1}^{\mathbf{sp}(Xi)} \alpha^i_{j|\mathbf{h}}\right)}{\Gamma\left(\sum_{j=1}^{\mathbf{sp}(X_i)} \alpha^i_{j|\mathbf{h}} + n^i_{j|\mathbf{h}}\right)} \prod_{j=1}^{\mathbf{sp}(Xi)} \frac{\Gamma\left(\alpha^i_{j|\mathbf{h}} + n^i_{j|\mathbf{h}}\right)}{\Gamma\left(\alpha^i_{j|\mathbf{h}}\right)}. \tag{177}$$

In (175) we can interchange integrals with products, since each term only depends on a single $\boldsymbol{\theta}^i_{\cdot|\mathbf{h}}$. The solution of the integrals is given in (143). Heckerman et al. [1995] pointed out that the parameters $\boldsymbol{\alpha}^{\boldsymbol{i}}_{\mathbf{h}}$ can not be chosen arbitrarily, when one desires a consistent, likelihood-equivalent score, meaning that the same score is assigned for two networks representing the same set of independence assumptions. They show how such parameters can be constructed, leading to the likelihood-equivalent Bayesian-Dirichlet score (BDe). A simple choice for $\boldsymbol{\alpha}^{\boldsymbol{i}}_{\mathbf{h}}$, already proposed in [Buntine, 1991], is

$$\alpha^i_{j|\mathbf{h}} = \frac{A}{\mathbf{sp}\,(X_i)\ \mathbf{sp}\,(\mathbf{Pa}_{\mathcal{G}}(X_i))}, \tag{178}$$

where A is the equivalent sample size, i.e. the number of virtual a-priori samples (a typical value would be $A = 1$). These parameters result from the a-priori assumption of an uniform distribution over the state space of $\mathbf{X}$. Consequently, the BDe score using parameters according (178) is called BDe uniform score (BDeu).

Furthermore, there is a connection between the BD score and the MDL score: It can be shown [Koller and Friedman, 2009], that for $L \to \infty$, the logarithm of the BD score converges to the so-called Bayesian information criterion score (BIC)

$$\mathrm{BIC}(\mathcal{G}) = \sum_{l=1}^{L} \log \mathrm{P}(\mathbf{x}^{(l)}|\mathcal{G}, \boldsymbol{\Theta}_{\mathrm{ML}}) - \frac{\log N}{2} \mathrm{T}\,(\mathcal{G}) + const., \tag{179}$$

which is the negative MDL score (170), up to some constant. Thus, although derived from two different perspectives, the MDL score and the BIC score are equivalent.

We now turn to the problem of finding a graph $\mathcal{G}$ which maximizes a particular score. All the scores discussed so far, namely log-likelihood (168), MDL (170) and BIC (179), and (the logarithm of) the BD/BDe score (177) can be written in the form

$$\mathrm{score}\,(\mathcal{G}) = \sum_{i=1}^{N} \mathrm{score}_i(X_i, \mathbf{Pa}_{\mathcal{G}}(X_i)), \tag{180}$$

which means that the global network score decomposes into a sum of local scores, depending only on the individual variables and their parents. This property is very important to find efficient structure learning algorithms. For example, if an algorithm decides to add a parent to some variable X_i, then the overall score is affected only via the the change of a single additive term. Therefore, decomposability of the scoring function alleviates the problem of finding a score-maximizing graph $\mathcal{G}$. Unfortunately, even for decomposable scores the problem of finding a score-maximizing BN structure $\mathcal{G}$ is NP-hard in general, even if the maximum number of parents is restricted to $K \geq 2$ [Chickering et al., 1994,

Chickering, 1996]. For $K = 1$, i.e. when the BN is constrained to be a directed tree, the Chow-Liu algorithm [Chow and Liu, 1968] learns the optimal network structure in polynomial time. Here the structure learning problem is reduced to a maximum spanning tree problem, where the connection weights are given by mutual information estimates. It is shown [Chow and Liu, 1968] that this is equivalent to maximizing the log-likelihood over the class of BNs with a directed tree structure. The class of directed trees is typically restrictive enough to avoid overfitting, i.e. the likelihood score is an appropriate choice in this case.

For learning more complicated structures, there exist a large variety of approximative techniques. One of the first general search algorithms was the K2 algorithm [Cooper and Herskovits, 1992] which relies on an initial variable ordering, and greedily adds parents for each node, in order to increase the BD score. A more general scheme is hill-climbing (HC) [Chickering et al., 1995, Heckerman et al., 1995], which does not rely on an initial variable ordering. HC starts with an unconnected graph $\mathcal{G}$. In each iteration, the current graph $\mathcal{G}$ is replaced with a neighboring graph $\mathcal{G}'$ with maximal score, if $\text{score}(\mathcal{G}') > \text{score}(\mathcal{G})$. Neighboring graphs are those graphs which can be created from the current graph by single edge-insertions, edge-deletions and edge-reversals, maintaining acyclicity constraints. When no neighboring graph has higher score than the current graph, the algorithm stops. HC greedily increases the score, and will typically terminate in a local maximum. HC can be combined with tabu-search [Glover, 1977] in order to escape local maxima: When the algorithm is trapped in a local maximum, then the last several iterations leading to the local maximum are reverted and marked as forbidden. In this way, the algorithm is forced to use alternative iterations, possibly leading to a better local maximum. A related method is simulated annealing (SA) which randomly generates a neighbor solution in each step. If the new solution has higher score than the current one, it is immediately accepted. However, also a solution with lower score is accepted with a certain probability. While this probability is high in the beginning, leading to a large exploration over the search space, it is successively reduced according to a cooling schedule. At the end of the search process, SA only accepts score-increasing solutions. Chickering et al. [1995] applied SA for BN structure learning. While these heuristic methods perform a greedy search in the space of all DAGs, Chickering [2002] proposed to greedily search over the space of equivalence classes. An equivalence class contains all BN structures which represent the same set of independence assumptions. A related approach can be found in [Teyssier and Koller, 2005], which proposes to perform a search over possible variable orderings. This approach uses the fact, as already mentioned in [Cooper and Herskovits, 1992], that for a given variable ordering the optimal structure can be found in polynomial time.

Recently, several exact approaches for finding globally optimal structures have been proposed . Methods based on dynamic programming [Koivisto and Sood, 2004, Singh and Moore, 2005, Silander and Myllymä 2006] have exponential time and memory requirements, which restricts their application to approximately 30 variables. Furthermore, there are branch-and-bound methods [Suzuki, 1996, de Campos et al., 2009, Jaakkola et al., 2010] which cast the combinatorial structure learning problem to a relaxed continuous problem. The relaxed continuous problem is successively divided (branched) into sub-problems, until the solutions of the relaxed sub-problems coincide with feasible structures (DAGs). The scores of the relaxed solutions provide an upper bound of the achievable score, and each feasible solution provides a lower bound of this score. Branches whose upper bound is lower than the currently best lower bound can consequently be pruned. Although branch-and-bound methods also have an exponential runtime, they provide two key advantages: (i) They maintain (after some initial time) a feasible solution, i.e. they can be prematurely terminated, returning the currently best solution. Methods based on dynamic programming do not offer this possibility; (ii) They provide upper and lower bounds on the maximal score, which represents a worst-case certificate of sub-optimality.

4.3 Discriminative Learning in Bayesian Networks

So far, we discussed generative learning of BNs, aiming to retrieve the underlying true distribution $\mathrm{P}^*(\mathbf{X})$ from a sample $\mathcal{D} = (\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(L)})$. Now, let us assume that our final goal is classification, i.e. we aim to predict the value of a class variable $C \in \mathbf{X}$, given the values of the features $\mathbf{Z} = \mathbf{X} \setminus \{C\}$. Without loss of generality, we can assume that C is X_1, and we renumber the remaining variables, such that $\mathbf{X} = \{C, \mathbf{Z}\} = \{C, X_1, X_2, \ldots, X_{N-1}\}$. In classification, we are interested in finding a decision

function $f(\mathbf{Z})$, which can take values in $\{1, \ldots, \mathbf{sp}(C)\}$, such that the expected classification rate

$$\mathrm{CR}(f) = \mathbb{E}_{\mathrm{P}^*}\left[\mathbb{1}_{\{f(\mathbf{z})=c\}}\right] \tag{181}$$

$$= \sum_{\mathbf{x}\in\mathbf{val}(\mathbf{X})} \mathrm{P}^*(c, \mathbf{z})\, \mathbb{1}_{\{f(\mathbf{z})=c\}} \tag{182}$$

is maximized. It is easily shown that a function $f^*(\mathbf{Z})$ maximizing (181) is given by [Berger, 1985]

$$f^*(\mathbf{Z}) = \arg\max_{c\in\mathbf{val}(C)} \mathrm{P}^*(c|\mathbf{Z}) = \arg\max_{c\in\mathbf{val}(C)} \mathrm{P}^*(c, \mathbf{Z}). \tag{183}$$

Therefore, since the generative approach aims to retrieve $\mathrm{P}^*(\mathbf{X})$, we are in principle able to learn an optimal classifier, when (i) we use a sufficiently expressive model, (ii) we have a sufficient amount of training data, and (iii) we train the model using a generative approach, such as ML. Classifiers trained this way are called *generative classifiers*. Generative classifiers are amenable to interpretation and for incorporating prior information, and are flexible to versatile inference scenarios. However, we rarely have sufficient training data, and typically we restrict the model complexity, partly to avoid overfitting (which occurs due to a lack of training data), partly due to computational reasons. Therefore, although theoretically sound, generative classifiers usually do not yield the best classification results.

Discriminative approaches, on the other hand, directly represent aspects of the model which are important for classification accuracy. For example, some approaches model the class posterior probability $\mathrm{P}(C|\mathbf{Z})$, while others, such as support vector machines (SVMs) [Schölkopf and Smola, 2001, Burges, 1998] or neural networks [Bishop, 2006, 1995], model information about the decision function, without using a probabilistic model. In each case, a discriminative objective does not necessarily agree with accurate joint distribution estimates, but rather aims for a low classification error on the training data. Discriminative models are usually restricted to the mapping from observed input features $\mathbf{Z}$ to the unknown class output variable C, while inference of $\mathbf{Z}$ for given C is impossible or unreasonable. There are several reasons for using discriminative rather than generative classifiers, one of which is that the classification problem should be solved most simply and directly, and never via a more general problem such as the intermediate step of estimating the joint distribution [Vapnik, 1998]. Discriminative classifiers typically lead to better classification performance, particularly when the class conditional distributions poorly approximate the true distribution [Bishop, 2006]. The superior performance of discriminative classifiers has been reported in many application domains.

When using PGMs for classification, both parameters and structure can be learned using either generative or discriminative objectives [Pernkopf and Bilmes, 2010]. In the generative approach, the central object is the parameter posterior $\mathrm{P}(\boldsymbol{\Theta}|\mathcal{D})$, yielding the MAP, ML and the Bayesian approach. In the discriminative approach, we use alternative objective functions such as conditional log-likelihood (CLL), classification rate (CR), or margin, which can be applied for both structure learning and for parameter learning. Unfortunately, while essentially all generative scores (likelihood, MDL/BIC, BDe) decompose into a sum over the nodes, it turns out that this does not happen for discriminative scores. Therefore, although many of the structure learning algorithms discussed for the generative case can also be applied for discriminative scores, the problem becomes computationally more expensive in the discriminative case. We restrict our discussion to BN, since most of the literature for discriminative learning is concerned about this type of PGMs.

Conditional Log-Likelihood (CLL). The log-likelihood over a model class $\mathcal{C}$ can be written as

$$\log \mathcal{L}(\mathcal{C}; \mathcal{D}) = \sum_{l=1}^{L} \log \mathrm{P}(\mathbf{x}^{(l)}|\mathcal{C}) \tag{184}$$

$$= \sum_{l=1}^{L} \log \mathrm{P}(\mathbf{c}^{(l)}|\mathbf{z}^{(l)}, \mathcal{C}) + \sum_{l=1}^{L} \log \mathrm{P}(\mathbf{z}^{(l)}|\mathcal{C}) \tag{185}$$

$$= \mathrm{CLL}(\mathcal{C}; \mathcal{D}) + \log \mathcal{L}(\mathcal{C}; \mathcal{D}_{\mathbf{Z}}), \tag{186}$$

where the log-likelihood decomposes into two terms. The term $\log \mathcal{L}(\mathcal{C}; \mathcal{D}_{\mathbf{Z}})$ is the log-likelihood of the attributes $\mathbf{Z}$, where the data set $\mathcal{D}_{\mathbf{Z}}$ contains only the samples of the features $\mathbf{Z}$, without the values

of C. The term $\mathrm{CLL}\,(\mathcal{C};\mathcal{D})$ is called *conditional log-likelihood*, which is the log-likelihood of the class variable C, conditioned on the respective values of $\mathbf{Z}$. While $\mathrm{CLL}\,(\mathcal{C};\mathcal{D})$ reflects how well some model $M \in \mathcal{C}$ fits the empirical conditional distribution $\mathrm{P}_{\mathcal{D}}(C|\mathbf{Z})$, the term $\mathcal{L}(\mathcal{C};\mathcal{D}_{\mathbf{Z}})$ measures the fitness of M to represent the empirical distribution over the features $\mathrm{P}_{\mathcal{D}_{\mathbf{Z}}}(\mathbf{Z})$. For discriminative learning, we are primarily interested in optimizing $\mathrm{CLL}\,(\mathcal{C};\mathcal{D})$, but not $\mathcal{L}(\mathcal{C};\mathcal{D}_{\mathbf{Z}})$. Therefore, for discriminative learning we discard the second term in (186) and maximize the conditional likelihood:

$$\mathrm{CLL}\,(\mathcal{C};\mathcal{D}) = \sum_{l=1}^{L}\left[\log \mathrm{P}(c^{(l)},\mathbf{z}^{(l)}|\mathcal{C}) - \log \sum_{c=1}^{|C|} \mathrm{P}(c,\mathbf{z}^{(l)}|\mathcal{C})\right]. \tag{187}$$

Suppose we found a model M^* maximizing the CLL, and we wish to classify a new sample $\mathbf{z}$. The decision function according to M^* is then

$$f_{M^*}(\mathbf{z}) = \arg \max_{c \in \mathbf{val}(C)} \mathrm{P}(c|\mathbf{z}, M^*). \tag{188}$$

However, although CLL is connected with classification rate, maximizing CLL is not the same as maximizing the classification rate on the training set. Friedman [1997] shows that improving CLL even can be obstructive for classification. This can be understood in a wider context, when our goal is not to maximize the number of correct classifications, but to make an optimal decision based on the classification result. As an example given in [Bishop, 2006], assume a system which classifies if a patient has a lethal disease, based on some medical measurements. There are 4 different combinations of the state of the patient and the system diagnosis, which have very different consequences: the scenario that a healthy patient is diagnosed as ill is not as disastrous as the scenario when an ill patient is diagnosed as healthy. For such a system it is not enough to return a 0/1-decision, but additionally a value of confidence has to be provided. These requirements are naturally met by maximum CLL learning. Generally, one can define a loss-function $L(k,l)$ which represents the assumed loss when a sample with class $c = k$ is classified as $c = l$. Maximizing CLL is tightly connected with minimizing the expected loss for a loss-function $L(k,l)$ satisfying $L(k,k) = 0$, and $L(k,l) \geq 0, k \neq l$.

In [Greiner and Zhou, 2002], an algorithm for learning the BN parameters for a fixed structure $\mathcal{G}$ was proposed. This algorithm uses gradient ascend to increase the CLL, while maintaining parameter feasibility, i.e. nonnegativity and sum-to-one constraints. Although CLL is a concave function, these constraints are non-convex, such that the algorithm generally finds only a local maximum of the CLL. However, in [Wettig et al., 2003, Roos et al., 2005] it is shown that for certain network structures the parameter constraints can be neglected, yielding a convex optimization problem and thus a globally optimal solution. More precisely, the network structure $\mathcal{G}$ has to fulfill the condition:

$$\forall Z \in \mathbf{Ch}_{\mathcal{G}}(C):\ \exists X \in \mathbf{Pa}_{\mathcal{G}}(Z):\ \mathbf{Pa}_{\mathcal{G}}(Z) \subseteq \mathbf{Pa}_{\mathcal{G}}(X) \cup \{X\}. \tag{189}$$

In words, every class child Z_i must have a parent X, which together with its own parents $\mathbf{Pa}_{\mathcal{G}}(X)$ contain all parents $\mathbf{Pa}_{\mathcal{G}}(Z_i)$. It is shown that for structures fulfilling this condition, always a correctly normalized BN with the same CLL objective value as for the unconstrained solution can be found. This implies that the obtained BN parameters globally maximize the CLL. In [Grossman and Domingos, 2004], a greedy hill-climbing algorithm for BN structure learning is proposed, using CLL as scoring function.

Empirical Classification Rate (CR). Often one is interested in the special case of 0/1-loss, where each correctly classified sample has loss 0, while each misclassified sample causes a loss of 1. As pointed out in [Friedman, 1997], maximum CLL training does not directly correspond to optimal 0/1-loss. An objective directly reflecting the 0/1-loss is the empirical classification rate

$$\mathrm{CR}_{\mathcal{D}}\,(\mathcal{C};\mathcal{D}) = \frac{1}{L}\sum_{l=1}^{L} \mathbb{1}_{\{c^{(l)} = \arg\max_{c \in \mathbf{val}(C)} \mathrm{P}(c|\mathbf{z}^{(l)},\mathcal{C})\}}.$$

Parameter learning using $\mathrm{CR}_{\mathcal{D}}$ is hard, due to the non-differentiability and non-convexity of the function. In [Keogh and Pazzani, 1999] and [Pernkopf and Bilmes, 2010], $\mathrm{CR}_{\mathcal{D}}$ was used for greedy hill-climbing structure learning.

Maximum Margin (MM). The probabilistic multi-class margin [Guo et al., 2005] for the l^{th} sample can be expressed as

$$d_{\mathcal{C}}^{(l)} = \min_{c \neq c^{(l)}} \frac{\mathrm{P}\left(c^{(l)}|\mathbf{z}^{(l)}, \mathcal{C}\right)}{\mathrm{P}\left(c|\mathbf{z}^{(l)}, \mathcal{C}\right)} = \frac{\mathrm{P}\left(c^{(l)}, \mathbf{z}^{(l)}|\mathcal{C}\right)}{\max_{c \neq c^{(l)}} \mathrm{P}\left(c, \mathbf{z}^{(l)}|\mathcal{C}\right)}. \tag{190}$$

If $d_{\mathcal{C}}^{(l)} > 1$, then sample l is correctly classified and vice versa. The magnitude of $d_{\mathcal{C}}^{(l)}$ is related to the confidence of the classifier about its decision. Taking the logarithm, we obtain

$$\log d_{\mathcal{C}}^{(l)} = \log \mathrm{P}\left(c^{(l)}, \mathbf{z}^{(l)}|\mathcal{C}\right) - \max_{c \neq c^{(l)}} \left(\log \mathrm{P}\left(c, \mathbf{z}^{(l)}|\mathcal{C}\right)\right). \tag{191}$$

Usually, the maximum margin approach maximizes the margin of the sample with the smallest margin for a separable classification problem [Schölkopf and Smola, 2001], i.e. we use the objective function

$$\mathrm{MM}\left(\mathcal{C}; \mathcal{D}\right) = \min_{l=1,\ldots,L} \log d_{\mathcal{C}}^{(l)}. \tag{192}$$

For non-separable problems, this is relaxed by introducing a so-called soft margin, which has been used for parameter learning [Guo et al., 2005], [Pernkopf et al., 2011b] and for structure learning [Pernkopf et al., 2011a].

5 Inference

In the last section the basics of learning PGMs have been introduced. Now, assuming that the process of learning has already been performed, i.e. the model structure and parameterization are established, we move on to the task of *inference*. This task deals with assessing the marginal and/or most likely configuration of variables given certain observations. For this, the set of RVs $\mathbf{X}$ in the PGM is partitioned into the three mutually disjoint subsets $\mathbf{O}$, $\mathbf{Q}$ and $\mathbf{H}$, i.e. $\mathbf{X} = \mathbf{O} \cup \mathbf{Q} \cup \mathbf{H}$ and $\mathbf{O} \cap \mathbf{Q} = \mathbf{O} \cap \mathbf{H} = \mathbf{Q} \cap \mathbf{H} = \emptyset$. Here, $\mathbf{O}$ denotes the set of *observed nodes*, i.e. the evidence variables, $\mathbf{Q}$ denotes the set of *query variables*, and $\mathbf{H}$ refers to the set of nodes which belong neither to $\mathbf{O}$ or $\mathbf{Q}$, also known as latent or hidden variables.

There are two basic types of inference queries:

(i) **Marginalization query.** This first type of query infers the marginal distribution of the query variables conditioned on the observation $\mathbf{O}$, i.e. it computes

$$\mathrm{P}(\mathbf{Q}|\mathbf{O} = \mathbf{o}) = \frac{\mathrm{P}(\mathbf{Q}, \mathbf{O} = \mathbf{o})}{\mathrm{P}(\mathbf{O} = \mathbf{o})}. \tag{193}$$

The terms $\mathrm{P}(\mathbf{Q}, \mathbf{O} = \mathbf{o})$ and $\mathrm{P}(\mathbf{O} = \mathbf{o})$ can be determined by marginalization over $\mathbf{H}$ and $\mathbf{H} \cup \mathbf{Q}$ of the joint probability $\mathrm{P}(\mathbf{X}) = \mathrm{P}(\mathbf{O}, \mathbf{Q}, \mathbf{H})$ represented by the PGM, respectively. That is,

$$\mathrm{P}(\mathbf{Q}, \mathbf{O} = \mathbf{o}) = \sum_{\mathbf{h} \in \mathbf{val}(\mathbf{H})} \mathrm{P}(\mathbf{O} = \mathbf{o}, \mathbf{Q}, \mathbf{H} = \mathbf{h}), \quad \text{and} \tag{194}$$

$$\mathrm{P}(\mathbf{O} = \mathbf{o}) = \sum_{\mathbf{q} \in \mathbf{val}(\mathbf{Q})} \mathrm{P}(\mathbf{O} = \mathbf{o}, \mathbf{Q} = \mathbf{q}). \tag{195}$$

Example 22. *Assume a PGM used by a car insurance company to calculate the insurance fees for individuals. Then, a query could ask for the probabilities of certain damage sums (this corresponds to query variables) arising for the case of a driver with age 25 and no children (this corresponds to evidence variables). Other variables in the model, such as e.g. the number of accidents in the past, are marginalized out since they are not observed.*

(ii) **Maximum a-posteriori query.** This query determines the most likely instantiation of the query variables given some evidence, i.e. it computes

$$\mathbf{q}^* = \arg \max_{\mathbf{q} \in \mathbf{val}(\mathbf{Q})} \mathrm{P}(\mathbf{Q}|\mathbf{O} = \mathbf{o}). \tag{196}$$

This is equivalent to

$$\mathbf{q}^* = \arg \max_{\mathbf{q} \in \mathbf{val}(\mathbf{Q})} \sum_{\mathbf{h} \in \mathbf{val}(\mathbf{H})} \mathrm{P}(\mathbf{Q} = \mathbf{q}, \mathbf{H} = \mathbf{h} | \mathbf{O} = \mathbf{o}) \tag{197}$$

$$= \arg \max_{\mathbf{q} \in \mathbf{val}(\mathbf{Q})} \sum_{\mathbf{h} \in \mathbf{val}(\mathbf{H})} \mathrm{P}(\mathbf{Q} = \mathbf{q}, \mathbf{H} = \mathbf{h}, \mathbf{O} = \mathbf{o}). \tag{198}$$

Note, that the most likely instantiation of the joint variables $\mathbf{q}^*$ does in general not correspond to the states obtained by selecting the most likely instantiation for all variables in $\mathbf{Q}$ individually. An example for this is shown in Koller and Friedman [2009], Chapter 2.

Example 23. *In hidden Markov models (HMMs) this amounts to finding the most likely state sequence* $\mathbf{q}^*$ *explaining the given observation sequence* $\mathbf{o}$ *(in HMMs* $\mathbf{H} = \emptyset$*) and is usually determined by applying the Viterbi algorithm. A short introduction to HMMs is provided in Section 6.1.*

While both inference queries can in principle be answered by directly evaluating the sums in the corresponding equations, this approach becomes intractable for PGMs with many variables. This is due to the number of summands growing exponentially with the number of variables. Therefore, more efficient inference methods exploiting the structure of the underlying PGMs have been developed: The exact solution to (i) can be found using the *sum-product algorithm*, also known as *belief propagation* or *message passing algorithm* [Pearl, 1988, Yedidia et al., 2002], assuming tree-structured graphical models, and (ii) can be calculated by applying the *max-product* or in the log-space the *max-sum* algorithm. In this tutorial, the focus is on marginalization queries $\mathrm{P}(\mathbf{Q}|\mathbf{O})$. Details on MAP queries are provided in Koller and Friedman [2009], Chapter 9.

The remainder of this section is structured as follows: In Section 5.1 the sum-product algorithm is derived and the junction tree algorithm for exact inference in arbitrary graphs is introduced. In cases, where exact inference is computationally too demanding, approximate inference techniques can be used. A short introduction to these techniques is given in Section 5.2.

5.1 Exact Inference

Given some evidence $\mathbf{o} \in \mathbf{val}(\mathbf{O})$, the direct calculation of the marginal in (195) of the joint probability distribution

$$\mathrm{P}(\mathbf{O} = \mathbf{o}) = \sum_{\mathbf{q} \in \mathbf{val}(\mathbf{Q})} \sum_{\mathbf{h} \in \mathbf{val}(\mathbf{H})} \mathrm{P}(\mathbf{O} = \mathbf{o}, \mathbf{Q} = \mathbf{q}, \mathbf{H} = \mathbf{h}) \tag{199}$$

expands to $\prod_{i=1}^{|\mathbf{Q}|} \mathbf{sp}\left(\mathbf{Q}_i\right) \prod_{i=1}^{|\mathbf{H}|} \mathbf{sp}\left(\mathbf{H}_i\right)$ summations. Hence, assuming that each variable can be in one of k states the evaluation of Equation (199) requires $\mathcal{O}\left(k^{|\mathbf{Q}|+|\mathbf{H}|}\right)$ summations, i.e. the evaluation of a sum with exponentially many terms in the number of variables. This renders the direct computation intractable for large values of k and many variables. Fortunately, the underlying graph structure, and the thereby introduced factorization of the joint probability, can be exploited. This is demonstrated in the following example:

Example 24. *Consider the MN $\mathcal{M}$ in Figure 21. The joint probability factorizes according to the maximal cliques $\mathbf{C}_1 = \{X_1, X_4\}$, $\mathbf{C}_2 = \{X_2, X_4\}$, $\mathbf{C}_3 = \{X_3, X_4\}$, $\mathbf{C}_4 = \{X_4, X_5\}$, $\mathbf{C}_5 = \{X_5, X_6\}$, and $\mathbf{C}_6 = \{X_5, X_7\}$ as*

$$P_{\mathcal{M}}(X_1, X_2, X_3, X_4, X_5, X_6, X_7) = \frac{1}{Z}\Psi_{\mathbf{C}_1}(X_1, X_4)\Psi_{\mathbf{C}_2}(X_2, X_4)\Psi_{\mathbf{C}_3}(X_3, X_4)\cdot \tag{200}$$

$$\Psi_{\mathbf{C}_4}(X_4, X_5)\Psi_{\mathbf{C}_5}(X_5, X_6)\Psi_{\mathbf{C}_6}(X_5, X_7). \tag{201}$$

Suppose, we are interested in determining the marginal probability $P_{\mathcal{M}}(X_4)$. Then,

$$\begin{aligned}
P_{\mathcal{M}}(X_4) &= \frac{1}{Z}\sum_{x_1}\cdots\sum_{x_3}\sum_{x_5}\cdots\sum_{x_7} P_{\mathcal{M}}(X_1, X_2, X_3, X_4, X_5, X_6, X_7) \\
&= \frac{1}{Z}\underbrace{\left[\sum_{x_1\in\mathbf{val}(X_1)}\Psi_{\mathbf{C}_1}(x_1, x_4)\right]}_{=\mu_{X_1\to X_4}(x_4)}\underbrace{\left[\sum_{x_2\in\mathbf{val}(X_2)}\Psi_{\mathbf{C}_2}(x_2, x_4)\right]}_{=\mu_{X_2\to X_4}(x_4)}\underbrace{\left[\sum_{x_3\in\mathbf{val}(X_3)}\Psi_{\mathbf{C}_3}(x_3, x_4)\right]}_{=\mu_{X_3\to X_4}(x_4)} \\
&\quad\cdot\underbrace{\left[\sum_{x_5\in\mathbf{val}(X_5)}\Psi_{\mathbf{C}_4}(x_4, x_5)\underbrace{\left[\sum_{x_6\in\mathbf{val}(X_6)}\Psi_{\mathbf{C}_5}(x_5, x_6)\right]}_{=\mu_{X_6\to X_5}(x_5)}\underbrace{\left[\sum_{x_7\in\mathbf{val}(X_7)}\Psi_{\mathbf{C}_6}(x_5, x_7)\right]}_{=\mu_{X_7\to X_5}(x_5)}\right]}_{=\mu_{X_5\to X_4}(x_4)} \\
&= \frac{1}{Z}\mu_{X_1\to X_4}(x_4)\,\mu_{X_2\to X_4}(x_4)\,\mu_{X_3\to X_4}(x_4)\,\mu_{X_5\to X_4}(x_4)\,,
\end{aligned}$$

where the terms $\mu_{X_i\to X_j}(x_j)$ denote so-called messages *from X_i to X_j. The grouping of the factors with the corresponding sums essentially exploits the distributive law [Aji and McEliece, 2000] and reduces the computational burden. In this example, we require $\mathcal{O}(k^2)$ summations and multiplications assuming that $\mathbf{sp}(X_i) = k$ for all variables X_i. Direct calculation of $P_{\mathcal{M}}(X_4)$ would instead require $\mathcal{O}(k^N)$ arithmetic operations, where $N = 7$. The normalization constant Z can be determined by summing the product of incoming messages over x_4, i.e.*

$$Z = \sum_{x_4\in\mathbf{val}(X_4)}\mu_{X_1\to X_4}(x_4)\,\mu_{X_2\to X_4}(x_4)\,\mu_{X_3\to X_4}(x_4)\,\mu_{X_5\to X_4}(x_4)\,. \tag{202}$$

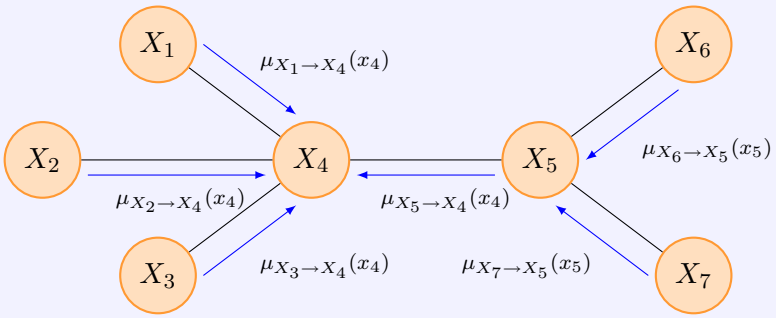


Figure 21: Message passing in a tree-shaped MN $\mathcal{M}$ to determine the marginal $\mathrm{P}_{\mathcal{M}}(X_4)$.

In the sequel, we generalize the example above to pairwise MNs, while inference in more general MNs, BNs and FGs is covered later in this section. This generalization leads to the *sum-product rule*: Messages $\mu_{X_i\to X_j}(x_j)$ are sent on each edge $(X_i - X_j) \in \mathbf{E}$ from X_i to X_j. These messages are updated

according to

$$\mu_{X_i \to X_j}(x_j) = \sum_{x_i \in \mathbf{val}(X_i)} \Psi_{\{X_i, X_j\}}(x_i, x_j) \prod_{X_k \in (\mathbf{Nb}_{\mathcal{G}}(X_i) \setminus \{X_j\})} \mu_{X_k \to X_i}(x_i)\,, \tag{203}$$

i.e. the new message $\mu_{X_i \to X_j}(x_j)$ is calculated as the product of the *incoming* messages with the potential function $\Psi_{\{X_i, X_j\}}$ and summation over all variables except X_j. This message update is sketched in Figure 22.

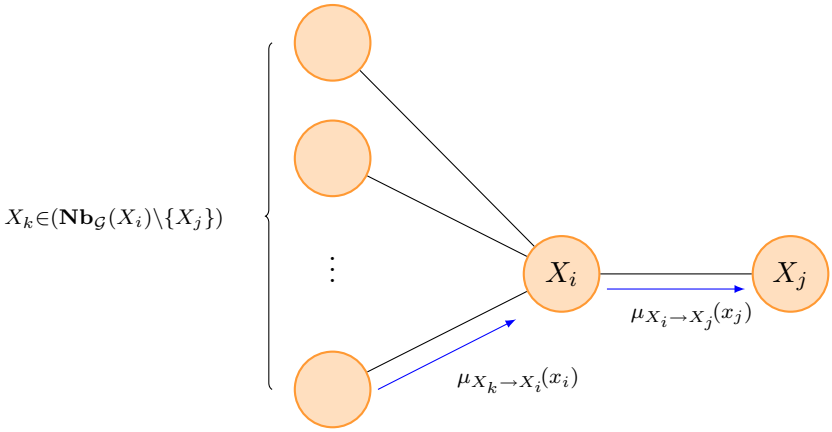


Figure 22: Sum-product update rule.

The *sum-product algorithm* schedules the messages such that an updated message is sent from X_i to X_j when X_i has received the messages from all its neighbors except X_j, i.e. from all nodes in $\mathbf{Nb}_{\mathcal{G}}(X_i) \setminus \{X_j\}$ [Jensen, 1996].

Example 25. *This message update schedule of the sum-product algorithm for our previous example is illustrated in Figure 23. The sent messages are illustrated as arrows in the figure, where the numbers next to the arrows indicate the iteration count at which message updates are sent to neighboring nodes. After three iterations the updates converge for all nodes in the graph.*

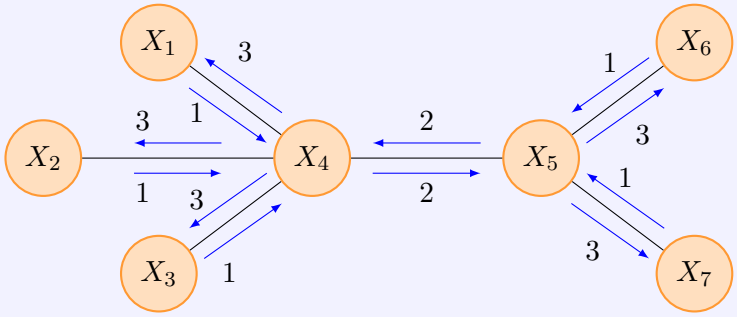


Figure 23: Message update of the sum-product algorithm.

The sum-product algorithm is *exact* if the MN is free of cycles [Pearl, 1988], i.e. if it is a tree. *Exact* means that the marginals $\mathrm{P}(X_i)$ (also called *beliefs*) obtained from the sum-product algorithm according to

$$\mathrm{P}(X_i) = \frac{1}{Z} \prod_{X_k \in \mathbf{Nb}_{\mathcal{G}}(X_i)} \mu_{X_k \to X_i}(x_i) \tag{204}$$

are guaranteed to converge to the true marginals. Again, considering the graph in Figure 23, each clique factor $\Psi_{\mathbf{C}}$ includes two variables. Consequently, the complexity of the sum-product algorithm is $\mathcal{O}(k^2)$ assuming that $\mathbf{sp}\,(X_i) = k$ for all variables X_i.

So far, we are able to obtain the marginals $\mathrm{P}(X_i)$ for any $X_i \in \mathbf{Q}$ for tree-shaped MNs. However, the aim is to determine the marginal distribution over the query variable $Q_i \in \mathbf{Q}$ conditioned on the evidence $\mathbf{O} = \mathbf{o}$. This requires the computation of $\mathrm{P}(Q_i, \mathbf{o})$, cf. Equation (193) assuming only one query variable. The evidence is introduced to each clique factor $\Psi_{\mathbf{C}}$ that depends on evidence nodes, i.e. to each $\Psi_{\mathbf{C}}$ for which there is an $O_i \in \mathbf{C}$ that is also in $\mathbf{O}$. Therefore, we identify all factors $\Psi_{\mathbf{C}}$ with $O_i \in \mathbf{C}$ for any $O_i \in \mathbf{O}$ and determine the new factor by multiplying $\Psi_{\mathbf{C}}$ with the indicator function $\mathbb{1}_{\{O_i=o_i\}}$, i.e. $\Psi_{\mathbf{C}}(\cdot)$ is modified according to $\Psi_{\mathbf{C}}(\cdot)\mathbb{1}_{\{O_i=o_i\}}$. In other words, factors are set to zero for instantiations inconsistent with $\mathbf{O} = \mathbf{o}$. Interestingly, it turns out that running the sum-product algorithm on the replaced factors leads to an unnormalized representation of $\mathrm{P}(Q_i, \mathbf{o})$. The regular $\mathrm{P}(Q_i|\mathbf{o})$ is obtained by normalizing $\mathrm{P}(Q_i, \mathbf{o})$ by $Z = \sum_{q_i \in \mathbf{val}(Q_i)} \mathrm{P}(Q_i, \mathbf{o})$ [Jensen, 1996, Cowell et al., 1999].

Message Passing in FGs. For FGs, the sum-product algorithm for determining the marginal distribution of the variable X_i, given all the observations can be formulated analogously [Kschischang et al., 2001]. The observations are absorbed into the factors instead of the clique factors. Neighboring nodes communicate with each other: whenever a node, either a variable or factor node, has received all messages except on one edge, it sends a new message to its neighbor over this edge. At the beginning, messages from variable nodes are initialized to 1. The message from variable node X to factor node F is

$$\mu_{X \to F}(x) = \prod_{F_k \in (\mathbf{Nb}_{\mathcal{G}}(X) \setminus \{F\})} \mu_{F_k \to X}(x)\,,$$

while a message from factor node F to variable node X is computed as

$$\mu_{F \to X}(x) = \sum_{\mathbf{Nb}_{\mathcal{G}}(F) \setminus \{X\}} \left(f\left(\mathbf{Nb}_{\mathcal{G}}(F)\right) \prod_{X_k \in (\mathbf{Nb}_{\mathcal{G}}(F) \setminus \{X\})} \mu_{X_k \to F}(X_k) \right).$$

The marginal at each variable node can be determined as $\mathrm{P}(X_i) = \frac{1}{Z} \prod_{F \in \mathbf{Nb}_{\mathcal{G}}(X_i)} \mu_{F \to X_i}(X_i)$, where Z is a normalization constant.

So far, we have shown how to efficiently compute the marginals of a factorized joint probability distribution given some evidence in the case of tree-structured MNs and FGs. Generally, the sum-product algorithm is exact only if performed on a graphical model that is a tree. The remaining question is how to perform inference in BNs and arbitrarily structured BNs and MNs?

Message Passing in arbitrarily structured BNs and MNs. Inference in BNs can be performed by first transforming the BN into an MN. Inference is then performed on this transformed model. The transformation of the model is known as *moralization*. Further, any arbitrary directed (and undirected) graphical model can be transformed into an undirected tree (*junction tree* or *clique tree*) where each node in the tree corresponds to a subset of variables (cliques) [Cowell, 1999, Lauritzen and Spiegelhalter, 1988, Jensen, 1996, Cowell et al., 1999, Dawid, 1992, Jordan, 1999, Huang and Darwiche, 1996]. This is done with the goal of performing inference on this tree, exploiting that inference on trees is exact. The transformation of an arbitrary graph to a junction tree involves the following three steps, where the first step is only required when transforming a BN to an MN:

(i) **Moralization.** In the case of a directed graphical model, undirected edges are added between all pairs of parents $\mathbf{Pa}_{\mathcal{G}}(X_i)$ having a common child X_i for all $X_i \in \mathbf{X}$. This is known as moralization [Cowell et al., 1999]. The *moral graph* is then obtained by dropping the directions of the edges. Moralization essentially results in a loss of conditional independence statements represented by the original graph. However, this is necessary for representing the original factorization by the undirected graph.

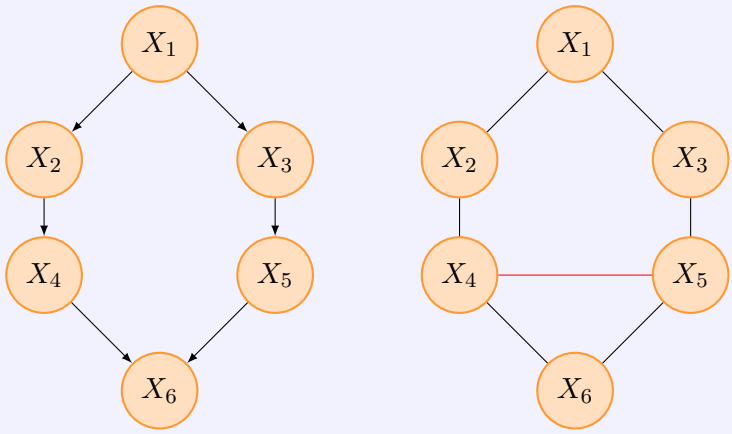
Example 26 (Moralization). *The joint probability of the BN $\mathcal{B}$ in Figure 24(a) is*

$$P_{\mathcal{B}}(X_1,\dots,X_6) = P_{X_1}(X_1)P_{X_2}(X_2|X_1)P_{X_3}(X_3|X_1)P_{X_4}(X_4|X_2)P_{X_5}(X_5|X_3)P_{X_6}(X_6|X_4,X_5). \tag{205}$$

The corresponding moral graph is shown in Figure 24(b). It factorizes as a product of the factors of the maximal cliques $\mathbf{C}_1 = \{X_1, X_2\}$, $\mathbf{C}_2 = \{X_1, X_3\}$, $\mathbf{C}_3 = \{X_2, X_4\}$, $\mathbf{C}_4 = \{X_3, X_5\}$, *and* $\mathbf{C}_5 = \{X_4, X_5, X_6\}$ *as*

$$P_{\mathcal{M}}(X_1,\dots,X_6) = \Psi_{\mathbf{C}_1}(X_1,X_2)\Psi_{\mathbf{C}_2}(X_1,X_3)\Psi_{\mathbf{C}_3}(X_2,X_4)\Psi_{\mathbf{C}_4}(X_3,X_5)\Psi_{\mathbf{C}_5}(X_4,X_5,X_6). \tag{206}$$

Each CPD $P_{X_i}(X_i|\mathbf{Pa}_{\mathcal{G}}(X_i))$ *of the BN is mapped into the factor of exactly one clique, i.e. each clique factor is initialized to one and then each* $P_{X_i}(X_i|\mathbf{Pa}_{\mathcal{G}}(X_i))$ *is multiplied into a clique factor which contains* $\{X_i\} \cup \mathbf{Pa}_{\mathcal{G}}(X_i)$. *In this example, this results in the following factors:* $\Psi_{\mathbf{C}_1}(X_1,X_2) = P_{X_1}(X_1)P_{X_2}(X_2|X_1)$, $\Psi_{\mathbf{C}_2}(X_1,X_3) = P_{X_3}(X_3|X_1)$, $\Psi_{\mathbf{C}_3}(X_2,X_4) = P_{X_4}(X_4|X_2)$, $\Psi_{\mathbf{C}_4}(X_3,X_5) = P_{X_5}(X_5|X_3)$, *and* $\Psi_{\mathbf{C}_5}(X_4,X_5,X_6) = P_{X_6}(X_6|X_4,X_5)$. *The normalization factor of* $P_{\mathcal{M}}(X_1,\dots,X_6)$ *in this case is* $Z = 1$.



(a) BN to be transformed to an MN. (b) Moral graph; the added edge is red.

Figure 24: Moralization example.

(ii) **Triangulation.** The next step in converting an arbitrary MN into a junction tree is the triangulation of the moral graph. The *triangulated graph* is sometimes called *chordal graph*.

Definition 28 (Triangulation [Cowell et al., 1999]). *A graph $\mathcal{G}$ is* triangulated *if there is no cycle of length ≥ 4 without an edge joining two non-neighboring nodes.*

The triangulated graph can be found by the *elimination algorithm* [Koller and Friedman, 2009] that eliminates one variable from the graph in each iteration: Assuming a given *elimination order* of the variables the first node from the ordering is selected and in the moral graph all pairs of neighbors of that node are connected. The resulting graph is collected in a set of graphs $\mathcal{G}_e$. Afterwards, this node and all its edges are removed from the graph. Then, the next node in the elimination ordering is selected for elimination. We proceed in the same way as above until all nodes have been eliminated. The graph resulting from the union of all the graphs in $\mathcal{G}_e$ is triangulated. Depending on the ordering, there typically exist many different triangulations. For efficient inference, we would like to choose a ordering such that e.g. the state-space size of the largest clique is minimal. Finding the optimal elimination ordering is NP-hard [Koller and Friedman, 2009]. Greedy algorithms for determining an ordering so that the induced graph is close to optimal are summarized in [Koller and Friedman, 2009], Chapter 9.

Example 27 (Triangulation)**.** *An example of a triangulated graph obtained from Figure 24(b) with elimination ordering $X_6, X_5, X_4, X_4, X_2, X_1$ is shown in Figure 25. The elimination of node X_5 and X_4 introduces the edges $(X_3 - X_4)$ and $(X_2 - X_3)$, respectively. The dashed edges are added to triangulate the moral graph.*

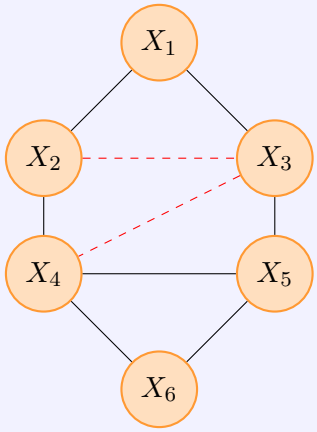


Figure 25: Triangulated graph; added edges are dashed.

Each maximal clique of the moral graph is contained in the triangulated graph $\mathcal{G}_t$. Using a similar mechanism as in Example 26, we can map the factors of the moral graph to the factors of the triangulated graph. The cliques in $\mathcal{G}_t$ are maximal cliques and

$$\mathrm{P}_{\mathcal{M}}(\mathbf{X}) = \frac{1}{Z} \prod_{\mathbf{C} \in \mathcal{G}_t} \Psi_{\mathbf{C}}(\mathbf{C}). \tag{207}$$

For efficiency, the cliques of the triangulated graph can be identified during variable elimination: A node which is eliminated from the moral graph forms a clique with all neighboring nodes, when connecting all pairs of neighbors. If this clique is not contained in a previously determined maximal clique, it is a maximal clique in $\mathcal{G}_t$.

Example 28. *The maximal cliques of the triangulated graph in Figure 25 are $\{X_4, X_5, X_6\}$, $\{X_3, X_4, X_5\}$, $\{X_2, X_3, X_4\}$, and $\{X_1, X_2, X_3\}$.*

(iii) **Building a junction tree.** In the last step, the maximal cliques are treated as nodes and are connected to form a tree. The junction tree has to fulfill the *running intersection property*, meaning that a variable X_i existing in two different cliques $\mathbf{C}_i$ and $\mathbf{C}_j$, $i \neq j$, has to be also in each clique on the path connecting these two cliques[2]. The cliques in the junction tree are connected by a *separator set* $\mathbf{S} = \mathbf{C}_i \cap \mathbf{C}_j$. The number of variables in $\mathbf{S}$ determine the edge weight for joining these two cliques in the junction tree. These weights can be used to find the junction tree with maximal sum of the weights using the *maximum spanning tree* algorithm [Kruskal, 1956]. The identified tree is guaranteed to satisfy the running intersection property.

[2]This property is important for the message passing algorithm to achieve local consistency between cliques. Because of this property, local consistency implies global consistency [Koller and Friedman, 2009].

Example 29 (Junction Tree)**.** *A junction tree of the example in Figure 25 with separators (square nodes) is shown in Figure 26.*

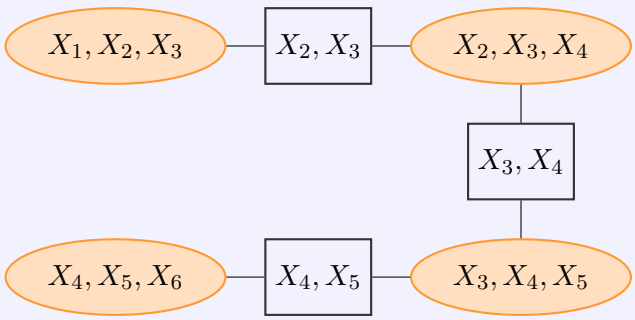


Figure 26: Junction tree.

Once the junction tree is determined, message passing between the cliques is performed for inference. This essentially amounts to the sum-product algorithm. First, the potential of a separator $\Psi_{\mathbf{S}}(\mathbf{S})$ is initialized to unity. The clique potential of neighboring cliques $\mathbf{C}_i$ and $\mathbf{C}_j$ is $\Psi_{\mathbf{C}_i}(\mathbf{C}_i)$ and $\Psi_{\mathbf{C}_j}(\mathbf{C}_j)$, respectively. The message passed from $\mathbf{C}_i$ to $\mathbf{C}_j$ is determined as

$$\tilde{\Psi}_{\mathbf{S}}(\mathbf{S}) \leftarrow \sum_{\mathbf{C}_i \setminus \mathbf{S}} \Psi_{\mathbf{C}_i}(\mathbf{C}_i), \quad \text{and}$$

$$\Psi_{\mathbf{C}_j}(\mathbf{C}_j) \leftarrow \frac{\tilde{\Psi}_{\mathbf{S}}(\mathbf{S})}{\Psi_{\mathbf{S}}(\mathbf{S})} \Psi_{\mathbf{C}_j}(\mathbf{C}_j),$$

where the sum is over all variables in $\mathbf{C}_i$ which are not in $\mathbf{S}$.

The message passing schedule is similar as for tree-structured models, i.e. a clique sends a message to a neighbor after receiving all the messages from its other neighbors. Furthermore, the introduction of evidence in the junction tree can be performed likewise as above. After each node has sent and received a message, the junction tree is *consistent* and the marginals $\mathrm{P}(Q_i, \mathbf{o})$ can be computed by identifying the clique potential $\Psi_{\mathbf{C}_i}(\mathbf{C}_i)$ which contains Q_i and summing over $\mathbf{C}_i \setminus \{Q_i\}$ as

$$\mathrm{P}(Q_i, \mathbf{o}) = \sum_{\mathbf{C}_i \setminus \{Q_i\}} \Psi_{\mathbf{C}_i}(\mathbf{C}_i).$$

The computational complexity of exact inference in discrete networks is exponential in the size of the largest clique (the width of the tree[3]). Hence, there is much interest in finding an optimal triangulation. Nevertheless, exact inference is intractable in large and dense PGMs and approximation algorithms are necessary.

5.2 Approximate Inference

In the past decade, the research in PGMs has focused on the development of efficient *approximate inference* algorithms. While exact inference algorithms provide a satisfactory solution to many inference and learning problems, there are large-scale problems where a recourse to approximate inference is inevitable. In the following, we provide a brief overview of the most popular approximate inference methods. Details can be found in Wainwright and Jordan [2008], Koller and Friedman [2009], Bishop [2006].

5.2.1 Variational Methods

The underlying idea in variational methods [Bishop, 2006, Wainwright and Jordan, 2008, Jordan et al., 1999, Ghaharamani and Jordan, 1997, Jaakkola, 2000] is the approximation of the posterior probability

[3]The tree-width of a graph is defined as the size (i.e. number of variables) of the largest clique of the moralized and triangulated directed graph minus one.

distribution $\mathrm{P}(\mathbf{Q}|\mathbf{O})$, assuming $\mathbf{H} = \emptyset$, by a tractable surrogate distribution $g(\mathbf{Q})$, where we assume that the evaluation of $\mathrm{P}(\mathbf{Q}, \mathbf{O})$ is computational less expensive than that of $\mathrm{P}(\mathbf{Q}|\mathbf{O})$ and $\mathrm{P}(\mathbf{O})$. The logarithm of $\mathrm{P}(\mathbf{O})$ can be decomposed as

$$\ln \mathrm{P}(\mathbf{O}) = \underbrace{\sum_{\mathbf{q}} g(\mathbf{q}) \ln \left[\frac{\mathrm{P}(\mathbf{O}, \mathbf{q})}{g(\mathbf{q})} \right]}_{\mathrm{LB}(g)} + \underbrace{\left\{ - \sum_{\mathbf{q}} g(\mathbf{q}) \ln \left[\frac{\mathrm{P}(\mathbf{q}|\mathbf{O})}{g(\mathbf{q})} \right] \right\}}_{\mathrm{KL}(g||\mathrm{P})},$$

where the term $\mathrm{LB}(g)$ denotes a lower bound for $\ln \mathrm{P}(\mathbf{O})$, i.e. $\mathrm{LB}(g) \leq \ln \mathrm{P}(\mathbf{O})$, and $\mathrm{KL}\,(g||\mathrm{P})$ denotes the Kullback-Leibler divergence between $g(\mathbf{Q})$ and $\mathrm{P}(\mathbf{Q}|\mathbf{O})$. Since $\ln \mathrm{P}(\mathbf{O})$ does not depend on $g(\mathbf{q})$, and $\mathrm{LB}(g)$ and $\mathrm{KL}\,(g||\mathrm{P})$ are non-negative, maximizing $\mathrm{LB}(g)$ amounts to minimizing $\mathrm{KL}\,(g||\mathrm{P})$. Hence, maximizing $\mathrm{LB}(g)$ with respect to $g(\mathbf{q})$ results in the best approximation of the posterior $\mathrm{P}(\mathbf{Q}|\mathbf{O})$.

In variational approaches, $g(\mathbf{Q})$ is restricted to simple tractable distributions. The simplest possibility, also called *mean-field approximation*, assumes that $g(\mathbf{Q})$ factorizes as

$$g(\mathbf{Q}) = \prod_{i=1}^{|\mathbf{Q}|} g_i(Q_i). \tag{208}$$

The variational inference approach can be combined with exact inference applied on parts of the graph. This is called *structured variational approximation* [Ghaharamani and Jordan, 1997]. Thereby, much of the structure of the original graph is preserved.

5.2.2 Loopy message passing

While message passing algorithms can be shown to be exact on PGMs with tree structure, convergence of these algorithms on arbitrary graphs with cycles (loops) is not guaranteed. Moreover, even after convergence, the resulting solution might be only an approximation of the exact solution. Surprisingly, message passing on *loopy graphs* often converges to stable posterior/marginal probabilities. The most significant breakthrough came with the insight that for certain graph structures the fixed points of the message passing algorithm are actually the stationary points of the Bethe free energy [Yedidia et al., 2000]. This clarified the nature of message passing and led to more efficient algorithms. Further, this established a bond to a large body of physics literature. Furthermore, generalized belief propagation (GBP) algorithms [Yedidia et al., 2005] were developed. The GBP algorithms operate on regions of nodes and messages are passed between these regions. In contrast to GBP, Yuille [2002] proposes a concave-convex procedure to directly optimize the Bethe free energy. Experiments on spin class configurations show that this method is stable, converges rapidly, and leads to even better solutions than message passing and GBP in some cases.

Convergence of loopy message passing has been experimentally confirmed for many applications (the maybe most popular being "turbo codes" [McEliece et al., 1998, Kschischang and Frey, 1998]). There has also been a lot of theoretical work investigating convergence properties of loopy message passing [Yedidia et al., 2005, Weiss, 2000, Weiss and Freeman, 2001, Aji et al., 1998]. Recently, theoretical conditions guaranteeing convergence of loopy message passing to a unique fixed point have been proposed in Mooij and Kappen [2007]. Minka [2005] showed that many of the proposed approximate inference approaches, such as variational message passing, loopy message passing, expectation propagation amongst others can be viewed as instances of minimizing particular information divergences.

5.2.3 Sampling Methods

Sampling methods are computational tractable methods aiming at computing the quantities of interest by means of Monte Carlo procedures. One of the simplest of such methods is known as *importance sampling* and *sampling importance resampling* [MacKay, 2003] for estimating expectations of functions. There are severe limitations of importance sampling in high dimensional sample spaces. However, Markov Chain Monte Carlo methods scale well with the dimensionality. Special cases are Gibbs

sampling and the *Metropolis-Hastings* algorithm (see, e.g. [Gilks et al., 1996, Neal, 1993] for an introduction). One of the most prominent application of Monte Carlo methods (i.e. sequential importance sampling) are *particle filters* [Doucet, 1998, Arulampalam et al., 2002] for online, non-linear and non-Gaussian tracking, where the posterior distribution is represented by a finite set of particles (samples). The particle filters generalize the traditional Kalman filters which assume that the evolution of the state space is linear and that the probability densities are Gaussian. Classical particle filters solve the inference task of computing the posterior $\mathrm{P}(\mathbf{Q}|\mathbf{O}=\mathbf{o})$ for a limited family of graph structures such as Kalman filters. Recently, Koller et al. [1999], Sudderth et al. [2003], Ihler and McAllester [2009] extended particle methods to solve inference problems defined on general graphical model structures.

6 Applications

PGMs have a long tradition in modeling uncertainty in intelligent systems and are important in many application areas such as computer vision [Geman and Geman, 1984, Besag, 1986, Li, 2009, Willsky, 2002], speech processing [Rabiner, 1989, Bilmes and Bartels, 2005], time-series and sequential modeling [Barber and Cemgil, 2010], cognitive science [Chater et al., 2006], bioinformatics [Husmeier et al., 2005], probabilistic robotics [Thrun et al., 2006], signal processing, communications and error-correcting coding theory [McEliece et al., 1998, Kschischang et al., 2001, MacKay, 2003, Gallager, 1963], and in the area of artificial intelligence.

In this tutorial, we restrict ourselves to present some typical applications for each of the three PGM representations, namely for BNs, MNs and FGs. The first application is about dynamic BNs for time-series modeling of speech signals, followed by BNs for expert systems. Then, we present MRFs for image analysis, and FGs for decoding error-correcting codes.

6.1 Dynamic BNs for Speech Processing and Time-series Modeling

Speech recognition is the task of retrieving the sequence of words from a speech recording. The main components of an automatic speech recognition (ASR) system are

1. **Feature extraction.** A feature sequence $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ is determined by extracting T feature vectors from the audio signal. Commonly, mel-cepstral coefficients, delta, and double delta features [Rabiner, 1989] are derived from overlapping windows of ∼25ms length of the speech signal.

2. **Recognition.** Based on the extracted features, a sequence of N words $\mathbf{w} = (w_1, \ldots, w_N)$ is determined.

The most likely word sequence $\mathbf{w}^*$ given some sequence of feature vectors can be determined by the posterior probability $\mathrm{P}(\mathbf{w}|\mathbf{X})$ as

$$\mathbf{w}^* = \arg\max_{\mathbf{w}\in\mathcal{W}} \mathrm{P}(\mathbf{w}|\mathbf{X}) \tag{209}$$

$$= \arg\max_{\mathbf{w}\in\mathcal{W}} \frac{\mathrm{P}(\mathbf{X}|\mathbf{w})\,\mathrm{P}(\mathbf{w})}{\mathrm{P}(\mathbf{X})} \qquad = \arg\max_{\mathbf{w}\in\mathcal{W}} \mathrm{P}(\mathbf{X}|\mathbf{w})\,\mathrm{P}(\mathbf{w})\,, \tag{210}$$

where $\mathcal{W}$ denotes the set of all possible word sequences. The *acoustic model* probability $\mathrm{P}(\mathbf{X}|\mathbf{w})$ for an utterance $\mathbf{w}$ can be modeled as the concatenation of HMMs representing acoustic units such as words. In practice, it might even be beneficial to model subunits of words, e.g. bi- or triphones[4] or syllable units.

An HMM is a (hidden) Markov chain over RVs $Q_1, \ldots, Q_N$ which generates an observation sequence $X_1, \ldots, X_N$ such that the observation X_i stochastically depends only on state Q_i, i.e.

$$\mathrm{P}(X_i|Q_1, \ldots, Q_N, X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_N) = \mathrm{P}(X_i|Q_i)\,. \tag{211}$$

An HMM can be represented by the BN shown in Figure 27. In the case of Gaussian distributed state variables and observation probabilities $\mathrm{P}(X_i|Q_i)$, this model structure is known as *Kalman filter*.

[4] In linguistics, a biphone (triphone) is a sequence of two (three) phonemes.

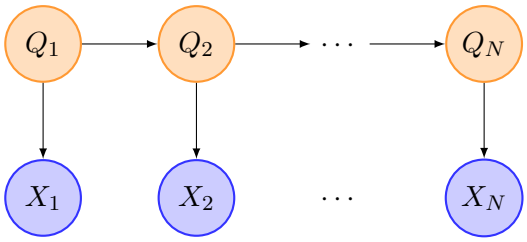


Figure 27: Hidden Markov Model.

For modeling the term $\mathrm{P}(\mathbf{w})$ in Equation (210) a natural language processing model is used. For instance, stochastic language models [Jurafsky and Martin, 2008] such as *n-grams* decompose the probability of a sequence of words as

$$\mathrm{P}(w_1, \ldots, w_N) = \prod_{i=1}^{N} \mathrm{P}(w_i | w_1, \ldots, w_{i-1}) \approx \prod_{i=1}^{N} \mathrm{P}\left(w_i | w_{i-(n-1)}, \ldots, w_{i-1}\right). \tag{212}$$

A bigram model (2-gram), which basically is a first order Markov chain, leads to

$$\mathrm{P}(\mathbf{w}) = \prod_{i=1}^{N} \mathrm{P}(w_i | w_{i-1}). \tag{213}$$

The most probable word sequence $\mathbf{w}^*$ in Equation (210) is determined using a decoding algorithm such as the *Viterbi algorithm* which is actually an instance of the max-product algorithm for HMMs. Hence, the Viterbi algorithm determines the most likely instantiation of $\mathbf{Q}$ given $\mathbf{O} = \mathbf{o}$

$$\mathbf{q}^* = \arg \max_{\mathbf{q} \in \mathbf{val}(\mathbf{Q})} \mathrm{P}(\mathbf{Q} | \mathbf{O} = \mathbf{o}). \tag{214}$$

The approach in Equation (210) can also be used in statistical machine translation [Brown et al., 1993], where the acoustic model $\mathrm{P}(\mathbf{X}|\mathbf{w})$ is replaced by a translation model $\mathrm{P}(\mathbf{f}|\mathbf{w})$ which models the relation of a pair of word sequences of different languages $\mathbf{f}$ and $\mathbf{w}$, where $\mathbf{f} = (f_1, \ldots, f_M)$ is a sequence with M words. The probability $\mathrm{P}(\mathbf{f}|\mathbf{w})$ can be interpreted as the probability that a translator produces sequence $\mathbf{f}$ when presented sequence $\mathbf{w}$.

HMMs [Rabiner, 1989] for acoustic modeling of speech signals have enjoyed remarkable success over the last forty years. However, recent developments showed that there are many sophisticated extensions to HMMs represented by dynamical BNs, cf. [Bilmes and Bartels, 2005] for an overview. Furthermore, in the seminal paper of Bahl et al. [1986], discriminative HMM parameter learning based on the maximum mutual information (MMI) criterion – closely related to optimizing the conditional log-likelihood objective – has been proposed, which attempts to maximize the posterior probability of the transcriptions given the speech utterances. This leads to significantly better recognition rates compared to conventional *generative* maximum likelihood learning. In speech processing, discriminative training of HMMs celebrated success over the years and in this context the extended Baum-Welch algorithm [Gopalakishnan et al., 1991, Woodland and Povey, 2002] has been introduced.

ASR for high-quality single-talker scenarios is performing reliably with good recognition performance. However, in harsh environments where the speech signal is distorted by interference with other acoustic sources, e.g. in the *cocktail party problem* [Cherry, 1953], ASR performs far from satisfactory. Recently, factorial-HMMs (FHMMs) including speaker interaction models and constraints on temporal dynamics have won the monaural speech separation and recognition challenge [Cooke et al., 2010]. Remarkably, these models slightly outperformed human listeners [Hershey et al., 2010] and are able to separate audio mixtures containing up to four speech signals. In the related task of multipitch tracking, Wohlmayr et al. [2011] have obtained promising results using a similar FHMM model. FHMMs [Ghaharamani and Jordan, 1997] enable to track the states of multiple Markov processes evolving in parallel over time, where the available observations are considered as a joint effect of all

single Markov processes. Combining two single speakers in this way using an interaction model, we obtain the FHMM shown in Figure 28. The hidden state RVs denoted by $X_k^{(t)}$ model the temporal dynamics of the pitch trajectory, where k indicates the Markov chain representing the k^{th} speaker and t is the time frame. Realizations of observed RVs of the speech mixture spectrum at time t are collected in a D-dimensional vector $\mathbf{Y}^{(t)} \in \mathbb{R}^D$ and $\mathbf{S}_k^{(t)}$ models the single-speaker spectrum conditioned on state $X_k^{(t)}$. At each time frame, the observation $\mathbf{Y}^{(t)}$ is considered to be produced jointly by the two single speech emissions $\mathbf{S}_1^{(t)}$ and $\mathbf{S}_2^{(t)}$. This mixing process is modelled by an interaction model as for example by the MIXMAX model described in Nadas et al. [1989].

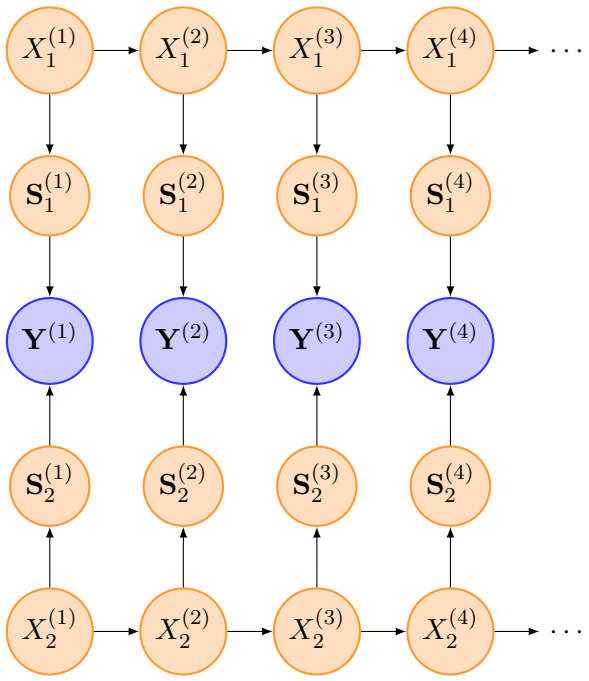


Figure 28: FHMM for speaker interaction. Each Markov chain models the pitch trajectory of a single speaker. At each time frame, the single speech emissions $\mathbf{S}_1^{(t)}$ and $\mathbf{S}_2^{(t)}$ jointly produce the observation $\mathbf{Y}^{(t)}$.

6.2 BNs for Expert Systems

Expert and decision support systems enjoy great popularity, especially in the medical domain. They consist of two parts: a knowledge base and an inference engine. The knowledge base contains specific knowledge of some domain and the inference engine is able to process the encoded knowledge and extract information. Many of these systems are *rule-based* using logical rules. BNs have been applied in order to deal with uncertainty in these systems [Cowell et al., 1999]. For instance, the Quick Medical Reference (QMR) database [Shwe et al., 1991] has been developed to support medical diagnosis which consists of 600 diseases and 4000 symptoms as depicted in Figure 29. The aim of inference is to determine the marginal probabilities of some diseases given a set of observed symptoms. In Jordan et al. [1999], variational inference has been introduced as exact inference is infeasible for most of the practically occurring inference queries.

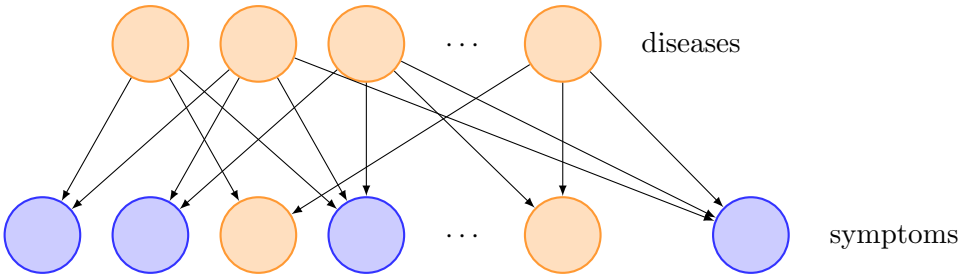


Figure 29: QMR database.

6.3 MRFs for Image Analysis

MRFs [Besag, 1986, Geman and Geman, 1984, Li, 2009] have been widely used in image processing tasks over the past decades. They are employed in all stages of image analysis, i.e. for low-level vision such as image denoising, segmentation, texture and optical flow modelling, and edge detection as well as for high-level tasks like face detection/recognition and pose estimation. Image processing problems can often be represented as the task of assigning labels to the image pixels. For example, in the case of image segmentation, each pixel belongs to one particular label representing a segment: in the MRF in Figure 30, the image pixels $\mathbf{Y}$ are observed and the underlying labels are represented as latent, i.e. unobserved, variables $\mathbf{X}$. The edges between the variables $\mathbf{X}$ model contextual dependencies, e.g. using the fact that neighboring pixels have similar features. Inferring the distribution $\mathrm{P}_{\mathcal{M}}(\mathbf{X}|\mathbf{Y})$ is intractable for large-scale MRFs using exact methods. Approximate inference methods, such as loopy belief propagation, energy optimization, and sampling methods (MCMC) amongst others, have been deployed.

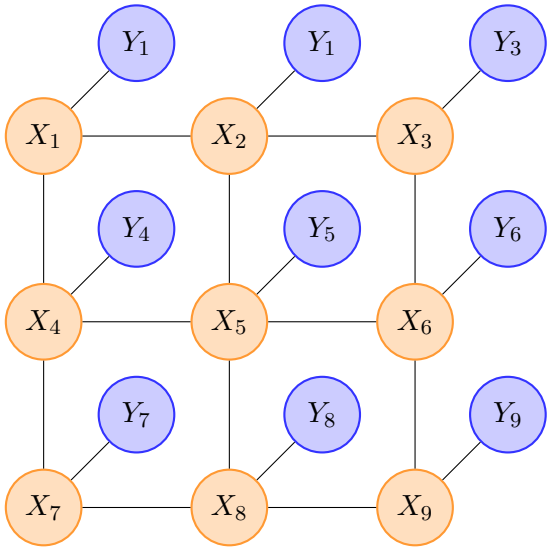


Figure 30: Markov random field for image analysis.

6.4 FGs for Decoding Error-correcting Codes

Transmission of information over noisy channels requires redundant information to be added to the transmitted sequence of symbols in order to facilitate the recovery of the original information from a corrupted received signal. Most error-correcting codes are either block codes or convolutional codes, including the low-density parity-check codes developed by Gallager [1963].

In a block code [MacKay, 2003] the source symbol sequence of length K is converted into a symbol sequence $\mathbf{x}$ of length N for transmission. Redundancy is added in the case of N being greater than K. Hence, a (N, K) block code is a set of 2^K codewords $\left\{\mathbf{x}^1, \ldots, \mathbf{x}^{2^K}\right\}$, where each codeword has a

length of N symbols of a finite alphabet $\mathcal{A}$, i.e. $\mathbf{x}^i \in \mathcal{A}^N$ [MacKay, 2003, Loeliger, 2004]. Here, we assume a binary code, i.e. $\mathcal{A} \in \{0, 1\}$.

In a *linear block code*, the added extra bits are a linear function of the original symbol. Those bits are known as *parity-check* bits and are redundant information. The transmitted codeword $\mathbf{x}$ is obtained from the source sequence $\mathbf{s}$ by calculating $\mathbf{x} = \mathbf{G}^T\mathbf{s}$ using modulo-2 arithmetic, where $\mathbf{G}$ is the *generator matrix* of the code. For the $(7, 4)$ Hamming code [MacKay, 2003] a generator matrix is

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}, \tag{215}$$

and the corresponding *parity-check matrix* $\mathbf{H}$ can be derived (details are given in MacKay [2003]) as

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}. \tag{216}$$

Each valid codeword $\mathbf{x}$ must satisfy $\mathbf{Hx} = \mathbf{0}$. This parity-check can be expressed by the indicator function

$$I(x_1, \ldots, x_7) = \mathbb{1}_{\{\mathbf{Hx}=\mathbf{0}\}} \tag{217}$$

$$= \delta(x_1 \oplus x_2 \oplus x_3 \oplus x_5)\delta(x_2 \oplus x_3 \oplus x_4 \oplus x_6)\delta(x_1 \oplus x_3 \oplus x_4 \oplus x_7), \tag{218}$$

where $\delta(a)$ denotes the Dirac delta function which is 1 in the case that $a = 1$ and 0 otherwise and $\oplus$ is the modulo-2 addition. That is, $I(x_1, \ldots, x_7) = 1$ if and only if $\mathbf{x}$ is a valid codeword. Each row in $\mathbf{H}$ corresponds to one $\delta(\cdot)$ term. The function $I(x_1, \ldots, x_7)$ can be represented as the FG shown in Figure 31. It consists of the following factors: $f_1(X_1, X_2, X_3, X_5) = \delta(x_1 \oplus x_2 \oplus x_3 \oplus x_5)$, $f_2(X_2, X_3, X_4, X_6) = \delta(x_2 \oplus x_3 \oplus x_4 \oplus x_6)$, and $f_3(X_1, X_3, X_4, X_7) = \delta(x_1 \oplus x_3 \oplus x_4 \oplus x_7)$.

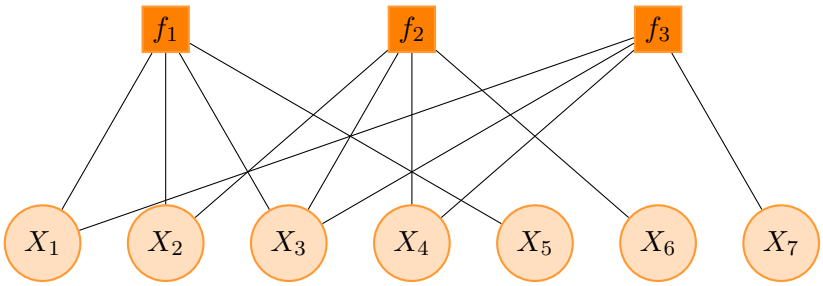


Figure 31: FG for the binary $(7, 4)$ Hamming code.

After encoding, the codewords are transmitted over a noisy channel. One of the simplest noisy channel models is the *binary symmetric channel* where each transmitted bit is flipped (corrupted) with probability ϵ. That is, if the input to the channel is denoted as X and the output of the channel is Y, then

$$\mathrm{P}(Y = 0|X = 1) = \epsilon, \tag{219}$$

$$\mathrm{P}(Y = 1|X = 1) = 1 - \epsilon, \tag{220}$$

$$\mathrm{P}(Y = 0|X = 0) = 1 - \epsilon, \text{ and} \tag{221}$$

$$\mathrm{P}(Y = 1|X = 0) = \epsilon. \tag{222}$$

This channel is memoryless and the channel model can be represented as $\mathrm{P}(\mathbf{Y}|\mathbf{X}) = \prod_{i=1}^{N} \mathrm{P}(Y_i|X_i) = \prod_{i=1}^{N} g_i(X_i, Y_i)$. Decoding the received symbol $\mathbf{Y}$ requires to determine the transmitted codeword

$\mathbf{X}$ based on the observed sequence $\mathbf{Y}$. This is possible using the posterior probability $\mathrm{P}(\mathbf{X}|\mathbf{Y})$ and exploiting that

$$\mathrm{P}(\mathbf{X}|\mathbf{Y}) = \frac{\mathrm{P}(\mathbf{Y}|\mathbf{X})\mathrm{P}(\mathbf{X})}{\mathrm{P}(\mathbf{Y})} \tag{223}$$

$$\propto \mathrm{P}(\mathbf{Y}|\mathbf{X})I(\mathbf{X}) = \prod_{i=1}^{N} \mathrm{P}(Y_i|X_i)I(\mathbf{X}), \tag{224}$$

where the probability $\mathrm{P}(\mathbf{X})$ is represented by the indicator function $I(X)$ of the code, i.e. all codewords are assumed to be equally likely. The code together with the noisy channel model can be represented by the FG shown in Figure 32. The probability $\mathrm{P}(\mathbf{X}|\mathbf{Y})$ is obtained by performing the sum-product algorithm (or the max-product algorithm). If the FG contains cycles, as in this example, the decoding problem is iterative and amounts to loopy message passing, also known as turbo decoding in the coding community [McEliece et al., 1998].

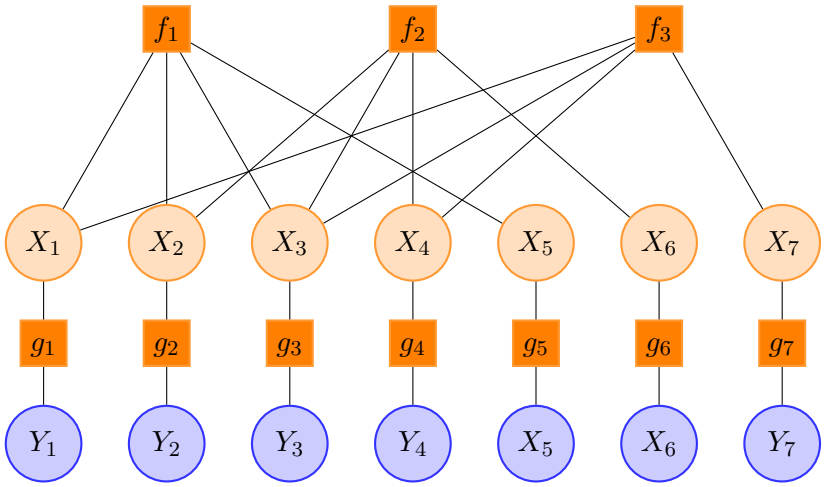


Figure 32: FG for the binary $(7, 4)$ Hamming code and the binary symmetric channel.

7 Implementation/code

In this section, we provide a list of recently developed code applicable to various tasks related to PGMs. With respect to the features of the listed implementations and comparisons to other software, the interested reader is referred to the given references and websites.

- Infer.NET: Infer.NET provides a framework for large-scale Bayesian inference in PGMs and is developed at Microsoft Research Cambridge (`research.microsoft.com/en-us/um/cambridge/projects/infernet`).
- WEKA: WEKA [Hall et al., 2009] is a data mining software applicable to various tasks in machine learning. Some of the implemented methods are directly related to PGMs such as the discriminative learning or structure learning heuristics of BNs.
- libDAI: LibDAI [Mooij, 2010] is an open source C++ library providing various exact and approximate inference methods for FGs with discrete variables.
- HTK: The HTK toolkit from Cambridge University (`htk.eng.cam.ac.uk`) is a state-of-the-art tool used for time-series modeling such as speech recognition. It implements an HMM using discrete and continuous observation probabilities, e.g. mixtures of Gaussians. Furthermore, the tool supports parameter tying, discriminative HMM training and many other methods important for speech recognition.

- GMTK: GMTK (`ssli.ee.washington.edu/~bilmes/gmtk`) implements dynamic BNs for sequential data processing. It is mainly designed for speech recognition.
- MMBN: We recently developed a maximum margin parameter learning algorithm for BN classifiers. Details about the algorithm are given in [Pernkopf et al., 2011b]. The maximum margin parameter training method can be downloaded at `www.spsc.tugraz.at/tools/MMBN`. Furthermore, some data sets and example code are provided.

8 Data Sets

In the following, we list a selection of common data sets and repositories in machine learning. This list includes only a small subset of publicly available data. Detailed information are supplied in the given references.

- **UCI repository.** [Frank and Asuncion, 2010] One of the largest collection of data sets is the UCI repository. It includes more than 200 data sets for classification, clustering, regression, and time-series modeling. The types of features are categorical, real, and mixed. Benchmark results for most of the data are given in accompanying references.
- **MNIST data.** [LeCun et al., 1998] The MNIST dataset of handwritten digits contains 60000 samples for training and 10000 for testing. The digits are centered in a 28×28 gray-level image. Benchmark classification results and the data are available at `http://yann.lecun.com/exdb/mnist/`.
- **USPS data.** This data set contains 11000 handwritten digit images collected from zip codes of mail envelopes (`www.cs.nyu.edu/~roweis/data.html`). Each digit is represented as a 16×16 gray-scale image. Other variants of this data set with different sample sizes are available. More details are given in the Appendix of Schölkopf and Smola [2001].
- **TIMIT data.** [Garofolo et al., 1993] This data set is very popular in the speech processing community. It contains read speech of eight major dialects of American English sampled at 16kHz. The corpus includes time-aligned orthographic, phonetic, and word transcriptions. This data has been used for various tasks, like time-series modeling, phone recognition and classification [Halberstadt and Glass, 1997, Pernkopf et al., 2009].
- **Caltech101 and Caltech256.** For image analysis two data sets have been provided by Caltech containing images of objects belonging to either 101 or 256 categories. The data, performance results, and relevant papers are provided at `http://www.vision.caltech.edu/Image_Datasets/Caltech101`.

9 Conclusions

PGMs turned out to be one of the best approaches for modeling uncertainty in many domains. They offer an unifying framework which allows to transfer methods among different domains. In this article, we summarized the main elements of PGMs. In particular, we reviewed three types of representations – Bayesian networks, Markov networks, and factor graphs, and we provided a gentle introduction to structure and parameter learning approaches of Bayesian networks. Furthermore, inference techniques were discussed with focus on exact methods. Approximate inference was covered briefly. The interested reader can immerse oneself following the references provided throughout the article.

Many relevant and interesting topics are not covered in detail within this review. One of these topics is approximate inference methods. In fact, tractable approximate inference has been one of the most challenging and active research fields over the past decade. Many of these advanced techniques are discussed in [Wainwright and Jordan, 2008, Koller and Friedman, 2009, Bishop, 2006, Minka, 2005, Yedidia et al., 2005]. Another focal point of current research interests are learning techniques for MNs. While we concentrated on structure and parameter learning of BNs under various perspectives, we completely omitted learning of MNs and learning with incomplete data. Parameter learning of

undirected models is essentially performed using iterative gradient-based methods. Each iteration requires to perform inference which makes parameter learning computationally expensive. Score-based structure learning suffers from similar considerations. One advantage of learning the structure of MNs is that there are no acyclicity constraints which renders structure learning in BNs hard.

References

S. Aji, G. Horn, and R. McEliece. On the convergence of iterative decoding on graphs with a single cycle. In *IEEE International Symposium on Information Theory*, pages 276–282, 1998.

S.M. Aji and R.J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–543, 2000.

S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line on-linear/non-gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

L.R. Bahl, P.F. Brown, P.V. de Souza, and R.L. Mercer. Maximum mutual information estimation of HMM parameters for speech recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 49–52, 1986.

D. Barber and A.T. Cemgil. Graphical models for time-series. *Signal Processing Magazine, IEEE*, 27 (6):18–28, 2010.

J.O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag, 1985.

Joseph Berkson. Limitations of the application of fourfold table analysis to hospital data. *Biometrics Bulletin*, 2(3):pp. 47–53, 1946. ISSN 00994987. URL `http://www.jstor.org/stable/3002000`.

J.E. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society, Series B*, 48:259–302, 1986.

J.A. Bilmes. Dynamic Bayesian Multinets. In *Proceedings of the 16th conf. on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 2000.

J.A. Bilmes and C. Bartels. Graphical model architectures for speech recognition. *Signal Processing Magazine, IEEE*, 22(5):89–100, 2005.

C. Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1995.

C.M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

P.F. Brown, S.A. Della Pietra, V.J. Della Pietra, and R.L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

W.L. Buntine. Theory refinement on Bayesian networks. In *International Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 52–60, 1991.

C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

N. Chater, J.B. Tenenbaum, and A. Yuille. Probabilistic models of cognition: Conceptual foundations. *Trends in Cognitive Sciences*, 10(7):287–291, 2006.

E. C. Cherry. Some experiments on the recognition of speech, with one and with two ears. *Journal of Acoustical Society of America*, 25:975–979, 1953.

David M. Chickering, Dan Geiger, and David Heckerman. Learning Bayesian networks is NP-hard. Technical report, Technical Report No. MSR-TR-94-17, Microsoft Research, Redmond, Washington, 1994.

D.M. Chickering. Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*, 2:445–498, 2002.

D.M. Chickering. Learning Bayesian networks is NP-Complete. In *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag, 1996.

D.M. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian networks: Search methods and experimental results. In *Proc. of Fifth Conference on Artificial Intelligence and Statistics*, pages 112–128, 1995.

C.K. Chow and C.N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transaction on Information Theory*, 14:462–467, 1968.

M. Cooke, J.R. Hershey, and S. J. Rennie. Monaural speech separation and recognition challenge. *Computer, Speech & Language*, 24:1–15, 2010.

G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

T. Cover and J. Thomas. *Elements of information theory*. John Wiley & Sons, 1991.

R. Cowell. *Introduction to inference for Bayesian networks*. in Learning in Graphical Models, (M.I. Jordan, editor), MIT Press, 1999.

R.G. Cowell, A.P. Dawid, S.L. Lauritzen, and D.J. Spiegelhalter. *Probabilistic networks and expert systems*. Springer, 1999.

P. Dawid. Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2:25–36, 1992.

C.P. de Campos, Z. Zeng, and Q. Ji. Structure learning of Bayesian networks using constraints. In *International Conference on Machine Learning (ICML)*, pages 113–120, 2009.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistic Society*, 30(B):1–38, 1977.

A. Doucet. On sequential Monte Carlo sampling methods for Bayesian filtering. Technical Report CUED/F-INFENG/TR. 310, Cambridge University, Department of Engineering, 1998.

A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

J.H. Friedman. On bias, variance, 0/1-loss, and the curse of dimensionality. *Data Mining and Knowledge Discovery*, 1:55–77, 1997.

N. Friedmann and D. Koller. Being Bayesian about network structure. In *Proc. Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 201–210, 2000.

R.G. Gallager. *Low-Density Parity-Check Codes*. MIT Press, 1963.

J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren. Darpa timit acoustic phonetic continuous speech corpus cdrom, 1993.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

Z. Ghaharamani and M.I. Jordan. Factorial hidden Markov models. *Machine Learning*, 29:245–275, 1997.

W. Gilks, S. Richardson, and D. Spiegelhalter. *Markov Chain Monte Carlo in practice*. Chapman and Hall, 1996.

Fred Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1): 156–166, 1977.

P.S. Gopalakishnan, D. Kanevsky, A. Nadas, and D. Nahamoo. An inequality for rational functions with applicaitons to some statistical estimation problems. *IEEE Transactions on Information Theory*, 37(1):107–113, 1991.

R. Greiner and W. Zhou. Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. In *Conference of the AAAI*, pages 167–173, 2002.

D. Grossman and P. Domingos. Learning Bayesian network classifiers by maximizing conditional likelihood. In *International Conference of Machine Lerning (ICML)*, pages 361–368, 2004.

Y. Guo, D. Wilkinson, and D. Schuurmans. Maximum margin Bayesian networks. In *International Conference on Uncertainty in Artificial Intelligence (UAI)*, 2005.

A.K. Halberstadt and J. Glass. Heterogeneous measurements for phonetic classification. In *Proceedings of EUROSPEECH*, pages 401–404, 1997.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11:10–18, 2009.

David Heckerman, Dan Geiger, and David M. Chickering. Learning Bayesian networks: the combination of knowledge and statistical data. *Machine Learning (Kluwer Academic Publishers)*, 20: 197–243, 1995.

J.R. Hershey, S. J. Rennie, P.A. Olsen, and T.T. Kristjansson. Super-human multi-talker speech recognition: A graphical modeling approach. *Computer, Speech & Language*, 24:45–66, 2010.

Cecil Huang and Adnan Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15:225–263, 1996.

D. Husmeier, R. Dybowski, and S. Roberts. *Probabilistic Modelling in Bioinformatics and Medical Informatics*. Springer-Verlag, 2005.

A. Ihler and D. McAllester. Particle belief propagation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 256–263, 2009.

T. Jaakkola. *Tutorial on variational approximation methods*. In Advanced Mean Field Methods: Theory and Practice. MIT Press, 2000.

T. Jaakkola, D. Sontag, A. Globerson, and M. Meila. Learning Bayesian network structure using LP relaxations. In *Intern. Conf. on Artificial Intelligence and Statistics (AISTATS)*, pages 358–365, 2010.

F.V. Jensen. *An introduction to Bayesian networks*. UCL Press Limited, 1996.

M.I. Jordan. *Learning in graphical models*. MIT Press, 1999.

M.I. Jordan, Z. Ghahramani, T.S. Jaakkola, and L.K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

D. Jurafsky and J.H Martin. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Prentice Hall, 2008.

E.J. Keogh and M.J. Pazzani. Learning augmented Bayesian classifiers: A comparison of distribution-based and classification-based approaches. In *International Workshop on Artificial Intelligence and Statistics*, pages 225–230, 1999.

M. Koivisto and K. Sood. Exact bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, 5:549–573, 2004.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

D. Koller, U. Lerner, and D. Angelov. A general algorithm for approximate inference and its application to hybrid Bayes nets. In *International Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 324–333, 1999.

J.B. Kruskal. On the shortest spanning subtree and the traveling salesman problem. In *Proceedings of the American Mathematical Society*, volume 7, pages 48–50, 1956.

F.R. Kschischang and B.J. Frey. Iterative decoding of compound codes by probability propagation in graphical models. *IEEE Journal on Selected Areas in Communications*, 16:219–230, 1998.

F.R. Kschischang, B.J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

Wai Lam and Fahiem Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10(3):269–293, 1994.

S. Lauritzen and D. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *J. Royal Statistical Society B*, 50:157–224, 1988.

S.L. Lauritzen. *Graphical Models*. Oxford Science Publications, 1996.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

S.Z. Li. *Markov Random Field Modeling in Image Analysis*. Springer-Verlag, 2009.

H.-A. Loeliger. An introduction to factor graphs. *IEEE Signal Processing Magazine*, 21(1):28–41, 2004.

D.J.C. MacKay. *Information theory, inference, and learning algorithms*. Cambridge University Press, 2003.

R.J. McEliece, D.J.C. MacKay, and Cheng J.F. Turbo decoding as an instance of Pearl's 'belief propagation' algorithm. *IEEE Journal on Selected Areas in Communication*, 16(2):140–152, 1998.

T. Minka. Divergence measures and message passing. Technical Report MSR-TR-2005-173, Microsoft Research Ltd., Cambridge, UK, 2005.

J.M. Mooij. libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *Journal of Machine Learning Research*, 11:2169–2173, 2010.

J.M. Mooij and H.J. Kappen. Sufficient conditions for convergence of the sum-product algorithm. *IEEE Transactions on Information Theory*, 52(2):4422–4437, 2007.

A. Nadas, D. Nahamoo, and M.A. Picheny. Speech recognition using noise-adaptive prototypes. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(10):1495–1503, 1989.

R.M. Neal. Probabilistic inference using Markov Chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto, Department of Computer Science, 1993.

A. Papoulis and S. U. Pillai. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 2002.

J. Pearl. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann, 1988.

F. Pernkopf and J. Bilmes. Efficient heuristics for discriminative structure learning of Bayesian network classifiers. *Journal of Machine Learning Research*, 11:2323–2360, 2010.

F. Pernkopf, T. Van Pham, and J. Bilmes. Broad phonetic classification using discriminative Bayesian networks. *Speech Communication*, 51(2):151–166, 2009.

F. Pernkopf, M. Wohlmayr, and M. Mücke. Maximum margin structure learning of bayesian network classifiers. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 2076–2079, 2011a.

F. Pernkopf, M. Wohlmayr, and S. Tschiatschek. Maximum margin Bayesian network classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence, accepted*, 2011b.

L.R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

T. Roos, H. Wettig, P. Grünwald, P. Myllymäki, and H. Tirri. On discriminative Bayesian network classifiers and logistic regression. *Machine Learning*, 59:267–296, 2005.

B. Schölkopf and A.J. Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT Press, 2001.

M.A. Shwe, B. Middleton, D.E. Heckerman, M. Henrion, E.J. Horvitz, H.P. Lehmann, and G.E. Cooper. Probabilistic diagnosis using a reformulation of the internist-1/qmr knowledge base. ii. evaluation of diagnostic performance. *Methods of Information in Medicine*, 30:256–267, 1991.

T. Silander and P. Myllymäki. A simple approach for finding the globally optimal Bayesian network structure. In *Proc. 22nd Conference on Uncertainty in Artificial Intelligence*, 2006.

Ajit P. Singh and Andrew W. Moore. Finding optimal Bayesian networks by dynamic programming. Technical report, Carnegie Mellon University, 2005.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. The MIT Press, 2001.

E.B. Sudderth, A.T. Ihler, W.T. Freeman, and A.S. Willsky. Nonparametric belief propagation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 605–612, 2003.

J. Suzuki. Learning Bayesian belief networks based on the minimum description length principle: An efficient algorithm using the B&B technique. In *Int. Conference on Machine Learning*, pages 462–470, 1996.

Joe Suzuki. A construction of Bayesian networks from databases based on an MDL principle. In *Proc. 9th Conference on Uncertainty in Artificial Intelligence*, pages 266–273, 1993.

M. Teyssier and D. Koller. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In *21th Conference on Uncertainty in AI (UAI)*, pages 584 – 590, 2005.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, 2006.

V. Vapnik. *Statistical learning theory*. Wiley & Sons, 1998.

T. Verma and J. Pearl. An algorithm for deciding if a set of observed independencies has a causal explanation. In *Proc. 4th Conference on Uncertainty in Artificial Intelligence*, 1992.

M.J. Wainwright and M.I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1:1–305, 2008.

Y. Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Computation*, 12:1–41, 2000.

Y. Weiss and W.T. Freeman. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Computation*, 13(10):2173–2200, 2001.

H. Wettig, P. Grünwald, T. Roos, P. Myllymäki, and H. Tirri. When discriminative learning of Bayesian network parameters is easy. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 491–496, 2003.

A.S. Willsky. Multiresolution markov models for signal and image processing. *Proceedings of the IEEE*, 90(8):1396–1458, 2002.

M. Wohlmayr, M. Stark, and F. Pernkopf. A probabilistic interaction model for multipitch tracking with factorial hidden markov model. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):799–810, 2011.

P.C. Woodland and D. Povey. Large scale discriminative training of hidden Markov models for speech recognition. *Computer Speech and Language*, 16:25–47, 2002.

J. Yedidia, W.T. Freeman, and Y. Weiss. Generalized belief propagation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 689–695, 2000.

J. Yedidia, W.T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.

J.S. Yedidia, W.T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. Technical Report TR-2001-22, Mitsubishi Electric Research Laboratories, 2002.

A.L. Yuille. CCCP algorithms to minimize the Bethe and Kikuchi free energies: Convergent alternatives to belief propagation. *Neural Computation*, 14(7):1691–1722, 2002.

Franz Pernkopf: Franz Pernkopf received his MSc (Dipl. Ing.) degree in Electrical Engineering at Graz University of Technology, Austria, in summer 1999. He earned a PhD degree from the University of Leoben, Austria, in 2002. In 2002 he was awarded the Erwin Schrödinger Fellowship. He was a Research Associate in the Department of Electrical Engineering at the University of Washington, Seattle, from 2004 to 2006. Currently, he is Associate Professor at the Laboratory of Signal Processing and Speech Communication, Graz University of Technology, Austria. His research interests include machine learning, discriminative learning, graphical models, feature selection, finite mixture models, and image- and speech processing applications.

Robert Peharz: Robert Peharz received his MSc degree in Telematics at Graz University of Technology (TUG) in 2010. He currently pursues his PhD studies at the SPSC Lab, TUG. His research interests include probabilistic graphical models, sparse coding, nonnegative matrix factorization, and machine learning in general, with applications to signal processing, audio engineering and computer vision.

Sebastian Tschiatschek: Sebastian Tschiatschek received the BSc degree and MSc degree in Electrical Engineering at Graz University of Technology (TUG) in 2007 and 2010, respectively. He conducted his Master thesis during a one-year stay at ETH Zürich, Switzerland. Currently, he is with the Signal Processing and Speech Communication Laboratory at TUG where he is pursuing the PhD degree. His research interests include Bayesian networks, information theory in conjunction with graphical models and statistical pattern recognition.