31. Probabilistic Modeling in Machine Learning

Davide Bacciu, Paulo J.G. Lisboa, Alessandro Sperduti, Thomas Villmann

Probabilistic methods are the heart of machine learning. This chapter shows links between core principles of information theory and probabilistic methods, with a short overview of historical and current examples of unsupervised and inferential models. Probabilistic models are introduced as a powerful idiom to describe the world, using random variables as building blocks held together by probabilistic relationships. The chapter discusses how such probabilistic interactions can be mapped to directed and undirected graph structures, which are the Bayesian and Markov networks. We show how these networks are subsumed by the broader class of the probabilistic graphical models, a general framework that provides concepts and methodological tools to encode, manipulate and process probabilistic knowledge in a computationally efficient way. The chapter then introduces, in more detail, two topical methodologies that are central to probabilistic modeling in machine learning. First, it discusses latent variable models, a probabilistic approach to capture complex relationships between a large number of observable and measurable events (data, in general), under the assumption that these are generated by an unknown, nonobservable process. We show how the parameters of a probabilistic model involving such nonobservable information can be efficiently estimated using the concepts underlying the expectation–maximization algorithms. Second, the chapter introduces a notable example of latent variable model, that is of particular relevance for representing the time evolution of sequence data, that is the hidden Markov model. The chapter ends with a discussion on advanced approaches for modeling complex data-generating processes comprising nontrivial probabilistic interactions between latent variables and observed information.

31.1 Probabilistic and Information-Theoretic Methods

Information theory is closely connected to probability theory and statistics. In particular, the standard definition of information contained in a random variable X with a probability density function $P(X)$ is well known to be $I(X) = -\log(P(X))$, with the corresponding *Shannon entropy*, in differential form, given by the average information

$$H(P) = -\int P(X) \log(P(X)) \, \mathrm{d}x \,. \tag{31.1}$$

One of the fundamental theorems of information theory, the second Gibbs theorem, states that the normal distribution achieves maximum entropy, hence maximal average information from all distributions with known variance. To show this in the univariate case, consider the normal distribution in the standard form

$$P(X) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(X-\mu)^2}{2\sigma^2}\right).$$

It is straightforward to show that for the natural logarithm

$$-\int P(X)\log(P(X))\mathrm{d}x = \frac{1}{2} + \log\left(\sqrt{2\pi\sigma^2}\right) = -\int G(X)\log(P(X))\mathrm{d}x\,,$$

where $G(X)$ is any arbitrary density function with variance $\int G(X)(X-\mu)^2\mathrm{d}x = \sigma^2$. Therefore, the difference in average information between the two density functions necessarily observes the following

$$\begin{aligned}
&-\int P(X)\log(P(X))\mathrm{d}x + \int G(X)\log(G(X))\mathrm{d}x \\
&= -\int G(X)\log(P(X))\mathrm{d}x + \int G(X)\log(G(X))\mathrm{d}x \\
&= -\int G(X)\log\left(\frac{P(X)}{G(X)}\right)\mathrm{d}x \\
&\geq \int G(X)\left(1-\frac{P(X)}{G(X)}\right)\mathrm{d}x = 0
\end{aligned}$$

using Jensen's inequality $\log(x) \leq x-1$ and the normalization property $\int P(X) = \int Q(X) = 1$. This is a particular instance of Gibbs inequality and proves that the asymptotic distribution of the *central limit theorem* also maximizes entropy.

This led, in probability theory, to the definition of natural measures of dissimilarity closely related to the expectation of information difference, e.g., the *Kullback–Leibler* (KL) divergence [31.1]

$$D_{\mathrm{KL}}(P\|Q) = \int P(X)\log\left(\frac{P(X)}{Q(X)}\right)\mathrm{d}x\,, \qquad (31.2)$$

as generalized distances between probability distributions P and Q.

The KL divergence occurs frequently in machine learning, where the development of learning strategies links information theory with statistical and biologically motivated concepts. For instance, the perceptron model was established as a simple but mathematically tractable model of a biological neuron as the smallest information processing unit in brains [31.2]. Recognition that gradient descent provided a pragmatic but effective solution to the credit assignment problem, namely which values the hidden nodes should have, led to the multilayer perceptron as powerful computational tools for classification and regression. Initially *maximum likelihood* optimization was used for parameter estimation, following the tried and tested statistical concepts of normally distributed errors leading to a sum-of-squares loss function in regression and, for classification, the Bernoulli distribution for binary data and the so-called cross-entropy (31.2) for multinomial class assignments, the latter two likelihood functions measuring information divergence averaged over the true distribution given by the empirical class labels.

Information theoretic aspects (e.g., mutual information) were also considered in neural models in order to avoid overtraining [31.3], for instance in Boltzmann networks which directly mirror information principles in statistical mechanics [31.4]. Related approaches are used currently for deep learning models, where information principles drive the feature representations [31.5].

The correspondence between maximum entropy and maximum likelihood outlined above is just one aspect of the application of information-theoretic concepts in machine learning. The next section outlines further developments linked first to source identification through blind signal separation and matrix factorization methods. These concepts from signal processing identify important degrees of freedom that may be used as hidden variables in probabilistic models, discussed later in the chapter. Furthermore, the application of information-theoretic methods extends also to the automatic identification of prototypes for use in compact data representations that include *dictionaries* defined by methods such as vector quantization, typically with unsupervised approaches.

Supervised methods are introduced as probabilistic models, focusing first on discriminative methods. This indicates that the maximum likelihood approach is limited in its predictive power in generalization to out-of-sample data, because it allows models to be generated with very little bias but with considerable variance – for a more detailed discussion of this point refer [31.6]. What this means in practice is that flexible models such as neural networks are prone to overfitting unless the complexity of the model is controlled along with the extent to which the model fits the data. The

latter is described by the likelihood, but the model complexity can be controlled in a number of different ways. In probabilistic models an efficient framework to maximize the generality of probabilistic inference models is to apply the maximum a posteriori (MAP) framework which optimizes the posterior probability of the model parameters given the data but also given prior distributions for the parameters, typically limiting their size by assuming a zero-centred normal distribution as the prior. This is the basis of the method of *automatic relevance determination*, explained in Sect. 31.2.

While discriminative models are efficient approximators for nonlinear response functions, both in regression and in the estimation of class conditional density functions, they are difficult to interpret and can generally be considered as black boxes, meaning that they are not readily interpreted to give insights about the data. A topical and widely used alternative approach is to model the joint distribution of the data directly. This is ideally done by factorization into subgraphs into which the multivariate structure of the data is broken-up using strict conditional independence requirements, as discussed in Sect. 31.2. Inference can then proceed using Bayes theorem introduced in (31.6).

An alternative approach to modeling the joint distribution of the covariates is to use the mutual correlation in the data to identify important degrees of freedom that may be *hidden* in the sense that they are not directly observed. This generates *latent variable* representations that naturally fit into the framework of probabilistic modeling. However, the introduction of additional variables also introduces complexity into the optimization process for estimating their values. This leads naturally to the introduction of *expectation maximization* (EM), a general approach of particular value for estimating mixture models, discussed in Sect. 31.3.

So far the modeling methodologies focus on snapshots of the data, without taking into consideration the time evolution of the covariates. To do this requires explicit parametrization, for which arguably the most widely used probabilistic approach is *hidden Markov models* (HMM). These models are build on the concepts of conditional independence, latent variables, and expectation maximization to model the time evolution of sequences of covariate measurements, in the last substantive Sect. 31.4

31.1.1 Information-Theoretic Methods

While the statistical properties of perceptrons are widely investigated [31.6], the more difficult problem of establishing statistical independence is becoming increasingly important and novel algorithms have been presented during the last decade [31.7]. Their applicability is enormous, ranging from variable selection, to *blind source separation* (BSS) and statistical causality. Frequently, the difficult question of statistical dependence in data is replaced by the easier consideration of estimation and application of data correlations for learning strategies. A recent approach tries to determine independence by generalized correlation functions [31.8]. In this context of decorrelation and independence, BSS and nonnegative matrix factorizations [31.9] of data channels are based on statistical deconvolution. A comprehensive overview for BSS, independent component analysis (ICA) and nonnegative matrix and tensor factorization (NMF) can be found in [31.10–12], respectively. Different aspects can be investigated, like ICA and BSS maximizing conditional probabilities [31.11]. A relevant connection exists between NMF and probabilistic graphical models comprising hidden variables [31.13], which is briefly discussed in Sect. 31.3.4.

Other recent approaches in this field incorporate information theoretic principles directly: *Pham* [31.14] investigated BSS based on mutual information, whereas [31.15] applied β-divergences. The *infomax* principle for ICA was considered in depth [31.16], as was the problem of learning overcomplete data representations and performing overcomplete noisy blind source separation, e.g., the sparse coding neural gas (SCNG) [31.17]. Recent results including modern divergences (generalized α-β-divergences) were recently published [31.18]. Obviously, information theoretic divergence measures like Rényi-divergences (belonging to the family of α-divergences) capture directly the statistical information contained in the data, as expressed by the probability density function [31.19, 20]. This property can be used for unsupervised model estimation for instance in vector quantization, when divergences are used as dissimilarity measure [31.21].

Information optimum vector quantization by prototypes is a widely investigated topic in clustering and data compression, based on the optimization of the γ-reconstruction error

$$E_{VQ}(\gamma) = \int \|\mathbf{v} - \mathbf{w}(\mathbf{v})\|_E^{\gamma} P(V = \mathbf{v}) \, \mathrm{d}\mathbf{v} \,,$$

where $P(V = \mathbf{v})$ is the data density of the vector data $\mathbf{v}$ and $\|\mathbf{v} - \mathbf{w}(\mathbf{v})\|_E$ is the Euclidean distance of the data vector and the prototype $\mathbf{w}(\mathbf{v})$ representing it. One of

the key results concerning information theoretic principles for vector quantization is *Zador*'s magnification law [31.22]: if the data vectors $\mathbf{v}$ are given in q-dimensional Euclidean space, then the magnification law $\rho \sim P^{\alpha}$ holds. Here, $\rho(\mathbf{w})$ is the prototype density with the magnification factor

$$\alpha = \frac{q}{q+\gamma} \, .$$

This is the basic principle of vector quantization based on Euclidean distances. For different schemes like self-organizing maps, *Neural Gas* variants with slightly different magnification factors are obtained depending on the choice of neighborhood cooperation scheme applied during prototype adaptation [31.23–25]. Information optimum magnification for $\alpha = 1$ is equivalent to maximum mutual information [31.22]. Yet, it is possible to control the magnification for most of these algorithms by different strategies like localized or frequency sensitive competitive learning. For an overview, we refer to [31.23]. If the Euclidean distance is replaced by divergence measures, optimum magnification $\alpha = 1$ can also be achieved by maximum entropy learning [31.26], or by the utilization of correntropy [31.27]. Vector quantization algorithms directly derived from information theoretic principles based on Rényi entropies are intensively studied in [31.28], also highlighting its connection to graph clustering and Mercer kernel-based learning [31.29].

Other information theoretic vector quantizers optimize the mutual information between data and prototypes, or the respective KL divergence, instead of minimizing a reconstruction error [31.30]. Based on this principle, several data embedding, or dimensionality reduction techniques, have been developed as alternatives to multidimensional scaling. These approaches are frequently used to visualize data. Prominent examples are *stochastic neighborhood embedding* (SNE) [31.31] or variants thereof: for instance, t-SNE uses outlier-robust Student-t-distributions for data characterization instead of Gaussians [31.32]. The generalization to other divergences than KL can be found in [31.33].

Another role for information theory in machine learning is in *feature selection*. Removing irrelevant or redundant features not only leads to a simplification of the model and a reduced requirement for data acquisition, but it is also central for maximizing the generality of the model when it is applied to future data. Most feature selection approaches are supervised schemes, hence using class information or expected regression values. Strategies to achieve this goal can be classical Bayesian inference schemes of which *automatic relevance determination* (ARD) is a good example (described further in Sect. 31.2), or statistical approaches based on mutual correlation or covariances [31.34, 35]. An alternative approach to feature selection is to use *mutual information*

$$I(X,Y) = D_{\mathrm{KL}}(J(X,Y) \| P(X)Q(Y))$$

between random variables X and Y with probability densities P and Q, respectively, and joint density J [31.36]. Here, the features are treated as random variables to be compared and mutual information measures the information loss resulting from removal of variables from the model. Learning classification together with feature weighting in vector quantization is known as *relevance learning* [31.37]. Recent developments to introduce sparseness according to information theoretic constraints are discussed in [31.38, 39].

Information-theoretic measures such as mutual information, can be explicitly estimated from data [31.40]. This is used in the context of vectorial data analysis to obtain consistent and reliable estimators with topographic maps or kernels [31.41]. Further applications of information theoretic learning also use Rényi entropy

$$H_a(P) = \frac{1}{1-\alpha} \log \left(\int (P(X))^{\alpha} \, \mathrm{d}x \right)$$

as a cost function instead of the mean squared error, resorting, for computational efficiency, to Parzen estimators [31.42] or nearest neighbor entropy estimation models. For effective computation of an approximate of the mutual information $I(X,Y)$, the quadratic Rényi entropy $H_2(p)$ or the closely related information energy are common choices [31.43]. Parzen window-based estimators for some information theoretic cost functions have also been shown to be cost functions in a corresponding Mercer kernel space [31.44]. In particular, a classification rule based on an information theoretic criterion has been shown to correspond to a linear classifier in the kernel space. This leads to the formulation of the support vector machine (SVM) from information theory principles.

31.1.2 Probabilistic Models

Kernel models are known for having excellent discrimination performance, but they are typically not well

calibrated. This is because they are designed to be efficient binary class allocation models rather than estimators of the posterior probability for membership of each class C. As an example, SVMs allocate inputs to classes on the basis on a binary-valued indicator variable that generally does not have a link function to a probability density estimate. This type of models is known as discriminative models, a well-known variant being Fisher's linear discriminant. As the name implies, the central model is linear in the covariates,

$$y = w^T x$$

optimizing, for binary classification, a discriminant function derived from the mean m_i and variance s_i of each class (i. e., $i = 1, 2$), namely

$$J(w) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2} .$$

In general, given the two data cohorts, the covariance matrix of the data S has a strict decomposition into within- and between-class covariance matrices as $S = S_w + S_b$. For an overall data mean vector m and a total of N_j data points in each class, these matrices are given by

$$S = \sum_{i=1}^{N} \left((x_i - m)^T (x_i - m)\right) ,$$

$$S_w = \sum_{j=1}^{2} \sum_{i=1}^{N} \left((x_i - m_j)^T (x_i - m_j)\right) ,$$

$$S_b = \sum_{j=1}^{2} \left(N_j (m_j - m)^T (m_j - m)\right) .$$

The solution to the optimization of $J(w)$ is

$$w \propto S_w^{-1} (m_2 - m_1) ,$$

where the inverse of the within-class covariance matrix S_w positions the discriminant hyperplane so as to minimize the overlap between the projections of the data points in each class onto the direction of the weight w. This illustrates the observation that, in general, this projection will not be calibrated with a probabilistic estimate such as the logit

$$\text{logit}(P(C|X)) = \log \left(\frac{P(C|X)}{1 - P(C|X)} \right) .$$

The correct calibration is found in a class of generalized linear models of the form

$$y(x) = f(w^T x + w_0) ,$$

where $f(\cdot)$ is known as the *activation function* in machine learning and its inverse is called a *link function* by statisticians [31.6]. Perhaps the best-known choice of activation is the sigmoid function, where the probabilistic model becomes logistic regression and the linear index $w^T x$ represents exactly the logit $(P(C|X))$. This is very widely used and a generally well-calibrated model, even when severe class imbalance is present.

It is often quoted that generalized linear models are limited by the discriminant forms determined by the linear scores, which must therefore be hyperplanes. However, this ignores the observation that, in most practical applications, suitable attribute representations are defined using domain knowledge, typically by binning variables into discrete states. This turns the probabilistic estimators into linear-in-the-parameter models with significant discrimination potential for nonlinearly separable data. In effect, if the link function is properly tuned to the noise structure of the data and in particular when there are larger numbers of independent covariates, well-designed generalized linear models are competitive with flexible machine learning models, the more so as the limitation of using a linear-in-the-parameters scoring index now works as a form of regularization limiting the complexity of the model. Moreover, the linear index provides a strong element of interpretability whose importance to application domain experts cannot be overestimated. Notwithstanding the power of machine learning, generalized linear models should always be used as benchmarks to set against nonlinear models.

An alternative to probabilistic linear models is the wide range of flexible direct estimators of $P(C|X)$ among which arguably the most widely used model remains the multilayer perceptron (MLP). Similarly to linear statistical models, however, it is important to note that the estimation of class conditional probabilities with an MLP is contingent on using a correct activation function at the output node together with a suitable choice of loss function, which must be one of the entropy functions outlined in the previous section. So, in binary classification, the log-likelihood function with a Bernoulli distribution should be used in conjunction with a sigmoid activation function. In the multinomial case, we would need an extension of the sigmoid function, the softmax activation, together

with the cross-entropy as the loss function, since this is the correct measure of the divergence between the estimated and observed probability density functions. Similarly, for nonlinear regression, the activation function should be linear with the usual sum-of-squares error function, provided the inherent noise in the data can be assumed to be normally distributed with zero mean, since this is where the loss function is derived from. In the event where the noise variance, for instance, is dependent on the covariates, heteroscedastic noise models must be used to derive appropriate loss functions [31.6].

While the strength of neural networks is their universal approximation capability, in the sense of fitting any multivariate surface to an arbitrarily small error, this flexibility also makes them prone to overfitting, potentially resulting in data models with little bias but large variance, in direct contrast to generalized linear models. In both cases, it is necessary to control the complexity of the model and this is best done by adding a penalty term to enforce the principle of parsimony, colloquially known as Occam's razor (*lex parsimoniae*). Arguably, the most commonly used and effective scheme is to apply Bayes' theory at the level of fitting the model parameters, then to the regularization hyperparameters, and finally to model selection itself.

As we saw previously, the output of the MLP represents a direct estimate of the posterior probability of class membership $P(C|X)$. This approach can be generalized for the analysis of longitudinal data where each individual subject is follow up over a period of time starting with a defined recruitment point and ending either at the end of a defined observation period or when an event of interest is observed, whichever occurs first. This is often called *survival modeling* and is typically used to estimate event rates in the presence of censorship, e.g., where the outcome of interest, for instance recovery from an illness, is observed in some subjects for only part of the allowed period of follow-up due to other events taking over, such as another condition setting-in, which prevent the observer from ever knowing whether or not the subject would have recovered from the original illness, which is the event of interest. For discrete time, these models can be estimated using the standard MLP with an additional input node coding the time intervals. The output of the MLP again represents a conditional probability, but now the probability of the subject surviving each time interval given that the subject survived until the start of the time interval. This defines the hazard function $h_l(\mathbf{x}_i)$, for subject with covariate vector $\mathbf{x}_i$ and predictions over the lth discrete time interval, which is given by

$$h_l(\mathbf{x}_i) = P(T \leq t_l | T > t_{l-1}, \mathbf{x}_i) \, .$$

For a single event of interest, i. e., a single *risk* factor, the log-likelihood function exactly mirrors that used in binary classification, treating as independent the probability estimates for each of the N subjects and over the discrete time intervals where the subject was observed, i. e., up to the end of the follow-up period or until censorship. This leads to the following loss function

$$L_B = \prod_{i=1}^{N} \prod_{l=1}^{l_i} \left\{ h_l(\mathbf{x}_i)^{d_{il}} \left[1 - h_l(\mathbf{x}_i) \right]^{(1-d_{il})} \right\} \, , \qquad (31.3)$$

where the binary indicator variable $d_{il} = 0$ if the event of interest was not observed for the subject during the specific time interval, and is 1 otherwise. This loss function is known as a partial likelihood, since it is measured only over time periods where the outcomes for each subject are observed, an approach that has been extended to the multinomial case to provide a rigorous treatment of censorship with flexible models in the context known as competing risks [31.45].

Application of the Bayesian regularization framework consists in maximizing the posterior probability for the model parameters w, given the data set $\mathcal{D}$, the regularization hyperparameters α and the choice of the model structure, e.g., selected covariates H, namely

$$P(w|\mathcal{D}, \alpha, H) = \frac{P(\mathcal{D}|w, \alpha, H) P(w|\alpha, H)}{P(\mathcal{D}|\alpha, H)} \, . \qquad (31.4)$$

The first term on the right-hand side of Eq. (31.4) denotes the probability of the model fitting the data, represented by the exponential of the entropy term discussed in the introduction and defined for longitudinal data by (31.3), hence

$$P(\mathcal{D}|w, \alpha, H) = \mathrm{e}^{-L_B} \, .$$

The second term in (31.4) represents a *prior* distribution of the model parameters typically with a quadratic loss term corresponding to independent zero-mean univariate Gaussian distributions, sometimes called weight decay terms. A particularly efficient implementation of Bayesian regularization is to assign a separate weight decay term to each covariate, indexing the covariates by m of which there are N_α, with the N_m hidden nodes indexed by n. This allows each covariate to be separately turned on or off depending on how informative it is

for fitting the observations about the outcome variable, a process known as *automatic relevance determination* (ARD) [31.4]. Expressed in full, this gives

$$P(w|\alpha,H) = \frac{\mathrm{e}^{-G(w,\alpha)}}{Z_w(\alpha)}\,, \quad \text{where} \quad G(w,\alpha) = \frac{1}{2}\sum_{m=1}^{N_\alpha}\alpha_m\sum_{n=1}^{N_m}w_{mn}^2 \quad \text{and} \quad Z_w = \prod_{m=1}^{N_\alpha}\left(\frac{2\pi}{\alpha_m}\right)^{\frac{N_m}{2}}\,.$$

In principle, the best values for the regularization hyperparameters, i. e., the weight decay parameters α, are those which minimize their *posterior* probability

$$P(\alpha|\mathcal{D},H) = \frac{P(\mathcal{D}|\alpha,H)P(\alpha|H)}{P(\mathcal{D}|H)}\,.$$

However, the denominator of (31.4) cannot be obtained in closed form, so a Laplace approximation is typically around a stationary point in the loss function as a function of the weights. This amounts to a local Taylor expansion of

$$P(\mathcal{D}|\alpha,H) = \int P(\mathcal{D}|w,\alpha,H)P(w|\alpha,H)\mathrm{d}w = \int \frac{\mathrm{e}^{-S(w,\alpha)}}{Z_w(\alpha)}\mathrm{d}w\,,$$

where the linear term in the weights vanishes because of stationarity leading to

$$S^*(w,\alpha) \approx S(w^{\mathrm{MP}},\alpha) + \frac{1}{2}(w-w^{\mathrm{MP}})^T A(w-w^{\mathrm{MP}})\,,$$

from which the posterior probability for the hyperparameter results

$$P(\alpha|\mathcal{D},H) \propto \frac{\mathrm{e}^{-S(w^{\mathrm{MP}},\alpha)}}{Z_W(\alpha)}(2\pi)^{\frac{N_W}{2}}\det(A)^{-1/2}\,.$$

In practice, what this means is that the log-odds ratio, given by the activation of the output node of the MLP can be assumed to have a univariate normal distribution whose variance is given by the Hessian of the matrix S with respect to the weights; g is the gradient of the activation a with respect to the weights, namely

$$P(a|X,\mathcal{D}) = \frac{1}{1}(2\pi s^2)^{1/2}\mathrm{e}^{-\left(\frac{(a-a_{\mathrm{MP}})^2}{2s^2}\right)}$$

with a_{MP} denoting the most probable value of the activation function, i. e., the direct output of the MLP without marginalization, and

$$s^2(x) = g^T A^{-1} g\,.$$

The so-called marginalized estimate of the MLP output is now the posterior distribution integrated over the activation a. In the above expression, g is the gradient of the activation with respect to the network weights and A is the corresponding Hessian; hence the matrix of second partial derivatives. For binary classification and single-risks modeling, this is given by a neat analytical expression

$$h(x_i,l) = \int g(a)P(a|X_i=x_i,l,\mathcal{D})\mathrm{d}a = g\left(\frac{a^{\mathrm{MP}}(x_i,l)}{\sqrt{1+(\pi/8)g^T A^{-1}g}}\right) \tag{31.5}$$

with $g(\cdot)$ denoting the sigmoid function. This adjustment to the original MLP output, i. e., a^{MP}, shows the regularization process in operation: stationary points, where the weights are well defined, have small variance s^2 and therefore their value remains almost unchanged. Conversely, flat valleys in the loss function, where stationary points for the weights have broad Gaussian distributions, are penalized by reducing the value of the argument of the sigmoid function in (31.5) toward nil, reflecting an increase in uncertainty by shifting the MLP output toward the *don't know* threshold.

A probabilistic alternative to discriminative approaches consists of generative models, where Bayes' theorem is once again put into practice to estimate the *posterior* probability of class membership $P(C_k|X)$, from the *class conditional density* functions $P(X|C_k)$ and *prior* probabilities for the classes $P(C_k)$, that is

$$P(C_k|X) = \frac{P(X|C_k)P(C_k)}{\sum_k P(X|C_k P(C_k))}\,, \tag{31.6}$$

where classes are indexed by k and the sum-rule has been used to expand the denominator. Suitable models for the probability density functions (pdf) of the data given each class will depend on the nature of the data. However, it is straightforward to show for two classes that if the pdfs are normal distributions with equal variance, then the posterior probability will have exactly the functional form of the logistic regression model. This can be taken as an explanation in probabilistic terms of the potential limitations of this linear model, since different classes in practice tend to have distinct variances, even when that data sets for each class are approximately normally distributed. A natural

extension of this approach is to use a mixture of Gaussian distributions. This is a very flexible model that can parameterize also multimodal density functions. In the interest of space, we refer the interested reader to a standard textbook [31.6].

The two approaches of discriminative and generative models may be combined by using generative models to build kernels. These kernels define similarity between two covariate vectors $\mathbf{x}$ and $\mathbf{x}'$ by correlation between the respective pdfs, with the values of the kernel function given by $k(\mathbf{x}, \mathbf{x}') = P(X = \mathbf{x}) \cdot P(X' = \mathbf{x}')$ for suitable choices of the probability functions. A kernel so designed will naturally form a Gram matrix. Such kernels lead naturally to the use of latent variables

$$k(\mathbf{x}, \mathbf{x}') = \sum_i P(X = \mathbf{x}|Z = i)P(X' = \mathbf{x}'|Z = i) \times P(Z = i) ,$$

with weighting coefficients $P(Z)$ reflecting the strength of the latent variable Z indexed by i. An example of this approach in practice will be seen in the HMMs later in this chapter (see Sect. 31.4.2).

31.2 Graphical Models

In this section, we give a basic introduction to *graphical models*, a general framework for dealing with uncertainty in a computationally efficient way. Probabilistic models that we treat in the next sections belong to this framework. Here, we introduce the two main classes of graphical models, Bayesian and Markov networks, discussing different methods for performing probabilistic inference. Specific instances of learning within this framework, are given in the next two sections. For the sake of presentation, here we limit our presentation to discrete random variables; however, graphical models can be defined on continuous variables or mixed variables. The material covered in this section is based on [31.6, 46, 47].

A graphical model allows us to represent a family of joint probability distributions in terms of a directed or undirected graph, where nodes are associated with random variables, and edges represent some form of direct probabilistic interaction between variables. Being able to compactly represent the joint probability distribution of a set of random variables $\mathcal{X} = \{X_1, \ldots, X_n\}$ is very important: any probabilistic query involving the variables $X_1, \ldots, X_n$ can be answered by knowing their joint probability distribution $P(X_1, \ldots, X_n)$. For example, assume variables to be discrete, and suppose we want to know the posterior probability of X_1 and X_2 given all the other variables, i. e., $P(X_1, X_2|X_3, \ldots, X_n)$. We can easily answer this query by computing

$$P(X_1, X_2|X_3, \ldots, X_n) = \frac{P(X_1, \ldots, X_n)}{\sum_{\substack{X_1 \in dom(X_1) \\ X_2 \in dom(X_2)}} P(X_1 = x_1, X_2 = x_2, X_3, \ldots, X_n)} .$$

Unfortunately, storing the joint probability values associated with all the different assignments $x_1, \ldots, x_n$ is not feasible: if d_i is the size of $dom(X_1)$, all the different assignments are $\prod_{i=1}^{n} d_i$, i. e., an exponential number of entries. This situation, however, constitutes the worst case. In fact, in many application domains, independence properties allow us to factorize the joint distribution into compact parts which can be stored efficiently. Graphical models provide the *language* to compactly represent these factors, enabling in many cases inference and learning over a compact parameterization of the joint distribution as graphical manipulations.

Graphical models can be characterized according to the type of probabilistic interaction between variables they model. Directed graphs (*Bayesian networks*) are used to express causal relationships between random variables (i. e., *cause* → *effect* relationships), while undirected graphs (*Markov networks*) are better suited to express probabilistic constraints among subset of variables to which it is difficult to ascribe a directionality (graphical models containing both directed and undirected edges are possible; however, they will not be covered here). In both cases, the joint distribution is factorized according to the notion of *conditional independence*.

Definition 31.1 Conditional Independence

Let $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{Z}$ be sets of random variables with $X_i \in \mathcal{X}$, $Y_i \in \mathcal{Y}$, $Z_i \in \mathcal{Z}$. $\mathcal{X}$ is conditionally independent of $\mathcal{Y}$ given $\mathcal{Z}$ (denoted as $\mathcal{X} \perp\!\!\!\perp \mathcal{Y}|\mathcal{Z}$) in a distribution P if, for all values $x_i \in dom(X_i)$, $y_i \in dom(Y_i)$, $z_i \in dom(Z_i)$

$$P(\mathcal{X} = \mathbf{x}, \mathcal{Y} = \mathbf{y}|\mathcal{Z} = \mathbf{z}) = P(\mathcal{X} = \mathbf{x}|\mathcal{Z} = \mathbf{z}) \times P(\mathcal{Y} = \mathbf{y}|\mathcal{Z} = \mathbf{z}) ,$$

where $\mathcal{X} = \mathbf{x}$ denotes $X_1 = x_1, \ldots, X_{n_X} = x_{n_X}$, $\mathcal{Y} = \mathbf{y}$ denotes $Y_1 = y_1, \ldots, Y_{n_Y} = y_{n_Y}$, $\mathcal{Z} = \mathbf{z}$ denotes $Z_1 = z_1, \ldots, Z_{n_Z} = z_{n_Z}$, and $n_X = |\mathcal{X}|$, $n_Y = |\mathcal{Y}|$, $n_Z = |\mathcal{Z}|$.

It is not difficult to see that if $\mathcal{X} \perp\!\!\!\perp \mathcal{Y} | \mathcal{Z}$, then it is also true that $P(\mathcal{X}|\mathcal{Y}, \mathcal{Z}) = P(\mathcal{X}|\mathcal{Z})$. In fact, using the product rule for probabilities, we have $P(\mathcal{X}, \mathcal{Y}|\mathcal{Z}) = P(\mathcal{X}|\mathcal{Y}, \mathcal{Z})P(\mathcal{Y}|\mathcal{Z})$.

In the following, we will discuss how conditional independence is used within Bayesian and Markov networks to factorize the joint distribution. Inference and learning will be discussed as well.

31.2.1 Bayesian Networks

Bayesian networks are directed acyclic graphs used to model causal relationships between random variables: an edge $X_1 \to X_2$ is used to express the fact that variable X_1 (*cause*) influences variable X_2 (*effect*). The combination of this interpretation in conjunction with the exploitation of conditional independence, where applicable, allows the efficient probabilistic modeling of many relevant application domains. In general, the product rule can be used to factorize the joint distribution of variables $X_1, X_2, X_3, \ldots, X_n$ as

$$P(X_1, X_2, X_3, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | X_1, X_2, \ldots, X_{i-1}) \,. \tag{31.7}$$

The conditional independence relationships can be used to *simplify* the form of each factor in (31.7), i. e., by eliminating variables from the conditioning part, thus drastically reducing the number of probability values that need to be specified to define the factor. For example, if we assume that all the variables are Boolean, then the number of entries needed to define $P(X_n|X_1, X_2, \ldots, X_{n-1},)$ would be 2^{n-1}. If we consider a simple scenario in which the variable X_n is dependent only on X_{n-1}, the corresponding simplified factor becomes $P(X_n|X_1, X_2, \ldots, X_{n-1}) = P(X_n|X_{n-1})$, which only requires two entries.

The *naïve Bayes* model used in classification tasks can be understood as a Bayesian network, where the variable associated with the class label C is the cause and the variables $X_1, \ldots, X_n$ used to describe the attributes of the current input are the effects. The underlying conditional independence assumption is fairly simplistic, but allows a very parsimonious factorization of the joint distribution. By assuming that the class label does not depend on the attributes, and that the attributes are conditionally independent with respect to each other given the class label, i. e., $\forall i, j \; P(X_i, X_j|C) = P(X_i|C)P(X_j|C)$, *naïve Bayes* factorizes the joint distribution as

$$P(C, X_1, X_2, X_3, \ldots, X_n) = P(C) \prod_{i=1}^{n} P(X_i|C) \,.$$

The details of this model are not discussed in this chapter, but a good didactic reference is [31.6].

In general, after simplification via conditional independence, factors are in the form $P(X_i|X_{j_1}, \ldots, X_{j_k})$, where $X_{j_1}, \ldots, X_{j_k}$ are denoted as *parents* of X_i, and the notation $\text{pa}(X_i)$ is used with the following meaning $\text{pa}(X_i) = \{X_{j_1}, \ldots, X_{j_k}\}$. The factor associated with variable X_i can thus be rewritten as $P(X_i|\text{pa}(X_i))$ and the joint distribution as

$$P(X_1, X_2, X_3, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|\text{pa}(X_i)) \,. \tag{31.8}$$

The graphical representation of a Bayesian network is shown in Fig. 31.1. The graphical model includes one node for each involved variable. Moreover, a variable that is conditioned (effect) with respect to a parent one (cause) receives a directed edge from that variable. For example, in the Bayesian network represented in Fig. 31.1, we have $\text{pa}(X_7) = \{X_2, X_3\}$, i. e., the set constituted by the two nodes from which X_7 receives an edge. This means that the factor associated with X_7 is $P(X_7|X_2, X_3)$. In Fig. 31.1, we have reported one popular way to specify the parameters of $P(X_7|X_2, X_3)$ when the involved variables are discrete, i. e., the *conditional*

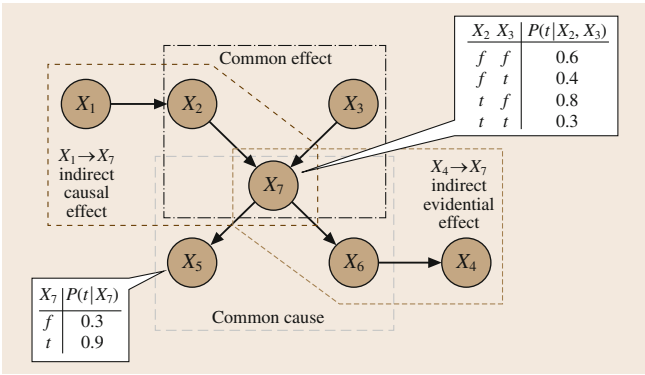


Fig. 31.1 An example of Bayesian network. Conditional probability tables are shown only for variables X_5 and X_7. Different types of probabilistic influence among variables are highlighted

probability distribution table (CPD table). The CPD of X_7 in Fig. 31.1, for instance, reports the probability of $X_7 = t$, given each possible assignment of values to its parents. The CPD table associated with X_5 is reported as well. By using the CPD tables associated to all nodes, the joint distribution can be rewritten as

$$P(X_1, \ldots, X_7) = P(X_1)P(X_3)P(X_2|X_1)P(X_7|X_2, X_3) \times P(X_5|X_7)P(X_6|X_7)P(X_4|X_6) \, .$$

Note that different distributions can be obtained by using different values for the entries of the CPD tables. Thus, a Bayesian network is actually representing a family of distributions: all the distributions that are consistent with the conditional independence assumptions used to simplify the factors. In fact, up to now, we have discussed how starting from a *universal* decomposition of the joint distribution via the product rule (note that such decomposition is not unique as it depends on the presentation order assigned to the variables), a set of conditional independence assumptions can be used to simplify the factors, leading to the corresponding graphical representation given by the Bayesian network. An important question, however, is whether the topological structure of a Bayesian network allows for the direct identification of other (conditional) independence relationships, i. e., whether there exist other (conditional) independence relationships that *must* hold for *any* joint distribution P that is compatible with the structure of a specific Bayesian network (note that additional relationships may hold only for *some* specific distributions, i. e., some specific assignment of values to the entries of the CDP tables). As we will see later, the answer to this question is important to devise general-purpose inference algorithms on Bayesian networks. A general procedure, called *d-separation* (directed separation), can answer the question. It is based on the observation that two variables are not independent if one can influence the other via one or more paths in the graph. Let us exemplify this concept on the Bayesian network reported in Fig. 31.1, where we have highlighted four different basic cases:

1. *Indirect causal effect*: X_1 can influence X_7 via X_2 if and only if X_2 is not observed (a variable is said to be observed if the value assigned to that variable is known).
2. *Indirect evidential effect*: X_4 can influence X_7 via X_6 if and only if X_6 is not observed.
3. *Common cause*: X_5 can influence X_6 (and viceversa) via X_7 if and only if X_7 is not observed.
4. *Common effect*: X_2 can influence X_3 (and viceversa) if and only if either X_7 or one of X_7's descendants (in this case, X_5, X_6, X_4) is observed.

The topological structure encountered in the *common effect* is called *v-structure* and it plays a relevant role in the *d-separation* procedure. In general, it is clear from above that probabilistic influence does not follow edge direction. Thus, when considering a longer trail, e.g., the path from X_1 to X_4, we have to consider whether each part of the trail allows probabilistic influence to flow or not (according to the four basic cases described above).

Definition 31.2 Active Trail

Let $X_1, \ldots, X_k$ be a trail in a Bayesian network $\mathbf{G}$, and $\mathcal{E}$ be a subset of observed variables in $\mathbf{G}$. The trail $X_1, \ldots, X_k$ is active given $\mathcal{E}$ if:

- Whenever a *v*-structure $X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$ does occur, X_i or one of its descendants belong to $\mathcal{E}$;
- No other node along the trail belongs to $\mathcal{E}$.

Of course, by definition, if $X_1 \in \mathcal{E}$ or $X_n \in \mathcal{E}$ the trail is not active. Examples of active/not active trails from the Bayesian network represented in Fig. 31.1 are: the trail X_1, X_2, X_7, X_6, X_4 is active given the set $\mathcal{E} = \{X_3, X_5\}$, while it is not active whenever either X_2 or X_7 or X_6 belongs to $\mathcal{E}$; on the other hand, the trail X_1, X_2, X_7, X_3 is active if $X_2 \notin \mathcal{E}$ and either X_7 or X_5 or X_6 or X_4 belongs to $\mathcal{E}$.

The Bayesian network represented in Fig. 31.1 does not allow more than one trail between any couple of nodes. In general, however, two nodes may have several trails connecting them and one node can influence the other one as long as there exist at least one active trail among them. This intuition is captured by the definition of the concept of *d-separation*.

Definition 31.3 d-Separation

Let $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{Z}$ be nonintersecting sets of nodes of a Bayesian network. $\mathcal{X}$ and $\mathcal{Y}$ are *d*-separated given $\mathcal{Z}$ if there is no active trail between any node $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$ given $\mathcal{Z}$.

The *d*-separation test can be used to precisely characterize the independence relationships which hold for probabilistic distributions that factorize according to the given Bayesian network.

In the following, we introduce another class of graphical models, i. e., Markov networks, which are described by undirected graphs.

31.2.2 Markov Networks

Directed edges in Bayesian networks are suited to describe causal relationships between random variables. In many cases, however, the probabilistic interaction between two variables is not directional. In these cases, it is natural to consider *undirected graphs*, i. e., *Markov networks*. An undirected edge between variables X and Y represents a probabilistic constraint between the two variables. On the other hand, if X and Y are not connected, then we can state a conditional independence assertion involving them if and only if there are no active trails connecting them in the graph. Note that, since edges are now undirected, a trail is not active if and only if any of the variables in the trail is observed. This leads us to discuss which kind of joint distribution factorization a Markov network does represent.

If we go back to the concept of active trail, it is clear that if we consider a subset S of fully connected nodes in the undirected graph, i. e., nodes in S are connected to each other, then any $X, Y \in S$ will be connected by so many trails involving nodes in $S \setminus \{X, Y\}$ that it is wise to consider a single factor ϕ_S involving all nodes in S. Technically, S is called a *clique*, and we are actually interested in *maximal cliques*, i. e., cliques which cannot be extended in size by considering another node of the graph. For example, the maximal cliques of the Markov network given in Fig. 31.2 are

$$c_1 = \{X_1, X_3, X_5\}\,, c_2 = \{X_1, X_2\}\,,$$
$$c_3 = \{X_2, X_4\}\,, c_4 = \{X_3, X_4\}\,.$$

Note that, while $\{X_1, X_5\}$ is a clique, it is not maximal since we can add X_3 obtaining a larger clique.

A different factor can be associated with each maximal clique c_i. By using a global normalization constant for the joint distribution factorization, a factor associated with a clique c_i can be modeled by a *potential function* $\phi_{c_i}(\cdot)$, i. e., any nonnegative function (see Fig. 31.2 for involving Boolean variables). Thus, the factorization of the joint distribution for the example in Fig. 31.2 is

$$P(X_1, X_2, X_3, X_4, X_5) = \frac{1}{Z}\phi_{c_1}(X_1, X_3, X_5)\phi_{c_2}(X_1, X_2) \times \phi_{c_3}(X_2, X_4)\phi_{c_4}(X_3, X_4)\,,$$

where the normalization constant

$$Z = \sum_{\forall i, x_i \in X_i} \phi_{c_1}(X_1, X_3, X_5)\phi_{c_2}(X_1, X_2) \times \phi_{c_3}(X_2, X_4)\phi_{c_4}(X_3, X_4)$$

is called the *partition function*. If with $\mathbf{x}$ we denote an assignment of values to the variables $X_1, \ldots, X_n$ and with $\mathbf{x}_{c_i}$ the corresponding assignments associated with variables in the clique c_i, the general formulas for a Markov network are

$$P(X_1, \ldots, X_n) = \frac{1}{Z}\prod_{\forall i, c_i} \phi_{c_i}(\mathbf{x}_{c_i})\,,$$

where

$$Z = \sum_{\mathbf{x}} \prod_{\forall i, c_i} \phi_{c_i}(\mathbf{x}_{c_i})\,.$$

If the potential functions are restricted to be strictly positive, then it is possible to find a precise correspondence between factorization and conditional independence. In fact, if we consider the set of all possible distributions defined over variables of a given Markov network, then the set of such distributions that are consistent with the conditional independence statements that can be derived by using the adapted concept of active trails and d-separation coincides with the set of distributions that can be expressed as a factorization of the form given above with respect to maximal cliques of the network (*Hammersley–Clifford* theorem).

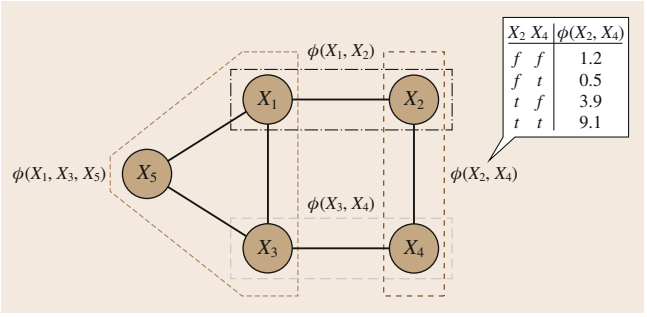


Fig. 31.2 An example of Markov network involving five variables. *Maximal cliques* and corresponding *potential functions* are highlighted. An example of potential function is given for clique $\{X_2, X_4\}$, where we have assumed that X_2 and X_4 are Boolean variables

For practical reasons, it is convenient to express a strictly positive potential function as a *Boltzmann distribution*, i. e.,

$$\phi_{c_i}(\mathbf{x}_{c_i}) = \mathrm{e}^{-E(\mathbf{x}_{c_i})} ,$$

where $E(\mathbf{x}_{c_i})$ is called an *energy function*. Since the joint distribution is the product of potentials, the total energy is obtained by adding the energy functions of each of the maximal cliques. Energy functions are very useful since, in the absence of a specific probabilistic interpretation for the potential functions, assignments of values that have high probability can be given low energies, while less probable assignments will correspond to high energies.

Let us give an example of application of Markov networks: image de-noising. The task is to remove noise from a binary image $\mathcal{Y}$ where the pixels Y_i are -1 or $+1$. Each observed pixel Y_i is obtained by a noise-free image $\mathcal{X}$ with pixels X_i where, with some small probability, the sign of the pixel is flipped. Since neighboring pixels in the noise-free image are strongly correlated, as well as the two variables Y_i and X_i, due to the small flipping probability, we can use a Markov network like the one depicted in Fig. 31.3 to capture this knowledge. The total energy function encoding such prior knowledge would be

$$E(\mathcal{X}, \mathcal{Y}) = -\beta \sum_{X_i, X_j \in \mathcal{X}} X_i X_j - \eta \sum_{\substack{X_i \in \mathcal{X} \\ Y_i \in \mathcal{Y}}} X_i Y_i ,$$

where all the maximal cliques are considered and couples of pixels with the same sign get lower energy values. Since we are interested in removing noise from the observed pixels Y_i, we add a bias toward pixel values that have one particular sign, by summing a term hX_i to the energy function for each pixel in the noise-free image

$$E(\mathcal{X}, \mathcal{Y}) = h \sum_{X_i \in \mathcal{X}} X_i - \beta \sum_{X_i, X_j \in \mathcal{X}} X_i X_j - \eta \sum_{\substack{X_i \in \mathcal{X} \\ Y_i \in \mathcal{Y}}} X_i Y_i .$$

Note that his operation is legal since it corresponds to multiplying the potential function, which are arbitrary nonnegative functions, by a nonnegative function.

The factorized joint distribution over $\mathcal{Y}$ and $\mathcal{X}$ is then defined as

$$P(\mathcal{X}, \mathcal{Y}) = \frac{1}{Z} \mathrm{e}^{-E(\mathcal{X}, \mathcal{Y})} .$$

Probabilistic inference can now be performed by clamping the value of $\mathcal{Y}$ to the observed image, which implicitly corresponds to a conditional distribution $P(\mathcal{X}|\mathcal{Y})$ over free images, and by computing the assignments to $\mathcal{X}$ that minimizes the total energy of the Markov model, i. e., the assignment of values to pixels of $\mathcal{X}$ with highest probability given the observed image $\mathcal{Y}$. The resulting assignment of values to $\mathcal{X}$ will return the (presumed) noise-free version of $\mathcal{Y}$.

In the following, we briefly present different approaches to perform probabilistic inference in Bayesian and Markov networks.

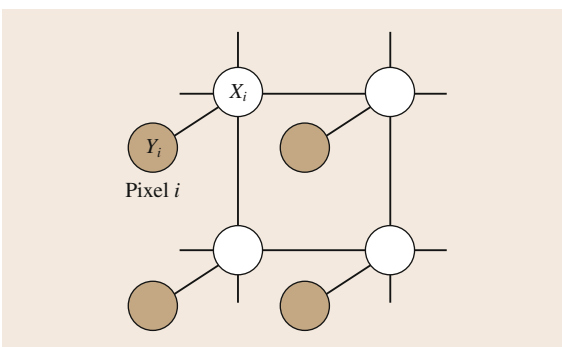


Fig. 31.3 A Markov network for image de-noising. Y_i is the binary variable representing the state of pixel i in the noisy observed image, while X_i refers to the noise-free image

31.2.3 Inference

Performing probabilistic inference in a graphical model over a set of random variables $\mathcal{X}$ means being able to answer any probabilistic query involving $\mathcal{X}$. Since a graphical model, either a Bayesian or a Markov network, describes a factorization of the joint distribution, any probabilistic query can be answered, so the problem reduces to find efficient procedures to perform inference. In the following, we report some of the most typical form of queries:

- *Conditional*: In this case, we are interested in computing $P(\mathcal{Y}|\mathcal{E} = \mathbf{e})$, where $\mathcal{Y}, \mathcal{E} \subset \mathcal{X}$, with $\mathcal{Y} \cap \mathcal{E} = \emptyset$, where $\mathcal{Y}$ are the *query* variables and $\mathcal{E} = \{E_1, \ldots, E_k\}$ are the *evidence* variables for which specific values $\mathbf{e} = \{e_1, \ldots, e_k\}$ have been observed.
- *Most probable assignment*: Given evidence $\mathcal{E} = \mathbf{e}$, we are interested in computing the most likely assignment $\mathbf{y}^*$ to $\mathcal{Y} \subseteq \mathcal{X} \setminus \mathcal{E}$. There are two main variants for this kind of query: *most probable explanation* (MPE) and *maximum a posteriori* (MAP).

A MPE query must solve the problem

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} P(\mathcal{Y} = \mathbf{y}, \mathcal{E} = \mathbf{e}) ,$$

where $\mathcal{Y} = \mathcal{X} \setminus \mathcal{E}$, while a MAP query must solve the problem

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \sum_{\mathbf{z}} P(\mathcal{Y} = \mathbf{y}, \mathcal{Z} = \mathbf{z} | \mathcal{E} = \mathbf{e}) ,$$

where $\mathcal{Z} = \mathcal{X} \setminus \mathcal{E} \setminus \mathcal{Y}$.

From the point of view of inference, both directed and undirected networks can be treated in the same way. In fact, directed networks can be converted to undirected networks. This is done by observing that factors in directed networks can be understood as factors corresponding to cliques in an undirected graph obtained by mutually connecting all the parents of each node by new undirected edges and by dropping direction from the original directed edges. This procedure is known as *moralization* and the resulting undirected graph is the *moral* graph. By this means, all the variables involved in factors of the directed graph (e.g., CPTs) will be contained in corresponding cliques of the moral graph. Thus, we can focus on undirected graphs.

From a computational point of view, in the worst case, probabilistic inference is difficult: every type of probabilistic inference in graphical models is $\mathcal{NP}$-hard or harder. Specifically, the complexity of inference is related to a topological property of the graphical network called *treewidth*. Approximate inference methods have been devised to deal with such computational complexity. Unfortunately, approximate inference turns out to be hard, in the worst case. Nevertheless, if the *treewidth* of the graphical network is not too large (e.g., in polytrees), exact inference can be performed in a reasonable amount of time. Moreover, in many practical cases, approximate inference is efficient and adequate.

There are three major approaches to perform inference: *exact* algorithms, *sampling* algorithms, and *variational* algorithms. The former tries to compute the exact probabilities while avoiding repeated computations. The second approach aims to efficiently approximate probabilities by sampling, in a smart way, the universe of events. Finally, the third approach allows us to treat both exact and approximate inference within the same conceptual framework. In the following, we briefly sketch the main ideas underpinning these approaches.

Exact Algorithms

Let us illustrate one of the basic ideas of exact algorithms, i. e., *variable elimination*, by using the Markov network shown in Fig. 31.2, where we assume all variables to be Boolean. Suppose we are interested in computing the marginal probability $P(X_2)$. We can get it by summing the factorized joint distribution over the remaining variables

$$P(X_2) = \sum_{\mathbf{x}_1} \sum_{\mathbf{x}_3} \sum_{\mathbf{x}_4} \sum_{\mathbf{x}_5} \frac{1}{Z} \phi(X_1, X_3, X_5) \times \phi(X_1, X_2) \phi(X_2, X_4) \phi(X_3, X_4) .$$

Naïve computation of the above equation would require $O(2^5)$ operations, since each summand involves five Boolean variables. However, we can rearrange the summands in a smarter way

$$\begin{aligned} P(X_2) &= \frac{1}{Z} \sum_{\mathbf{x}_1} \phi(X_1, X_2) \sum_{\mathbf{x}_4} \phi(X_2, X_4) \sum_{\mathbf{x}_3} \phi(X_3, X_4) \\ &\quad \times \sum_{\mathbf{x}_5} \phi(X_1, X_3, X_5) \\ &= \frac{1}{Z} \sum_{\mathbf{x}_1} \phi(X_1, X_2) \sum_{\mathbf{x}_4} \phi(X_2, X_4) \\ &\quad \times \sum_{\mathbf{x}_3} \phi(X_3, X_4) m_5(X_1, X_3) \\ &= \frac{1}{Z} \sum_{\mathbf{x}_1} \phi(X_1, X_2) \sum_{\mathbf{x}_4} \phi(X_2, X_4) m_3(X_1, X_4) \\ &= \frac{1}{Z} \sum_{\mathbf{x}_1} \phi(X_1, X_2) m_4(X_1, X_2) \\ &= \frac{1}{Z} m_1(X_2) , \end{aligned}$$

where the m_i terms are the intermediate factors obtained by summation on variable X_i. Note that Z can be computed by summing on variable X_2. Moreover, the total computational complexity reduces to $O(2^3)$ since no more than three variables occur together in any summand. In general, the maximal number of variables that occur in any summand is determined by the elimination order. Since many different elimination orders may be used, the lowest complexity is obtained by the order that minimizes this maximal number, which is related to the *treewidth* of the graph. Unfortunately, finding the optimal elimination order is $\mathcal{NP}$-hard.

One positive aspect of the elimination approach is that it also works for continuous variables since

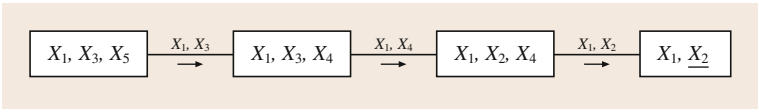
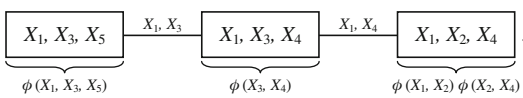
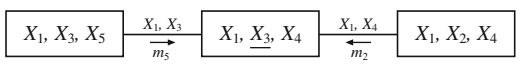


Fig. 31.4 Example of cluster graph, where the direction of the flow of computation is shown under each edge, while the scope of the computed factor transmitted to the other node after variable elimination is shown over each edge

it is only based on the topology of the graph. However, the elimination procedure returns only a single marginal probability, while it is often of interest to compute more than one marginal probability. Luckily, we can generalize the idea to efficiently compute all the single marginals. Here we give some hints on how to do it. Consider the sequence of intermediate factors generated in the example above. They can be indexed by the variables in their scope, i. e., $\psi_{1,3,5} = \phi(X_1, X_3, X_5), \psi_{1,3,4} = \phi(X_3, X_4)m_5(X_1, X_3), \psi_{1,2,4} = \phi(X_2, X_4)m_3(X_1, X_4), \psi_{1,2} = \phi(X_1, X_2)m_4(X_1, X_2)$. Graphically, we can represent them via a *cluster graph*, where each node is associated with a subset of variables (i. e., the scope of intermediate factors) and the undirected edges *support* the flow of computation of the elimination process. In our example, the cluster graph is shown in Fig. 31.4, where we have shown the direction of the flow of computation under each edge, and the scope of the computed factor *transmitted* to the other node after variable elimination over each edge. The variable X_2 in the rightmost node is underlined to remark that it is the *target* of the flow of computation. In general, since each edge is associated with a variable elimination, it is not difficult to realize that the cluster graph is in fact a tree (called *clique tree* or *junction tree*). This structure can also be used for computing other marginals. In order to see that, we have to observe that the scope of the rightmost node is a subset of the scope of the node at its left, so it can be merged with this last node; moreover, each *initial potential* must be associated with a node with consistent scope, e.g.,



Now, suppose we want to compute $P(X_3)$ by eliminating all the other variables. We have to select a node which contains X_3 in its scope, e.g., the middle node. The flow of computation should now converge toward that node, as shown in



Any elimination order consistent with the above flow will do the work, e.g., we first consider the leftmost node and eliminate X_5 by transmitting the *message*

$$m_5(X_1, X_3) = \sum_{\mathbf{x}_5} \phi(X_1, X_3, X_5)$$

to the middle node. Then, we do the same for the rightmost node, by eliminating X_2 and transmitting the message

$$m_2(X_1, X_4) = \sum_{\mathbf{x}_2} \phi(X_1, X_2)\phi(X_2, X_4) \, .$$

Finally, the middle node can merge the two received messages with the local potential obtaining

$$\phi(X_3, X_4)m_5(X_1, X_3)m_2(X_1, X_4) \, ,$$

which is an unnormalized version of the joint distribution $P(X_1, X_3, X_4)$. Marginal $P(X_3)$ can then be computed by summing out X_1 and X_4 and normalizing the result. Note that the same flow can be used to compute $P(X_1)$ and $P(X_4)$: in the first case, the final stage will sum out X_3 and X_4, while in the second case it will sum out X_1 and X_3.

In general, *all* the factors needed by *all* the nodes to compute the marginals of the variables in their scope, can be computed by a *sum-product message passing* scheme where, having selected an arbitrary node as *root*, messages are transmitted from the leaves up to the root and then back from the root to the leaves. If evidence is present, *restricted* potentials (i. e., potentials where evidence variables are bound to the observed values) are used. MEP and MAP queries can be answered by using a *max-sum algorithm*, which is a variation of the sum-product algorithm exploiting a *trellis* over all the values the variables can take. The message passing scheme sketched above can also be implemented using *division*, giving raise to the *Belief Update* algorithm.

Sampling Algorithms

The strategy adopted by sampling algorithms to perform (approximate) inference is to approximate the joint distribution via estimates computed on a set of

representative instantiations of all, or some of, the variables of the graphical model. Unlike exact inference, some techniques are specialized for directed networks. For example, a simple approach to estimate the joint probability in a Bayesian network is *Forward Sampling*. It starts by considering any topological ordering of the variables, e.g., for the network in Fig. 31.1 the order $X_1, X_3, X_2, X_7, X_5, X_6, X_4$ will do the job. Then random samples are generated by following the order and by picking a value for each variable according to its distribution. Note that variables with conditional distributions will be considered only when specific values for their parents have already been generated, so that the conditional probability for those variables is fully specified. Once M full samples are generated in this way, the probability of a specific event $P(\mathcal{E} = \mathbf{e})$ is estimated as the fraction of samples where variables in $\mathcal{E}$ take values $\mathbf{e}$. If the query is of the form $P(\mathcal{Y}|\mathcal{E} = \mathbf{e})$, samples which are not consistent with the evidence are rejected (*rejection sampling*) and the remaining samples used to estimate the conditional distribution on variables $\mathcal{Y}$. With this approach, however, a large amount of generated samples are discarded.

An improvement on this aspect is given by the *likelihood weighting* algorithm, which is based on the observation that evidence variables can be forced to assume *only* the observed values in a sample as long as the sample is weighted by the likelihood of the evidence. This means that a weight is associated with each sample and the weight is given by the product of all the posterior probabilities corresponding to the observed values for the evidence variables, i. e.,

$$w_{\text{sample}} = \prod_{E_i \in \mathcal{E}} P(E_i = e_i | \text{pa}(E_i)) \, .$$

Estimates are then computed considering weighted samples. Likelihood weighting turns out to be a special case of a more general approach called *importance sampling* which aims at estimating the expectation of a function relative to some distribution.

Improved sampling methods, which can also be applied to Markov networks, are given by *Markov chain Monte Carlo* methods. Unlike the methods described so far, these methods generate a *sequence* of samples, in such a way that later samples are generated by distributions that provably approximate with increasing precision the target posterior probability (i. e., the query $P(\mathcal{Y}|\mathcal{E} = \mathbf{e})$).

The simpler method uses *Gibbs sampling*: an initial assignment of values for the unobserved variables is generated from an initial distribution; subsequently, in turn, each unobserved variable is sampled using the posterior probability *given* the current sample for all other variables. This distribution can be computed efficiently by using only factors associated with the *Markov blanket*, i. e., the neighbors of the variable to be resampled in the Markov network (in Bayesian networks, the Markov blanket of a node is given by the set of its parents, its children and the parents of its children). Using the theory of *Markov chains* (discussed in Sect. 31.4.1), it is possible to show that, under some assumptions, the sequence of generated distributions converges to a stationary distribution, where the fraction of time in which a specific assignment of values to variables (sample) does occur in the sequence is exactly proportional to the posterior probability of that assignment.

A drawback of Gibbs sampling is that it uses only *local moves* (i. e., resampling of a single variable), leading to very slow convergence for assignments with low probability. More effective methods, based on the *Metropolis–Hastings* approach, enable for a broader range of moves. Further, more advanced approaches allow us to consider partial assignments in conjunction with a closed-form distribution for unassigned variables. Others use deterministic methods to explicitly search for high-probability assignments to approximate the joint distribution.

Variational Algorithms

Probabilistic inference can be formulated as a constrained optimization problem. This allows both to rediscover exact inference algorithms, such as the ones we have briefly discussed above, and to design approximated inference algorithms, by simplifying either the objective function to optimize and/or the admissible region for optimization. The possibility to devise theoretically founded approximation algorithms is particularly appealing in cases where the joint distribution is characterized by a factorization with associated large treewidth. Research in this area has been recently very active, yielding to several interesting results. Here we do not have the space for a proper technical treatment, so we try to give only a brief introduction to the main ideas.

Variational approaches are based on the idea of approximating an intractable probabilistic distribution with a simpler one, which allows for inference. This simpler distribution is selected from a family of tractable distributions, as the distribution that is the *best* approximation to the desired one. Can we define a mea-

sure of the quality of the approximation that can be used for the minimization process? A good measure is the KL-divergence introduced in (31.2). Let us denote a distribution that factorizes according to the graphical model $\mathcal{G}$ as

$$P_{\mathcal{G}}(\mathcal{X}) = \frac{1}{Z} \prod_{\forall i, c_i} \phi_{c_i}(\mathbf{x}_{c_i}) \tag{31.9}$$

and let $Q(\mathcal{X})$ be a member of the tractable distributions we use to approximate $P_{\mathcal{G}}(\mathcal{X})$. Then, a nice feature of KL-divergence is that it allows us to efficiently solve the optimization problem

$$\arg\min_{Q(\mathcal{X})} D_{\mathrm{KL}}(Q(\mathcal{X}) \| P_{\mathcal{G}}(\mathcal{X}))$$

without requiring to perform inference in $P_{\mathcal{G}}(\mathcal{X})$. In fact, using the factorization of $P_{\mathcal{G}}(\mathcal{X})$ in (31.9), it is not difficult to show that

$$D_{\mathrm{KL}}(Q(\mathcal{X}) \| P_{\mathcal{G}}(\mathcal{X})) = \log Z - \sum_{\forall i, c_i} \mathbb{E}_Q[\log \phi_{c_i}] + \mathbb{E}_Q[\log Q(\mathcal{X})] \,, \tag{31.10}$$

and, since $\log Z$ does not depend on $Q(\mathcal{X})$, minimizing $D_{\mathrm{KL}}(Q(\mathcal{X}) \| P_{\mathcal{G}}(\mathcal{X}))$ is equivalent to maximizing the *energy functional* term

$$\sum_{\forall i, c_i} \mathbb{E}_Q[\log \phi_{c_i}] - \mathbb{E}_Q[\log Q(\mathcal{X})] \,.$$

Following from the definition in (31.1), $H_Q(\mathcal{X}) = -\mathbb{E}_Q[\log Q(\mathcal{X})]$ is the entropy of Q, while the first term in (31.10) is referred to as *energy term*.

Different variational methods correspond to different strategies for optimizing the energy functional. The name *variational* is used since all of them adopt the general strategy of reformulating the optimization problem by introducing new variational parameters to be used for optimization. In particular, each specific choice of values for the variational parameters expresses one member, i. e., $Q(\mathcal{X})$, of the family of tractable distributions we want to use. The optimization procedure searches the space of variational parameters to find the $Q^*(\mathcal{X})$ that best approximates $P_{\mathcal{G}}(\mathcal{X})$. It is important to understand that the family of tractable distributions will actually corresponds to a set of constraints, involving the variational parameters that must be satisfied while maximizing the energy functional. By using *Lagrange multipliers* these constraints can be merged together with the energy functional, giving rise to a *Lagrangian* function that must be maximized. By taking the partial derivatives with respect to the variational parameters and the Lagrange multipliers, the solution to the optimization problem can be characterized by a set of *fixed-point equations*. These equations can then be used to straightforwardly devise an iterative solution.

Different variational methods work with different types of approximations. There are two main sources of approximation, which can be used singularly or in conjunction. One source is the energy functional, which can be substituted by a functional easy to manipulate while preserving a good degree of approximation. Another source of approximation are the constraints, i. e., the definition of the family of tractable distributions, which may not be fully consistent with the factorization represented by the graphical model (in this case, denoted as pseudo-distributions).

We do not have space here to give more details; however, it is worth to mention that while convergence proofs of several variational methods are available, it is not so common to find theoretical guarantees on the approximation error made by the specific method.

31.3 Latent Variable Models

Knowledge hindered in the complex relation between a large number of observable variables can be surfaced under the assumption that a simpler and unobservable process exists, which is responsible for generating the complex behavior of manifest data. Such an unobservable generative process can be modeled through the use of *latent variables*, as opposed to observable variables, that are not directly measurable, but can be inferred from observations and can explain the relation between manifest data. Intuitively, latent variables can be understood as an attempt to model the unknown physical process generating the observations or as an abstraction providing a simplified representation of the manifest data, e.g., clusters.

Probabilistic models that attempt to explain observations in terms of latent variables are called *latent variable models*. In probabilistic terms, the simplification introduced by latent variables results in conditional independence assumptions, such that (subsets of) observable variables can be considered conditionally

independent when their hidden explanation, i. e., the latent variable assignment, is given. Similarly to observed variables, latent variables can be discrete or continuous: their nature, together with that of the observations, determines different types of probabilistic models. Nevertheless, parameter estimation in the different latent variable models can be achieved through a general iterative principle, known as *expectation–maximization*.

31.3.1 Latent Space Representation

To understand the intuition at the basis of latent space representation, consider a joint distribution $P(\mathcal{X}) = P(X_1, \ldots, X_N)$ defined over N joint observed random variables X_i. As discussed in Sect. 31.2.1, without any simplifying assumption, the number of free parameters of this simple model grows as $O(2^{N-1})$ for Boolean variables, which quickly becomes unmanageable for large N. One way to control the number of free parameters of a model, without taking too simplistic assumptions (e.g., X_i being i.i.d.), is to introduce a collection of *latent*, or hidden, variables $\mathcal{Z} = \{Z_1, \ldots, Z_K\}$. The latent variables are unobserved but can be used to factorize the joint distribution $P(\mathcal{X})$ while allowing to capture (some of) the correlations between the $\mathcal{X} = \{X_1, \ldots, X_N\}$ observed variables. More formally, latent variables are such that

$$P(\mathcal{X}) = \int_{\mathbf{z}} P(\mathcal{X}|\mathcal{Z} = \mathbf{z}) P(\mathcal{Z} = \mathbf{z}) \mathrm{d}\mathbf{z} \,, \tag{31.11}$$

that is the general formulation for the likelihood of a *latent variable model*. The details of the latent variable model, and the tractability of the integral in (31.11), are determined by the form of the conditional distribution $P(\mathcal{X}|\mathcal{Z})$ and by the marginal probability $P(\mathcal{Z})$. A common approach in latent variable models is to assume that observed variables become conditionally independent given the latent variables, that is

$$P(\mathcal{X}) = \int_{\mathbf{z}} \prod_{i=1}^{N} P(X_i|\mathcal{Z} = \mathbf{z}) P(\mathcal{Z} = \mathbf{z}) \mathrm{d}\mathbf{z} \,. \tag{31.12}$$

A basic assumption for this latent model to be effective, is that the conditional and marginal distributions should be more *tractable* than the joint distribution $P(\mathcal{X})$. For instance, in a simple scenario with discrete observations and latent variables, this entails that $K \ll N$. Not surprisingly, the same intuition is applied, in a deterministic context, for dimensionality reduction (cf. the number of projection directions in PCA) and clustering.

Different types of latent variable models are defined based on the nature of the latent and observed variables, as well as depending on the form of the conditional and marginal probabilities. In the following, we discuss two general classes of latent variable models with continuous and discrete hidden variables, which are *factor analysis* and *mixture models*, respectively.

31.3.2 Learning with Latent Variables: The Expectation–Maximization Algorithm

Learning, in a probabilistic setting, entails working with the model likelihood. In latent variable models, the likelihood in (31.11) might be difficult to treat due to the marginalization inside the logarithm, which can potentially couple all the model parameters. Despite the diversity of the models that can be designed, based on the general expression in (31.11), there exist a general principle to estimate their parameters.

The *expectation–maximization* (EM) algorithm [31.48] is a general iterative method for the maximization of the likelihood under latent variables. The key intuition of the EM algorithm is to define an alternative objective function where the parameter coupling introduced by the marginalization of the hidden variables is removed. The EM algorithm maximizes the marginal data likelihood $P(\mathcal{X}|\theta)$, where θ are the model parameters, through a tractable lower bound defined by introducing a function of the latent variables, i. e., $Q(\mathcal{Z})$, into the data likelihood through marginalization. For notational simplicity, consider the case of discrete latent variables. For any nonzero distribution $Q(\mathcal{Z})$, it holds

$$\begin{aligned}
\mathcal{L}(\theta) &= \log P(\mathcal{X}|\theta) \\
&= \log \sum_{\mathbf{z}} P(\mathcal{X}, \mathcal{Z} = \mathbf{z}|\theta) \\
&= \log \sum_{\mathbf{z}} Q(\mathbf{z}) \frac{P(\mathcal{X}, \mathcal{Z} = \mathbf{z}|\theta)}{Q(\mathbf{z})} \\
&\geq \sum_{\mathbf{z}} Q(\mathbf{z}) \log P(\mathcal{X}, \mathcal{Z}|\theta) \\
&\quad - \sum_{\mathbf{z}} Q(\mathbf{z}) \log Q(\mathbf{z}) = \tilde{\mathcal{L}}(Q, \theta) \,,
\end{aligned} \tag{31.13}$$

where the lower bound $\tilde{\mathcal{L}}(Q, \theta) \leq \mathcal{L}(\theta)$ is obtained by the application of the Jensen inequality to the concave log function. The joint distribution $P(\mathcal{X}, \mathcal{Z}|\theta)$ is known as the *complete data likelihood*, where the term

complete refers to the fact that the marginal data likelihood $P(\mathcal{X}|\theta)$ is *completed* with the observations $\mathbf{z}$ for the latent variables.

The *Expectation–maximization* algorithm defines an alternate optimization process where the bound $\tilde{\mathcal{L}}(Q,\theta)$ is maximized with respect to $Q(\cdot)$ and θ. In general, this is performed by two independent maximization steps that are repeated until convergence:

- Expectation (E) Step: For θ fixed, find the distribution $Q^{(t+1)}(\mathbf{z})$ that maximizes the bound $\tilde{\mathcal{L}}(Q,\theta^{(t)})$;
- Maximization (M) Step: Given the distribution $Q(\mathbf{z})^{(t+1)}$, estimate the model parameters $\theta^{(t+1)}$ that maximize the bound $\tilde{\mathcal{L}}(Q^{(t+1)},\theta)$;

where the superscript denotes the estimate at time t. Clearly, the optimal solution for the *E-step* is attained when

$$Q^{(t+1)}(\mathbf{z}) = P(\mathcal{Z}=\mathbf{z}|\mathcal{X},\theta^{(t)})\,, \tag{31.14}$$

that is when the lower bound in (31.13) becomes an equality. In practice, to explicitly evaluate the complete likelihood in $\tilde{\mathcal{L}}(Q,\theta^{(t)})$, we would need to observe the $\mathbf{z}$ assignments. These are unknown, since latent variables are unobservable. However, given the marginalization of $\mathbf{z}$ in (31.13), we can substitute the unavailable $\mathbf{z}$ observations with their expected values, by considering them as another random variable. To this end, it suffices that the E-step computes the expected value of the *complete log-likelihood* $\log P(\mathcal{X},\mathcal{Z}|\theta)$ with respect to $\mathcal{Z}$. These observations provide the final form of the *classical EM* algorithm:

- *E*-step: Given the current estimate of the model parameters $\theta^{(t)}$, compute

$$Q^{(t+1)}(\theta|\theta^{(t)}) = E_{\mathcal{Z}|\mathcal{X},\theta^{(t)}}[\log P(\mathcal{X},\mathcal{Z}|\theta)]\,; \tag{31.15}$$

- *M*-step: Find the new estimate of the model parameters

$$\theta^{(t+1)} = \arg\max_{\theta} Q^{(t+1)}(\theta|\theta^{(t)})\,. \tag{31.16}$$

In other words, the *E-step* estimates the value of the otherwise unobserved latent variables, while the *M-step* finds the parameters that maximize the current estimate of the log-likelihood. In practice, the *E-step* often reduces to estimating the expectation of $\mathcal{Z}$ as its posterior $P(\mathcal{Z}|\mathcal{X},\theta^{(t)})$, while the *M-step* uses these values as sufficient statistics to update the model parameters $\theta^{(t+1)}$. This alternate optimization is typically iterated until the log-likelihood does not change much between consecutive estimates, or when a number of maximum iterations is reached. Note that the two-step EM optimization process is prone to local optima. Hence, its convergence can be slow and, often, its solutions tend to be dependent on the initialization.

The EM algorithm assumes that we can calculate the expected value of the complete log-likelihood. However, there are cases in which the required summation is not computationally feasible (e.g., with infinite summations where the integral has no close-form solution): in this cases, the approximated inference methods described in Sect. 31.2.3 can be used to define nonexact EM algorithms. For instance, stochastic versions of the EM algorithm are obtained by approximating the infeasible summation using (e.g., Gibbs) sampling from the posterior distribution $P(\mathcal{Z}|\mathcal{X},\theta)$. The classical EM algorithm is a ML method providing point estimates of the model parameters θ. The *variational Bayes* (VB) [31.6] method has been introduced to obtain a fully Bayesian solution that returns a posterior distribution of the parameters $P(\theta)$, instead of their point estimate. VB is based on an analytical approximation of the joint posterior of the latent variables and model parameters that yields to a generalization of the EM alternate optimization, where the maximization at the *M-step* is taken over possible distributions $Q(\theta)$, instead of on θ itself.

31.3.3 Linear Factor Analysis

Factor analysis (FA) is an example of a latent variable model for continuous hidden and manifest variables. In its simplest linear form, it is a classical statistical model widely used for generative dimensionality reduction. Similarly to its deterministic counterparts, e.g., PCA, it forms a low-dimensional embedding of a set of observations $\mathcal{D} = (\mathbf{x}_1,\ldots,\mathbf{x}_\mathbf{n})$, where each observation $\mathbf{x}$ is a D-dimensional vector of reals. FA finds a lower dimensional probabilistic representation of $\mathcal{D}$, by assuming that the features of each $\mathbf{x}$ are independently generated by K real-valued latent variables $\mathcal{Z} = \{Z_1,\ldots,Z_K\}$, with $K \ll D$ (see the associated graphical model in Fig. 31.5).

The FA model, assumes that observations are linked to the latent vectors through a linear model

$$\mathbf{x} = F\mathbf{z} + b + \epsilon\,, \tag{31.17}$$

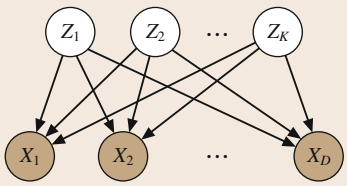


Fig. 31.5 Linear factor analysis: the observed D-dimensional variable $\mathcal{X}$ is related to the K latent variables $\mathcal{Z} = \{Z_1, \ldots, Z_K\}$ through a linear mapping

where $\epsilon \sim \mathcal{N}(\epsilon|0, \Psi)$ is the Gaussian distributed noise with zero mean and covariance Ψ, b is a bias vector and F is the factor loading matrix. The latent variables are the *factors* and are generally assumed to be distributed as $\mathcal{Z} \sim \mathcal{N}(\mathbf{z}|0, I_K) = P(\mathcal{Z})$, where I_K is the K-dimensional identity matrix. Under such Gaussian assumptions, and given the linear model in (31.17), the conditional distribution of the observations is

$$P(\mathcal{X} = \mathbf{x}|\mathcal{Z} = \mathbf{z}) = \mathcal{N}(\mathbf{x}|F\mathbf{z} + b, \Psi) \,, \tag{31.18}$$

which, inserted in (31.11), provides the distribution for the FA complete likelihood

$$P(\mathcal{X}) = \int_{\mathbf{z}} P(\mathcal{X}|\mathcal{Z})P(\mathcal{Z})d\mathbf{z} = \mathcal{N}(\mathbf{x}|b, FF^T + \Psi) \,. \tag{31.19}$$

The form of the noise covariance Ψ determines the type of FA model: in general, this is chosen as a diagonal matrix with a vector of $(\psi_1, \ldots, \psi_D)$ values on the main diagonal. When the diagonal elements are all equal to a single value $\sigma^2 \in \mathbb{R}$, the FA reduces to the special case of the *Probabilistic PCA* [31.49].

Learning of the FA parameters $\theta = (\Psi, F)$ (b is usually set a priori to the mean of the data) is obtained by maximum likelihood estimation. The most popular approach to obtain such estimates is based on solving an eigen-decomposition problem. Given the nature of FA as a latent variable model, its θ parameters can also be estimated by applying EM to the logarithm of the complete likelihood in (31.19). The latter approach is, however, less used in general, given its slower convergence.

31.3.4 Mixture Models

The term *mixture models* identifies a large family of latent variable models comprising discrete hidden variables and generic manifest variables. A mixture model assumes that each observation is generated by a weighted contribution of a number of simple distributions, selected by the hidden variables. The simplest form of mixture model assumes that an observation is independently generated by a single mixture component. Widely popular elements of this family are the *Gaussian mixture model* for continuous observations and the *mixture of unigrams* for multinomial data. In the following, we discuss an example of more articulated generative processes comprising observations with mixed component memberships.

Probabilistic Latent Semantic Analysis

Probabilistic latent semantic analysis (pLSA) [31.50] has been introduced to model mixed membership observations, where a manifest sample is allowed to be generated by multiple latent variables. Its primary application is on documental analysis, where latent variables are interpreted as topics to be identified in a collection of documents. Intuitively, in the mixture of unigrams, each document is assigned to a unique topic and, as a consequence, all the words in a document are constrained to belong to a single topic. The pLSA model relaxes this assumption by allowing words in a document to belong to different topics, obtaining a multitopic representation for the documents in the collection.

The typical pLSA setting includes a dataset of multinomial samples, which are the documents $\mathcal{D} = \{d_1, \ldots, d_N\}$. Each document is an L-dimensional vector of word counts of length equal to the size of the reference dictionary. In other words, the ith observed sample is a vector $d_i = (w_1^i, \ldots, w_L^i)$, where w_j^i is the number of occurrences of the jth word of the vocabulary in the ith document. This data is typically summarized in a rectangular $L \times N$ integer matrix n, such that each row $n(\cdot, d_i)$ contains the word counts for document d_i. The variables identifying words and documents, i. e., W_j and D_i, are observed, in contrast with the set of topics $\mathcal{Z} = \{Z_1, \ldots, Z_K\}$, which are the *latent variables*. In pLSA, every observation $n(w_j, d_i)$ is associated with a latent topic z_k by means of the hidden variable Z_k.

The fundamental probabilities associated with this model are $P(D = d_i)$, that is, the document probability, $P(W = w_j|Z = z_k)$, that is, the probability of word w_j conditioned on topic z_k, and $P(Z = z_k|D = d_i)$, that is the conditional probability of topic z_k given document d_i. Given the nature of the manifest and hidden variables, all probabilities involved in pLSA are multinomials. The pLSA defines a (quasi) generative model for the word/document co-occurrences whose gener-

ative process is described by Fig. 31.6, using *plate notation*. This is a concise representation for graphical models involving replications: rectangular plates denote replication of their content for a number of times given by term on the bottom right (e.g., N and L_d for the outer and inner plates in Fig. 31.6, respectively); each shaded circular item denotes an observed variable, while empty circles identify latent variables.

The conditional independence relationships in Fig. 31.6 allow us to factorize the joint word-topic distribution: by using the parent decomposition rule introduced in (31.8), it yields

$$P(W_j, D_i) = P(D_i)P(W_j|D_i) = P(D_i)\sum_{k=1}^{K} P(Z_k|D_i)P(W_j|Z_k)\,, \quad (31.20)$$

that is the specific pLSA form of the general latent topic factorization in (31.12). The second equality in (31.20) is given by the marginalization of the latent topics Z_k and by the conditional independence assumption of the pLSA model, stating that word w_j and document d_i can be considered independent given the state of the latent variable Z_k. In other words, the word distribution of a document is modeled as a convex combination of K topic-specific distributions $P(W_j|Z_k)$. Such decomposition has a well-known characterization in terms of *Nonnegative matrix factorization* [31.13].

Estimation of the pLSA parameters $\theta = \{P(W_j|Z_k), P(Z_k|D_i)\}$ is obtained by maximization of the log-likelihood

$$\mathcal{L}(\theta) = \log \prod_{i=1}^{D}\prod_{j=1}^{W} P(W_j, D_i)^{n(w_j,d_i)} = \sum_{i=1}^{D}\sum_{j=1}^{W} n(w_j, d_i) \times \log\left\{P(D_i)\sum_{k=1}^{K} P(Z_k|D_i)P(W_j|Z_k)\right\}, \quad (31.21)$$

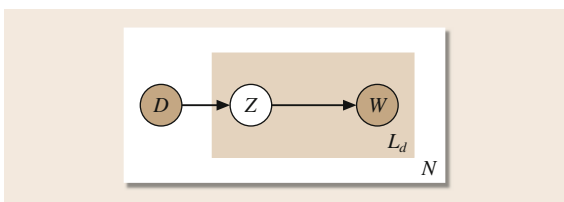


Fig. 31.6 Graphical model for the probabilistic latent semantic analysis: indices for the random variables D, Z, and W are omitted in the plate notation. The term L_d denotes replication for the L_d words present in the dth document

where $P(W_j, D_i)$ has been expanded using the formulation in (31.20). As with other latent topic models, this maximization problem can be solved through the iterative EM-algorithm discussed in Sect. 31.3.2. Following (31.15), the *E-step* computes the expectation of the complete likelihood $P(Z, W, D)$ with respect to the pLSA latent topics, assuming observed documents and words. It easily shows that the resulting E-step computes

$$P(Z_k|W_j, D_i) = \frac{P(Z_k|D_i)^{(t)}P(W_j|Z_k)^{(t)}}{\sum_{k'=1}^{K} P(Z_{k'}|D_i)^{(t)}P(W_j|Z_{k'})^{(t)}}\,, \quad (31.22)$$

that is the probability of the topic Z_k given word W_j in document D_i, estimated using the current values (at time t) of the model parameters $\theta^{(t)} = \{P(W_j|Z_k)^{(t)}, P(Z_k|D_i)^{(t)}\}$. Note that the decomposition on the right-hand side of Eq. (31.22) has been obtained by factorization of the posterior $P(Z_k|W_j, D_i)$ using the Bayes theorem.

The *M-step* equations (31.16) are obtained by differentiating the pLSA log-likelihood, extended with appropriate Lagrange multipliers for normalization, with respect to the $P(Z_k|D_i)$ and $P(W_j|Z_k)$ parameters. The resulting update equations are

$$P(Z_k|D_i)^{(t+1)} = \frac{\sum_{j=1}^{W} n(w_j, d_i)P(Z_k|W_j, D_i)}{\sum_{j=1}^{W} n(w_j, d_i)}\,, \quad (31.23)$$

$$P(W_j|Z_k)^{(t+1)} = \frac{\sum_{i=1}^{D} n(w_j, d_i)P(Z_k|W_j, D_i)}{\sum_{j=1}^{W}\sum_{i=1}^{D} n(w_j, d_i)P(Z_k|W_j, D_i)}\,. \quad (31.24)$$

The two-step optimization is iterated until a likelihood convergence criterion is met: often a validation set, or a *tempered* version of the EM are used in order to avoid model overfitting [31.50].

Advanced Topic Models

The pLSA was the first mixed membership model allowing a single observed sample to be generated by multiple latent topics at the same time. However, pLSA cannot be considered a fully generative model. In fact, the document-specific mixing weights for the topics are not sampled from a distribution, rather they are selected from $P(z_k|d_i)$ based on the index of document d_i. Hence, pLSA indexes only those

documents that are in the training set $\mathcal{D}$ and cannot directly model the generative process of unseen test documents. In other words, the pLSA is basically assigning null probabilities to all inputs that are not in the training set. The folding-in heuristic has been proposed to opportunistically solve this limitation, by assigning latent variables in the test-data to their MAP values before computing the test-set perplexity. However, the folding-in approach has been shown to lead to overly optimistic estimates of the test-set log-likelihood [31.51].

The *latent Dirichlet allocation* (LDA) [31.52] has been proposed as a Bayesian approach to address such modeling limitation of pLSA. It extends pLSA by treating the multinomial weights $P(Z|D)$ as additional latent random variables, sampling them from a Dirichlet distribution, that is the conjugate prior of a multinomial distribution. Using conjugate distribution eases inference as it ensures that the posterior distribution has the same form of the prior. The latent variable decomposition of the LDA log-likelihood is

$$P(W = w|\phi, \alpha, \beta) = \int \sum_z P(W = w|Z = z, \phi) \times P(Z = z|\theta)P(\theta|\alpha)P(\phi|\beta)\mathrm{d}\theta \,, \quad (31.25)$$

where $P(W|Z, \phi)$ is the multinomial word-topic distribution with parameters ϕ sampled from the Dirichlet distribution $P(\phi|\beta)$. The term $P(Z|\theta)$ is the topic distribution having θ as document-specific multinomial parameter being sampled from the Dirichlet $P(\theta|\alpha)$.

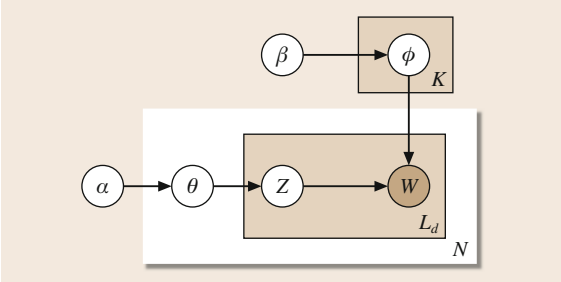


Fig. 31.7 Graphical model for the latent Dirichlet allocation

The terms α and β are the hyperparameters of the Dirichlet distribution, see Fig. 31.7 for the model plate notation. Direct EM inference is impossible for LDA, since the integral in (31.25) is intractable due to the couplings between the parameters within the topic marginalization. Again, approximate and stochastic Bayesian inference methods, such as those in Sect. 31.2.3, are used to fit the LDA parameters, including VB [31.52], expectation propagation [31.53], and Gibbs sampling [31.54].

The principles underlying pLSA and LDA have inspired the development of latent topic models that account for more articulated assumptions on the form of the hidden generative process. For instance, hierarchical LDA [31.55] proposes a generative process where observations are generated by a topic tree instead of being drawn from a flat topic collection. Further, specialized latent variable models have been developed for specific applications, such as author-topic analysis in scientific literature [31.56] and image understanding [31.57].

31.4 Markov Models

Time series and, more generally, sequences are a form of structured data that represents a list of observations for which a complete order can be defined, e.g., time in a temporal sequence. Let a sequence of length T be $\mathbf{y}_n = y_1, \ldots, y_T$, where the bold notation is used to denote the fact that $\mathbf{y}$ is a compound object (in practice, however, this is can be treated as a set of random variables). The term y_t is used to denote the tth observation with respect to the total order. Position t is often referred to as time when dealing with time-series data.

Two sequences are generally the results of independent trials, hence they can be considered i.i.d. samples. However, the elements composing a sequence fail to meet such i.i.d. property. Therefore, in principle, a probabilistic model for $\mathbf{y}$ would be required to specify the joint distribution $P(Y_1, \ldots, Y_T)$. For discrete valued observations y_t, the joint distribution grows exponentially with the size of the observation domain. Clearly, this would make the use of the probabilistic model fairly impractical due to the exponential size of the parameter space. To reduce such parameterization, *Markov chains* make the simplifying assumption that an observation occurring at some position t of the sequence, only depends on a limited number of its predecessors with respect to the complete order. In a time series, this entails that an observation at the present time, only depends on the history of a limited number of past observations. Markov chains allow us to model such

history dependence and are the heart of the *hidden Markov model* (HMM), which is the most popular approach to model the generative process of sequential data.

The HMM is a notable example of latent variable model: in the following, we provide an overview of the associated learning and inference problems. For simplicity, presentation focuses on sequences of finite length T and discrete time t. Sequence elements y_t can be either discrete valued or defined over reals, without major impact on the model. The section also discusses how the HMM causation assumption can be modified to give rise to alternative approaches, with interesting applications that overshoot simple sequence modeling.

31.4.1 Markov Chains

A Markov chain is a simple stochastic process for sequences. It assumes that an observation y_t at time (position) t only depends on a finite set of $L \geq 1$ predecessors in the sequence. The number of predecessor L influencing the new observation is the order of the Markov chain.

Definition 31.4 Markov Chain

An L-order Markov chain is a sequence of random variables $\mathbf{Y} = Y_1, \ldots, Y_T$ such that for every $t \in \{1, \ldots, T\}$, it holds

$$P(Y_t = y_t | Y_1, \ldots, Y_{t-1}, Y_{t+1}, \ldots, Y_T) = P(Y_t = y_t | Y_{t-L}, \ldots, Y_{t-1}) \,. \tag{31.26}$$

Following from the discussions in Sect. 31.2.1, (31.26) states that the L predecessors of Y_t define the set of its Bayesian parents $\text{pa}(Y_t) = \{Y_{t-L}, \ldots, Y_{t-1}\}$. For a first-order Markov chain, i. e., $L = 1$, (31.26) reduces to $P(Y_t = y_t | Y_{t-1} = y_{t-1})$. Such conditional independence assumption formally encodes the intuition that the current observation can be predicted from the sole knowledge of the preceding sample. The graphical model of a first-order Markov chain is shown in Fig. 31.8, whose joint distribution decomposes as

$$\begin{aligned} P(Y_1, \ldots, Y_T) &= P(Y_1)P(Y_2|Y_1), P(Y_3|Y_2) \\ &\quad \times \ldots P(Y_T|Y_{T-1}) \\ &= P(Y_1) \prod_{t=2}^{T} P(Y_t|Y_{t-1}) \,. \end{aligned} \tag{31.27}$$

The first element Y_1 has an empty conditioning part given that is has no predecessor. Its probability $P(Y_1)$

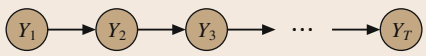


Fig. 31.8 Graphical model for a first-order Markov chain of length T, where $\text{pa}(Y_t) = \{Y_{t-1}\}$

is referred to as *marginal* or *prior* probability, while the term $P(Y_t|Y_{t-1})$ is the *transition* probability.

A Markov chain is *stationary* or *homogeneous*, if the transition probability does not depend on the time (position) t. In other words, the parameterization of the Markov chain is such that

$$P(Y_t = y' | Y_{t-1} = y) = f(y', y) \,,$$

where the transition distribution is a function $f(y', y)$ of the sole observations y, y'. An interesting stationary first-order Markov chain is that whose random variables take values from a finite alphabet of discrete symbols $i, j \in \{1, \ldots, M\}$. In these chains, the transition probability

$$A_{ij} = P(Y_t = i | Y_{t-1} = j) \tag{31.28}$$

denotes the probability of occurrence of the ith symbol preceded by symbol j. For convenience, such probability is represented by the element A_{ij} of the $M \times M$ transition matrix $A = [A_{ij}]_{i,j=1}^{M}$. Similarly, the marginal distribution defines the elements

$$\pi_i = P(Y_1 = i) \tag{31.29}$$

of the $M \times 1$ initial state vector $\pi = [\pi_i]_{i=1}^{M}$. These Markov chains can be straightforwardly interpreted as state-transition systems, where each symbol i of the alphabet is a state and a state-transition arrow exists between states i and j having a nonzero A_{ij} entry in the transition matrix.

The Markov chains described by (31.28) and (31.29), despite their simplicity, have found wide application, e.g., in modeling of physical phenomena, economic time series, and information retrieval. Learning Markov chains requires fitting the M^2 parameters of the transition matrix plus an M-dimensional prior, where M is the size of the observation alphabet. Efficient methods exists to fit stationary first-order Markov chains by maximum likelihood (ML). By using the decomposition in (31.26), substituting the definitions in (31.28) and (31.29), the Markov chain log-likelihood

for a generic sequence $\mathbf{y}$ writes

$$\mathcal{L}(\theta) = \log P(\mathbf{Y} = \mathbf{y}|\theta) = \log \prod_{i'=1}^{M} \pi_{i'}^{\delta(y_1 = i')} \times \prod_{t=2}^{T} \prod_{i,j=1}^{M} A_{ij}^{\delta(y_t = i, y_{t-1} = j)} , \tag{31.30}$$

where $\theta = (A, \pi)$ are the model parameters and $\delta(y_t = i, y_{t-1} = j)$ is the indicator function. For instance, it equals 1 if a transition from $y_{t-1} = j$ to $y_t = i$ can be observed in the sequence and it is 0 otherwise. Similarly, $\delta(y_1 = i') = 1$ if and only if the first symbol of the sequence is i'. The final expression of the log-likelihood is obtained by taking the log into the products and adding appropriate Lagrange multipliers for normalization. The ML estimate is obtained by differentiating this final expression with respect to parameters A_{ij} and π_i, yielding

$$A_{ij} = \frac{\sum_{t=2}^{T} \delta(y_t = i, y_{t-1} = j)}{\sum_{t=2}^{T} \sum_{i=1}^{M} \delta(y_t = i, y_{t-1} = j)} , \tag{31.31}$$

$$\pi_i = \frac{\delta(y_1 = i)}{\sum_{i=1}^{M} \delta(y_1 = i)} . \tag{31.32}$$

Intuitively, the ML estimate corresponds to counting the number of transitions from symbol j to i across time (similarly for the initial state). Generalization to a set of N samples sequences $\mathbf{y}^n$ is straightforward: it suffices to count transitions both in time and across samples, and similarly for the initial symbols y_1^n.

31.4.2 Hidden Markov Models

Markov chains model sequential data assuming that sequence elements are generated by a fully observable stochastic process. In the discrete-state Markov chain, this requires each state of the process to correspond to an observable element of the sequence, i. e., en event. On the other hand, most real-world systems generate observable events that are correlated, but not coincident, with the state of the generating process. More importantly, the only available information can be the outcome of the stochastic process at each time, i. e., event y_t, while the state of the system remains unobservable, i. e., *hidden*. The HMM allows modeling more general stochastic processes where the state transition dynamics is disentangled from the observable information generated by the process. The state-transition dynamics is assumed to be nonobservable and is modeled by a Markov chain of discrete and finite latent variables, i. e., the *hidden states*. The observable information is then generated by such hidden states similarly to how latent variables generate observations in mixture models (see Sect. 31.3.4).

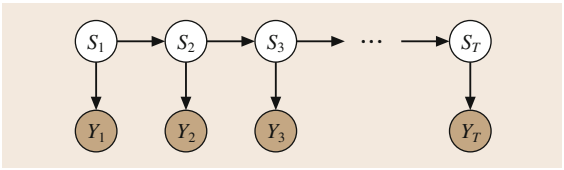


Fig. 31.9 A first-order HMM with hidden states S_t chosen on the discrete domain $\{1, \ldots, C\}$, for $t = 1 \ldots T$

The graphical model of an HMM is exemplified in Fig. 31.9: the hidden states are latent variables S_t, while the sequence elements Y_t are observed.

The conditional dependence expressed by the arrow $S_t \rightarrow Y_t$ indicates that the observed element of the sequence at time t is generated by the corresponding hidden state S_t through the *emission distribution* $b_{s_t}(y_t) = P(Y_t = y_t|S_t = s_t)$. The unknown state-transition dynamics is modeled by the first-order Markov chain of discrete and finite hidden states S_t. By applying the Markovian decomposition in (31.27) to the hidden states chain, the joint distribution of the observed sequence $\mathbf{y} = y_1, \ldots, y_T$ and associated hidden states $\mathbf{s} = s_1, \ldots, s_T$ writes as

$$P(\mathbf{Y} = \mathbf{y}, \mathbf{S} = \mathbf{s}) = P(S_1) \prod_{t=2}^{T} P(S_t|S_{t-1}) P(Y_t|S_t) . \tag{31.33}$$

The actual parameterization of the probabilities in (31.33) depends on the form of the observation and hidden states variables. From Sect. 31.8, a stationary hidden states chain is known to be regulated by the $C \times C$ matrix of *state transitions* $A_{ij} = P(S_t = i|S_{t-1} = j)$ and by the C-dimensional vector of *initial state* probabilities $\pi_i = P(S_t = i)$, where i, j are drawn from $\{1, \ldots, C\}$. For discrete sequence observations $y_t \in \{1, M\}$, the emission distribution is an $M \times C$ emission matrix B such that its elements are

$$b_i(k) = B_{ki} = P(Y_t = k|S_t = i) . \tag{31.34}$$

For continuous observations y_t, the state assignment $S_t = i$ selects the ith emission distributions $b_i(y_t) = P(Y_t|S_t = i)$ from a mixture of C candidates.

An HMM is a latent variable model defined by the $\theta = (\pi, A, B)$ parameters and, implicitly, by the (unkown) number of hidden states C. In [31.58], three notable inference problems are identified for an HMM.

Definition 31.5 Evaluation Problem
Given a model θ and an observed sequence $\mathbf{y}$, determine the likelihood $P(\mathbf{Y} = \mathbf{y}|\theta)$ of the sequence being generated by the model.

Definition 31.6 Learning Problem
Given a dataset of N observed sequences $\mathcal{D} = \{\mathbf{y}^1, \ldots, \mathbf{y}^N\}$ and the number of hidden states C, find the parameters π, A and B that maximize the probability of model $\theta = \{\pi, A, B\}$ having generated the sequences in $\mathcal{D}$.

Definition 31.7 Optimal States Problem
Given a model θ and an observed sequence $\mathbf{y}$, find an optimal state assignment $\mathbf{s} = s_1^*, \ldots, s_T^*$ for the underlying hidden Markov chain.

These classical inference problems are addressed using efficient and numerically stable recursive algorithms that exploit message passing on the HMM junction tree (Sect. 31.2.3) to factorize the, otherwise hardly tractable, joint maximization problems. The underlying intuition is a recursive computation of intermediate probabilities (messages) that are passed forward and backward along the sequence (the junction tree, in practice) to accumulate evidence for solving the joint problem. A discussion of the key aspects of these solutions is provided in the following.

Evaluation

The evaluation problem refers to measuring how well a given HMM matches an observed sequences. Let the model be $\theta = (\pi, A, B)$ and the observed sequence $\mathbf{y} = y_1, \ldots, y_T$, the objective is to find $P(\mathbf{Y} = \mathbf{y}|\theta)$. To effectively compute this probability in the HMM assumption, it is needed to introduce the hidden states assignment corresponding to the observed sequence $\mathbf{y}$. Following the general approach for latent variable models in Eq. (31.11), these are introduced through marginalization on the joint assignment $\mathbf{s} = s_1, \ldots, s_T$

$$P(\mathbf{Y}|\theta) = \sum_{\mathbf{s}} P(\mathbf{Y}, \mathbf{S} = \mathbf{s}|\theta) = \sum_{s_1,\ldots,s_T} P(S_1) \prod_{t=2}^{T} P(S_t|S_{t-1}) P(Y_t|S_t) \,, \tag{31.35}$$

where the joint probability $P(\mathbf{Y}, \mathbf{S}|\theta)$ has been factorized according to the HMM assumption in (31.33).

Direct computation of (31.35) is generally infeasible, as it would require $O(TC^T)$ operations. This probability can be efficiently computed, with $O(TC^2)$ operations, through accumulation of a recursive term that is computed by scanning the sequence from left to right. The procedure is known as *forward algorithm*: let $\mathbf{y}_{1:t}$ be the observed subsequence from position 1 to t, define the *forward probability* as

$$\alpha_t(i) = P(\mathbf{Y}_{1:t} = \mathbf{y}_{1:t}, S_t = i|\theta) \tag{31.36}$$

that is the probability of observing a partial sequence up to position t and the underlying hidden process being in state i at time t. A recursive formulation of the $\alpha_t(i)$ term is obtained by introducing the hidden state S_{t-1} by marginalization, yielding

$$\begin{aligned} \alpha_t(i) &= \sum_{j=1}^{C} P(\mathbf{Y}_{1:t} = \mathbf{y}_{1:t}, S_t = i, S_{t-1} = j|\theta) \\ &= \sum_{j=1}^{C} P(Y_t = y_t|S_t = i, \theta) \\ &\quad \times P(S_t = i|S_{t-1} = j, \theta) \\ &\quad \times P(\mathbf{Y}_{1:t-1} = \mathbf{y}_{1:t-1}, S_{t-1} = j|\theta) \\ &= b_i(y_t) \sum_{j=1}^{C} A_{ij} \alpha_{t-1}(j) \,, \end{aligned} \tag{31.37}$$

where the second equality follows from the conditional independence assumptions of the model. Since, $\text{pa}(S_t) = \{S_{t-1}\}$, the chain element S_t is completely determined by the hidden state at previous time S_{t-1}; similarly, emission Y_t is conditional independent from the rest, given the hidden state S_t.

The forward recursion scans the observed sequence from left to right and recursively computes the $\alpha_t(i)$ values in each position $t = 1, \ldots, T$ using (31.37). At each observed position t, the $\alpha_t(i)$ values are computed for each $i \in \{1, \ldots, C\}$, since the hidden states are not observed. The basis of the recursion is at $t = 1$, where the

(31.37) reduces to $\alpha_1(i) = b_i(y_1)\pi_i$, such that y_1 is the first element of the observed sequence. The likelihood of the full sequence $\mathbf{y} = \mathbf{y}_{1:T}$ is computed at the end of the forward recursion as

$$P(\mathbf{Y}|\theta) = \sum_{i=1}^{C} P(\mathbf{Y}_{1:T}, S_T = i|\theta) = \sum_{i=1}^{C} \alpha_T(i) \,. \tag{31.38}$$

Learning

Learning of an HMM $\theta = (\pi, A, B)$ amounts to finding the values of the parameters π, A and B that are most likely to have generated a dataset of observed i.i.d. sequences $\mathcal{D} = \{\mathbf{y}^1, \ldots, \mathbf{y}^N\}$. From the evaluation problem, we know how to measure the quality of the matching between a sequence $\mathbf{y}$ and a model θ using the likelihood $P(\mathbf{Y}|\theta)$. The HMM learning problem can be solved through ML estimation of θ parameters considering the hidden states as latent variables. As discussed in Sect. 31.3.2, this problem can be solved through application of the EM algorithm, whose HMM version is referred to as *Baum–Welch algorithm* [31.59], which is a form of *sum-product* inference algorithm introduced in Sect. 31.2.3. Marginalization of the hidden states as in (31.35), yields to the HMM log-likelihood on the dataset $\mathcal{D}$

$$\begin{aligned}\mathcal{L}(\theta) &= \log \prod_{n=1}^{N} P(\mathbf{Y}^n|\theta) \\ &= \log \prod_{n=1}^{N} \left\{ \sum_{s_1^n, \ldots, s_{T_n}^n} P(S_1^n) \right. \\ &\qquad \left. \times \prod_{t=2}^{T_n} P(S_t^n|S_{t-1}^n) P(Y_t^n|S_t^n) \right\} \,,\end{aligned} \tag{31.39}$$

where overscript n refers to the nth sequence $\mathbf{y}^n$ and T_n is the corresponding length. The likelihood in (31.39) is intractable due to the nonobservable state assignment that introduces the marginalization term. Following the principles of the EM algorithm, we assume to know the unobserved state assignment, as in (31.30). This can be achieved by introducing indicator variables z_{ti}^n for the unknown assignment, such that $z_{ti}^n = 1$ if the chain is in state i at position t of the nth sequences, and it is 0 otherwise. Given this (assumed) knowledge about the hidden state assignments, if is possible to write the corresponding *completed* likelihood

$$\begin{aligned}&\mathcal{L}_c(\theta) \\ &= \log \prod_{n=1}^{N} \left\{ \prod_{i=1}^{C} P(S_1^n = i)^{z_{1i}^n} \right. \\ &\qquad \left. \times \prod_{t=2}^{T_n} \prod_{i,j=1}^{C} P(S_t^n = i|S_{t-1}^n = j)^{z_{ti}^n z_{(t-1)i}^n} P(Y_t^n|S_t^n = i)^{z_{ti}^n} \right\} \\ &= \sum_{n=1}^{N} \left\{ \sum_{i=1}^{C} z_{1i}^n \log \pi_i + \sum_{t=2}^{T_n} \sum_{i,j=1}^{C} z_{ti}^n z_{(t-1)i}^n \right. \\ &\qquad \left. \times \log A_{ij} + z_{ti}^n \log b_i(y_t^n) \right\} \,,\end{aligned} \tag{31.40}$$

where the latter equality introduces the parameters θ in place of the corresponding probabilities and brings the logarithms into the products.

The EM procedure is applied to the complete log-likelihood in (31.40). Following (31.15), the E-step computes the expected value of $\mathcal{L}_c(\theta)$ with respect to the distribution of the indicator variables $\mathcal{Z} = \{z_{ti}^n\}$, conditional on the observed sequences $\mathcal{D}$ and the current estimate of the parameters $\theta^{(k)}$. Given $\mathcal{L}_c(\theta)$ as in (31.40), taking its conditional expectation with respect to the hidden variables $\mathcal{Z}$, it yields to the following posterior probability:

$$E_{\mathcal{Z}|\mathbf{Y},\theta^{(k)}}[z_{ti}] = P(S_t = i|\mathbf{y}) \,, \tag{31.41}$$

where superscript n is omitted for notational simplicity. The estimation of this posterior is known as the *smoothing* problem. In the Baum–Welch algorithm, this is efficiently solved by a double recursion that exploits the following decomposition of the joint probability

$$\begin{aligned}P(S_t = i, \mathbf{y}) &= P(S_t = i, \mathbf{Y}_{1:t}, \mathbf{Y}_{t+1:T}) \\ &= P(S_t = i, \mathbf{Y}_{1:t}) \\ &\quad \times P(\mathbf{Y}_{t+1:T}|S_t = i) = \alpha_t(i)\beta_t(i) \,,\end{aligned} \tag{31.42}$$

where the observed contribution from the predecessors of t (i. e., $\mathbf{Y}_{1:t}$) is separated from that of its successors (i. e., $\mathbf{Y}_{t+1:T}$). The cancelations in (31.42) follow from the fact that S_t *d-separates* (see definition in Sect. 31.2.1) the elements of the two subsequences, i. e., $\mathbf{Y}_{1:t}$ and $\mathbf{Y}_{t+1:T}$.

The first term in (31.42) is the $\alpha_t(i)$ probability defined in (31.36), which can be computed through the *forward algorithm*. The $\beta_t(i)$ term can also be computed through a recursive procedure known as *backward algorithm*, due to the inverted direction with respect to the forward recursion. Consider the following recursive decomposition

$$\begin{aligned}\beta_{t-1}(j) &= P(\mathbf{Y}_{t:T}|S_{t-1}=j)\\ &= \sum_{i=1}^{C} P(\mathbf{Y}_{t:T}, S_t=i|S_{t-1}=j)\\ &= \sum_{i=1}^{C} P(Y_t|S_t=i)P(\mathbf{Y}_{(t+1):T}|S_t=i)\\ &\quad \times P(S_t=i|S_{t-1}=j)\\ &= \sum_{i=1}^{C} b_i(y_t)\beta_t(i)A_{ij}\,, \end{aligned} \tag{31.43}$$

it can be computed for $2 \le t \le T$ by scanning the sequence backward, assuming $\beta_T(j)=1$ for each $j \in \{1,\dots,C\}$.

The final expression of the smoothed posterior in (31.41) is given by the joint $\alpha-\beta$ recursions, known as the *forward–backward algorithm*, that is

$$\begin{aligned}\gamma_t(i) = P(S_t=i|\mathbf{Y}) &= \frac{P(S_t=i,\mathbf{Y})}{P(\mathbf{Y})}\\ &= \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{C}\alpha_t(j)\beta_t(j)}\,. \end{aligned} \tag{31.44}$$

Note that the forward and backward recursions can be ran in parallel, since the values of α and β do not depend on each other. To complete the derivations of the sufficient statistics for the M-step, it is also necessary to estimate the joint posterior

$$E_{Z|\mathbf{Y},\theta^{(k)}}[z_{ti}z_{(t-1)j}] = P(S_t=i, S_t=j|\mathbf{Y})\,, \tag{31.45}$$

which can be straightforwardly factorized into known probabilities along the lines of (31.42). It turns out that such joint posterior can be estimated using the $\alpha-\beta$ probabilities computed by the *forward–backward algorithm*, that is

$$\begin{aligned}\gamma_{t,t-1}(i,j) &= P(S_t=i, S_t=j|\mathbf{Y})\\ &= \frac{\alpha_{t-1}(j)A_{ij}b_i(y_t)\beta_t(i)}{\sum_{m,l=1}^{C}\alpha_{t-1}(m)A_{lm}b_3(y_t)\beta_t(l)}\,. \end{aligned} \tag{31.46}$$

Parameters $\theta=(\pi,A,B)$ are re-estimated at the M-step, with update equations that follow straightforwardly from the maximization problem in (31.16). It suffices to differentiate (31.40), extended with appropriate Lagrange multipliers to account for the sum-to-one constraints. Intuitively, the update equations can be straightforwardly written from the ML estimates for observable Markov chains in (31.31) and (31.32). It suffices to substitute the observed state counts, obtained through the indicator function $\delta(\cdot)$, with the virtual counts $\gamma(\cdot)$ estimated by (31.44) and (31.46) at the E-step. For the hidden state transition and initial state distributions this yields to

$$A_{ij} = \frac{\sum_{n=1}^{N}\sum_{t=2}^{T^n}\gamma_{t,t-1}^n(i,j)}{\sum_{n=1}^{N}\sum_{t=2}^{T^n}\gamma_{t-1}^n(j)}$$

$$\text{and}\quad \pi_i = \sum_{n=1}^{N}\gamma_1^n(i)\,. \tag{31.47}$$

The estimate of the parameters B depends on the form of the emission distribution: if the observed sequences take values k from a finite alphabet $\{1,\dots,M\}$, the corresponding multinomial emission in (31.34) is updated by

$$B_{ki} = \sum_{n=1}^{N}\sum_{t=1}^{T_n}\gamma_t^n(i)\delta(y_t=k)\,, \tag{31.48}$$

where $\delta(\cdot)$ is the indicator function counting the occurrences of the symbols k in the observed sequences. Real-valued sequences are modeled usually through Gaussian emissions, whose parameters are fit as usual through maximization of the complete log-likelihood.

Particular care must be taken to avoid numerical problems when implementing the *forward–backward algorithm*. Both recursions work with multiplications of small numbers: hence, the values of α and β can underflow for long sequences. To this end, it is advisable to perform them in log-space or to work with scaled versions of the α and β probabilities [31.60]. A sequential version of the smoothing algorithm exists [31.61] that directly computes the smoothed posterior $\gamma_t(i) = P(S_t=i|\mathbf{Y})$ through a γ-recursion that uses the α values generated by the forward algorithm.

Optimal State

Once a model θ has been trained, it can be interesting to determine the most likely hidden state assignment $\mathbf{s}^*$ that has generated an observed sequence $\mathbf{y}$. This inference problem, known also as decoding, has different solutions, since several optimal assignment exists,

depending on the interpretation of what an optimal assignment is. For instance, the optimal hidden sequence can be the one maximizing the expected count of correct states. On the other hand, an optimal assignment might be the sequence of hidden states $\mathbf{s}^*$ with the maximum joint probability $P(\mathbf{Y} = \mathbf{y}, \mathbf{S} = \mathbf{s}^*)$.

The former optimality condition is solved by selecting, at each position t, the most likely state given by the sequence, i. e.,

$$s_t^* = \arg \max_{i=1,\dots,C} P(S_t = i|\mathbf{Y}) \,. \tag{31.49}$$

Clearly, this amounts to select the most likely state for each position independently, using the posterior computed by the Baum–Welch algorithm. Conversely, the latter optimality condition estimates the joint hidden state assignment

$$\mathbf{s}^* = \arg \max_{\mathbf{s}} P(\mathbf{Y}, \mathbf{S} = \mathbf{s}) \,. \tag{31.50}$$

This is a complex inference problem that can be efficiently solved though a dynamic programming approach, known as the *Viterbi algorithm*. Note that the two optimality definitions generally lead to different solutions. For instance, the Viterbi solution is constrained to provide only state transitions allowed by the generating distribution, while this is not the case for the Baum, Welch solution, given that hidden states are selected independently.

The *Viterbi algorithm* is based on a backward recursion that exploits a factorization of the maximization problem in (31.50). Consider the restricted problem of determining the hidden state of the tail element T

$$\begin{aligned}\max_{s_T} P(\mathbf{Y}, S_T = s_T) &= \max_{s_T} \prod_{t=1}^{T} P(Y_t|S_t)P(S_t|S_{t-1}) \\ &= \prod_{t=1}^{T-1} P(Y_t|S_t)P(S_t|S_{t-1}) \max_{s_T} P(Y_t|S_T)P(S_T|S_{T-1}) \,,\end{aligned} \tag{31.51}$$

where the joint probability factorizes according to the Markov chain assumption. We can isolate the maximization problem in the rightmost term

$$\begin{aligned}\epsilon_{T-1}(s_{T-1}) = \max_{s_T} &P(Y_t|S_T = s_T) \\ &\times P(S_T = s_T|S_{T-1} = s_{T-1}) \,,\end{aligned} \tag{31.52}$$

that is a message conveying information on the maximization of the tail element to the penultimate position. Substituting the definition of $\epsilon_{T-1}(s_{T-1})$ back in (31.51) and adding the maximization with respect to s_{T-1}, suggests the recursive formulation of $\epsilon.(\cdot)$ for a generic position $t-1$, i. e.,

$$\begin{aligned}\epsilon_{t-1}(s_{t-1}) = \max_{s_t} &P(Y_t|S_t = s_t) \\ &\times P(S_t = s_t|S_{t-1} = s_{t-1})\epsilon_t(s_t) \,,\end{aligned} \tag{31.53}$$

for $2 \le t \le T$, where $\epsilon_T(s_T) = 1$ is the basis of the recursion. At each step t of the backward recursion, the Viterbi algorithm computes the ϵ-message for each possible assignment of the hidden state of t and propagates it to the predecessor $t-1$. The recursion ends at the initial element of the sequence, where the initial optimal state is obtained as

$$s_1^* = \arg \max_{s} P(Y_t|S_1 = s)P(S_1 = s_1)\epsilon_1(s) \,. \tag{31.54}$$

The assignment of the remaining hidden states is obtained by backtracking through the forward recursion

$$\begin{aligned}s_t^* = \arg \max_{s} &P(Y_t|S_t = s) \\ &\times P(S_t = s|S_{t-1} = s_{t-1}^*)\epsilon_t(s) \,.\end{aligned} \tag{31.55}$$

Note that the *Viterbi* algorithm is a special case of a *max-sum* inference algorithm introduced in Sect. 31.2.3.

31.4.3 Related Models

Higher Order Markov Models

Hidden Markov models serve as a starting point for the design on more complex Markov generative processes, besides the obvious extension to higher order hidden chains [31.62]. *Factorial HMMs* [31.63] generalize the original model by defining super states that are collections of K discrete hidden states, each being part of an independent Markov chain (see Fig. 31.10). This factorial model results in K hidden Markov chains running in parallel: at each time step, the emission depends on the K-dimensional super state, but each state variable is decoupled from those of the other chains and evolves according to its own dynamics. By this means, it is possible to efficiently encode the state dynamics of K objects evolving independently that interact to jointly determine the observation (e.g., K cars moving in the traffic and jointly determining traffic jams).

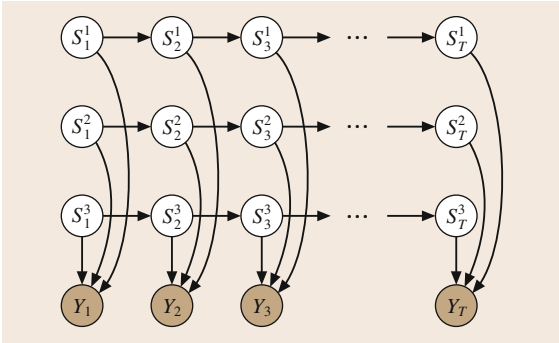


Fig. 31.10 Factorial HMM with $K = 3$ independent hidden Markov chains

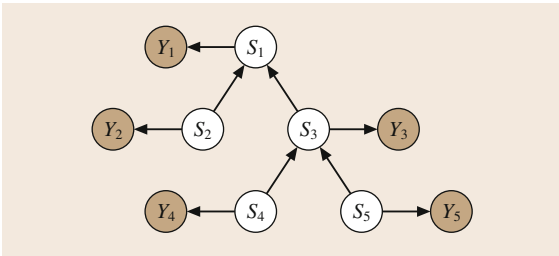


Fig. 31.11 A bottom-up hidden tree Markov model for a simple structure with five nodes: the generative process follows the direction of the *arrows*, i. e., from the leaves to the root ($t = 1$)

Nonhomogenous HMMs

Relaxation of the homogeneity assumption led to the *input/output hidden Markov model* (IO-HMM) [31.64] that allow modeling the causal dependence of the hidden generative process from an additional input sequence **x**. Basically, the IO-HMM enables nonhomogeneous transition and emission distributions that are explicitly dependent (i. e., parameterized) on the currently observed label of the input sequence. An IO-HMM implements a mapping, referred to as *transduction*, from an observed input sequence **x** into an output (target) sequence **y**, realized by the input-conditional hidden process $P(\mathbf{Y}|\mathbf{X})$. Interesting applications of IO-HMM are in learning transformations between modalities in multimedia data [31.65], exploratory analysis of financial time series [31.66] and gene data analysis [31.67].

HMMs for Structured Data

Hidden tree Markov models represent the generative process of more complex, tree-structured information (see Fig. 31.11). Differently from the sequential domain, the direction of the generative process leads to different representational capabilities when dealing with trees. Top-down approaches [31.68] model all possible paths from the root to the leaves of the tree. Bottom-up models [31.69] propose a generative process from the leaves to the root, where complex structures are generated by composition of simpler substructures. Recently, an extension of the IO-HMM has been proposed to learn transductions between trees [31.70].

Bayesian and Nonparametric Extensions

HMMs have been extended to allow a countably infinite number of hidden states through a Bayesian approach where state distributions are modeled by Dirichlet processes [31.71]. Abstracting from the direction of the arrows in Fig. 31.9 leads to a discriminative probabilistic model known as liner-chain *conditional random fields* [31.72], whose capability to model long term dependences is widely used in natural language parsing and computer vision.

31.5 Conclusion and Further Reading

Graphical models have been discussed as an excellent framework for probabilistic modeling of articulated processes that can be described by a *static* set of random variables tied up by probabilistic relationships. Such relationships need not to be necessarily known, a-priori. Several approaches exists to infer them from data, i. e., to determine the presence of a corresponding edge in the graphical model. However, the same approaches tend to *fix* the structure of the graphical model, once this is determined from the data. In other words, these graphical models represent a *static* picture of the process, where the set of random variables and associated relationships is held fixed from a point onward. The nature of sequence data calls for the ability to model more *dynamic* phenomena. Processing of video information requires Markov networks that can unfold their structure across the video sequence. Even *classic* text analysis needs to account for novel generative dynamics, where texts are produced as dynamic streams instead of being static collections of words, e.g., consider blog posts and associated comments, or the stream of social networks status updates. Therefore, the hori-

zon of current research is pushing graphical models to more *dynamic* formulations where, on the one hand, the structure is allowed to change over time and, on the other hand, the model is allowed to dynamically self-tune the number of parameters that is most adequate to represent the process at each time. Following the intuitions underlying the HMM approach, dynamical graphical models are being proposed that are capable of unfolding their structure across time, to better model the dynamics of complex time-varying processes. At the same time, concepts from nonparametric Bayesian statics are being used to develop models where latent variables can be dynamically adjusted to sample from a virtually infinite set of events and where the very same structure of the latent space is adapted across time, i. e., through variable addition and pruning. Such a new class of dynamic graphical models introduces novel computational challenges associated with inference and representation of dynamic knowledge. The answers to this challenges can be partly found in the chapter, in the approximated inference methods discussed for static models and in the principles underlying the unfolding of Markov chains. Finally, it is worth to note that deep learning, described in Chap. 2, is an instance of graphical model where both nonlinearity and dynamic representations play an important role.

References

31.1 S. Kullback, R.A. Leibler: On information and sufficiency, Ann. Math. Stat. **22**, 79–86 (1951)

31.2 F. Rosenblatt: The perceptron: A probabilistic model for information storage and organization in the brain, Psychol. Rev. **65**, 386–408 (1958)

31.3 G. Deco, W. Finnoff, H.G. Zimmermann: Unsupervised mutual information criterion for elemination of overtraining in supervised mulilayer networks, Neural Comput. **7**, 86–107 (1995)

31.4 D.J.C. Mackay: *Information Theory, Inference and Learning Algorithms* (Cambridge Univ. Press, Cambridge 2003)

31.5 R. Salakhutdinov, G. Hinton: Using deep belief nets to learn covariance kernels for Gaussian processes, Adv. Neural Inf. Process. Syst. **20**, 1249–1256 (2008)

31.6 C.M. Bishop: *Pattern Recognition and Machine Learning* (Springer, New York 2006)

31.7 S. Seth, J.C. Principe: Variable selection: A statistical dependence perspective, Proc. Int. Conf. Mach. Learn. Appl. (ICMLA) (2010)

31.8 M. Rao, S. Seth, J. Xu, Y. Chen, H. Tagare, J.C. Principe: A test of independence based on a generalized correlation function, Signal Process. **91**, 15–27 (2011)

31.9 D.D. Lee, H.S. Seung: Learning the parts of objects by non-negative matrix factorization, Nature **401**(6755), 788–791 (1999)

31.10 P. Comon, C. Jutten: *Handbook of Blind Source Separation* (Academic, Oxford 2010)

31.11 A. Hyvärinen, J. Karhunen, E. Oja: *Independent Component Analysis* (Wiley, New York 2001)

31.12 A. Cichocki, R. Zdunek, A.H. Phan, S.-I. Amari: *Nonnegative Matrix Tensor Factorizations* (Wiley, Chichester 2009)

31.13 E. Gaussier, C. Goutte: Relation between plsa and nmf and implications, Proc. 28th Int. ACM Conf. Res. Dev. Inf. Retr. (SIGIR'05) (ACM, New York 2005) pp. 601–602

31.14 D.T. Pham: Mutual information approach to blind separation of stationary sources, IEEE Trans. Inf. Theory **48**, 1935–1946 (2002)

31.15 M. Minami, S. Eguchi: Robust blind source separation by beta divergence, Neural Comput. **14**, 1859–1886 (2002)

31.16 T.-W. Lee, M. Girolami, T.J. Sejnowski: Independent component analysis using an extended infomax algorithm for mixed sub-Gaussian and super-Gaussian sources, Neural Comput. **11**(2), 417–441 (1999)

31.17 K. Labusch, E. Barth, T. Martinetz: Sparse coding neural gas: Learning of overcomplete data representations, Neuro **72**(7–9), 1547–1555 (2009)

31.18 A. Cichocki, S. Cruces, S.-I. Amari: Generalized alpha-beta divergences and their application to robust nonnegative matrix factorization, Entropy **13**, 134–170 (2011)

31.19 I. Csiszár: Axiomatic characterization of information measures, Entropy **10**, 261–273 (2008)

31.20 F. Liese, I. Vajda: On divergences and informations in statistics and information theory, IEEE Trans. Inf. Theory **52**(10), 4394–4412 (2006)

31.21 T. Villmann, S. Haase: Divergence based vector quantization, Neural Comput. **23**(5), 1343–1392 (2011)

31.22 P.L. Zador: Asymptotic quantization error of continuous signals and the quantization dimension, IEEE Trans. Inf. Theory **28**, 149–159 (1982)

31.23 T. Villmann, J.-C. Claussen: Magnification control in self-organizing maps and neural gas, Neural Comput. **18**(2), 446–469 (2006)

31.24 B. Hammer, A. Hasenfuss, T. Villmann: Magnification control for batch neural gas, Neurocomputing **70**(7–9), 1225–1234 (2007)

31.25 E. Merényi, A. Jain, T. Villmann: Explicit magnification control of self-organizing maps for "forbidden" data, IEEE Trans. Neural Netw. **18**(3), 786–797 (2007)

31.26 T. Villmann, S. Haase: Magnification in divergence based neural maps, Proc. Int. Jt. Conf. Artif. Neural Netw. (IJCNN 2011), ed. by R. Mikkulainen (IEEE, Los Alamitos 2011) pp. 437–441

31.27 R. Chalasani, J.C. Principe: Self organizing maps with the correntropy induced metric, Proc. Int. Jt. Conf. Artif. Neural Netw. (IJCNN 2010) (IEEE, Barcelona 2010) pp. 1–6

31.28 T. Lehn-Schiøler, A. Hegde, D. Erdogmus, J.C. Principe: Vector quantization using information theoretic concepts, Nat. Comput. **4**(1), 39–51 (2005)

31.29 R. Jenssen, D. Erdogmus, J.C. Principe, T. Eltoft: The Laplacian PDF distance: A cost function for clustering in a kernel feature space, Adv. Neural Inf. Process. Syst., Vol. 17 (MIT Press, Cambridge 2005) pp. 625–632

31.30 A. Hegde, D. Erdogmus, T. Lehn-Schiøler, Y.N. Rao, J.C. Principe: Vector quantization by density matching in the minimum Kullback-Leibler-divergence sense, Proc. Int. Jt. Conf. Artif. Neural Netw. (IJCNN), Budapest (IEEE, New York 2004) pp. 105–109

31.31 G.E. Hinton, S.T. Roweis: Stochastic neighbor embedding, Adv. Neural Inf. Process. Syst., Vol. 15 (MIT Press, Cambridge 2002) pp. 833–840

31.32 L. van der Maaten, G. Hinten: Visualizing data using t-SNE, J. Mach. Learn. Res. **9**, 2579–2605 (2008)

31.33 K. Bunte, S. Haase, M. Biehl, T. Villmann: Stochastic neighbor embedding (SNE) for dimension reduction and visualization using arbitrary divergences, Neurocomputing **90**(9), 23–45 (2012)

31.34 M. Strickert, F.-M. Schleif, U. Seiffert, T. Villmann: Derivatives of pearson correlation for gradient-based analysis of biomedical data, Intel. Artif. Rev. Iberoam. Intel. Artif. **37**, 37–44 (2008)

31.35 M. Strickert, B. Labitzke, A. Kolb, T. Villmann: Multispectral image characterization by partial generalized covariance, Proc. Eur. Symp. Artif. Neural Netw. (ESANN'2011), Louvain-La-Neuve, ed. by M. Verleysen (2011) pp. 105–110

31.36 V. Gómez-Verdejo, M. Verleysen, J. Fleury: Information-theoretic feature selection for functional data classification, Neurocomputing **72**(16–18), 3580–3589 (2009)

31.37 B. Hammer, T. Villmann: Generalized relevance learning vector quantization, Neural Netw. **15**(8/9), 1059–1068 (2002)

31.38 T. Villmann, M. Kästner: Sparse functional relevance learning in generalized learning vector quantization, Lect. Notes Comput. Sci. **6731**, 79–89 (2011)

31.39 M. Kästner, B. Hammer, M. Biehl, T. Villmann: Functional relevance learning in generalized learning vector quantization, Neurocomputing **90**(9), 85–95 (2012)

31.40 A. Kraskov, H. Stogbauer, P. Grassberger: Estimating mutual information, Phys. Rev. E **69**(6), 66–138 (2004)

31.41 Y.-I. Moon, B. Rajagopalan, U. Lall: Estimating mutual information by kernel density estimators, Phys. Rev. E **52**, 2318–2321 (1995)

31.42 J.C. Principe: *Information Theoretic Learning* (Springer, Heidelberg, 2010)

31.43 R. Andonie, A. Cataron: An information energy LVQ approach for feature ranking, Eur. Symp. Artif. Neural Netw. 2004, ed. by M. Verleysen (d-side, Evere 2004) pp. 471–476

31.44 R. Jenssen, D. Erdogmus, J.C. Principe, T. Eltoft: Some equivalences between kernel methods and information theoretic methods, J. VLSI Signal Process. **45**, 49–65 (2006)

31.45 P.J.G. Lisboa, T.A. Etchells, I.H. Jarman, C.T.C. Arsene, M.S.H. Aung, A. Eleuteri, A.F.G. Taktak, F. Ambrogi, P. Boracchi, E. Biganzoli: Partial logistic artificial neural network for competing risks regularized with automatic relevance determination, IEEE Trans. Neural Netw. **20**(9), 1403–1416 (2009)

31.46 M.I. Jordan: Graphical models, Stat. Sci. **19**, 140–155 (2004)

31.47 D. Koller, N. Friedman: *Probabilistic Graphical Models: Principles and Techniques – Adaptive Computation and Machine Learning* (MIT Press, Cambridge 2009)

31.48 A.P. Dempster, N.M. Laird, D.B. Rubin: Maximum likelihood from incomplete data via the EM algorithm, J. R. Stat. Soc. Ser. B **39**(1), 1–38 (1977)

31.49 M.E. Tipping, C.M. Bishop: Probabilistic principal component analysis, J. R. Stat. Soc. Ser. B **61**(3), 611–622 (1999)

31.50 T. Hofmann: Unsupervised learning by probabilistic latent semantic analysis, Mach. Learn. **42**(1/2), 177–196 (2001)

31.51 M. Welling, C. Chemudugunta, N. Sutter: Deterministic latent variable models and their pitfalls, SIAM Int. Conf. Data Min. (2008)

31.52 D.M. Blei, A.Y. Ng, M.I. Jordan: Latent Dirichlet allocation, J. Mach. Learn. Res. **3**, 993–1022 (2003)

31.53 T. Minka, J. Lafferty: Expectation propagation for the generative aspect model, Proc. Conf. Uncertain. AI (2002)

31.54 T. Griffiths, M. Steyvers: Finding scientific topics, Proc. Natl. Acad. Sci. USA **101**, 5228–5235 (2004)

31.55 M. Blei, D. Blei, T. Griffiths, J. Tenenbaum: Hierarchical topic models and the nested Chinese restaurant process, Adv. Neural Inf. Process. Syst., Vol. 16 (MIT Press, Cambridge 2004) p. 17

31.56 M. Rosen-Zvi, T. Griffiths, M. Steyvers, P. Smyth: The author-topic model for authors and documents, Proc. 20th Conf. Uncertain. Artif. Intell., UAI '04 (AUAI, Corvallis 2004) pp. 487–494

31.57 L.-J. Li, L. Fei-Fei: What, where and who? classifying events by scene and object recognition, IEEE 11th Int. Conf. Comput. Vis. (ICCV) 2007 (2007), pp. 1–8

31.58 L.R. Rabiner: A tutorial on hidden markov models and selected applications in speech recognition, Proc. IEEE **77**(2), 257–286 (1989)

31.59 L.E. Baum, T. Petrie: Statistical inference for probabilistic functions of finite state Markov chains, Ann. Math. Stat. **37**(6), 1554–1563 (1966)

31.60 S.E. Levinson, L.R. Rabiner, M.M. Sondhi: An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition, Bell Syst. Tech. J. **62**(4), 1035–1074 (1983)
31.61 P.A. Devijver: Baum's forward-backward algorithm revisited, Pattern Recogn. Lett. **3**(6), 369–373 (1985)
31.62 M. Brand, N. Oliver, A. Pentland: Coupled hidden Markov models for complex action recognition, Computer Vision and Pattern Recognition, Proc., 1997 IEEE (1997) pp. 994–999
31.63 Z. Ghahramani, M.I. Jordan: Factorial hidden Markov models, Mach. Learn. **29**(2), 245–273 (1997)
31.64 Y. Bengio, P. Frasconi: Input-output HMMs for sequence processing, IEEE Trans. Neural Netw. **7**(5), 1231–1249 (1996)
31.65 Y. Li, H.Y. Shum: Learning dynamic audio-visual mapping with input-output hidden Markov models, IEEE Trans. Multimed. **8**(3), 542–549 (2006)
31.66 B. Knab, A. Schliep, B. Steckemetz, B. Wichern: Model-based clustering with hidden Markov models and its application to financial time-series data, Proc. GfKl 2002 Data Sci. Appl. Data Anal. (Springer, Berlin, Heidelberg 2003) pp. 561–569
31.67 M. Seifert, M. Strickert, A. Schliep, I. Grosse: Exploiting prior knowledge and gene distances in the analysis of tumor expression profiles with extended hidden Markov models, Bioinformatics **27**(12), 1645–1652 (2011)
31.68 M. Diligenti, P. Frasconi, M. Gori: Hidden tree markov models for document image classification, IEEE Trans. Pattern Anal. Mach. Intell. **25**(4), 519–523 (2003)
31.69 D. Bacciu, A. Micheli, A. Sperduti: Compositional generative mapping for tree-structured data – Part I: Bottom-up probabilistic modeling of trees, IEEE Trans. Neural Netw. Learn. Syst. **23**(12), 1987–2002 (2012)
31.70 D. Bacciu, A. Micheli, A. Sperduti: An input-output hidden Markov model for tree transductions, Neurocomputing **112**, 34–46 (2013)
31.71 M.J. Beal, Z. Ghahramani, C.E. Rasmussen: The infinite hidden Markov model, Adv. Neural Inf. Process. Syst. **14**, 577–584 (2002)
31.72 C. Sutton, A. McCallum: An introduction to conditional random fields for relational learning. In: *Introduction to Statistical Relational Learning*, ed. by L. Getoor, B. Taskar (MIT Press, Cambridge 2006) pp. 93–128