

# Bayesian Networks

## Factor Graphs

### the Case-Factor Algorithm

### and the Junction Tree Algorithm

## 1 Bayesian Networks

We will use capital letters for random variables and lower case letters for values of those variables. A Bayesian network is a triple  $\langle V, G, \mathcal{P} \rangle$  where  $V$  is a set of random variables  $X_1, \dots, X_n$ ,  $G$  is a directed acyclic graph (DAG) whose nodes are the variables in  $V$ , and  $\mathcal{P}$  is a set of conditional probability tables as described below. The conditional probability tables determine a probability distribution over the values of the variables. If there is a directed edge in  $G$  from  $X_j$  to  $X_i$  then we will say that  $X_j$  is a parent of  $X_i$  and  $X_i$  is a child of  $X_j$ . To determine values for the variables one first selects values for variables that have no parents and then repeatedly picks a value for any node all of whose parents already have values. When we pick a value for a variable we look only at the values of the parents of that variable. We will write  $P(x_i \mid \text{parents of } x_i)$  to abbreviate  $P(x \mid x_{i_1}, \dots, x_{i_k})$  where  $x_{i_1}, \dots, x_{i_k}$  are the parents of  $x$ . For example, if  $x_7$  has parents  $x_2$  and  $x_4$  then  $P(x_7 \mid \text{parents of } x_7)$  abbreviates  $P(x_7 \mid x_2, x_4)$ . Formally, the probability distribution on the variables of a Bayesian network is determined by the following equation.

$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i \mid \text{parents of } x_i) \quad (1)$$

The conditional probabilities of the form  $P(x_i \mid \text{parents of } x_i)$  are called conditional probability tables (CPTs). Suppose that each of the variables  $x_i$  has  $d$  possible values (this is not required in general). In this case if a variable  $x$  has  $k$  parents then  $P(x \mid \text{parents of } x)$  has  $d^{k+1}$  values (with  $(d-1)d^k$  degrees of freedom). These  $d^{k+1}$  values can be stored in a table with  $k+1$  indices. Hence the term “table”. Note that the number of indices of the CPTs (conditional probability tables) is different for the different variables.

Note that an HMM is a Bayesian network with a variable for each hidden state and each observable token.

Bayesian networks are often used in medical diagnosis where variables represent the presence or absence of certain diseases or the presence or absence of certain measurable quantities such as blood sugar or presence of a certain protein in the blood.

In a Bayesian network the edges of the directed graph are often interpreted as “causation” with the parent causally influencing the child and parents getting assigned values temporally before children.

We are interested in “Bayesian inference” which means, intuitively, inferring causes by observing their effects using Bayes’ rule. In an HMM for example, we want to infer the hidden states from the observations that they cause.

In general we can formulate the inference problem as the problem of determining the probability of an unknown variable (a hidden cause) from observed values of other variables. In general we can consider the variables in any order.

$$P(x_5, x_7 \mid x_3, x_2) = \frac{P(x_5, x_7, x_2, x_3)}{P(x_2, x_3)} \quad (2)$$

So in general, for inference it suffices to be able to compute probabilities of the form  $P(x_{i_1}, \dots, x_{i_k})$ . We give an algorithm for doing this in section 5.

## 2 Inference in Bayesian Network is #P hard

A Boolean variable (also called Bernoulli variable) is a variable that has only the two possible values of 0 or 1. A disjunctive clause is a disjunction of literals where each literal is either a Boolean variable or the negation of a Boolean variable. For example we have that  $(X_5 \vee \neg X_2 \vee X_3)$  is a clause with three literals. A 3SAT problem is a set of clauses with three literals in each clause. It is hard (in fact #P hard) to take a 3SAT problem and determine the number of complete assignments of values to variables that satisfy the clauses, i.e., that make every clause true.

Take  $X_1, \dots, X_n$  be independent Boolean (Bernoulli) variables with  $P(X_i = 1) = 1/2$ . Let  $\Sigma$  be a set of clauses over these variables where each clause has only three literals. Let  $C_j$  be a random variable which is 1 if the  $j$ th clause is satisfied. Since we can compute  $c_j$  from  $x_1, \dots, x_n$  using a (deterministic) conditional probability table having only three parents. Let  $A_j$  be a Boolean variable that is true if all of  $C_1, \dots, C_j$  are true.  $a_1$  can be computed with a CPT from  $c_1$  and for  $j > 1$  we have that  $a_j$  can be computed using a CPT from  $a_{j-1}$  and  $c_j$ .

Now we have that  $P(A_k = 1)$  is proportional to the number of truth assignments that satisfying all the clauses. This implies that computing the probability of a partial assignment in a Bayesian network is #P hard. It is widely believed that there are no polynomial time algorithms for #P hard problems.

### 3 Value Assignments

First we introduce some formal notation which will allow certain equation (1) to be written more precisely. We let  $V$  be a finite set of random variables (the nodes of the network) where for  $X \in V$  we let  $\mathcal{D}(X)$  be the set of values that variable  $X$  can have (the support of the variable  $X$ ). We let  $\sigma$  range over assignments of values to the variables — for  $X \in V$  we let  $\sigma(X)$  denote the value that  $\sigma$  assigns to the variable  $X$ . We require  $\sigma(X) \in \mathcal{D}(X)$ . For an assignment  $\sigma$ , and a subset  $E$  of  $V$ , we write  $\sigma|_E$  to be  $\sigma$  restricted to the variables in  $E$  so that  $\sigma|_E$  is an assignment only to the variables in  $E$ .

Now consider a Bayesian network as defined in section 1. For  $X \in V$  let  $\mathcal{P}(X)$  be the set of parents of the variable  $X$  —  $\mathcal{P}(X)$  is the set of  $Y$  such that there is a directed arc in the Bayesian network from  $Y$  to  $X$ . Equation (1) can now be written as follows where  $F_X$  is a the function on assignments to the set  $\{X\} \cup \mathcal{P}(X)$  giving the conditional probability in (3).

$$P(\sigma) = \prod_{X \in V} P(X = \sigma(x) \mid \sigma|_{\mathcal{P}(X)}) \quad (3)$$

$$= \prod_{X \in V} F_X(\sigma|_{\{X\} \cup \mathcal{P}(X)}) \quad (4)$$

We now introduce notation which allows a more general and preciser version of equation (2). We let  $\rho$  range over partial assignments of values to variables —  $\rho$  assigns values to *some* of the variables in  $V$ . We can implement  $\rho$  as a finite list  $\langle\langle X_1, x_1 \rangle\rangle, \dots, \langle\langle X_K, x_K \rangle\rangle$  where  $X_k \in V$  and  $x_k \in \mathcal{D}(X_k)$ . We say that a partial assignment  $\rho'$  is an extension of  $\rho$ , written  $\rho' \sqsubseteq \rho$ , if  $\rho'$  assigns a value to every variable that  $\rho$  assigns a value to and every variable where  $\rho$  assigns a value,  $\rho'$  assigns the same value. We can think of a partial assignment  $\rho$  as representing a statement about, or constraint on, total assignments. We then have that the probability of partial assignment satisfies the following where  $\sigma$  ranges over total assignments.

$$P(\rho) = \sum_{\sigma \sqsubseteq \rho} P(\sigma) \quad (5)$$

The notation of partial assignments also allows us to write a general and more precise version of (2) as follows where  $\rho'$  is any extension of a partial assignment  $\rho$ .

$$P(\rho'|\rho) = \frac{P(\rho')}{P(\rho)} \quad \text{for } \rho' \sqsubseteq \rho \quad (6)$$

To compute arbitrary conditional probabilities between partial assignments it suffices to be able to compute probabilities of the form  $P(\rho)$ .

## 4 Factor Graphs

A factor graph is a hypergraph with a factor term associated with each hyperedge. More specifically, a factor graph consists of a set  $V$  of random variables and a tuple  $\langle E_1, \dots, E_k \rangle$  of “hyperedges” where each hyperedge  $E_k$  is a subset of  $V$  and for each hyperedge  $E_k$  there is a factor term  $F_k$  described below. The set  $V$  and the hyperedges  $\langle E_1, \dots, E_k \rangle$  together form a hypergraph.<sup>1</sup> A factor graph determines a probability distribution on total assignments  $\sigma$  as follows.

$$P(\sigma) = \frac{1}{Z} \prod_{k=1}^K F_k(\sigma|_{E_k}) \quad (7)$$

$$Z = \sum_{\sigma} \prod_{k=1}^K F_k(\sigma|_{E_k}) \quad (8)$$

Note that the factor term  $F_k$  is a function on assignments of values to the variables on the hyperedge  $E_k$ . We now have that (4) is a special case of (7) — for the case of Bayesian networks we have that  $Z = 1$ .

For a partial assignment  $\rho$  we define the partition function  $Z(\rho)$  as follows.

$$Z(\rho) = \sum_{\sigma \sqsubseteq \rho} \prod_{k=1}^K F_k(\sigma|_{E_k}) \quad (9)$$

The partition function values  $Z(\rho)$  should be thought of as an unnormalized probabilities. In particular we have the following where  $Z$  is defined by (8).

$$\begin{aligned} P(\rho) &= \sum_{\sigma \sqsubseteq \rho} P(\sigma) \\ &= \sum_{\sigma \sqsubseteq \rho} \frac{1}{Z} \prod_{k=1}^K F_k(\sigma|_{E_k}) \\ &= \frac{1}{Z} \sum_{\sigma \sqsubseteq \rho} \prod_{k=1}^K F_k(\sigma|_{E_k}) \\ &= \frac{Z(\rho)}{Z} \end{aligned} \quad (10)$$

---

<sup>1</sup>Factor graphs are often formulated as bipartite graphs with one partition representing the random variables and the other partition representing the hyperedges (the “factors”). The formulation as a bipartite graph is equivalent to the formulation as a hypergraph.

For  $\rho' \sqsubseteq \rho$  we have the following.

$$\begin{aligned}
P(\rho'|\rho) &= \frac{P(\rho')}{P(\rho)} \\
&= \frac{\binom{Z(\rho')}{Z}}{\binom{Z(\rho)}{Z}} \\
&= \frac{Z(\rho')}{Z(\rho)} \tag{11}
\end{aligned}$$

Equation (11) shows that to compute conditional probabilities between partial assignments it suffices to compute the unnormalized values  $Z(\rho)$ .

## 5 The Case-Factor Algorithm

Consider a fixed case-factor graph with nodes (random variables)  $V$  and with  $K$  factors so that for  $1 \leq k \leq K$  we have the hyperedge  $E_k \subseteq V$  and we have the function  $F_k$  assigning real values scores to assignments of values to the variables in the hyperedge  $E_k$ . We want consider subgraphs of this factor graph. A subgraph will be taken to consist of a subset of nodes  $\mathcal{V} \subseteq V$  and a subset of the factors  $\mathcal{F} \subset \{1, \dots, K\}$  such that for all  $k \in \mathcal{F}$  we have that the hyperedge  $E_k$  is a subset of  $\mathcal{V}$ . The case-factor algorithm is based on computing  $Z$  values for subgraphs. These values have the form  $Z(\rho, \mathcal{V}, \mathcal{F})$  defined as follows where  $\rho$  is a partial assignment to the variables in  $\mathcal{V}$  and  $\sigma$  ranges over total assignments to the variables in  $\mathcal{V}$ .

$$Z(\rho, \mathcal{V}, \mathcal{F}) = \sum_{\sigma \sqsubseteq \rho} \prod_{k \in \mathcal{F}} F_k(\sigma|_{E_k}) \tag{12}$$

To define the case-factor algorithm we use some additional notation. For a partial assignment  $\rho$  we write  $\text{dom}(\rho)$  for the set of variables that are assigned values by  $\rho$ . For  $X \notin \text{dom}(\rho)$  and  $x \in \mathcal{D}(X)$  we write  $\rho[X; = x]$  for the partial assignment that is  $\rho$  extended with the one additional assignment that assigns  $X$  the value  $x$ . We write  $S \setminus U$  for set difference —  $S \setminus U$  is the subset of elements of  $S$  that are not elements of  $U$ . Equation (12) is the definition of  $Z(\rho, \mathcal{V}, \mathcal{F})$  and from this definition we can prove the following two equations which define the case-factor

algorithm.

$$Z(\rho, \mathcal{V}, \mathcal{F}) = \sum_{x \in \mathcal{D}(X)} Z(\rho[X := x], \mathcal{V}, \mathcal{F}) \text{ if } \begin{cases} X \in \mathcal{V} \\ X \notin \text{dom}(\rho) \end{cases} \quad (13)$$

$$Z(\rho, \mathcal{V}, \mathcal{F}) = Z(\rho|_{\mathcal{V}_1}, \mathcal{V}_1, \mathcal{F}_1) Z(\rho|_{\mathcal{V}_2}, \mathcal{V}_2, \mathcal{F}_2) \quad (14)$$

$$\text{if } \begin{cases} E_k \subseteq \mathcal{V}_1 \text{ for } k \in \mathcal{F}_1 \\ E_k \subseteq \mathcal{V}_2 \text{ for } k \in \mathcal{F}_2 \\ (\mathcal{V}_1 \setminus \text{dom}(\rho)) \cap (\mathcal{V}_2 \setminus \text{dom}(\rho)) = \emptyset \\ (\mathcal{V}_1 \setminus \text{dom}(\rho)) \cup (\mathcal{V}_2 \setminus \text{dom}(\rho)) = \mathcal{V} \setminus \text{dom}(\rho) \\ \mathcal{F}_1 \cap \mathcal{F}_2 = \emptyset \\ \mathcal{F}_1 \cup \mathcal{F}_2 = \mathcal{F} \end{cases}$$

$$Z(\rho, \mathcal{V}, \mathcal{F}) = \prod_{k \in \mathcal{F}} F_k(\rho) \text{ if } \text{dom}(\rho) = \mathcal{V} \quad (15)$$

Each of these three equations gives a potential way to make progress in computing  $Z(\rho, \mathcal{V}, \mathcal{F})$ . Equation (13) is the case rule which does a case analysis on the value of one of the variables not in  $\text{dom}(\rho)$ . Equation (14) gives a way to factor the problem into a product of two subproblems. Equation (14) can be used whenever the variables in  $\mathcal{V} \setminus \text{dom}(\rho)$  can be divided into two disjoint sets such that no factor involves variables from both of these sets. To computing the factoring we simply need to compute the connected components of the graph on  $\mathcal{V} \setminus \text{dom}(\rho)$  where two nodes are connected if they occur in the same factor in  $\mathcal{F}$ . To apply a case-factor analysis we first see if the factor rule can be used. If not, we apply the case rule unless all variables in  $\mathcal{V}$  are already in  $\text{dom}(\rho)$  in which case we apply the base case equation (15). In the case-factor algorithm it is important to memoize the values of subcomputations.

## 6 The Junction Tree of the Case-Factor Algorithm

Consider using the junction tree algorithm to compute  $Z(\rho, \mathcal{V}, \mathcal{F})$ . The choice of which rule to apply — case, factor, or the base case — is determined by the three sets  $\text{dom}(\rho)$ ,  $\mathcal{V}$ , and  $\mathcal{F}$  independent of the particular partial assignment  $\rho$ . Therefore one can construct an abstract problem-subproblem graph each node of which is labeled by the three sets  $\text{dom}(\rho)$ ,  $\mathcal{V}$  and  $\mathcal{F}$  and where there is a directed arc from one node to another if either the case or factor rule generates the second node a subproblem the first. Because the factor rule generates subproblems with

disjoint sets of unassigned variables, this graph forms a tree. The junction tree generated by the case-factor algorithm is the tree that results from erasing the sets  $\mathcal{V}$  and  $\mathcal{F}$  at each node of the abstract problem-supproblem tree so that the junction tree is a tree each node of which is labeled with the set  $\text{dom}(\rho)$ .

## 7 General Junction Trees and Tree Width

The junction tree of the case-factor algorithm is an instance of the more general concept of junction tree. The concept of a junction tree can be defined for any hypergraph. Consider a hypergraph with node set  $V$  and hyperedges  $E_1, \dots, E_k$ . A junction tree for this hypergraph is a tree  $T$  each node  $n$  of which is associated with a set a variables denoted  $\text{dom}(n)$  with  $\text{dom}(n) \subseteq V$  and such that the following conditions hold.

- **Cover Property:** For each  $E_k$  there exists a node  $n$  with  $E_k \subseteq \text{dom}(n)$ .
- **Running Intersection Property:** For any node  $X \in V$ , the set of nodes  $n$  with  $X \in \text{dom}(n)$  is a subtree of  $T$ .

The width of a junction tree is the maximum size of the set  $\text{dom}(n)$  over all nodes  $n$  of the tree minus 1. The tree width of a hypergraph is the minimum width of any junction tree for that hypergraph. Note that a graph is a special case of a hypergraph and it is common to talk about the tree width of a graph. We subtract one from  $\text{dom}(n)$  so that the width of a tree equals 1.

It is NP hard to determine tree width even for graphs. However, good heuristics exists for constructing junction trees of small width. It can be shown that there exists a variable ordering, used to select the variable used in the case rule, such that the case-factor junction tree generated by that variable-selection ordering has minimum width. So one can also define tree width to be the minimum over all variable orderings of the width of the case-factor junction tree for that ordering. Again, good heuristics exist for selecting the next variable to case on.

## 8 The Junction Tree Algorithm

Let  $n$  and  $m$  be any two nodes connected by an edge in the junction tree. It turns out that we do not need to root the tree — we do not need to distinguish which node is the parent and which node is the child. We write  $n \rightarrow m$  if there is an edge in the tree from  $n$  to  $m$ . We will treat the direction  $n \rightarrow m$  differently from  $m \rightarrow n$  — for each direction there is a message and the two messages are

different. We write  $s \rightarrow^* n \rightarrow m$  if the path in the tree from  $s$  to  $m$  includes  $n$ . We allow  $s$  to be  $n$  so that we have  $n \rightarrow^* n \rightarrow m$ . To define the messages we also need to assign each factor to a single node of the junction tree — in general there might be several different nodes  $n$  of the junction tree with  $E_k \subseteq \text{dom}(n)$ . So for each factor we pick one node to cover that factor and then let  $\mathcal{F}(n)$  be the set of factors whose selected node is  $n$ . We now define the message  $Z_{n \rightarrow m}$  to be a function of assignments to  $\text{dom}(n) \cap \text{dom}(m)$  as follows.

$$Z_{n \rightarrow m}(\rho) = Z(\rho, \mathcal{V}_{n \rightarrow m}, \mathcal{F}_{n \rightarrow m}) \quad (16)$$

$$\mathcal{V}_{n \rightarrow m} = \{X \in V : \exists s s \rightarrow^* n \rightarrow m, X \in \text{dom}(s)\} \quad (17)$$

$$\mathcal{F}_{n \rightarrow m} = \{k : \exists s s \rightarrow^* n \rightarrow m, k \in \mathcal{F}(s)\} \quad (18)$$

Given this definition of  $Z_{n \rightarrow m}(\rho)$  we have the following equation which can be used to compute these messages recursively and where  $\rho'$  ranges over assignments to the variables in  $\text{dom}(n)$ .

$$Z_{n \rightarrow m}(\rho) = \sum_{\rho' \sqsubseteq \rho} \left( \prod_{k \in \mathcal{F}(n)} F_k(\rho') \right) \left( \prod_{s \rightarrow n, s \neq m} Z_{s \rightarrow n}(\rho' |_{\text{dom}(s)}) \right) \quad (19)$$

Equation (19) defines the junction tree algorithm. It can be viewed as a recursive definition of the messages, although the semantics of the messages is defined by (16) and (12). The recursion in (19) is well founded — the set of nodes  $s$  with  $s \rightarrow^* n \rightarrow m$  is reduced on each recursive call. Therefore the recursion terminates and equation (19) can be used to compute the messages. It is important to memoize the messages so that each message is only computed once.

## 9 Problems

1. Consider a Bayesian network with three variables  $X_1, X_2, X_3$  where  $X_1$  and  $X_2$  have no parents and  $X_3$  has parents  $X_1$  and  $X_2$  so that we have the following equation.

$$P(X_1 = x_1, X_2 = x_2, X_3 = x_3) = F_1(x_1)F_2(x_2)F_3(x_3, x_1, x_2) \quad (20)$$

a) Prove that  $X_1$  and  $X_2$  are independent, i.e., prove the following.

$$P(X_1 = x_1, X_2 = x_2) = P(X_1 = x_1)P(X_2 = x_2) \quad (21)$$

Now consider a Markov random field on  $X_1, X_2,$  and  $X_3$  with hyperedges

$\{X_1\}$ ,  $\{X_2\}$ , and  $\{X_1, X_2, X_3\}$  so that we have the following.

$$P(X_1 = x_1, X_2 = x_2, X_3 = x_3) = \frac{1}{Z} F_1(x_1) F_2(x_2) F_3(x_1, x_2, x_3) \quad (22)$$

Note the similarity between (20) and (22). Suppose that each of the variables  $X_1$ ,  $X_2$  and  $X_3$  only take on values in the two element set  $\{-1, 1\}$ .

b) Given an example of the energy functions  $E_1$ ,  $E_2$  and  $E_3$  for which equation (21) does *not* hold.

2. Consider an image defined by an  $n \times m$  array  $I$  of integers in the range 0 to 255. A segmentation of an image is a division of the image into segments. We assume that each segment has an identifier which is also an integer in the range from 0 to 256. Let  $I(x, y)$  be the image value at coordinates  $x, y$  and let  $S(x, y)$  be the segment index for the segment containing pixel  $x, y$ . Suppose that we want to find a segmentation  $S$  minimizing the following energy where  $[\Phi]$  is the indicator function for  $\Phi$ , i.e.,  $[\Phi] = 1$  if  $\Phi$  is true and  $[\Phi] = 0$  if  $\Phi$  is false.

$$E(I, S) = \left( \sum_{x,y} (I(x, y) - S(x, y))^2 \right) + \left( \sum_{x,y,\delta \in \{-1,1\}, \gamma \in \{-1,1\}} \lambda [S(x, y) \neq S(x + \delta, y + \gamma)] \right)$$

$\lambda$  is a parameter of the energy function called a “regularization parameter”. Explain the effect of increasing or decreasing  $\lambda$ . Also formalize the problem of minimizing this energy as a Markov random field problem. What are the variables, the hyperedges, and the energy functions associated with each hyperedge.

3. Consider the Bayesian network  $X \rightarrow Z \leftarrow Y \rightarrow W$ . Show that  $X$  and  $W$  are independent. Give an instance of this network (particular conditional probability tables) where  $X$  and  $W$  are not independent given  $Z$ .

4. Consider a graph with nodes  $A_1, A_2, \dots, A_n$  and  $B_1, B_2, \dots, B_n$  and edges from  $A_i$  to  $B_i$ , from  $A_i$  to  $A_{i+1}$  and from  $B_i$  to  $B_{i+1}$ . These edges form a “ladder”. Consider a Markov random field where  $\mathcal{F}$  has a function for each edge of this graph. Give a junction tree for this MRF with width 2.

5. A “series parallel” graph is a graph with special structure (defined below) and with two distinguished nodes called the “interface nodes” of the graph. Series parallel graphs can be defined recursively as follows.

- **edge:** A two node graph with one edge between the two nodes is a series parallel graph where the two given nodes are the interface nodes.

- **series combination:** If  $G_1$  is a series parallel graph with interface nodes  $A$  and  $B$  and  $G_2$  is a series parallel graph with interface nodes  $B$  and  $C$ , and  $B$  is the only node in both  $G_1$  and  $G_2$ , then  $G_1 \cup G_2$  is a series parallel graph with interface nodes  $A$  and  $C$ .
- **parallel combination:** If  $G_1$  is a series parallel graph with interface nodes  $A$  and  $B$  and  $G_2$  is a series parallel graph which also has interface nodes  $A$  and  $B$ , and  $A$  and  $B$  are the only nodes in both  $G_1$  and  $G_2$ , then  $G_1 \cup G_2$  is a series parallel graph with interface nodes  $A$  and  $B$ .

a) Draw a series-parallel graph that is a parallel combination of two series combinations of edges.

b) Consider a Markov random field (MRF) where each edge of the graph in part a) is considered to be a hyperedge of the field. Draw a junction tree of width 2 for this MRF.

c) Prove, by structural induction on the definition of a series-parallel graph, that for every series-parallel graph there exists a width 2 junction tree.