Unraveling temporal processes using probabilistic graphical models

de Paula Bueno, M.L.

2

PRELIMINARIES

In this chapter, we fix the notation used throughout this work and present definitions on probabilistic graphical models that are relevant for the following chapters. We start off by covering the basics of Bayesian networks, then move to dynamic Bayesian networks and hidden Markov models, which extend the framework of Bayesian networks for handling temporal problems. Learning models from data is also discussed.

2.1 NOTATION

We first introduce the notation and a few conventions used throughout this work. In probability theory, *random variables* are typically denoted by upper case letters, such as X, while the *domain* of a random variable X is represented by $\mathrm{dom}(X)$, which represents the set of values that X takes on [52]. A *discrete random variable* is a random variable which has a finite or countably infinite domain, while a *continuous random variable* has as domain a subset of the real numbers. A random variable is associated to a *probability distribution*, which assigns a probability value to each value of its domain (for discrete variables) or to real intervals of its domain (for continuous variables).

The probability distribution of a discrete random variable X will be denoted by $P(X)$, and the probability of a certain value $x \in \mathrm{dom}(X)$ will be denoted by $P(X = x)$ or simply $P(x)$ when no confusion can arise. The probability distribution of a continuous random variable Y with probability density function $f(Y)$ is denoted by $p(Y)$, and the probability that Y takes values on a real interval $[y_1, y_2]$ is indicated as $p(y_1 \leq Y \leq y_2)$. A set of random variables will be denoted by a bold face letter, e.g., $\mathbf{X} = \{X_1, \ldots, X_n\}$. A probability distribution assigned to a single random variable as in $P(X)$ is called a *univariate* distribution, while the joint distribution assigned to set of variables $\{X_1, \ldots, X_n\}$ is called a *multivariate* distribution and is denoted by $P(X_1, \ldots, X_n)$ or $P(\mathbf{X})$.

In *temporal modeling*, each variable is often measured repeatedly, such that a variable X at time t will be referred to as $X^{(t)}$. This means that the domains of $X^{(t)}$, for all $t \geq 0$, are the same. For the discrete time points $\{t_1, \ldots, t_2\}$, where $t_2 \geq t_1 \geq 0$, the notation $X^{(t_1:t_2)}$ will be used to refer to the set of variables $\{X^{(t_1)}, \ldots, X^{(t_2)}\}$.

2.2 BAYESIAN NETWORKS

2.2.1 *Origin*

A Bayesian network (BN, for short) is a graphical model of a multivariate probability distribution with independence constraints. Bayesian networks date back to the 1980s [97, 104, 136]; their goal was to overcome the limitations of rule-based expert systems from the previous decade that incorporated uncertainty in the form of numbers that had some resemblance to probabilities [115]. One important limitation of such AI systems was the need for representing an unrealistic number of probabilities to perform probabilistic inference, a problem which was dealt with by making many simplifying assumptions. While this led to a substantial reduction in the needed number of probabilistic parameters, it also gave rise to poor performance in solving real-world problems, for example in medical diagnosis [70]. By marrying probability theory with graph theory, Bayesian networks allowed to provide the right balance in the number of probabilistic parameters needed to realistically represent the problem domain at hand.

A Bayesian network is a two-fold representation, as it encodes both qualitative and quantitative information about probability distributions. The qualitative side of a Bayesian network is given by a graph, whose semantics is associated to statistical independence statements. The quantitative side regards numerical probabilities, which are specified following the structure of the graph. As a result, BNs provide a compact, yet expressive way of representing probability distributions.

2.2.2 *Representation*

To define Bayesian networks over a set of random variables of interest, a few definitions are introduced first. A *graph* $\mathcal{G}$ is a pair $\mathcal{G} = (\mathbf{V}, \mathbf{A})$, where $\mathbf{V}$ is a set of objects $i \in \{1, \dots, n\}$, $n = |\mathbf{V}|$, called *nodes*, and $\mathbf{A} \subseteq \mathbf{V} \times \mathbf{V}$ is a set of node pairs called *edges*. If $\mathcal{G}$ is a *directed graph*, then each edge of A is an ordered pair (i, j), also represented by $i \to j$, such that $(j, i) \notin \mathbf{A}$. The edges are then called *directed edges* or *arcs*. If $i \to j \in \mathbf{A}$ is an arc, then i is called the *parent* of node j, and j is called the *child* of node i. If there is a directed path from node i to node j, then i is called the *ancestor* of j, whereas j is called the *descendant* of i.

If $\mathcal{G}$ is an *undirected graph*, then its edges are unordered pairs, i.e., if $(i, j) \in \mathbf{A}$ then also $(j, i) \in \mathbf{A}$, simply represented as a set $\{i, j\}$. A *directed acyclic graph* (DAG, for short) is a directed graph with no cycles, i.e., there is no sequence of arcs of the form $i \to j \to \cdots \to i$ (first and last node in the sequence are the same). As usual, each node i in the DAG with $V = \{1, \dots, n\}$ will be associated in a one-to-one way to a random variable X_i from the set of variables $X_1, \dots, X_n$ for the convenience of defining a Bayesian network. In the following, we shall refer to nodes and variables interchangeably and use X_i to refer to both the node and the variable.

One way to define Bayesian networks is from the notion of factorizing a joint probability distribution according to the structure of a graph as follows.

Definition 2.1 (Factorization). *Let $\mathcal{G} = (\mathbf{V}, \mathbf{A})$ be a directed acyclic graph with nodes $\mathbf{V} = \{X_1, \ldots, X_n\}$. A joint probability distribution P over the same variables factorizes according to $\mathcal{G}$ if P can be written as:*

$$P(\mathbf{X}) = P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \pi(X_i)) \tag{2.1}$$

where $\pi(X_i)$ refers to the parents of the node X_i in $\mathcal{G}$, and each factor $P(X_i \mid \pi(X_i))$ is called a conditional probability table (CPT, for short).

Definition 2.2 (Bayesian network). *A Bayesian network is a pair $\mathcal{B} = (\mathcal{G}, P)$, where P is a joint probability distribution that factorizes according to a directed acyclic graph $\mathcal{G}$.*

The joint probability distribution P associated with a Bayesian network $\mathcal{G}$ encodes conditional independences, if it holds for three mutually disjoint sets of variables $\mathbf{U}, \mathbf{W}, \mathbf{Z} \subseteq \mathbf{X}$ that if $P(\mathbf{U} \mid \mathbf{W}, \mathbf{Z}) = P(\mathbf{U} \mid \mathbf{Z})$ for any set of values of the variables in $\mathbf{U}, \mathbf{W}, \mathbf{Z}$. It is said that the variables $\mathbf{U}$ and $\mathbf{W}$ are *conditionally independent* (under P) given $\mathbf{Z}$, written as $\mathbf{U} \perp\!\!\!\perp_P \mathbf{W} \mid \mathbf{Z}$. The set of all conditional independent triplets associated to a joint probability distribution P is sometimes defined as $I(P) = \{(\mathbf{U}, \mathbf{W}, \mathbf{Z}) \mid \mathbf{U} \perp\!\!\!\perp_P \mathbf{W} \mid \mathbf{Z}\}$.

The Bayesian network graph encodes independence relationships, which can be read off by means of a graphical property called *d-separation* (directed separation) [136]. The notion of d-separation defines potential probabilistic influence between variables based on the structure of the BN graph. This can be described by means of the notion of *active trail* [104]. A sequence of nodes $\sigma = X_1, \ldots, X_m$ in the graph $\mathcal{G}$ is a *trail* if either $X_i \to X_{i+1}$ or $X_i \leftarrow X_{i+1}$ is an arc in $\mathcal{G}$ on the trail σ, $i = 1, \ldots, m-1$, i.e., the direction of the arcs is ignored and only the fact that X_1 is connected by the trail to X_m is taken into account. Now the trail between X_1 and X_m is called *active* if for any Y on the trail σ, the connections of the neighboring nodes U and W have the following directions:

- $U \leftarrow Y \to W$ (divergent connection), $U \to Y \to W$ (serial connection), or $U \leftarrow Y \leftarrow W$ (serial connection), and Y has *not* been observed, or
- $U \to Y \leftarrow W$ (convergent connection or v-structure), whereas Y or any of its descendants have been observed.

If a trail is not active, it is called *inactive*.

Now consider the following three mutually disjoint sets of nodes $\mathbf{U}, \mathbf{W}, \mathbf{Z} \subseteq \mathbf{V}$. If all trails between any node in $\mathbf{U}$ and any node in $\mathbf{W}$ are inactive given (possibly empty) observations in $\mathbf{Z}$, it is said that the set of nodes $\mathbf{Z}$ *d-separates* the set of nodes $\mathbf{U}$ and $\mathbf{W}$, written as

$$\mathbf{U} \perp\!\!\!\perp^d_{\mathcal{G}} \mathbf{W} \mid \mathbf{Z}$$

For the graph $\mathcal{G}$ we can now collect all d-separation triplets:

$$I(\mathcal{G}) = \{(\mathbf{U}, \mathbf{W}, \mathbf{Z}) \mid \mathbf{U} \perp\!\!\!\perp^d_{\mathcal{G}} \mathbf{W} \mid \mathbf{Z}\}$$

The above definition can be used to provide a semantics of the BN graph in terms of independence statements that are entailed by the graph. For Bayesian networks $\mathcal{B} = (\mathcal{G}, P)$ where the distribution P factorizes according to the DAG $\mathcal{G}$, it holds that $I(\mathcal{G}) \subseteq I(P)$ [136]. This means that every independence that holds in the BN graph must hold in the distribution. This explains why the interpretation of d-separation as conditional independence is meaningful. However, the semantics makes also clear that the two independence relations $I(P)$ and $I(\mathcal{G})$ may not coincide.

Because of d-separation, the network structure of a Bayesian network can be seen as its *qualitative* part, while the *quantitative* part corresponds to the probabilities encoded in the CPTs. A Bayesian network example is provided in Example 2.1.

Example 2.1. *Assume we are interested in diagnosing lung cancer, as represented by the variable C with dom(C) = {no, yes}. Other variables of interest are smoking (S), gender (G), and age (A), where dom(S) = {no, yes}, dom(G) = {female, male} and dom(A) = {adult, elderly}. Figure 2.1 shows the graphical structure and the CPTs associated to this Bayesian network. Independence relationships can be deduced from the graphical structure, e.g., having knowledge about smoking will make age and gender irrelevant for the prediction of lung cancer.*

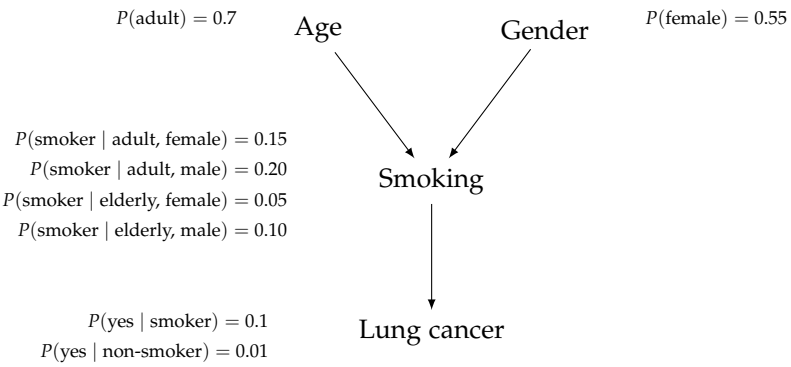


Figure 2.1: Bayesian network of the lung cancer example. Note that the structure and the CPTs in this example are fictional.

One advantage of the BN framework is that in the network structure one typically captures only the essential variable interactions, which often results in a substantial reductions in the needed number of probabilities to be specified. The BN of Example 2.1 requires $1 + 1 + 4 + 2 = 8$ independent parameters, while the explicit specification of the joint distribution of $P(A, G, S, C)$ would require $2^{(4)} - 1 = 15$ independent parameters. This advantage tends to be more significant if one deals with BNs with more variables, or variables with larger domains, for example.

The first step to using the BN representation is usually finding a suitable network structure for the problem at hand. One way to construct such a network structure is by manually defining the interactions among the involved variables that are supposed to hold in the domain, based on prior background knowledge and supported by domain experts. In that case, it is usually easier to think of interactions in terms of cause-effect relationships [47], although the semantics of a Bayesian network per se does not embody a causality notion. The network structure can also be obtained from data. In any case, once the network structure is obtained, the parameters of the network nodes need to be estimated, which can also be done manually [71] or algorithmically [47].

2.3 LEARNING BAYESIAN NETWORKS

In practical situations, prior knowledge about the problem at hand might not be available for handcrafting a Bayesian network for the domain, for example, if it is too expensive to be obtained, nonexistent, or is prone to be incorrect. This motivates the need for Bayesian network learning algorithms [44, 85, 169], whose goal is to automatically find a Bayesian network that suitably represents the distribution associated to the data. Bayesian network learning involves two steps: learning a network structure and learning numerical parameters. Handling each task also depends whether the data is complete or incomplete. In this section, we consider the case of complete data.

2.3.1 *Parameter learning*

In the parameter learning task, we assume the network structure is known. The goal is to estimate the CPTs of all the variables, i.e. the distributions $P(x_i \mid \pi(x_i))$ for every x_i. Let us denote by θ the set of parameters associated to the Bayesian network which are to be estimated, and let D be a set of data points $\mathbf{x}[1], \ldots, \mathbf{x}[m]$ of the form $\mathbf{x}[j] = \{x_1[j], \ldots, x_n[j]\}$, where $x_i[j]$ refers to the value of the variable X_i taken in the jth data point. Each X_i is assumed to follow a categorical distribution taking values on $\mathrm{dom}(X_i)$. We further assume that all data points of D are independent and identically distributed (i.e., i.i.d. samples). The likelihood function of the Bayesian network with structure $\mathcal{G}$ parameterized by θ given the data D corresponds to the probability of D under such model and is given by:

$$\mathcal{L}(\theta : D) = P(\mathbf{x}[1], \ldots, \mathbf{x}[m] : \theta) \tag{2.2}$$

$$= \prod_{j=1}^{m} P(\mathbf{x}[j] : \theta) \tag{2.3}$$

From the factorization of Bayesian networks we obtain:

$$\mathcal{L}(\theta\colon D) = \prod_{j=1}^{m}\prod_{i=1}^{n} P(x_i[j] \mid \pi(x_i)[j]\colon \theta) \tag{2.4}$$

$$= \prod_{i=1}^{n} P(x_i[1] \mid \pi(x_i)[1]\colon \theta) \ldots P(x_i[j] \mid \pi(x_i)[j]\colon \theta) \tag{2.5}$$

Equation 2.5 shows that the likelihood function can be decomposed into a product of independent terms, one for each node of the network structure. If we denote by θ_{irk} the parameter $P(X_i = x_k \mid \pi(X_i) = \mathbf{x}_r)$, then the likelihood function can be further expanded as follows:

$$\mathcal{L}(\theta\colon D) = \prod_{i=1}^{n}\prod_{x_k,\mathbf{x}_r} \theta_{irk}^{N_{irk}} \tag{2.6}$$

where N_{irk} is the number of times the configuration $(X_i = x_k, \pi(X_i) = \mathbf{x}_r)$ is seen in D. The goal now is to find the set of parameters θ that maximize this function, an approach known as *maximum likelihood estimation* (MLE, for short). The parameters that maximize the likelihood function are denoted by $\hat{\theta}$. It is usually easier to work with the logarithm of Equation 2.6, which is referred to as the log-likelihood of the data. It is possible to show [104] that the maximization of the log-likelihood leads to closed-form formulas for parameter learning as follows:

$$\hat{\theta}_{irk} = \frac{N_{irk}}{N_{ir}} \tag{2.7}$$

where N_{ir} is the number of times the configuration $(\pi(X_i) = \mathbf{x}_r)$ occurs in D. These quantities are known as *sufficient statistics*, and convey the idea that each parameter corresponds to the node's proportional counts with respect to its parents. Once the optimal parameters $\hat{\theta}$ are computed, the likelihood computed based on $\hat{\theta}$ is denoted by $\hat{\mathcal{L}}$.

2.3.2 *Structure learning*

When the network structure is unknown, one resorts to learning the network structure from data. The goal of structure learning is to recover the structure of the hypothetical joint probability distribution underlying the data [50]. With structure learning, one is able to discover the dependence structure of the domain, which can yield insight about qualitative influences that hold in the domain, both direct and indirect. Network structure learning is also important to make the parameter estimation in Bayesian networks feasible, although the structure should not be overly simplistic, otherwise relevant correlations might be missed.

The problem of structure learning can be formulated as an optimization problem [7, 42], also known as *score-based* approach, whose goal is to find the network structure $\hat{\mathcal{G}}$ that optimizes a scoring function:

$$\hat{\mathcal{G}} = \arg\max_{\mathcal{G}\in\boldsymbol{\mathcal{G}}} \text{Score}(\mathcal{G}, D) \tag{2.8}$$

where $\mathcal{G}$ is the space of network structures, i.e. the set of directed acyclic graphs with nodes $\{X_1, \ldots, X_n\}$. In general, finding optimal Bayesian networks has been shown intractable [42, 43]. Network structures can also be learned by means of the *constraint-based* approach [118, 136], which determines a network structure that is consistent with the independence relationships that hold on the data. In the case of constraint-based learning, the worst case requires an exponential number of tests [118]. In the remainder of this chapter, we consider the score-based approach and further elaborate on it.

Scoring functions for structure learning play a central role in this task and the literature offers a variety of them. One of the simplest score functions is the likelihood score [104], which indicates the probability of the data given the model and was defined in Equation 2.2. In the situation of unknown structure, the likelihood score seeks the model (i.e. graph and parameters) that maximizes the likelihood:

$$\max \mathcal{L}(\mathcal{G}, \theta_{\mathcal{G}} : D) = \max_{\mathcal{G}} \left[\max_{\theta_{\mathcal{G}} \in \Theta} \mathcal{L}(\mathcal{G}, \theta_{\mathcal{G}} : D) \right] \tag{2.9}$$

$$= \max_{\mathcal{G}} \left[\mathcal{L}(\mathcal{G}, \hat{\theta}_{\mathcal{G}} : D) \right] \tag{2.10}$$

where Θ is the space of CPTs with regard to the graph $\mathcal{G}$. Hence, in order to maximize the likelihood, one needs to find the structure $\hat{\mathcal{G}}$ that maximizes Equation 2.10, where each candidate structure has parameters fitted via MLE. However, as the goal with model learning is to capture the true distribution of the data, using the likelihood score typically has severe limitations as follows. By adding more arcs to the network, the likelihood score never decreases and instead tends to increase [104]. Hence, by completely fitting to the data, one is also fitting to the noise on the data, and the resulting network tends to be a fully connected graph. This usually leads to the problem of model overfitting, which means that the model does not generalize well (i.e. it performs poorly on new data).

One alternative score function is the Bayesian score, which adopts a Bayesian approach to modeling the structure and parameters that are to be estimated. In the Bayesian approach, one defines a structure prior $P(\mathcal{G})$ and a parameter prior $P(\theta_{\mathcal{G}} \mid \mathcal{G})$ for the possible ways a given structure can be parameterized. For a candidate graph $\mathcal{G}$, we can apply Bayes' rule to obtain:

$$P(\mathcal{G} \mid D) \propto P(D \mid \mathcal{G}) P(\mathcal{G}) \tag{2.11}$$

where the denominator $P(D)$ can be dropped because it is the same for all the graphs. The Bayesian score is then defined by taking the logarithm of the right-hand side of Equation 2.11:

$$\text{Score}_B(\mathcal{G} \mid D) = \log P(D \mid \mathcal{G}) + \log P(\mathcal{G}) \tag{2.12}$$

In the prior $P(\mathcal{G})$ one can model a prior distribution that might favor, e.g., sparser graphs. The term $P(D \mid \mathcal{G})$ is known as marginal likelihood as it can be written as:

$$P(D \mid \mathcal{G}) = \int_{\theta_{\mathcal{G}} \in \Theta} P(D \mid \theta_{\mathcal{G}}, \mathcal{G}) P(\theta_{\mathcal{G}} \mid \mathcal{G}) d\theta_{\mathcal{G}} \tag{2.13}$$

Intuitively, the marginal likelihood weights the likelihood of the data $P(D \mid \theta_{\mathcal{G}}, \mathcal{G})$ by different ways of selecting the parameters given the network $\mathcal{G}$. Hence, the marginal likelihood can be seen as an average of the likelihoods for the structure $\mathcal{G}$, as opposed to the maximum likelihood score, which looks only at the score that maximizes the term $P(D \mid \theta_{\mathcal{G}}, \mathcal{G})$. The Bayesian score tends to favor simpler structures if little data is available for learning [104], which provides a mechanism to combat overfitting. By using Dirichlet priors on all parameters of the network, it is possible to show [104] that an approximation of the Bayesian score results in the so-called Bayesian information criterion (BIC, for short), which is given as follows:

$$\text{BIC}(\mathcal{G} \mid D) = -2 \cdot \log \mathcal{L}(\hat{\theta}_{\mathcal{G}} : D) + K \cdot \log m \tag{2.14}$$

where K is the number of parameters of the network structure $\mathcal{G}$, and m is the size of the dataset.

The goal now is to find the structure $\mathcal{G}$ that minimizes the BIC score, where the term $K \cdot \log m$ in Equation 2.14 acts as a penalty term. Equation 2.14 suggests that the problem of structure learning can be seen as a model selection problem [182], where one wishes to find the network structure that balances goodness-of-fit (the likelihood term) and model size. The scoring function is then coupled to a search procedure, which is often a heuristic procedure such as tabu search [77], hill climbing, simulated annealing, among others [152], resulting in sub-optimal network structures obtained in feasible running time.

Although heuristic procedures are often used in BN structure learning, research has shown that optimal structure learning can be done efficiently in some situations [31, 158]. Some techniques are able to scale to problems with hundreds variables [50]. Research has also shown that it is possible to predict which algorithms would be more suitable for optimal learning of a given instance [116].

2.3.3 *Decomposable scores*

In structure learning, a key computational property is that of decomposability. A score is decomposable if it is defined locally per node [50]. This allows for the score of a candidate Bayesian network $\mathcal{B}$, also referred to as its *global score*, to be given as a sum of *local scores*, one for each variable:

$$\text{Score}(\mathcal{B}) = \sum_{X_i \in \mathbf{X}} \text{Score}(X_i, \pi_{\mathcal{B}}(X_i)) \tag{2.15}$$

where $\pi_{\mathcal{B}}(X_i)$ refers to the parents of X_i in $\mathcal{B}$. Decomposable scores allow for the efficient evaluation of small changes to the structure, such as arc removal and arc

addition, as such operations affect only the associated local scores. As a result, by exploiting this property, structure learning algorithms can scale reasonably well. Many scores commonly used in structure learning are decomposable, where the BIC is one such score [50].

2.4 DYNAMIC BAYESIAN NETWORKS

In this section we discuss extensions of Bayesian networks for modeling temporal processes by means of dynamic Bayesian networks (DBNs, for short). Learning DBNs from data is also discussed.

2.4.1 *Representation*

Dynamic Bayesian networks [68, 124] extend Bayesian networks for modeling temporal processes where uncertainty plays an important role. We restrain ourselves to dynamic systems in which all the variables of a set $\mathbf{X} = \{X_1, \ldots, X_n\}$ are measured together and repeatedly over time, which is represented by $\mathbf{X}^{(0)}, \mathbf{X}^{(1)}, \ldots$. Further, the time interval between two measurements $\mathbf{X}^{(t)}$ and $\mathbf{X}^{(t+1)}$, for any $t \geq 0$, is assumed fixed. This means that in such dynamic systems the sequential behavior of the involved variables is abstracted from the absolute time of their measurement.

In order to keep the model compact, a few additional assumptions about the process involved in the generation of $\mathbf{X}$ are often considered [104], which we describe as follows.

Definition 2.3 (Markovian dynamic system)**.** *A dynamic system over the variables* $\mathbf{X}$ *is first-order Markovian (or simply Markovian) if, for all* $t \geq 0$*,*

$$P(\mathbf{X}^{(t+1)} \mid \mathbf{X}^{(0:t)}) = P(\mathbf{X}^{(t+1)} \mid \mathbf{X}^{(t)}) \tag{2.16}$$

The Markovian assumption means that predicting the future state of the process depends only on its current state and not on previous states it has assumed. In this case, the process is also said to be *memoryless*. Another useful property is given as follows.

Definition 2.4 (Time-homogeneous dynamic system)**.** *A dynamic system over the variables* $\mathbf{X}$ *is time homogeneous (or time invariant) if* $P(\mathbf{X}^{(t+1)} \mid \mathbf{X}^{(t)})$ *is the same for every* $t \geq 0$*.*

Dynamic Bayesian networks provide a representation for Markovian time-homogeneous dynamic systems grounded on graphical models as defined next.

Definition 2.5 (Dynamic Bayesian network)**.** *A dynamic Bayesian network is a Markovian time-homogeneous system* $(\mathcal{B}_0, \mathcal{B}_{\rightarrow})$ *over* $\mathbf{X}$*, where:*

- $\mathcal{B}_0 = (\mathcal{G}_0, P_0)$ *is a Bayesian network over the variables* $\mathbf{X}^{(0)}$ *called initial network.*

- $\mathcal{B}_{\rightarrow} = (\mathcal{G}_{\rightarrow}, P_{\rightarrow})$ *is a Bayesian network over the variables* $\{\mathbf{X}^{(t)}, \mathbf{X}^{(t+1)}\}$ *called transition network. The variables of* $\mathbf{X}^{(t)}$ *have no parents in the transition network.*

The transition network can also be seen as a *conditional Bayesian network* [104], because it suffices to define the distribution $P(\mathbf{X}^{(t+1)} \mid \mathbf{X}^{(t)})$ for defining this network. Although DBNs can be defined as semi-infinite systems [68], in practice one reasons with a finite horizon $\{0, \ldots, T\}$. In this case, the DBN is unrolled so that a joint distribution over the process duration is specified as follows: the structure and parameters of all the nodes at time $t = 0$ come from the initial model, while the structure and parameters for any node $X_i^{(t)}$, where $t > 0$, come from the transition model.

From the previous definitions and assumptions, the joint distribution of a DBN over a time horizon $\{0, \ldots, T\}$ is as follows:

$$P(\mathbf{X}^{(0:T)}) = P(\mathbf{X}^{(0)}) \prod_{t=0}^{T-1} P(\mathbf{X}^{(t+1)} \mid \mathbf{X}^{(t)}) \tag{2.17}$$

$$= \prod_{i=1}^{n} P_0(X_i^{(0)} \mid \pi(X_i^{(0)})) \prod_{t=0}^{T-1} \prod_{i=1}^{n} P_{\rightarrow}(X_i^{(t+1)} \mid \pi(X_i^{(t+1)})) \tag{2.18}$$

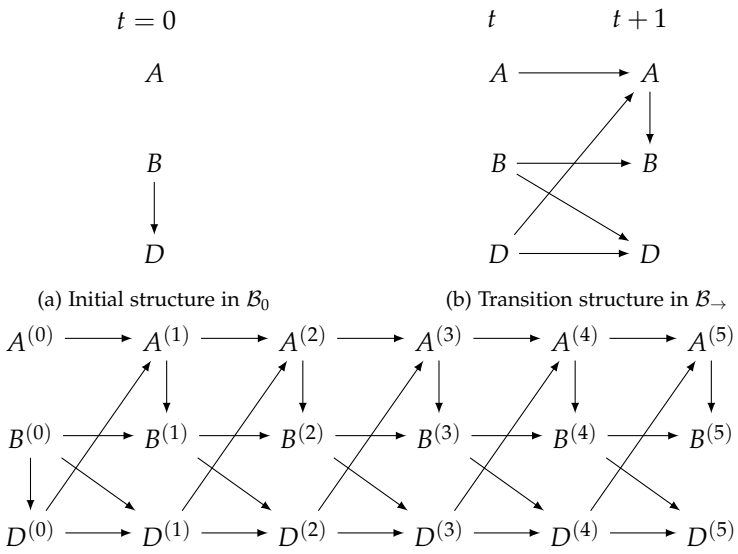
where in Equation 2.18 it is shown that the joint can be written in a modular way based on the factorization provided by the distributions P_0 and $P_{\rightarrow}$. An example of DBN for a medical problem is described in Example 2.2.

Example 2.2. *In a disease process, two symptoms (denoted by A and B) and the administered drug quantity (denoted by D) are observed at regular time intervals for each patient. A DBN is used to model patient evolution, where the structure of the initial model* $\mathcal{B}_0$ *and the transition model* $\mathcal{B}_{\rightarrow}$ *are shown at the top of Figure 2.2. From* $\mathcal{B}_0$ *and* $\mathcal{B}_{\rightarrow}$*, an unrolled DBN over six time points can be obtained, as shown at the bottom of Figure 2.2.*

In the transition model of a DBN the set of arcs from a variable at time t to a variable at time $t + 1$ are often called *intra-temporal* arcs, e.g., the arcs $B^{(0)} \rightarrow D^{(0)}$ and $A^{(1)} \rightarrow B^{(1)}$ in Example 2.2. On the other hand, arcs between variables from the same time point are called *inter-temporal* arcs, such as $A^{(0)} \rightarrow A^{(1)}$ in this example.

2.4.2 *Learning*

Learning DBNs is to a considerable extent similar to learning (static) Bayesian networks. Let us consider a training set D of m i.i.d. sequences, where the jth sequence has observations of the form $\mathbf{x}[j]^{(0)}, \ldots, \mathbf{x}[j]^{(m_j)}$. For convenience, we denote by D_0 the initial slices of D, which amount to m observations, whereas we denote by $D_{\rightarrow}$ the transition instances of D, which amount to m' observations, where $m' = \sum_{i=1}^{m} m_i$. The initial model $\mathcal{B}_0$ and the transition model $\mathcal{B}_{\rightarrow}$ are learned from D_0 and $D_{\rightarrow}$ respectively.



(a) Initial structure in $\mathcal{B}_0$

(b) Transition structure in $\mathcal{B}_{\rightarrow}$

(c) Unrolled DBN for six time points.

Figure 2.2: An example of DBN for a disease process. The CPTs of $\mathcal{B}_0$ and $\mathcal{B}_{\rightarrow}$ are not shown.

In an MLE approach, by a similar reasoning as done with static Bayesian networks (Section 2.3.1) it can be shown [68] that the BIC of a DBN $(\mathcal{B}_0, \mathcal{B}_{\rightarrow})$ with structure $\mathcal{G} = (\mathcal{G}_0, \mathcal{G}_{\rightarrow})$ is given by:

$$\text{BIC}(\mathcal{G} : D) = \text{BIC}_0 + \text{BIC}_{\rightarrow} \tag{2.19}$$

where

$$\text{BIC}_0 = -2 \cdot \log \mathcal{L}(\hat{\theta}_{\mathcal{G}_0} : D_0) + K_0 \cdot \log m \tag{2.20}$$

and

$$\text{BIC}_{\rightarrow} = -2 \cdot \log \mathcal{L}(\hat{\theta}_{\mathcal{G}_{\rightarrow}} : D_{\rightarrow}) + K_{\rightarrow} \cdot \log m' \tag{2.21}$$

such that K_0 and $K_{\rightarrow}$ denote the number of parameters of the initial and transition models respectively.

Equation 2.21 in fact uses the conditional log-likelihood of the transition instances, which is given by $\log \mathcal{L}(\hat{\theta}_{\mathcal{G}_{\rightarrow}} : D_{\rightarrow}) = \sum_{j=1}^{m} \sum_t \log P(\mathbf{x}[j]^{(t+1)} \mid \mathbf{x}[j]^{(t)})$. By maximizing the BIC of $\mathcal{B}_0$ and the BIC of $\mathcal{B}_{\rightarrow}$ independently, the BIC of the complete DBN is maximized as well.

2.5 HIDDEN MARKOV MODELS

In several situations, we might be interested in modeling latent (or hidden) variables, which allow for capturing unmeasured quantities related to the observed

quantities [178]. This might provide improved understanding of the problem at hand, along with other potential advantages such as simplified model structure [67] and better model fit [179].

In this section, we discuss hidden Markov models (HMMs, for short), which can be seen as instances of DBNs from a representation perspective. We focus on several representation aspects of HMMs, while learning is covered in the next section.

2.5.1 *Model architectures*

In a general problem setting, we denote by $\mathbf{X} = \{X_1, \ldots, X_n\}$ the set of observable features, and we assume that there is a set of state variables $\mathbf{S} = \{S_1, \ldots, S_\ell\}$ that we do not observe and are involved in the generation of $\mathbf{X}$ over time. In such problem, we are interested in a temporal model that can be constructed and used feasibly, yet is realistic and insightful. To this end, different sets of assumptions are very often used, taking also into account domain characteristics. As a consequence, the variety of existing HMMs renders different probabilistic interactions between $\mathbf{X}$ and $\mathbf{S}$ (by interaction we refer to unconditional probabilistic dependence).

A general HMM framework is illustrated in Figure 2.3 [25], where the exact form of interactions within states and within observables is abstracted. We start by defining the HMM which captures the interactions denoted by solid lines in Figure 2.3 and can be seen as a basis for several other HMMs.

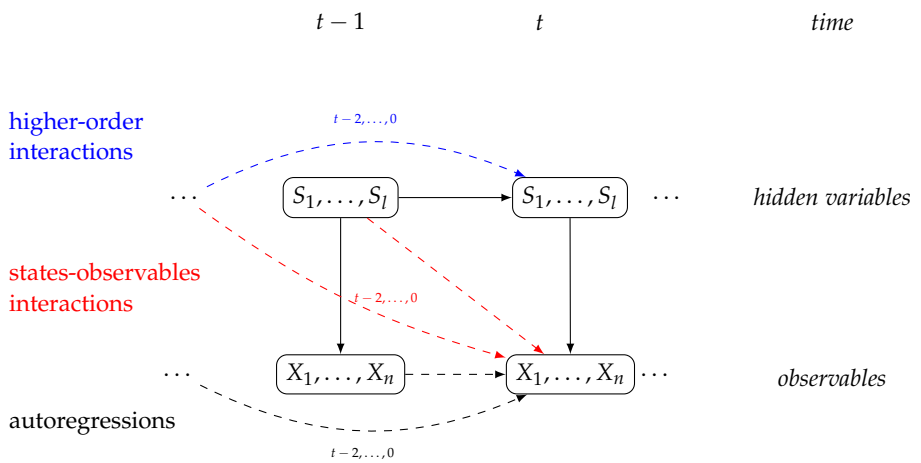


Figure 2.3: An abstracted general HMM with hidden variables $\{S_1, \ldots, S_\ell\}$ and observables $\{X_1, \ldots, X_n\}$. Solid arcs indicate interactions present in the independent HMM and related models.

Definition 2.6 (Hidden Markov model). *A hidden Markov model is a Markovian time-homogeneous system $\lambda = (A, B, v)$ over $\{S, \mathbf{X}\}$, where:*

- $A = P(S^{(t+1)} \mid S^{(t)})$ *is the transition distribution*
- $B = P(\mathbf{X}^{(t)} \mid S^{(t)})$ *is the emission distribution*
- $v = P(S^{(0)})$ *is the initial state distribution*

and $dom(S) = \{s_1, \ldots, s_k\}$ *is called the state space of the model.*

The above definition is based on those of dynamic systems given in Section 2.4, except that in an HMM we repeatedly measure not only observables $\mathbf{X}$, but also a latent variable S. In this case, there is a single latent variable per time point, hence $\mathbf{S} = \{S\}$. It is customary to view A as a matrix $[a_{ij}]$, B as a set $\{b_j(\mathbf{k})\}_{s_j \in \mathrm{dom}(S)}$, and v as a vector $[v(s_i)]$, where:

$$a_{ij} = P(S^{(t+1)} = s_j \mid S^{(t)} = s_i) \tag{2.22}$$

$$b_j(\mathbf{k}) = P(\mathbf{X}^{(t)} = \mathbf{x_k} \mid S^{(t)} = s_j) \tag{2.23}$$

$$v(i) = P(S^{(0)} = s_i) \tag{2.24}$$

The above notation will be useful when describing HMM learning (see Section 2.6.2). By unrolling an HMM over a finite time horizon $\{0, \ldots, T\}$, and from the given assumptions and definitions, the joint distribution of an HMM is:

$$P(\mathbf{X}^{(0:T)}, S^{(0:T)}) = P(S^{(0)}) \prod_{t=0}^{T} P(\mathbf{X}^{(t)} \mid S^{(t)}) \prod_{t=0}^{T-1} P(S^{(t+1)} \mid S^{(t)}) \tag{2.25}$$

A well-known class of HMMs is the *independent* HMM (HMM-I, for short) [102, 138, 141], in which the observables at a given time point are assumed conditionally independent given the state. This additional assumption means that $P(X_i^{(t)} \mid X_j^{(t)}, S^{(t)}) = P(X_i^{(t)} \mid S^{(t)})$ whenever $P(X_j^{(t)}, S^{(t)}) > 0$, for all $t \geq 0$ and $i \neq j$.

Based on the previous assumptions, the joint distribution of an HMM-I is as follows:

$$P(\mathbf{X}^{(0:T)}, S^{(0:T)}) = P(S^{(0)}) \prod_{t=0}^{T} \prod_{i=1}^{n} P(X_i^{(t)} \mid S^{(t)}) \prod_{t=0}^{T-1} P(S^{(t+1)} \mid S^{(t)}) \tag{2.26}$$

2.5.2 *Families of HMMs*

By relaxing the assumptions of the independent HMM based on the general architecture shown in Figure 2.3, different families of HMMs can be derived, as summarized in Figure 2.4. The emissions of a *standard* HMM can be defined as:

$$P(\mathbf{X} \mid S) = \prod_{i=1}^{n} P(X_i \mid S, \pi^{-}(X_i)) \tag{2.27}$$

where $\pi^{-}(X_i)$ denotes the set of parents of X_i excluding the state, and it depends on the Bayesian network associated to the feature space. In the literature, the standard HMM is also known as HMM/BN [119].

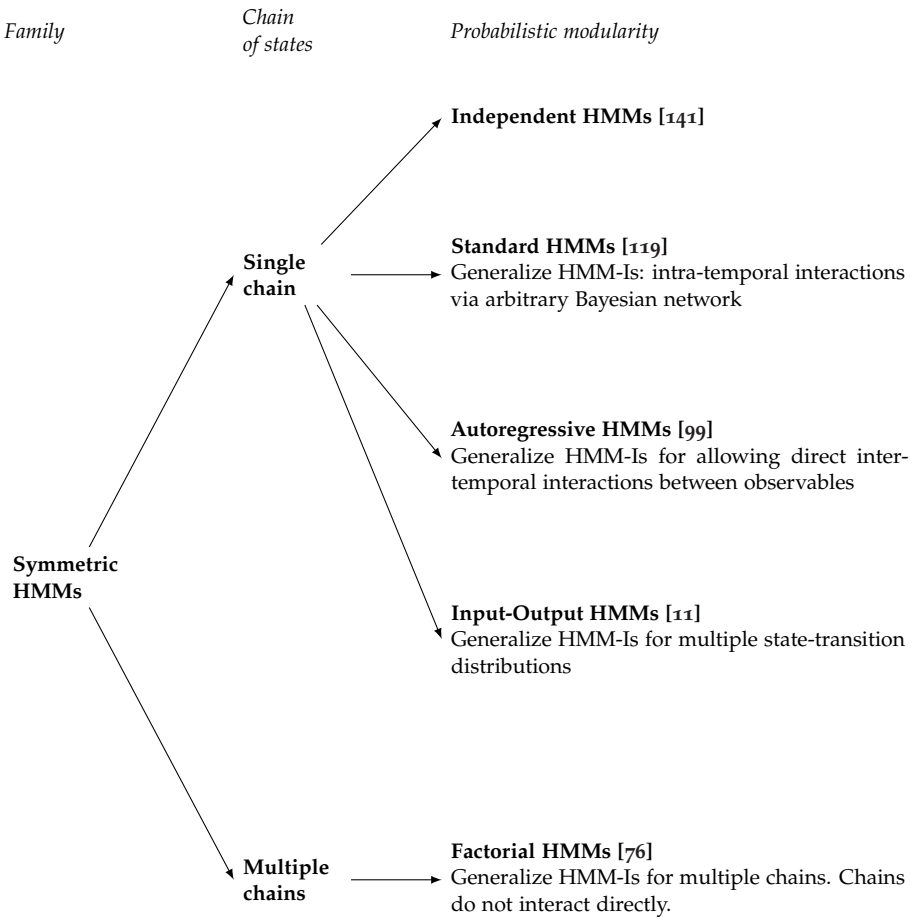


Figure 2.4: HMM families, where *intra-temporal* interactions refer to probabilistic interaction among observables from the same time point, while *inter-temporal* interactions refer to interactions between distinct time points.

The models from Figure 2.4 can be distinguished based on a number of factors, where probabilistic modularity, i.e. the sets of variables that are directly connected in the model, has a major importance. For example, the very high modularity of HMM-Is requires that the state variable summarize more history information, which can imply larger state spaces [76], as opposed to autoregressive HMMs, where the direct interaction between observables is prone to reduce the burden over the hidden states. On the other hand, as autoregressive HMMs might require more parameters, they are prone to be less stable. This illustrates that there can be several trade-offs when a decision must be made about the design of an HMM.

2.5.3 *Learning*

Learning HMMs often reduces to parameter learning, because the structure of models such as the independent HMM and input-output HMM is defined beforehand. However, structure learning might be needed, e.g., for learning standard HMMs and autoregressive HMMs if one wants to decide which autoregressions should be present based on the data. Due to the presence of latent variables, learning HMMs is addressed differently than discussed so far and will be covered in more detail in Section 2.6.

2.6 LEARNING WITH LATENT VARIABLES

In many real-life situations the data might have variables with missing values due to several reasons, e.g., when patients drop out of treatment, do not carry out a measurement or forget to register the result of a measurement, or when a sensor breaks down. At other times, we might be interested in modeling latent (or hidden) variables, i.e., the situation when no values have been observed for a variable, which is the case of probabilistic models such as HMMs. Dealing with missing values or with situations when all values of a variable are missing is done by similar algorithms. In this section we discuss learning of models with latent variables.

Learning Bayesian networks with latent variables is more difficult than the case of complete data, as closed-form solutions are in general not available [15, 21]. Iterative methods are often used, including gradient-based methods [104] and the expectation maximization algorithm (EM, for short) [53, 65, 67]. More recently, research has shown that in some situations parameter learning of Bayesian networks with latent variables can be done in closed form [21], while structure learning has been done by approximating predictive distributions, also in an iteration-free approach [143].

2.6.1 *The expectation-maximization algorithm*

In the situation of latent variables, the space of variables is given by a set of observables $\mathbf{X}$ together with a set of latent variables $\mathbf{S}$. Then, the log-likelihood function of model parameters θ given a probabilistic structure $P(\mathbf{X}\colon \theta)$ and data on $\mathbf{X}$ can be written as:

$$\log \mathcal{L}(\theta\colon \mathbf{x}) = \log P(\mathbf{x}\colon \theta) = \log \sum_{\mathbf{s}} P(\mathbf{x}, \mathbf{s}\colon \theta) \tag{2.28}$$

The presence of the summation inside the logarithm makes it difficult to optimize Equation 2.28 analytically, because the marginal $P(\mathbf{x}\colon \theta)$ might not belong to the exponential family even if $P(\mathbf{x}, \mathbf{s}\colon \theta)$ does [15]. The expectation maximization algorithm is an alternative approach for finding the maximum likelihood in an iterative way. In EM, we refer to $\mathbf{X}$ as the *incomplete data*, and $\mathbf{X} \cup \mathbf{S}$ as the *complete data*. However, as we do not have access to the complete data, EM

resorts to the expected likelihood of the complete data. A general description of the EM procedure is provided in Algorithm 1. In the context of (dynamic) Bayesian networks, the described EM algorithm assumes the network structure is known and the goal is to perform parameter estimation, thus θ refers to model parameters.

Algorithm 1 Expectation-maximization algorithm.

Input: D: a set of data points of the form $\mathbf{X} = \{X_1, \ldots, X_n\}$; $\mathbf{S} = \{S_1, \ldots, S_\ell\}$: a set of latent variables.
Output: θ: model parameters for the distribution $P(\mathbf{x}, \mathbf{s})$.

1: Choose initial model parameters θ^{old}.
2: **repeat**
3: **E step**: Compute $P(\mathbf{s} \mid \mathbf{x}\colon \theta^{\text{old}})$
4: **M step**: Compute θ^{new} as follows:

$$\theta^{\text{new}} \leftarrow \arg\max_{\theta} Q(\theta, \theta^{\text{old}}) \tag{2.29}$$

$$\text{where} \quad Q(\theta, \theta^{\text{old}}) \stackrel{\text{def}}{=} \mathbb{E}_{\mathbf{s}}\left[\log P(\mathbf{x}, \mathbf{s}\colon \theta) \mid \mathbf{x}\colon \theta^{\text{old}}\right] \tag{2.30}$$

$$= \sum_{\mathbf{s}} P(\mathbf{s} \mid \mathbf{x}\colon \theta^{\text{old}}) \log P(\mathbf{x}, \mathbf{s}\colon \theta) \tag{2.31}$$

$$\theta^{\text{old}} \leftarrow \theta^{\text{new}} \tag{2.32}$$

5: **until** convergence of the log-likelihood or the model parameters is achieved
6: **return** θ^{new}

The goal of Algorithm 1 is to maximize the expected log-likelihood with regard to the latent states (Equation 2.30) in an iterative way. Algorithm 1 starts with initial model parameters θ^0, which is often randomly generated. Then, EM calculates the term of Equation 2.30 regarding the current parameters, i.e. $P(\mathbf{s} \mid \mathbf{x}\colon \theta^{\text{old}})$, which is known as the *E step*. Once this is done, it is possible to evaluate candidate model parameters for the expected log-likelihood. The goal now is to find new model parameters that optimizes this expectation, which is known as the *M step*. The process of generating new model parameter estimates from the current one is repeated until no improvement is possible, which means that a local maxima or a saddle point of the likelihood function has been achieved [65]. It is possible to show [15] that each cycle of E and M steps generates a new model that is at least as good as the previous one.

2.6.2 *The Baum-Welch algorithm*

Models with latent variables such as HMMs are often learned by means of the EM algorithm, which is also known as the Baum-Welch algorithm in this context [141]. In this section, we consider the learning of HMMs with fixed structure, for which learning reduces to parameter learning. We use independent HMMs to illustrate the necessary calculations. These involve estimating the parameters θ, which include the emission distributions $P(X_i^{(t)} \mid S^{(t)})$, with $X_i \in \mathbf{X}$ and $\mathrm{dom}(S) = \{s_1, \ldots, s_k\}$, the transitions $P(S^{(t+1)} \mid S^{(t)})$ and the initial probabilities $P(S^{(0)})$.

We consider the same learning setting as that of learning DBNs (Section 2.4.2), but in order to simplify the notation, we consider that the data D consists of a single sequence over $\{0, \ldots, T\}$. This can be easily extended to the case of multiple i.i.d. sequences. Let us denote by θ^{old} the parameters of current HMM-I and by θ the parameters of new HMM-I to be obtained. The Q function of Equation 2.30 for HMMs becomes:

$$Q(\theta, \theta^{\text{old}}) = \sum_{s^{(0:T)}} P(s^{(0:T)}, \mathbf{x}^{(0:T)} \colon \theta^{\text{old}}) \cdot \log P(\mathbf{x}^{(0:T)}, s^{(0:T)} \colon \theta) \tag{2.33}$$

For convenience, in Equation 2.33 the joint $P(s, \mathbf{x} \colon \theta^{\text{old}})$ was used. It holds that $P(s, \mathbf{x} \colon \theta^{\text{old}}) = P(\mathbf{x} \colon \theta^{\text{old}}) P(s \mid \mathbf{x} \colon \theta^{\text{old}})$, thus one can use either the joint or the conditional probability for this development because they are independent of θ. By substituting the joint of HMMs (Equation 2.26) into Equation 2.33, we obtain:

$$\begin{aligned} Q(\theta, \theta^{\text{old}}) = & \sum_{s^{(0:T)}} P(s^{(0:T)}, \mathbf{x}^{(0:T)} \colon \theta^{\text{old}}) \cdot \log P(s^{(0)}) \\ & + \sum_{s^{(0:T)}} P(s^{(0:T)}, \mathbf{x}^{(0:T)} \colon \theta^{\text{old}}) \left(\sum_{t=0}^{T} \sum_{i=1}^{n} \log P(X_i^{(t)} \mid s^{(t)}) \right) \\ & + \sum_{s^{(0:T)}} P(s^{(0:T)}, \mathbf{x}^{(0:T)} \colon \theta^{\text{old}}) \left(\sum_{t=0}^{T-1} \log P(s^{(t+1)} \mid s^{(t)}) \right) \end{aligned} \tag{2.34}$$

In order to maximize such Q function, one can maximize each term independently. The term referring to the initial distribution can be written as:

$$\sum_{s^{(0:T)}} P(s^{(0:T)}, \mathbf{x}^{(0:T)} \colon \theta^{\text{old}}) \cdot \log P(s^{(0)}) \tag{2.35}$$

$$= \sum_{s^{(0:T)}} P(s^{(0:T)}, \mathbf{x}^{(0:T)} \colon \theta^{\text{old}}) \cdot \log v(i) \tag{2.36}$$

By setting the derivative with respect to $v(i)$ to zero, and introducing the Lagrange α multiplier to ensure the constraint $\sum_{i=1}^{k} v(i) = 1$ we obtain:

$$\frac{\partial}{\partial v} \left(\sum_{s^{(0:T)}} P(s^{(0:T)}, \mathbf{x}^{(0:T)} \colon \theta^{\text{old}}) \cdot \log v(i) - \alpha \left(\sum_{i=1}^{k} v(i) - 1 \right) \right) = 0 \tag{2.37}$$

It is possible to show [14] that this results in:

$$\overline{v}(i) = \frac{P(\mathbf{x}^{(0:T)}, s_i^{(0)} : \theta^{\text{old}})}{P(\mathbf{x}^{(0:T)} : \theta^{\text{old}})} \tag{2.38}$$

By similar reasoning we obtain the transitions and emissions:

$$\overline{a}_{ij} = \frac{\sum_{i=0}^{T-1} P(\mathbf{x}^{(0:T)}, s_i^{(t)}, s_j^{(t+1)} : \theta^{\text{old}})}{\sum_{i=0}^{T-1} P(\mathbf{x}^{(0:T)}, s_i^{(t)} : \theta^{\text{old}})} \tag{2.39}$$

$$\overline{b}_j(\mathbf{k}) = \frac{\sum_{i=0}^{T} P(\mathbf{x}^{(0:T)}, s_j^{(t)} : \theta^{\text{old}}) \cdot \mathbb{1}(\mathbf{k})}{\sum_{i=0}^{T} P(\mathbf{x}^{(0:T)}, s_j^{(t)} : \theta^{\text{old}})} \tag{2.40}$$

where $\mathbb{1}(\mathbf{k}) = 1$ if $\mathbf{X}^{(t)} = \mathbf{x}_\mathbf{k}$, and 0 otherwise.

In order to actually compute the above probabilities, some useful quantities are defined:

$$\gamma_t(i) \stackrel{\text{def}}{=} P(s_i^{(t)} \mid \mathbf{x}^{(0:T)} : \theta^{\text{old}}) \tag{2.41}$$

$$\xi_t(i,j) \stackrel{\text{def}}{=} P(s_i^{(t)}, s_j^{(t+1)} \mid \mathbf{x}^{(0:T)} : \theta^{\text{old}}) \tag{2.42}$$

In many families of HMMs, the γ and ξ values can be efficiently computed by the forward-backward procedure based on dynamic programming [14, 15]. This fact is important for keeping the EM iterations efficient. Then, it is possible to show [141] that parameter update leads to $\theta^{\text{new}} = (\overline{A}, \overline{B}, \overline{v})$ given by:

$$\overline{v}(i) = \gamma_0(i) \tag{2.43}$$

$$\overline{a}_{ij} = \frac{\sum_{t=0}^{T-1} \xi_t(i,j)}{\sum_{t=0}^{T-1} \gamma_t(i)} \tag{2.44}$$

$$\overline{b}_j(\mathbf{k}) = \frac{\sum_{t=0}^{T} \gamma_t(j) \cdot \mathbb{1}(\mathbf{k})}{\sum_{t=0}^{T} \gamma_t(j)} \tag{2.45}$$

2.6.3 *Number of latent states*

One important hyperparameter of HMMs is the number of latent states. It is often the case that this hyperparameter is not known in advance, hence one can resort to estimating it using data. Information-theoretic criteria, such as AIC and BIC, can be used for this task [35, 138], with the advantage of low computational cost, something that is important for tasks as Bayesian-network structure learning.

Selecting the number of states can also be done by means of cross validation. While cross validation is typically more costly than penalized scoring functions, with cross validation one is able to directly estimate the performance of the model on new data, which allows for dealing with overfitting. A particular advantage of cross validation in HMMs is that it avoids assumptions made by penalized scoring functions, which require that the actual distribution of the data belongs to one of the models being compared [35].

One approach for selecting the number of states by means of cross validation is by incrementally increasing the number of states until model generalization stabilizes or deteriorates [102, 144]. To evaluate a model with q states by means of k-fold cross-validation, the training data is obtained from $(k-1)/k$ percent of sequences, while the validation data is obtained from the remaining sequences. The average log-likelihood of validation data is reported as the performance of the model, which can then be compared against, e.g., a model with $q+1$ states.

2.6.4 *Structure learning with missing data*

It is often the case that models with latent variables also need to have some of their structure estimated from data. For example, more general HMMs which adopt less restrictive assumptions on the probabilistic interaction among variables (see Figure 2.4) might require structure learning on the emission space, if there is no *a priori* domain knowledge about typical interactions. Using an inadequate or too restrictive model structure can considerably limit model emissions, which has been considered important for allowing HMMs to properly capture complex distributions [12, 102, 123] since it governs how observations are emitted to the external world.

In the situation of learning with latent variables and unknown structure, one can resort to an extension of the EM algorithm called *structural EM* (SEM, for short) [65, 67]. This is the case of Bayesian networks with latent variables and unknown structure, as well as standard HMMs (Equation 2.27), where the emission distribution follows an arbitrary Bayesian network, thus the observable variables might not be independent given the state.

In the following development, we consider that the model to be learned is a Bayesian network with structure $\mathcal{G}$ and parameters θ (for convenience, we use θ instead of P in this algorithm). The SEM procedure is described as follows:

1. Choose $\mathcal{B}^0 = (\mathcal{G}^{(0)}, \theta^{(0)})$ randomly
2. Loop for $n = 0, 1, \ldots$ until convergence:
   a) Find $\mathcal{G}^{\text{new}}$, as follows:

$$\mathcal{G}^{\text{new}} \leftarrow \arg\max_{\mathcal{G}} Q((\mathcal{G}, \theta_{\mathcal{G}}), (\mathcal{G}^{\text{old}}, \theta^{\text{old}})) \tag{2.46}$$

   where

$$Q(\mathcal{B}, \mathcal{B}^{\text{old}}) \stackrel{\text{def}}{=} \mathbb{E}_s \left[\log P(\mathbf{x}, s \colon \mathcal{B}) - \text{Pen}(\mathcal{B}, \mathbf{x}) \mid \mathbf{x}, \mathcal{B}^{\text{old}} \right] \tag{2.47}$$

b) Let $\theta^{\text{new}} \leftarrow \arg\max_\theta Q((\mathcal{G}^{\text{new}}, \theta), (\mathcal{G}^{\text{old}}, \theta^{\text{old}}))$

One important insight of SEM is placing structure learning inside EM calculations. In the SEM procedure, each iteration involves structure learning and parameter learning, both based on expected counts. The expected counts are based on probabilities computed over the current model $\mathcal{B}^{\text{old}}$. In step(a), structure learning is executed based on the expected counts of the current parameters θ^{old}. As SEM involves selecting model structure, an additional factor is included in the expected score to penalize for model complexity. Once this is done, in step(b) new parameters θ^{new} are computed for this structure $\mathcal{G}^{\text{new}}$. Note that step(b) can be seen as the parametric part of SEM.

It is possible to show [65] that in order to guarantee convergence it is sufficient to find a BN $\mathcal{B}^{\text{new}}$ with improved score compared to the BN found on the previous SEM iteration, instead of maximizing the expected score shown above. This is useful because it is often the case that heuristic search is used for obtaining $\mathcal{B}^{\text{new}}$, which does not guarantee that in each iteration the expected score is maximized.