

CS228 Winter 2016 Homework 2

SUNet ID: dguo1113

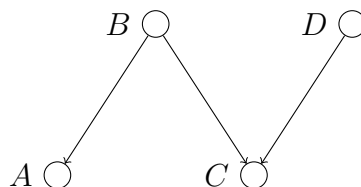
Name: Dan Guo

Collaborators: Aran Nayebi

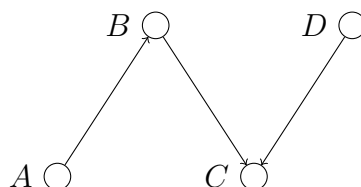
By turning in this assignment, I agree by the Stanford honor code and declare that all of this is my own work.

1 Problem 1

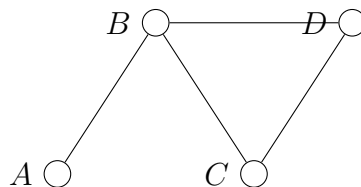
(a)



(b) Yes there is one I-equivalent graph. The arrow between A-B flipped.



(c)



This is not a perfect map as $B \perp D \in I(P)$, but $B \perp D \notin I(G)$ where G is the Markov Network above.

2 Problem 2

(a) Through marginalization of the full joint, we get that $P(C = 0) = .5$, $P(D = 0) = .5$, $P(B = 1) = \frac{5}{8}$

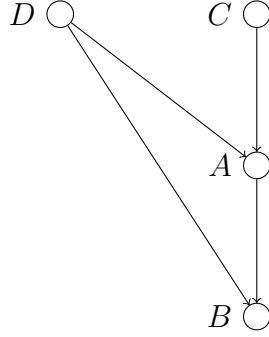
Similarly, through marginalization, we arrive that

$$P(C = c, D = d) = .25, \forall c = 0, 1, d = 0, 1$$

$$P(B = 1, C = 0) = \frac{3}{8}$$

Thus, it is clear that $(C \perp D) \in I(P), (C \perp B) \notin I(P)$

(b)



Yes this solution is unique. Because we know the structure of the BN already, it suffices to assign edges until we assign the whole graph, or we reach a state of contradiction. Specifically, if any assignment creates an active path between C, D or C, B then it is not a valid I-map of P , and hence cannot be its perfect map. Or, if any assignment introduces a cycle, this cannot be a BN.

Here are all possible assignments, with a 1 in edge signifying directed from 1st node to second, i.e. for an edge $(u, v), u \rightarrow v$:

(A, C)	(A, B)	(A, D)	(B, D)	Valid?
0	0	0	0	No, no active path from C, B
0	0	0	1	No, no active path from C, B
0	0	1	0	No, active path $C - A - D$
0	0	1	1	No, active path $C - A - D$
0	1	0	0	Yes
0	1	0	1	No, active path $C - A - B - D$
0	1	1	0	No, active path $C - A - D$
0	1	1	1	No, active path $C - A - D$
1	0	0	0	No, active path $C - A - D$
1	0	0	1	No, active path $C - A - D$
1	0	1	0	No, active path $C - A - D$
1	0	1	1	No, active path $C - A - D$
1	1	0	0	No, active path $C - A - D$
1	1	0	1	No, active path $C - A - D$
1	1	1	0	No, active path $C - A - D$
1	1	1	1	No, active path $C - A - D$

Note another explanation for why the network is not valid is that some of the networks have arrows that will introduce a cycle. The network must be a DAG.

(c) CPDs for each node

C	$P(C)$
0	.5
1	.5

D	$P(D)$
0	.5
1	.5

A	C	D	$P(A C, D)$
0	0	0	$1/2$
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	$1/2$
1	0	1	0
1	1	0	1
1	1	1	1

A	B	D	$P(B A, D)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

3 Problem 3

$$\begin{aligned}
P(\mathbf{Y} = m|\mathbf{X}) &= \frac{1}{Z(\mathbf{X})} \hat{P}(\mathbf{Y}, \mathbf{X}) \\
&= \frac{\prod_{i=0}^k \phi_i(\mathbf{D}_i)}{\sum_{j=1}^M \hat{P}(j, \mathbf{X})} \\
&= \frac{\exp(w_0^m) \prod_{i=1}^k \exp(w_i^m \mathbf{X}_i)}{\sum_{j=1}^M \exp(w_0^j) \prod_{i=1}^k \exp(w_i^j \mathbf{X}_i)} \\
&= \frac{\exp(\sum_{i=1}^k w_i^m \mathbf{X}_i + w_0^m)}{\sum_{j=1}^M \exp(\sum_{i=1}^k w_i^j \mathbf{X}_i + w_0^j)} \\
&= \frac{\exp(\theta_m^T x + b_m)}{\sum_{j=1}^M \exp(\theta_j^T x + b_j)}
\end{aligned}$$

where for the last set, we define column vector $\theta_j = [w_1^j, \dots, w_k^j]$, column vector $\theta_m = [w_1^m, \dots, w_k^m]$, $b_m = w_0^m$, $b_j = w_0^j$.

4 Problem 4

(a)

$$\begin{aligned}
 P(X_i) &= \frac{\sum_{X_{i+1}} P(X_i, X_{i+1})}{1} \\
 &= \frac{\sum_{X_{i+1}} P(X_i, X_{i+1})}{\sum_{X_i, X_{i+1}} P(X_i, X_{i+1})} \\
 &= \frac{\sum_{X_{i+1}} k\beta(C_i)}{\sum_{X_i, X_{i+1}} k\beta(C_i)} \\
 &= \frac{\sum_{X_{i+1}} \beta(C_i)}{\sum_{X_i, X_{i+1}} \beta(C_i)}
 \end{aligned}$$

Thus to calculate $P(X_i)$, we simply need to calculate the numerator and denominator. For example, we can sum over X_{i+1} of the belief of clique i , $\beta(C_i)$, to arrive at the numerator and sum over X_i, X_{i+1} to arrive at the denominator.

The numerator is summing over all values of X_i , with the belief being a table lookup, hence $O(d)$. The denominator is summing over all pairs of X_i, X_{i+1} , with the summand being a lookup, hence $O(d^2)$. In total, for a given X_i , we arrive at $O(d^2)$.

The result does not depend on which clique we use to extract the marginal form as the clique tree is calibrated. We can extract the probability $P(X_i)$ from either clique C_i, C_{i-1} that are dependent on X_i as shown here.

$$\begin{aligned}
 \frac{\sum_{X_{i-1}} \beta(C_{i-1})}{\sum_{X_{i-1}, X_i} \beta(C_{i-1})} &= \frac{\sum_{X_{i-1}} l\beta(C_{i-1})}{\sum_{X_{i-1}, X_i} l\beta(C_{i-1})} \\
 &= \frac{\sum_{X_{i-1}} P(X_{i-1}, X_i)}{\sum_{X_{i-1}, X_i} P(X_{i-1}, X_i)} \\
 &= P(X_i) \\
 &= \frac{\sum_{X_{i+1}} P(X_i, X_{i+1})}{\sum_{X_i, X_{i+1}} P(X_i, X_{i+1})} \\
 &= \frac{\sum_{X_{i+1}} k\beta(C_i)}{\sum_{X_i, X_{i+1}} k\beta(C_i)} \\
 &= \frac{\sum_{X_{i+1}} \beta(C_i)}{\sum_{X_i, X_{i+1}} \beta(C_i)}
 \end{aligned}$$

(b) The runtime complexity is $O(n^3 d^3)$.

We derived in lecture that the runtime complexity of Sum-Product-VE is $O(mv^{N_{max}})$. In our case the number of factors $m = O(n)$. The number of possible assignments of a given random variable is d , hence $v = d$. The maximum scope of any factor generated is $N_{max} = 3$ because when we sum out a particular variable it involves at most two other variables, its adjacent ones in the Markov Chain. This is $O(nd^3)$. The other operations, creating a subtree and looping to update factors, in Algorithm 1 are also $O(nd^3)$.

Because this is only one run of Algorithm 1, and we need to run $O(n^2)$ times for all marginal pairs, we arrive at our desired runtime.

(c) We show that $P(X_i, X_j) = \sum_{X_{j-1}} P(X_i, X_{j-1}), P(X_j|X_{j-1}), i < j - 1$

$$\begin{aligned} P(X_i, X_j) &= \sum_{X_{j-1}} P(X_i, X_{j-1}, X_j) \\ &= \sum_{X_{j-1}} P(X_i, X_j) P(X_j|X_i, X_{j-1}) \\ &= \sum_{X_{j-1}} P(X_i, X_j) P(X_j|X_{j-1}) \end{aligned}$$

where the second equality uses that there is no clique involving X_j, X_k , where $k < j - 1$. Thus, conditioning on X_{j-1} , X_j independent on all such X_k . In particular, $X_j \perp X_i | X_{j-1}$

(d) First note the following:

$$P(X_j|X_{j-1}) = P(X_j, X_{j-1})/P(X_{j-1}) = \frac{k\beta(C_{j-1})}{\sum_{X_j} k\beta(C_{j-1})} = \frac{\beta(C_{j-1})}{\sum_{X_j} \beta(C_{j-1})}$$

and,

$$P(X_i) = \frac{\sum_{X_{i+1}} \beta(C_i)}{\sum_{X_i, X_{i+1}} \beta(C_i)}$$

, from part (a).

Let us define the recurrence as follows:

$$P(X_i, X_j) = \begin{cases} P(X_j|X_{j-1})P(X_{j-1}) & \text{if } i = j - 1 \\ \sum_{X_{j-1}} P(X_i, X_{j-1}), P(X_j|X_{j-1}) & \text{if } i < j - 1 \end{cases}$$

Let the algorithm be as follows: First compute all $P(X_i)$ marginal probabilities as in part (a). Specifically $P(X_i) = \frac{\sum_{x_{i+1}} \beta(C_i)}{\sum_{x_i, x_{i+1}} \beta(C_i)}$

Next we compute all $P(X_j|X_{j-1})$ conditional probabilities, by the formula above.

Finally we calculate all pairwise marginal probabilities $P(X_i, X_j)$. The order we calculate is for every X_i , starting from $P(X_i, X_{i+1}), P(X_i, X_{i+2}), \dots, P(X_i, X_n)$

Here is the runtime analysis: For calculating $P(X_i)$, from part (a), for a given X_i is $O(d^2)$ time. Calculating for all values of X_i , and over all i , we get total time of $O(nd^3)$.

For calculating $P(X_j|X_{j-1})$, the costliest operation is summing in the denominator over d values, hence $O(d)$ operation. For all such X_j, X_{j-1} pairings, it costs $O(nd)$.

For the inductive step, for a particular pairing X_i, X_j , the calculation has two cases. If $i = j - 1$, we require two lookups and a product, hence $O(1)$. Note that by the order of our algorithm, we have already calculated the conditional and single marginal probabilities. If $i < j - 1$, the calculation is simply summing over d values of X_{j-1} and the summand is two lookups, as $P(X_i, X_{j-1})$ term has been calculated already and $P(X_j|X_{j-1})$ term has been calculated before hand. Thus regardless of case, the calculation is $O(d)$ time. But note that we must calculate $P(X_i, X_j)$ for all possible assignments of X_i, X_j and between all pairings of random variables. For a particular pair of random variables, there are $O(d^2)$ assignments. Finally, there are $O(n^2)$ pairings of random variables. Hence this inductive step is $O(n^2d^3)$.

The computation is $O(n^2d^3)$.

5 Problem 5

Let us define a Bayesian Network with the following structure:
and conditional probability distributions:

A	$P(A)$
0	.4
1	.6

A	B	$P(B A)$
0	0	.1
0	1	.9
1	0	.5
1	1	.5

Note calculating MAP on variable A is as follows:

$$\text{MAP}(A) = \arg \max_a P(A = a) = 1$$

Note the probabilities for all possible assignments of A, B are as follows. This is computed using the joint specified by the structure, that $P(A, B) = P(A)P(B|A)$.

A	B	$P(A, B)$
0	0	.04
0	1	.36
1	0	.3
1	1	.3

This shows that the MPE assignment is $A = 0, B = 1$. However the MAP assignment for $A = 1$. Thus the assignments do not match.

6 Problem 6

- (a) Implemented factors. Used Variable Elimination. The source code is in Section 7, but for full experience refer to submitted code.
- (b) -15.801923869942652

FromNode	ToNode	A	C	G	T
1	3	0.032	0.816	0.028	0.076
2	3	0.052	0.11	0.042	0.842
3	5	0.00768265	0.08039275	0.00635416	0.06082532
4	5	0.032	0.816	0.028	0.076
6	8	0.085	0.028	0.808	0.029
7	8	0.032	0.816	0.028	0.076
8	10	0.0050291	0.019645	0.01934434	0.00439247
9	10	0.831	0.046	0.122	0.053
5	11	0.00255902	0.05405479	0.00220472	0.00889616
10	11	0.00371453	0.00102133	0.00245183	0.00055463
12	14	0.831	0.046	0.122	0.053
13	14	0.831	0.046	0.122	0.053
11	15	1.03817927e-05	4.61807163e-05	7.28043697e-06	9.01085538e-06
14	15	0.57533511	0.0342182	0.09645194	0.03955736
15	17	5.09236150e-06	1.62308990e-06	1.35531163e-06	7.57157749e-07
16	17	0.085	0.028	0.808	0.029
17	19	4.55377840e-07	9.00733950e-08	9.39836682e-07	7.66409639e-08
18	19	0.831	0.046	0.122	0.053

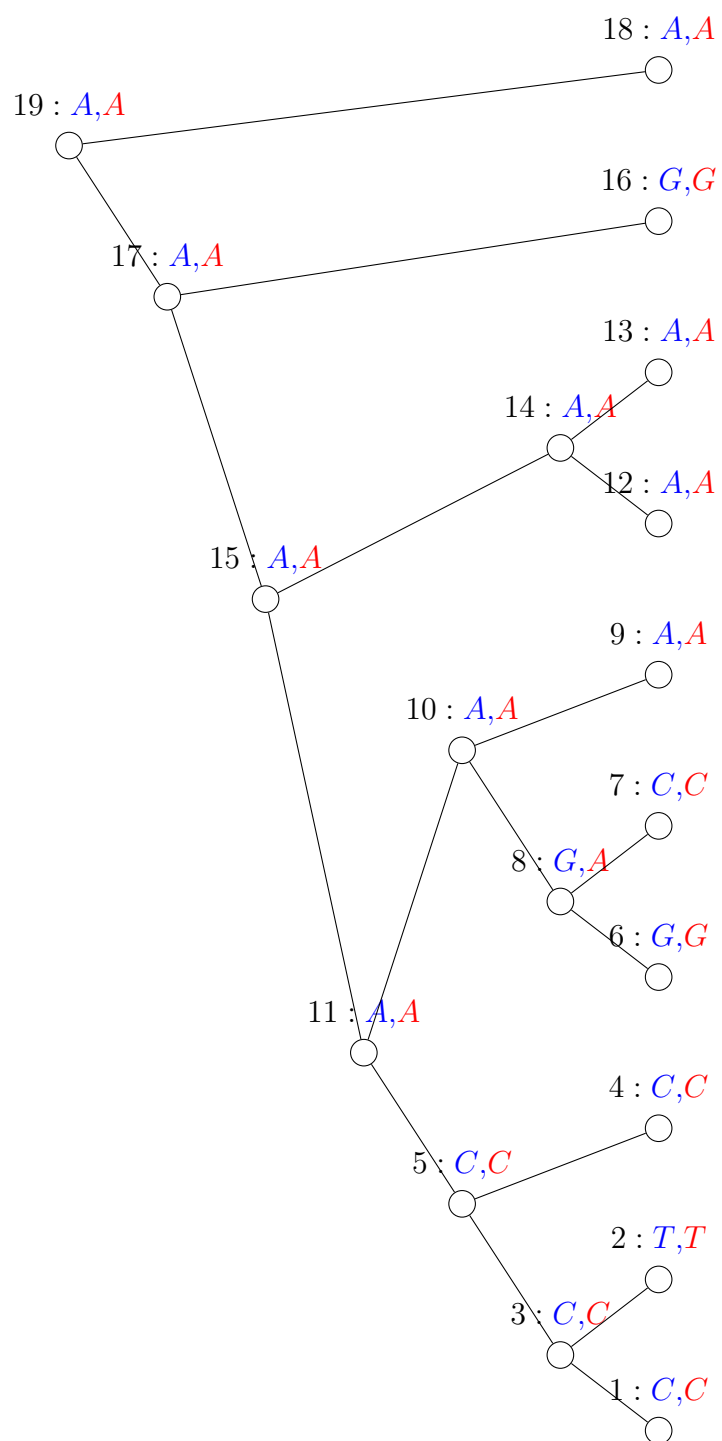
(c) Table and Tree

NodeIndex	A	C	G	T	MaxAssignment
3	0.01097032	0.80709913	0.002892	0.17903854	C
5	0.05363426	0.84801463	0.01202623	0.08632488	C
8	0.31047568	0.25534941	0.40861052	0.02556439	G
10	0.68168057	0.15212077	0.14858574	0.01761291	A
11	0.6494745	0.19702325	0.12504436	0.0284579	A
14	0.97700779	0.00136516	0.0205555	0.00107156	A
15	0.82186714	0.02269878	0.14744669	0.00798738	A
17	0.65637297	0.00577348	0.33337198	0.00448156	A
19	0.81373552	0.00619151	0.17133829	0.00873468	A

Tree in part (d)

(d) The maximal assignment achieves log probability of -17.70357925.

NodeIndex	MaxAssignment
3	C
5	C
8	A
10	A
11	A
14	A
15	A
17	A
19	A



c assignments in blue, d assignments in red.

The assignments do differ in parts (c) and (d) on assignment of node 8. In part (c), we are calculating with sum product variable elimination the marginal, thereby sum marginalizing over nodes not leaf or x_v . In part (d), we are actually not summing but max marginalizing over nodes not leaf or x_v . In this case, we are calculating the assignment of x_v in the maximal assignment of the joint $p(\text{leaves}, \text{ancestors})$. Specifically, the part c is the MAP of each ancestor variable while d is MPE. We showed in problem 5 that these two assignments need not agree.

- (e) -9379.70771125. python problemSolutions.py e; see Section 7 Source Code.
- (f) -9474.43836219. The tree from part(e) is a better fit as it affords a higher probability of the assignments in modern species.
- (g) We could theoretically enumerate all possible phylogeny trees that can possibly exist amongst the species that are currently alive. Next for each such tree, we calculate the log likelihood of the bases at a particular spot for all species, across multiple gene locations (as in parts e,f). This will give a log likelihood of how likely it is we have the current species. The most likely phylogeny would be the tree that outputs the highest probability for base assignments in modern species. A problem is having to enumerate many trees and computing log likelihoods of each, which will be expensive.