In [66], a hierarchial Pitman-Yor n-gram language model is proposed. It turns out that the proposed model has the best performance compared with the state-of-the-art methods, and has demonstrated that Bayesian approach can be competitive with the best smoothing techniques in languages modeling.

3.4 Others

There are many other stochastic processes that can be used in Bayesian nonparametric models, such as Indian buffet process [27], Beta process [69], Gaussian process [58] for infinite Gaussian mixture model, Gaussian process regression, and so on. We now briefly introduce them in the following, and the readers can refer to the references for more details.

- Indian buffet process. In mixture models, one data point can only belong to one cluster, with the probability determined by the mixing proportions. However, sometimes one data point can have multiple features. For example, a person can participate in a number of communities, all with a large strength. Indian buffet process is a stochastic process that can define the infinite-dimensional features for data points. It has a metaphor of people choosing (infinite) dishes arranged in a line in Indian buffet restaurant, which is where the name "Indian buffet process" is from.

- Beta process. As mentioned in [69], a beta process (BP) plays the role for the Indian buffet process that the Dirichlet process plays for the Chinese restaurant process. Also, a hierarchical beta process (hBP)-based method is proposed in [69] for the document classification task.

- Gaussian process. Intuitively, a Gaussian process (GP) extends a multivariate Gaussian distribution to the one with infinite dimensionality, similar to DP's role to Dirichlet distribution. Any finite subset of the random variables in a GP follows a multivariate Gaussian distribution. The applications for GP include Gaussian process regression, Gaussian process classification, and so on, which are discussed in [58].

4. Graphical Models

A Graphical model [32, 36] is a probabilistic model for which a graph denotes the conditional independence structure between random variables. Graphical model provides a simple way to visualize the structure of a probabilistic model and can be used to design and motivate new

models. In a probabilistic graphical model, each node represents a random variable, and the links express probabilistic relationships between these variables. The graph then captures the way in which the joint distribution over all of the random variables can be decomposed into a product of factors each depending only on a subset of the variables. There are two branches of graphical representations of distributions that are commonly used: *directed* and *undirected*. In this chapter, we discuss the key aspects of graphical models and their applications in text mining.

4.1 Bayesian Networks

Bayesian networks (BNs), also known as *belief networks* (or Bayes nets for short), belong to the *directed graphical models*, in which the links of the graphs have a particular directionality indicated by arrows.

4.1.1 Overview. Formally, BNs are directed acyclic graphs (DAG) whose nodes represent random variables, and edges represent conditional dependencies. For example, a link from x to y can be informally interpreted as indicating that x "causes" y.

Conditional Independence. The simplest conditional independence relationship encoded in a BN can be stated as follows: a node is conditionally independent of its non-descendants given its parents, where the parent relationship is with respect to some fixed topological ordering of the nodes. This is also called *local Markov property*, denoted by $X_v \perp\!\!\!\perp X_{V \setminus de(v)} | X_{pa(v)}$ for all $v \in V$, where $de(v)$ is the set of descendants of v. For example, as shown in Figure 8.1(a), we obtain $x_1 \perp\!\!\!\perp x_3 | x_2$.

Factorization Definition. In a BN, the joint probability of all random variables can be factored into a product of density functions for all of the nodes in the graph, conditional on their parent variables. More precisely, for a graph with n nodes (denoted as $x_1, ..., x_n$), the joint distribution is given by:

$$p(x_1, ..., x_n) = \Pi_{i=1}^{n} p(x_i | pa_i), \tag{8.1}$$

where pa_i is the set of parents of node x_i. By using the chain rule of probability, the above joint distribution can be written as a product of conditional distributions, given the topological order of these random variables:

$$p(x_1, ..., x_n) = p(x_1)p(x_2|x_1)...p(x_n|x_{n-1}, ..., x_1). \tag{8.2}$$

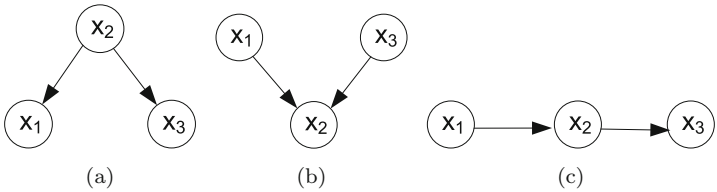


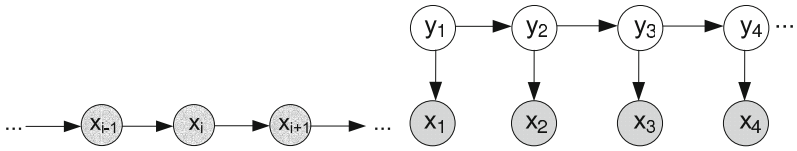
Figure 8.1. Examples of directed acyclic graphs describing the joint distributions.

The difference between the two expressions is the *conditional independence* of the variables encoded in a BN, that variables are conditionally independent of their non-descendants given the values of their parent variables.

Consider the graph shown in Figure 8.1, we can go from this graph to the corresponding representation of the joint distribution written in terms of the product of a set of conditional probability distributions, one for each node in the graph. The joint distributions for Figure 8.1(a)-(c) are therefore $p(x_1, x_2, x_3) = p(x_1|x_2)p(x_2)p(x_3|x_2)$, $p(x_1, x_2, x_3) = p(x_1)p(x_2|x_1, x_3)p(x_3)$, and $p(x_1, x_2, x_3) = p(x_1)p(x_2|x_1)p(x_3|x_2)$, respectively.

4.1.2 The Learning Algorithms. Because a BN is a complete model for the variables and their relationships, a complete joint probability distribution (JPD) over all the variables is specified for a model. Given the JPD, we can answer all possible inference queries by summing out (marginalizing) over irrelevant variables. However, the JPD has size $O(2^n)$, where n is the number of nodes, and we have assumed each node can have 2 states. Hence summing over the JPD takes exponential time. The most common *exact inference* method is **Variable Elimination** [19]. The general idea is to perform the summation to eliminate the non-observed non-query variables one by one by distributing the sum over the product. The reader can refer to [19] for more details. Instead of exact inference, a useful *approximate algorithm* called *Belief propagation* [46] is commonly used on general graphs including Bayesian network, which will be introduced in Section 4.3.

4.1.3 Applications in Text Mining. Bayesian networks have been widely used in many applications in text mining, such as spam filtering [61] and information retrieval [20]. In [61], a Bayesian approach is proposed to identify spam email by making use of a naive Bayes classifier. The intuition is that particular words have particular probabilities of occurring in spam emails and in legitimate emails. For



(a) Regular Markov model (b) Hidden Markov model

Figure 8.2. Graphical structures for the regular and hidden Markov model.

instance, the words "free" and "credit" will frequently appear in spam emails, but will seldom occur in other emails. To train the filter, the user must manually indicate whether an email is spam or not for a training set. With such a training dataset, Bayesian spam filters will learn a spam probability for each word, e.g., a high spam probability for the words "free" and "credit", and a relatively low spam probability for words such as the names of friends. Then, the email's spam probability is computed over all words in the email, and if the total exceeds a certain threshold, the filter will mark the email as a spam.

4.2 Hidden Markov Models

In a regular Markov model as Figure 8.2(a), the state x_i is directly visible to the observer, and therefore the state transition probabilities $p(x_i|x_{i-1})$ are the only parameters. Based on the Markov property, the joint distribution for a sequence of n observations under this model is given by

$$p(x_1, ..., x_n) = p(x_1) \prod_{i=2}^{n} p(x_i|x_{i-1}). \tag{8.3}$$

Thus if we use such a model to predict the next observation in a sequence, the distribution of predictions will depend on the value of the immediately preceding observation and will be independent of all earlier observations, conditional on the preceding observation.

4.2.1 Overview. A hidden Markov model (HMM) can be considered as the simplest dynamic Bayesian network. In a hidden Markov model, the state y_i is not directly visible, and only the output x_i is visible, which is dependent on the state. The hidden state space is discrete, and is assumed to consist of one of N possible values, which is also called latent variable. The observations can be either discrete or continuous, which are typically generated from a categorical distribution or a Gaussian distribution. Generally, a HMM can be considered as a

generalization of a *mixture model* where the hidden variables are related through a Markov process rather than independent of each other.

Suppose the latent variables form a first-order Markov chain as shown in Figure 8.2(b). The random variable y_t is the hidden state at time t, and the random variable x_t is the observation at time t. The arrows in the figure denote conditional dependencies. From the diagram, it is clear that y_{t-1} and y_{t+1} are independent given y_t, so that $y_{t+1} \perp\!\!\!\perp y_{t-1}|y_t$. This is the key conditional independence property, which is called the *Markov property*. Similarly, the value of the observed variable x_t only depends on the value of the hidden variable y_t. Then, the joint distribution for this model is given by

$$p(x_1, ..., x_n, y_1, ..., y_n) = p(y_1) \prod_{t=2}^{n} p(y_t|y_{t-1}) \prod_{t=1}^{n} p(x_t|y_t), \tag{8.4}$$

where $p(y_t|y_{t-1})$ is the state transition probability, and $p(x_t|y_t)$ is the observation probability.

4.2.2 The Learning Algorithms. Given a set of possible states $\Omega_Y = \{q_1, ..., q_N\}$ and a set of possible observations $\Omega_X = \{o_1, ..., o_M\}$. The parameter learning task of HMM is to find the best set of state transition probabilities $A = \{a_{ij}\}$, $a_{ij} = p(y_{t+1} = q_j|y_t = q_i)$ and observation probabilities $B = \{b_i(k)\}$, $b_i(k) = p(x_t = o_k|y_t = q_i)$ as well as the initial state distribution $\Pi = \{\pi_i\}, \pi_i = p(y_0 = q_i)$ for a set of output sequences. Let $\Lambda = \{A, B, \Pi\}$ denote the parameters for a given HMM with fixed Ω_Y and Ω_X. The task is usually to derive the *maximum likelihood estimation* of the parameters of the HMM given the set of output sequences. Usually a local maximum likelihood can be derived efficiently using the *Baum-Welch algorithm* [5], which makes use of *forward-backward algorithm* [55], and is a special case of the generalized EM algorithm [22].

Given the parameters of the model Λ, there are several typical inference problems associated with HMMs, as outlined below. One common task is to compute the probability of a particular output sequence, which requires summation over all possible state sequences: The probability of observing a sequence $X_1^T = o_1, ..., o_T$ of length T is given by $P(X_1^T|\Lambda) = \sum_{Y_1^T} P(X_1^T|Y_1^T, \Lambda)P(Y_1^T|\Lambda)$, where the sum runs over all possible hidden-node sequences $Y_1^T = y_1, ..., y_T$.

This problem can be handled efficiently using the **forward-backward algorithm**. Before we describe the algorithm, let us define the forward (alpha) values and backward (beta) values as follows: $\alpha_t(i) = P(x_1 = o_1, ..., x_t = o_t, y_t = q_i|\Lambda)$ and $\beta_t(i) = P(x_{t+1} = o_{t+1}, ..., x_T = o_T|y_t =$

$q_i, \Lambda)$. Note the forward values enable us to solve the problem through marginalizing, then we obtain

$$P(X_1^T|\Lambda) = \sum_{i=1}^{N} P(o_1, ..., o_T, y_T = q_i|\Lambda) = \sum_{i=1}^{N} \alpha_T(i).$$

The forward values can be computed efficiently with the principle of *dynamic programming*:

$$\begin{aligned} \alpha_1(i) &= \pi_i b_i(o_1), \\ \alpha_{t+1}(j) &= \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}). \end{aligned}$$

Similarly, the backward values can be computed as

$$\begin{aligned} \beta_T(i) &= 1, \\ \beta_t(i) &= \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j). \end{aligned}$$

The backward values will be used in the Baum-Welch algorithm.

Given the parameters of HMM and a particular sequence of observations, another interesting task is to compute the most likely sequence of states that could have produced the observed sequence. We can find the most likely sequence by evaluating the joint probability of both the state sequence and the observations for each case. For example, in part-of-speech (POS) tagging [37], we observe a token (word) sequence $X_1^T = o_1, ..., o_T$, and the goal of POS tagging is to find a stochastic optimal tag sequence $Y_1^T = y_1 y_2 ... y_T$ that maximizes $P(Y_1^n, X_1^n)$. In general, finding the most likely explanation for an observation sequence can be solved efficiently using the **Viterbi algorithm** [24] by the recurrence relations:

$$\begin{aligned} V_1(i) &= b_i(o_1)\pi_i, \\ V_t(j) &= b_i(o_t) \max_i \left(V_{t-1}(i) a_{ij}\right). \end{aligned}$$

Here $V_t(j)$ is the probability of the most probable state sequence responsible for the first t observations that has q_j as its final state. The Viterbi path can be retrieved by saving back pointers that remember which state $y_t = q_j$ was used in the second equation. Let $Ptr(y_t, q_i)$ be the function that returns the value of y_{t-1} used to compute $V_t(i)$ as follows:

$$\begin{aligned} y_T &= \arg \max_{q_i \in \Omega_Y} V_T(i), \\ y_{t-1} &= Ptr(y_t, q_i). \end{aligned}$$

The complexity of this algorithm is $O(T \times N^2)$, where T is the length of observed sequence and N is the number of possible states.

Now we need a method of adjusting the parameters Λ to maximize the likelihood for a given training set. The **Baum-Welch algorithm** [5] is used to find the unknown parameters of HMMs, which is a particular case of a generalized EM algorithms [22]. We start by choosing arbitrary values for the parameters, then compute the expected frequencies given the model and the observations. The expected frequencies are obtained by weighting the observed transitions by the probabilities specified in the current model. The expected frequencies obtained are then substituted for the old parameters and we iterate until there is no improvement. On each iteration we improve the probability of being observed from the model until some limiting probability is reached. This iterative procedure is guaranteed to converge to a local maximum [56].

4.2.3 Applications in Text Mining. HMM models have been applied to a wide variety of problems in information extraction and natural language processing, which have been introduced in Chapter 2, including POS tagging [37] and named entity recognition [6]. Taking POS tagging [37] as an example, each word is labeled with a tag indicating its appropriate part of speech, resulting in annotated text, such as: "[VB heat] [NN water] [IN in] [DT a] [JJ large] [NN vessel]". Given a sequence of words X_1^n, e.g., "heat water in a large vessel", the task is to assign a sequence of labels Y_1^n, e.g., "VB NN IN DT JJ NN", for the words. Based on HMM models, we can determine the sequence of labels by maximizing a joint probability distribution $p(X_1^n, Y_1^n)$.

With the success of HMMs in POS tagging, it is natural to develop a variant of an HMM for the name entity recognition task [6]. Intuitively, the locality of phenomena may indicate names in the text, such as titles like "Mr." preceding a person's name. The HMM classifier models such kinds of dependencies, and performs sequence classification by assigning each word to one of the named entity types. The states in the HMM are organized into regions, one region for each type of named entity. Within each of the regions, a statistical bi-gram language model is used to compute the likelihood of words occurring within that region (named entity type). The transition probabilities are computed by deleted interpolation, and the decoding is done through the Viterbi algorithm.

4.3 Markov Random Fields

Now we turn to another major class of graphical models that are described by undirected graphs and that again specify both a factorization and a set of conditional independence relations.

4.3.1 Overview. A Markov random field (MRF), also known as an undirected graphical model [35], has a set of nodes each of which corresponds to a variable or group of variables, as well as a set of links each of which connects a pair of nodes. The links are undirected, that is they do not carry arrows.

Conditional Independence. Given three sets of nodes, denoted A, B, and C, in an undirected graph G, if A and B are separated in G after removing a set of nodes C from G, then A and B are conditionally independent given the random variables C, denoted as $A \perp\!\!\!\perp B|C$. The conditional independence is determined by simple graph separation. In other words, a variable is conditionally independent of all other variables given its neighbors, denoted as $X_v \perp\!\!\!\perp X_{V\setminus\{v\cup ne(v)\}}|X_{ne(v)}$, where $ne(v)$ is the set of neighbors of v. In general, an MRF is similar to a Bayesian network in its representation of dependencies, and there are some differences. On one hand, an MRF can represent certain dependencies that a Bayesian network cannot (such as cyclic dependencies); on the other hand, MRF cannot represent certain dependencies that a Bayesian network can (such as induced dependencies).

Clique Factorization. As the Markov properties of an arbitrary probability distribution can be difficult to establish, a commonly used class of MRFs are those that can be factorized according to the cliques of the graph. A *clique* is defined as a subset of the nodes in a graph such that there exists a link between all pairs of nodes in the subset. In other words, the set of nodes in a clique is fully connected.

We can therefore define the factors in the decomposition of the joint distribution to be functions of the variables in the cliques. Let us denote a clique by C and the set of variables in that cliques by x_C. Then the joint distribution is written as a product of *potential functions* $\psi_C(x_C)$ over the maximal cliques of the graph

$$p(x_1, x_2, ..., x_n) = \frac{1}{Z}\Pi_C \psi_C(x_C),$$

where the *partition function* Z is a normalization constant and is given by $Z = \sum_x \Pi_C \psi_C(x_C)$. In contrast to the factors in the joint distribution for a directed graph, the potentials in an undirected graph do

not have a specific probabilistic interpretation. Therefore, how to motivate a choice of potential function for a particular application seems to be very important. One popular potential function is defined as $\psi_C(x_C) = \exp(-\epsilon(x_C))$, where $\epsilon(x_C) = -\ln \psi_C(x_C)$ is an *energy function* [45] derived from statistical physics. The underlying idea is that the probability of a physical state depends inversely on its energy. In the logarithmic representation, we have

$$p(x_1, x_2, ..., x_n) = \frac{1}{Z} \exp\left(-\sum_C \epsilon(x_C)\right).$$

The joint distribution above is defined as the product of potentials, and so the total energy is obtained by adding the energies of each of the maximal cliques.

A *log-linear model* is a Markov random field with feature functions f_k such that the joint distribution can be written as

$$p(x_1, x_2, ..., x_n) = \frac{1}{Z} \exp\left(\sum_{k=1}^{K} \lambda_k f_k(x_{C_k})\right),$$

where $f_k(x_{C_k})$ is the function of features for the clique C_k, and λ_k is the weight vector of features. The log-linear model provides a much more compact representation for many distributions, especially when variables have large domains such as text.

4.3.2 The Learning Algorithms. In MRF, we may compute the conditional distribution of a set of nodes given values A to another set of nodes B by summing over all possible assignments to $v \notin A,\ B$, which is called *exact inference*. However, the exact inference is computationally intractable in the general case. Instead, approximation techniques such as MCMC approach [3] and loopy *belief propagation* [46, 8] are often more feasible in practice. In addition, there are some particular subclasses of MRFs that permit efficient maximum-a-posterior (MAP) estimation, or more likely assignment, inference, such as associate networks. Here we will briefly describe belief propagation algorithm.

Belief propagation is a message passing algorithm for performing inference on graphical models, including Bayesian networks and MRFs. It calculates the marginal distribution for each unobserved node, conditional on any observed nodes. Generally, belief propagation operates on a factor graph, which is a bipartite graph containing nodes corresponding to variables V and factors U, with edges between variables and the factors in which they appear. Any Bayesian network and MRF can be represented as a factor graph. The algorithm works by passing

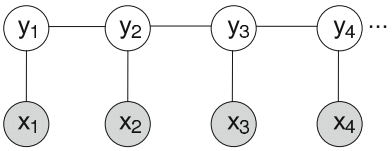


Figure 8.3. Graphical structure for the conditional random field model.

real valued function called *messages* along the edges between the nodes. Taking pairwise MRF as an example, let $m_{ij}(x_j)$ denote the message from node i to node j, and a high value of $m_{ij}(x_j)$ means that node i "believes" the marginal value $P(x_j)$ to be high. Usually the algorithm first initializes all messages to uniform or random positive values, and then updates message from i to j by considering all messages flowing into i (except for message from j) as follows:

$$m_{ij}(x_j) = \sum_{x_i} f_{ij}(x_i, x_j) \prod_{k \in ne(i) \setminus j} m_{ki}(x_i),$$

where $f_{ij}(x_i, x_j)$ is the potential function of the pairwise clique. After enough iterations, this process is likely to converge to a consensus. Once messages have converged, the marginal probabilities of all the variables can be determines by

$$p(x_i) \propto \prod_{k \in ne(i)} m_{ki}(x_i).$$

The reader can refer to [46] for more details. The main cost is the message update equation, which is $O(N^2)$ for each pair of variables (N is the number of possible states).

4.3.3 Applications in Text Mining. Recently, MRF has been widely used in many text mining tasks, such as text categorization [16] and information retrieval [44]. In [44], MRF is used to model the term dependencies using the joint distribution over queries and documents. The model allows for arbitrary text features to be incorporated as evidence. In this model, an MRF is constructed from a graph G, which consists of query nodes q_i and a document node D. The authors explore full independence, sequential dependence, and full dependence variants of the model. Then, a novel approach is developed to train the model that directly maximizes the mean average precision. The results show significant improvements are possible by modeling dependencies, especially on the larger web collections.

4.4 Conditional Random Fields

So far, we have described the Markov network representation as a joint distribution. In this subsection, we introduce one notable variant of an MRF, i.e., conditional random field (CRF) [38, 65], which is yet another popular model for sequence labeling and has been widely used in information extraction as described in Chapter 2.

4.4.1 Overview. A CRF is an undirected graph whose nodes can be divided into exactly two disjoint sets, the observed variables X and the output variables Y, which can be parameterized as a set of factors in the same way as an ordinary Markov network. The underlying idea is that of defining a conditional probability distribution $p(Y|X)$ over label sequences Y given a particular observation sequence X, rather than a joint distribution over both label and observation sequences $p(Y, X)$. The primary advantage of CRFs over HMMs is their conditional nature, resulting in the relaxation of the independence assumptions required by HMMs in order to ensure tractable inference.

Considering a linear-chain CRF with $Y = \{y_1, y_2, ..., y_n\}$ and $X = \{x_1, x_2, ..., x_n\}$ as shown in Figure 8.3, an input sequence of observed variable X represents a sequence of observations and Y represents a sequence of hidden state variables that needs to be inferred given the observations. The y_i's are structured to form a chain, with an edge between each y_i and y_{i+1}. The distribution represented by this network has the form:

$$p(y_1, y_2, ..., y_n|x_1, x_2, ..., x_n) = \frac{1}{Z(X)} \exp\left(\sum_{k=1}^{K} \lambda_k f_k(y_i, y_{i-1}, x_i)\right),$$

where $Z(X) = \sum_{y_i} \exp\left(\sum_{k=1}^{K} \lambda_k f_k(y_i, y_{i-1}, x_i)\right)$.

4.4.2 The Learning Algorithms. For general graphs, the problem of exact inference in CRFs is intractable. Basically, the inference problem for a CRF is the same as for an MRF. If the graph is a chain or a tree, as shown in Figure 8.3, message passing algorithms yield exact solutions, which are similar to the forward-backward [5, 55] and Viterbi algorithms [24] for the case of HMMs. If exact inference is not possible, generally the inference problem for a CRF can be derived using approximation techniques such as MCMC [48, 3], loopy *belief propagation* [46, 8], and so on. Similar to HMMs, the parameters are typically learned by maximizing the likelihood of training data. It can be solved using an iterative technique such as iterative scaling [38] and gradient-descent methods [63].

4.4.3 Applications in Text Mining. CRF has been applied to a wide variety of problems in natural language processing, including POS tagging [38], shallow parsing [63], and named entity recognition [40], being an alternative to the related HMMs. Based on HMM models, we can determine the sequence of labels by maximizing a joint probability distribution $p(\mathcal{X}, \mathcal{Y})$. In contrast, CRMs define a single log-linear distribution, i.e., $p(\mathcal{Y}|\mathcal{X})$, over label sequences given a particular observation sequence. The primary advantage of CRFs over HMMs is their conditional nature, resulting in the relaxation of the independence assumptions required by HMMs in order to ensure tractable inference. As expected, CRFs outperform HMMs on POS tagging and a number of real-word sequence labeling tasks [38, 40].

4.5 Other Models

Recently, there are many extensions of basic graphical models as mentioned above. Here we just briefly introduce the following two models, probabilistic relational model (PRM) [25] and Markov logic network (MLN) [59]. A Probabilistic relational model is the counterpart of a Bayesian network in statistical relational learning, which consists of relational schema, dependency structure, and local probability models. Compared with BN, PRM has some advantages and disadvantages. PRMs allow the properties of an object to depend probabilistically both on other properties of that object and on properties of related objects, while BN can only model relationships between at most one class of instances at a time. In PRM, all instances of the same class must use the same dependency mode, and it cannot distinguish two instances of the same class. In contrast, each instance in BN has its own dependency model, but cannot generalize over instances. Generally, PRMs are significantly more expressive than standard models, such as BNs. The well-known methods for learning BNs can be easily extended to learn these models.

A Markov logic network [59] is a probabilistic logic which combines first-order logic and probabilistic graphical models in a single representation. It is a first-order knowledge base with a weight attached to each formula, and can be viewed as a template for constructing Markov networks. Basically, probabilistic graphical models enable us to efficiently handle uncertainty. First-order logic enables us to compactly represent a wide variety of knowledge. From the point of view of probability, MLNs provide a compact language to specify very large Markov networks, and the ability to flexibly and modularly incorporate a wide range of domain knowledge into them. From the point of view of first-order logic, MLNs

add the ability to handle uncertainty, tolerate imperfect and contradictory knowledge, and reduce brittleness. The inference in MLNs can be performed using standard Markov network inference techniques over the minimal subset of the relevant Markov network required for answering the query. These techniques include belief propagation [46] and Gibbs sampling [23, 3].

5. Probabilistic Models with Constraints

In probabilistic models, domain knowledge is encoded into the model implicitly for most of the time. In this section, we introduce several situations that domain knowledge can be modeled as explicit constraints to the original probabilistic models.

By merely using PLSA or LDA, we may derive different topic models when the algorithms converge to different local maximums. It will be very useful if users can explicitly state which topic model they favor. A simple way to handle this issue is to list the terms that are desired by the users in each topic. For example, if "sport", "football" must be contained in Topic 1, users can indicate a related term distribution as prior distribution for Topic 1. This prior can be integrated into PLSA. Another sort of guidance is to specify which terms should have similar probabilities in one topic (must-link) and which terms should not have similar probabilities in any topic (cannot-link). This kind of prior can be modeled as Dirichlet forest prior, which is discussed in [4].

In traditional topic models, documents are considered indepedent with each other. However, in reality there could be correlations among documents. For example, linked webpages tends to be similar with each other, a paper cites another paper indicates the two papers are somehow similar. NetPLSA [41] and iTopicModel [64] are two algorithms that improve the original PLSA by consider the network constraints among the documents. NetPLSA takes the network constraints as an additional graph regularization term that forces two linked documents much similar, while iTopicModel models the network constraints using a Markov random field and also considers the direction of links in the network. These algorithms can still be solved by EM algorithm. By looking at the E-step, we can see that the constraints can be integrated into E-step, with a nice interpretation.

In [26], it proposes a framework of posterior regularization for probabilistic models. Different from traditional priors that are directly applied onto parameters, posterior regularization framework allows users to specify the constraints which are dependent on data space. For example, in an unsupervised part-of-speech tagging task, users may require