Probabilistic Graphical Models

Table of Contents

14.1 Hidden Markov Model

The most important problem in machine learning is to estimate and infer the value of unknown variables (e.g., class label) based on the observed evidence (e.g., training samples). Probabilistic models provide a framework that considers learning problems as computing the probability distributions of variables. In probabilistic models, the process of inferring the distributions of unknown variables conditioned on known variables is called *inference*. More specifically, let Y denote the set of target variables, O denote the set of observable variables, and R denote the set of other variables. Then, *generative* models consider the joint distribution $P(Y, R, O)$, while *discriminative* models consider the conditional distribution $P(Y, R \mid O)$. Given the values of a set of observed variables, inference aims to obtain the conditional probability $P(Y \mid O)$ from $P(Y, R, O)$ or $P(Y, R \mid O)$.

A learner makes inference when it predicts the ripeness of watermelon based on information such as texture, color, and root. However, the inference is more than just prediction. For example, when we eat a ripe watermelon and try to infer the shape of its root reversely, it is also inference.

It is impractical to eliminate R by probability marginalization since the computational complexity is prohibitive. For example, it costs at least $O(2^{|Y|+|R|})$ operations even if each variable has just two possible values. Besides, the learning process of probabilistic models, that is, estimating the parameters of variable distributions from the data set, is not easy since there are often complex relationships between variables. Hence, we must develop a methodology to concisely represent the relationships between variables for the study of efficient learning and inference.

Probabilistic Graphical Models (PGM) are a family of probabilistic models that represent the relationships between variables with graph structures, in which, each node (also known as vertex) represents one or a set of random variables and each link (also known as edge) between two nodes represents the probabilistic relationship between the variables. Depending on the properties of edges, probabilistic graphical models can be roughly divided into two categories. The first category is called directed graphical models or Bayesian networks, which employ Directed Acyclic Graphs (DAG) to represent the dependence between variables. The second category is called undirected graphical models or Markov networks, which employ undirected graphs to represent the dependence between variables.

Bayesian networks are often used when explicit causal relationships exist between variables. Markov networks are often used when correlations exist between variables while explicit causal relationships are difficult to obtain.

See Sect. 7.5 for static Bayesian network.

The simplest *dynamic Bayesian network* is Hidden Markov Model (HMM), a well-known directed graphical model commonly used for modeling time-series data and has been widely used in speech recognition and natural language processing.

As illustrated in ◘ Figure 14.1, there are two sets of variables in an HMM. The first set of variables are state variables $\{y_1, y_2, \ldots, y_n\}$, where $y_i \in \mathcal{Y}$ represents the system state at

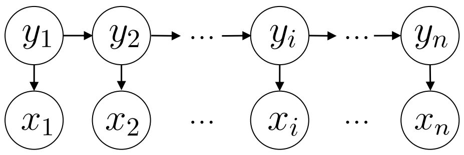

Fig. 14.1 The graph structure of HMM

the ith time point. The state variables are usually assumed to be hidden and unobserved, hence they are also called *hidden variables*. The second set of variables are observed variables $\{x_1, x_2, \ldots, x_n\}$, where $x_i \in \mathcal{X}$ represents the observation at the ith time point. In HMM, the system changes between different states $\{s_1, s_2, \ldots, s_N\}$, and hence the state space $\mathcal{Y}$ is usually a discrete space with N possible values. The observed variables x_i, however, can be either discrete or continuous. For ease of discussion, we assume that the observed variables are discrete, i.e., $\mathcal{X} = \{o_1, o_2, \ldots, o_M\}$.

The directed links in Figure 14.1 represent the dependence between variables. For each time point, the value of an observed variable only depends on the state variable, that is, x_t is solely determined by y_t. Meanwhile, the state y_t at time point t only depends on the state y_{t-1} at time point $t-1$ and is independent of the previous $t-2$ states. Such a model is known as *Markov chain*, in which the system state at the next time point does not depend on any previous states but the current state. Under this dependence setting, the joint probability of all variables is

“The future depends on what you do today.”—Mahatma Gandhi.

$$P(x_1, y_1, \ldots, x_n, y_n) = P(y_1)P(x_1 \mid y_1) \prod_{i=2}^{n} P(y_i \mid y_{i-1})P(x_i \mid y_i). \tag{14.1}$$

In addition to the structure information, an HMM has three more sets of parameters

- State transition probabilities: the probabilities that the model changes between states, usually denoted by matrix $\mathbf{A} = [a_{ij}]_{N \times N}$, where

  $$a_{ij} = P(y_{t+1} = s_j \mid y_t = s_i), \quad 1 \leqslant i, j \leqslant N$$

  indicates the probability that the next state is s_j when the current state is s_i at time point t.
- Output observation probabilities: the probabilities of observations based on the current state, usually denoted by matrix $\mathbf{B} = [b_{ij}]_{N \times M}$, where

$$b_{ij} = P(x_t = o_j \mid y_t = s_i), \quad 1 \leqslant i \leqslant N, 1 \leqslant j \leqslant M$$

indicates the probability of observing o_j when the current state is s_i at time point t.

- Initial state probabilities: the probability of each state that appears at the initial time point, usually denoted by $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_N)$, where

$$\pi_i = P(y_1 = s_i), \quad 1 \leqslant i \leqslant N$$

indicates the probability that the initial state is s_i.

The above three sets of parameters together with the state space $\mathcal{Y}$ and the observation space $\mathcal{X}$ determine an HMM, usually denoted by $\lambda = [\mathbf{A}, \mathbf{B}, \boldsymbol{\pi}]$. Given an HMM λ, it generates the observed sequence $\{x_1, x_2, \ldots, x_n\}$ by the following process:

(1) Set $t = 1$ and select the initial state y_1 based on the initial state probability $\boldsymbol{\pi}$;
(2) Select the value of the observed variable x_t based on the state variable y_t and the output observation probability matrix $\mathbf{B}$;
(3) Transition to the next state y_{t+1} based on the current state y_t and the state transition probability matrix $\mathbf{A}$;
(4) If $t < n$, set $t = t+1$ and return to step (2); otherwise, stop.

$y_t \in \{s_1, s_2, \ldots, s_N\}$ and $x_t \in \{o_1, o_2, \ldots, o_M\}$ are the state and observation at time point t, respectively.

There are three fundamental problems when applying HMM in practice:

- Given a model $\lambda = [\mathbf{A}, \mathbf{B}, \boldsymbol{\pi}]$, how can we effectively calculate the probability $P(\mathbf{x} \mid \lambda)$ for generating the observed sequence $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$? In other words, how to evaluate the matching degree between a model and an observed sequence?
- Given a model $\lambda = [\mathbf{A}, \mathbf{B}, \boldsymbol{\pi}]$ and an observed sequence $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$, how can we find the best state sequence $\mathbf{y} = \{y_1, y_2, \ldots, y_n\}$ that matches $\mathbf{x}$? In other words, how to infer the hidden states from the observed sequence?
- Given an observed sequence $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$, how can we adjust the model parameter $\lambda = [\mathbf{A}, \mathbf{B}, \boldsymbol{\pi}]$ such that the probability $P(\mathbf{x} \mid \lambda)$ of observing the given sequence is maximized? In other words, how to train the model such that it can better describe the observed data?

The above problems are critical in real-world applications. For example, in many tasks, we need to estimate the most likely value of the current observation x_n based on the previously observed sequence $\{x_1, x_2, \ldots, x_{n-1}\}$; this problem can be solved by finding $P(\mathbf{x} \mid \lambda)$, that is, the first listed problem. In speech recognition problems, the observations are audio signals, the hidden states are spoken texts, and the task is to infer the most likely state sequence (i.e., spoken texts) based on the observed sequence (i.e., audio signals), that is, the second listed problem. In many applications, manually specifying model parameters is becoming impractical, and hence model parameters need to be learned from data, that is, the third listed problem. Fortunately, thanks to the conditional independence given in (14.1), all of the three listed problems can be solved efficiently.

14.2 Markov Random Field

Markov Random Field (MRF) is a typical Markov network and a well-known undirected graphical model. In MRF, each node represents one or a set of variables, and the edges between nodes represent the variable dependence. Besides, there is a set of *potential functions*, also known as *factors*, which are non-negative real-valued functions defined over variable subsets mainly for defining probability distribution functions.

A simple MRF is illustrated in Figure 14.2. A subset of nodes in the graph is called a *clique* if there exists a link between any two nodes. We say a clique is a *maximal clique* if adding any extra node makes it no longer a clique; in other words, a maximal clique is a clique that is not contained in any other cliques. For example, the cliques in Figure 14.2 are $\{x_1, x_2\}$, $\{x_1, x_3\}$, $\{x_2, x_4\}$, $\{x_2, x_5\}$, $\{x_2, x_6\}$, $\{x_3, x_5\}$, $\{x_5, x_6\}$, and $\{x_2, x_5, x_6\}$, which are also maximal cliques except $\{x_2, x_5\}$, $\{x_2, x_6\}$, and $\{x_5, x_6\}$; $\{x_1, x_2, x_3\}$ is not a clique since there is no link between x_2 and x_3. We notice that every node appears in at least one maximal clique.

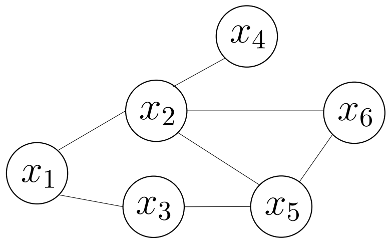

Fig. 14.2 A simple MRF

In MRF, the joint probability of multiple variables can be decomposed into the product of multiple factors based on the cliques, and each factor corresponds to one clique. To be specific, let $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ denote the set of n variables, $\mathcal{C}$ denote the set of all cliques, and $\mathbf{x}_Q$ denote the set of variables in clique $Q \in \mathcal{C}$, then the joint probability $P(\mathbf{x})$ is defined as

$$P(\mathbf{x}) = \frac{1}{Z} \prod_{Q \in \mathcal{C}} \psi_Q(\mathbf{x}_Q), \tag{14.2}$$

where ψ_Q is the potential function that captures the dependence between variables in clique Q; $Z = \sum_{\mathbf{x}} \prod_{Q \in \mathcal{C}} \psi_Q(\mathbf{x}_Q)$ is the normalization factor that ensures $P(\mathbf{x})$ is a properly defined probability. In practice, it is often difficult to calculate Z exactly, but we usually do not need the exact value.

When there are many variables, the number of cliques can be quite large. For example, every pair of linked variables forms a clique, and hence there will be many terms multiplied in (14.2), leading to high computational cost. We notice that, if clique Q is not a maximal clique, then it is contained in a maximal clique Q^* (i.e., $\mathbf{x}_Q \subseteq \mathbf{x}_{Q^*}$). Hence, the dependence between variables $\mathbf{x}_Q$ is not only encoded in the potential function ψ_Q, but also the potential function ψ_{Q^*}. Therefore, it is also possible to define the joint probability $P(\mathbf{x})$ based on the maximal cliques. Let $\mathcal{C}^*$ denote the set of all maximal cliques, we have

$$P(\mathbf{x}) = \frac{1}{Z^*} \prod_{Q \in \mathcal{C}^*} \psi_Q(\mathbf{x}_Q), \tag{14.3}$$

where $Z^* = \sum_{\mathbf{x}} \prod_{Q \in \mathcal{C}^*} \psi_Q(\mathbf{x}_Q)$ is the normalization factor. Taking ◘ Figure 14.2 as an example, the joint probability $P(\mathbf{x})$ can be defined as

$$P(\mathbf{x}) = \frac{1}{Z} \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \psi_{24}(x_2, x_4) \psi_{35}(x_3, x_5) \psi_{256}(x_2, x_5, x_6),$$

where the potential function $\psi_{256}(x_2, x_5, x_6)$ is defined over the maximal clique $\{x_2, x_5, x_6\}$. Since we have $\psi_{256}(x_2, x_5, x_6)$, there is no need to construct the potential functions for the cliques $\{x_2, x_5\}$, $\{x_2, x_6\}$, and $\{x_5, x_6\}$.

How can we obtain *conditional independence* in MRF? We still utilize the concept of *separation*. As illustrated in ◘ Figure 14.3, the path connecting a node in the node set A to a node in node set B passes through the node set C, and we say C is a *separating set* that separates A and B. For MRF, we have the *global Markov property*: two subsets of variables are conditionally independent given a separating set of these two subsets.

See Sect. 7.5.1.

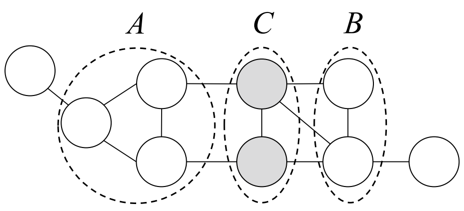

Fig. 14.3 The node set C separates the node set A and the node set B

Taking ◘ Figure 14.3 as an example, let $\mathbf{x}_A$, $\mathbf{x}_B$, and $\mathbf{x}_C$ denote the sets of variables for A, B, and C, respectively. Then, $\mathbf{x}_A$ and $\mathbf{x}_B$ are conditionally independent given $\mathbf{x}_C$, denoted by $\mathbf{x}_A \perp \mathbf{x}_B \mid \mathbf{x}_C$.

Now, let us do a simple verification. For ease of discussion, let A, B, and C correspond to single variables x_A, x_B, and x_C, respectively. Then, ◘ Figure 14.3 is simplified to ◘ Figure 14.4.

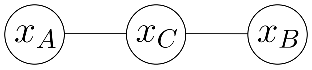

Fig. 14.4 A simplified version of ◘ Figure 14.3

From (14.2), the joint probability of the variables in ◘ Figure 14.4 is given by

$$P(x_A, x_B, x_C) = \frac{1}{Z}\psi_{AC}(x_A, x_C)\psi_{BC}(x_B, x_C). \tag{14.4}$$

According to the definition of conditional probability, we have

$$\begin{aligned} P(x_A, x_B \mid x_C) &= \frac{P(x_A, x_B, x_C)}{P(x_C)} = \frac{P(x_A, x_B, x_C)}{\sum_{x'_A}\sum_{x'_B} P(x'_A, x'_B, x_C)} \\ &= \frac{\frac{1}{Z}\psi_{AC}(x_A, x_C)\psi_{BC}(x_B, x_C)}{\sum_{x'_A}\sum_{x'_B} \frac{1}{Z}\psi_{AC}(x'_A, x_C)\psi_{BC}(x'_B, x_C)} \\ &= \frac{\psi_{AC}(x_A, x_C)}{\sum_{x'_A}\psi_{AC}(x'_A, x_C)} \cdot \frac{\psi_{BC}(x_B, x_C)}{\sum_{x'_B}\psi_{BC}(x'_B, x_C)}. \end{aligned} \tag{14.5}$$

$$\begin{aligned} P(x_A \mid x_C) &= \frac{P(x_A, x_C)}{P(x_C)} = \frac{\sum_{x'_B} P(x_A, x'_B, x_C)}{\sum_{x'_A}\sum_{x'_B} P(x'_A, x'_B, x_C)} \\ &= \frac{\sum_{x'_B}\frac{1}{Z}\psi_{AC}(x_A, x_C)\psi_{BC}(x'_B, x_C)}{\sum_{x'_A}\sum_{x'_B} \frac{1}{Z}\psi_{AC}(x'_A, x_C)\psi_{BC}(x'_B, x_C)} \\ &= \frac{\psi_{AC}(x_A, x_C)}{\sum_{x'_A}\psi_{AC}(x'_A, x_C)}. \end{aligned} \tag{14.6}$$

From (14.5) and (14.6), we have

$$P(x_A, x_B \mid x_C) = P(x_A \mid x_C)P(x_B \mid x_C), \tag{14.7}$$

that is, x_A and x_B are conditionally independent given x_C.

From the global Markov property, we can derive the following two useful corollaries:

The set of parents, children, and children's parents is called the *Markov blanket* of a variable.

- **Local Markov property**: a variable is conditionally independent of other variables given its adjacent variables. Formally, we have $\mathbf{x}_v \perp \mathbf{x}_{V \setminus n^*(v)} \mid \mathbf{x}_{n(v)}$, where V is the set of all nodes in the graph, $n(v)$ are the adjacent nodes of node v in the graph, and $n^*(v) = n(v) \cup \{v\}$.
- **Pairwise Markov property**: two non-adjacent variables are conditionally independent given all other variables. Formally, we have $\mathbf{x}_u \perp \mathbf{x}_v \mid \mathbf{x}_{V \setminus \langle u,v \rangle}$ if $\langle u, v \rangle \notin E$, where u and v are two nodes in the graph, and V and E are, respectively, the set of all nodes and the set of all edges in the graph.

Now, let us take a look at the potential functions in MRF. A potential function $\psi_Q(\mathbf{x}_Q)$ describes the dependence between a set of variables $\mathbf{x}_Q$. It should be a non-negative function that returns a large value when the variables take preferred values. For example, suppose that all variables in ◘ Figure 14.4 are binary variables, and the potential functions are

$$\phi_{AC}(x_A, x_C) = \begin{cases} 1.5, & \text{if } x_A = x_C; \\ 0.1, & \text{otherwise,} \end{cases}$$

$$\phi_{BC}(x_B, x_C) = \begin{cases} 0.2, & \text{if } x_B = x_C; \\ 1.3, & \text{otherwise,} \end{cases}$$

then the model is biased towards $x_A = x_C$ and $x_B \neq x_C$, that is, x_A and x_C are positively correlated, while x_B and x_C are negatively correlated. From (14.2), we know that the joint probability would be high when the variable assignments satisfy $x_A = x_C$ and $x_B \neq x_C$.

To satisfy the non-negativity, we often use exponential functions to define potential functions

$$\psi_Q(\mathbf{x}_Q) = e^{-H_Q(\mathbf{x}_Q)}. \tag{14.8}$$

$H_Q(\mathbf{x}_Q)$ is real-valued function defined on variable $\mathbf{x}_Q$, usually in the form of

$$H_Q(\mathbf{x}_Q) = \sum_{u,v \in Q, u \neq v} \alpha_{uv} x_u x_v + \sum_{v \in Q} \beta_v x_v, \tag{14.9}$$

where a_{uv} and β_v are parameters. In (14.9), the first term considers pairs of nodes and the second term considers individual nodes.

14.3 Conditional Random Field

Conditional Random Field (CRF) is a discriminative undirected graphical model. In Sect. 14.1, we mentioned that generative models consider joint distributions, while discriminative models consider conditional distributions. The previously introduced HMM and MRF are examples of generative models, and now we introduce CRF as an example of discriminative models.

We can regard CRF as MRF with observed values, or as an extension of logistic regression. See Sect. 3.3.

CRF aims to model the conditional probability of multiple variables given some observed values. To be specific, CRF constructs a conditional probability model $P(\mathbf{y} \mid \mathbf{x})$, where $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ is the observed sequence, and $\mathbf{y} = \{y_1, y_2, \ldots, y_n\}$ is the corresponding label sequence. Note that the label variable $\mathbf{y}$ can be structural, that is, there are some correlations among its components. For example, in part-of-speech tagging problems, the observations are natural language sentences (i.e., sequences of words), and the labels are sequences of part-of-speech tags, as shown in ◘ Figure 14.5(a). In syntactic analysis, the output labels are parse trees, as shown in ◘ Figure 14.5(b).

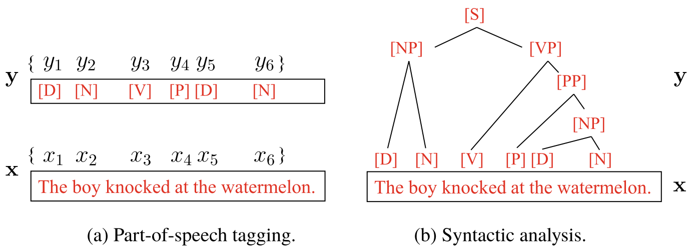
y
{ y_1 y_2 y_3 y_4 y_5 y_6 }
[D] [N] [V] [P] [D] [N]
x
{ x_1 x_2 x_3 x_4 x_5 x_6 }
The boy knocked at the watermelon.
(a) Part-of-speech tagging.
[S]
[NP]
[VP]
[PP]
[NP]
[D] [N] [V] [P] [D] [N]
The boy knocked at the watermelon.
y
x
(b) Syntactic analysis.

Fig. 14.5 The part-of-speech tagging problem and the syntactic analysis problem in natural language processing

Let $G = \langle V, E \rangle$ be an undirected graph in which each node corresponds to one component in the label vector $\mathbf{y}$, where y_v is the component corresponding to node v, and let $n(v)$ denote the adjacent nodes of node v. Then, we say $(\mathbf{y}, \mathbf{x})$ forms a CRF if every label variable y_v in graph G satisfies the Markov property

$$P(y_v \mid \mathbf{x}, \mathbf{y}_{V \setminus \{v\}}) = P(y_v \mid \mathbf{x}, \mathbf{y}_{n(v)}), \tag{14.10}$$

Theoretically, the structure of graph G can be arbitrary as long as it encodes the conditional independence between label variables. In practice, however, especially when modeling label sequences, the most common structure is the chain structure,

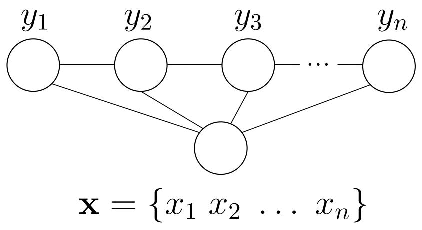

Fig. 14.6 The graph structure of chain-structured CRF

as illustrated in Figure 14.6. Such a CRF is called *chain-structured CRF*, which is the focus of the rest of our discussions.

Similar to how joint probability is defined in MRF, CRF defines conditional probability $P(\mathbf{y} \mid \mathbf{x})$ according to the potential functions and cliques in the graph structure. Given an observed sequence $\mathbf{x}$, the chain-structured CRF in Figure 14.6 mainly contains two types of cliques about label variables, that is, one for single label variable $\{y_i\}$ and the other for adjacent label variables $\{y_{i-1}, y_i\}$. With appropriate potential functions, we can define conditional probability like (14.2). In CRF, by using exponential potential functions and introducing *feature functions*, the conditional probability is defined as

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z} \exp\left(\sum_j \sum_{i=1}^{n-1} \lambda_j t_j(y_{i+1}, y_i, \mathbf{x}, i) + \sum_k \sum_{i=1}^{n} \mu_k s_k(y_i, \mathbf{x}, i)\right), \tag{14.11}$$

where $t_j(y_{i+1}, y_i, \mathbf{x}, i)$ is the *transition feature function* defined on two adjacent labels in the observed sequence, describing the relationship between the two adjacent labels as well as measuring the impact of the observed sequence on them; $s_k(y_i, \mathbf{x}, i)$ is the *status feature function* defined on the label index i in the observed sequence, describing the impact of the observed sequence on the label variable; λ_j and μ_k are parameters; Z is the normalization factor that ensures (14.11) to be a properly defined probability.

We also need to define appropriate feature functions, which are usually real-valued functions that describe empirical properties that are likely or expected to be held about the data. Taking the part-of-speech tagging in Figure 14.5 (a) as an example, we can employ the following transition feature function:

$$t_j(y_{i+1}, y_i, \mathbf{x}, i) = \begin{cases} 1, & \text{if } y_{i+1} = [P],\ y_i = [V] \text{ and } x_i = \text{“}\textit{knock}\text{”}; \\ 0, & \text{otherwise,} \end{cases}$$

which says the labels y_i and y_{i+1} are likely to be $[V]$ and $[P]$ when the ith observation x_i is the word "knock". We can also employ the following status feature function:

$$s_k(y_i, \mathbf{x}, i) = \begin{cases} 1, & \text{if } y_i = [V] \text{ and } x_i = \text{"}\textit{knock}\text{"}; \\ 0, & \text{otherwise,} \end{cases}$$

which says that the label y_i is likely to be $[V]$ if the observation x_i is the word "knock".

By comparing (14.11) and (14.2), we can observe that both CRF and MRF define the probabilities using potential functions on cliques. The difference is that CRF models conditional probabilities, whereas MRF models joint probabilities.

14.4 Learning and Inference

Given the joint probability distributions defined on probabilistic graphical models, we can infer the *marginal distribution* or *conditional distribution* of the target variables. We have encountered conditional distributions previously. For example, in HMM, we infer the conditional probability distribution of an observed sequence $\mathbf{x}$ given certain parameter λ. By contrast, marginal distribution refers to probabilities obtained by summing out or integrating out irrelevant variables. Taking Markov networks as an example, the joint distribution of variables is expressed as the product of maximal cliques' potential functions, and therefore, finding the distribution of variable x given parameter Θ is equivalent to integrating out irrelevant variables in the joint distribution, known as *marginalization*.

In probabilistic graphical models, we also need to determine the parameters of distributions by parameter estimation (i.e., parameter learning), which is often solved via maximum likelihood estimation or maximum a posteriori estimation. If we consider the parameters as variables to be inferred, then the parameter estimation process is similar to the inference process, that is, it can be absorbed into the inference problem. Hence, we mainly discuss the inference methods for the rest of our discussions.

The Bayesian school thinks that unknown parameters are random variables, just like all other variables. Hence, parameter estimation and variable inference can be performed within the same inference framework. The frequentist school disagrees.

To be specific, suppose that the set of variables $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$ in a graphical model can be divided into two disjoint variable sets $\mathbf{x}_E$ and $\mathbf{x}_F$, then the inference problem is about finding the marginal probability $P(\mathbf{x}_F)$ or the conditional probability $P(\mathbf{x}_F \mid \mathbf{x}_E)$. From the definition of conditional probability, we have

$$P(\mathbf{x}_F \mid \mathbf{x}_E) = \frac{P(\mathbf{x}_E, \mathbf{x}_F)}{P(\mathbf{x}_E)} = \frac{P(\mathbf{x}_E, \mathbf{x}_F)}{\sum_{\mathbf{x}_F} P(\mathbf{x}_E, \mathbf{x}_F)}, \tag{14.12}$$

where the joint probability $P(\mathbf{x}_E, \mathbf{x}_F)$ can be obtained from the probabilistic graphical model. Hence, the core of the inference problem is how to efficiently compute the marginal distribution, that is

$$P(\mathbf{x}_E) = \sum_{\mathbf{x}_F} P(\mathbf{x}_E, \mathbf{x}_F). \tag{14.13}$$

There are two types of inference methods for probabilistic graphical models: exact inference methods and approximate inference methods. Exact inference methods compute the exact values of marginal distributions or conditional distributions. However, such methods are often impractical since their computational complexity increases exponentially to the number of maximal cliques. By contrast, approximate inference methods find approximate solutions with tractable time complexity and are more practical in real-world applications. The rest of this section introduces two representative exact inference methods, and we will introduce approximate inference methods in the next section.

14.4.1 Variable Elimination

Exact inference methods are essentially a kind of dynamic programming methods. Such methods attempt to reduce the cost of computing the target probability by exploiting the conditional independence encoded by the graphical model. Among them, *variable elimination* is the most intuitive one and is the basis of other exact inference methods.

We demonstrate variable elimination with the directed graphical model in ◘ Figure 14.7 (a).

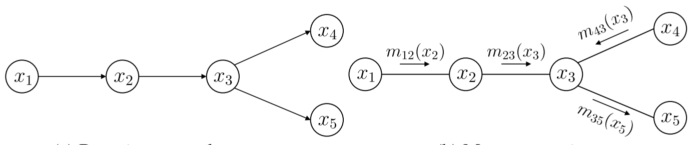

Fig. 14.7 The process of variable elimination and message passing

Suppose that the inference objective is to compute the marginal probability $P(x_5)$. To compute it, we only need to eliminate the variables $\{x_1, x_2, x_3, x_4\}$ by summation, that is

$$\begin{aligned}P(x_5) &= \sum_{x_4}\sum_{x_3}\sum_{x_2}\sum_{x_1} P(x_1, x_2, x_3, x_4, x_5)\\ &= \sum_{x_4}\sum_{x_3}\sum_{x_2}\sum_{x_1} P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_2)P(x_4 \mid x_3)P(x_5 \mid x_3).\end{aligned} \tag{14.14}$$

Using the conditional independence encoded by the directed graphical model.

By doing the summations in the order of $\{x_1, x_2, x_4, x_3\}$, we have

$$\begin{aligned}P(x_5) &= \sum_{x_3} P(x_5 \mid x_3)\sum_{x_4} P(x_4 \mid x_3)\sum_{x_2} P(x_3 \mid x_2)\sum_{x_1} P(x_1)P(x_2 \mid x_1)\\ &= \sum_{x_3} P(x_5 \mid x_3)\sum_{x_4} P(x_4 \mid x_3)\sum_{x_2} P(x_3 \mid x_2)m_{12}(x_2),\end{aligned} \tag{14.15}$$

where $m_{ij}(x_j)$ is an intermediate result in the summation, the subscript i indicates that the term is the summation result with respect to x_i, and the subscript j indicates other variables in the term. We notice that $m_{ij}(x_j)$ is a function of x_j. By repeating the process, we have

$$\begin{aligned}P(x_5) &= \sum_{x_3} P(x_5 \mid x_3)\sum_{x_4} P(x_4 \mid x_3)m_{23}(x_3)\\ &= \sum_{x_3} P(x_5 \mid x_3)m_{23}(x_3)\sum_{x_4} P(x_4 \mid x_3)\\ &= \sum_{x_3} P(x_5 \mid x_3)m_{23}(x_3)m_{43}(x_3)\\ &= m_{35}(x_5).\end{aligned} \tag{14.16}$$

$m_{35}(x_5)$ is a function of x_5 and only depends on the value of x_5.

The above method also applies to undirected graphical models. For example, if we ignore the directions of the edges in ◘ Figure 14.7 (a) and consider it as an undirected graphical model, then we have

$$P(x_1, x_2, x_3, x_4, x_5) = \frac{1}{Z}\psi_{12}(x_1, x_2)\psi_{23}(x_2, x_3)\psi_{34}(x_3, x_4)\psi_{35}(x_3, x_5), \tag{14.17}$$

where Z is the normalization factor. The marginal distribution $P(x_5)$ is given by

$$\begin{aligned}P(x_5) &= \frac{1}{Z}\sum_{x_3}\psi_{35}(x_3, x_5)\sum_{x_4}\psi_{34}(x_3, x_4)\sum_{x_2}\psi_{23}(x_2, x_3)\sum_{x_1}\psi_{12}(x_1, x_2)\\ &= \frac{1}{Z}\sum_{x_3}\psi_{35}(x_3, x_5)\sum_{x_4}\psi_{34}(x_3, x_4)\sum_{x_2}\psi_{23}(x_2, x_3)m_{12}(x_2)\\ &= \cdots\\ &= \frac{1}{Z}m_{35}(x_5).\end{aligned} \tag{14.18}$$

By using the distributive law of multiplication to addition, variable elimination converts the problem of calculating summations of products of multiple variables to the problem of alternately calculating summations and products of some of the variables. Doing so simplifies the calculations by restricting the summations and products to local regions that involve only some of the variables.

Nevertheless, variable elimination has a clear disadvantage: there is a considerable amount of redundancy in the calculations of multiple marginal distributions. Taking the Bayesian network in ◘ Figure 14.7 (a) as an example, if we compute $P(x_4)$ after computing $P(x_5)$, then the calculations of $m_{12}(x_2)$ and $m_{23}(x_3)$ are repetitive when the summation order is $\{x_1, x_2, x_5, x_3\}$.

14.4.2 Belief Propagation

Also known as the *sum-product* algorithm.

The *belief propagation* algorithm avoid repetitive calculations by considering the summation operations in variable elimination as a process of message passing. In variable elimination, a variable x_i is eliminated by the summation operation

$$m_{ij}(x_j) = \sum_{x_i} \psi(x_i, y_j) \prod_{k \in n(i) \setminus j} m_{ki}(x_i), \tag{14.19}$$

where $n(i)$ are the adjacent nodes of x_i. In belief propagation, however, the operation is considered as passing the message $m_{ij}(x_j)$ from x_i to x_j. By doing so, the variable elimination process in (14.15) and (14.16) becomes a message passing process, as illustrated in ◘ Figure 14.7 (b). We see that each message passing operation involves only x_i and its adjacent nodes, and hence the calculations are restricted to local regions.

In belief propagation, a node starts to pass messages after receiving the messages from all other nodes. The marginal distribution of a node is proportional to the product of all received messages, that is

$$P(x_i) \propto \prod_{k \in n(i)} m_{ki}(x_i). \tag{14.20}$$

Taking ◘ Figure 14.7 (b) as an example, x_3 must receive the messages from x_2 and x_4 before it passes the message to x_5, and the message $m_{35}(x_5)$ that x_3 passes to x_5 is exactly $P(x_5)$.

When there is no cycle in the graph, belief propagation can compute marginal distributions of all variables via the following two steps:

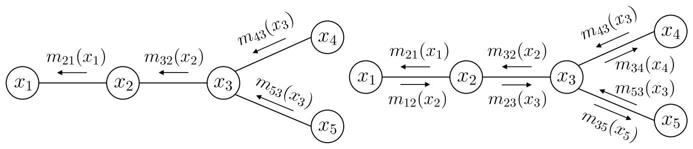

Fig. 14.8 An illustration of the belief propagation algorithm

- Select a root node, and then pass messages from all leaf nodes to the root node until the root node has received messages from all adjacent nodes;
- Pass messages from the root node toward leaf nodes until all leaf nodes have received messages.

Taking Figure 14.7 (a) as an example, let x_1 be the root node, and x_4 and x_5 be the leaf nodes. The two steps of message passing are illustrated in Figure 14.8, where each edge has two messages on it with different directions. From the messages and (14.20), we have the marginal probabilities of all variables.

14.5 Approximate Inference

Exact inference methods are usually computationally expensive, and hence we often use approximate inference methods in practice. Roughly speaking, there are two types of approximate inference methods, namely sampling, which accomplishes approximation by stochastic methods, and deterministic approximations, represented by variational inference.

14.5.1 MCMC Sampling

In many tasks, we are interested in probability distributions just because we need them to calculate some expectations for decision-making. Taking the Bayesian network in Figure 14.7 (a) as an example, the goal of inference could be finding the expectation of x_5. It turns out that, sometimes, it can be more efficient to calculate or approximate the expectations directly without finding the probability distributions first.

The above idea motivates the sampling methods. Suppose our objective is to find the expectation of the function $f(x)$ with respect to the probability density function $p(x)$

Replace integration with summation if x is discrete.

$$\mathbb{E}_p[f] = \int f(x)p(x)dx. \tag{14.21}$$

We can approximate the objective expectation $\mathbb{E}[f]$ by sampling a set of samples $\{x_1, x_2, \ldots, x_N\}$ from $p(x)$ and then compute the mean of $f(x)$ on these samples

Or a distribution related to $p(x)$.

$$\hat{f} = \frac{1}{N}\sum_{i=1}^{N} f(x_i), \tag{14.22}$$

According to the law of large numbers, we can obtain an accurate approximation from the *i.i.d.* samples $\{x_1, x_2, \ldots, x_N\}$ by large-scale sampling. The problem here is how to sample? For example, in probabilistic graphical models, how can we efficiently obtain samples from the probability distribution described by the graphical model?

One of the most commonly used sampling techniques for probabilistic graphical models is the Markov Chain Monte Carlo (MCMC) method. Given the probability density function $p(x)$ of a continuous variable $x \in X$, the probability that x lies in the interval A is

$$P(A) = \int_A p(x)dx. \tag{14.23}$$

If $f : X \mapsto \mathbb{R}$, then the expectation of $f(x)$ is given by

$$p(f) = \mathbb{E}_p[f(X)] = \int_x f(x)p(x)dx. \tag{14.24}$$

However, the integration in (14.24) is not easy to compute when x is not univariate but a high-dimensional multivariate variable $\mathbf{x}$ that follows a complex distribution. Hence, MCMC first constructs some *i.i.d.* samples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ that follow the distribution p, and then obtains an unbiased estimate of (14.24) as

$$\tilde{p}(f) = \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_i). \tag{14.25}$$

Nevertheless, constructing *i.i.d.* samples that follow the distribution p can still be difficult if the probability density function $p(\mathbf{x})$ is complex. The key idea of MCMC is to generate samples by constructing a "Markov chain with stationary distribution p". To be specific, by letting the Markov chain run for a sufficiently long time (i.e., converged to a stationary distribution), the generated samples approximately follow the distribution p. How do we know if the Markov chain has arrived at a stationary state? We say a Markov chain T has arrived at a stationary state with a stationary distribution $p(\mathbf{x}^t)$ once the following stationary condition is met at time point t:

$$p(\mathbf{x}^t)T(\mathbf{x}^{t-1} \mid \mathbf{x}^t) = p(\mathbf{x}^{t-1})T(\mathbf{x}^t \mid \mathbf{x}^{t-1}), \tag{14.26}$$

where $T(\mathbf{x}' \mid \mathbf{x})$ is the state transition probability (i.e., the probability of transitioning from state $\mathbf{x}$ to state $\mathbf{x}'$), and $p(\mathbf{x}^t)$ is the distribution at time point t.

In short, MCMC starts by constructing a Markov chain and let it converge to the stationary distribution, which is exactly the posterior distribution of the parameters to be estimated. Then, it uses the Markov chain to generate the desired samples for further estimations. A vital step in this process is constructing the state transition probabilities of the Markov chain, and different construction methods lead to different MCMC algorithms.

The Metropolis–Hastings (MH) algorithm is an important representative of MCMC methods, which approximates the stationary distribution p via *reject sampling*. The MH algorithm is given in ◘ Figure 14.1. In each round, the MH algorithm draws a candidate state sample $\mathbf{x}^*$ based on the sample $\mathbf{x}^{t-1}$ of the last round, where x^* has a certain probability of being "rejected". Once $\mathbf{x}^*$ converged to a stationary state, from (14.26), we have

The Metropolis–Hastings algorithm is named after the original authors Metropolis et al. (1953) and Hastings (1970), who extended the algorithm to a general form afterwards.

$$p(\mathbf{x}^{t-1})Q(\mathbf{x}^* \mid \mathbf{x}^{t-1})A(\mathbf{x}^* \mid \mathbf{x}^{t-1}) = p(\mathbf{x}^*)Q(\mathbf{x}^{t-1} \mid \mathbf{x}^*)A(\mathbf{x}^{t-1} \mid x^*), \tag{14.27}$$

where $Q(\mathbf{x}^* \mid \mathbf{x}^{t-1})$ is the user-specified prior probability, $A(\mathbf{x}^* \mid \mathbf{x}^{t-1})$ is the probability of accepting $\mathbf{x}^*$, and $Q(\mathbf{x}^* \mid \mathbf{x}^{t-1})A(\mathbf{x}^* \mid \mathbf{x}^{t-1})$ is the state transition probability from state $\mathbf{x}^{t-1}$ to state $\mathbf{x}^*$. To arrive at the stationary state, we just need to set the acceptance probability to

$$A(\mathbf{x}^* \mid \mathbf{x}^{t-1}) = \min\left(1, \frac{p(\mathbf{x}^*)Q(\mathbf{x}^{t-1} \mid \mathbf{x}^*)}{p(\mathbf{x}^{t-1})Q(\mathbf{x}^* \mid \mathbf{x}^{t-1})}\right). \tag{14.28}$$

Algorithm 14.1 Metropolis–Hastings Sampling

```
Input: Prior probability $Q(\mathbf{x}^* \mid \mathbf{x}^{t-1})$.
Process:
 1: Initialize $\mathbf{x}^0$;
 2: for $t = 1, 2, \ldots$ do
 3:     Sample the candidate sample $\mathbf{x}^*$ according to $Q(\mathbf{x}^* \mid \mathbf{x}^{t-1})$;
 4:     Sample the threshold $u$ from range $(0, 1)$ according to uniform distribution;
 5:     if $u \leqslant A(\mathbf{x}^* \mid \mathbf{x}^{t-1})$ then
 6:         $\mathbf{x}^t = \mathbf{x}^*$;
 7:     else
 8:         $\mathbf{x}^t = \mathbf{x}^{t-1}$.
 9:     end if
10: end for
11: return $\mathbf{x}^1, \mathbf{x}^2, \ldots$
Output: A list of sampled samples: $\mathbf{x}^1, \mathbf{x}^2, \ldots$
```

Repeat enough times to arrive at the stationary distribution.

According to (14.28).

In practice, we often discard the samples in the beginning of the list since we wish to use samples generated from the stationary distribution.

See Sect. 7.5.3 for Gibbs sampling.

Gibbs sampling is sometimes considered as a special case of the MH algorithm, since it also obtains samples using Markov chains with the target sampling distribution $p(\mathbf{x})$ as the stationary distribution. Specifically, let $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$ be the set of variables, and $p(\mathbf{x})$ be the objective distribution, then the Gibbs sampling algorithm generates samples by repeating the following steps after initializing $\mathbf{x}$:

(1) Select a variable x_i either randomly or according to a certain ordering;
(2) Compute the conditional probability $p(x_i \mid \mathbf{x}_{\bar{i}})$, where $\mathbf{x}_{\bar{i}} = \{x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_N\}$ is the current value of $\mathbf{x}$ excluding x_i;
(3) Sample a new value of x_i from $p(x_i \mid \mathbf{x}_{\bar{i}})$ and replace the original value.

14.5.2 Variational Inference

Variational inference approximates complex distributions with simple and known distributions. It restricts the type of the approximate distribution, such that the approximate posterior distribution is locally optimal with a deterministic solution.

Before introducing the details of variational inference, let us see a concise way of representing graphical models—*plate notation* (Buntine 1994). Figure 14.9 gives an example. ◘ Figure 14.9 (a) shows that there are N variables $\{x_1, x_2, \ldots, x_N\}$ dependent on the variable $\mathbf{z}$. In ◘ Figure 14.9 (b), the plate notation compactly describes the same relationship, where multiple variables independently generated by the same mechanism are placed in the same rectangle (plate), which allows nesting, and there is a label N indicating the number of repeti-

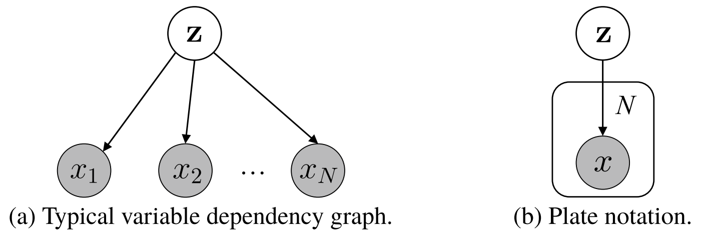

Fig. 14.9 An example of plate notation

tions. Observable or known variables are usually shaded, e.g., x in ◘ Figure 14.9. The plate notation provides a very concise way of representing variable relationships in various learning problems.

In ◘ Figure 14.9 (b), the probability density function of all observable variables x is

Approximate distributions used in variational inference should have nice mathematical properties. They are usually probability density functions of continuous variables.

$$p(\mathbf{x} \mid \Theta) = \prod_{i=1}^{N} \sum_{\mathbf{z}} p(x_i, \mathbf{z} \mid \Theta), \tag{14.29}$$

which has the corresponding log-likelihood function

$$\ln p(\mathbf{x} \mid \Theta) = \sum_{i=1}^{N} \ln \left\{ \sum_{\mathbf{z}} p(x_i, \mathbf{z} \mid \Theta) \right\}, \tag{14.30}$$

where $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$, Θ includes the parameters of the distributions that $\mathbf{x}$ and $\mathbf{z}$ follow.

Generally speaking, the inference and learning task in ◘ Figure 14.9 is mainly estimating the hidden variable $\mathbf{z}$ and the distribution parameter variable Θ, that is, finding $p(\mathbf{z} \mid \mathbf{x}, \Theta)$ and Θ.

The parameters of graphical models are often estimated by maximum likelihood estimation. For (14.30), we can apply the EM algorithm. In the E-step, we infer $p(\mathbf{z} \mid \mathbf{x}, \Theta^t)$ by the parameter variable Θ^t at time point t, and then compute the joint likelihood function $p(\mathbf{x}, \mathbf{z} \mid \Theta)$. In the M-step, we use the current parameter Θ^t obtained in the E-step to find the parameter Θ^{t+1} of the next time point by optimizing the function $\mathcal{Q}(\Theta; \Theta^t)$

See Sect. 7.6 for the EM algorithm.

$$\begin{aligned}\Theta^{t+1} &= \arg\max_{\Theta} \mathcal{Q}(\Theta; \Theta^t) \\ &= \arg\max_{\Theta} \sum_{\mathbf{z}} p(\mathbf{z} \mid \mathbf{x}, \Theta^t) \ln p(\mathbf{x}, \mathbf{z} \mid \Theta).\end{aligned} \tag{14.31}$$

where $\mathcal{Q}(\Theta; \Theta^t)$ is actually the expectation of the joint log-likelihood function $\ln p(\mathbf{x}, \mathbf{z} \mid \Theta)$ with respect to the distribution $p(\mathbf{z} \mid \mathbf{x}, \Theta^t)$. It approximates the log-likelihood function when the distribution $p(\mathbf{z} \mid \mathbf{x}, \Theta^t)$ equals to the ground-truth posterior distribution of the hidden variable $\mathbf{z}$. Hence, the EM algorithm estimates not only the parameter Θ but also the distribution of the hidden variable $\mathbf{z}$.

Note that $p(\mathbf{z} \mid \mathbf{x}, \Theta^t)$ is not necessarily the ground-truth distribution of $\mathbf{z}$ but only an approximate distribution. Let $q(\mathbf{z})$ denote the approximate distribution, we can derive

$$\ln p(\mathbf{x}) = \mathcal{L}(q) + \mathrm{KL}(q \| p), \tag{14.32}$$

where

$$\mathcal{L}(q) = \int q(\mathbf{z}) \ln \left\{ \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})} \right\} d\mathbf{z}, \tag{14.33}$$

$$\mathrm{KL}(q \| p) = -\int q(\mathbf{z}) \ln \frac{p(\mathbf{z} \mid \mathbf{x})}{q(\mathbf{z})} d\mathbf{z}. \tag{14.34}$$

See Appendix C.3 for the KL divergence.

In practice, however, it may be difficult to find $p(\mathbf{z} \mid \mathbf{x}, \Theta^t)$ in the E-step due to the intractable multivariate $\mathbf{z}$, and this is when variational inference comes in handy. We typically assume that $\mathbf{z}$ follows the distribution

$$q(\mathbf{z}) = \prod_{i=1}^{M} q_i(\mathbf{z}_i). \tag{14.35}$$

In other words, we assume that we can decompose the complex multivariate variable $\mathbf{z}$ into a series of independent multivariate variables $\mathbf{z}_i$. With this assumption, the distribution q_i can be made simple or has a good structure. For example, suppose q_i is an *exponential family* distribution, then we have

To make the notation uncluttered, we abbreviate $q_i(\mathbf{z}_i)$ as q_i.

$$\begin{aligned}\mathcal{L}(q) &= \int \prod_i q_i \left\{ \ln p(\mathbf{x}, \mathbf{z}) - \sum_i \ln q_i \right\} d\mathbf{z} \\ &= \int q_j \left\{ \int \ln p(\mathbf{x}, \mathbf{z}) \prod_{i \neq j} q_i d\mathbf{z}_i \right\} d\mathbf{z}_j - \int q_j \ln q_j d\mathbf{z}_j + \mathrm{const} \\ &= \int q_j \ln \tilde{p}(\mathbf{x}, \mathbf{z}_j) d\mathbf{z}_j - \int q_j \ln q_j d\mathbf{z}_j + \mathrm{const},\end{aligned} \tag{14.36}$$

const is a constant.

where

$$\ln \tilde{p}(\mathbf{x}, \mathbf{z}_j) = \mathbb{E}_{i \neq j}[\ln p(\mathbf{x}, \mathbf{z})] + \mathrm{const}, \tag{14.37}$$

$$\mathbb{E}_{i \neq j}[\ln p(\mathbf{x}, \mathbf{z})] = \int \ln p(\mathbf{x}, \mathbf{z}) \prod_{i \neq j} q_i d\mathbf{z}_i. \tag{14.38}$$

Since we are interested in q_j, we can maximize $\mathcal{L}(q)$ with $q_{i\neq j}$ fixed. We notice that (14.36) equals to $-\mathrm{KL}(q_j \parallel \tilde{p}(\mathbf{x}, \mathbf{z}_j))$, that is, $\mathcal{L}(q)$ is maximized when $q_j = \tilde{p}(\mathbf{x}, \mathbf{z}_j)$. Hence, the optimal distribution q_j^* that the variable subset $\mathbf{z}_j$ follows should satisfy

$$\ln q_j^*(\mathbf{z}_j) = \mathbb{E}_{i\neq j}[\ln p(\mathbf{x}, \mathbf{z})] + \text{const}, \tag{14.39}$$

so we have

$$q_j^*(\mathbf{z}_j) = \frac{\exp\left(\mathbb{E}_{i\neq j}[\ln p(\mathbf{x}, \mathbf{z})]\right)}{\int \exp\left(\mathbb{E}_{i\neq j}[\ln p(\mathbf{x}, \mathbf{z})]\right) d\mathbf{z}_j}. \tag{14.40}$$

In other words, with the assumption (14.35), (14.40) provides the best approximation to the ground-truth distribution of the variable subset $\mathbf{z}_j$.

With the assumption (14.35), we can often find a closed form solution to $\mathbb{E}_{i\neq j}[\ln p(\mathbf{x}, \mathbf{z})]$ by properly partitioning variable subsets $\mathbf{z}_j$ and selecting the distribution that q_i follows; hence the hidden variable $\mathbf{z}$ can be inferred efficiently by (14.40). From (14.38), we observe that the estimation of the distribution q_j^* of $\mathbf{z}_j$ not only considers $\mathbf{z}_j$ but also $\mathbf{z}_{i\neq j}$. Since this is achieved by finding the expectation of the joint log-likelihood function $\ln p(\mathbf{x}, \mathbf{z})$ with respect to $\mathbf{z}_{i\neq j}$, such a method is also called the *mean field* method.

"mean" refers to expectation and "field" refers to distribution.

When applying variational inference in practice, the most important thing is to find the proper hidden variable decomposition and the proper distribution hypothesis of each subset of hidden variables. With the hidden variable decomposition and distribution hypotheses, the parameter estimation and inference of probabilistic graphical models can be made by the EM algorithm and the consideration of (14.40). Clearly, the performance of variational inference is subject to the quality of hidden variable decomposition and distribution hypotheses.

14.6 Topic Model

Topic model is a family of generative directed graphical models mainly used for modeling discrete data, e.g., text corpus. As represented by Latent Dirichlet Allocation (LDA), topic models have been widely used in information retrieval and natural language processing.

There are three key concepts in topic models, namely *word*, *document*, and *topic*. A *word* is the basic discrete unit in the data, e.g., an English word in text processing. A *document* is a data object, such as a paper or a Web page, containing a set of words without considering the ordering of words. Such a representation is known as *bag-of-words*. Topic models apply

We can describe an image using bag-of-words by considering the small blocks in the image as words, and then topic models are applicable.

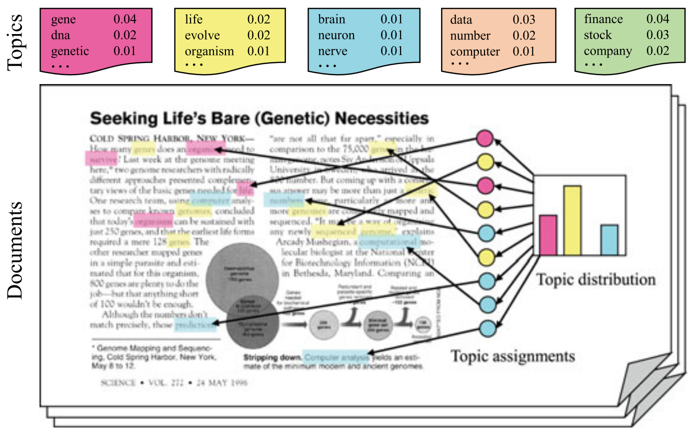

Fig. 14.10 An illustration of the document generation process of LDA

to any data objects that can be described by bag-of-words. A *topic* describes a concept represented by a series of related words together with the probabilities that they appear in the concept.

Figure 14.10 provides an intuitive example of topic model. A topic is like a box containing those words with high probability to appear under the concept of the topic. Suppose that we have a data set of T documents on K topics, where all words in the documents are from a dictionary of N distinct words. The data set (i.e., collection of documents) is denoted by $T \times N$-dimensional vectors $\mathbf{W} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_T\}$, where $w_{t,n}$ (i.e., the nth component of $\mathbf{w}_t \in \mathbb{R}^N$) is the frequency of the word n appeared in the document t. The topics are denoted by K N-dimensional vectors $\boldsymbol{\beta}_k$ $(k = 1, 2, \ldots, K)$, where $\beta_{k,n}$ (i.e., the nth component of $\boldsymbol{\beta}_k \in \mathbb{R}^N$) is the frequency of the word n in the topic k.

14

In practice, we can obtain the word frequency vectors $\mathbf{w}_i$ $(i = 1, 2, \ldots, T)$ by counting the words in documents, though we do not know which topic is mentioned in which document. LDA assumes that each document contains multiple topics that can be modeled by a generative model. More specifically, let $\Theta_t \in \mathbb{R}^K$ denote the proportion of each topic in document t, and $\Theta_{t,k}$ denote the proportion of topic k in document t. Then, LDA assumes a document t is "generated" by the following steps:

Some words are usually excluded, such as *stop words*.

(1) Randomly draw a topic distribution Θ_t from a Dirichlet distribution with parameter $\boldsymbol{\alpha}$;

See Appendix C.1.6 for Dirichlet distribution.

(2) Generate N words for document t by the following steps:
  (a) Obtain a topic assignment $z_{t,n}$ according to topic distribution Θ_t for each word n in document t;

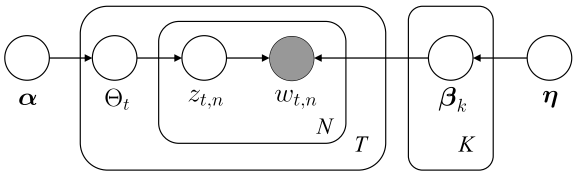

Fig. 14.11 The plate notation of LDA

(b) Generate a word through random sampling according to the word frequency distribution $\boldsymbol{\beta}_k$ corresponding to topic assignment $z_{t,n}$.

The above document generation process is illustrated in ◘ Figure 14.10. Note that a generated document will have different proportions of topics (step 1), and each word in the document comes from a topic (step 2b) generated according to the topic distribution (step 2a).

The plate notation in ◘ Figure 14.11 describes the relationships between variables, where word frequency $w_{t,n}$ is the only observed variable that depends on topic assignment $z_{t,n}$ and the corresponding word frequency distribution $\boldsymbol{\beta}_k$. The topic assignment $z_{t,n}$ depends on topic distribution Θ_t, which depends on a parameter $\boldsymbol{\alpha}$. The word frequency distribution $\boldsymbol{\beta}_k$ depends on a parameter $\boldsymbol{\eta}$. Then, the probability distribution of LDA is

$$p(\mathbf{W}, \mathbf{z}, \boldsymbol{\beta}, \Theta \mid \boldsymbol{\alpha}, \boldsymbol{\eta}) = \prod_{t=1}^{\top} p(\Theta_t \mid \boldsymbol{\alpha}) \prod_{i=1}^{K} p(\boldsymbol{\beta}_k \mid \boldsymbol{\eta}) \left(\prod_{n=1}^{N} P(w_{t,n} \mid z_{t,n}, \boldsymbol{\beta}_k) P(z_{t,n} \mid \Theta_t) \right), \tag{14.41}$$

where $p(\Theta_t \mid \boldsymbol{\alpha})$ is usually set to a K-dimensional Dirichlet distribution with a parameter $\boldsymbol{\alpha}$, and $p(\boldsymbol{\beta}_k \mid \boldsymbol{\eta})$ is usually set to an N-dimensional Dirichlet distribution with a parameter $\boldsymbol{\eta}$. For example

$$p(\Theta_t \mid \boldsymbol{\alpha}) = \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_k \Theta_{t,k}^{\alpha_k - 1}, \tag{14.42}$$

where $\Gamma(\cdot)$ is the Gamma function. Clearly, $\boldsymbol{\alpha}$ and $\boldsymbol{\eta}$ in (14.41) are the model parameters to be determined.

See Appendix C.1.5 for Gamma function.

Given a data set $\mathbf{W} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_T\}$, the parameters of LDA can be estimated by maximum likelihood estimation, that is, finding $\boldsymbol{\alpha}$ and $\boldsymbol{\eta}$ by maximizing the log-likelihood

The word frequencies in training documents.

$$LL(\boldsymbol{\alpha}, \boldsymbol{\eta}) = \sum_{t=1}^{T} \ln p(\mathbf{w}_t \mid \boldsymbol{\alpha}, \boldsymbol{\eta}). \tag{14.43}$$

However, it is difficult to solve (14.43) directly since $p(\mathbf{w}_t \mid \boldsymbol{\alpha}, \boldsymbol{\eta})$ is not easy to compute. In practice, we often use variational inference to find an approximate solution.

Once $\boldsymbol{\alpha}$ and $\boldsymbol{\eta}$ are found, we can use word frequency $w_{t,n}$ to infer the topic structure of a document, that is, inferring Θ_t, $\boldsymbol{\beta}_k$, and $z_{t,n}$ by solving

$$p(\mathbf{z}, \boldsymbol{\beta}, \Theta \mid \mathbf{W}, \boldsymbol{\alpha}, \boldsymbol{\eta}) = \frac{p(\mathbf{W}, \mathbf{z}, \boldsymbol{\beta}, \Theta \mid \boldsymbol{\alpha}, \boldsymbol{\eta})}{p(\mathbf{W} \mid \boldsymbol{\alpha}, \boldsymbol{\eta})}. \tag{14.44}$$

Similarly, (14.44) is hard to solve since $p(\mathbf{w} \mid \boldsymbol{\alpha}, \boldsymbol{\eta})$ is not easy to compute. In practice, Gibbs sampling or variational inference is often employed to find an approximate solution.

14.7 Further Reading

Koller and Friedman (2009) is a book dedicated to probabilistic graphical models. Pearl (1982) initialized the study on Bayesian networks, and (Pearl 1988) summarized relevant studies in the early days. Geman and Geman (1984) proposed Markov random fields, which are often used together with Bayesian networks in real-world applications. Hidden Markov model and its application to speech recognition can be found in Rabiner (1989). Lafferty et al. (2001) proposed conditional random fields, and more information can be found in Sutton and McCallum (2012).

Pearl (1986) proposed the belief propagation algorithm as an exact inference method, and it was derived into many approximate inference methods. For typical cyclic graphs, the initialization and message passing mechanisms of belief propagation need to be modified, resulting in the Loopy Belief Propagation algorithm (Murphy et al. 1999). Its theoretical properties are still unclear, though some progress can be found in Mooij and Kappen (2007), Weiss (2000). Some cyclic graphs can be described by *factor graphs* (Kschischang et al. 2001), and then converted into factor trees for belief propagation. There are attempts (Lauritzen and Spiegelhalter 1988) to enable belief propagation for arbitrary graph structures. Recent advances in parallel computing motivated studies on parallelized belief propagation. For example, Gonzalez et al. (2009) proposed the concept of τ_ϵ approximate inference and designed a multi-core parallelized belief propagation algorithm with a time complexity decreasing linearly to the number of cores.

The modeling and inference techniques, particularly variational inference, for graphical models became mature in the middle 1990s, and (Jordan 1998) summarized the major outcomes in that period. See (Wainwright and Jordan 2008) for more information about variational inference.

An advantage of graphical models is that we can intuitively and quickly define models for specific learning problems. A prominent representative of such methods is LDA (Blei et al. 2003), which has many variants (Blei 2012). One research direction of probabilistic graphic models is to make the model adaptive to the data (i.e., *non-parametric* methods), such as Hierarchical Dirichlet Processes (Teh et al. 2006) and Infinite Latent Feature Model (Ghahramani and Griffiths 2006).

Non-parametric means parameters, such as assumptions on data distribution, are not required to be specified, and this is an important advancement of Bayesian learning. See Sect. 7.7 for Bayesian learning.

Not all topic models are Bayesian learning methods. For example, Probabilistic Latent Semantic Analysis (PLSA) (Hofmann 2001) is a probabilistic extension of Latent Semantic Analysis (LSA).

LSA is a variant of SVD for textual data.

Monte Carlo methods are a family of numerical methods developed in the 1940s based on random numbers, and probability and statistical theory. MCMC is the combination of Markov chains and the Monte Carlo method, and (Pearl 1987) introduced it to Bayesian network inference. See Neal (1993) for more information about applying MCMC to probabilistic inference. See Andrieu et al. (2003), Gilks et al. (1996) for more information about MCMC.

See “Break Time” of Chap. 11 for Monte Carlo methods.

Exercises

14.1 Use plate notation to represent conditional random field and naïve Bayes classifier.

14.2 Prove the local Markov property in graphical models: a variable is conditionally independent of other variables given the adjacent variables.

14.3 Prove the pairwise Markov property in graphical models: two non-adjacent variables are conditionally independent given all other variables.

14.4 Explain why the potential functions are only needed for the maximal cliques in Markov random field.

14.5 Discuss the similarities and differences between conditional random field and logistic regression.

14.6 Prove that the computational complexity of the variable elimination method increases exponentially to the number of maximal cliques in graphical models, but does not necessarily increase exponentially to the number of nodes.

14.7 Gibbs sampling can be seen as a special case of the Metropolis–Hastings algorithm, but it does not take the reject sampling strategy. Discuss the advantages of doing so.

14.8 Mean field is an approximate inference method. Considering (14.32), discuss the difference between the approximated problem solved by the mean field method and the original problem, and discuss how to select the prior probability of variables in practice.

14.9 * Download or implement the LDA algorithm, and apply it to a novel book (e.g., Robinson Crusoe) to see how the topics evolve over chapters.

14.10 * Design an improved LDA algorithm that does not require the predefined number of topics.

Break Time

Short Story: Judea Pearl—Pioneer of Probabilistic Graphical Models

We must mention the Israeli-American computer scientist Judea Pearl (1936–) when talking about graphical probabilistic models. Pearl was born in Tel Aviv. After obtaining his B.S. degree from the Technion in 1960, Pearl emigrated to the United States to continue his study at the Newark College of Engineering and received his Ph.D. degree in Electrical Engineering from the Polytechnic Institute of Brooklyn in 1965. After graduation, he worked at RCA Research Laboratories on superconductive amplifiers and storage devices, and later on, in 1970, he joined the University of California, Los Angeles.

Research on artificial intelligence in the early days focused on symbolism learning and logic reasoning, which can hardly process and represent uncertainties in a quantitative manner. In the 1970s, Pearl introduced probabilistic methods into artificial intelligence and invented a series of techniques, including Bayesian network and Belief propagation, which led to the framework of probabilistic graphical models. By using Bayesian networks as a tool, Pearl created a new research area known as causal inference. Pearl received the ACM/AAAI Allen Newell Award in 2003, and later on, the Turing Award in 2011 for his fundamental contributions to artificial intelligence through the development of a calculus for probabilistic and causal reasoning. ACM commented that "*Pearl's work not only revolutionized the field of artificial intelligence, but also became an important tool for many other branches of engineering and the natural sciences.*" In 2011, Pearl also received the Lakatos Award, which is the most prestigious award in the philosophy of science.

See Sect. 1.5.

The ACM/AAAI Allen Newell Award is presented to people for career contributions that have breadth within computer science, or that bridge computer science and other disciplines. The award is named after Allen Newell (1927–1992), who is a Turing Award winner and a pioneer in the field of artificial intelligence. The second recipient of this award from the field of machine learning is Michael Jordan, who received the award in 2009.

References

Andrieu C, Freitas ND, Doucet A, Jordan MI (2003) An introduction to MCMC for machine learning. Mach Learn 50(1–2):5–43

Blei DM (2012) Probabilistic topic models. Commun ACM 55(4):77–84

Blei DM, Ng A, Jordan MI (2003) Latent Dirichlet allocation. J Artif Intell Res 3:993–1022

Buntine W (1994) Operations for learning with graphical models. J Artif Intell Res 2:159–225

Geman S, Geman D (1984) Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. IEEE Trans Patt Anal Mach Intell 6(6):721–741

Ghahramani Z, Griffiths TL (2006) Infinite latent feature models and the Indian buffet process. In: Weiss Y, Scholkopf B, Platt JC (eds) Advances in Neural Information Processing Systems 18 (NIPS). MIT Press, Cambridge, MA, pp 475–482

Gilks WR, Richardson S, Spiegelhalter DJ (1996) Markov chain monte carlo in practice. Chapman & Hall/CRC, Boca Raton, FL

Gonzalez JE, Low Y, Guestrin C (2009) Residual splash for optimally parallelizing belief propagation. In Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS), Clearwater Beach, FL, pp 177–184

Hastings WK (1970) Monte Carlo sampling methods using Markov chains and their applications. Biometrica 57(1):97–109

Hofmann T (2001) Unsupervised learning by probabilistic latent semantic analysis. Mach Learn 42(1):177–196

Jordan MI (ed) (1998) Learning in graphical models. Kluwer, Dordrecht, Netherlands

Koller D, Friedman N (2009) Probabilistic graphical models: principles and techniques. MIT Press, Cambridge, MA

Kschischang FR, Frey BJ, Loeliger H-A (2001) Factor graphs and the sum-product algorithm. IEEE Trans Inf Theory 47(2):498–519

Lafferty JD, McCallum A, Pereira FCN (2001) Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the 18th International Conference on Machine Learning (ICML), Williamstown, MA, pp 282–289

Lauritzen SL, Spiegelhalter DJ (1988) Local computations with probabilities on graphical structures and their application to expert systems. J R Stat Soc -Ser B 50(2):157–224

Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953) Equations of state calculations by fast computing machines. J Chem Phys 21(6):1087–1092

Mooij JM, Kappen HJ (2007) Sufficient conditions for convergence of the sum-product algorithm. IEEE Trans Inf Theory 53(12):4422–4437

Murphy KP, Weiss Y, Jordan MI (1999) Loopy belief propagation for approximate inference: an empirical study. In: Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI), Stockholm, Sweden, pp 467–475

Neal RM (1993) Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto

Pearl J (1982) Asymptotic properties of minimax trees and game-searching procedures. In: Proceedings of the 2nd National Conference on Artificial Intelligence (AAAI), Pittsburgh, PA

Pearl J (1986) Fusion, propagation and structuring in belief networks. Artif Intell 29(3):241–288

Pearl J (1987) Evidential reasoning using stochastic simulation of causal models. Artif Intell 32(2):245–258

Pearl J (1988) Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmanns, San Francisco, CA

Rabiner LR (1989) A tutorial on hidden Markov model and selected applications in speech recognition. Proc IEEE 77(2):257–286

Sutton C, McCallum A (2012) An introduction to conditional random fields. Found Trends Mach Learn 4(4):267–373

Teh YW, Jordan MI, Beal MJ, Blei DM (2006) Hierarchical Dirichlet processes. J Am Stat Assoc 101(476):1566–1581

Wainwright MJ, Jordan MI (2008) Graphical models, exponential families, and variational inference. Found Trends Mach Learn 1(1–2):1–305

Weiss Y (2000) Correctness of local probability propagation in graphical models with loops. Neural Comput 12(1):1–41