Notes on Probabilistic Graphical Models

Paulo Eduardo Rauber

1 Probability Theory

Probability theory is a well developed mathematical framework for reasoning under uncertainty. This section presents the fundamentals of the theory.

A sample space Ω is the set of possible outcomes of an experiment. Elements of Ω are also called sample outcomes. Events are subsets of Ω. An event is said to occur if the outcome of the experiment belongs to the event.

A set of events $\{A_1, ...\}$ is said to be disjoint if $A_i \cap A_j = \emptyset$ for every $i \neq j$. A partition of Ω is a disjoint set of events such that $\bigcup_i A_i = \Omega$.

If P is a function from the set of all subsets of Ω to $\mathbb{R}$ and follows the properties:

$$
\begin{aligned}
&P(A) \geq 0 \text{ for every event } A,\\
&P(\Omega) = 1,\\
&\text{If } \{A_1, ...\} \text{ is a disjoint set of events, } P(\bigcup_i A_i) = \sum_i P(A_i),
\end{aligned}
$$

then P is a probability distribution over Ω[1].

For any events A and B, the following properties of a probability distribution P over Ω can be derived from the three properties above:

$$
\begin{aligned}
&P(\emptyset) = 0,\\
&A \subseteq B \implies P(A) \leq P(B),\\
&0 \leq P(A) \leq 1,\\
&P(A^c) = 1 - P(A),\\
&P(A \cup B) = P(A) + P(B) - P(A \cap B).
\end{aligned}
$$

Intuitively, $P(A)$ represents the degree of belief that event A will contain the sample outcome of an experiment. For instance, if the sample space is finite and there is no reason to believe that $P(\{u\}) > P(\{w\})$ for any $u, w \in \Omega$, then, for any event A, $P(A) = \frac{|A|}{|\Omega|}$, and P is said to be a uniform distribution.

An example may be useful to illustrate the application of these definitions. Suppose we are interested in computing the probability of obtaining tails exactly once after tossing two unbiased coins. The sample space can be represented as $\Omega = \{(H, H), (H, T), (T, H), (T, T)\}$. We can let $P(\{w\}) = \frac{1}{4}$ for every $w \in \Omega$, since there is no reason to believe that a sample outcome is more likely than any other. The event of obtaining tails exactly once can be represented as $A = \{(T, H), (H, T)\}$. Therefore, $P(A) = P(\{(T, H)\}) + P(\{(H, T)\}) = \frac{1}{2}$, since these two events are disjoint.

Let $P(B) > 0$ for an event B. The conditional probability of an event A given B is defined as:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

Intuitively, $P(A|B)$ can be interpreted as the degree of belief that A will occur given that B is already known to have occurred. The formula for conditional probability also gives the equality $P(A \cap B) = P(B)P(A|B)$.

Let A and B be events on Ω and $P'(A) = P(A|B)$. Then P' is a probability distribution over Ω, as it follows the three properties mentioned in the beginning of this section. Therefore:

[1] The function P may also be defined over a set of events that contains Ω and is closed under union and complementation, but the distinction is not useful in this text.

$P(A|B) \geq 0$ for any event A,
$P(\Omega|B) = 1$,
If $\{A_1, ...\}$ is a disjoint set of events, $P(\bigcup_i A_i|B) = \sum_i P(A_i|B)$,
$P(\emptyset|B) = 0$,
$A \subseteq C \implies P(A|B) \leq P(C|B)$,
$0 \leq P(A|B) \leq 1$,
$P(A^c|B) = 1 - P(A|B)$,
$P(A \cup C|B) = P(A|B) + P(C|B) - P(A \cap C|B)$,
$P(A \cap C|B) = P(C|B)P(A|B \cap C)$.

Let $A = \{A_1, ...\}$ be a partition of Ω, then:

$$P((\bigcup_i A_i) \cap B) = P(\Omega \cap B) = P(B).$$

Let $\{A_1, ...\}$ be a partition of Ω. Then, if $P(B) \neq 0$, the definition of conditional probability can be restated as:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_j P(B|A_j)P(A_j)}.$$

The equation above is called Bayes' Theorem.

In the special case when $P(A \cap B) = P(A)P(B)$, A and B are said to be independent events.

Introduced the fundamentals, we now present random variables, which are essential to model complicated events. It is often possible to omit the sample space entirely when modeling problems using random variables.

A random variable X is a function $X : \Omega \to \mathbb{R}$. The inverse X^{-1} of a random variable is defined as $X^{-1}(R) = \{w \in \Omega | X(w) \in R\}$, where $R \subseteq \mathbb{R}$.

Let P be a probability distribution over Ω. Then, by definition:

$$P(X = x) = P(X^{-1}(\{x\}))$$
$$P(a < X < b) = P(X^{-1}((a, b))).$$

In words, $P(X = x)$ represents the probability of an event that contains all sample outcomes that are mapped by X to x, while $P(a < X < b)$ represents the probability of an event that contains all sample outcomes that are mapped by X to a real number in the open interval (a, b). If the event $X^{-1}(\{x\})$ occurs, the random variable X is said to be in state x or assigned to x.

Consider the experiment mentioned earlier of tossing two unbiased coins. We can let $X(w)$ be the number of tails in a sample outcome $w \in \Omega$. Therefore, $P(X = 0) = \frac{1}{4}$, $P(X = 1) = \frac{1}{2}$ and $P(X = 2) = \frac{1}{4}$.

When working with random variables X and Y, we write $P(X = x \cap Y = y)$ as $P(X = x, Y = y)$ for convenience.

We denote the image of a random variable X by $\text{Val}(X)$, i.e., the set of values assigned to some sample outcome. If $\text{Val}(X)$ is countable, then X is said to be a discrete random variable. If Y is also a random variable, $\text{Val}(X, Y)$ denotes the set of valid assignments to X and Y. This notation also extends to sets of random variables. Let $\mathcal{X}_1, \ldots \mathcal{X}_n$ be sets random variables. Then $\text{Val}(\mathcal{X}_1, \ldots \mathcal{X}_n)$ denotes the set of valid assignments for all variables in every $\mathcal{X}_i$.

If P is a probability distribution over Ω and X is a discrete random variable over Ω, we define the probability mass function f as $f(x) = P(X = x)$ for every $x \in \mathbb{R}$.

If X is a continuous random variable and f is a function such that $f(x) \geq 0$ for all x, $\int_{-\infty}^{\infty} f(x)\ dx = 1$, $P(a < X < b) = \int_a^b f(x)\ dx$ for every $a \leq b$, then f is called a probability density function. It is important to note that $f(x)$ does not represent $P(X = x)$ in the continuous case.

The Gaussian distribution $\mathcal{N}(\mu; \sigma)$ has the following probability density function p:

$$p(x) = \frac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}.$$

Some properties of a probability distribution P are restated in terms of random variables bellow.

For a discrete random variable distributed according to a probability distribution P:

$$\sum_{x \in \text{Val}(X)} P(X = x) = 1.$$

For a continuous random variable distributed according to a probability density function f:

$$\int_{-\infty}^{\infty} f(x)\ dx = 1,$$

The notation $X \sim P$ denotes that a variable X is distributed according to P.

In the discrete case, a function that maps every x and y to $P(X = x, Y = y)$ is called a joint distribution of X and Y. A function that maps every x and y to $P(X = x|Y = y)$ is called a conditional distribution of X given Y. Analogous definitions apply to continuous random variables.

The probability of the union of events can be restated as:

$$P((X = x) \cup (Y = y)) = P(X = x) + P(Y = y) - P(X = x, Y = x)$$

For discrete random variables, Bayes' theorem can be restated as follows:

$$P(X = x|Y = y) = \frac{P(Y = y|X = x)P(X = x)}{\sum_{x' \in \text{Val}(X)} P(Y = y|X = x')P(X = x')}.$$

Bayes' theorem for continuous random variables is analogous.

Let $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ be sets of random variables over Ω. Then $\mathcal{X} \perp\!\!\!\perp \mathcal{Y}|\mathcal{Z}$ if and only if, for every variable $X \in \mathcal{X}$, every $Y \in \mathcal{Y}$ and every $Z \in \mathcal{Z}$, $P(X = x, Y = y|Z = z) = P(X = x|Z = z)P(Y = y|Z = z)$ for all x, y and z. Intuitively, this means that the knowledge of the state of variables in $\mathcal{Y}$ does not give additional information about the state of variables in $\mathcal{X}$ when the state of the variables in $\mathcal{Z}$ is already known. Note that $\mathcal{Z}$ may be empty.

When ambiguity is impossible, it is common to omit the name of a random variable when denoting probabilities. For instance, $P(X = x, Y = y) = P(X = x|Y = y)P(Y = y)$ can be written as $P(x, y) = P(x|y)P(y)$.

Let X be a discrete random variable. The expectation of X under the probability distribution P, denoted $\mathbb{E}_P[X]$, is defined as:

$$\mathbb{E}_P[X] = \sum_{x \in \text{Val}(X)} xP(X = x).$$

The expectation of random variables X and Y (under any probability distribution) has the following properties:

$$\mathbb{E}[aX + b] = a\mathbb{E}[X] + b$$
$$\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y].$$

If X and Y are independent,

$$\mathbb{E}[X \cdot Y] = \mathbb{E}[X]\mathbb{E}[Y].$$

Let $\mathcal{X} = \{X_1, \ldots, X_n\}$ be a set of random variables. A factor ϕ is a function from $\text{Val}(\mathcal{X})$ to $\mathbb{R}$. In this case, the set $\mathcal{X}$ is called the scope of the factor. By convention, it is common to denote such factor by $\phi(\mathcal{X})$ or $\phi(X_1, \ldots, X_n)$, which is *not* the value of ϕ when applied to an element of its domain. Let $\mathbf{x} \in \text{Val}(\mathcal{X})$ such that $X_i = x_i$. The value of the factor ϕ for assignment $\mathbf{x}$ is denoted by $\phi(X_1 = x_1, \ldots, X_n = x_n)$. When ambiguity is impossible, the random variables may be omitted. Therefore, despite the fact that ϕ is a function, the order of the parameters is not relevant by convention. Joint probability distributions and conditional probability distributions are constrained factors. When all variables involved are discrete, it is useful to imagine a factor as a table.

Let $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ be three disjoint sets of variables, $\phi_1(\mathcal{X}, \mathcal{Y})$ (short for $\phi_1(\mathcal{X} \cup \mathcal{Y})$) and $\phi_2(\mathcal{Y}, \mathcal{Z})$ two factors. The product $\psi(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$ of factors ϕ_1 and ϕ_2 is defined as a factor $\psi(\mathcal{X}, \mathcal{Y}, \mathcal{Z}) = \phi_1(\mathcal{X}, \mathcal{Y})\phi_2(\mathcal{Y}, \mathcal{Z})$

Let $\psi(\mathcal{Y})$ be a factor and $\mathbf{u}$ and assignment for $\mathcal{U} \subseteq \mathcal{Y}$. The reduction of a factor ψ to the context $\mathbf{u}$, denoted $\psi[\mathcal{U} = \mathbf{u}]$ or $\psi[\mathbf{u}]$ is a factor over the scope $\mathcal{Y}' = \mathcal{Y} - \mathcal{U}$, such that $\psi[\mathbf{u}](\mathbf{y}') = \psi(\mathbf{y}', \mathbf{u})$. When $\mathcal{U} \not\subset \mathcal{Y}$, $\psi[\mathbf{u}] = \psi[\mathbf{u}']$, where $\mathcal{U}' = \mathcal{U} \cap \mathcal{Y}$, and $\mathbf{u}' = \mathbf{u}\langle\mathcal{U}'\rangle$, where $\mathbf{u}\langle\mathcal{U}'\rangle$ denotes the assignment in $\mathbf{u}$ to the variables that are in $\mathcal{U}'$.

Let $\mathcal{X}$ be a set of random variables and $Y \notin \mathcal{X}$ a variable. Let $\phi(\mathcal{X}, Y)$ be a factor. The factor marginalization $\psi = \sum_Y \phi$, is defined as $\psi(\mathcal{X}) = \sum_y \phi[Y = y](\mathcal{X})$.

Let $\mathcal{X}$ be a set of random variables and $Y \notin \mathcal{X}$ a variable. Let $\phi(\mathcal{X}, Y)$ be a factor. The factor maximization $\psi = \max_Y \phi$ is defined as $\psi(\mathcal{X}) = \max_y \phi[Y = y](\mathcal{X})$.

The definitions of factor product, reduction and marginalization lead to properties on operations over joint or conditional probability distributions that are analogous to properties on operations over particular assignments of random variables. For instance, if P is a probability distribution, the fact that $P(X = x, Y = y) = P(X = x|Y = y)P(Y = y)$ for all x and y relates to the fact that $P(X, Y) = P(X|Y)P(Y)$ for factors $P(X)$ and $P(X|Y)$ and $P(X, Y)$.

Let P be a joint probability distribution and ϕ_1, ϕ_2 two factors. If $P(X, Y, Z) \propto \phi_1(X, Z)\phi_2(Y, Z)$, then $X \perp\!\!\!\perp Y|Z$.

Probability theory is a very powerful framework. Propositional logic can be seen as a special case subsumed by probability theory. Suppose, for instance, that $A \to B$ is a true logical proposition. It follows that $\neg B \to \neg A$. In probability theory, this could be represented by random variables A' and B', with $\text{Val}(A') = \text{Val}(B') = \{0, 1\}$. The proposition $A \to B$ could be represented as $P(B' = 1|A' = 1) = 1$. From this, it follows that $P(B' = 0|A' = 1) = 0$. Therefore, $P(A' = 1|B' = 0) = \frac{P(B'=0|A'=1)P(A'=1)}{P(B'=0)} = 0$ and $P(A' = 0|B' = 0) = 1$, which represents $\neg B \to \neg A$, as we wanted to show. Note that it was not necessary to explicit the sample space to solve this problem, which is often the case.

Even the scientific method can be framed as probabilistic reasoning. If θ is a random variable representing a hypothesis and D is a random variable representing the observed data:

$$P(\theta|D) = \frac{P(\theta)P(D|\theta)}{P(D)}.$$

Therefore,

$$P(\theta|D) \propto P(\theta)P(D|\theta).$$

In words, the probability of a hypothesis given data is proportional to the prior probability of the hypothesis, which may be proportional to its simplicity, times the probability of the data occurring given the hypothesis, which may be proportional to how well the data fits the hypothesis.

Probabilistic reasoning consists in the process of identifying all relevant random variables $X_1, ..., X_n$ necessary to model a problem and discovering the joint distribution $P(X_1, ..., X_n)$ for all states of the variables. This joint distribution can be used to perform inference by introducing evidence in the form of knowledge about the state of some random variables and computing conditional probabilities. Learning to model real problems in this framework is often a challenging task.

Suppose that each random variable $X_1, ...X_n$ that models an experiment has 2 possible states. There would be 2^n elements in a table that represented the joint distribution for all variables. The practical problem is often filling, and not storing, this table with data. Probabilistic graphical models are powerful tools to represent joint probability distributions.

2 Graph theory

Graphs are mathematical objects useful for representing problems involving connections and dependencies.

A graph is a pair $G = (V, E)$ where $E \subseteq [V]^2$. The elements of V are called vertices or nodes and the elements of E are called edges. Graphs are commonly represented by drawing a dot for each vertex and a line between each pair of vertices that form an edge. An edge $e = \{u, v\}$ is said to end in u and v. In this case, u and v are said to be incident with e. When no confusion can arise, an edge $\{u, v\}$ can also be denoted as uv. A vertex v is neighbor to vertex u if $\{v, u\} \in E$. A vertex v with exactly n neighbors is said to have degree n. Given a graph $G = (A, B)$, we also denote the set of vertices A as $V(G)$ and the set of edges B as $E(G)$. A graph with no vertices is called empty.

A *directed* graph is a pair $G = (V, E)$ of disjoint sets together with two maps: $init : E \to V$ and $ter : E \to V$. An edge e in a directed graph is said to begin at $init(e) = u$ and end at $ter(e) = v$ and can be represented as the pair $e = (u, v)$. In this case, it is common to say that $e \in E$. In this text, when the correspondence is obvious, many definitions made for graphs also apply to directed graphs. The underlying (undirected) graph of a directed graph G is the graph obtained by replacing all edges of G with undirected edges.

Let G and G' be graphs. By definition, $G \cup G' = (V \cup V', E \cup E')$ and $G \cap G' = (V \cap V', E \cap E')$. If $V' \subseteq V$ and $E' \subseteq E$, then G' is a subgraph of G, written as $G' \subseteq G$, and G is a supergraph of G'. If $G' \subseteq G$ and G' contains all the edges $xy \in E(G)$ such that $x, y \in V(G')$, then G' is a subgraph of G induced by $V(G')$.

A clique C in a graph G is a subset of vertices of $V(G)$ such that there is an edge between every pair of vertices in C. A clique is maximal if it is not a proper subset of any other clique in G.

A graph $G = (V, E)$ with a function $f : E \to \mathbb{R}$ is said to be a weighted graph. A weighted graph can represent, for instance, cities (vertices) connected by roads (edges) with a certain length (the value of f for a given road).

A path is a non-empty graph $P = (V, E)$ of the form $V = \{x_0, \ldots, x_k\}$, $E = \{x_0x_1, \ldots, x_{k-1}x_k\}$, where the x_i are all distinct. The vertices x_0 and x_k are said to be linked by P. The length of a path is its number of edges. It is often useful to represent a path simply as a sequence of vertices $P = x_0 \ldots x_k$. If $P = x_0 \ldots x_k$ is a path, the graph $C = (V(P), E(P) \cup \{\{x_k, x_0\}\})$ is a cycle. If $P \subseteq G$, then P is a path in G. An undirected path in a directed graph G is a path in the underlying graph of G.

A non-empty graph G is called connected if any two of its vertices are linked by some path in G. A maximal connected subgraph of G is called a (connected) component of G. If there is a single path connecting every pair of vertices in $V(G)$, then G is singly-connected.

If there is a path linking x and y, then x is an ancestor of y and y is a descendant of x.

A graph that contains no cycles is a forest. A connected forest is a tree. In this way, a forest is a graph whose components are trees. Vertices in a tree with degree 1 are called leaves.

The following definitions apply to a directed acyclic graph $G = (V, E)$. If x and y are vertices and $xy \in E$, then y is a parent of x and x is a child of y. The set of parents of x in G is denoted by Pa_x^G. The Markov blanket of x is the set composed of its children, parents and other parents of its children.

Let G be a connected graph. A spanning tree of G is a tree contained in G that contains all vertices in G. A spanning tree of a weighted graph is said to have maximum weight if the sum of weights attributed to each of its edges is not lower than that of any other spanning tree.

A sequence of vertices $x_0 \ldots x_k$ from a directed acyclic graph G is said to be in ancestral order when there is no path in G from x_i to x_j when $i > j$.

A walk of length k in a graph G is a non-empty sequence $v_0e_0v_1 \ldots e_{k-1}v_k$ such that $v_i \in V(G)$, $e_i = v_iv_{i+1}$ and $e_i \in E$. If $v_0 = v_k$ the walk is closed. If the vertices in a walk are all distinct, the walk corresponds to a path in G.

Let $G = (\{v_0, \ldots, v_n\}, E)$ be a graph. Let $A = (a_{i,j})_{n \times n}$ be a matrix such that $a_{i,j} = 1$ if and only if $v_iv_j \in E$ and 0 otherwise. The matrix A is called the adjacency matrix of G. If the graph is undirected, the adjacency matrix is symmetric. Interestingly, A^k represents the number of walks of length k between any pair of vertices in G.

Let $C_1, \ldots, C_k$ be distinct cliques of $G = (\{v_0, \ldots, v_n\}, E)$. Then a clique matrix $A = (a_{i,j})_{n \times k}$ is a matrix such that $a_{i,j} = 1$ if and only if $v_i \in C_j$ and 0 otherwise. If each C_l corresponds to an edge in E, then A is called an incidence matrix.

3 Bayesian Networks

A Bayesian network is a pair $B = (G, P)$, where $G = (V, E)$ is a directed acyclic graph and P is a joint probability distribution (a *factor*, as defined in Section 1). Furthermore, $V = \{X_1, \ldots, X_n\}$ is a set of random variables and P can be written as the following product of other factors:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n-1} P(X_i | \mathrm{Pa}_{X_i}^G),$$

where $\mathrm{Pa}_{X_i}^G$ denotes the parents of X_i in G. Whenever the equation above holds, it is said that P factorizes over G.

Let P be any joint probability distribution over random variables $X_1, \ldots, X_n$. The following equation, which can be derived from the definition of conditional probability, is called the chain rule of probability:

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

Therefore, every joint probability distribution can be represented as a Bayesian network.

In a Bayesian network, every random variable is conditionally independent of its non-descendants given its parent variables. This is called the local Markov property. Every variable is also independent of every other given its Markov blanket: its children, parents and other parents of its children.

Bayesian networks are useful to encode independencies between variables. For instance, a joint probability distribution over binary random variables $X_1, \ldots, X_n$ may be represented by a table with $2^n - 1$ values. However, the number of values needed to represent the same probability distribution may be considerably lower when independencies are considered. If a probability distribution is represented by a Bayesian network, only G and $\mathrm{Pa}_{X_i}^G$ need to be known for every X_i.

The following example illustrates the use of Bayesian networks. Let A, B, E and R be random variables. Variable A represents whether an alarm in a particular house is sounding, B represents whether the same house has been burgled recently, E represents whether there was an earthquake recently and R represents whether the radio is broadcasting an earthquake alert. Let $G = (\{A, B, E, R\}, \{(B, A), (E, A), (E, R)\})$ be a graph and $B = (G, P)$ a Bayesian network. From the graph, $P(A, B, E, R) = P(A|B, E)P(R|E)P(E)P(B)$. Without loss of generality, P could also be written as $P(A, B, E, R) = P(A|B, E, R)P(R|B, E)p(E|B)P(B)$. Using the local Markov property, it is possible to show that $P(A|B, E, R) = P(A|B, E)$, $P(R|B, E) = P(R|E)$, $P(E|B) = P(E)$. In this case, it is possible to represent P in a table with only 8 values instead of 15. Any posterior probability (probability after observing the state of some variables) can be inferred from this joint probability distribution.

Different directed acyclic graphs may represent the same independence assumptions between random variables in a Bayesian network. Two directed acyclic graphs are Markov equivalent if they represent the same set of conditional independence statements. Let an immorality in a directed acyclic graph $G = (V, E)$ be defined as a set of distinct vertices $\{u, v, w\}$ such that $(u, v) \in E$, $(w, v) \in E$, $(u, w) \notin E$ and $(w, u) \notin E$. Two directed acyclic graphs are Markov equivalent if and only if they they have the same set of immoralities and the same underlying graph.

Causal relationships may be useful for building models intuitively. In this case, a causal variable often has an edge ending at an effect variable. In causal models, causal reasoning is defined as computing the posterior probability of effects given causes. The posterior probability of causes given effects is called evidential reasoning. Intercausal reasoning combines causal and evidential reasoning. *Explaining away* is the term given to the intercausal reasoning pattern in that a cause explains an effect and, by consequence, lowers the probability of another cause. However, it must be clear that independencies in Bayesian networks are not (in general) causal. A model in which the edges are causally reversed may be equally valid to represent the same set of independencies.

It may be impossible to represent all independency statements on given set of variables by a Bayesian network without adding variables.

The following steps can be followed to model a real inference problem as a Bayesian network. Firstly, the relevant random variables need to be identified. If the model is intuitively causal, this is often accomplished from the bottom up (effects to causes). Hidden variables (which can never observed) may be introduced to represent additional independency statements. The graph is defined in such a way as to encode the independencies between variables. The independencies may be checked by inspecting the joint probability distribution defined by the graph or by the local Markov property. Probabilistic inference is performed by computing the probability of each state of interest conditioned on the known states of other variables.

The following definitions are useful to establish if a random variable X is independent of Y given Z in a Bayesian network.

Let $G = (V, E)$ be a directed acyclic graph, G' its underlying graph and P' a path on G'. A set of three distinct nodes $\{u, v, w\}$ in P' is a v-structure if $(u, v) \in E$ and $(w, v) \in E$ and these two edges are in P'. In this structure, v is called a collider. In simple terms, in a v-structure, a collider v has two distinct neighbors u and w in P' with edges ending in v in the corresponding directed acyclic graph.

An undirected path P is said to be active given the set of random variables $\mathcal{Z}$ if all non-colliders in P are not in $\mathcal{Z}$ and all colliders are in $\mathcal{Z}$ or have a descendant in $\mathcal{Z}$.

In a Bayesian network $B = (G, P)$, if all undirected paths between a random variables X and Y are inactive given the set of random variables $\mathcal{Z}$, then $X \perp\!\!\!\perp Y|\mathcal{Z}$. In such case, it is said that X and Y are d-separated by $\mathcal{Z}$, denoted d-sep$_G(X; Y|Z)$. The notation extends to sets of variables naturally. Random variables that are not d-separated are d-connected. It is important to note that d-connected variables may still be independent according to P. However, variables that are d-separated by $\mathcal{Z}$ are always independent given $\mathcal{Z}$ in any joint probability distribution represented by the Bayesian network.

Let P be a distribution and $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ be sets of random variables. We define $\mathcal{I}(P)$ as the set of independency statements of the form $\mathcal{X} \perp\!\!\!\perp \mathcal{Y}|\mathcal{Z}$ that hold in P. If G is a graph in a Bayesian network, then $\mathcal{I}(G)$ denotes the set of statements of the form $\mathcal{X} \perp\!\!\!\perp \mathcal{Y}|\mathcal{Z}$ such that d-sep$_G(\mathcal{X}; \mathcal{Y}|Z)$. A graph is an I-map for a set of dependencies I if $\mathcal{I}(G) \subseteq I$. Therefore, if $B = (G, P)$ is a Bayesian network, $\mathcal{I}(G)$ is an I-map for $\mathcal{I}(P)$. If $\mathcal{I}(G) = \mathcal{I}(P)$, then G is a perfect map for P.

A conditional Bayesian network B over $\mathcal{Y}$ given $\mathcal{X}$ is defined by a graph G and a conditional probability distribution P. The nodes of G are $\mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$, the three sets being disjoint. The variables in $\mathcal{X}$ are called inputs, the variables in $\mathcal{Y}$ outputs and in $\mathcal{Z}$ encapsulated. The variables in $\mathcal{X}$ have no parents in G. The conditional probability distribution P can be written as:

$$P(\mathcal{Y}, \mathcal{Z}|\mathcal{X}) = \prod_{X \in \mathcal{Y} \cup \mathcal{Z}} P(X|\operatorname{Pa}_X^G).$$

In simple terms, a conditional Bayesian network represents a joint probability distribution over $\mathcal{Y} \cup \mathcal{Z}$ conditioned on $\mathcal{X}$.

In a scientific experiment represented by a Bayesian network, when the value of a variable X is fixed to x, the variable should not have any parents in the directed graph, otherwise the inferences performed will take into account the probability of fixing X to x given the state of other variables. Failing to model scientific experiments in this way leads to counterintuitive results such as Simpson's paradox. Therefore, such a variable X should belong to the input of a conditional Bayesian network.

In supervised learning, a subarea of machine learning, naive Bayes is a very simple model for classification based on a joint probability distribution that factorizes as:

$$P(C, X_1, \ldots, X_n) = P(C) \prod_i^n P(X_i|C),$$

where C is a random variable representing the class of a sample and each X_i a random variable representing an attribute of the same sample. Note the assumption that $X_i \perp\!\!\!\perp X_j|C$ for every $i \neq j$. Given training data to estimate $P(X_i = x|C = c)$ for every pair (x, c) and the features for a new sample, it is easy to find its most likely classification according to the model. Naive Bayes has been traditionally used for textual classification.

4 Template Models

A template model represents a set of probabilistic graphical models that have some structure in common. Two examples of template models are described in this section: dynamic Bayesian Networks and plate models.

4.1 Dynamic Bayesian Networks

A dynamic Bayesian network is a template model that may be used to represent systems whose evolution is studied in discrete steps. The evolution is often temporal, and the analogy is used throughout this section. In a dynamic Bayesian network, the state of a system at time t is represented by an assignment to random variables.

Let $X^{(t)}$ be a random variable that represents some aspect of a system at time t. We use $X^{(t_1:t_2)}$ to denote the set $\{X^{(t)}|t \in [t_1, t_2]\}$, where all variables have the same set of possible states. If $\mathcal{X}^{(t)} = \{X_1^{(t)}, \ldots, X_n^{(t)}\}$ is a set of variables, then $\mathcal{X}^{(t1:t2)} = \{X_i^{(t)}|i \in [1, n] \text{ and } t \in [t_1, t_2]\}$.

Let $\mathcal{X}^{(0:T)}$ be a set of variables as above. By the chain rule of probabilities in its factor formulation:

$$P(\mathcal{X}^{(0:T)}) = P(\mathcal{X}^{(0)}) \prod_{t=1}^{T} P(\mathcal{X}^{(t)}|\mathcal{X}^{(0:t-1)}).$$

A system is called Markovian if $(\mathcal{X}^{(t+1)} \perp\!\!\!\perp \mathcal{X}^{(0:t-1)}|\mathcal{X}^{(t)})$ for all t. Intuitively, this means that the state of the system at a given time represents all information needed to predict the future. If the system is Markovian, its joint probability distribution can be written as:

$$P(\mathcal{X}^{(0:T)}) = P(\mathcal{X}^{(0)}) \prod_{t=1}^{T} P(\mathcal{X}^{(t)}|\mathcal{X}^{(t-1)}). \tag{1}$$

Using Markovian models for systems of interest depends on including enough information in $\mathcal{X}^{(t)}$ so that the equation above becomes a reasonable approximation.

A Markovian system is stationary if $P(\mathcal{X}^{(t+1)}|\mathcal{X}^{(t)})$ is the same factor for all t. In other terms, the Markovian system is stationary if, for all t:

$$P(\mathcal{X}^{(t+1)} = \xi'|\mathcal{X}^{(t)} = \xi) = P(\mathcal{X}' = \xi'|\mathcal{X} = \xi),$$

for some conditional probability distribution $P(\mathcal{X}'|\mathcal{X})$, called the transition model, over sets of random variables $\mathcal{X}$ and $\mathcal{X}'$.

Consider a stationary Markovian system represented, at time t, by the set of variables $\mathcal{X}^{(t)}$ and $P(\mathcal{X}'|\mathcal{X})$ the conditional probability distribution that follows the equation above. A 2-time-slice Bayesian network is a conditional Bayesian network over $\mathcal{X}'$ given $\mathcal{X}_I \subseteq \mathcal{X}$, which defines the following conditional probability distribution:

$$P(\mathcal{X}'|\mathcal{X}) = \prod_{X'_i \in \mathcal{X}'} P(X'_i|\operatorname{Pa}^G_{X'_i}).$$

Because of the stationary property, the 2-time-slice Bayesian network defines the probability distribution $P(\mathcal{X}^{(t+1)})|\mathcal{X}^{(t)})$ for every t.

An edge between elements of $\mathcal{X}$ and $\mathcal{X}'$ is called an inter-time-slice edge, and an edge inside $\mathcal{X}'$ is called an intra-time-slice edge. Edges between corresponding variables across time are called persistence edges.

Consider a stationary Markovian system represented, at time t, by the set of variables $\mathcal{X}^{(t)}$. A dynamic Bayesian network is a pair $(B_0, B_\rightarrow)$ where B_0 is a Bayesian network over $\mathcal{X}^{(0)}$ and $B_\rightarrow$ is a 2-time-slice Bayesian network for the system.

A Bayesian network B_T can be obtained, for any time T, from a dynamic Bayesian network $(B_0, B_\rightarrow)$. This Bayesian network is said to be unrolled from the dynamic Bayesian network. The graph of B_T contains the graph in B_0 and the structure represented by $B_\rightarrow$ replicated for every pair $(\mathcal{X}^{(t)}, \mathcal{X}^{(t+1)})$, for every $1 \leq t+1 \leq T$. The joint probability distribution over $\mathcal{X}^{(0:T)}$ can be written from the conditional probability distributions defined by B_0 and $B_\rightarrow$.

A hidden Markov model is a very simple dynamic Bayesian network, which has a persisting variable $S^{(t)}$, which represents the system state, with a single intra-time-slice connection to a variable $O^{(t)}$, which represents the observed state. This model leads to more complicated models, such as the factorial hidden Markov Model and the coupled hidden Markov Model, which are widely used in practical problems. A hidden Markov model is often represented by a graph whose vertices are states (possible assignments to $S^{(t)}$) and every edge (u, v) is labeled by $P(S^{(t+1)} = v|S^{(t)} = u)$. This representation should not be confused with the probabilistic graphical models described in this text.

A linear dynamic system (sometimes called Kalman filter) represents a system of variables that evolve linearly over time with Gaussian noise. Such systems can be represented by a dynamic Bayesian network where the variables are all continuous and the dependencies are linear Gaussian.

4.2 Plate Models

A plate model is a compact representation of a set of Bayesian networks whose structure contains many replications of a simpler structure. A formal definition of this model is available in [1]. This text presents only a superficial overview of the model. A plate model can be conveniently represented by a graphical depiction, which is also introduced.

Let $Q_1, \ldots, Q_k$ be disjoint sets called classes. The set of all random variables of the form $X(q_1, \ldots, q_k)$, where each $q_i \in Q_i$, can be represented by a circle inscribed with the letter X inside rectangles (called plates) representing each class (with their respective names also inscribed in unambiguous locations). In this model, X is called an attribute. Arrows drawn between attributes represent that there should be a directed edge between every random variable represented by an attribute in an instance of the plate model. In a plate model, there are no arrows between an attribute X and an attribute Y if the classes associated with X are not a subset of the classes associated with Y. In a manner similar to dynamic Bayesian networks, valid plate models can be unraveled into Bayesian networks. The conditional probability distributions can be shared or not among replicated structures, often diminishing the number of parameters to be described.

For example, consider the problem of modeling how the difficulty of a course and intelligence of a student determine the grade of the student in the course. For a particular student, course, and grade, this could be represented by a Bayesian network whose graph is $G = (\{D, I, G\}, \{(D, G), (I, G)\})$.

However, suppose one is interested in using the same model to reason about all courses, students and grades in a school (ignoring the fact that not all students have frequented all courses). This could be accomplished by defining a plate model (using the graphical language described above) containing the attribute I inside a plate for the class "Students", an attribute D inside a plate for the class "Courses" and an attribute G inside both plates. The graphical depiction should also contain an arrow between D and G and I and G. From this graphical depiction defining a plate model, the unraveled Bayesian network would represent the fact that, for a specific student s and a class c, the grade $G(s, c)$ (a random variable) is directly dependent of $D(c)$ and $I(s)$. Additionally, if the assumption is made that the joint probability distribution $P(D(c), I(s), G(s, c))$ is the same across all pairs of students and courses, very powerful inferences can be performed with very few parameters.

Additional template models are covered in [1]. Alternatively, algorithms are powerful tools to describe complex Bayesian networks.

5 Structured Conditional Probability Distributions

This section presents several ways to represent conditional probability distributions (factors) for variables in a Bayesian network.

In a tabular representation, a conditional probability distribution $P(X|\operatorname{Pa}_X^G)$ is a table with one entry for each valid assignment x to the random variable X and each valid assignment pa_X to its parents Pa_X^G in the Bayesian network. Naturally, $P(x|\operatorname{pa}_X)$ must be positive for every x and pa_X. Also, $\sum_x P(x|pa_X) = 1$.

Let X be a random variable with k parents Pa_X^G. If $m = |\operatorname{Val}(X)|$ and $n = \max_{Y \in \operatorname{Pa}_X^G}(|\operatorname{Val}(Y)|)$, a tabular representation requires $O((m-1)n^k)$ memory.

A conditional distribution $P(X|\operatorname{Pa}_X^G)$ is deterministic if there is a function $f : \operatorname{Val}(\operatorname{Pa}_X^G) \to \operatorname{Val}(X)$ such that $P(x|\operatorname{pa}_X) = 1$ if $f(\operatorname{pa}_X) = x$ and 0 otherwise. In other words, given an assignment to the parents, the assignment to X is fully determined. A variable in a Bayesian network with a deterministic conditional distribution may induce additional independency statements, since the observation of Pa_X^G reveals the state of X.

Let $\mathcal{X}, \mathcal{Y}$ and $\mathcal{Z}$ be pairwise disjoint sets of variables and $\mathcal{C}$ a set of variables. Let $c \in \operatorname{Val}(\mathcal{C})$. $\mathcal{X}$ and $\mathcal{Y}$ are contextually independent given $\mathcal{Z}$ and the *context* c if $P(\mathcal{X}|\mathcal{Y}, \mathcal{Z}, c) = P(\mathcal{X}|\mathcal{Z}, c)$ whenever $P(Y, Z, c) > 0$. This is denoted by $(\mathcal{X} \perp\!\!\!\perp_c \mathcal{Y}|Z, c)$.

As an example of contextual independence, consider a binary random variable X that is a deterministic *and* of its parents Y and Z. Clearly:

$$P(X = 0|Z = 0) = P(X = 0|Z = 0, Y = 1) = P(X = 0|Z = 0, Y = 0) = 1.$$

Therefore, $(X \perp\!\!\!\perp_c Y|Z = 0)$. However, it is not necessary that $P(X = 0|Z = 1) = P(X = 0|Z = 1, Y = 1) = 0$. Therefore, in general, it is not true that $(X \perp\!\!\!\perp Y|Z)$.

There are several ways to use context-specific independencies to simplify the representation of conditional probability distributions. A tree conditional probability distribution is one of the simplest.

A tree conditional probability distribution for variable X is a rooted tree. Each leaf is labeled with a probability distribution over X. Each interior node is labeled with a variable $Z \in \operatorname{Pa}_X^G$. Every interior node Z has, for each $z \in \operatorname{Val}(Z)$, an edge labeled with z. A branch β in a tree conditional probability distribution is a path from the root to a leaf. In a valid tree conditional probability distribution, no branch contains two nodes labeled by the same variable. The context c induced by a branch β is the set of variable assignments $Z = z$ such that z is an edge in β. For a context c induced by any branch β, the conditional probability distribution represented by the tree has the property $(X \perp\!\!\!\perp_c \mathcal{Y}|c)$, where $\mathcal{Y}$ is the set of variables not in the branch. Also, the probability distribution over X that labels the leaf in β is $P(X|c)$.

A tree conditional probability distribution is a representation of a factor, not a joint probability distribution, and should not be confused with a Bayesian network. Not all context-specific independency statements can be represented by a tree conditional probability distribution.

A conditional probability distribution $P(Y|A, Z_1, \ldots, Z_k)$ is a multiplexer if $\operatorname{Val}(A) = \{1, \ldots, k\}$ and $P(Y = y|A = a, Z_1 = z_1, \ldots, Z_k = z_k) = 1$ if $y = z_a$ (and 0 otherwise). Such conditional probability distributions are useful to represent that a choice $A = a$ makes Y independent of every parent except Z_a. Introducing the variables A and Y may considerably reduce the number of parents that a variable dependent on $Z_1, \ldots, Z_k$ must have, making the conditional probability distribution simpler to represent.

A conditional probability distribution $P(Y|X_1, \ldots, X_k)$ over binary variables is a noisy-or if there are $k+1$ noise parameters $\lambda_0, \lambda_1, \ldots, \lambda_k$ such that:

$$P(Y = 0|X_1 = x_1, \ldots, X_k = x_k) = (1 - \lambda_0) \prod_{i=1}^{k} (1 - \lambda_i)^{x_i}.$$

Intuitively, when applied to causal modeling, this conditional distribution represents that, for $i > 1$, when no other cause is present and $\lambda_0 = 0$, X_i causes Y with probability λ_i. The parameter λ_0 can be interpreted as the probability that Y is caused by unspecified causes, when no specified cause is present.

Another way to understand the noisy-or model is to consider Y as a deterministic *or* function of parents $X'_0, \ldots, X'_k$, such that $P(X'_i = 1|X_i = 1) = \lambda_i$ and $P(X'_i = 1|X_i = 0) = 0$, for $i > 1$, and $P(X'_0 = 1) = \lambda_0$.

The sigmoid (or logistic) function f is defined as $f(x) = \frac{1}{1+e^{-x}}$. For every $x \in \mathbb{R}$, $0 < f(x) < 1$.

Let Y and $X_1, \ldots, X_k$ be binary random variables. Then $P(Y|X_1, \ldots, X_k)$ is a logistic conditional probability distribution if there are $k + 1$ weights $w_0, \ldots, w_k$ such that:

$$P(Y = 1|X_1 = x_1, \ldots, X_k = x_k) = \frac{1}{1 + e^{-(w_0 + \sum_i w_i x_i)}}.$$

Intuitively, the weight w_i dictates whether X_i makes Y more or less likely. The sigmoid function is monotonically increasing and guarantees that $P(Y = 1|\cdot)$ is between 0 and 1. This model can be further generalized to deal with the discrete multinomial case using a binary coding scheme [1].

All the conditional probability distributions described so far involved only discrete random variables.

Let Y and $X_1, \ldots, X_k$ be continuous random variables. Then $P(Y|X_1, \ldots, X_k)$ is a linear Gaussian model if there are parameters $\beta_0, \ldots, \beta_k$ and σ^2 such that:

$$f(Y|x_1, \ldots, x_k) = \mathcal{N}(\beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k; \sigma),$$

where f is the probability density function for P, σ^2 is a constant and $\mathcal{N}(\mu; \sigma)$ is the Gaussian distribution. In this model, Y can be interpreted as a linear combination of $X_1, \ldots, X_k$ with Gaussian noise.

6 Markov Networks

Markov networks are undirected graphical models. Undirected models are more natural representations for some problems.

Let $\Phi = \{\phi_1(D_1), \ldots, \phi_K(D_K)\}$ be a set of non-negative factors such that every $D_i \subseteq \{X_1, \ldots, X_n\}$. A joint probability distribution P is a Gibbs distribution parameterized by Φ if it can be written as:

$$P(X_1, \ldots, X_n) = \frac{\prod_{i=1}^{K} \phi_i(D_i)}{Z},$$

where Z (historically called *partition function*) is defined as:

$$Z = \sum_{X_1, \ldots, X_n} \prod_{i=1}^{K} \phi_i(D_i).$$

It is important to notice that the the factor P as defined above is a joint probability distribution for any set of non-negative factors as long as $Z > 0$.

A Gibbs distribution P parameterized by $\Phi = \{\phi_1(D_1), \ldots, \phi_K(D_K)\}$ *factorizes* over the (undirected) graph $\mathcal{H}$ if each D_i induces a complete subgraph in $\mathcal{H}$. In this context, $\mathcal{H}$ is called a Markov network. Also, each $\phi_i(D_i)$ is called a *potential*.

Intuitively, for a factor $\phi(\mathcal{X}) \in \Phi$, the value $\phi(\mathcal{X} = \mathbf{x})$ should be proportional to the *compatibility* of the assignment $\mathbf{x}$ to the variables in $\mathcal{X}$. In general, this compatibility should not be interpreted as the joint probability of the assignment. However, when $\phi(\mathcal{X} = \mathbf{x}) = 0$, it follows that $P(\mathcal{X} = \mathbf{x}) = 0$.

From the definition of factorization of a Gibbs distribution P over a Markov network $\mathcal{H}$, it follows that it is always possible to define a joint probability distribution equivalent to P using factors defined on maximal cliques of $\mathcal{H}$. However, in general, doing so may obscure the definition of each factor. It is important to note that Gibbs distributions over the same set of variables, but parameterized by different factors, may factorize over the same Markov network graph.

If a Gibbs distribution is parameterized by factors whose scope has always 2 variables or less and factorizes over a Markov Network $\mathcal{H}$, then $\mathcal{H}$ is called a pairwise Markov network. These networks are widely used in practice.

Let $\mathcal{X}$ and $\mathcal{Y}$ be disjoint sets of random variables. A conditional random field is an undirected graph $\mathcal{H}$ whose nodes are $\mathcal{X} \cup \mathcal{Y}$. The graph $\mathcal{H}$ is associated with a set of factors $\Phi = \{\phi_1(D_1), \ldots, \phi_K(D_K)\}$, such that each $D_i \not\subseteq \mathcal{X}$ and induces a complete subgraph in $\mathcal{H}$. The conditional random field represents the following conditional distribution:

$$P(\mathcal{Y}|\mathcal{X}) = \frac{\prod_{i=1}^{K} \phi_i(D_i)}{Z(\mathcal{X})}$$

$$Z(\mathcal{X}) = \sum_{\mathcal{Y}} \prod_{i=1}^{K} \phi_i(D_i).$$

It is important to notice that the the factor P as defined above is a conditional probability distribution as long as Z is a positive factor. By definition, there is no factor whose scope involves only variables in $\mathcal{X}$.

The advantage of using a conditional random field over a Markov network appears when the model only needs to represent probabilities conditioned on $\mathcal{X}$, since the definition of the factors may become much simpler. A conditional random field is an idea analogous to conditional Bayesian networks.

Let $\mathcal{H} = (\mathcal{X}, E)$ be a Markov network and $P = X_1, \ldots, X_k$ a path in $\mathcal{H}$. Let $\mathcal{Z} \subseteq \mathcal{X}$ be a set of variables. The path P is active given $\mathcal{Z}$ if $X_i \notin \mathcal{Z}$ for all i.

A set of nodes $\mathcal{Z}$ separates $\mathcal{X}$ and $\mathcal{Y}$ in a Markov network $\mathcal{H}$, denoted $\text{sep}_{\mathcal{H}}(\mathcal{X}; \mathcal{Y}|\mathcal{Z})$ if there is no active path between any two nodes $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$ given $\mathcal{Z}$.

The set of independencies associated with $\mathcal{H}$ is $\mathcal{I}(\mathcal{H}) = \{(\mathcal{X} \perp\!\!\!\perp \mathcal{Y}|\mathcal{Z}) | \text{sep}_{\mathcal{H}}(\mathcal{X}; \mathcal{Y}|\mathcal{Z})\}$. It can be shown that $\mathcal{I}(\mathcal{H})$ is an I-map for any Gibbs distribution that factorizes over $\mathcal{H}$.

In contrast with Bayesian networks, it is important to note that observing more variables can never cause additional dependencies in a Markov network.

Let P be a positive distribution over $\mathcal{X}$ and $\mathcal{H}$ a Markov network graph over $\mathcal{X}$. If $\mathcal{H}$ is an I-map for P, then P is a Gibbs distribution that factorizes over $\mathcal{H}$.

A factor graph $\mathcal{F}$ is an undirected graph containing nodes representing random variables and factors. There are edges only between factors and random variables, and only when a random variable belongs to the scope of a factor. Factor graphs are useful to unambiguously represent the structure of a Gibbs distribution.

As already mentioned, a Gibbs distribution P can be uniquely defined by a set of non-negative factors $\Phi = \{\phi_1(D_1), \ldots, \phi_K(D_K)\}$. When all factors in Φ are positive, another way to define P is using the factors $\epsilon_i(D_i)$:

$$\epsilon_i(D_i) = -\ln(\phi_i(D_i)).$$

The Gibbs distribution P can be written as:

$$P(X_1, \ldots, X_n) = \frac{\prod_{i=1}^{K} \phi_i(D_i)}{Z}$$
$$P(X_1, \ldots, X_n) = \frac{e^{-\sum_{i=1}^{K} \epsilon_i(D_i)}}{Z},$$

where (using factor marginalization):

$$Z = \sum_{X_1, \ldots, X_n} e^{-\sum_{i=1}^{K} \epsilon_i(D_i)}.$$

A distribution P is a log-linear model over a Markov network $\mathcal{H}$ if it can be written as:

$$P(X_1, \ldots, X_n) = \frac{e^{-\sum_{i=1}^{K} w_i f_i(D_i)}}{Z},$$

for a set of weights $\{w_1, \ldots, w_K\}$ and a set of factors $\{f_1(D_1), \ldots, f_K(D_K)\}$. This is a very useful representation for many models of practical interest.

The rest of this section describes the application of the undirected models in practical problems.

The objective of image denoising is to restore the value of possibly noisy pixel values. Each pixel i can be associated to a random variable X_i representing its denoised intensity value and a variable Y_i representing its observed intensity value. The conditional probability distribution $P(X_1, \ldots, X_n|Y_1, \ldots, Y_n)$ can be represented by a conditional random field. This conditional random field is parameterized by energy functions for every pair (X_i, X_j) of adjacent pixels, e.g., $\epsilon(X_i = x_i, X_j = x_j) = \min(||x_i - x_j||, d_{\max})$ and energy functions for every pair (X_i, Y_i) in a similar manner. The denoised image is encoded by the assignment to the variables X_i, conditioned on the variables Y_i, with maximum probability. A similar idea can be applied to the image segmentation problem, where the task is associating a label to every pixel in an image. Given the classification by a supervised learning technique to a pixel (or image region), local coherence can be enforced by a conditional random field.

An Ising model is a model in statistical physics for systems involving interacting atoms. Each random variable X_i is binary, with $\text{Val}(X_i) = \{+1, -1\}$. The energy function between two variables X_i and X_j is defined as $\epsilon(x_i, x_j) = w_{i,j} x_i x_j$. There are such energy functions for each pair of atoms that interact directly. There is also one energy function $\epsilon(x_i) = u_i x_i$ defined for each variable, which biases individual variables to have one value or another. These factors define the following probability for an assignment to the variables X_i:

$$P(x_1, \ldots, x_n) = \frac{e^{-\sum_{i<j} w_{i,j} x_i x_j - \sum_i u_i x_i}}{Z}.$$

In this model, when $w_{i,j} < 0$, atoms that interact directly tend to adopt the same state. The opposite happens when $w_{i,j} > 0$.

A Boltzmann machine is a similar model in which $\text{Val}(X_i) = \{0, 1\}$. In this model, given a variable Y and its neighbors $X_1, \ldots, X_n$ in the corresponding Markov network, the following property holds:

$$P(Y = 1|x_1, \ldots, x_n) = \frac{1}{1 + e^{-\sum_{i=1}^{n} w_i x_i}}.$$

This is highly related to the activation function commonly used in artificial neural networks.

Another useful class of Markov networks is comprised of metric Markov networks. A metric Markov network is defined by a graph $\mathcal{H} = (\mathcal{X}, E)$ over random variables $X_1, \ldots, X_n$. The objective is to assign to each X_i a label in the set $\mathcal{V} = \{v_1, \ldots, v_K\}$. Each node has a preference over labels, represented by an energy function $\epsilon_i(X_i)$. There is also an energy function $\epsilon_i(X_i, X_j)$ associated to each edge (X_i, X_j) in the graph, which is usually used to impose a constraint that neighboring nodes should have similar assignments.

Since the objective in a metric Markov network is to find the most likely assignment, the problem can be formulated finding the assignment that minimizes the following function:

$$E(x_1, \ldots, x_n) = \sum_{i=1}^{n} \epsilon_i(x_i) + \sum_{(X_i, X_j) \in E} \epsilon_{i,j}(x_i, x_j).$$

For every neighboring pair (X_i, X_j), the energy function $\epsilon_{i,j}$ can be defined as a distance metric between labels $d : \mathcal{V} \times \mathcal{V} \to [0, +\infty)$, making the parameterization simpler.

7 Variable Elimination

One of the most common queries performed on a probabilistic graphical model for a probability distribution P over variables $\mathcal{X}$ is the computation of a factor $P(\mathbf{Y}|\mathbf{E} = e)$. Let $\mathbf{W} = \mathcal{X} - \mathbf{Y} - \mathbf{E}$. If y, e, w are assignments to $\mathbf{Y}, \mathbf{E}, \mathbf{W}$:

$$\begin{aligned} P(y|e) &= \frac{P(y, e)}{P(e)} \\ P(y, e) &= \sum_{w} P(y, e, \mathbf{W} = w) \\ P(e) &= \sum_{y} P(\mathbf{Y} = y, e). \end{aligned}$$

Therefore, by computing $P(\mathbf{Y}, \mathbf{E} = e)$, it is easy to arrive at $P(\mathbf{Y}|\mathbf{E} = e)$

In the general case, inference in probabilistic graphical models is a *hard* problem. Concretely, given a Bayesian network $\mathcal{B}$ over variables $\mathcal{X}$ representing the joint distribution P and a variable $X \in \mathcal{X}$, deciding whether $P(X = x) > 0$ for a given $x \in \text{Val}(X)$ is an NP-complete problem. This implies that finding $P(X = x)$ for a given x is NP-hard.

Despite this worst case result, efficient inference can be performed in many models of practical interest.

Variable elimination is an algorithm for exact inference in graphical models. Let $\mathcal{X}$ be a set of variables, Φ be a set of factors involving only variables in $\mathcal{X}$. Let $\mathcal{Y} \subseteq \mathcal{X}$ and $\mathcal{Z} = \mathcal{X} - \mathcal{Y}$. For any ordering $\prec$ over $\mathcal{Z}$, the sum product variable elimination algorithm (Alg. 1) returns a factor $\phi^*(\mathcal{Y})$ such that:

$$\phi^*(\mathcal{Y}) = \sum_{\mathcal{Z}} \prod_{\phi \in \Phi} \phi.$$

The algorithm can be applied to compute the joint distribution $P(\mathcal{Y})$ for a Bayesian network over variables $\mathcal{X}$ by letting Φ be the set of conditional probability distributions whose product represents $P(\mathcal{X})$.

In the case of Markov networks, Φ is the set of clique potentials, and the result of the algorithm needs only to be divided by the partition function to represent $P(\mathcal{Y})$.

The algorithm depends on the following properties of factors. If ϕ_1, ϕ_2 and ϕ_3 are factors, then $\phi_1 \cdot \phi_2 = \phi_2 \cdot \phi_1$ and $\phi_1 \cdot (\phi_2 \cdot \phi_3) = (\phi_1 \cdot \phi_2) \cdot \phi_3$. Also, $\sum_X \sum_Y \phi_1 = \sum_Y \sum_X \phi_1$. If $X \notin \text{Scope}[\phi_1]$, $\sum_X \phi_1 \cdot \phi_2 = \phi_1 \cdot \sum_X \phi_2$.

Consider the naive algorithm for inference in graphical models that explicitly computes the full joint probability distribution $P(\mathcal{X})$ and then marginalizes $\mathcal{Z}$ to arrive at $P(\mathcal{Y})$. Clearly, in the general case, some factor products are computed several times. The efficiency of variable elimination is related to the fact that intermediary factors are stored and computed only once. This is possible because not all factors (necessarily) involve a given variable, and therefore are not involved in the product and marginalization occurring in lines 5 and 6 of Alg. 1.

Algorithm 1 Sum-product variable elimination

Input: set of factors Φ over variables $\mathcal{X}$, set of variables to be eliminated $\mathcal{Z}$, ordering $\prec$ over $\mathcal{Z}$.
Output: factor $\phi^*(\mathcal{Y}) = \sum_{\mathcal{Z}} \prod_{\phi \in \Phi} \phi$, where $\mathcal{Y} = \mathcal{X} - \mathcal{Z}$

1: Let $Z_1, \ldots, Z_k$ be an ordering of $\mathcal{Z}$ respecting $\prec$
2: **for** each i in $1, \ldots, k$ **do**
3: $\quad \Phi' \leftarrow \{\phi \in \Phi | Z_i \in \text{Scope}[\Phi]\}$
4: $\quad \Phi'' \leftarrow \Phi - \Phi'$
5: $\quad \psi \leftarrow \prod_{\phi \in \Phi'} \phi$
6: $\quad \tau \leftarrow \sum_{Z_i} \psi$
7: $\quad \Phi \leftarrow \Phi'' \cup \{\tau\}$
8: **end for**
9: $\phi^*(\mathcal{Y}) \leftarrow \prod_{\phi \in \Phi} \phi$

Computing $P(\mathbf{Y}|\mathbf{E} = e)$ using sum-product variable elimination is easily accomplished by letting $\Phi = \{\phi[\mathbf{E} = e] | \phi \in \Phi'\}$, where Φ' is the set of factors that parameterize the Bayesian or Markov network in question. The factor $\phi^*(\mathcal{Y})$ can be used to compute the joint distribution $P(\mathbf{Y}|\mathbf{E} = e)$ as follows:

$$P(\mathbf{Y}|\mathbf{E} = e) = \frac{\phi^*(\mathcal{Y})}{\sum_{y' \in \text{Val}(\mathcal{Y})} \phi^*(\mathcal{Y} = y')}.$$

The complexity of the variable elimination algorithm is dominated by the number of entries in the factors generated in line 5, which is potentially exponential on the number of variables for elimination. This complexity is highly dependent on the order in which the variables are eliminated. Finding the best ordering (which leads to minimal complexity) is another NP-hard problem. However, even after finding the best ordering, the inference problem is still intractable.

If the set of factors Φ in a given iteration (before line 7) is represented by a Markov network $\mathcal{H}$, line 7 can be interpreted as removing vertex Z_i from $\mathcal{H}$ and directly connecting all neighbors of Z_i in the graph. Edges added in this way are called fill edges. The *induced graph* for a given elimination order is the union of the Markov networks created in this way for all iterations. The complexity of the algorithm is directly related to the size of the largest clique in the induced graph.

If a graph is an I-map for a set of factors and every minimal cycle in the graph is of length three, then there is an elimination ordering that does not introduce fill edges.

Several greedy heuristics can be employed to find a good elimination ordering. Considering a set of factors Φ represented by a Markov network $\mathcal{H}$ and a set of variables $\mathcal{Z}$ that still need to be eliminated, the next variable for elimination may be:

- The variable with the least amount of neighbors in $\mathcal{H}$ (min-neighbors heuristic)
- The variable that would introduce the least amount of fill edges (min-fill heuristic)
- The variable that would introduce the least sum of fill edge weights, where the weight of a fill edge (X_i, X_i) is defined as $|\text{Val}(X_i)||\text{Val}(X_j)|$ (weighted-min-fill heuristic).

In general, the min-fill and weighted-min-fill heuristics are observed to work well in practice.

It is important to notice that finding the elimination ordering heuristically can be performed before the sum product variable elimination algorithm even begins. It is possible to combine greedy heuristics with stochastic choices to produce several different orderings, which can be evaluated by the sum of the number of entries in the factors produced in line 5 of the variable elimination algorithm.

8 Belief Propagation

This section presents belief propagation, a family of techniques for inference in probabilistic graphical models.

A cluster graph $\mathcal{U}$ for a set of factors Φ over variables $\mathcal{X}$ is an undirected graph where every node i is associated with a set $\boldsymbol{C}_i \subseteq \mathcal{X}$ called *cluster*. Each factor $\phi \in \Phi$ must be associated to a single node, denoted by $\alpha(\phi)$, such that $\text{Scope}[\phi] \subseteq \boldsymbol{C}_{\alpha(\phi)}$. This property is called family preservation. Each edge between a pair of clusters $(\boldsymbol{C}_i, \boldsymbol{C}_j)$ is associated with a *sepset* $\boldsymbol{S}_{i,j} \subseteq \boldsymbol{C}_i \cap \boldsymbol{C}_j$.

When all variables are eliminated, the variable elimination algorithm described in the previous section implicitly defines a cluster graph. Let ψ_i denote the factor computed on line 5 of algorithm 1 at iteration i. Consider a cluster graph $\mathcal{U}$, where there is cluster $\boldsymbol{C}_i = \text{Scope}[\psi_i]$ for each iteration i and an edge between i and j if the factor τ_i (line 6) is directly used to compute ψ_j (line 5). Also, let $\boldsymbol{S}_{i,j} = \boldsymbol{C}_i \cap \boldsymbol{C}_j$.

Consider a cluster graph $\mathcal{U}$ over factors Φ. If, whenever $X \in \boldsymbol{C}_i$ and $X \in \boldsymbol{C}_j$, there is an unique path between i and j such that $X \in \boldsymbol{S}_e$ for every edge e in the path, then $\mathcal{U}$ has the running intersection property. In other words, this property implies that the induced subgraph containing all edges whose sepset contains X form a spanning tree between all clusters that contain X.

A cluster tree with the running intersection property is also called a clique tree, and each of its clusters is called a clique. This should not to be confused with the common concept of a clique in a graph. It can be shown that the cluster graph defined by variable elimination is a clique tree. Furthermore, it is always possible to obtain a clique tree such that $\boldsymbol{C}_i \not\subseteq \boldsymbol{C}_j$, for any pair of distinct nodes i and j, from a clique tree defined by variable elimination.

Clique tree message passing is an algorithm for exact inference based on clique trees. Consider a clique tree $\mathcal{T}$ over factors Φ. In the clique tree message passing algorithm, the initial potential ψ_j of cluster C_j is defined as:

$$\psi_j(\boldsymbol{C}_j) = \prod_{\phi \mid \alpha(\phi)=j} \phi.$$

It follows from the definition above, and by the construction of the clique tree, that:

$$\prod_{\phi \in \Phi} \phi = \prod_j \psi_j.$$

Let r be any node in the clique tree. We let $p_r(i)$ denote the predecessor of i in the unique path coming from r in the clique tree. We also denote by Nb_i the set of neighbors of i.

We define a message from i to j, denoted $\delta_{i \to j}$, as the following factor:

$$\delta_{i \to j} = \sum_{C_i - \boldsymbol{S}_{i,j}} (\psi_i(\boldsymbol{C}_i) \prod_{k \in (\text{Nb}_i - \{j\})} \delta_{k \to i}).$$

In words, the message going from i to j is a factor corresponding to the product between the initial potential of cluster $\boldsymbol{C}_i$ with all messages received by i, except those sent by j, marginalized over the variables that are not in the sepset between i and j.

Using these definitions, the upward pass of variable elimination by message passing in a clique tree is shown in algorithm 2. The reason for calling this an upward pass will be explained shortly. In the algorithm, a clique $\boldsymbol{C}_i$ is said to be ready after the messages from all of its neighbors except for $p_r(i)$ are computed. The algorithm can be shown to compute the belief factor β_r:

$$\beta_r(\boldsymbol{C}_r) = \sum_{\mathcal{X} - \boldsymbol{C}_r} \prod_{\phi \in \Phi} \phi.$$

It can be easily shown that there is always a clique ready at line 5.

Algorithm 2 Upward pass of variable elimination in clique tree

Input: set of factors Φ over variables $\mathcal{X}$, clique tree $\mathcal{T}$ containing k cliques, function assigning factors to cliques α, root r.

Output: factor $\beta_r(\boldsymbol{C}_r)$.

1: **for** each i in $1, \ldots, k$ **do**
2: $\quad \psi_i(\boldsymbol{C}_i) \leftarrow \prod_{\{\phi \in \Phi \mid \alpha(\phi)=i\}} \phi$
3: **end for**
4: **while** C_r is not ready **do**
5: $\quad C_i \leftarrow$ ready clique
6: $\quad \delta_{i \to p_r(i)} \leftarrow \sum_{C_i - \boldsymbol{S}_{i,p_r(i)}} (\psi_i(\boldsymbol{C}_i) \prod_{k \in (\text{Nb}_i - \{p_r(i)\})} \delta_{k \to i})$
7: **end while**
8: $\beta_r(\boldsymbol{C}_r) \leftarrow \psi_r(\boldsymbol{C}_r) \prod_{k \in \text{Nb}_r} \delta_{k \to r}$

In words, this algorithm first computes the initial potentials. Then, starting from a leaf of the rooted tree, cliques that are ready send a message to their predecessors. The process continues until the root is ready.

The resulting factor $\beta_r(\boldsymbol{C}_r)$ can be used to compute the joint probability distribution over any set of variables $\mathbf{Y} \subseteq \boldsymbol{C}_r$ by marginalization (and division by the partition function, in case of factors from Markov networks).

Consider a clique tree with cliques $\boldsymbol{C}_i$. We define the $\boldsymbol{C}_i$-side of the edge (i, j) as the set of cliques that reach j through i. Let $\mathcal{F}_{\prec(i\to j)}$ denote the set of initial potentials on the $\boldsymbol{C}_i$-side of the edge (i, j), and $\mathcal{V}_{\prec(i\to j)}$ the set of variables that are in a clique on the $\boldsymbol{C}_i$-side but are not in $\boldsymbol{S}_{i,j}$. It can be shown that:

$$\delta_{i\to j} = \sum_{\mathcal{V}_{\prec(i\to j)}} \prod_{\phi\in\mathcal{F}_{\prec(i\to j)}} \phi.$$

This property to the major advantage of exact inference by message passing. Consider the task of computing the distribution over every variable $X \in \mathcal{X}$ in a clique tree. The naive approach would be to compute, using algorithm 2, the joint probability distribution β_i for each clique $\boldsymbol{C}_i$. However, there is a considerably more efficient alternative.

Consider adjacent cliques $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ in a clique tree. As long as the root clique is on the $\boldsymbol{C}_j$-side of the tree, precisely the same message is sent from i to j. The reasoning is analogous for j. Therefore, by computing the messages $\delta_{i\to j}$ and $\delta_{j\to i}$ for each edge (i, j) in the clique tree, it is possible to compute β_k for every $\boldsymbol{C}_k$. This can be accomplished by first performing an upward pass (algorithm 2) for any given root, and then performing a downward pass of messages from the root to every other node. The messages can then be used to compute the beliefs β_k for every $\boldsymbol{C}_k$.

Two adjacent cliques $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ are said to be calibrated if:

$$\sum_{\boldsymbol{C}_i-\boldsymbol{S}_{i,j}} \beta_i(\boldsymbol{C}_i) = \sum_{\boldsymbol{C}_j-\boldsymbol{S}_{i,j}} \beta_j(\boldsymbol{C}_j).$$

A clique tree is said to be calibrated when all of its pairs of adjacent cliques are calibrated. In a calibrated clique tree, the expression on either side is denominated sepset belief $\mu_{i,j}(\boldsymbol{S}_{i,j})$.

In a calibrated clique tree $\mathcal{T}$ for factors Φ, it can be shown that:

$$\prod_{\phi\in\Phi} \phi = \frac{\prod_{i\in V(\mathcal{T})} \beta_i(\boldsymbol{C}_i)}{\prod_{(i,j)\in E(\mathcal{T})} \mu_{i,j}(\boldsymbol{S}_{i,j})}.$$

Consider the problem of of performing inference several times using the same calibrated clique tree while increasing the number of observed variables. This is called incremental updating. Let $Z = z$ be the additional observation and $\boldsymbol{C}_i$ any clique that contains Z. Incremental updating can be accomplished by updating the previous belief using the rule $\beta_i(\boldsymbol{C}_i) \leftarrow \beta_i(\boldsymbol{C}_i)\mathbb{1}(Z = z)$, where $\mathbb{1}$ is an indicator function for $Z = z$, and sending the updated message to the other cliques that contain Z. Retracting evidence is not possible in this scheme.

It is also possible to perform approximate inference using cluster graphs. One example is the sum-product belief propagation algorithm for cluster graphs, algorithm 3, that tries to achieve calibration in cluster graphs that have the running intersection property.

Algorithm 3 Sum-product belief propagation for cluster graphs

Input: set of factors Φ over variables $\mathcal{X}$, cluster graph (with running intersection) $\mathcal{U}$ containing k cliques, function assigning factors to cliques α

Output: calibrated beliefs $\beta_i(\boldsymbol{C}_i)$ for every $1 \le i \le k$.

for each i in $1, \ldots, k$ **do**
  $\beta_i(\boldsymbol{C}_i) \leftarrow \psi_i(\boldsymbol{C}_i) \leftarrow \prod_{\{\phi\in\Phi|\alpha(\phi)=i\}} \phi$
end for
for each (i, j) in $E(\mathcal{U})$ **do**
  $\delta_{i\to j} \leftarrow \delta_{j\to i} \leftarrow 1$
end for
while beliefs are not calibrated **do**
  $(i, j) \leftarrow$ selected (possibly random) edge in $E(\mathcal{U})$
  $\delta_{i\to j} \leftarrow \sum_{C_i-\boldsymbol{S}_{i,j}} (\psi_i(\boldsymbol{C}_i) \prod_{k\in(\mathrm{Nb}_i-\{j\})} \delta_{k\to i})$
  $\beta_j(\boldsymbol{C}_j) \leftarrow \psi_j(\boldsymbol{C}_j) \prod_{k\in\mathrm{Nb}_j} \delta_{k\to j}$
end while

However, algorithm 3 is not guaranteed to converge. In general, a belief $\beta_i(\boldsymbol{C}_i)$ also does not represent the joint probability distribution over the variables $\boldsymbol{C}_i$, but only an approximation. The conditions for convergence and the

quality of the approximation are further detailed in [1]. Both are highly dependent on the structure of the cluster graph.

A Bethe cluster graph for a set of factors Φ over a set of variables $\mathcal{X}$ is a simple cluster graph with the running intersection property. In this cluster graph, each variable $X \in \mathcal{X}$ is represented by a cluster. Also, each factor $\phi_i \in \Phi$ is represented by a cluster $\boldsymbol{C}_i = \text{Scope}[\phi_I]$, which has an edge for every node representing each of its variables.

Consider the pairwise Markov networks for image denoising or segmentation introduced in section 6. Inference based on variable elimination or belief propagation in clique trees can be shown to be exponential on the number of variables. However, a Bethe cluster graph for the set of factors that define the Markov network is a very compact representation, and approximate inference is often the algorithm of choice.

9 Maximum a Posteriori estimation

Let $P(\mathcal{X})$ be a joint probability distribution. Finding an assignment x^* such that

$$x^* = \arg\max_{x \in \text{Val}(\mathcal{X})} P(x)$$

consists on the maximum a posteriori problem. This problem is extremely important in practical applications.

Given a Bayesian network $\mathcal{B}$ over $\mathcal{X}$ and a number τ, the task of deciding whether there exists an assignment x to $\mathcal{X}$ such that $P(x) > \tau$ is an NP-complete problem. Therefore, finding the maximum a posteriori assignment is an NP-hard problem.

Consider a Gibbs distribution parameterized by Φ. The assignment that maximizes the product of the factors in Φ also maximizes its division by the partition function.

Additionally, consider a joint probability distribution $P(\mathcal{X})$. Let $\mathcal{Y}$ and $\mathcal{E}$ be disjoint subsets of $\mathcal{X}$ and $e \in \text{Val}(\mathcal{E})$. Finding the maximum probability assignment for $P(\mathcal{Y}|e)$ is also equivalent to finding the maximum probability assignment for $P(\mathcal{Y}, e)$.

Let ϕ_1 and ϕ_2 be positive factors, and X a variable such that $X \notin \text{Scope}[\phi_1]$. Factor maximization has the following property:

$$\max_X(\phi_1\phi_2) = \phi_1 \max_X \phi_2$$

.

This leads to algorithm 4, max-product variable elimination, which is very similar to the variable elimination algorithm introduced in the previous section.

Algorithm 4 Max-product variable elimination

Input: set of factors Φ over variables $\mathcal{X}$, ordering $\prec$ over $\mathcal{X}$.
Output: maximum probability assignment $x^* = \arg\max_{x \in \text{Val}(\mathcal{X})} P(x)$

1: Let $X_1, \ldots, X_k$ be an ordering of $\mathcal{X}$ respecting $\prec$
2: **for** each i in $1, \ldots, k$ **do**
3: $\quad \Phi' \leftarrow \{\phi \in \Phi | X_i \in \text{Scope}[\Phi]\}$
4: $\quad \Phi'' \leftarrow \Phi - \Phi'$
5: $\quad \psi_i \leftarrow \prod_{\phi \in \Phi'} \phi$
6: $\quad \tau \leftarrow \max_{Z_i} \psi_i$
7: $\quad \Phi \leftarrow \Phi'' \cup \{\tau\}$
8: **end for**
9: **for** each i in $k, \ldots, 1$ **do**
10: $\quad u_i \leftarrow (x^*_{i+1}, \ldots, x^*_k)\langle \text{Scope}[\psi_i] - \{X_i\}\rangle$
11: $\quad x^*_i \leftarrow \arg\max_{x_i} \psi_i(x_i, u_i)$
12: **end for**

When compared to max-sum variable elimination, the main additions to the algorithm above are the steps for recovering an assignment after all variables are eliminated. Intuitively, this is accomplished by using the factors defined at line 5 to progressively discover the assignment to each variable in inverse order of elimination . In the end, the factor Φ has an empty scope and corresponds to the value of x^* in the product of the initial factors.

For numerical stability, this algorithm is usually implemented as the maximization of the sum of the logarithm of the original factors. The complexity analysis presented in the previous section for variable elimination is analogous for algorithm 4.

In pairwise Markov networks over binary variables, it is possible to find the most likely assignment using a polynomial time algorithm for finding minimum cuts in graphs.

Let V be a set of vertices and s and t two vertices not in V. Consider the graph $G = (V \cup \{s, t\}, E)$ and a cost function $c : E \to \mathbb{R}_{\geq 0}$. An s-t graph cut is a pair of sets of vertices (V_s, V_t) that partition the vertices in G such that $s \in V_s$ and $t \in V_t$.

The cost c of a cut (V_s, V_t) is defined as:

$$c(V_s, V_t) = \sum_{(v_s, v_t) \in E | v_s \in V_s, v_t \in V_t} c(v_s, v_t).$$

In other words, the cost of a cut is the sum of the costs of the edges that cross the partitions. A minimum cut is one that has minimum cost among all possible cuts.

Consider a pairwise Markov network $\mathcal{H} = (\mathcal{X}, \mathcal{E})$ over binary variables $\mathcal{X}$. We let $G = (V, E)$ be a directed graph containing one vertex v_i for each $X_i \in \mathcal{X}$. We also let this graph have special vertices s and t.

The next step is associating a cost to each edge in G in such a way that a minimum cut in G corresponds to a maximum probability assignment. This will be illustrated using a simplified problem.

Consider again the Markov network $\mathcal{H} = (\mathcal{X}, \mathcal{E})$. Suppose the energy function for this network can be written as:

$$E(x_1, \ldots, x_n) = \sum_i \epsilon_i(x_i) + \sum_{(X_i, X_j) \in \mathcal{E}} \epsilon_{i,j}(x_i, x_j).$$

Also, let $\epsilon_{i,j}(x_i, x_j) = \lambda_{i,j}$, if $x_i = x_j$ and 0 otherwise. Intuitively, $\lambda_{i,j}$ is an energy penalty for disagreeing neighbors. Additionally, consider that all the energy factors are non-negative and that, for every $X_i \in \mathcal{X}$, either $\epsilon_i(0) = 0$ or $\epsilon_i(1) = 0$.

The costs for the edges in G will now be defined. If $\epsilon_i(0) = 0$, then let $(s, v_i) \in E$ and let $c(s, v_i) = \epsilon_i(1)$. Otherwise, if $\epsilon_i(1) = 0$, then let $(s, t_i) \in E$ and let $c(t, v_i) = \epsilon_i(0)$. Intuitively, this will correspond to a penalty for assigning a variable to its least preferred value.

Additionally, for each pair $(X_i, X_j) \in \mathcal{E}$, let exist two edges (v_i, v_j) and (v_j, v_i) with cost $\lambda_{i,j}$. Intuitively, this will penalize cuts that separate variables in proportion to their affinity.

Given a cut (V_s, V_t) in G, the corresponding assignment x_i for each $X_i \in \mathcal{X}$ is defined as:

$$x_i = \begin{cases} 0 & \text{if } v_i \in V_s \\ 1 & \text{otherwise} \end{cases}$$

In words, we represent the assignment 0 using V_s and the assignment 1 using V_t.

It is possible to find the minimum cut in G in polynomial time. It is easy to show that this minimum cut corresponds to the maximum a posteriori assignment.

This reduction can be extended for arbitrary pairwise Markov networks over binary variables, as long as $\epsilon_{i,j}(1,1) + \epsilon_{i,j}(0,0) \leq \epsilon_{i,j}(0,1) + \epsilon_{i,j}(1,0)$ for every $(X_i, X_j) \in \mathcal{E}$. The reduction for the general case is more elaborate and can be found in [1].

The problem of finding a maximum probability assignment for pairwise Markov networks over n-ary variables is NP-hard. Approximations can be obtained using other techniques based on minimum cuts [1].

Another technique to perform maximum a posteriori estimation in general probabilistic graphical models is dual decomposition, which will not be presented here. In broad terms, it uses an optimization approach to divide the problem into tractable subproblems, which are motivated to agree on a maximum probability assignment.

10 Sampling Methods

Consider the task of computing $P(\mathbf{Y}|\mathbf{E} = e)$. Sampling is the process of generating full assignments to $\mathcal{X}$ in a way that is useful for estimating $P(\mathbf{Y}|\mathbf{E} = e)$. A sample $\mathcal{D} = \{\xi[1], \ldots, \xi[M]\}$ of $\mathcal{X}$ is a set composed of M elements in $\text{Val}(\mathcal{X})$.

Generating sample elements from a probability distribution P over a discrete random variable X such that $\text{Val}(X) = \{x_1, \ldots, x_k\}$ is a simple process. The interval $[0, 1]$ is partitioned into k intervals $I_l = [\sum_{i=1}^{l-1} P(X = x_i), \sum_{i=1}^{l} P(X = x_i))$ for each $l \in \{1, \ldots, k\}$. A (pseudo)random number generator, uniform in the interval $[0, 1)$, can be used to sample a value s. If $s \in I_l$, then x_l is sampled.

Forward sampling is a simple technique that can be applied to a Bayesian networks (Alg. 5). A topological ordering of the vertices $\mathcal{X}$ in a Bayesian network $\mathcal{B} = (\mathcal{X}, \mathcal{E})$ is a sequence $X_1, \ldots, X_k$ such that there is no edge $(X_j, X_i) \in \mathcal{E}$ if $j > i$. Every directed acyclic graph admits a topological ordering.

Algorithm 5 Forward sampling in Bayesian network

Input: Bayesian network $\mathcal{B}$ over variables $\mathcal{X}$.
Output: Sample element $x \in \text{Val}(\mathcal{X})$.
1: Let $X_1, \ldots, X_k$ be a topological ordering of $\mathcal{X}$
2: **for** each i in $1, \ldots, k$ **do**
3: $\quad x_i \leftarrow$ sample element from $P(X_i | (x_1, \ldots, x_{k-1})\langle \text{Pa}_{X_i} \rangle)$.
4: **end for**

Let $\mathcal{D} = \{\xi[1], \ldots, \xi[M]\}$ be a sample of $\mathcal{X}$ generated via algorithm 5. The estimate $\hat{P}_{\mathcal{D}}(y)$ of $P(y)$ given the sample $\mathcal{D}$, for $y \in \text{Val}(\mathcal{Y})$, $\mathcal{Y} \subseteq \mathcal{X}$, is defined as:

$$\hat{P}_{\mathcal{D}}(y) = \frac{1}{M} \sum_{m=1}^{M} \mathbb{1}[\xi[m]\langle Y \rangle = y].$$

In other words, the estimate $\hat{P}_{\mathcal{D}}(y)$ corresponds to the ratio between the number of sample elements where y occurs and M. It is important to note that the probability of any assignment to a subset of $\mathcal{X}$ can be computed from the same sample.

Let $y \in \text{Val}(\mathcal{Y})$ and $\mathcal{Y} \subseteq \mathcal{X}$ and $\mathcal{D}$ a sample of $\mathcal{X}$ containing M elements. Each sample element $\xi[i] \in \mathcal{D}$ has probability $P(y)$ of being consistent with y, in other words, of $\xi[i]\langle Y \rangle = y$. From this, important bounds can be derived about the probability $P_{\mathcal{D}}$ of obtaining a set $\mathcal{D}$ whose estimate is incorrect by a established margin defined by ϵ:

$$P_{\mathcal{D}}(\hat{P}_{\mathcal{D}}(y) \notin [P(y) - \epsilon, P(y) + \epsilon]) \leq 2e^{-2M\epsilon^2}$$
$$P_{\mathcal{D}}(\hat{P}_{\mathcal{D}}(y) \notin P(y)(1 \pm \epsilon)) \leq 2e^{-\frac{1}{3}MP(y)\epsilon^2}.$$

From the first equation, to obtain an estimate $\hat{P}_{\mathcal{D}}(y)$ within $\pm\epsilon$ of $P(y)$ with probability at least $1 - \delta$, the number of samples M needed is:

$$M \geq \frac{\ln(2) - \ln(\delta)}{2\epsilon^2}.$$

The analogous derivation from the second equation gives:

$$M \geq 3\frac{\ln(2) - \ln(\delta)}{P(y)\epsilon^2}.$$

Consider the task of estimating $P(\mathbf{Y} = y | \mathbf{E} = e)$. A simple way to adapt algorithm 5 is to *reject* (ignore) sample elements incompatible with e, and compute the estimate $\hat{P}_{\mathcal{D}}(\mathbf{Y} = y | \mathbf{E} = e)$ based only on the *accepted* elements. This idea is called rejection sampling, and is equivalent to estimating $P(\mathbf{Y} = y, \mathbf{E} = e)$ and $P(\mathbf{E} = e)$ to derive the desired estimate. The probability of obtaining elements compatible with the evidence e is precisely $P(\mathbf{E} = e)$, which may be low, leading to the rejection of a large number of sample elements.

Consider a Gibbs distribution P over factors Φ' and variables $\mathcal{X}$. Let $\mathbf{Y}$ and $\mathbf{E}$ be a partition of $\mathcal{X}$. Under some assumptions about P, Gibbs sampling (Alg. 6) is a process that generates a sequence of sample elements that can be used to estimate $P(\mathbf{Y} = y | \mathbf{E} = e)$. Let $\Phi = \{\phi[E = e] | \phi \in \Phi'\}$, and $y^{(0)}$ be an assignment to $\mathbf{Y}$. In the case of a Bayesian network, $y^{(0)}$ may be obtained by forward sampling after reducing each factor by the evidence, which in general is not the same as sampling from $P(\mathbf{Y} = y | \mathbf{E} = e)$. We let Y_{-j} denote $\mathbf{Y} - \{Y_j\}$, and $\Phi_Y = \{\phi \in \Phi | Y \in \text{Scope}[\phi]\}$.

In line 5, sampling is performed from the product of all the factors involving Y_i, reduced by the current assignment to all other variables in their scopes, divided by the marginalization of this product over Y_i. This division guarantees that $P(Y_i | y_{-i})$ is a probability distribution, and is, in general, called renormalization.

A Markov chain is a model that defines a next-state distribution for every state in $\text{Val}(\mathcal{X})$. This model can be seen as a directed graph of states with edges labeled by the probability of transitioning from one state to the other. Precisely, the transition model $\mathcal{T}$ of a Markov chain specifies the probability $\mathcal{T}(x \to x')$ of going from state x to state x', which applies whenever the chain is in state x.

Algorithm 6 Gibbs sampling

Input: Set of variables to be sampled $\mathbf{Y}$, set of (reduced) factors Φ, initial sample element $y^{(0)}$, number of time steps T.

Output: Sequence of sample elements $y^{(1)}, \ldots, y^{(T)}$.

1: Let $Y_1, \ldots, Y_n$ be an ordering of $\mathbf{Y}$
2: **for** each t in $1, \ldots, T$ **do**
3: $\quad y^{(t)} \leftarrow y^{(t-1)}$
4: $\quad$ **for** each i in $1, \ldots, n$ **do**
5: $\quad\quad y_i^{(t)} \leftarrow$ sample from $P(Y_i | y_{-i}^{(t)}) = \frac{\prod_{\phi \in \Phi_{Y_i}} \phi[y_{-j}]}{\sum_{Y_i} \prod_{\phi \in \Phi_{Y_i}} \phi[y_{-i}]}$
6: $\quad$ **end for**
7: **end for**

Given an initial probability distribution $P^{(0)}$ over the states in $\text{Val}(\mathcal{X})$ and a transition model $\mathcal{T}$, the sampling process that generates the sample element sequence $x^{(0)}, x^{(1)}, x^{(2)}, \ldots$ based on $\mathcal{T}(x^{(t)} \to x^{(t+1)})$ is called Markov chain Monte Carlo. It is possible to define a probability distribution $P^{(t+1)}$ over the state of this process at step $t+1$ as:

$$P^{(t+1)}(\mathcal{X}^{t+1} = x') = \sum_{x \in \text{Val}(\mathcal{X})} P^{(t)}(\mathcal{X}^t = x) \mathcal{T}(x \to x').$$

A probability distribution $\pi(\mathcal{X})$ is stationary for the transition model $\mathcal{T}$ if:

$$\pi(\mathcal{X} = x') = \sum_{x \in \text{Val}(\mathcal{X})} \pi(\mathcal{X} = x) \mathcal{T}(x \to x').$$

Consider the matrix A where $A_{i,j} = \mathcal{T}(x_j \to x_i)$. A stationary distribution can be represented by a vector $(\pi(x_1), \ldots, \pi(x_n))$, which is an eigenvector of A with eigenvalue 1.

In general, the Markov chain Monte Carlo process is not guaranteed to converge to a stationary distribution. More precisely, $P^{(t)}$ may never be equal to $\pi(\mathcal{X})$ for any t. There is also no guarantee that a stationary distribution is unique. The distribution obtained on convergence is also dependent on the initial distribution $P^{(0)}$.

A Markov chain is regular if there exists a number k, such that, for every $x, x' \in \text{Val}(\mathcal{X})$, the probability of getting from x to x' in *exactly* k steps is non-zero. A regular Markov chain over a finite set of states has a unique stationary distribution. Two conditions are sufficient, although not necessary, to ensure regularity: there is a positive probability path between every two states and a non-zero probability of self-transitions for every state.

Consider a set of transition models $\mathcal{T}_i$, called kernels, for each variable $X_i \in \mathcal{X}$ and let $\mathcal{X}_{-i} = \mathcal{X} - \{X_i\}$. Each model $\mathcal{T}_i$ in this set takes a state (x_i, x_{-i}) into a state (x_i', x_{-i}). In other words, each transition model may change the state of a single variable in $\mathcal{X}$. Given an initial state, these transition models may be applied in turn, following any criteria. After every transition model $\mathcal{T}_i$ is applied, the *aggregate step* can be interpreted as a step in a transition model $\mathcal{T}$ between every pair of states. If each kernel has a unique stationary distribution, then the aggregate transition model has a unique stationary distribution.

Gibbs sampling can be seen as a particular case of a multiple kernel transition model, where each transition model $\mathcal{T}_i$ is defined as:

$$\mathcal{T}((x_i, x_{-i}) \to (x_i', x_{-i})) = P(x_i' | x_{-i}).$$

It is possible to show that P is always a stationary distribution of this (multiple kernel) transition model. The converse is not generally true.

Beyond having one transition model for every variable, it is also possible to define kernels for disjoint sets that partition $\mathcal{X}$. This variant is called block Gibbs sampling. When the variables in a block are independent given all others, this makes the sampling process more efficient.

If all the factors in a Gibbs distribution P are positive, the Markov chain defined by Gibbs sampling is regular, and thus its unique stationary distribution is P. Importantly, it may take a very large number of steps to achieve this convergence, which is also called *mixing*.

In practice, several decisions must be made when using Markov chain Monte Carlo methods. One of them is the burn-in time T, the number of steps taken before we collect a sample from a chain. This time is highly dependent on the probabilities of transitioning between states.

Although it is possible to burn-in a Markov chain to generate every sample element, it is also possible to consider a sample of M elements obtained after burn-in. Although estimates made from this sample tend, on the limit, to be correct, the sample elements are not independent, which may lead to high variance. The idea of skipping sufficient sample elements to reduce this dependence may occur naturally, although it is provably worst than using all the sample elements obtained after burn-in.

It is very hard to establish whether a Markov chain has mixed. One common practice is running several Markov chains at the same time, and comparing their estimates.

One of the main advantages of Markov chain Monte Carlo methods is being independent of the probability of the evidence, since the factors are reduced before sampling. Given sufficient sample sizes, these methods can get arbitrarily close to the posterior. However, it is often challenging to create a Markov chain with good mixing properties.

Variable elimination, belief propagation and Markov chain Monte Carlo can be used to perform inference in unrolled template models. However, tractable inference often requires a careful examination of the structure of a model. For example, independencies can simplify queries immensely by removing the conditioning evidence. In practice, different methods are often combined to achieve tractable inference.

11 Decision Making

Decision theory is the study of decision making under uncertainty. An agent is a subject that makes decisions. Each possible outcome in a decision scenario is associated to a numerical value called utility, which encodes the agent's preferences.

A decision situation $\mathcal{D}$ is composed of the following elements: a set of outcomes $\mathcal{O}$, a set of possible actions $\mathcal{A}$, a probability distribution π_a for each $a \in \mathcal{A}$, which represents the probability of each outcome $o \in \mathcal{O}$ given the action a, and a utility function $U : \mathcal{O} \to \mathbb{R}$, which maps a utility to each outcome.

The expected utility $\mathrm{EU}[\mathcal{D}[a]]$ of an action a in a decision situation $\mathcal{D}$ is defined as:

$$\mathrm{EU}[\mathcal{D}[a]] = \sum_{o \in \mathcal{O}} \pi_a(o) U(o).$$

Decision theory builds on the assumption that the agent should choose an action which maximizes expected utility. This assumption requires that the magnitude of the utilities be representative of the agent's preferences, taking into consideration the multiple aspects of an outcome. An agent is rational when it always chooses the action with maximum expected utility. Judging the correctness of the actions taken by an agent generally requires knowing its utility function. However, some choices are contradictory for any utility function.

Suppose an observation $o \in \mathcal{O}$ can also be represented by an assignment $\mathrm{Val}(\mathbf{V})$ to a set of variables $\mathbf{V}$. In this case, $U : \mathrm{Val}(\mathbf{V}) \to \mathbb{R}$. It is often useful to write a utility function as the sum of subutility functions:

$$U(\mathbf{V}) = \sum_{i=1}^{k} U_i(\mathbf{Z}_i),$$

where $\mathbf{Z}_i \subseteq \mathbf{V}$ for every $0 \leq i \leq k$. This simplifies the enumeration of utilities for every possible assignment to $\mathbf{V}$.

An influence diagram $\mathcal{I}$ is a directed acyclic graph whose nodes correspond to elements of a set $\mathcal{Z} = \mathcal{X} \cup \mathcal{D} \cup \mathcal{U}$, where the elements in $\mathcal{U}$ have no descendants in $\mathcal{I}$. The three sets on the right side of the equation contain, respectively, chance variables, decision variables and utility variables. A chance variable V is a random variable associated with a conditional probability distribution $P(V|\operatorname{Pa}_V)$. A chance variable in an influence diagram can be interpreted in the same way as a variable in a Bayesian network. A decision variable D may be associated with a conditional probability distribution $\delta_D(D, \operatorname{Pa}_D) = P(D|\operatorname{Pa}_D)$, also called the decision rule for D, which represents the probability of a given decision (action) by the agent given the parents of the variable. A utility variable $V \in \mathcal{U}$ is associated with a factor $U_V(\operatorname{Pa}_V)$ that gives a utility to each assignment to the parents of the variable. When an influence diagram is represented pictorially, each chance variable is inscribed inside an oval, each decision variable in a rectangle and each utility variable in a rhombus.

An influence diagram $\mathcal{I}$ is independent of the choice of the decision rules for its decision variables, which collectively compose a strategy $\sigma = \{\delta_D(D, \operatorname{Pa}_D) | D \in \mathcal{D}\}$ of the agent. An influence diagram with strategy σ is denoted by $\mathcal{I}[\sigma]$. The acyclicity of the graph imposes a partial order over the decisions. However, the dependence between decisions must be explicitly stated in the diagram.

The expected utility of an influence diagram $\mathcal{I}[\sigma]$ with strategy σ is defined as:

$$\mathrm{EU}[\mathcal{I}[\sigma]] = \sum_{\mathcal{X}\cup\mathcal{D}} [(\prod_{X\in\mathcal{X}} P(X|\,\mathrm{Pa}_X))(\prod_{D\in\mathcal{D}} \delta_D(D,\mathrm{Pa}_D))(\sum_{V\in\mathcal{U}} U_V(\mathrm{Pa}_V))].$$

This corresponds to computing the sum of the probability of each joint assignment to the variables in $\mathcal{X}\cup\mathcal{D}$ multiplied by its utility. The utility of an assignment is defined as the sum of the utility of that assignment for each utility factor. Notice that the summation over utility factors is not a marginalization over the variables in $\mathcal{U}$, which are not in the scope of any factor, but factor addition.

There are two simple ways to compute the expected utility of $\mathcal{I}[\sigma]$. One is to collapse every utility factor into a larger utility factor $U(\mathcal{Y}) = \sum_{V\in\mathcal{U}} U_V(\mathrm{Pa}_V)$, where $\mathcal{Y}\subseteq\mathcal{X}\cup\mathcal{D}$. The necessary computation becomes:

$$\mathrm{EU}[\mathcal{I}[\sigma]] = \sum_{\mathcal{X}\cup\mathcal{D}} [(\prod_{X\in\mathcal{X}} P(X|\,\mathrm{Pa}_X))(\prod_{D\in\mathcal{D}} \delta_D(D,\mathrm{Pa}_D))U(\mathcal{Y})].$$

This is a simple marginalization over a product of factors, which can be solved by sum-product variable elimination, presented in section 7. Since the scope of U may be very large, this is not always the best alternative.

Another equally simple solution is to use the linearity of the expected utility of $\mathcal{I}[\sigma]$:

$$\mathrm{EU}[\mathcal{I}[\sigma]] = \sum_{V\in\mathcal{U}} \sum_{\mathcal{X}\cup\mathcal{D}} [(\prod_{X\in\mathcal{X}} P(X|\,\mathrm{Pa}_X))(\prod_{D\in\mathcal{D}} \delta_D(D,\mathrm{Pa}_D))U_V(\mathrm{Pa}_V)].$$

This corresponds to computing the expected utility independently for each utility variable, which can be easily accomplished with sum-product variable elimination, and summing these expected utilities. Notice that the outermost summation is not a marginalization over variables in $\mathcal{U}$, but a sum of real numbers. This approach avoids defining a large global utility factor, however, it also causes work replication.

The maximum expected utility considering all possible strategies for $\mathcal{I}$ is denoted by $\mathrm{MEU}[\mathcal{I}]$. A strategy σ^* is optimal if $\mathrm{EU}[\mathcal{I}[\sigma^*]] = \mathrm{MEU}[\mathcal{I}]$.

Consider an influence diagram $\mathcal{I}$ with a single decision variable D . For any strategy $\sigma = \{\delta_D\}$, the expected utility can also be written as:

$$\begin{aligned}\mathrm{EU}[\mathcal{I}[\sigma]] &= \sum_{\{D\}\cup\mathrm{Pa}_D} \delta_D(D,\mathrm{Pa}_D) \sum_{\mathcal{X}-\mathrm{Pa}_D} [(\prod_{X\in\mathcal{X}} P(X|\,\mathrm{Pa}_X))(\sum_{V\in\mathcal{U}} U_V(\mathrm{Pa}_V))] \\ &= \sum_{\{D\}\cup\mathrm{Pa}_D} \delta_D(D,\mathrm{Pa}_D)\mu_{-D}(D,\mathrm{Pa}_D).\end{aligned}$$

In this case, defining a decision rule δ_D^* that maximizes expected utility is very simple:

$$\delta_D^*(d,\mathrm{pa}_D) = \begin{cases} 1 & \text{if } d = \arg\max_{d'\in\mathrm{Val}(D)} \mu_{-D}(d',\mathrm{pa}_D) \\ 0 & \text{otherwise}\end{cases}$$

This assumes that ties are broken consistently, such that a single decision is given maximum probability for each assignment to Pa_D. Intuitively, this corresponds to a deterministic decision rule that always chooses the decision with maximum expected utility for each situation. The same reasoning can be extended to more than one decision variable. However, there are more efficient ways of performing this maximization task, which are detailed in [1].

Let $\mathcal{I}$ be an influence diagram, X a chance variable and D a decision variable, such that there is no path from D to X. Let $\mathcal{I}'$ be an influence diagram which is equal to $\mathcal{I}$ plus an edge from X to D and from X to every decision that follows D in any topological ordering of the decision variables. The value of information X at D is defined as $\mathrm{VPI}_{\mathcal{I}}(D|X) = \mathrm{MEU}[\mathcal{I}'] - \mathrm{MEU}[\mathcal{I}]$. Intuitively, the value of perfect information indicates how much expected utility would be gained, under optimum decision-making, if the value of X were known when the decision D is made. The value of perfect information is always non-negative, and is zero if knowing the value of X does not change the actions selected by the agent.

An alternative to evaluate the desirability of observing a chance variable is encoding the cost of such observation in decision and utility factors. In this case, the optimum strategy already defines which chance variables should be observed.

12 Parameter Estimation

In the context of graphical models, learning is the process of building models based on data. Although models may be created by experts, there are three main reasons to employ learning. The correct model for the underlying process may be too large to be described exhaustively, there may be no explicit knowledge of how to model the process or the model may need to adapt to changes in the underlying process.

More concretely, consider the problem of defining a graphical model for a probability distribution $P^*(\mathcal{X})$, having access only to a sample $\mathcal{D} = \{\xi[1], \ldots, \xi[M]\}$ of sample elements in $\text{Val}(\mathcal{X})$. Consider each assignment $\xi[i] \in \mathcal{D}$ as an assignment to a set of random variables $\mathcal{X}[i]$, that together follow a joint distribution P.

The sample elements in $\mathcal{D}$ are independent if, for every $i, j \in \{1, \ldots, M\}$:

$$P(\mathcal{X}[i]|\mathcal{X}[j]) = P(\mathcal{X}[i])P(\mathcal{X}[j]).$$

They are identically distributed according to P^* if, for every $i, j \in \{1, \ldots, M\}$ and assignment $\xi \in \text{Val}(\mathcal{X})$:

$$P(\mathcal{X}[i] = \xi) = P(\mathcal{X}[j] = \xi) = P^*(\mathcal{X} = \xi).$$

All data sets considered in this chapter are composed of independent, identically distributed (*IID*) elements.

The probabilistic models presented so far can be described by factors. Parameter learning corresponds to learning the value given by each factor to each assignment in its domain. Structure learning corresponds to decomposing the joint probability distribution P^* as a product of factors. This section focuses on parameter learning.

12.1 Maximum Likelihood Estimation

Intuitively, maximum likelihood estimation corresponds to finding the parameters that would produce the observed sample with maximum probability.

Consider a binary random variable X and a sample $\mathcal{D} = \{x[1], \ldots, x[M]\}$ of independent, identically distributed sample elements in $\text{Val}(X)$. Let $P(X : \theta)$ denote a probability distribution over X *parameterized* by θ, in such a way that $P(X = 0 : \theta) = \theta$ and $P(X = 1 : \theta) = 1 - \theta$. Clearly, the possible values for θ are in $\Theta = [0, 1]$. The set of parameters under consideration Θ is usually called parameter space.

Based on our assumptions:

$$P(x[1], \ldots, x[M] : \theta) = \prod_{i=1}^{M} P(X[i] = x[i] : \theta) = \theta^{M[X=0]}(1 - \theta)^{M[X=1]},$$

where $M[X = x] = \sum_{i=1}^{M} \mathbb{1}\{x[i] = x\}$ denotes the number of sample elements in $\mathcal{D}$ equal to x. The likelihood of the data $\mathcal{D}$ under the parameter θ is defined as $L(\theta : \mathcal{D}) = P(x[1], \ldots, x[M] : \theta)$.

Consider the task of finding a $\hat{\theta}$ such that $L(\hat{\theta} : \mathcal{D}) = \max_{\theta \in \Theta} L(\theta : \mathcal{D})$. This is equivalent to finding a $\hat{\theta}$ that maximizes the log-likelihood $\ell(\theta : \mathcal{D}) = \log L(\theta : \mathcal{D})$. Notably:

$$\ell(\theta : \mathcal{D}) = \log[\theta^{M[X=0]}(1 - \theta)^{M[X=1]}] = M[X = 0] \log \theta + M[X = 1] \log(1 - \theta).$$

If $\hat{\theta}$ is a global maximum of $\ell(\theta : \mathcal{D})$, then $\ell'(\theta : \mathcal{D}) = 0$. Using calculus:

$$\ell'(\theta : \mathcal{D}) = \frac{M[X = 0]}{\theta} + \frac{M[X = 1]}{1 - \theta}.$$

Therefore, if a global maximum $\hat{\theta}$ exists, $\hat{\theta} = \frac{M[X=0]}{M[X=0]+M[X=1]}$. Since $\ell''(\theta : \mathcal{D}) < 0$ for any $\theta \in (0, 1)$, $\hat{\theta}$ is guaranteed to be a global maximum. As could be expected, $\hat{\theta}$ corresponds precisely to the relative frequency of occurrences of $X = 0$ in the sample $\mathcal{D}$. The parameter $\hat{\theta}$ is the maximum likelihood estimate.

As another example of parameterized model, consider a discrete random variable X with $\text{Val}(X) = \{1, \ldots, k\}$. In this case, the most natural parameter space consists on $\Theta = \{\theta \in [0, 1]^k | \sum_{i=1}^{k} \theta_i = 1\}$ and $P(X = x : \theta) = \theta_x$. This is also called a multinomial distribution model.

As yet another example, consider a continuous random variable X that can take any real value. Assuming that X follows a Gaussian distribution, $\Theta = \mathbb{R} \times \mathbb{R}^+$ and for a given $\theta \in \Theta$, such that $\theta = (\mu, \sigma)$, the probability density function P can be written as:

$$P(x : \theta) = \frac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi\sigma}}.$$

Given a set of random variables $\mathcal{X}$, a function $\tau : \text{Val}(\mathcal{X}) \to \mathbb{R}^l$, for an $l \in \mathbb{N}$, is a sufficient statistic if, for any two data sets $\mathcal{D}$ and $\mathcal{D}'$ and any $\theta \in \Theta$:

$$\sum_{i=1}^{M} \tau(\xi[i]) = \sum_{i=1}^{M'} \tau(\xi'[i]) \implies L(\theta : \mathcal{D}) = L(\theta : \mathcal{D}').$$

In words, when the sum of the sufficient statistic of all assignments in two distinct datasets is the same, the likelihood of the two datasets under any parameter θ is the same. This sum is often called the sufficient statistics (plural) of the data.

In the case of a multinomial distribution model for a random variable X with $\text{Val}(X) = \{1, \ldots, k\}$, $\tau(x)$ is a k-dimensional vector which is zero in every element except for the x-th, which is one. The sufficient statistics in this case is the vector $(M[1], \ldots, M[k])$, which is all that is needed to compute the likelihood for a given θ.

In the case of a Gaussian model, the sufficient statistic is $\tau(x) = (1, x, x^2)$, and the sufficient statistics is the vector $(M, \sum_i^M x[i], \sum_i^M x[i]^2)$. It is also the case that this vector is sufficient to compute the likelihood of any θ.

The task of maximum likelihood estimation can easily be generalized to involve a set of random variables $\mathcal{X}$, given a data set $\mathcal{D} = \{\xi[1], \ldots, \xi[M]\}$ of *IID* sample elements in $\text{Val}(\mathcal{X})$. A parameter space Θ contains all parameters under consideration, and, for any $\theta \in \Theta$:

$$\sum_{\xi \in \text{Val}(\mathcal{X})} P(\xi : \theta) = 1.$$

A similar condition should hold for distributions over continuous random variables. The likelihood function in the general case is defined as:

$$L(\theta : \mathcal{D}) = \prod_{m=1}^{M} P(\xi[m] : \theta).$$

Maximum likelihood estimation is the problem of finding a $\hat{\theta} \in \Theta$ such that $L(\hat{\theta} : \mathcal{D}) = \max_{\theta \in \Theta} L(\theta : \mathcal{D})$

In the case of a multinomial model for a random variable X with $\text{Val}(X) = \{1, \ldots, k\}$, $\hat{\theta} = (\frac{M[1]}{M}, \ldots, \frac{M[k]}{M})$, as could be intuitively expected.

In the case of a Gaussian model, $\hat{\theta} = (\hat{\mu}, \hat{\sigma})$, where $\hat{\mu}$ is the empirical mean (mean value in the data) and $\hat{\sigma}$ is the empirical standard deviation.

Consider a set $\mathcal{X} = \{X_i, \ldots, X_n\}$ of discrete random variables. Let $P^*(\mathcal{X})$ be a joint probability distribution represented by a Bayesian network, and $\mathcal{D} = \{\xi[1], \ldots, \xi[M]\}$ a sample of independent assignments to $\mathcal{X}$ distributed according to P^*. A parameter θ for this distribution may be defined as the set of $\theta_{X=x|\,\text{pa}_X} = P(X = x|\,\text{Pa}_X = \text{pa}_X : \theta)$ for every $X \in \mathcal{X}$, $x \in \text{Val}(X)$ and $\text{pa}_X \in \text{Pa}_X$. Naturally, $\sum_x \theta_{X=x|\,\text{pa}_X}$ must be equal to 1 for every $X \in \mathcal{X}$ and $\text{pa}_X \in \text{Val}(\text{Pa}_X)$. The parameter space Θ consists on the set of all θ that satisfy this constraint.

Under these definitions, the likelihood $L(\theta : \mathcal{D})$ can be written as:

$$L(\theta : \mathcal{D}) = \prod_{m=1}^{M} P(\xi[m] : \theta) = \prod_{m=1}^{M} \prod_{i=1}^{n} P(x_i[m]|\,\text{pa}_{X_i}[m] : \theta),$$

where $x_i[m] = \xi[m]\langle X_i \rangle$, and $\text{pa}_{X_i}[m] = \xi[m]\langle \text{Pa}_{X_i} \rangle$.

Consider the set of parameters $\theta_{X|\,\text{pa}_X} = \{\theta_{X=1|\,\text{pa}_X}, \ldots, \theta_{X=k|\,\text{pa}_X}\}$ of probabilities attributed to each assignment to $x \in \text{Val}(X)$ given a particular assignment pa_X to the parents of X, and let $\theta_{X|\,\text{Pa}_X} = \cup_{\text{pa}_X} \theta_{X|\,\text{pa}_X}$. It follows that the likelihood can be decomposed into the product of *local* likelihood functions:

$$L(\theta : \mathcal{D}) = \prod_{i=1}^{n} \left[\prod_{m=1}^{M} P(x_i[m]|pa_{X_i}[m] : \theta_{X_i|\,\text{pa}_{X_i}[m]}) \right] = \prod_{i=1}^{n} L_i(\theta_{X_i|\,\text{Pa}_{X_i}} : \mathcal{D}).$$

In words, the outermost product is between terms that depend on completely disjoint sets of parameters. Therefore, each innermost product (which involves only parameters in $\theta_{X_i|\,\text{Pa}_{X_i}}$) corresponds to an individual conditional probability distribution in the network. It follows that maximum likelihood estimation can be performed independently. This result is independent of how the conditional probability distributions are represented, as long as parameters are not shared between them.

As an example, consider the parameterization of a conditional probability distribution $P(X|\,\text{Pa}_X)$ for a variable X in a Bayesian network. For each assignment $x \in \text{Val}(X)$ and $u \in \text{Val}(\text{Pa}_X)$, there will be a parameter $\theta_{x|u} = P(x|u : \theta)$. The local likelihood $L_X(\theta_{X|U} : \mathcal{D})$ can be written as:

$$L_X(\theta_{X|U} : \mathcal{D}) = \prod_{m=1}^{M} \theta_{x[m]|u[m]} = \prod_{u \in \mathrm{Val}(\mathrm{Pa}_X)} \prod_{x \in \mathrm{Val}(X)} \theta_{x|u}^{M[x,u]}.$$

Consider the task of finding the parameters in $\hat{\theta}_{X|U}$ that maximize the local likelihood $L_X(\theta_{X|U} : \mathcal{D})$ of the data, under the obvious constraint that, for all $u \in \mathrm{Val}(\mathrm{Pa}_X)$, $\sum_x \theta_{x|u} = 1$. It is possible to show that:

$$\hat{\theta}_{x|u} = \frac{M[x,u]}{M[u]}.$$

It is very important to notice how a large number of samples may be required to perform estimation when $|\mathrm{Val}(\mathrm{Pa}_X)|$ is large.. This phenomenon is called data fragmentation, and is one of the biggest problems with learning Bayesian networks from data. This also highlights the importance of an appropriate network structure.

Naive Bayes, already presented in this text, can be seen as a particular instance of a Bayesian classifier that assumes a structure for a Bayesian network and performs maximum likelihood estimation of parameters.

12.2 Bayesian Parameter Estimation

A different approach to parameter learning is Bayesian estimation, which allows for the integration of previous knowledge about the parameters encoded as a probability distribution.

As an initial example, consider again a binary random variable X and *IID* sample elements in $\mathcal{D} = \{x[1], \ldots, x[M]\}$. Consider also a continuous random variable $\boldsymbol{\theta}$ such that $\mathrm{Val}(\boldsymbol{\theta}) = [0, 1]$. Consider a joint probability distribution over the variables $X[1], \ldots, X[m]$ and $\boldsymbol{\theta}$, and let θ denote an assignment to $\boldsymbol{\theta}$.

Let $P(x[i] = 0|\theta) = \theta$ for every $x[i] \in \mathcal{D}$. This parameterization is equivalent to the one we denoted by $P(x[i] = 0 : \theta)$ in the first example of the previous subsection. Based on our assumptions about P:

$$P(x[1], \ldots, x[m], \theta) = P(\theta) \prod_{i=1}^{M} P(x[i]|\theta).$$

Clearly, a *prior* probability distribution $P(\boldsymbol{\theta})$ over the parameters is required to compute the joint distribution using the equation above.

Notice that the joint distribution $P(X[1], \ldots, X[M], \boldsymbol{\theta})$ can be interpreted as a Bayesian network where $\boldsymbol{\theta}$ has an outgoing edge to each $X[i]$. Therefore, $X[i]$ is independent of $X[j]$ for every $i \neq j$ only when given $\boldsymbol{\theta}$.

In the Bayesian approach, the likelihood function is defined as $L(\theta : \mathcal{D}) = \prod_{i=1}^{M} P(x[i]|\theta)$. However, this approach is not restricted to finding the maximum likelihood parameter $\hat{\theta}$.

Different queries can be made about the joint probability distribution $P(X[1], \ldots, X[M], \boldsymbol{\theta})$. For example, it may be interesting to compute the posterior distribution:

$$P(\boldsymbol{\theta}|x[1], \ldots, x[M]) = \frac{P(x[1], \ldots, x[M], \boldsymbol{\theta})}{P(x[1], \ldots, x[M])}.$$

Based on this conditional distribution, it is possible to find the maximum a posteriori $\hat{\theta}$.

Prediction can be performed by introducing an additional variable $X[M+1]$ into the joint probability distribution:

$$\begin{aligned} P(x[m+1]|x[1], \ldots, x[m]) &= \int_0^1 P(x[m+1], \theta|x[1], \ldots, x[m])\ d\theta \\ &= \int_0^1 P(x[m+1]|\theta) P(\theta|x[1], \ldots, x[m])\ d\theta. \end{aligned}$$

Based on the equation above, the probability of observing $x[m+1]$ can be interpreted as the average probability of observing $x[m+1]$ under each parameter θ, weighted by the probability of that parameter given the previously seen data.

Therefore, in the Bayesian approach, parameter learning can be seen as a particular case of inference.

As a more general example, consider the task of performing Bayesian estimation for a multinomial distribution model. Let X be a discrete random variable and $\mathrm{Val}(X) = \{1, \ldots, k\}$. Also let $\mathcal{D} = \{x[1], \ldots, x[M\}$ contain *IID* sample elements of the underlying distribution $P^*(X)$. This estimation task corresponds to modeling a joint distribution $P(x[1], \ldots, x[M], \boldsymbol{\theta})$, where $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k\}$ is a set of continuous random variables that parameterize a

distribution over X. For every $\boldsymbol{\theta}_i \in \boldsymbol{\theta}$, $\text{Val}(\boldsymbol{\theta}_i) = [0, 1]$. Analogously to the previous example, $P(X = x|\theta) = \theta_x$. We denote by $\Theta \subseteq \text{Val}(\boldsymbol{\theta})$ the set of valid joint assignments to the variables in $\boldsymbol{\theta}$. In this case, $\Theta = \{\theta \in \text{Val}(\boldsymbol{\theta}) | \sum_i \theta_i = 1\}$. Naturally:

$$P(x[1], \ldots, x[m], \theta) = P(\mathcal{D}, \theta) = P(\mathcal{D}|\theta)P(\theta) = L(\theta : \mathcal{D})P(\theta).$$

Considering our assumptions:

$$L(\theta : \mathcal{D}) = \prod_{i=1}^{k} \theta_i^{M[i]}.$$

Thus, the definition of $P(\boldsymbol{\theta})$ completes the model. We will consider a Dirichlet distribution $\text{Dirichlet}(\alpha_1, \ldots, \alpha_k)$ over $\boldsymbol{\theta}$. Using the notation introduced previously, a Dirichlet distribution can be seen as a distribution over a set of continuous random variables, each taking values in the interval $[0, 1]$. Each positive real number α_i is a *hyperparameter* (a parameter about a parameter of the model) of the distribution. If $\boldsymbol{\theta} \sim \text{Dirichlet}(\alpha_1, \ldots, \alpha_k)$, then:

$$P(\boldsymbol{\theta}) \propto \prod_{i=1}^{k} \theta_i^{\alpha_i - 1},$$

Notice how the prior distribution has some similarity to the likelihood function. The reason for using a Dirichlet prior derives from the following important property regarding the posterior over $\boldsymbol{\theta}$.

If $P(\boldsymbol{\theta})$ is $\text{Dirichlet}(\alpha_1, \ldots, \alpha_k)$, then $P(\boldsymbol{\theta}|\mathcal{D})$ is $\text{Dirichlet}(\alpha_1 + M[1], \ldots, \alpha_k + M[k])$. Therefore, by defining the Dirichlet hyperparameters, the posterior can be concisely expressed given the sufficient statistics of a multinomial distribution model. This provides a simple way of representing the beliefs over the parameters after observing the data, and can be used to perform queries about the parameters, as already shown.

More generally, a family of priors of the form $P(\boldsymbol{\theta} : \alpha)$ is *conjugate* to a particular model $P(\xi|\boldsymbol{\theta})$ if, for any data $\mathcal{D}$ of *IID* sample elements from $P(\xi|\boldsymbol{\theta})$ and any valid choice of hyperparameters α, there are valid hyperparameters α' such that:

$$P(\boldsymbol{\theta} : \alpha') \propto P(\mathcal{D}|\boldsymbol{\theta})P(\boldsymbol{\theta} : \alpha).$$

Consider the task of predicting a new observation $x[M + 1]$. As before, this can be modeled as:

$$\begin{aligned} P(x[M+1] = x|\mathcal{D}) &= \int_{\Theta} P(x[M+1] = x, \theta|\mathcal{D}) \; d\theta \\ &= \int_{\Theta} P(x[M+1] = x|\theta)P(\theta|\mathcal{D}) \; d\theta \\ &= \int_{\Theta} \theta_x P(\theta|\mathcal{D}) \; d\theta \\ &= \mathbb{E}_{P(\theta|\mathcal{D})}[\theta_x]. \end{aligned}$$

Consider $\boldsymbol{\theta} \sim \text{Dirichlet}(\alpha_1, \ldots, \alpha_k)$. It can be shown that $\mathbb{E}_{P(\boldsymbol{\theta})}[\theta_x] = \frac{\alpha_x}{\alpha}$, where $\alpha = \sum_i \alpha_i$. Because $P(\boldsymbol{\theta}|\mathcal{D})$ is $\text{Dirichlet}(\alpha_1 + M[1], \ldots, \alpha_k + M[k])$, it follows that:

$$P(x[M+1] = x|\mathcal{D}) = \frac{\alpha_x + M[x]}{\alpha + M}.$$

Intuitively, α can be interpreted as the number of virtual observations that represent previous knowledge, and $\frac{\alpha_x}{\alpha}$ as the ratio of virtual occurrences of x. The identity above can be rewritten as:

$$P(x[M+1] = x|\mathcal{D}) = \frac{\alpha}{(\alpha + M)} \frac{\alpha_x}{\alpha} + \frac{M}{(\alpha + M)} \frac{M[k]}{M}$$

Therefore, the probability of x given the data can be interpreted as an average between the virtual frequency of x and the observed frequency of x, weighted (respectively) by the ratio between the number of virtual (real) observations and total number of observations. When $M \to \infty$, this is equivalent to prediction based on maximum likelihood estimation. The hyperparameter α represents the *strength* of the previous knowledge. It is important to notice that the prior distribution is *implicit* in prediction based on maximum likelihood estimation.

We now consider the case of Bayesian parameter estimation for Bayesian networks. Consider the set $\mathcal{D} = \{\xi[1], \ldots, \xi[M]\}$ of *IID* sample elements from the distribution P^* over the set of variables $\mathcal{X} = \{X_1, \ldots, X_n\}$. Using an analogous notation to that presented in the previous subsection, this tasks consists on defining a joint probability distribution $P(\xi[1], \ldots, \xi[M], \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the union of all $\boldsymbol{\theta}_{X|Pa_X}$. Each $\boldsymbol{\theta}_{X|Pa_X}$ is a set of continuous random variables, and an assignment in $\text{Val}(\boldsymbol{\theta}_{X|Pa_X})$ defines $P(x|pa_X, \theta_{X|Pa_X})$ for every possible assignment in $\text{Val}(X, Pa_X)$. An assignment to the variables in $\boldsymbol{\theta}$, denoted by θ, defines $P(\xi|\theta)$ for every $\xi \in \text{Val}(\mathcal{X})$.

The likelihood can be written as:

$$\begin{aligned} P(\mathcal{D}|\theta) &= P(\xi[1], \ldots, \xi[M]|\theta) \\ &= \prod_{m=1}^{M} \prod_{i=1}^{n} P(x_i[m]|\operatorname{pa}_{X_i}, \theta_{X_i|\operatorname{Pa}_{X_i}}) \\ &= \prod_{i=1}^{n} L_{X_i}(\theta_{X_i|\operatorname{Pa}_{X_i}} : \mathcal{D}), \end{aligned}$$

where $x_i[m] = \xi[m]\langle X_i \rangle$, and $\operatorname{pa}_{X_i}[m] = \xi[m]\langle \operatorname{Pa}_{X_i} \rangle$. This is analogous to the decomposition based on local likelihoods presented previously.

Assuming once again that the parameters for different conditional probability distributions are independent:

$$P(\theta) = \prod_{i=1}^{n} P(\theta_{X_i|\operatorname{Pa}_{X_i}}).$$

The combination of these results gives, for every $\theta \in \text{Val}(\boldsymbol{\theta})$:

$$P(\theta|\mathcal{D}) = \prod_{i=1}^{n} \frac{L_{X_i}(\theta_{X_i|\operatorname{Pa}_{X_i}} : \mathcal{D}) P(\theta_{X_i|\operatorname{Pa}_{X_i}})}{P(\mathcal{D})} = \prod_{i=1}^{n} P(\theta_{X_i|\operatorname{Pa}_{X_i}}|\mathcal{D}).$$

Therefore, the global posterior is the product of local posteriors.

In the prediction task, the probability of a new assignment also relates to the product of local probabilities of that assignment:

$$\begin{aligned} P(X_1[M+1], \ldots, X_n[M+1]|\mathcal{D}) &= \int_{\Theta} P(X_1[M+1], \ldots, X_n[M+1], \theta|\mathcal{D})\ d\theta \\ &= \int_{\Theta} P(X_1[M+1], \ldots, X_n[M+1]|\theta) P(\theta|\mathcal{D})\ d\theta \\ &= \int_{\Theta} \prod_{i=1}^{n} P(X_i[M+1]|\operatorname{Pa} X_i[M+1], \theta) P(\theta|\mathcal{D})\ d\theta \\ = \prod_{i=1}^{n} \int_{\Theta_{X_i|\operatorname{Pa}_{X_i}}} & P(X_i[M+1]|\operatorname{Pa} X_i[M+1], \theta_{X_i|\operatorname{Pa}_{X_i}}) P(\theta_{X_i|\operatorname{Pa}_{X_i}}|\mathcal{D})\ d\theta_{X_i|\operatorname{Pa}_{X_i}}. \end{aligned}$$

In conclusion, computing posteriors over $\boldsymbol{\theta}$ and performing prediction depends on computing local posteriors $P(\boldsymbol{\theta}_{X_i|\operatorname{Pa}_{X_i}}|\mathcal{D})$.

We now define the local prior distributions $P(\boldsymbol{\theta}_{X_i|\operatorname{Pa}_{X_i}})$ over the parameters, which lead naturally to the local posteriors $P(\boldsymbol{\theta}_{X_i|\operatorname{Pa}_{X_i}}|\mathcal{D})$.

For each random variable $X \in \mathcal{X}$ and each each $\operatorname{pa}_X \in \operatorname{Pa}_X$, with $\text{Val}(X) = \{1, \ldots, k\}$, we define a Dirichlet distribution $\text{Dirichlet}(\alpha_{X=1|\operatorname{pa}_X}, \ldots, \alpha_{X=k|\operatorname{pa}_X})$ as $P(\boldsymbol{\theta}_{X|\operatorname{pa}_X})$. In other words, we define a distinct multinomial model for each assignment to the parents of X.

From the fact that the parameterizations given distinct assignment to the parents are independent:

$$P(\boldsymbol{\theta}_{X|\operatorname{Pa}_X}|\mathcal{D}) = \prod_{\operatorname{pa}_x} P(\boldsymbol{\theta}_{X|\operatorname{pa}_X}|\mathcal{D}).$$

Therefore, the local posterior is the product of Dirichlet posteriors. All of these assumptions lead to natural local (and, by consequence, global) prediction:

$$P(x[M+1]|\operatorname{pa}_X[M+1],\mathcal{D}) = \frac{\alpha_{X=x[m+1]|\operatorname{pa}_X[M+1]} + M[X = x[m+1], \operatorname{pa}_X[M+1]]}{\sum_i \alpha_{X=i|\operatorname{pa}_X[M+1]} + M[X = i, \operatorname{pa}_X[M+1]]}.$$

Defining priors over a Bayesian network is a very important task. Given a parameter α, a uniform BDe prior for a Bayesian network is one such that:

$$\alpha_{X=x|\operatorname{Pa}_X=\operatorname{pa}_X} = \frac{\alpha}{|\operatorname{Val}(X, \operatorname{Pa}_X)|},$$

for every $X \in \mathcal{X}$ and $(x, \operatorname{pa}_X) \in \operatorname{Val}(X, \operatorname{Pa}_X)$. This is a principled way of representing ignorance about the parameters.

12.3 Parameter estimation in Markov networks

The previous sections have focused on parameter estimation for Bayesian networks. This section introduces parameter estimation for Markov networks.

Consider a Markov network defined by a set of positive factors $\Phi = \{\phi_1(D_1), \ldots, \phi_K(D_K)\}$ over a set of variables $\mathcal{X}$. By definition:

$$P(\mathcal{X}) = \frac{\prod_{i=1}^K \phi_i(D_i)}{Z},$$

where:

$$Z = \sum_{\mathcal{X}} \prod_{j=1}^K \phi_j(D_j)$$

Another way of representing the same network is:

$$P(\mathcal{X}) = \frac{\prod_{i=1}^K e^{\log \phi_i(D_i)}}{Z} = \frac{e^{\sum_{i=1}^K \log \phi_i(D_i)}}{Z}.$$

However, in the context of parameter learning, it is convenient to represent the parameters of this network more explicitly. Consider a set of parameters $\theta = \{\theta_{i,u} \in \mathbb{R} | i \in \{1, \ldots, K\} \text{ and } u \in \operatorname{Scope}[\phi_i]\}$. By the equation above:

$$P(\mathcal{X} = \xi) = \frac{1}{Z} \exp\left\{ \sum_{\theta_{i,u} \in \theta} \theta_{i,u} f_{i,u}(\xi\langle D_i \rangle) \right\},$$

where $f_{i,u}(u') = 1$ if $u' = u$ and 0 otherwise, and $\theta_{i,u} = \log \phi_i(u)$.

Using an appropriate numbering of the parameters, the equation above can be rewritten as the following parameterized joint distribution:

$$P(\mathcal{X} : \theta) = \frac{1}{Z(\theta)} \exp\left\{ \sum_{i=1}^k \theta_i f_i(F_i) \right\}.$$

This apparently redundant reformulation will be very useful to derive the likelihood of the training data.

Consider a data set $\mathcal{D} = \{\xi[1], \ldots, \xi[M]\}$ of *IID* sample elements from a probability distribution P^*. Consider a Markov network parameterized by a set of parameters θ, as defined above. It is easy to show that the log-likelihood of the data can be written as:

$$\ell(\theta : \mathcal{D}) = \sum_{i=1}^k \theta_i \left(\sum_{j=1}^M f_i(\xi[j]\langle F_i \rangle) \right) - M \ln Z(\theta)$$

$$\frac{\ell(\theta : \mathcal{D})}{M} = \sum_{i=1}^k \theta_i \mathbb{E}_{\mathcal{D}}[f_i(F_i)] - \ln Z(\theta),$$

where $\mathbb{E}_{\mathcal{D}}$ represents the empirical expectation (empirical average). Because the log-likelihood is bounded by 1, the term $\ln Z(\theta)$ clearly must balance the first term. It can be shown that the log-likelihood is concave with respect to θ (every local maxima is a global maxima).

It can also be shown that θ is the maximum likelihood parameter if and only if $\mathbb{E}_{\mathcal{D}}[f_i(F_i)] = \mathbb{E}_{P(\mathcal{X}:\theta)}[f_i(F_i)]$ for every i. In other words, if the average value of f_i in the data set matches the expected value of f_i in a distribution parameterized by θ. Unfortunately, there is no closed analytical formula for maximum likelihood estimation in Markov networks.

For iterative optimization (e.g. gradient ascent), the theorem mentioned in the previous paragraph is prohibitively expensive, because it requires the computation of the probability of every possible assignment at each optimization step.

13 Structure Learning

The previous section focused on parameter learning in probabilistic graphical models: learning the value given by each factor to each assignment in its domain. This section focuses on structure learning for Bayesian networks: decomposing a joint probability distribution P^* into a product of conditional probability distributions.

Once again, we will assume that we have a sample $\mathcal{D} = \{\xi[1], \ldots, \xi[M]\}$ of independent, identically distributed (according to P^*) sample elements. We will also assume that $P^*(\mathcal{X})$ is induced by some Bayesian network $\mathcal{G}^*$ over $\mathcal{X} = \{X_1, \ldots, X_n\}$, even though there might not be a Bayesian network that is a perfect map for P^*.

Given the data $\mathcal{D}$, we say $\hat{P}(\mathcal{X})$ is the empirical (joint) distribution according to $\mathcal{D}$ if:

$$\hat{P}(\mathcal{X} = \xi) = \frac{1}{M} \sum_{m=1}^{M} \mathbb{1}(\xi = \xi[m]) = \frac{M[\xi]}{M},$$

for every $\xi \in \mathrm{Val}(\mathcal{X})$. A very important observation is that, even if two variables $X, Y \in \mathcal{X}$ are perfectly independent with respect to P^*, it is generally unlikely that $\hat{P}(X, Y) = \hat{P}(X)\hat{P}(Y)$ for a given $\mathcal{D}$.

There are several reasons for learning structure from data. One is knowledge discovery: finding relationships between the variables that represent a problem of interest. However, it is important to note that many structures may represent the correct set of independencies (the I-equivalence class of $\mathcal{G}^*$), and there might be no objective way of distinguishing between these structures.

Another goal might be density estimation: learning the underlying distribution P^* from the data. Given enough data, $\hat{P}$ converges to P^*. However, decomposing a joint probability distribution as a product of conditional probability distributions reduces the issue of data fragmentation, which allows better parameter estimation and, by consequence, generalization. For the same reason, unnecessary edges are undesirable in structure learning. Somewhat surprisingly, models that make incorrect independence assumptions (sparser models) might be better for generalization given insufficient data.

Structure learning can be decomposed into two steps: evaluating structures according to some criterion and finding optimal structures with respect to that criterion. Firstly, we will describe three ways of evaluating structures: likelihood score, Bayesian score and Bayesian information criterion.

The likelihood scoring function score_L is defined as:

$$\mathrm{score}_L(\mathcal{G} : \mathcal{D}) = \ell(\hat{\theta}_{\mathcal{G}} : \mathcal{D}),$$

where $\hat{\theta}_{\mathcal{G}} = \arg\max_{\theta_{\mathcal{G}}} \ell(\theta_{\mathcal{G}} : \mathcal{D})$. In words, the likelihood score for a graph is the log-likelihood of the data given the parameterization that maximizes its probability of occurring.

It can be shown that the likelihood score can be rewritten as:

$$\mathrm{score}_L(\mathcal{G} : \mathcal{D}) = M \sum_{i=1}^{n} \mathbf{I}_{\hat{P}}(X_i; \mathrm{Pa}_{X_i}^{\mathcal{G}}) - M \sum_{i=1}^{n} \mathbf{H}_{\hat{P}}(X_i),$$

where $\mathbf{I}_{\hat{P}}(X_i; \mathrm{Pa}_{X_i}^{\mathcal{G}})$ is the empirical mutual information between X_i and its parents, and $\mathbf{H}_{\hat{P}}(X_i)$ is the empirical entropy of X_i. These quantities are defined as:

$$\mathbf{I}_P(X; U) = \sum_{(x,u)\rangle \in \mathrm{Val}(X,U)} P(x, u) \log \frac{P(x, u)}{P(x)P(u)},$$

and

$$\mathbf{H}_P(X) = -\sum_{x \in \text{Val}(X)} P(x) \log P(x).$$

Mutual information can be seen as a measure of dependence between random variables. It can be shown to be non-negative and, for any sets of variables $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$ and distribution P, $\mathbf{I}_P(\mathbf{X}; \mathbf{Y}) \leq \mathbf{I}_P(\mathbf{X}; \mathbf{Y} \cup \mathbf{Z})$.

Entropy can be seen as a measure of uncertainty about a random variable under a distribution P. It is highest when the distribution P is uniform.

From the equation for the likelihood and the properties of mutual information, it is possible to see that any subgraph of $\mathcal{G}$ has a lower or equal score to $\mathcal{G}$. Furthermore, a maximum score structure would only exhibit a conditional independency if it existed in the empirical distribution, which is often unlikely. For this reason, the maximum score structure is said to overfit the data: it represents the empirical distribution very well, but is not expected to make good predictions due to data fragmentation.

In a Bayesian formulation of the structure learning task, we define a joint probability distribution over graphs, parameters and data. The probability of a graph $\mathcal{G}$ given the data $\mathcal{D}$ is denoted by:

$$P(\mathcal{G}|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{G})P(\mathcal{G})}{P(\mathcal{D})}.$$

Therefore, a graph with maximum a posterior probability is one that maximizes the Bayesian scoring function score_B, defined as:

$$\text{score}_B(\mathcal{G} : \mathcal{D}) = \log P(\mathcal{D}|\mathcal{G}) + \log P(\mathcal{G}).$$

In practice, the term $P(\mathcal{G})$ is often considered unimportant, because it remains constant when $M \to \infty$. Usually, $P(\mathcal{G}) \propto c^{|E[\mathcal{G}]|}$, where $c \in (0, 1]$ is a constant.

By marginalization:

$$P(\mathcal{D}|\mathcal{G}) = \int_{\Theta_\mathcal{G}} P(\mathcal{D}|\theta_\mathcal{G}, \mathcal{G}) P(\theta_\mathcal{G}|\mathcal{G}) \; d\theta_\mathcal{G}.$$

For this reason, the term $P(\mathcal{D}|\mathcal{G})$ is also called the marginal likelihood of the data given the structure. It can be interpreted as the average probability of the data given each possible parameterization, weighted by the probability of that parameterization. This is significantly different from the maximum likelihood score for a structure, which is the log-probability of the data under the best parameterization.

The Bayesian score naturally tries to find models with good generalization properties. Intuition can be gained by rewriting the marginal likelihood using the chain rule:

$$P(\mathcal{D}|\mathcal{G}) = P(\xi[1], \ldots, \xi[M]|\mathcal{G}) = \prod_{m=1}^{M} P(\xi[m]|\xi[1], \ldots, \xi[m-1], \mathcal{G}).$$

From this equation, it is possible to see that the marginal likelihood takes into account the quality of the prediction given by the structure $\mathcal{G}$ in the available data, when the data are considered incrementally. Notice that this observation still holds in any reordering of the sample elements.

An analogous equation can be written for a graph $\mathcal{G}$ with maximum likelihood parameters $\hat{\theta}_\mathcal{G}$:

$$P(\mathcal{D} : \hat{\theta}_\mathcal{G}, \mathcal{G}) = \prod_{m=1}^{M} P(\xi[m]|\xi[1], \ldots, \xi[m-1] : \hat{\theta}_\mathcal{G}, \mathcal{G}).$$

However, notice that finding $\hat{\theta}_\mathcal{G}$ requires observing all the data first, and using this information in *retrospective* to compute the likelihood. This is a subtle, yet crucial distinction.

Consider a joint distribution over a binary random variable X and its parameters $\boldsymbol{\theta}$, such that $P(\boldsymbol{\theta})$ is Dirichlet(α_0, α_1) and $\alpha = \alpha_0 + \alpha_1$. Because the Dirichlet is a conjugate prior to this model:

$$P(x[m+1] = 0|x[1], \ldots, x[m]) = \frac{M_m[0] + \alpha_0}{m + \alpha},$$

where $M_m[x]$ denotes the number of occurrences of x up to (and including) the m-th sample element. By the chain rule:

$$P(x[1],\ldots,x[M]) = \frac{[\alpha_0 \cdot \ldots \cdot (\alpha_0 + M[1] - 2) \cdot (\alpha_0 + M[1] - 1)][\alpha_1 \cdot \ldots \cdot (\alpha_1 + M[1] - 2) \cdot (\alpha_1 + M[1] - 1)]}{\alpha \cdot \ldots \cdot (\alpha + M - 2) \cdot (\alpha + M - 1)}.$$

The Gamma function Γ respects the equality $\Gamma(x+1) = x\Gamma(x)$. The equation above can be rewritten in terms of the Gamma function as:

$$P(x[1],\ldots,x[M]) = \frac{\Gamma(\alpha)}{\Gamma(\alpha + M)} \frac{\Gamma(\alpha_0 + M[0])}{\Gamma(\alpha_0)} \frac{\Gamma(\alpha_1 + M[1])}{\Gamma(\alpha_1)}.$$

In the case of a multinomial distribution for X such that $\mathrm{Val}(X) = \{1,\ldots,k\}$:

$$P(x[1],\ldots,x[M]) = \frac{\Gamma(\alpha)}{\Gamma(\alpha + M)} \prod_{i=1}^{k} \frac{\Gamma(\alpha_i + M[i])}{\Gamma(\alpha_i)}.$$

The equation above is fundamental to define the marginal likelihood for a given Bayesian network graph. Let $\mathcal{G}$ be such a graph and $P(\boldsymbol{\theta}|\mathcal{G})$ be a parameter prior satisfying global and local parameter independence. It follows that:

$$\begin{aligned}
P(\mathcal{D}|\mathcal{G}) &= \int_{\Theta_\mathcal{G}} P(\mathcal{D}, \theta_\mathcal{G}|\mathcal{G}) \; d\theta_\mathcal{G} \\
&= \int_{\Theta_\mathcal{G}} P(\mathcal{D}|\theta_\mathcal{G}, \mathcal{G}) P(\theta_\mathcal{G}|\mathcal{G}) \; d\theta_\mathcal{G} \\
&= \prod_{i=1}^{n} \prod_{u_i \in \mathrm{Val}(\mathrm{Pa}_{X_i}^\mathcal{G})} \int_{\Theta_{X_i|u_i}} \prod_{m=1|\xi[m]\langle \mathrm{Pa}_{X_i}\rangle = u_i}^{M} P(X_i[m]|u_i, \theta_{X_i|u}, \mathcal{G}) P(\theta_{X_i|u_i}|\mathcal{G}) \; d\theta_{X_i|u_i}.
\end{aligned}$$

By using a Dirichlet prior:

$$P(\mathcal{D}|\mathcal{G}) = \prod_{i=1}^{n} \prod_{u_i \in \mathrm{Val}(\mathrm{Pa}_{X_i}^\mathcal{G})} \frac{\Gamma(\alpha_{X_i|u_i}^\mathcal{G})}{\Gamma(\alpha_{X_i|u_i}^\mathcal{G} + M[u_i])} \left[\prod_{j \in \mathrm{Val}(X_i)} \frac{\Gamma(\alpha_{X_i=j|u_i}^\mathcal{G} + M[X_i = j, u_i])}{\Gamma(\alpha_{X_i=j|u_i}^\mathcal{G})} \right].$$

The formula above should be substituted by its logarithm in numerical implementations of the Bayesian score.

Given a Bayesian network where all parameter priors are Dirichlet, when $M \to \infty$:

$$\log P(\mathcal{D}|\mathcal{G}) = \ell(\hat{\theta}_\mathcal{G} : \mathcal{D}) - \frac{\log M}{2} \mathrm{Dim}[\mathcal{G}] + c,$$

where $\mathrm{Dim}[\mathcal{G}]$ is the number of independent parameters in $\mathcal{G}$, and c is a constant with respect to $\mathcal{G}$. Thus, the Bayesian score trades off fitting the data (maximizing the likelihood) for model simplicity (having fewer edges).

The Bayesian information criterion is defined in a similar way to the asymptotic approximation to the Bayesian score:

$$\mathrm{score}_{\mathrm{BIC}}(\mathcal{G} : \mathcal{D}) = \ell(\hat{\theta}_\mathcal{G} : \mathcal{D}) - \frac{\log M}{2} \mathrm{Dim}[\mathcal{G}].$$

It can be shown that the maximum log-likelihood term decreases faster than the second term. Thus, when enough data are considered, a good fit to the data is preferred to model simplicity. The negative BIC score is highly related to minimum description length, an important concept in information theory.

The Bayesian score and the BIC score have an extremely important property: consistency. Let P^* be a distribution from which we sample the M IID sample elements, and $\mathcal{G}^*$ a perfect map for P^*. A scoring function is consistent if, when $M \to \infty$, $\mathcal{G}^*$ maximizes the score and every structure that is not I-equivalent to $\mathcal{G}^*$ has a strictly lower score. Of course, this property is asymptotic, and does not guarantee appropriate scoring for small M.

A scoring function satisfies score equivalence if all I-equivalent structures have the same score, for all possible data sets. The likelihood score and the BIC score satisfy score equivalence. When using a uniform BDe prior over parameterizations for every structure, score equivalence also holds for the Bayesian score. The property also holds for a larger class of BDe priors, which were not introduced in this text.

Having defined the scoring functions that evaluate every possible structure given the data, we now describe how to find high-scoring structures.

Using a non-trivial construction, the task of learning an optimal Bayesian network structure whose maximum in-degree is 1 can be reduced to the problem of finding a maximum spanning tree in a complete graph over $\mathcal{X}$. This makes the problem solvable in polynomial time in the number of variables.

The scoring functions presented so far have a property called family decomposition. A scoring function score satisfies family decomposition when it can be written as:

$$\text{score}(\mathcal{G} : \mathcal{D}) = \sum_{i=1}^{n} \text{FamScore}(X_i | \text{Pa}_{X_i}^{\mathcal{G}} : \mathcal{D}),$$

where, intuitively, $\text{FamScore}(X_i | \text{Pa}_{X_i}^{\mathcal{G}} : \mathcal{D})$ represents how well $\text{Pa}_{X_i}^{\mathcal{G}}$ fit as parents of X_i given the data $\mathcal{D}$.

Let $\boldsymbol{\mathcal{G}}_d = \{\mathcal{G} \mid d \geq |\text{Pa}_{X_i}|, \forall X_i \in V[\mathcal{G}]\}$ be the set of graphs over $\mathcal{X}$ with maximum in-degree d. Given an IID data set $\mathcal{D}$ and a family-decomposable scoring function score, the task of finding $\mathcal{G}^* = \arg\max_{\mathcal{G} \in \boldsymbol{\mathcal{G}}_d} \text{score}(\mathcal{G} : \mathcal{D})$ is NP-hard for any $d \geq 2$. Thus, no polynomial time algorithm is known for this task.

In practice, this problem is usually solved using a local search procedure, which does not guarantee finding an optimal structure. The procedure starts with a data set $\mathcal{D}$, an initial candidate structure $\mathcal{G}_0$, a scoring function score, parameter priors (if necessary), and a fixed number of iterations t. The initial candidate structure might be, for instance, the graph over $\mathcal{X}$ with no edges, or the optimum structure whose maximum in-degree is 1. The procedure works as follows. Given the current candidate structure $\mathcal{G}'$, all possible acyclic graphs that can be created from $\mathcal{G}'$ by the addition, removal or reversal of an edge are scored and, with high probability, the best is chosen to be the next candidate. This procedure continues for the given number of iterations t, or until the score does not change significantly according to some criterion.

Local search procedures are often combined with other heuristics to avoid non-global maxima, such as restarting, choosing a sequence of random candidates, and prohibiting changes to recently changed edges. The efficient implementation of local search procedures requires careful caching of sufficient statistics and family scores.

14 Learning with incomplete data

A data set contains incomplete data when at least one assignment to a variable in a sample element is unknown. Many important learning tasks can be formulated as learning from incomplete data.

Firstly, this section presents the conditions that must be met to allow principled learning from incomplete data.

An *observability* model is a combination of two other models: the distribution of the complete data (underlying model) and the distribution for the mechanism that hides the data (observation model).

Let $\mathcal{X} = \{X_1, \ldots, X_n\}$ be a set of random variables and $O_{\mathcal{X}} = \{O_{X_1}, \ldots, O_{X_n}\}$ be their observability variables. The observability model is a joint distribution $P_{\text{miss}}(\mathcal{X}, O_{\mathcal{X}}) = P(\mathcal{X}) P_{\text{miss}}(O_{\mathcal{X}} | \mathcal{X})$. Also, let $\mathcal{Y} = \{Y_1, \ldots, Y_n\}$ be a set of random variables such that $\text{Val}(Y_i) = \text{Val}(X_i) \cup \{?\}$, where ? is a number that represents an unobserved value. Every assignment to Y_i is a deterministic function of the assignment to X_i and O_{X_i}. Concretely, $y_i = x_i$ if $o_{X_i} = 1$ and $y_i =?$ if $o_{X_i} = 0$. In other words, O_{X_i} decides whether X_i is observable through Y_i. In this formulation, an incomplete data set $\mathcal{D}$ is a complete assignment to the set of variables $\mathcal{Y}$. From the incomplete data set, the assignment to O_X is always known, and the assignment to X_i is known whenever $o_{X_i} = 1$.

This text is only concerned with two models for learning with incomplete data: missing completely at random and missing at random.

An observability model is missing completely at random when $\mathcal{X} \perp\!\!\!\perp O_{\mathcal{X}}$. In this case, $P_{\text{miss}}(\mathcal{X}, O_{\mathcal{X}}) = P(\mathcal{X}) P_{\text{miss}}(O_{\mathcal{X}})$, where P is the underlying distribution of $\mathcal{X}$. In other words, the probability of observing an assignment does not depend on its values. Intuitively, no information is gained by knowing which values were missing.

Consider an observability model and an assignment $y \in \text{Val}(\mathcal{Y})$. Let x_{obs} denote the assignment to the observed variables. An observability model is missing at random when, for all y with $P_{\text{miss}}(y) > 0$, $o_X \perp\!\!\!\perp x_{\text{hid}} \mid x_{\text{obs}}$ holds for all possible assignment to the unobserved variables x_{hid}. In other words, the probability of observing an assignment may depend only on the values of the variables that are observed. Intuitively, no information is gained by knowing which variables were not observed after knowing the observed values.

By definition of an observability model:

$$P_{\text{miss}}(y) = \sum_{x_{\text{hid}}} P(x_{\text{obs}}, x_{\text{hid}}) P_{\text{miss}}(o_{\mathcal{X}} \mid x_{\text{obs}}, x_{\text{hid}}).$$

Therefore, given a missing at random model:

$$\begin{aligned} P_{\text{miss}}(y) &= \sum_{x_{\text{hid}}} P(x_{\text{obs}}, x_{\text{hid}}) P_{\text{miss}}(o_{\mathcal{X}} \mid x_{\text{obs}}) \\ &= P(x_{\text{obs}}) P_{\text{miss}}(o_{\mathcal{X}} \mid x_{\text{obs}}). \end{aligned}$$

The missing at random assumption must be considered very carefully. Very often, the absence of a variable is informative about the values of other unobserved variables until an expanded set of observed variables is adopted.

Consider a Bayesian network structure over a set of variables $\mathcal{X}$ and a corresponding incomplete data set $\mathcal{D}$. Consider the M sample elements in $\mathcal{D}$ as independent, identically distributed sample elements from an underlying distribution P after being subject to an observability model P_{miss}. Let parameters for $P(\mathcal{X})$ be represented by θ, as in the previous sections. Let $\boldsymbol{O}[m] \subseteq \mathcal{X}[m]$ denote the observed random variables in the m-th sample element and $\boldsymbol{H}[m] = \mathcal{X}[m] - \boldsymbol{O}[m]$ the unobserved variables in the same sample element. The likelihood of the incomplete data set is defined as:

$$\begin{aligned} L(\theta : \mathcal{D}) &= \prod_{m=1}^{M} P(\boldsymbol{o}[m] | \theta). \\ &= \prod_{m=1}^{M} \sum_{\boldsymbol{H}[m]} P(\boldsymbol{o}[m], \boldsymbol{H}[m] | \theta) \end{aligned}$$

If P_{miss} is a missing at random model, then $L(\theta, \psi : \mathcal{D}) = L(\theta : \mathcal{D}) L(\psi : \mathcal{D})$, where $L(\psi : \mathcal{D})$ is the likelihood of the the observation model under the parameters ψ. This property implies that the maximum likelihood parameters for the underlying model can be estimated independently of the maximum likelihood parameters of the observation model.

It is possible to show that the likelihood of an incomplete data set $\mathcal{D}$ is the sum of the likelihood of all complete data sets consistent with the incomplete data set. More formally:

$$L(\theta : \mathcal{D}) = \sum_{\mathcal{H} \in \text{Val}(\bigcup_{m=1}^{M} \boldsymbol{H}[m])} L(\theta : \langle \mathcal{D}, \mathcal{H} \rangle),$$

where $L(\theta : \langle \mathcal{D}, \mathcal{H} \rangle)$ is the likelihood of the incomplete data set $\mathcal{D}$ when completed by the assignment $\mathcal{H}$. The likelihood of an incomplete data set does not necessarily have a single global maximum with respect to θ. However, the incomplete data likelihood is the sum of the likelihood of complete data sets, each having a single global maximum with respect to θ. The fact that a maximum likelihood model is not necessarily unique is known as nonidentifiability. Therefore, even if we could find maximum likelihood parameters for incomplete data, the resulting model should be interpreted carefully.

Unfortunately, key properties of complete data do not generalize to incomplete data. In the case of Bayesian networks, the parameters for a variable given distinct assignments to its parents are not necessarily independent, and the likelihood function is not necessarily the product of likelihoods corresponding to each conditional probability distribution. This makes the parameter learning task considerably more challenging.

The expectation maximization algorithm performs maximum likelihood estimation for incomplete data. Intuitively, this algorithm divides this task into two steps: (virtually) estimating the maximum likelihood parameters given complete data and (virtually) hypothesizing values for unobserved variables given a complete model. Each of these tasks is easy given the correct solution to the other. Expectation maximization bootstraps this procedure by starting at an arbitrary point in the parameter (or assignment to hidden variables) space, and iteratively recomputing the solution to one of the two tasks given the other.

Consider the task of learning the parameters for a Bayesian network over the set of variables $\mathcal{X}$ given incomplete data under a missing at random observability model. The expectation maximization algorithm works as follows. Let $\theta^{(0)}$ be an initial (possibly random) assignment to the parameters of the network. Compute the *expected sufficient statistics* $\bar{M}_{\theta^{(t)}}[x, \text{pa}_X]$ for every $X \in \mathcal{X}$, $(x, \text{pa}_X) \in \text{Val}(X, \text{Pa}_X)$, defined as:

$$\bar{M}_{\theta^{(t)}}[X = x, \text{Pa}_X = \text{pa}_X] = \sum_{m=1}^{M} P(X = x[m], \text{Pa}_X = \text{pa}_X[m] \mid \boldsymbol{o}[m], \theta^{(t)})$$

When $X[m]$ and $\text{Pa}_X[m]$ are observed, notice that $P(X = x[m], \text{Pa}_X = \text{pa}_X[m] \mid \boldsymbol{o}[m], \theta^{(t)}) = 1$ if $(x[m], \text{pa}_X[m])$ is consistent with $\boldsymbol{o}[m]$ and 0 otherwise. Intuitively, the expected sufficient statistics replace *hard* statistics with

soft estimates based on conditional probabilities given the observed variables and the current parameters of the model.

This completes the *expectation* step. The next step is *maximization*, where the expected sufficient statistics are applied to update the parameters, as if they were sufficient statistics in maximum likelihood estimation for complete data:

$$\theta^{(t+1)}_{X=x|\mathrm{Pa}_X=\mathrm{pa}_X} = \frac{\bar{M}_{\theta^{(t)}}[X=x, \mathrm{Pa}_X=\mathrm{pa}_X]}{\bar{M}_{\theta^{(t)}}[\mathrm{Pa}_X=pa_X]}$$

This completes the maximization step. These two steps are repeated until a desired criterion is met (usually convergence, maximum number of iterations or decrease in likelihood of validation data). Algorithm 7 illustrates a particular (naive) implementation of expectation maximization for Bayesian networks. Several simplifications are made for presentation purposes.

Algorithm 7 Expectation maximization for Bayesian network parameter learning given incomplete data (missing at random).

Input: network structure $\mathcal{G}$ over variables $\mathcal{X}$, set of observations $\{\boldsymbol{o}[1], \ldots, \boldsymbol{o}[M]\}$, number of iterations T, equivalent sample size α
Output: parameter estimate θ

for each $X \in \mathcal{X}$ **do**
  for each $\mathrm{pa}_X \in \mathrm{Val}(\mathrm{Pa}_X)$ **do**
    Sample $\theta_{X|\mathrm{Pa}_X=\mathrm{pa}_X}$ from Dirichlet $\left(\frac{\alpha}{|\mathrm{Val}(X,\mathrm{Pa}_X)|}, \ldots, \frac{\alpha}{|\mathrm{Val}(X,\mathrm{Pa}_X)|}\right)$
  end for
end for
for each t in $1, \ldots, T$ **do**
  for each $X \in \mathcal{X}$ **do**
    for each $(x, \mathrm{pa}_X) \in \mathrm{Val}(X, \mathrm{Pa}_X)$ **do**
      $\bar{M}[X=x, \mathrm{Pa}_X=\mathrm{pa}_X] \leftarrow 0$
    end for
  end for
  for each m in $1, \ldots, M$ **do**
    Calibrate a clique tree for a structure $\mathcal{G}$ parameterized by θ, using $\boldsymbol{o}[m]$ as evidence.
    for each $X \in \mathcal{X}$ **do**
      for each $(x, \mathrm{pa}_X) \in \mathrm{Val}(X, \mathrm{Pa}_X)$ **do**
        $\bar{M}[X=x, \mathrm{Pa}_X=\mathrm{pa}_X] \leftarrow \bar{M}[X=x, \mathrm{Pa}_X=\mathrm{pa}_X] + P(X=x, \mathrm{Pa}_X=\mathrm{pa}_X \mid \boldsymbol{o}[m])$
      end for
    end for
  end for
  for each $X \in \mathcal{X}$ **do**
    for each $(x, \mathrm{pa}_X) \in \mathrm{Val}(X, \mathrm{Pa}_X)$ **do**
      $\theta_{X=x|\mathrm{Pa}_X=\mathrm{pa}_X} \leftarrow \frac{\bar{M}[X=x, \mathrm{Pa}_X=\mathrm{pa}_X]}{\bar{M}[\mathrm{Pa}_X=\mathrm{pa}_X]}$
    end for
  end for
end for

In Algorithm 7, the initial parameters are sampled from a Dirichlet distribution with a (global) equivalent sample size controlled by α for each variable given its parents. In general, the initial conditional probabilities of a variable given its parents should not be uniformly distributed. Whenever possible, prior knowledge should be incorporated into the initial parameters.

Notice that the expectation step requires inference. The ideal candidate in this case is usually belief propagation, because a single clique tree calibration (conditioned on the observed variables) allows computing all the conditional probabilities related to a single sample element.

In expectation maximization, the log-likelihood of the incomplete data is non-decreasing with respect to t. Furthermore, if $\theta^{(t)} = \theta^{(t+1)}$, then $\theta^{(t)}$ is a stationary (minimum, maximum or saddle) point of the log-likelihood with respect to the parameters. In general, the algorithm does not guarantee convergence to a global maximum.

One possible application of expectation maximization is Bayesian clustering. Consider a set of variables $\{X_1, \ldots, X_n\}$ that correspond to features of some object of interest, and let C be a variable that corresponds

to the group containing a given object. Consider a Bayesian network structure identical to a naive Bayes classifier, such that any two features are independent given C. The task in Bayesian clustering is to learn a model that predicts the class of each observation given a data set where C is always hidden, and all other features are always observed. In this problem, expectation maximization would try to define a natural clustering through maximizing the likelihood of the training data. Hopefully, this would lead to modeling distinct distributions for distinct groups.

Consider an incomplete data set $\mathcal{D}$ and let $\mathcal{H} \in \text{Val}(\bigcup_{m=1}^{M} \boldsymbol{H}[m])$ once again denote an assignment to the hidden variables in $\mathcal{D}$ (across all sample elements). Let $\langle \mathcal{D}, \mathcal{H} \rangle$ denote the corresponding complete data set. The log-likelihood of this data set can be written as:

$$\ell(\theta : \langle \mathcal{D}, \mathcal{H} \rangle) = \sum_{X \in \mathcal{X}} \sum_{(x,u) \in \text{Val}(X, \text{Pa}_X)} M_{\langle \mathcal{D}, \mathcal{H} \rangle}[X = x, \text{Pa}_X = u] \log \theta_{X=x|\text{Pa}_X=u}$$

Let Q be a joint probability distribution over all possible $\mathcal{H}$, representing the probability of each completion to the data set. The expected log-likelihood under Q of an incomplete data set parameterized by θ is defined as:

$$\mathbb{E}_Q[\ell(\theta : \mathcal{D})] = \sum_{\mathcal{H}} Q(\mathcal{H}) \ell(\theta : \langle \mathcal{D}, \mathcal{H} \rangle)$$

It can be shown that the maximization step of expectation maximization finds the set of parameters $\theta^{(t+1)}$ that maximize the expected log-likelihood of an incomplete data set given $Q(\mathcal{H}) = P(\mathcal{H} \mid \mathcal{D}, \theta^{(t)})$.

In practice, since there are no guarantees about the quality of the stationary point obtained, Algorithm 7 is repeated several times and the parameters with highest likelihood (or likelihood on a validation set) are kept.

Usually, the largest likelihood improvement occurs in the first iterations. It is possible to combine expectation maximization with gradient ascent in later iterations, because the latter is usually faster in later iterations.

A second issue is overfitting: a learned model may achieve high likelihood in the training set but perform poorly for generalization. In this case, it is often necessary to stop expectation maximization when the likelihood of a validation set decreases.

Structure learning can be combined naturally with parameter estimation given incomplete data, although the computational cost might be prohibitive.

License

References

[1] Koller, D. and Friedman, N. *Probabilistic Graphical Models: Principles and Techniques*, 2009.

[2] Wasserman, L. *All of Statistics: A Concise Course in Statistical Inference*. Springer, 2010.

[3] Diestel, R. *Graph Theory* Springer, 2000.