7.1 Graphical models and belief propagation

Outline

graphical models
Bayesian networks
pair wise Markov random fields
factor graphs
versions of problem
    MAP
    marginalization
    addition version, multiplication version
tree algorithm
message passing
single loop
clustering
maximum weight matching
relationship to physics

Terminology

marginalization compute $f(x) = \sum_{x_2,\cdots,x_n} f(x)$

maimum posteriori probability problem (MAP) find value of x that maximizes the function f(x).

belief propagation (belief update) algorithm for summing over variables (marginalization)

belief revision algorithm to obtain maximum posteriori probability

factor graph bipartite graph relating variables and factors, sometimes call Markov random field

belief marginal probability

Ising model

Gibbs measure

Table of terminology for belief propagation to help keep terminology clear.

7.1 Graphical models and belief propagation

A graphical model is a graph, directed or undirected, whose vertices correspond to variables that take on values from some finite set. The edges of the graph represent relationships or constraints between the variables. The graphical model is often used as a compact representation of a joint probability distribution over the variables.

In the directed model, which tends to appear in the AI and statistics literature, the joint probability of the variables is expressed as a product of conditional probabilities.

$$p(x_1, x_2, \cdots, x_n) = \prod_{i=1}^{n} p(x_i \mid \text{parents of } x_i)$$

The directed graphical model is called a Bayesian or belief network.

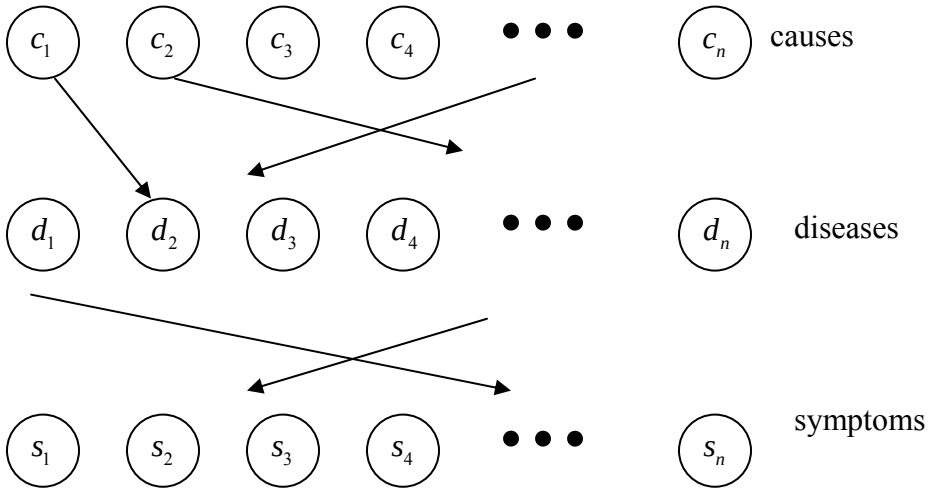
In the undirected model, which appears frequently in the vision and the statistical mechanics literature, the vertices correspond to variables and the edges correspond to constraints between pairs of variables. The undirected version, called a pair wise Markov field, is used to represent a joint probability distribution or often just a combinatorial optimization problem.

Bayesian networks

A Bayesian network is a directed graph where vertices correspond to variables that range over a discrete set of values and a directed edge from *y* to *x* represents a conditional probability p(x/y). If a vertex x has edges into it from $y_1, y_2, \cdots, y_k$, then the conditional probability is $p(x / y_1, y_2, \cdots, y_k)$. The variable at a vertex with no in edges has an unconditional probability distribution. If the value of a variable at some vertex is known, then the variable is called evidence. An important property of a Bayesian network is that a joint probability often can be expressed as the product of a number of conditional probabilities each involving only a few variables.

7.1 Graphical models and belief propagation

Example: A Bayesian network



$$p(c_1, c_2, d_1, d_2, s_1, s_2) = p(c_1)\, p(c_2)\, p(d_1 / c_1.c_2)\, p(d_2 / c_2)\, p(s_1 / d_1) P(s_2 / d_1, d_2)$$

■

In the above example, a patient is ill and sees a doctor. The doctor ascertains the symptoms of the patient and the possible causes such as whether the patient was in contact with farm animals, whether he had eaten certain foods, or whether the patient has an hereditary predisposition to any diseases. Using the above Bayesian network where the variables are true or false, the doctor may wish to determine one of two things. What is the marginal probability of a given disease or what is the most likely set of diseases. In determining the most likely set of diseases, we are given a T or F assignment to the causes and symptoms and ask what assignment of T or F to the diseases maximizes the joint probability. This latter problem is called the maximum a posteriori probability (MAP).

Given the conditional probabilities in the above example and the probabilities $p(c_1)$, $p(c_2), \cdots, p(c_n)$, the joint probability $p(c_1, c_2, \cdots, d_1, d_2 \cdots)$ can be computed easily for any combination of values of $c_1, c_2, \cdots, d_1, d_2, \cdots$. However, the obvious algorithm for computing a marginal probability, $p(d_i)$, requires summing $p(c_1, c_2, \cdots, d_1, d_2 \cdots)$ over exponentially many values of the other variables and the obvious algorithm for the maximum posteriori probability requires evaluating the probability $p(c_1, c_2, \cdots, d_1, d_2 \cdots)$ over exponentially many input values.

7.1 Graphical models and belief propagation

Pair wise Markov Random Fields

The pair wise Markov random field consists of an undirected graph where the vertices are associated with variables that take on values from some finite set. Associated with the edges are functions of the variables at the end points of the edge. Sometime the model is used to represent a joint probability distribution. Other times it is used as a representation for a combinatorial optimization problem.

When using this model to represent a joint probability distribution, it is often the case that some of the variables are known quantities, y_i, and some are unknown quantities, x_i, and corresponding x_i and y_i are related by a function $\phi_i(x_i, y_i)$. Since values for the y_i are known, the function $\phi_i(x_i, y_i)$ is written $\phi_i(x_i)$ and is called the evidence for x_i. In addition, certain pairs of x_i are related by a compatibility function $\Psi_i(x_i, x_j)$. The joint probability function is

$$p(x) = \prod_{ij} \psi(x_i, x_j) \prod_i \phi(x_i)$$

We want to compute marginal probabilities at each node x_i, $b(x_i) = \sum_{\sim x_i} p(x)$, called the *belief*. Since p(x) is expressed as a product of factors, we shall see that in some situations the belief propagation algorithm calculates the marginal probability efficiently.

One application of this model is in image enhancement. The y_i are observed pixel values and the x_i are the unknown true values. The functions $\Psi_i(x_i, x_j)$ may be constraints on the true values of adjacent pixels.

An alternative view of the pair wise Markov random field is as a combinatorial optimization problem. Here the goal is to find the assignment to the variables so as to maximize the function $\prod_{ij} \Psi(x_i, x_j)$. Maximizing the function is the same as maximizing the logarithm of the function and this leads to a summation version of the problem where we wish to maximize $\sum_{ij} \log(\Psi(x_i, x_j))$.

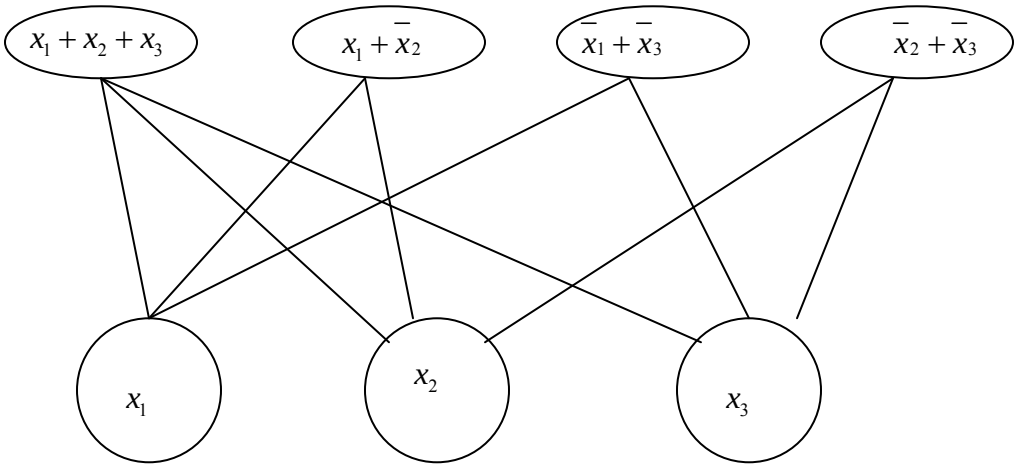
7.1 Graphical models and belief propagation

Factor graphs

Factor graphs arise when we have a function f of a variable $x = (x_1, x_2, \cdots, x_n)$ where f can be expressed as $f(x) = \prod_{\alpha} f_\alpha (x_\alpha)$ where each factor depends only on some small number of variables x_α. Associate a bipartite graph with the product of factors as follows. One set of vertices correspond to the factors and the other set to the variables. Place an edge between a variable and a factor if the factor contains the variable.

Example: The factor graph for the formula

$$f(x_1, x_2, x_3) = (x_1 + x_2 + x_3)(x_1 + x_2)(x_1 + x_3)(x_2 + x_3)$$



■

Tree algorithms

Let f(x) be a function that is a product of factors. When the factor graph is a tree there are efficient algorithms for solving certain problems. The algorithms we present will also solve the problems when the function is the sum of terms rather then a product of factors.

The first problem is called *marginalization* and involves evaluating the sum of f(x) over all variables except one. In the case where f is a probability distribution the algorithm computes the marginal probabilities and thus the word marginalization.

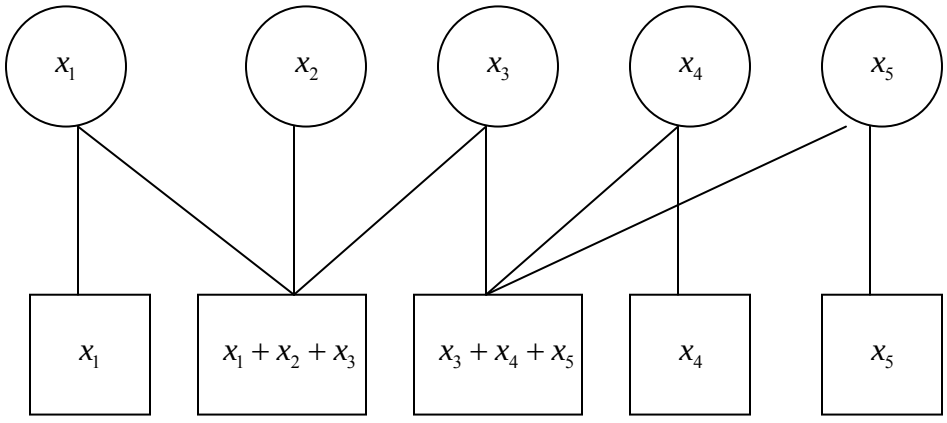
The second problem involves computing the assignment to the variables that maximizes the function f. When f is a probability distribution this problem is the maximum a posteriori probability or MAP problem.

If the factor graph is a tree then there exists an efficient algorithm for solving these problems. Note that there are four problems: the function f is either a product or sum and we are either marginalizing or finding the maximizing assignment to the variables. All four problems are solved by essentially the same algorithm and we present the algorithm for the marginalization problem where f is a product.

Consider the following example where

$$f = x_1\left(x_1 + x_2 + x_3\right)\left(x_3 + x_4 + x_5\right)x_4x_5$$

and the variables take on values 0 or 1. In this case the factor graph is a tree as shown below.

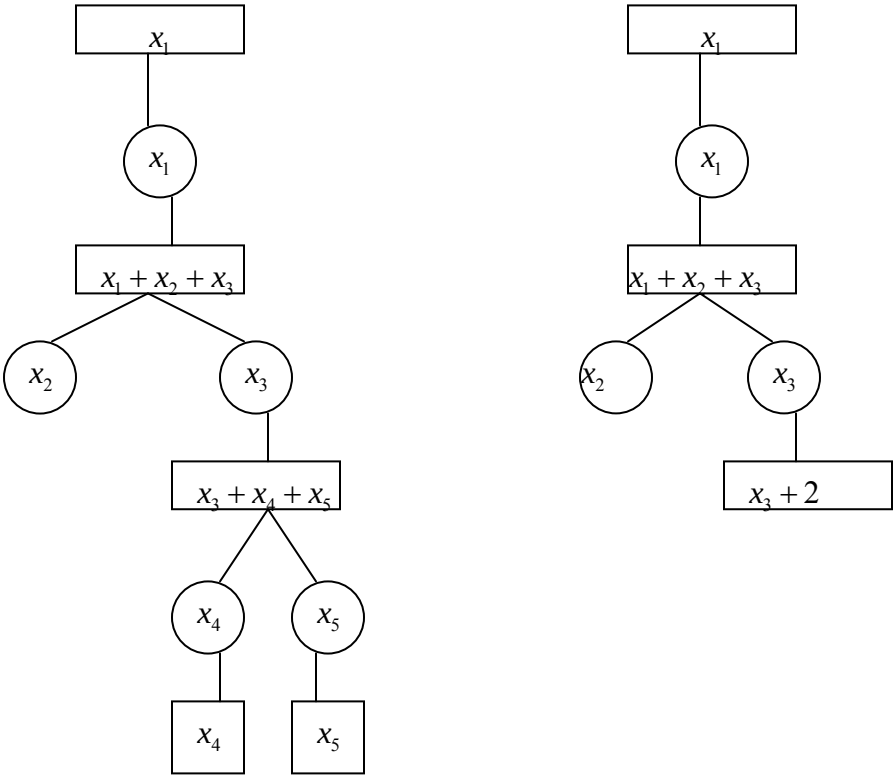


Consider marginalizing f and computing

$$f\left(x_1\right) = \sum_{x_2x_3x_4x_5} x_1\left(x_1 + x_2 + x_3\right)\left(x_3 + x_4 + x_5\right)x_4x_5\ ,$$

One could move the summation over x_4 and x_5 in past the first two terms of the product and then sum over all values of x_4 and x_5. Summing over x_4 and x_5 replaces the product of factors $(x_3 + x_4 + x_5)x_4x_5$ by a single factor $x_3 + 2$ in the formula and gives a new factor graph with the sub tree below the vertex 3 node replaced by a single factor node corresponding to $x_3 + 2$ as shown. This corresponds to the following computation.

$$\sum_{x_2x_3x_4x_5} x_1(x_1 + x_2 + x_3)(x_3 + x_4x_5)x_4x_5 = \sum_{x_2x_3} x_1(x_1 + x_2 + x_3)\sum_{x_4x_5}(x_3 + x_4 + x_5)x_4x_5$$
$$= \sum_{x_2x_3} x_1(x_1 + x_2 + x_3)(x_3 + 2)$$



Instead of modifying the tree we could instead simply send a message $x_3 + 2$ from vertex x_3 to the factor node $x_1 + x_2 + x_3$ indicating that eliminating all variables in the sub tree below x_3 would result in a factor $x_3 + 2$. One advantage to passing messages rather then modifying the tree is that when we want to compute all marginal's $f_i(x_i)$. Then summing over all variables in a sub tree need only be done once and the result used in all marginal's requiring it.

The rules for calculating the messages are as follows. At each leaf node that is a variable, sum the variable over all of its possible values and send the sum up the edge to its parent.

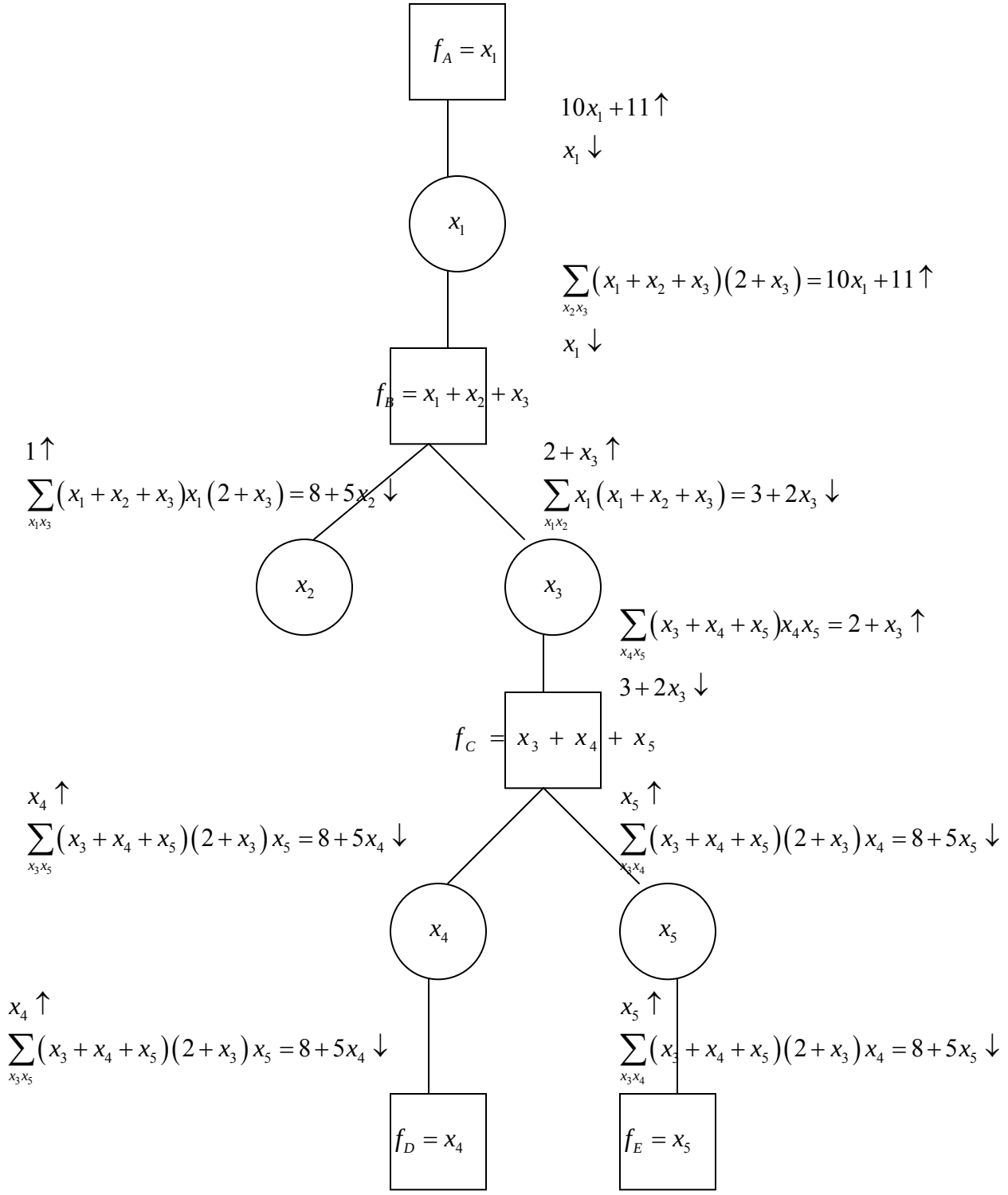
In this example, the message sent is 1 since the variables take on the values 0 and 1. For each leaf node that is a factor, send the one variable function associated with the node up to the parent. For interior nodes, wait until all edges but one have received a message. Then, for an interior variable node compute the product of all incoming messages and sum this product function over all assignments to the variables except for the variable of the node. Send the resulting function of one variable out along the remaining edge. For interior factor nodes, compute the product of the factor function along with incoming messages on all edges except one and send the resulting function out along the remaining edge. The message from a variable node is the value of the sub tree below the node when all variables except for the variable at the node have been summed out.

The calculation is shown on the next page. A check on the value calculated for g_2 is given below.

$$
\begin{aligned}
g_1(x_1) &= \sum_{x_1 x_3 x_4 x_5} x_1 (x_1 + x_2 + x_3)(x_3 + x_4 + x_5) x_4 x_5 \\
&= \sum_{x_3 x_4 x_5} (x_1 + x_2 + x_3)(x_3 + x_4 + x_5) x_4 x_5 \\
&= \sum_{x_3 x_4} (x_1 + x_2 + x_3)(1 + x_3 + x_4) x_4 \\
&= \sum_{x_3} (x_1 + x_2 + x_3)(2 + x_3) \\
&= 2(1 + x_2) + 3(2 + x_2) = 2 + 2x_2 + 6 + 3x_2 = 8 + 5x_2
\end{aligned}
$$

If instead of computing marginal's, we wanted the variable assignment that maximizes the function f, we would modify the above procedure by replacing the summation by a maximization operation. Obvious modifications handle the situation where f(x) is a sum of products.

7.1 Graphical models and belief propagation



The normalizing constant for a probability distribution

When a probability is expressed as a product of factors

$$p(x_1, x_2, \cdots, x_n) = \frac{1}{Z} \sum_{\alpha} f_\alpha (x_\alpha)$$

there is often a normalizing constant z. To evaluating this constant requires that $\sum_{\alpha} f_\alpha (x_\alpha)$ be summed over the exponentially many values in the domain of x a case of marginalization over all variables.

normalizing messages.

In a product of factors formula, we marginalize the product by summing over variables. Analogous to the product of factors is the maximization of a sum where we sum incoming messages and then marginalize by taking the maximum over certain variables. The message from a variable node x to a function node is a linear form ax+b which tells the value up to this point in a tree for x=0 and x=1. At the function node we add incoming messages along with the function and maximize over all variables except the target variable. Note that the value is a linear function of the target variable and depends on the linear functions ax+b. However, the value of the marginalized variables do not depend on the value of *b*, only the actual value of the maximum depends on *b*. Thus, if we only are interested in the value of the variables that maximize the function and not the value of the maximum, we can ignore constant terms.

Note that in a non tree graph if a function node that was in a cycle had a constant term, going around the cycle would repeatedly add the constant and hence prevent convergence. Some type of normalization would be necessary.