1 Introduction to Markov Random Fields

Andrew Blake and Pushmeet Kohli

This book sets out to demonstrate the power of the Markov random field (MRF) in vision. It treats the MRF both as a tool for modeling image data and, coupled with a set of recently developed algorithms, as a means of making inferences about images. The inferences concern underlying image and scene structure to solve problems such as image reconstruction, image segmentation, 3D vision, and object labeling. This chapter is designed to present some of the main concepts used in MRFs, both as a taster and as a gateway to the more detailed chapters that follow, as well as a stand-alone introduction to MRFs.

The unifying ideas in using MRFs for vision are the following:

- Images are dissected into an assembly of nodes that may correspond to pixels or agglomerations of pixels.
- Hidden variables associated with the nodes are introduced into a model designed to "explain" the values (colors) of all the pixels.
- A joint probabilistic model is built over the pixel values and the hidden variables.
- The direct statistical dependencies between hidden variables are expressed by explicitly grouping hidden variables; these groups are often pairs depicted as edges in a graph.

These properties of MRFs are illustrated in figure 1.1. The graphs corresponding to such MRF problems are predominantly gridlike, but may also be irregular, as in figure 1.1(c). Exactly how graph connectivity is interpreted in terms of probabilistic conditional dependency is discussed a little later.

The notation for image graphs is that the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of vertices $\mathcal{V} = (1, 2, \ldots, i, \ldots, N)$ corresponding, for example, to the pixels of the image, and a set of edges $\mathcal{E}$ where a typical edge is $(i, j),\ i, j \in \mathcal{V}$, and edges are considered to be undirected, so that (i, j) and (j, i) refer to the same edge. In the superpixel graph of figure 1.1), the nodes are superpixels, and a pair of superpixels forms an edge in $\mathcal{E}$ if the two superpixels share a common boundary.

The motivation for constructing such a graph is to connect the hidden variables associated with the nodes. For example, for the task of segmenting an image into foreground and background, each node i (pixel or superpixel) has an associated random variable X_i that

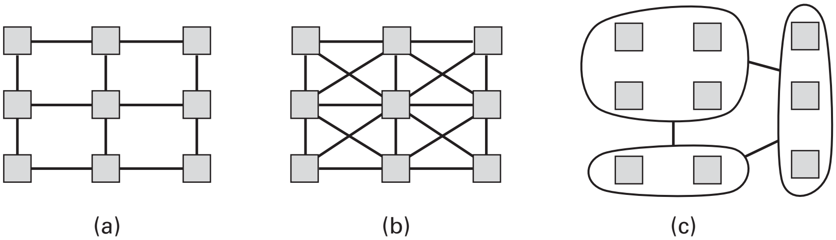

Figure 1.1
Graphs for Markov models in vision. (a) Simple 4-connected grid of image pixels. (b) Grids with greater connectivity can be useful—for example, to achieve better geometrical detail (see discussion later)—as here with the 8-connected pixel grid. (c) Irregular grids are also useful. Here a more compact graph is constructed in which the nodes are superpixels—clusters of adjacent pixels with similar colors.

may take the value 0 or 1, corresponding to foreground or background, respectively. In order to represent the tendency of matter to be coherent, neighboring sites are likely to have the same label. So where $(i, j) \in \mathcal{E}$, some kind of probabilistic bias needs to be associated with the edge (i, j) such that X_i and X_j tend to have the same label—both 0 or both 1. In fact, any pixels that are nearby, not merely adjacent, are likely to have the same label. On the other hand, explicitly linking all the pixels in a typical image, whose foreground/background labels have correlations, would lead to a densely connected graph. That in turn would result in computationally expensive algorithms. Markov models *explicitly* represent only the associations between relatively few pairs of pixels—those pixels that are defined as neighbors because of sharing an edge in $\mathcal{E}$. The great attraction of Markov Models is that they leverage a *knock-on effect*—that explicit short-range linkages give rise to implied long-range correlations. Thus correlations over long ranges, on the order of the diameters of typical objects, can be obtained without undue computational cost. The goal of this chapter is to investigate probabilistic models that exploit this powerful Markov property.

1.1 Markov Chains: The Simplest Markov Models

In a Markov chain a sequence of random variables $\mathbf{X} = (X_1, X_2, \ldots)$ has a joint distribution specified by the conditionals $P(X_i \mid X_{i-1}, X_{i-2}, \ldots, X_1)$. The classic tutorial example [381, sec. 6.2] is the weather, so that $X_i \in \mathcal{L} = \{sunny, rainy\}$. The weather on day i can be influenced by the weather many days previous, but in the simplest form of Markov chain, the dependence of today's weather is linked explicitly only to yesterday's weather. It is also linked *implicitly*, as a knock-on effect, to all previous days. This is a *first-order* Markov assumption, that

$$P(X_i \mid X_{i-1}, X_{i-2}, \ldots, X_1) = P(X_i \mid X_{i-1}). \tag{1.1}$$

This is illustrated in figure 1.2. The set of conditional probabilities $P(X_i \mid X_{i-1})$ is in fact a 2×2 matrix. For example:

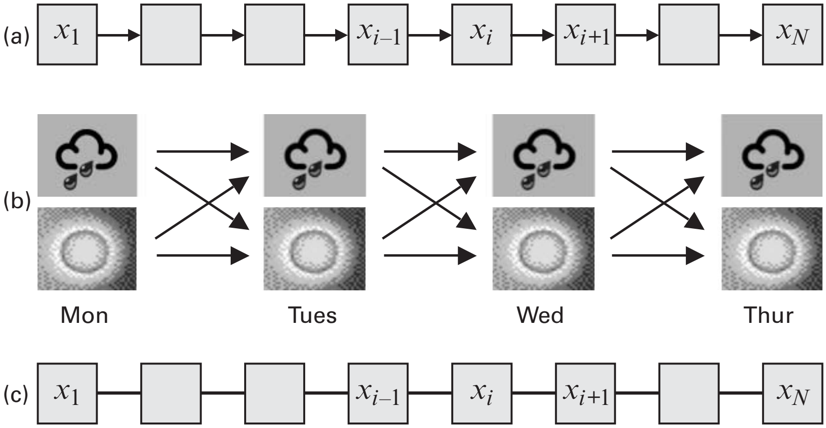

Figure 1.2
A simple first-order Markov chain for weather forecasting. (a) A directed graph is used to represent the conditional dependencies of a Markov chain. (b) In more detail, the state transition diagram completely specifies the probabilistic process of the evolving weather states. (c) A Markov chain can alternatively be expressed as an undirected graphical model; see text for details.

		Yesterday (X_{i-1})	
		Rain	Sun
Today (X_i)	Rain	0.4	0.8
	Sun	0.6	0.2

An interesting and commonly used special case is the *stationary* Markov chain, in which the matrix

$$M_i(x, x') = P(X_i = x \mid X_{i-1} = x') \tag{1.2}$$

is independent of time i, so that $M_i(.,.) = M_{i-1}(.,.)$. In the weather example this corresponds to the assumption that the statistical dependency of weather is a fixed relationship, the same on any day.

We will not dwell on the simple example of the Markov chain, but a few comments may be useful. First, the first-order *explicit* structure implicitly carries longer-range dependencies, too. For instance, the conditional dependency across three successive days is obtained by multiplying together the matrices for two successive pairs of days:

$$P(X_i = x \mid X_{i-2} = x'') = \sum_{x' \in \mathcal{L}} M_i(x, x') M_{i-1}(x', x''). \tag{1.3}$$

Thus the Markov chain shares the elegance of Markov models generally, which will recur later with models for images, that long-range dependencies can be captured for the "price" of explicitly representing just the immediate dependencies between neighbors. Second, higher-order Markov chains, where the *explicit* dependencies go back farther than immediate neighbors, can also be useful. A famous example is "predictive text," in which probable letters in a typical word are characterized in terms of the two preceding letters—taking just the one preceding letter does not give enough practical predictive power. Predictive text, then, is a second-order Markov chain.

The directed graph in figure 1.2a) is a graphical representation of the fact that, for a Markov chain, the joint density can be decomposed as a product of conditional densities:

$$P(\mathbf{x}) = P(x_N \mid x_{N-1}) \dots P(x_i \mid x_{i-1}) \dots P(x_2 \mid x_1) P(x_1), \tag{1.4}$$

where for simplicity, in a popular abuse of notation, $P(\mathbf{x})$ denotes $P(\mathbf{X} = \mathbf{x})$ and, similarly, $P(x_i \mid x_{i-1})$ denotes $P(X_i = x_i \mid X_{i-1} = x_{i-1})$. This convention is used frequently throughout the book. An alternative formalism that is commonly used is the *undirected* graphical model. Markov chains can also be represented in this way (figure 1.2c), corresponding to a factorized decomposition:

$$P(\mathbf{x}) = \Phi_{N,N-1}(x_N, x_{N-1}) \dots \Phi_{i,i-1}(x_i, x_{i-1}) \dots \Phi_{2,1}(x_2, x_1), \tag{1.5}$$

where $\Phi_{i,i-1}$ is a factor of the joint density. It is easy to see, in this simple case of the Markov chain, how the directed form (1.4) can be reexpressed in the undirected form (1.5). However, it is not the case in general, and in particular in 2D images, that models expressed in one form can easily be expressed in the other. Many of the probabilistic models used in computer vision are most naturally expressed using the undirected formalism, so it is the undirected graphical models that dominate in this book. For details on directed graphical models see [216, 46].

1.2 The Hidden Markov Model (HMM)

Markov models are particularly useful as prior models for state variables X_i that are to be inferred from a corresponding set of measurements or observations $z = (z_1, z_2, \dots, z_i, \dots, z_N)$. The observations z are themselves considered to be instantiations of a random variable Z representing the full space of observations that can arise. This is the classical situation in speech analysis [381, sec. 6.2], where z_i represents the spectral content of a fragment of an audio signal, and X_i represents a state in the time course of a particular word or phoneme. It leads naturally to an inference problem in which the *posterior* distribution for the possible states X, given the observations z, is computed via Bayes's formula as

$$P(\mathbf{X} = \mathbf{x} \mid \mathbf{Z} = \mathbf{z}) \propto P(\mathbf{Z} = \mathbf{z} \mid \mathbf{X} = \mathbf{x}) P(\mathbf{X} = \mathbf{x}). \tag{1.6}$$

Here $P(\mathbf{X} = \mathbf{x})$ is the prior distribution over states—that is, what is known about states $\mathbf{X}$ in the absence of any observations. As before, (1.6) is abbreviated, for convenience, to

$$P(\mathbf{x} \mid \mathbf{z}) \propto P(\mathbf{z} \mid \mathbf{x}) P(\mathbf{x}). \tag{1.7}$$

The omitted constant of proportionality would be fixed to ensure that $\sum_{\mathbf{x}} P(\mathbf{x} \mid \mathbf{z}) = 1$. Often multiple models are considered simultaneously, and in that case this is denoted

$$P(\mathbf{x} \mid \mathbf{z}, \omega) \propto P(\mathbf{z} \mid \mathbf{x}, \omega) P(\mathbf{x} \mid \omega), \tag{1.8}$$

where the model parameters $\omega \in \Omega$ may determine the prior model or the observation model or both. The constant of proportionality in this relation would of course depend on $\mathbf{z}$ and on ω.

The prior of an HMM is itself represented as a Markov chain, which in the first-order case was decomposed as a product of conditional distributions (1.4). The term $P(\mathbf{z} \mid \mathbf{x})$ is the *likelihood* of the observations, which is essentially a measure of the quality of the measurements. The more precise and unambiguous the measuring instrument, the more the likelihood will be compressed into a single, narrow peak. This captures the fact that a more precise instrument produces more consistent responses $\mathbf{z}$, under a given condition represented by the state $\mathbf{X} = \mathbf{x}$. It is often assumed—and this is true of the models used in many of the chapters of this book—that observations are independent across sites. The observation at site i depends only on the corresponding state. In other words:

$$P(\mathbf{z} \mid \mathbf{x}) = P(z_N \mid x_N) P(z_{N-1} \mid x_{N-1}) \ldots P(z_1 \mid x_1). \tag{1.9}$$

The directed graphical model for the conditional dependencies of such a first-order HMM is given in figure 1.3a). The figure captures the conditional dependencies both of the underlying

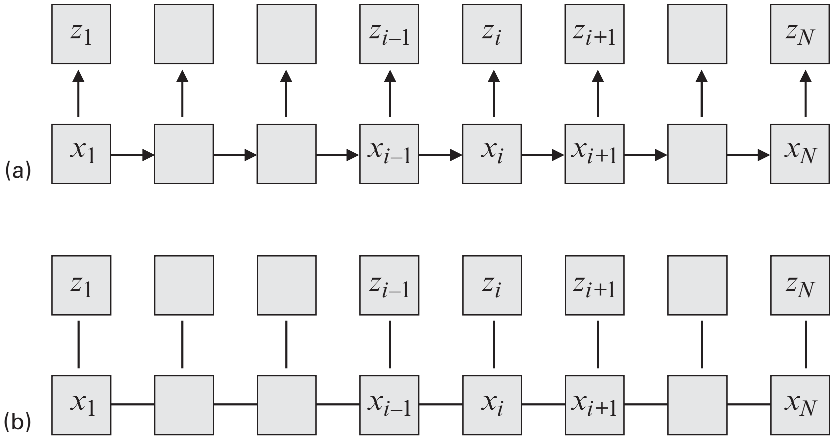

Figure 1.3
A first-order hidden Markov model (HMM). (a) A directed graph is used to represent the dependencies of a first-order HMM, with its Markov chain prior, and a set of independently uncertain observations. (b) Alternatively the HMM can be represented as an undirected graphical model (see text).

Markov chain and of the independence of the observations. Alternatively, an HMM can be expressed as an undirected graphical model, as depicted in figure 1.3(b), in which the prior is decomposed as in (1.5), and the likelihood is

$$P(\mathbf{z} \mid \mathbf{x}) = \Phi_N(x_N)\Phi_{N-1}(x_{N-1}) \dots \Phi_1(x_1), \tag{1.10}$$

where trivially $\Phi_i(x_i) = P(z_i \mid x_i)$.

Discrete HMMs, with a finite label set $\mathcal{L}$, are largely tractable. Rabiner and Juang [382] set out three canonical problems for HMMs, and algorithms to solve them. The problems are the following:

1. *Evaluating the observation probability* $P(\mathbf{z} \mid \omega)$ In this problem there is no explicit state dependence, because it has been "marginalized" out by summation over states:

$$P(\mathbf{z} \mid \omega) = \sum_{\mathbf{x} \in \mathcal{L}^N} P(\mathbf{z} \mid \mathbf{x}, \omega) P(\mathbf{x} \mid \omega). \tag{1.11}$$

The main application of this evaluation is to determine which of a set of known models fits the data best:

$$\max_{\omega \in \Omega} P(\mathbf{z} \mid \omega). \tag{1.12}$$

The quantity $P(\mathbf{z} \mid \omega)$ is also known as the *evidence* [328] for the model ω from the data $\mathbf{z}$.

2. *MAP estimation* Given a model ω and a set of data $\mathbf{z}$, estimate the most probable (maximum a posteriori) sequence $\mathbf{x}$ of states as the mode of the posterior distribution (1.8).

3. *Parameter estimation* Given a set of data $\mathbf{z}$, estimate the parameters $\omega \in \Omega$, a continuous parameter space that best fits the data. This is the problem that must be solved to build a particular model from training data. It is closely related to the model selection problem above, in that both maximize $P(\mathbf{z} \mid \omega)$, the difference being that the model space Ω is, respectively, discrete or continuous.

These three problems are essentially solved by using two algorithms and variants of them.

The first problem requires the *forward* algorithm that computes a marginal distribution node i from the distribution at the previous node $i - 1$:

$$P(x_i, z_1, \dots, z_i \mid \omega) = P(z_i \mid x_i, \omega) \sum_{x_{i-1}} P(x_i \mid x_{i-1}, \omega) P(x_{i-1}, z_1, \dots, z_{i-1} \mid \omega). \tag{1.13}$$

This is a special case of *Belief Propagation* (BP) that will be discussed later in this chapter and in various subsequent chapters in the book. In fact there are two forms of BP [367, 46], and this one is an example of *sum-product* BP. (The name derives from the summation and product steps in (1.13).) The other form is described shortly. In the case of the HMM, where the underlying prior model is simply a Markov chain, sum-product belief propagation is quite straightforward and is an exact algorithm for computing the marginal posteriors. After one complete forward pass, the final marginal distribution is $P(x_N, \mathbf{z} \mid \omega)$, and so finally

$$P(\mathbf{z} \mid \omega) = \sum_{x_N} P(x_N, \mathbf{z} \mid \omega) \tag{1.14}$$

can be computed as the evidence for a known model ω that solves problem 1 above. The forward pass (1.13) constitutes half of BP, the remaining part being a backward pass that recurs from node N back to node 1 (details omitted here, but see [382]). Using the forward and backward passes together, the full set of marginal posterior distributions

$$P(x_i \mid \mathbf{z}, \omega),\ i = 1, \ldots, N \tag{1.15}$$

can be computed. This is required for problem 3 above, in order to compute the expected values of the sufficient statistics that are needed to estimate the parameters ω by *expectation maximization* [121]—also known in the speech analysis literature as the Baum–Welch method [381].

The second algorithm is the Viterbi algorithm, a dynamic programming optimization algorithm applied to the state sequence $\mathbf{x}$. It is also equivalent to a special case of max-product belief propagation, which also is mentioned quite frequently in the book. The aim is to solve the second problem above, computing the MAP estimate of the state vector as

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} P(\mathbf{x} \mid \mathbf{z}, \omega) \tag{1.16}$$

via a forward recursion:

$$P_i(x_i) = P(z_i \mid x_i, \omega) \max_{x_{i-1}} P(x_i \mid x_{i-1}, \omega) P_{i-1}(x_{i-1}) \tag{1.17}$$

where P_i is defined by

$$P_i(x_i) \equiv \max_{x_1, \ldots, x_{i-1}} P(x_1, \ldots, x_i, z_1, \ldots, z_i \mid \omega). \tag{1.18}$$

Each forward step of the recursion can be viewed as a message-passing operation. In step $i - 1$ of the computation, node $i - 1$ sends the message $P_i(x_i)$ to node i.

After the forward steps are complete, the final component $\hat{x}_N$ of the MAP solution $\hat{x}$ can be computed as

$$\hat{x}_N = \arg\max_{x_N} P_n(x_N). \tag{1.19}$$

This is followed by a backward recursion

$$\hat{x}_{i-1} = \arg\max_{x_{i-1}} P(\hat{x}_i \mid x_{i-1}, \omega)\ P_{i-1}(x_{i-1}), \tag{1.20}$$

after which all components of $\hat{x}$ are determined.

The purpose of the discussion in this section has been largely to explain the nature of hidden variables in simple Markov models, as a precursor to later discussion of hidden variables in the more complex, two-dimensional kinds of models that are used in vision. However, even in vision the discrete HMM structure has some direct applications. It has proved useful for representing temporal problems that are somewhat analogous to speech analysis, but in which the audio input is replaced by a time sequence of visual features. Well-known examples include the recognition of American Sign Language [449] and the recognition of hand gestures for command and control [522]. This book deals mainly with discrete Markov models—that is, ones in which the states of each X_i belong to a finite set $\mathcal{L}$. However, in vision by far the greater application of timelike HMMs employs continuous state-space to represent position, attitude, and shape in visual tracking [333, 477, 50, 253, 209, 124]. In such continuous settings the HMM becomes a classical or nonclassical form of Kalman filter. Both exact solutions to the estimation problems that arise, and efficient approximate solutions, are much studied, but are outside the scope of this book.

1.3 Markov Models on Trees

In the following section 1.4, Markov Random Fields (MRFs) are defined as probabilistic models over undirected graphs. On the way there, we now consider undirected models on trees as intermediate in complexity between the linear graphs—chains and HMMs—of section 1.2, and graphs of unrestricted connectivity. Clearly the HMM graph (figure 1.3b) is a special case of an undirected model on a tree. Trees appear to be of intermediate complexity but, perhaps surprisingly, turn out to be closer to HMMs, in that inference can be performed exactly. The Viterbi and forward-backward algorithms for HMMs generalize to two different kinds of message passing on trees. However, once the nodes on two leaves of a tree are coalesced into a single leaf—for example, leaves **b** and **d** in figure 1.4—a circuit may be formed in the resulting graph, and message-passing algorithms are no longer an exact solution to the problems of inference.

As with Markov chains and HMMs, in undirected trees the topological structure conveys two aspects of the underlying probabilistic model. First are the conditional independence properties, that:

$$P(x_i \mid \{x_j,\ j \neq i\}) = P(x_i \mid \{x_j,\ (i,j) \in \mathcal{E}\}). \tag{1.21}$$

The set $B_i = \{j : (i,j) \in \mathcal{E}\}$ is known as the Markov blanket of i, its neighbors in the tree (or generally graph) $\mathcal{G}$. The second aspect is the decomposition of the joint distribution, the generalization of (1.5) and (1.10). How can a distribution with this independence property be constructed? The answer is, as a distribution that is factorized over the edges of the tree:

$$P(\mathbf{x}) = \prod_{(i,j)\in\mathcal{E}} F_{i,j}(x_i, x_j). \tag{1.22}$$

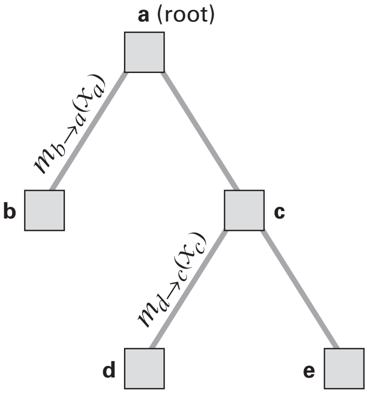

Figure 1.4
Message passing for MAP inference in tree structured graphical models. A graphical model containing five nodes (a, b, c, d, and e) connected in a tree structure.

1.3.1 Inference on Trees: Belief Propagation (BP)

The message-passing formulation of the Viterbi algorithm can be generalized to find marginal distributions over individual variables and the MAP estimate in a tree structured graphical model. The resulting algorithm is known as belief propagation [367] and has two variants: *max-product*, for computing the MAP solution, and sum-product (mentioned earlier), which allows computation of marginals of individual random variables.

Max-product message passing is similar in spirit to dynamic programming algorithms. Like the Viterbi algorithm, it works by passing messages between tree nodes in two stages. In the first stage, messages are passed from the leaf nodes to their parents, which in turn pass messages to their parents, and so on until the messages reach the root node. The message $m_{i\to j}$ from a node i to its parent j is computed as

$$m_{i\to j}(x_j) = \max_{x_i} P(x_j, x_i) \prod_{k\in\mathcal{N}_c(i)} m_{k\to i}(x_i) \tag{1.23}$$

where $\mathcal{N}_c(i)$ is the set of all children of node i. The MAP label of the variable at the root r of the tree can be computed as

$$\hat{x}_r = \arg\max_{x_r} \prod_{k\in\mathcal{N}_c(r)} m_{k\to r}(x_r). \tag{1.24}$$

Given the MAP label $\hat{x}_p$ of a variable X_p, the label of any of its children i can be found as

$$\hat{x}_i = \max_{x_i} P(\hat{x}_p, x_i) \prod_{k\in\mathcal{N}_c(i)} m_{k\to i}(x_i). \tag{1.25}$$

1.3.2 Example: Max-Product BP on a Five-Node Model

Consider the undirected, tree-structured graphical model shown in figure 1.4. The joint distribution factorizes as

$$P(\mathbf{x}) = P(x_a, x_b)P(x_a, x_c)P(x_c, x_e)P(x_c, x_d). \tag{1.26}$$

The messages computed by max-product are

$$m_{d\to c}(x_c) = \max_{x_d} P(x_c, x_d) \tag{1.27}$$

$$m_{e\to c}(x_c) = \max_{x_e} P(x_c, x_e) \tag{1.28}$$

$$m_{c\to a}(x_a) = \max_{x_c} P(x_a, x_c)m_{e\to c}(x_c)m_{d\to c}(x_c) \tag{1.29}$$

$$m_{b\to a}(x_a) = \max_{x_b} P(x_a, x_b). \tag{1.30}$$

The MAP labels can be found as

$$\hat{x}_a = \max_{x_a} m_{b\to a}(x_a)m_{c\to a}(x_a) \tag{1.31}$$

$$\hat{x}_b = \max_{x_b} P(\hat{x}_a, x_b) \tag{1.32}$$

$$\hat{x}_c = \max_{x_c} P(\hat{x}_a, x_c)m_{e\to c}(x_c)m_{d\to c}(x_c) \tag{1.33}$$

$$\hat{x}_d = \max_{x_d} P(\hat{x}_c, x_d) \tag{1.34}$$

$$\hat{x}_e = \max_{x_d} P(\hat{x}_c, x_e). \tag{1.35}$$

The sum product BP algorithm computes the marginal distributions $P(x_i)$ for all variables X_i. It essentially works in a way similar to max-product BP (1.23), except that rather than taking the max, a *sum* is performed over the different labels:

$$m_{i\to j}(x_j) = \sum_{x_i} P(x_j, x_i) \prod_{k\in\mathcal{N}_c(i)} m_{k\to i}(x_i) \tag{1.36}$$

where $\mathcal{N}_c(i)$ is the set of all children of node i. The marginal $P(x_i)$ can be computed by taking the product of the messages sent to the root node i:

$$P(x_i) = \prod_{k\in\mathcal{N}_c(i)} m_{k\to i}(x_i). \tag{1.37}$$

Now, by successively rearranging the tree so that each node i in turn becomes the root node, $P(x_i)$ can be computed for all nodes i.

1.4 MRFs: Markov Models on Graphs

At the start of the chapter, the choice of graphs for image processing was motivated by the need to establish dependencies between pixels that are nearby in an image. The graphs that were proposed for that purpose are not trees, but contain cycles of edges, typically many cycles, as in figure 1.1. The representation of independence in undirected graphs follows the methodology given above for trees (1.21). The random variable at a node is dependent directly only on random variables in its Markov blanket B_i. The Hammersley–Clifford theorem [98] gives the general form for a distribution with these Markov properties:

$$P(\mathbf{x}) = \prod_{c \in \mathcal{C}} F_c(\mathbf{x}) \tag{1.38}$$

where $\mathcal{C}$ is the set of maximal cliques of $\mathcal{G}$—defined to be the set of maximal subgraphs of $\mathcal{G}$ that are fully connected in $\mathcal{E}$, and maximal in the sense that no further node can be added to a clique without removing the full connectedness property. Note that for a tree the cliques are simply the edges of the graph, and so the decomposition (1.38) simplifies to the decomposition (1.22) for trees, above. Usually the decomposition is expressed in terms of an energy or cost function $E(\mathbf{x})$:

$$P(\mathbf{x}) \propto \exp\left(-E(\mathbf{x})\right) \;\; \text{where } E(\mathbf{x}) = \sum_{c \in \mathcal{C}} \Psi_c(\mathbf{x}). \tag{1.39}$$

More generally, there may be a dependence on parameters, so that

$$P(\mathbf{x}) = \frac{1}{Z(\omega)} \exp\left(-E(\mathbf{x}, \omega)\right) \;\; \text{where } E(\mathbf{x}, \omega) = \sum_{c \in \mathcal{C}} \Psi_c(\mathbf{x}_c, \omega), \tag{1.40}$$

and now the *partition function*

$$Z(\omega) = \sum_{\mathbf{x}} \exp\left(-E(\mathbf{x}, \omega)\right) \tag{1.41}$$

is included to maintain the normalization condition for the distribution, $\sum_{\mathbf{x}} P(\mathbf{x}) = 1$.

An alternative representation of the undirected graph in figure 1.1 is the *factor graph* [276]. The undirected graph (or *Markov network*) makes conditional dependencies explicit, but factorization properties are somewhat implicit—they need to be computed in terms of maximal cliques. Factor graphs are a little more complex in that they introduce a second type of node, the function node, in addition to the nodes for variables but have the advantage of making factorization explicit. In many of the cases used in this book—for example, figure 1.1a, with 4-way connectivity between pixels—the factorization structure is straightforward. Each edge in that example is a maximal clique, and therefore the factors are functions of the two variables on the nodes belonging to each edge. On the other hand, figure 1.1b, with its 8-way connectivity, has a more complex set of statistical dependencies, and the factors may not simply correspond to edges. The most general factor structure for the Markov model with the statistical dependencies denoted by that graph is a function of the *four variables* in each of the 2×2 squares of variables, which are the maximal cliques of the graph. In computer vision, it is usual to define models directly in terms of their factors, in contrast to normal practice in statistical modeling, where models are defined first in terms of their Markov properties, with factors specified subsequently over maximal cliques, as in (1.38). In the more complex cases of factors involving more than two variables at a time, factor graphs are useful, and are mentioned in chapters 21 and 24. For the most part, though, pairwise factors and simple undirected graphs suffice.

1.4.1 Example: Pseudo-Boolean Energy on a 4-Connected Graph of Pixels

A standard example of an MRF, just about the simplest one that is interesting, is the *Ising* model with the single parameter $\omega = \{\gamma\}$, whose origins are in statistical physics [523]. The state-space consists of Boolean variables $x_i \in \mathcal{L} = \{0, 1\}$, and the energy function $E(\mathbf{x})$ is termed a *Pseudo-Boolean Function* (PBF)—because its input is Boolean but its output is not (the energy is real valued). As for the graph, the maximal cliques are the horizontal and vertical edges of the rectangular graph of pixels shown in figure 1.5a. All cliques in this

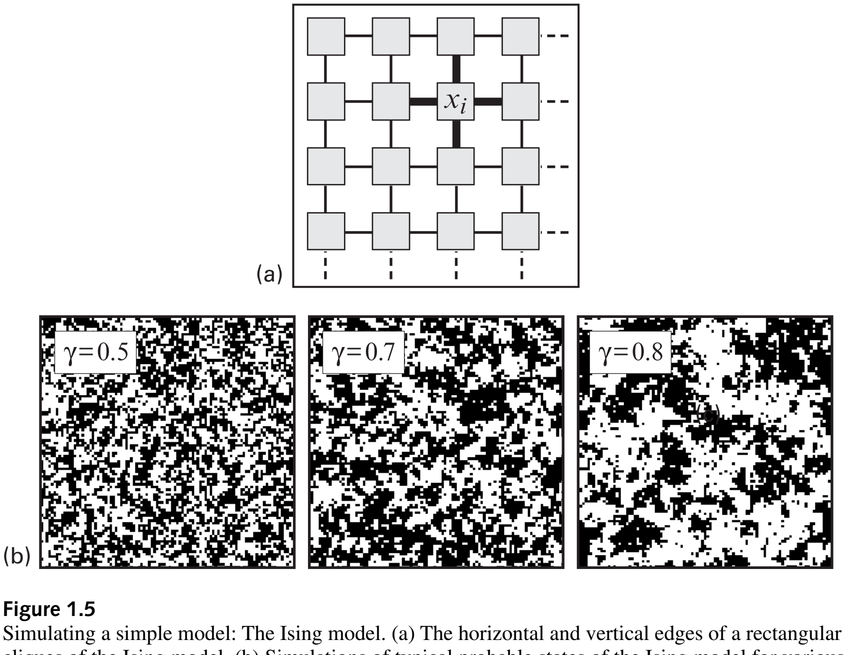

Figure 1.5
Simulating a simple model: The Ising model. (a) The horizontal and vertical edges of a rectangular grid form the cliques of the Ising model. (b) Simulations of typical probable states of the Ising model for various values of the coherence parameter γ.

case are of size 2, containing two nodes (pixels). The clique potentials, referred to in this kind of model as *pairwise* potentials because the cliques are pairs of pixels, are

$$\Psi_{ij}(x_i, x_j) = \gamma |x_i - x_j|. \tag{1.42}$$

This represents a penalty γ that increases the energy E wherever adjacent x_i and x_j have different values, and so reduces the joint probability $P(\mathbf{x})$ by a factor e^{γ}. This enhances the probability of configurations $\mathbf{x}$ in which there is large-scale agreement between the values of adjacent pixels. In fact, a moment's thought will make it clear that the *total* penalty $\sum_{(i,j)\in\mathcal{C}} \Psi_{ij}$ is simply γ times the total perimeter (in the Manhattan metric) of boundaries separating regions of value 1 from regions of value 0. Thus the distribution $P(\mathbf{x})$ favors configurations $\mathbf{x}$ in which that total perimeter is relatively small. The simulations in figure 1.5b show how higher values of γ indeed tend to favor larger regions of 1s and 0s.

It is worth saying at this stage, and this will come out repeatedly later, that algorithms for inference with general graphs that contain loops—many loops in the case of the example above—are hard. Even simulation from the Ising model is hard, compared with the equivalent simulation for a chain or tree, which is straighforward and can be done exactly. The simulations of the Ising model above were done using a form of *Markov chain Monte Carlo*, adapted specifically for the Ising model [165], the Swendsen–Wang iterative sampler [25].

1.5 Hidden MRF Models

A Markov random field $P(\mathbf{X} = \mathbf{x})$, as in the previous section, can act as a prior model for a set of hidden random variables $\mathbf{X}$, under a set of observations $\mathbf{z}$, in direct analogy to the HMM model of section 1.2. As with HMMs, the observations are most simply modeled as random variables that are independent when conditioned on the hidden variables $\mathbf{X}$. This is illustrated in figure 1.6, in which the simple 4-connected graph of figure 1.5a appears as a layer of hidden variables $\mathbf{x}$ with observations $\mathbf{z}$ distributed across sites. Each individual observation z_i depends statistically just on the state x_i of the corresponding pixel. Now the posterior for the state $\mathbf{x}$ of the pixels is obtained from Bayes's formula, just as it was for HMMs (1.6), as

$$P(\mathbf{x} \mid \mathbf{z}, \omega) \propto P(\mathbf{z} \mid \mathbf{x}, \omega) P(\mathbf{x} \mid \omega), \tag{1.43}$$

with the observation likelihood $P(\mathbf{z} \mid \mathbf{x})$ factorized across sites/pixels as it was earlier (1.9), and including the possibility of multiple models as before. It is common also to express this *posterior MRF* in terms of a sum of energies, generalizing the prior MRF (1.40) to include terms for the observation likelihood:

$$P(\mathbf{x} \mid \mathbf{z}, \omega) = \frac{1}{Z(\mathbf{z}, \omega)} \exp -E(\mathbf{x}, \mathbf{z}, \omega), \tag{1.44}$$

$$\text{where } E(\mathbf{x}, \mathbf{z}, \omega) = \sum_{c\in\mathcal{C}} \Psi_c(\mathbf{x}, \omega) + \sum_i \Phi_i(x_i, z_i). \tag{1.45}$$

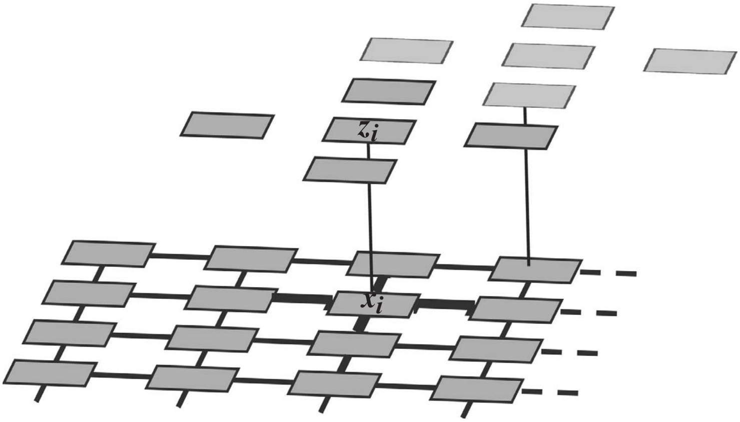

Figure 1.6
Two-dimensional hidden Markov model. An MRF on a regular grid, as in figure 1.5, serves here as the prior over hidden variables in a model that is coupled to an array $\mathbf{z}$ of observations.

Here $Z(\mathbf{z}, \omega)$ is the partition function for the posterior MRF. Unlike the HMM (1.8), for which Z can in fact be computed quite easily, computing the partition function $Z(\mathbf{z}, \omega)$ for the posterior MRF is intractable.

The most common form of inference over the posterior MRF in vision and image-processing problems, is Maximum A Posteriori (MAP) estimation. MAP inference of $\mathbf{x}$ is done in principle by computing $\hat{\mathbf{x}} = \arg\max P(\mathbf{x} \mid \mathbf{z})$, or equivalently by minimizing energy:

$$\hat{\mathbf{x}} = \arg\min E(\mathbf{x}, \mathbf{z}, \omega). \tag{1.46}$$

Note that this does not require knowledge of the partition function $Z(\mathbf{z}, \omega)$, which is just as well, given its intractability. The energy functions for many commonly used Markov models (see examples below) can be written as a sum of unary and pairwise terms:

$$E(\mathbf{x}, \mathbf{z}, \omega) = \sum_{i \in \mathcal{V}} \Phi_i(x_i, z_i, \omega) + \sum_{(i,j) \in \mathcal{E}} \Psi_{ij}(x_i, x_j, \omega). \tag{1.47}$$

Algorithms for computing the MAP are discussed in the following sections. Suffice it to say that the MAP can in fact be computed exactly in a time that is polynomial with respect to the size N of the image array, using *graph cut*, at least when the prior $P(\mathbf{x})$ is an Ising distribution.

1.5.1 Example: Segmentation on a 4-Connected Graph of Pixels

Here we give the simplest useful example of a hidden MRF model, for segmenting an image into foreground and background components. The state-space is Boolean, so $x_i \in \{0, 1\}$ denotes background/foreground labels, respectively. The model, originated by Boykov and

Jolly [66], has a number of variants. For tutorial purposes the simplest of them is illustrated here. It uses the Ising model as a prior to encourage the foreground and background components to be as coherent as possible. Thus the Ψ terms in the hidden MRF model (1.45) are the ones from the Ising prior (1.42). The likelihood terms (for details, see chapter 7 on MRF models for segmentation) can be specified by constructing histograms $h_{\mathrm{F}}(z)$ and $h_{\mathrm{B}}(z)$ in color space for foreground and background, respectively (taking care to avoid zeros), and setting

$$\Phi_i(z_i) = \log h_{\mathrm{F}}(z_i) - \log h_{\mathrm{B}}(z_i). \tag{1.48}$$

The resulting model specifies a posterior, which is maximized to obtain the estimated segmentation $\hat{x}$, and the resulting method is demonstrably effective, as figure 1.7 shows.

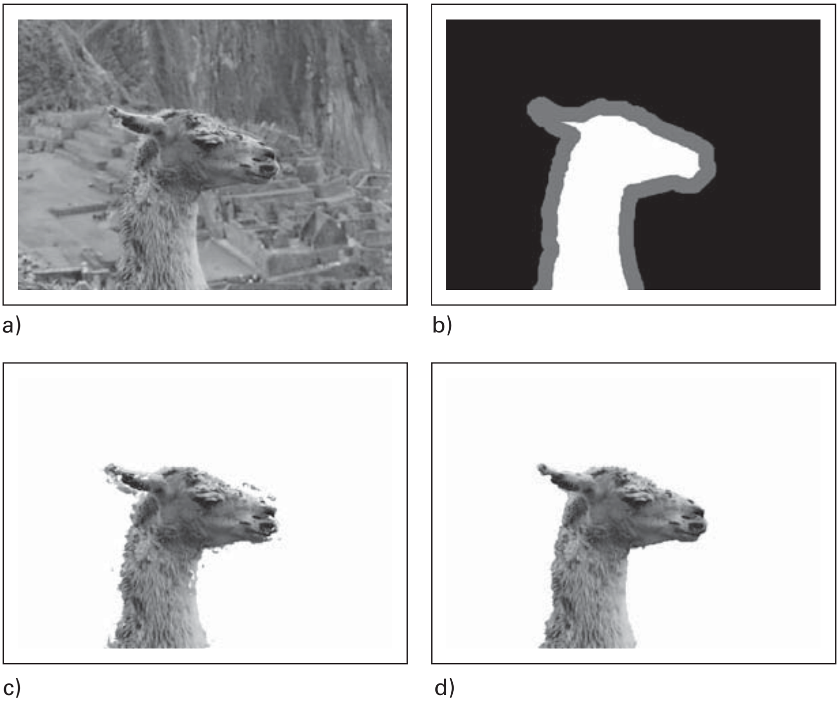

Figure 1.7
MRF model for bilevel segmentation. (a) An image to be segmented. (b) Foreground and background regions of the image are marked so x_i in those regions is no longer hidden but observed. The problem is to infer foreground/background labels in the remaining unlabeled region of the *trimap*. (c) Using simply a color likelihood model learned from the labeled regions, without the Ising prior, the inferred labeling is noisy. (d) Also introducing a pairwise Ising term, and calculating the MAP estimate for the inferred labels, deals substantially with the noise and missing data. (Results of the CRF variant of the Ising term, described below, are illustrated here.)