G

Graphical Models

Julian McAuley[1,2], Tibério Caetano, Wray Buntine
[1]Statistical Machine Learning Program,
NICTA, Locked Bag 8001,
Canberra ACT 2601,
Australia
[2]Research School of Information Sciences
and Engineering, Australian National University,
Canberra ACT 0200,
Australia

Synonyms

Bayesian Networks, Markov Random Field

Definition

The notation we shall use is defined in Table 1, and some core definitions are presented in Table 2. In each of the examples presented in Fig. 1, we are simply asserting that

$$\underbrace{p(x_A, x_B|x_C)}_{\text{function of three variables}} = \underbrace{p(x_A|x_C)p(x_B|x_C)}_{\text{functions of two variables}}, \quad (1)$$

which arises by a straightforward application of the product rule (Definition 1), along with the fact that X_A and X_B are *conditionally independent*, given X_C (Definition 3). The key observation we make is that while the left-hand side of (Eq. 1) is a function of three variables, its conditional independence properties allow it to be *factored* into functions of two variables.

In general, we shall have a series of conditional independence statements about X:

$$\left\{X_{A_i} \perp\!\!\!\perp X_{B_i} \,\middle|\, X_{C_i}\right\}. \quad (2)$$

It is precisely these statements that define the "structure" of our multivariate distribution, which we shall express in the form of a graphical model.

Graphical Models. Table 1 Notation

Notation	Description
$X = (X_1 \dots X_N)$	A random variable (we shall also use $X = (A, B, C \dots)$ in figures to improve readability)
$x = (x_1 \dots x_N)$	A *realization* of the random variable X
$\mathcal{X}$	The sample space (domain) of X
X_A	X can be indexed by a *set*, where we assume $A \subseteq \{1 \dots N\}$
$p(x)$	The probability that $X = x$
$\tilde{A}$	The *negation* of A, i.e., $\{1 \dots N\} \setminus A$
$X_A \perp\!\!\!\perp X_B$	X_A and X_B are *independent*
$X_A \perp\!\!\!\perp X_B \mid X_C$	X_A and X_B are *conditionally independent*, given X_C

Motivation and Background

Graphical models are often used to model multivariate data, since they allow us to represent high-dimensional distributions *compactly*; they do so by exploiting the *interdependencies* that typically exist in such data. Put simply, we can take advantage of the fact that high-dimensional distributions can often be decomposed into *low-dimensional factors* to develop efficient algorithms by making use of the distributive law: $ab + ac = a(b + c)$.

Some motivating examples are presented in Fig. 1; similar examples are ubiquitous in fields ranging from computer vision and pattern recognition, to economics and the social sciences. Although we are dealing with high-dimensional data, we can make certain statements about the *structure* of the variables involved, allowing us to express important properties about the distribution compactly. Some of the properties we would like to compute include the probabilities of particular outcomes, and the outcomes with the highest probability.

C. Sammut, G. Webb (eds.), *Encyclopedia of Machine Learning*, DOI 10.1007/978-0-387-30768-8,

Graphical Models. Table 2 Definitions

Definition 1 (product Rule). $p(x_A, x_B) = p(x_A\|x_B)p(x_B)$.
Definition 2 (marginalization). $p(x_A) = \sum_{x_{\tilde{A}} \in \mathcal{X}_{\tilde{A}}} p(x_A, x_{\tilde{A}})$.
Definition 3 *(conditional independence). X_A and X_B are said to be conditionally independent (given X_C) iff $p(x_a\|x_b, x_c) = p(x_a\|x_c)$, for all x_a, x_b, and x_c; the conventional definition of "independence" is obtained by setting $X_C = \varnothing$.*

Graphical Models. Figure 1. Some examples of conditional independence; we say that X_A and X_B are *conditionally independent*, given X_C, or more compactly $X_A \perp\!\!\!\perp X_B \mid X_C$

Theory

Directed Graphical Models

Due to the product rule (Definition 1), it is clear that *any* probability distribution can be written as

$$p(x) = \prod_{i=1}^{N} p(x_{\pi_i}|x_{<\pi_i}) \tag{3}$$

for an arbitrary permutation π of the labels, where we define $<i := \{1 \ldots i-1\}$. For example any four-dimensional distribution can be written as

$$p(x_a, x_b, x_c, x_d) = p(x_c)p(x_b|x_c)p(x_d|x_c, x_b) \\ p(x_a|x_c, x_b, x_d). \tag{4}$$

With this idea in mind, consider a model $p(x)$ for which we have the conditional independence statements

$$\left\{p(x_{\pi_i}|x_{<\pi_i}) = p(x_{\pi_i}|x_{pa_{\pi_i}})\right\}, \tag{5}$$

where $pa_{\pi_i} \subset <\pi_i$. We now have

$$p(x) = \prod_{i=1}^{N} p(x_{\pi_i}|x_{pa_{\pi_i}}). \tag{6}$$

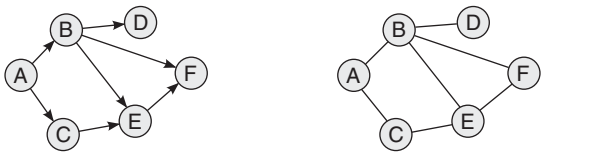
We can interpret pa_i as referring to the "parents" of the node i. Essentially, we are saying that a variable is conditionally independent on its nondescendants, given its parents.

We can represent (Eq. 6) using a directed acyclic graph (DAG) by representing each variable X_i as a node; an arrow is formed from X_j to X_i if $j \in pa_i$. An example of such a representation is given in Fig. 2. It can easily be shown that the resulting graph is always acyclic.

A *Bayesian Network* (a type of *directed* graphical model) is simply a set of probability distributions of the form $p(x) = \prod_{i=1}^{N} p(x_i|x_{pa_i})$. Every Bayesian Network can be represented as a DAG, though we often simply say that the Bayesian Network "is" the DAG. Some trivial examples, and the type of independence statements they imply are shown in Fig. 3.

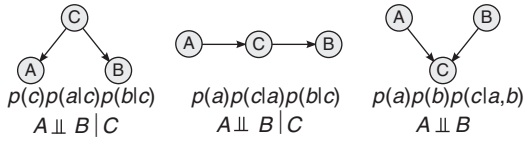
We finish this section with a simple lemma:

Lemma 4 (topological sort) *Every DAG has at least one permutation π that "sorts" the nodes such that each node has a larger index than its parents; in other words, the factorization associated to any DAG can be written in*



$p(a)p(b|a)p(c|a)p(d|b)p(e|b,c)p(f|b,e)$ $\frac{1}{Z}\psi(a,b)\psi(a,c)\psi(b,d)\psi(c,e)\psi(b,e,f)$

Graphical Models. Figure 2. A directed model (*left*) and an undirected model (*right*). The joint distributions they represent are shown

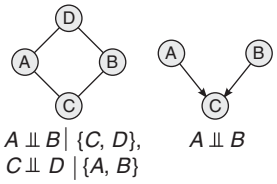


Graphical Models. Figure 3. Some simple Bayesian Networks, and their implied independence statements. Note in particular that in the rightmost example, we *do not* have $A \perp\!\!\!\perp B \mid C$

the form of (Eq. 6) for at least one π such that $\pi_i > j$ for all i, where $j \in pa_{\pi_i}$.

Undirected Graphical Models

Although we have shown how conditional independence statements in the form of (Eq. 5) can be modeled using a DAG, there exist certain conditional independence statements that are not satisfied by *any* Bayesian Network, such as those in Fig. 4.



Graphical Models. Figure 4. There is no Bayesian Network that captures precisely the conditional independence properties of the Markov random field at *left*; there is no Markov random field that captures precisely the conditional independence properties of the Bayesian Network at *right*

Markov random fields (or MRFs) allow for the specification of a different class of conditional independence statements, which are naturally represented by *undirected graphs* (UGs for short). The results associated with MRFs require a few additional definitions:

Definition 5 (clique) *A set of nodes X in a graph $\mathcal{G} = (V, E)$ is said to form a clique if $(X_i, X_j) \in E$ for every $X_i, X_j \in X$ (i.e., the subgraph X is fully connected).*

Definition 6 (maximal clique) *A clique X is said to be maximal if there is no clique Y such that $X \subset Y$.*

A Markov random field is a probability distribution of the form $p(x) = \frac{1}{Z}\prod_{c\in\mathcal{C}}\psi_c(x_c)$, where $\mathcal{C}$ is the set of maximal cliques of $\mathcal{G}$, ψ_c is an arbitrary nonnegative real-valued function and Z is simply a normalization constant ensuring that $\sum_x p(x) = 1$.

Conversion from Directed to Undirected Graphical Models It is possible to convert a directed graphical model to an undirected graphical model via the following simple procedure:

- For every node X_i with parents pa_{X_i}, add undirected edges between every $X_j, X_k \in pa_{X_i}$.
- Replace all directed edges with undirected edges.

In other words, we are replacing statements of the form $p(x_A|x_B)$ with $\psi(x_A, x_B)$, so that the nodes $\{X_i\} \cup pa_{X_i}$ now form a clique in the undirected model. This procedure of "marrying the parents" is referred to as *Moralization*. Naturally, the undirected model formed by this procedure does not precisely capture the conditional independence relationships in the directed version. For example, if it is applied to the graph in Fig. 4 (*right*),

then the nodes A, B, and C form a clique in the resulting model, which does not capture the fact that $A \perp\!\!\!\perp B$. However, we note that every term of the form $p(x_i|x_{pa_i})$ appears in some clique of the undirected model, meaning that it can include all of the factors implied by the Bayesian Network.

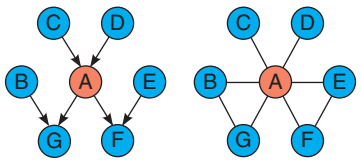
Characterization of Directed and Undirected Graphical Models

We can now present some theorems that characterize both Bayesian Networks and Markov random fields:

Lemma 7 (Local Markov Property) *A node in a DAG is conditionally independent of its non-descendants, given its parents (this is referred to as the "Directed" Local Markov Property); a node in a UG is conditionally independent of its non-neighbors, given its neighbors.*

Definition 8 (Markov Blanket) *Given a node A, its "Markov Blanket" is the minimal set of nodes C such that* $A \perp\!\!\!\perp B \mid C$ *for all other nodes B in the model (in other words, the minimal set of nodes that we must know to "predict" the behavior of A).*

Lemma 9 (Markov Blankets of Directed and Undirected Graphs) *In a directed network, the Markov Blanket of a node A (denoted* $MB(A)$*) consists of its parents, its children, and its children's (other) parents. In an undirected network, it simply consists of the node's neighbors (see Fig. 5).*



Graphical Models. Figure 5. The Markov Blanket of the node *A* consists of its parents, its children, and the parents of its children (*left*). The corresponding structure for *undirected* models simply consists of the neighbors of *A*. Note that if we convert the directed model to an undirected one (using the procedure described in Section "Conversion from directed to undirected graphical models"), then the Markov Blankets of the two graphs are identical

Definition 10 (d-separation) *The notion of a Markov Blanket can be generalized to the notion of "d-separation". A set of nodes A is said to be d-separated from a set B by a set C if every (undirected) path between A and B is "blocked" when C is in the conditioning set (i.e., when C is observed). A path is said to be blocked if* ***either*** *it contains* (p_1, p_2, p_3) *with* $p_1 \rightarrow p_2 \leftarrow p_3$ *(where arrows indicate edge directions) and neither* p_2 *nor any of its descendants are observed,* ***or*** *it contains* (p_1, p_2, p_3) *with* $p_1 \rightarrow p_2 \rightarrow p_3$ *and* p_2 *is observed* ***or*** *it contains* (p_1, p_2, p_3) *with* $p_1 \leftarrow p_2 \rightarrow p_3$ *and* p_2 *is observed.*

Applying (Definition 10) to the directed graphs in Fig. 1, we would say that the aqua regions (X_C) *d-separate* the red regions (X_A) from the white regions (X_B); *all conditional independence statements can simply be interpreted as d-separation in a DAG.*

The analogous notion of *graph separation* for Markov random fields is simpler than that of d-separation for Bayesian Networks. Given an undirected graph $\mathcal{G}$ and disjoint subsets of nodes A, B, C, if A is only reachable from B via C, this means that A is *separated* from B by C and these semantics encode the probabilistic fact that $A \perp\!\!\!\perp B \mid C$. This is illustrated in Fig. 6.

In both the directed and undirected case, A Markov Blanket of a node is simply the minimal set of nodes that d-separates/graph separates that node from all others.

A complete characterization of the class of probability distributions represented by Bayesian Networks can be obtained naturally once conditional independence statements are mapped to d-separation statements in a DAG. The following theorem settles this characterization.

Theorem 11 *Let p be a probability distribution that satisfies the conditional independence statements implied by d-separation in a DAG. Then p factors according to (Eq. 6). The converse also holds.*

For Markov random fields, an analogous characterization exists:

Theorem 12 (Hammersley-Clifford) *If a strictly positive probability distribution p satisfies the conditional independence statements implied by graph-separation in*

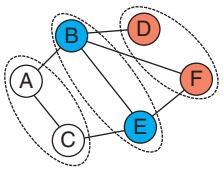
an undirected graph $\mathcal{G}$, then

$$p(x) = \frac{1}{Z}\prod_{c\in\mathcal{C}}\psi_c(x_c). \tag{7}$$

The converse also holds, albeit in a more general sense in that p need not be strictly positive.

It can be shown that

directed local Markov property		*local Markov property*
⇕		⇕
d-separation in a DAG	and (for positive p) that	*graph separation in a UG*
⇕		⇕
factorization of p by (Eq. 6)		*factorization of p by* (Eq. 7)



Graphical Models. Figure 6. The nodes $\{B, E\}$ form a *clique*; the nodes $\{B, E, F\}$ form a *maximal clique*. The nodes $\{B, E\}$ *separate* the nodes $\{A, C\}$ from $\{D, F\}$

Knowing that directed models can be converted to undirected models, we shall consider inference algorithms in undirected models only.

Applications

Inference Algorithms in Graphical Models

The key observation that we shall rely on in order to do inference efficiently is the *distributive law*:

$$\underbrace{ab + ac}_{\text{three operations}} = \underbrace{a(b + c)}_{\text{two operations}}. \tag{8}$$

By exploiting the factorization in a graphical model, we can use this law to perform certain queries efficiently (such as computing the marginal with respect to a certain variable).

As an example, suppose we wish to compute the marginal $p(x_1)$ in an MRF with the following factorization:

$$p(x) = \frac{1}{Z}\prod_{i=1}^{N-1}\psi(x_i, x_{i+1}). \tag{9}$$

Note that the graph representing this model is simply a *chain*. Computing the sum in the naïve way requires computing

$$p(x_1) = \frac{1}{Z}\sum_{x_{\{2\ldots N\}}}\prod_{i=1}^{N-1}\psi(x_i, x_{i+1}), \tag{10}$$

whose complexity is $\Theta(\prod_{i=1}^{N}|\mathcal{X}_i|)$. However, due to the distributive law, the same result is simply

$$p(x_1) = \frac{1}{Z}\sum_{x_2}\Big[\psi(x_1, x_2)\sum_{x_3}\Big[\psi(x_2, x_3)\cdots \sum_{x_{N-1}}\Big[\psi(x_{N-2}, x_{N-1})\sum_{x_N}\psi(x_{N-1}, x_N)\Big]\Big]\Big], \tag{11}$$

whose complexity is $\Theta(\sum_{i=1}^{N-1}|\mathcal{X}_i||\mathcal{X}_{i+1}|)$. As a more involved example, consider computing the marginal with respect to A in the undirected model in Fig. 2; here we wish to compute

$$p(a) = \frac{1}{Z}\sum_{b,c,d,e,f}\psi(a, b)\psi(a, c)\psi(b, d)\psi(c, e)\psi(b, e, f) \tag{12}$$

$$= \frac{1}{Z}\sum_{b}\psi(a, b)\sum_{c}\psi(a, c)\sum_{d}\psi(b, d)\sum_{e}\psi(c, e)\sum_{f}\psi(b, e, f). \tag{13}$$

Exploiting the distributive law in this way is often referred to as the *Elimination Algorithm*. It is useful for computing the marginal with respect to a single variable. However, should we wish to compute the marginal with respect to *each* variable (for example), it is not an efficient algorithm as several operations shall be repeated.

Belief-Propagation In tree-structured models, the elimination algorithm can be adapted to avoid repeated

computations, using a message-passing scheme known as *Belief Propagation*, or the *sum-product* algorithm. This is presented in Algorithm 1. Here the "cliques" in the model are simply edges. This algorithm was invented independently by many authors, and is the most efficient amongst many variations.

It can be easily demonstrated that the condition in Algorithm 1, Line 3 is always satisfied by some pair of edges until all messages have been passed: initially, it is satisfied by all of the "leaves" of the model; messages are then propagated inwards until they reach the "root" of the tree; they are then propagated outwards.

Algorithm 1 The *sum-product* algorithm

Input: an undirected, tree-structured graphical model $\mathcal{X}$ with cliques $\mathcal{C}$ {the cliques are simply *edges* in this case}

1: define $m_{A\to B}(x_{A\cap B})$ to be the "message" from an edge A to an adjacent edge B {for example if $A = (a,b)$ and $B = (b,c)$ then we have $m_{(a,b)\to(b,c)}(x_b)$}
2: **while** there exist adjacent edges $A, B \in \mathcal{C}$ for which $m_{A\to B}$ has not been computed **do**
3: find some $A \in \mathcal{C}$ such that $m_{C\to A}$ has been computed for every neighbor $C \in \Gamma(A)$, *except* B {$\Gamma(A)$ returns the edges neighboring A; initially the condition is satisfied by all leaf-edges}
4: $m_{A\to B}(x_{A\cap B}) := \sum_{x_{A\setminus B}} \left\{\psi_A(x_A) \prod_{C\in\Gamma(A)\setminus B} m_{C\to A}(x_{A\cap C})\right\}$
5: **end while**
6: **for** $A \in \mathcal{C}'$ **do**
7: $marginal_A(x_A) := \psi_A(x_A) \prod_{C\in\Gamma(A)} m_{C\to A}(x_{A\cap C})$
8: **end for**

Maximum A Posteriori (MAP) Estimation Algorithm 1 allows us to compute the *marginals* of the variables in a graphical model. There are other related properties that we may also wish to compute, such as finding which states have the highest probability (the *Maximum A Posteriori*, or simply "MAP" states). To do so, we note that the operations $(+,\times)$ used in Algorithm 1 can be replaced by $(\max,\times)$. This variant is usually referred to as the *max-product* (as opposed to *sum-product*) algorithm. Indeed, different quantities can be computed by replacing $(+,\times)$ by any pair of operations that form a *semiring* (Aji & McEliece, 2001).

The Junction-Tree Algorithm Algorithm 1 applies only for tree-structured graphs. We can generalize this algorithm to general graphs. We do so by working with a different type of tree-structured graph, whose *nodes* contain the *cliques* in our original graph. We begin with some definitions:

Definition 13 (chordal graph) *A graph $\mathcal{G}$ is said to be chordal if every cycle $(c_1 \ldots c_n)$ in $\mathcal{G}$ contains a chord (i.e., an edge (c_i, c_j) such that $j > (i+1)$).*

Definition 14 (clique-graph, clique-tree) *A clique-graph $\mathcal{H}$ of a graph $\mathcal{G}$ is a graph whose nodes consist of (maximal) cliques in $\mathcal{G}$, and whose edges correspond to intersecting cliques in $\mathcal{G}$. A clique-tree is a clique-graph without cycles.*

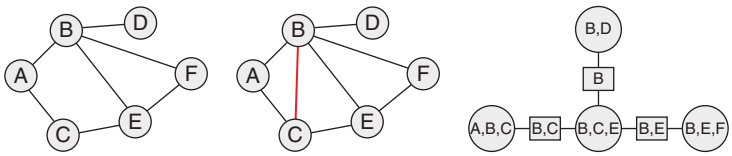
Definition 15 (Junction-Tree) *A clique-tree $\mathcal{H}$ of $\mathcal{G}$ is said to form a Junction-Tree if for every pair of nodes A, B (i.e., maximal cliques in $\mathcal{G}$), the path between them $(P_1 \ldots P_m)$ satisfies $(A \cap B) \subset P_i$ for all $i \in \{1 \ldots m\}$.*

The algorithms we shall define apply only if the graph in question is *chordal*, or "triangulated" (Definition 13); this can always be achieved by adding additional edges to the graph, as demonstrated in Fig. 7; adding additional edges means increasing the size of the maximal cliques in the graph.

Finding the "optimal" triangulation (i.e., the one that minimizes the size of the maximal cliques) is an NP-complete problem. In practice, triangulation algorithms vary from simple greedy heuristics (e.g., select a node that has as few neighbors as possible), to complex approximation algorithms working within a factor of the optimal solution (Amir, 2001).

The problem of actually *generating* a Junction-Tree from the triangulated graph is easily solved by a maximum spanning tree algorithm (where we prefer edges corresponding to pairs of cliques with large intersections).

Theorem 16 *Let $\mathcal{G}$ be a triangulated graph and $\mathcal{H}$ a corresponding clique-tree. If the sum of the cardinalities*

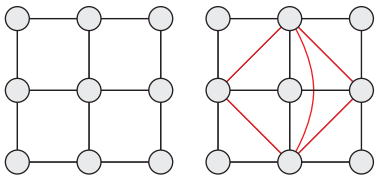


Graphical Models. Figure 7. **The graph at *left* is not chordal, since the cycle (A, B, E, C) does not contain a chord; adding the edge (B, C) results in a chordal (or triangulated) graph (*centre*). The graph at *right* is a Junction-Tree for the graph at *centre*; the cliques of the triangulated graph form the nodes (circles); their *intersection sets* are shown as squares. Note that this is not the only Junction-Tree that we could form – the node $\{B, D\}$ could connect to any of the other three nodes**

of the intersection sets of $\mathcal{H}$ is maximum, then $\mathcal{H}$ is a Junction Tree. The converse also holds.

If the nodes and edges in Algorithm 1 are replaced by the nodes (maximal cliques in $\mathcal{G}$) and edges (intersecting cliques in $\mathcal{G}$) of $\mathcal{H}$, then we recover the *Junction-Tree Algorithm*.

Approximate Inference The act of triangulating the graph in the Junction-Tree Algorithm may have the effect of increasing the size of its maximal cliques, as in Fig. 8. This may be a problem, as its running time is exponential in the size of the maximal cliques in the *triangulated* graph (this size minus one is referred to as the *tree-width* of the graph, e.g., a chain has a tree-width of 1).



Graphical Models. Figure 8. **The graph above at *left* has maximal cliques of size two; in order to triangulate it, we must introduce maximal cliques of size four (*right*)**

There are a variety of approximate algorithms that allow us to perform inference more efficiently:

Variational approximation. If doing inference in a graphical model $\mathcal{X}$ is intractable, we might search for a model $\mathcal{Y}$ for which inference is tractable, and which is "similar" to $\mathcal{X}$ in terms of the KL-divergence between $p(x)$ and $p(y)$. (Wainwright & Jordan, 2008).

Loopy belief-propagation. We can build a clique-graph from a graph that has not been triangulated, simply by connecting all cliques that intersect (in which case, the clique-graph will contain loops). If we then propagate messages in some *random* order, we can obtain good approximations under certain conditions (Ihler et al., 2005).

Gibbs sampling. Given an estimate $x_{A \setminus B}$ of a set of variables $X_{A \setminus B}$, we can obtain an estimate of x_B by sampling from the conditional distribution $p(x_B | x_{A \setminus B})$. If we choose $B = \{X_i\}$, and repeat the procedure for random choices of $i \in \{1 \ldots N\}$, we obtain the procedure known as *Gibbs Sampling* (Geman, S. & Geman, D., 1984).

There are several excellent books and tutorial papers on graphical models. A selection of tutorial papers includes Aji and McEliece (2001), Kschischang, Frey, and Loeliger (2001), Murphy (1998), Wainwright and Jordan (2008); review articles include Roweis and Ghahramani (1997) and Smyth (1998), to name but a few.

A selection of works includes Koller and Friedman (2009), Jensen (2001) (introductory books), Edwards (2000) (undirected models), Pearl (1988, 2000) (directed models), Cowell, Dawid, Lauritzen, and Spiegelhalter (2003) (exact inference), Jordan (1998) (learning and approximate inference) and Lauritzen (1996, Lauritzen and Spiegelhalter (1988) (a comprehensive mathematical theory).

There is also a variety of closely related models and extensions:

Gaussian graphical models. We have assumed throughout that our probability distributions are *discrete*; however, the only condition we require is that

they are *closed under multiplication and marginalization*. This property is also satisfied for *Gaussian* random variables.

Hidden Markov models. In many applications, the variables in our model may be *hidden*. The above algorithms can be adapted to infer properties about our hidden states, given a sequence of observations.

Kalman filters. Kalman filters employ both of the above ideas, in that they include hidden state variables taking values from a *continuous* space using a Gaussian noise model. They are used to estimate the states of *linear dynamic systems* under noise.

Factor graphs. Factor graphs employ an alternate message-passing scheme, which may be preferable for computational reasons. Inference remains approximate in graphs with loops, though approximate solutions may be obtained more efficiently than by Loopy Belief-Propagation (Kschischang et al., 2001).

Relational models. Relational models allow us to explore the relationships between objects in order to predict the behavior and properties of each. Graphical models are used to predict the properties of an object based on others that relate to it (Getoor & Taskar, 2007).

Learning. Often, we would like to *learn* either the parameters or the structure of the model from (possibly incomplete) data. There is an extensive variety of approaches; a collection of papers appears in Jordan (1998).

Cross References

►Expectation Propogation
►Hidden Markov Models
►Probabilistic Relational Models

Recommended Reading

Aji, S. M., & McEliece, R. J. (2000). The generalized distributive law. *IEEE transactions on information theory, 46*(2):325-343.

Amir, E. (2001). Efficient approximation for triangulation of minimum treewidth. In *Proceedings of the 17th conference on uncertainty in artificial intelligence* (pp. 7–15). San Francisco: Morgan Kaufmann.

Cowell, R. G., Dawid, P. A., Lauritzen, S. L., & Spiegelhalter, D. J. (2003). *Probabilistic networks and expert systems*. Berlin: Springer.

Edwards, D. (2000). *Introduction to graphical modelling*. New York: Springer.

Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions and the bayesian restoration of images. In *IEEE transactions on pattern analysis and machine intelligence, 6*, 721–741.

Getoor, L., & Taskar, B. (Eds.). (2007). *An introduction to statistical relational learning*. Cambridge, MA: MIT Press.

Ihler, A. T., Fischer III, J. W., & Willsky, A. S. (2005). Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research, 6*, 905–936.

Jensen, F. V. (2001). *Bayesian networks and decision graphs*. Berlin: Springer.

Jordan, M. (Ed.). (1998). *Learning in graphical models*. Cambridge, MA: MIT Press.

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

Kschischang, F. R., Frey, B. J., & Loeliger, H. A. (2001). Factor graphs and the sum-product algorithm. *IEEE transactions on information theory, 47*(2), 498–519.

Lauritzen, S. L. (1996). *Graphical models*. Oxford: Oxford University Press.

Lauritzen, S. L., & Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B, 50*, 157–224.

Murphy, K. (1998). *A brief introduction to graphical models and Bayesian networks*. San Francisco: Morgan Kaufmann.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. San Francisco: Morgan Kaufmann.

Pearl, J. (2000). *Causality*. Cambridge: Cambridge University Press.

Roweis, S., & Ghahramani, Z. (1997). A unifying review of linear Gaussian models. *Neural Computation, 11*, 305–345.

Smyth, P. (1998). Belief networks, hidden Markov models, and Markov random fields: A unifying view. *Pattern Recognition Letters, 18*, 1261–1268.

Wainwright, M. J., & Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning, 1*, 1–305.