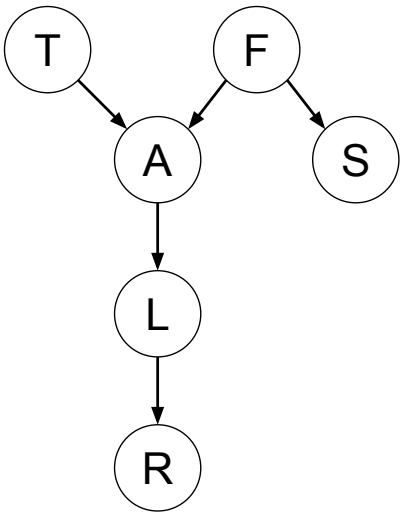
CSE 150. Assignment 3 *Fall 2018*

Out: *Tue Oct 16*
Due: *Tue Oct 23 (beginning of class)*

3.1 Variable Elimination Algorithm

Consider the following belief network with binary random variables.



(Source: Adapted from Poole and Mackworth, *Artificial Intelligence 2E*, Section 8.3.2. For a description of the meaning of the variables in this belief network, see Example 8.15.)

Suppose we want to compute a probability distribution over T given the evidence $S = 1$ and $R = 1$. That is, we want to compute $P(T = 0|S = 1, R = 1)$ and $P(T = 1|S = 1, R = 1)$. In this question, we will evaluate the efficiency of two different methods of computing these probabilities.

The conditional probabilities of this belief network (expressed as factor tables) are as follows:

T	$f_0(T) = P(T)$
0	0.98
1	0.02

F	$f_1(F) = P(F)$
0	0.99
1	0.01

T	F	A	$f_2(T, F, A) = P(A\|T, F)$
0	0	0	0.9999
0	0	1	0.0001
0	1	0	0.01
0	1	1	0.99
1	0	0	0.15
1	0	1	0.85
1	1	0	0.5
1	1	1	0.5

F	S	$f_3(F, S) = P(S\|F)$
0	0	0.99
0	1	0.01
1	0	0.1
1	1	0.9

A	L	$f_4(A, L) = P(L\|A)$
0	0	0.999
0	1	0.001
1	0	0.12
1	1	0.88

L	R	$f_5(L, R) = P(R\|L)$
0	0	0.99
0	1	0.01
1	0	0.25
1	1	0.75

Note that the conditional probabilities are specified with some redundancy. For example, the tables store both $P(A = 0|T = 1, F = 1)$ and $P(A = 1|T = 1, F = 1)$, even though this information is technically redundant. Therefore, no addition or subtraction operations are required to compute $P(A = 0|T = 1, F = 1)$ based on $P(A = 1|T = 1, F = 1)$ or vice versa.

(a) **Computation using variable elimination**

Use the variable elimination algorithm (with elimination order S, R, L, A, F) to compute

$$P(T = 0|S = 1, R = 1) \quad \text{and} \quad P(T = 1|S = 1, R = 1).$$

You do not need to write code for this; you should apply the algorithm step-by-step and show your work, including any new factor tables that are computed along the way.

(b) **Counting calculations used by the variable elimination algorithm**

In the following table, fill in the number of multiplication, addition, and division operations needed for each phase of the VE algorithm. You should count operations using the same method used in class: for example, $a \times b + c \times d + e \times f$ involves 3 multiplications and 2 additions.

Phase of algorithm	**# multiplications**	**# additions**	**# divisions**
Eliminate S (evidence)	0	0	0
Eliminate R (evidence)	0	0	0
Eliminate L			0
Eliminate A			0
Eliminate F			0
Combine T factors			0
Normalize distribution over T	0	1	2
Total			2

(c) **Counting calculations used by the enumeration algorithm**

In the brute-force enumeration method, we can compute the desired probabilities via

$$P(T = 0|S = 1, R = 1) = \frac{P(T = 0, S = 1, R = 1)}{P(T = 0, S = 1, R = 1) + P(T = 1, S = 1, R = 1)}$$
$$P(T = 1|S = 1, R = 1) = \frac{P(T = 1, S = 1, R = 1)}{P(T = 0, S = 1, R = 1) + P(T = 1, S = 1, R = 1)}$$

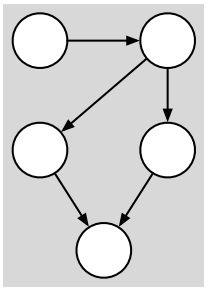
That is, we compute $P(T = 0, S = 1, R = 1)$ and $P(T = 1, S = 1, R = 1)$ and then normalize to obtain a probability distribution over T.

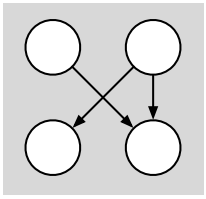
In the following table, fill in the number of multiplication, addition, and division operations needed for each phase of this algorithm. Assume that the algorithm does a full brute-force calculation, even if some operations are redundant.

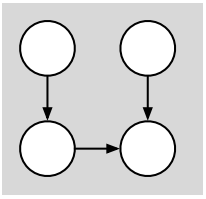
Phase of algorithm	**# multiplications**	**# additions**	**# divisions**
Compute $P(T = 0, S = 1, R = 1)$			0
Compute $P(T = 1, S = 1, R = 1)$			0
Normalize distribution over T	0	1	2
Total			2

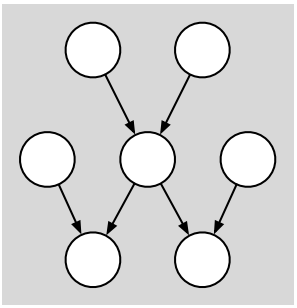
3.2 To be, or not to be, a polytree: that is the question.

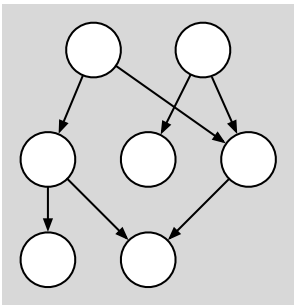
Circle the DAGs shown below that are polytrees. In the other DAGs, shade **two** nodes that could be *clustered* so that the resulting DAG is a polytree.

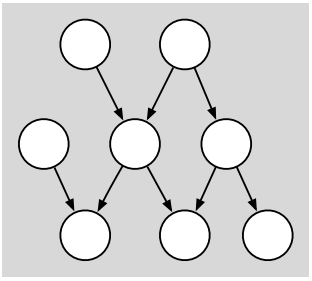


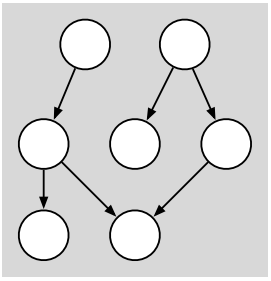


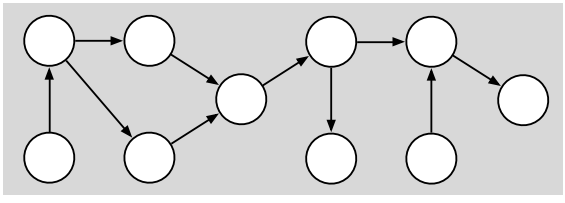


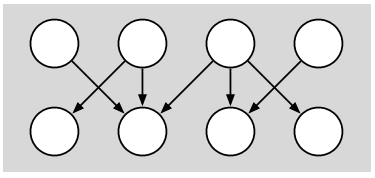


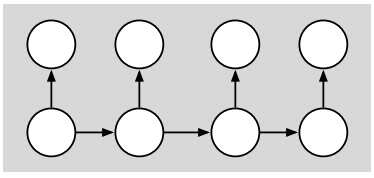








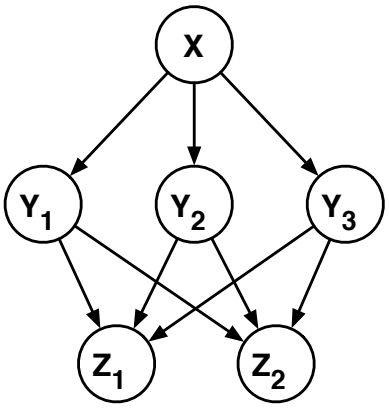




3.3 Node clustering

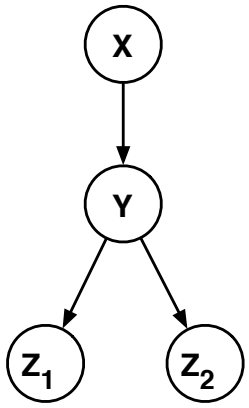
Consider the belief network shown below over binary variables X, Y_1, Y_2, Y_3, Z_1, and Z_2. The network can be transformed into a polytree by clustering the nodes Y_1, Y_2, and Y_3 into a single node Y. From the CPTs in the original belief network, fill in the missing elements of the CPTs for the polytree.

X	$P(Y_1 = 1\|X)$	$P(Y_2 = 1\|X)$	$P(Y_3 = 1\|X)$
0	0.85	0.30	0.50
1	0.25	0.35	0.70



Y_1	Y_2	Y_3	$P(Z_1 = 1\|Y_1, Y_2, Y_3)$	$P(Z_2 = 1\|Y_1, Y_2, Y_3)$
0	0	0	0.8	0.2
1	0	0	0.7	0.3
0	1	0	0.6	0.4
0	0	1	0.5	0.5
1	1	0	0.4	0.6
1	0	1	0.3	0.7
0	1	1	0.2	0.8
1	1	1	0.1	0.9

Y_1	Y_2	Y_3	Y	$P(Y\|X=0)$	$P(Y\|X=1)$	$P(Z_1=1\|Y)$	$P(Z_2=1\|Y)$
0	0	0	1				
1	0	0	2				
0	1	0	3				
0	0	1	4				
1	1	0	5				
1	0	1	6				
0	1	1	7				
1	1	1	8				



3.4 Maximum likelihood estimation for an n-sided die

A n-sided die is tossed T times, and the results recorded as data. Assume that the tosses $\{x^{(1)}, x^{(2)}, \ldots, x^{(T)}\}$ are identically distributed (i.i.d.) according to the probabilities $p_k = P(X = k)$ of the die, and suppose that over the course of T tosses, the k^{th} side of the die is observed C_k times.

(a) **Log-likelihood**

Express the log-likelihood $\mathcal{L} = \log P(\text{data})$ of the observed results $\{x^{(1)}, x^{(2)}, \ldots, x^{(T)}\}$ in terms of the probabilities p_k and the counts C_k. In particular, show that

$$\mathcal{L}(p) \;=\; \sum_{k=1}^{n} C_k \log p_k.$$

(b) **KL distance**

Define the distribution $q_k = C_k/T$, where $T = \sum_k C_k$ is the total number of counts. Show that

$$\text{KL}(q, p) \;=\; \sum_{k=1}^{n} q_k \log q_k \;-\; \frac{\mathcal{L}(p)}{T},$$

where $\text{KL}(q, p)$ is the KL distance from homework problem 1.6. Conclude that maximizing the log-likelihood $\mathcal{L}$ in terms of the probabilities p_k is equivalent to minimizing the KL distance $\text{KL}(q, p)$ in terms of these same probabilities. (Why is this true?)

(c) **Maximum likelihood estimation**

Recall from homework problem 1.6b that $\text{KL}(q, p) \geq 0$ with equality if and only if $q_k = p_k$ for all k. Use this to argue that $p_k = C_k/T$ is the maximum-likelihood estimate—i.e., the distribution that maximizes the log-likelihood of the observed data.

Note: you have done something quite clever and elegant here, perhaps without realizing it! Generally speaking, to maximize the log-likelihood in part (a), you need to solve the following optimization:

$$\textbf{Maximize} \quad \sum_k C_k \log p_k \quad \textbf{subject to} \quad \sum_k p_k = 1 \quad \textbf{and} \quad p_k \geq 0.$$

This is a problem in multivariable calculus requiring the use of Lagrange multipliers to enforce the constraint. You have avoided this calculation by a clever use of the result in homework problem 1.6.