T.61.5140 Machine Learning: Advanced Probablistic Methods
Hollmén, Raiko (Spring 2008)
Problem session, 22nd of February, 2008
http://www.cis.hut.fi/Opinnot/T-61.5140/

1. Draw the graphical model and factor graph (page 399), and run the sum-product algorithm (page 402, also known as belief propagation) on pencil and paper for the model below. Variables are S for sprinkler being on, R for raining, and W for the grass being wet. Compute $P(W = t)$.

P(S)	
S=f	0.7
S=t	0.3

P(R)	
R=f	0.8
R=t	0.2

$P(W \mid S, R)$	S=f, R=f	S=f, R=t	S=t, R=f	S=t, R=t
W=f	1.0	0.0	0.0	0.0
W=t	0.0	1.0	1.0	1.0

Solution:

Figure1 shows the graphical model and the factor graph. The factor graph has a round node for each variable and a square node for each factor. This model has three factors as can be seen from the model equation $P(S, R, W) = P(S)P(R)P(W \mid S, R)$.

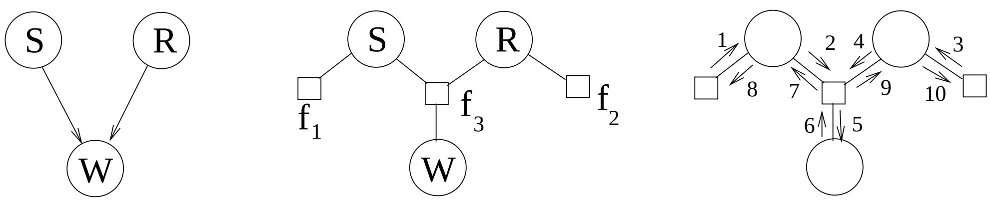

Figure 1: Problem 1. Left: graphical model. Middle: Factor graph. Right: Messages μ.

There are two types of messages, from variables to factors and from factors to variables. They are computed as follows:

$$\mu_{X_m \to f_s}(X_m) = \prod_{l \in ne(X_m) \setminus f_s} \mu_{f_l \to X_m}(X_m) \tag{1}$$

$$\mu_{f_x \to X_0}(X_0) = \sum_{X_1} \cdots \sum X_M f_s(X_0, X_1, \ldots, X_M) \prod_{m=1}^{M} \mu_{X_m \to f_s}(X_m), \tag{2}$$

where $ne(X_m)$ is the set of factors neighbouring the variable X_m and $X_0, X_1, \ldots, X_M$ are the variables neighbouring the factor f_s. Because they are recursive, all the other incoming messages must have been sent to a node before it can send its message.

The 10 messages (see right subfigure of Figure 1) can be sent for instance in the following order:

$$1: \quad \mu_{f_1 \to S}(S) = f_1(S) = \begin{pmatrix} 0.7 \\ 0.3 \end{pmatrix} \tag{3}$$

$$2: \quad \mu_{S \to f_3}(S) = \mu_{f_1 \to S}(S) = \begin{pmatrix} 0.7 \\ 0.3 \end{pmatrix} \tag{4}$$

$$3: \quad \mu_{f_2 \to R}(R) = f_2(R) = \begin{pmatrix} 0.8 \\ 0.2 \end{pmatrix} \tag{5}$$

$$4: \quad \mu_{R \to f_3}(R) = \mu_{f_2 \to R}(R) = \begin{pmatrix} 0.8 \\ 0.2 \end{pmatrix} \tag{6}$$

$$5: \quad \mu_{f_3 \to W}(W) = \sum_S \sum_R f_3(S, R, W) \mu_{S \to f_3}(S) \mu_{R \to f_3}(R) \tag{7}$$

$$= \sum_S \sum_R = \begin{pmatrix} 0.7 \cdot 0.8 & 0 & 0 & 0 \\ 0 & 0.7 \cdot 0.2 & 0.3 \cdot 0.8 & 0.3 \cdot 0.2 \end{pmatrix} \tag{8}$$

$$= \sum_S \sum_R = \begin{pmatrix} 0.56 & 0 & 0 & 0 \\ 0 & 0.14 & 0.24 & 0.06 \end{pmatrix} \tag{9}$$

$$= \begin{pmatrix} 0.56 \\ 0.44 \end{pmatrix} \tag{10}$$

$$
\begin{aligned}
6: &\quad \mu_{W\to f_3}(W) = \begin{pmatrix} 1.0 \\ 1.0 \end{pmatrix} && (11)\\
7: &\quad \mu_{f_3\to S}(S) = \sum_R \sum_W f_3(S,R,W)\mu_{R\to f_3}(R)\mu_{W\to f_3}(W) && (12)\\
&\quad = \sum_R \sum_W \begin{pmatrix} 0.8 & 0 & 0 & 0 \\ 0 & 0.2 & 0.8 & 0.2 \end{pmatrix} && (13)\\
&\quad = \begin{pmatrix} 1.0 \\ 1.0 \end{pmatrix} && (14)\\
8: &\quad \mu_{S\to f_1}(S) = \mu_{f_3\to S}(S) = \begin{pmatrix} 1.0 \\ 1.0 \end{pmatrix} && (15)\\
9: &\quad \mu_{f_3\to R}(R) = \sum_S \sum_W f_3(S,R,W)\mu_{S\to f_3}(S)\mu_{W\to f_3}(W) && (16)\\
&\quad = \sum_R \sum_W \begin{pmatrix} 0.7 & 0 & 0 & 0 \\ 0 & 0.7 & 0.3 & 0.3 \end{pmatrix} && (17)\\
&\quad = \begin{pmatrix} 1.0 \\ 1.0 \end{pmatrix} && (18)\\
10: &\quad \mu_{R\to f_2}(R) = \mu_{f_3\to R}(R) = \begin{pmatrix} 1.0 \\ 1.0 \end{pmatrix} && (19)
\end{aligned}
$$

After the messages have been sent in both directions, it is easy to compute different marginals for single variables and for variables within a factor:

$$P(X) = \prod_{s\in ne(X)} \mu_{f_s\to X}(X) \tag{20}$$

$$P(X_0, X_1, \ldots, X_M) = f_s(X_0, X_1, \ldots, X_M) \prod_{i=0}^{M} \mu_{X_i\to f_s}(X_i) \tag{21}$$

From this we can compute

$$P(W) = \mu_{f_3\to W}(W) = \begin{pmatrix} 0.56 \\ 0.44 \end{pmatrix} \tag{22}$$

and thus $P(W = t) = 0.44$.

2. The model in Problem 1 is extended with a variable C for the sky being cloudy. (a) Draw the graphical model and the factor graph. (b) Why

cannot you run the sum-product algorithm? (c) Where does the inference stop if you try anyway? (d) One could avoid the problem by using loopy belief propagation (page 417), explain shortly how.

P(C)	
C=f	0.5
C=t	0.5

$P(S \mid C)$	C=f	C=t
S=f	0.5	0.9
S=t	0.5	0.1

$P(R \mid C)$	C=f	C=t
R=f	1.0	0.4
R=t	0.0	0.6

$P(W \mid S, R)$	S=f, R=f	S=f, R=t	S=t, R=f	S=t, R=t
W=f	1.0	0.0	0.0	0.0
W=t	0.0	1.0	1.0	1.0

Solution:

(a) See Figure 2.

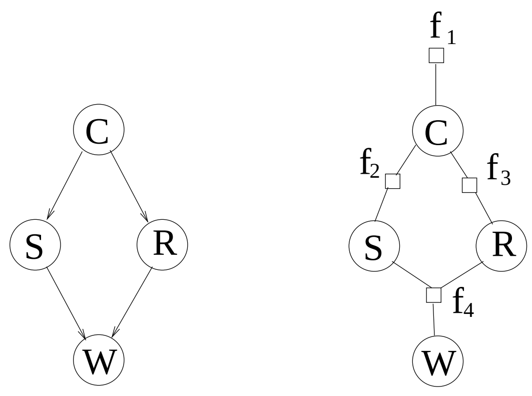

Figure 2: Problem 2. Left: graphical model. Right: Factor graph.

(b) The factor graph is not a tree (there is a loop).

(c) The messages in the loop cannot be sent because they are all waiting for incoming messages before they can send their own.

(d) In loopy belief propagation, each message μ is initialized to unity (such as $\begin{pmatrix} 1.0 \\ 1.0 \end{pmatrix}$). Then messages are updated iteratively using the normal message passing formulae. (This procedure might or might not converge to one or the other stable state, but it only gives an approximate solution. This is because the incoming messages are assumed to be independent, which is not true in the loopy case.)

3. The junction tree algorithm is the generalization of the sum-product algorithm to general graphs. When applied to the model in Problem 2, it would effectively cluster variables S and R into a single variable, say S_R, that has four possible values, f_f, f_t, t_f, and t_t to represent the different combinations of S and R being false or true. (a) Write the joint probability $P(C, S_R, W)$. (b) Draw the graphical model and a factor graph. (c) Compute the conditional probability tables for the modified model.

Solution:

$P(C, S_R, W) = P(C)P(S_R \mid C)P(W \mid S_R)$

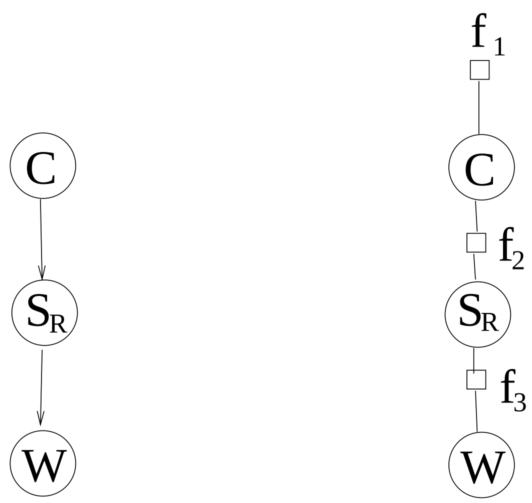

Figure 3: Problem 3. Left: graphical model. Right: Factor graph.

P(C)	
C=f	0.5
C=t	0.5

$P(S_R \mid C)$	C=f	C=t
$S_R = f_f$	0.5	0.36
$S_R = f_t$	0	0.54
$S_R = t_f$	0.5	0.04
$S_R = t_t$	0	0.06

$P(W \mid S_R)$	$S_R = f_f$	$S_R = f_t$	$S_R = t_f$	$S_R = t_t$
W=f	1.0	0.0	0.0	0.0
W=t	0.0	1.0	1.0	1.0

4. For the model in Figure 4, construct a junction tree (also known as join tree). This is formed by moralizing the graph (connecting co-parents), forgetting the direction of the edges, triangulating (adding a chord to each chordless cycle of four or more nodes), and finally creating a junction tree based on the resulting graph. The junction tree has a node for each maximal clique of the previous graph. Temporarily all the nodes are connected with weighted edges, the weight being the number of shared variables in the two cliques. A maximum weight spanning tree is formed, and other edges and all weight can be forgotten. These nodes (cliques) correspond to the small square nodes of the factor graph. One still needs to add the variables in the graph as round nodes. Variables in the intersection of neighbouring cliques need to be clustered into a single variable, just like S and R in Problem 3. (Note that a variable may appear many times. Note also that the junction tree is not unique, there can be many ways to triangulate and to choose a maximum spanning tree.)

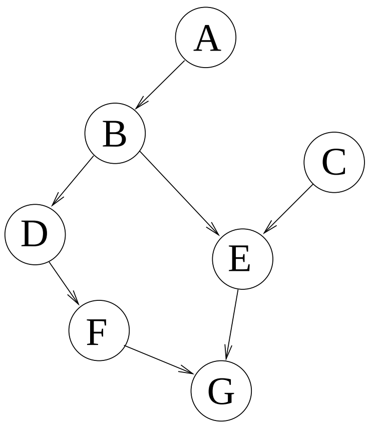

Figure 4: Problem 4.

Solution:

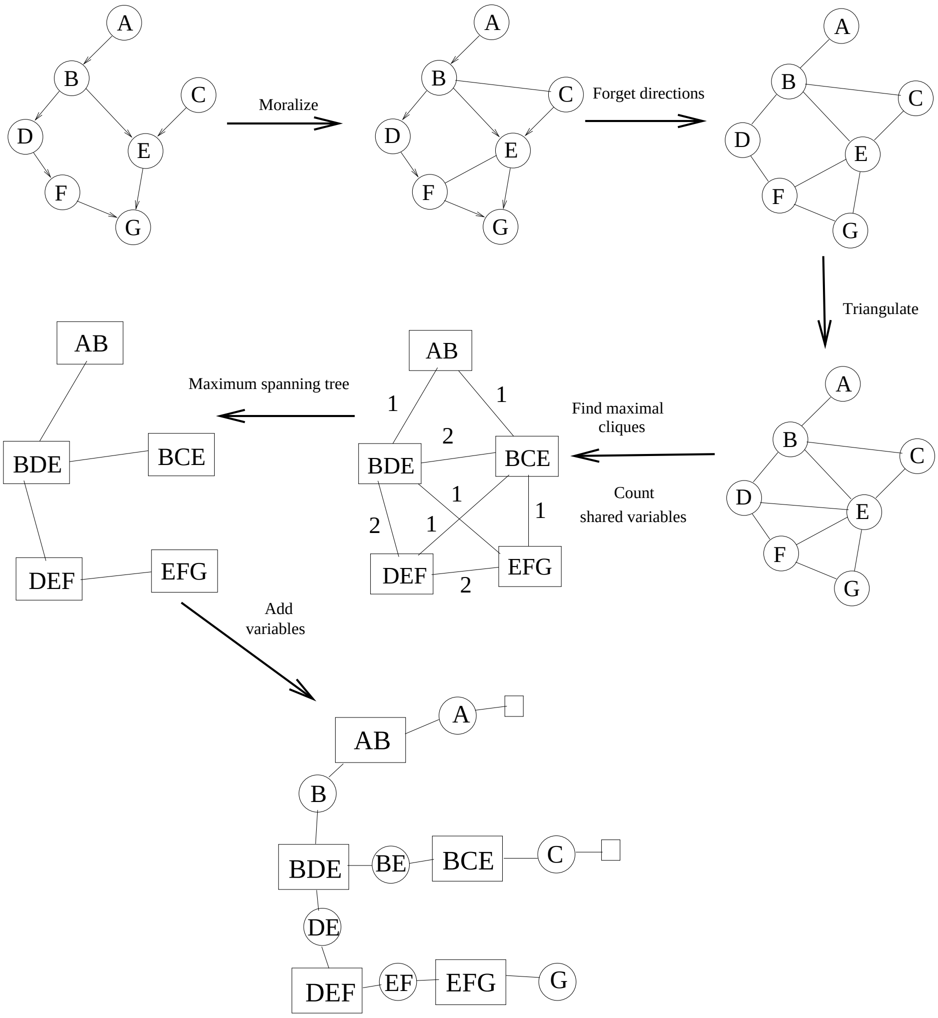

Figure 5: Solution of Problem 4.

Note that the subgraph with nodes where a particular variable $A \dots G$ appears, is unbroken. This is essential for the sum-product algorithm to work properly. The procedure with the maximum spanning tree etc. ensures that this property holds for each variable.