Instructor's Manual for *Probabilistic Graphical Models*

Daphne Koller, Benjamin Packer, and many generations of TAs

October 2010

Foreword and Disclaimer

This instructor's manual contains solutions for a *few* of the problems in the textbook *Probabilisic Graphical Models: Principles and Techniques*, by Daphne Koller and Nir Friedman. The exercises for which solutions are provided are those that were used over the years in Daphne Koller's course on this topic. (Most of these exercises were included in the textbook, but a few are new.) Hence, the coverage of the solutions is highly biased to those topics covered in the course.

For some solutions, we have included relative point allocation to different parts of the question, which can help indicate their relative difficulty. To assist in grading, we have also sometimes included common errors and the number of points deducted.

Since the solutions were developed over several years, while the book was being written (and rewritten), the notation and formatting used might not always be entirely consistent with the final version of the book. There may also be slight inaccuracies and errors. For all of these we apologize.

We hope that this instructor's manual will be an ongoing effort, with solutions added and improved over time. In particular, any instructor who wants to contribute material — new solutions to exercises, improvements or corrections to solutions provided, or even entirely new problems and solutions — please send an email to `pgm-solutions@cs.stanford.edu`, and I will add them to this instructor's manual (with attribution). Thank you.

1 Chapter 2

Exercise 2.7 — Independence properties [17 points] Note that $P(\boldsymbol{X}|\boldsymbol{Z}=\boldsymbol{z})$ is undefined if $P(\boldsymbol{Z}=\boldsymbol{z})=0$, so be sure to consider the definition of conditional independence when proving these properties.

1. [**7 points**] Prove that the Weak Union and Contraction properties hold for any probability distribution P.

   Weak Union:
   $$(\boldsymbol{X} \perp \boldsymbol{Y}, \boldsymbol{W} \mid \boldsymbol{Z}) \Longrightarrow (\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}, \boldsymbol{W})$$

   Answer: It is important to understand that the independence statements above hold for **all** specific values of the variables X, Y, Z, and W. Implicit in every such statement, and every probability statement such as $P(X,Y,W \mid Z) = P(X \mid Z) \cdot P(Y,W \mid Z)$, is the qualifier "for every possible possible of X, Y, Z, and W." See, for example, Def. 2.1.6 and the surrounding discussion.

   So our goal is to prove that the above statement holds for all possible values of the variables. We proceed by cases, first examining the cases in which the probabilities of some values may be 0, and then considering all other cases.

   If $P(W=w, Z=z)=0$, then $(X \perp Y \mid Z, W)$ holds trivially from Def. 2.1.4. Similarly, if $P(Z=z)=0 \Longrightarrow P(W=w,Z=z)=0$, so $(X \perp Y \mid Z, W)$ also holds.

   Now consider the case that $P(Z) \neq 0, P(Z,W) \neq 0$. We have

   $$\begin{aligned} P(X,Y \mid Z,W) &= \frac{P(X,Y,W \mid Z)}{P(W \mid Z)} \quad \text{[cond. indep rule, } P(W \mid Z) \neq 0 \text{ by assumption]} \\ &= \frac{P(X \mid Z) \cdot P(Y,W \mid Z)}{P(W \mid Z)} \quad \text{[since } (X \perp Y,W \mid Z)\text{]} \\ &= P(X \mid Z)P(Y \mid Z,W) \end{aligned}$$

   We can also use the Decomposition rule (2.8) $(X \perp Y,W \mid Z) \Longrightarrow (X \perp W, Z)$ to deduce $(X \perp W, Z) \Longleftrightarrow P(X \mid Z) = P(X \mid W, Z)$. Thus, we obtain

   $$P(X,Y \mid Z,W) = P(X \mid Z,W)P(Y \mid Z,W)$$

   which implies $(X \perp Y \mid Z, W)$.

   It's important to note that $(X \perp Y,W \mid Z)$ does not necessarily mean that $P(X,Y,W \mid Z) = P(X \mid Z)P(Y,W \mid Z)$; if $P(Z=z)=0$ for any $z \in Val(Z)$, then the conditional probabilities in the latter equations are undefined.

   Contraction:

   $$(\boldsymbol{X} \perp \boldsymbol{W} \mid \boldsymbol{Z}, \boldsymbol{Y}) \,\&\, (\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}) \Longrightarrow (\boldsymbol{X} \perp \boldsymbol{Y}, \boldsymbol{W} \mid \boldsymbol{Z}).$$

   Answer: Again, if $P(Z=z)=0$, $(X \perp Y,W \mid Z)$ holds trivially from the definition. If $P(Z=z) \neq 0$ but $P(Z=z, Y=y)=0$ you can check that $P(X,Y,W \mid Z) = 0 = P(X \mid Z)\ P(W,Y \mid Z)$.

Now assume $P(Z=z) \neq 0, P(Z=z, Y=y) \neq 0$. Then

$$\begin{aligned}
P(X,Y,W \mid Z) &= P(X,W \mid Z,Y)\ P(Y \mid Z) \\
&= P(X \mid Z,Y)\ P(W \mid Z,Y)\ P(Y \mid Z) \ \ \text{[since } (X \perp W \mid Z,Y)\text{]} \\
&= P(X \mid Z,Y)\ P(W,Y \mid Z) \\
&= P(X \mid Z)\ P(W,Y \mid Z) \ \ \text{[since } (X \perp Y \mid Z)\text{]}
\end{aligned}$$

which proves $(X \perp Y, W \mid Z)$.

This proves the Contraction rule for all distributions P.

Here, the common short proof was that $P(X \mid Y, W, Z) = P(X \mid Y, Z) = P(X \mid Z)$ where the first step follows from $(X \perp W \mid Z, Y)$ and the second step follow sfrom $(X \perp Y \mid Z)$. Again, this fails if $P(X \mid Y, W, Z)$ is undefined.

2. [**7 points**] Prove that the Intersection property holds for any positive probability distribution P. You should assume that $\boldsymbol{X}$, $\boldsymbol{Y}$, $\boldsymbol{Z}$, and $\boldsymbol{W}$ are disjoint.

   Intersection:

$$(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}, \boldsymbol{W})\,\&\,(\boldsymbol{X} \perp \boldsymbol{W} \mid \boldsymbol{Z}, \boldsymbol{Y}) \Longrightarrow (\boldsymbol{X} \perp \boldsymbol{Y}, \boldsymbol{W} \mid \boldsymbol{Z}).$$

   Answer: If we could prove $(X \perp Y \mid Z)$, we can use the Contraction rule to derive the desired conditional independence: $(X \perp W \mid Z,Y)$ & $(X \perp Y \mid Z) \Longrightarrow (X \perp Y, W \mid Z)$.

$$\begin{aligned}
P(X \mid Z) &= \sum_{w \in W} P(X, w \mid Z) = \sum_{w \in W} P(X \mid Z, w) P(w \mid Z) \\
&= \sum_{w \in W} P(X \mid Y, w, Z) P(w \mid Z) \quad \text{[since } (X \perp Y \mid Z, W)\text{]} \\
&= \sum_{w \in W} P(X \mid Y, Z) P(w \mid Z) \quad \text{[since } (X \perp W \mid Z, Y)\text{]} \\
&= P(X \mid Y, Z) \sum_{w \in W} P(w \mid Z) = P(X \mid Y, Z)
\end{aligned}$$

   This proves $(X \perp Y \mid Z)$, and by applying Contraction rule we are done. Note that if the distribution is non-positive, many of the conditional probabilities we use in the proof may be undefined.

3. [**3 points**] Provide a counter-example to the Intersection property in cases where the distribution P is not positive.

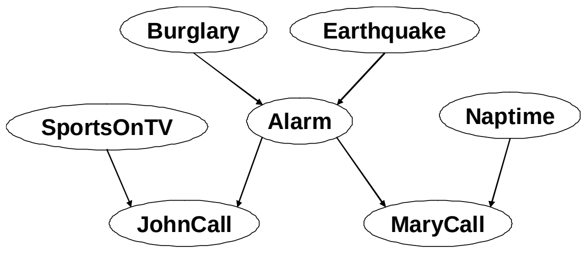
   Answer: Consider a distribution where $X = Y = W$ always holds, and their values are independent of the value of Z. Then $(X \perp W \mid Z, Y)$, since Y directly determines X, and similarly $(X \perp Y \mid Z, W)$. However, it is not the case that $(X \perp Y, W \mid Z)$ because knowing Z does not help us choose between the values $X = Y = W = 1$ or $X = Y = W = 0$.

   Note that the independence $(X \perp Y \mid Z)$ does not hold here. It is not the case that for any value of $Y = y$, $P(X \mid Z) = P(X \mid Y = y, Z)$.

Chapter 3

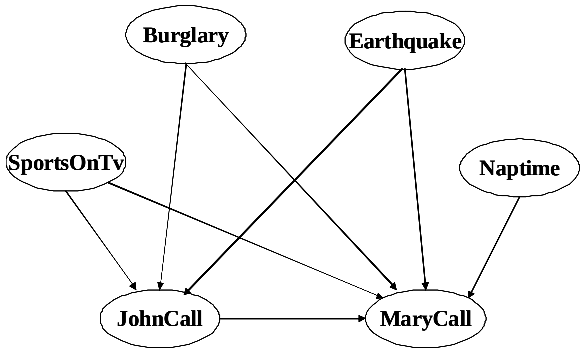
Exercise 3.11 — Marginalization and independencies [24 points]

1. [**6 points**] Consider the alarm network shown below:



Construct a Bayesian network structure with nodes Burglary, Earthquake, JohnCall, MaryCall, SportsOnTV, and Naptime, which is a minimal I-map for the marginal distribution over those variables defined by the above network. Be sure to get *all* dependencies that remain from the original network.

Answer:



In order to construct a minimal I-map, we would like to preserve all independencies (assuming Alarm is always unobserved) that were present in the original graph, without adding any unnecessary edges. Let's start with the remaining nodes and add edges only as needed.

We see that with Alarm unobserved, there exist active paths between Alarm's direct ancestors and children. Thus, direct edges between the parents, Burglary and Earthquake, should be added to connect to both children, JohnCall and Mary Call. Similarly, since any two children of Alarm also now have an active path between them, a direct edge between JohnCall and MaryCall should be added. Without loss of generality, we direct this edge to go from JohnCall to MaryCall.

Next, since SportsOnTv and JohnCall as well as Naptime and MaryCall were directly connected in the original graph, removing Alarm doesn't affect their dependencies and the two edges must be preserved.

Now we must consider any independencies that may have changed. In the old graph, because of the v-structure between Alarm and co-parent SportsOnTv, if Alarm was unobserved and JohnCall observed, there existed an active path between SportsOnTv and MaryCall. In the new graph however, because of the added direct edge between the two children JohnCall and

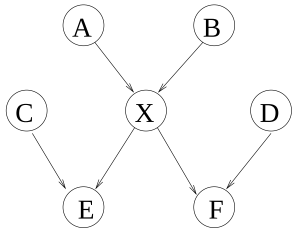
MaryCall, if JohnCall is observed, the path between SportsOnTv and MaryCall is actually rendered inactive. Thus, an alternate path that does not introduce any other dependencies needs to be introduced, and a direct edge is added between SportsOnTv and MaryCall.

Common mistakes:

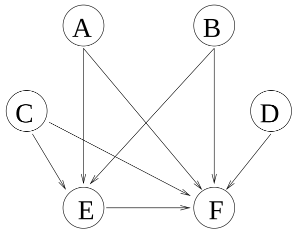
(a) **[3 points]** Missing edge from SportsOnTv to MaryCall (or Naptime to JohnCall).
(b) **[3 points]** Missing edge from JohnCall to MaryCall (or MaryCall to JohnCall).
(c) **[1 points]** Includes any additional edges not needed for preserving independencies.

2. [**18 points**] Generalize the procedure you used above to an arbitrary network. More precisely, assume we are given a network BN, an ordering $X_1, \ldots, X_n$ that is consistent with the ordering of the variables in BN, and a node X_i to be removed. Specify a network BN' such that BN' is consistent with this ordering, and such that BN' is a minimal I-map of $P_{\mathcal{B}}(X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n)$. Your answer must be an explicit specification of the set of parents for each variable in BN'.

Answer:



is transformed into



A general node elimination algorithm goes as follows. Suppose we want to remove X from BN. This is similar to skipping X in the I-map construction process. As the distribution in question is the same except for marginalizing X, the construction process is the same until we reach the removed node.

Suppose the algorithm now adds the first direct descendent of X, which we'll call E. What arcs must be added? Well, all the original parents of E must be added (or we'd be asserting incorrect independence assumptions). But how do we replace the arc between X and E? As before, we must add arcs between the parents of X and E — this preserves the v-structure d-separation between X and its parents as well as the dependence of E on the parents of X.

Now suppose we add another direct descendant, called F. Again, all the original parents of F must be added and we also connect the parents of X to F. Is that all? No, we must also connect E to F, in order to preserve the dependence that existed in the original graph (as an active path existed between E and F through X). Now is that all? No. Notice that if E, is observed then there is a active path between C and F. But this path is blocked in our new graph if all other parents of F (E, A, B, D) are observed. Hence, we have to add and edge between C and F.

So for every direct descendant of X, we add arcs to it from the parents of X, from the other direct descendants of X previously added and from the parents of the previously added direct descendants of X.

What about the remaining nodes in the ordering? No changes need to be made for the arcs added for these nodes: if a node X_m was independent of $X_1, \ldots, X_{m-1}$ (including X), given its parents, it is also independent of $\{X_1, \ldots, X_{m-1}\} - \{X\}$ given its (same set of) parents. Guaranteeing that the local Markov assumptions hold for all variables is sufficient to show that the new graph has the required I-map properties. The following specification captures this procedure.

Let $\mathcal{T} = \{X_1, \ldots, X_n\}$ be the topological ordering of the nodes and let $C = \{X_j \mid X_j \in \mathit{Children}_{X_i}\}$ be the set of children of X_i in BN. For each node X'_j in BN' we have the following parent set:

$$\mathbf{Pa}'_{X_j} = \begin{cases} \mathbf{Pa}_{X_j} & \text{for all } X_j \notin C \\ \mathbf{Pa}_{X_j} \cup \mathbf{Pa}_{X_i} \cup \{X_k, \mathbf{Pa}_{X_k} \mid X_k \in C, k < j\} \setminus X_i & \text{for all } X_j \in C \end{cases}$$

Common mistakes

(a) **[15 points]** Using the I-map construction algorithm or d-separation tests in order to construct the new graph. The problem asks for a *transformation* of the old graph into a new one, but doesn't ask you to construct the new graph from scratch. Essentially, we were looking for an efficient algorithm based solely on the structure of the graph, not one based on expensive d-separation queries.

(b) **[5 points]** Forgetting that the new graph should maintain consistent ordering with the new graph.

(c) **[5 points]** Forgetting original parent set.

(d) **[4 points]** Forgetting edges from $\mathbf{Pa}_{X_i}$ to X_j.

(e) **[4 points]** Forgetting edges from $X_k, k < j$ to X_j.

(f) **[4 points]** Forgetting edges from $\mathbf{Pa}_{X_k}, k < j$ to X_j.

(g) **[2 points]** Adding any additional edges not needed to satisfy I-map requirements.

(h) **[1 points]** Not writing explicit parent sets.

Exercise 3.16 — same skeleton and v-structures $\Rightarrow$ I-equivalence [16 points]

1. [**12 points**] Prove theorem 3.7: Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two graphs over $\mathcal{X}$. If they have the same skeleton and the same set of v-structures then they are I-equivalent.

   Answer: It suffices to show:

   (a) $\mathcal{G}_1$ and $\mathcal{G}_2$ have the same set of trails.

   (b) a trail is active in $\mathcal{G}_1$ iff it is active in $\mathcal{G}_2$.

We show that if the trail from A to C is an active trail in $\mathcal{G}_1$, then it is an active trail in $\mathcal{G}_2$. First note that because $\mathcal{G}_1$ and $\mathcal{G}_2$ have the same structure, if there is a trail A to C is $\mathcal{G}_1$, there is a trail A to C in $\mathcal{G}_2$.

Let X be a node on the active trail from A to C in $\mathcal{G}_1$. Let Y and Z be the neighbors of X on the trail. There are two cases to consider:

(a) (Y, X, Z) is not a v-structure in $\mathcal{G}_1$. Then if this subtrail is active, X is not set. Because $\mathcal{G}_2$ has the same v-structures as $\mathcal{G}_1$, we know (Y, X, Z) is not a v-structure in $\mathcal{G}_2$. Therefore we know we have one of the following three in $\mathcal{G}_2$: $Y \rightarrow X \rightarrow Z$, $Y \leftarrow X \rightarrow Z$ and $Y \leftarrow X \leftarrow Z$. In all three cases, if X is not set, the subtrail is also active in $\mathcal{G}_2$.

(b) (Y, X, Z) is a v-structure in $\mathcal{G}_1$. Then if this trail is active either X is set or one of its descendants is set. If X is set in $\mathcal{G}_1$, then because $\mathcal{G}_2$ has the same v-structure as $\mathcal{G}_1$, the subtrail will be active in $\mathcal{G}_2$. Suppose it is active through some node X' that is a descendant of X in $\mathcal{G}_1$, but is not a descendant of X in $\mathcal{G}_2$. Since $\mathcal{G}_1$ and $\mathcal{G}_2$ have the same skeleton, there is a trail from X to X' in $\mathcal{G}_2$. Since X' is not a descendent of X in $\mathcal{G}_2$, there must be some edge that does not point toward X' on the trail from X to X' in $\mathcal{G}_2$. This is a v-structure in $\mathcal{G}_2$ that does not exist in $\mathcal{G}_1$. We have a contradiction. Therefore, the subtrail will be active in $\mathcal{G}_2$.

We can use a similar argument to show that any active trail in $\mathcal{G}_2$ is active in $\mathcal{G}_1$.

The most common error in this problem was to forget that observing descendants of a v-structure middle node X can activate trails passing through X. Also, we need to prove that if a node is a descendant of X (the node in the middle of the v-structure) in $\mathcal{G}_1$, it must also be a descendant of this node in $\mathcal{G}_2$.

Common mistakes:

(a) [**6 points**] One of the most common mistakes on this problem was to forget to address the possibility of a descendant activating a v-structure.

(b) [**1 points**] Minor logical leap, informality, or ambiguity in the proof.

(c) [**3 points**] Although slightly less common, some made the mistake of not addressing the case when a subtrail is not a v-structure. This could be because the proof dealt with the issue at too abstract a level (not dealing with active three-node subtrails), or because it was simply missed entirely.

2. [**4 points**] How many networks are equivalent to the simple directed chain $X_1 \rightarrow X_2 \rightarrow \ldots \rightarrow X_n$?

Answer: The result in the previous part holds only in one direction: same skeleton and same set of v-structures guarantees I-equivalent networks. However, the reverse does not always hold. Instead, we will use Theorem 3.3.13, which says that two I-equivalent graphs must have the same skeleton and the same set of immoralities.

The first requirement constrains the possible edges to the original edges in the chain. Moreover, all original edges have to be present, because removing any edge breaks the set of nodes into two non-interacting subsets.

The second requirement limits us to networks which have no immoralities, and no v-structures (because the original chain has none). There are a total of $n-1$ such networks, excluding the original chain, where we are allowed a single structure $X_{i-1} \leftarrow X_i \rightarrow X_{i+1}$ in the chain (two such subgroups would create a v-structure). The $n-1$ possibilities come from,aligning this structure on top of any node $X_2, \ldots, X_n$, with the rest of the edges propagating away from node X_i.

Common mistakes:

(a) **[1 points]** Off by one. (Note that both n and (n-1) were acceptable answers)
(b) **[4 points]** Missed this part entirely. Answer way off of (n-1) or n. Common answers here are 2 and 2^n.

Exercise 3.20 — requisite CPD [15 points] In this question, we will consider the sensitivity of a particular query $P(X \mid \boldsymbol{Y})$ to the CPD of a particular node Z. Let X and Z be nodes, and $\boldsymbol{Y}$ be a set of nodes. Provide a sound and "complete" criterion for determining when the result of the query $P(X \mid \boldsymbol{Y})$ is not affected by the choice of the CPD $P(Z \mid \mathbf{Pa}_Z)$. More precisely, consider two networks $\mathcal{B}_1$ and $\mathcal{B}_2$ that have identical graph structure G and identical CPDs everywhere except at the node Z. Provide a transformation G' of G such that we can test whether $P_{\mathcal{B}_1}(Z \mid \mathbf{Pa}_Z) = P_{\mathcal{B}_2}(Z \mid \mathbf{Pa}_Z)$ using a single d-separation query on G'. Note that Z may be the same as X, and that Z may also belong to $\boldsymbol{Y}$. Your criterion should always be sound, but only be complete in the same sense that d-separation is complete, i.e., for "generic" CPDs.

Hint: Although it is possible to do this problem using laborious case analysis, it is significantly easier to think of how a d-separation query on a transformed graph G' can be used to detect whether a perturbation on the CPD $P(Z \mid \mathbf{Pa}_Z)$ makes a difference to $P(X \mid \boldsymbol{Y})$.

Answer:

Construct a new network G' from the original network G by adding one node W and one edge from W to Z, i.e., W is an extra parent of Z. We set the CPD for Z so that if $W = w^1$, then Z behaves as in $\mathcal{B}_1$, and if $W = w^2$, then Z behaves as in $\mathcal{B}_2$. More precisely, $P(Z \mid \mathbf{Pa}_Z, w^1) = P_{\mathcal{B}_1}(Z \mid \mathbf{Pa}_Z)$, and $P(Z \mid \mathbf{Pa}_Z, w^2) = P_{\mathcal{B}_2}(Z \mid \mathbf{Pa}_Z)$. It is therefore clear that $P_{\mathcal{B}_1}(X \mid \boldsymbol{Y}) = P_{\mathcal{B}_2}(X \mid \boldsymbol{Y})$ if and only if $P_{\mathcal{B}'}(X \mid \boldsymbol{Y}, w^1) = P_{\mathcal{B}'}(X \mid \boldsymbol{Y}, w^2)$, where $\mathcal{B}'$ is our modified network with W. We can guarantee that the value of W doesn't matter precisely when W is d-separated from X given $\boldsymbol{Y}$. This criterion is always sound, because when W is d-separated, the choice of CPD for Z cannot affect the query $P(X \mid \boldsymbol{Y})$. It is complete in the same sense that d-separation is complete: There might be some configuration of CPDs in the network where the CPD of Z happens not to make a difference, although in other circumstances it could. This criterion will not capture cases of this type.

It was possible to earn full points for a case-based solution. The two important points were

1. Unless Z or one of its descendents is instantiated, active paths which reach Z from X through one of Z's parents are not "CPD-active"; that is, changing the CPD of Z will not affect X along this path.
2. The case where Z is in Y needs to be handled, since our definition of d-sep means that if Z is in Y, d-sep(X;Z—Y), yet paths which reach Z through one of its parents are still "CPD-active".

Solutions which used the case-based approach were scored based on how they handled these two issues.

Common mistakes:

1. **[11 points]** Answer $dsep(X; Z|Y)$ or anything else which doesn't handle either observed Z or cases where Z does/does not have observed descedents
2. **[3 points]** Correct answer, but not fully justified (usually incomplete case analysis)

3. **[5 points]** Don't handle cases when Z is observed (in the case-based analysis)
4. **[1 points]** Minor miscellaneous error
5. **[6 points]** Don't correctly deal with case where Z has no observed descendents
6. **[3 points]** Medium miscellaneous error

Chapter 4

Exercise 4.4 — Canonical parameterization

Prove theorem 4.7 for the case where $\mathcal{H}$ consists of a single clique.

Answer: Our log probability is defined as:

$$\ln \tilde{P}(\xi) = \sum_{\boldsymbol{D}_i \subseteq \mathcal{X}} \epsilon_{\boldsymbol{D}_i}[\xi].$$

Substituting in the definition of $\epsilon_{\boldsymbol{D}_i}[\xi]$, we get

$$\begin{aligned} \ln \tilde{P}(\xi) &= \sum_{\boldsymbol{D}_i} \sum_{\boldsymbol{Z} \subseteq \boldsymbol{D}_i} (-1)^{|\boldsymbol{D}_i - \boldsymbol{Z}|} \ell(\sigma_{\boldsymbol{Z}}[\xi]) \\ &= \sum_{\boldsymbol{Z}} \ell(\sigma_{\boldsymbol{Z}}[\xi]) \sum_{\boldsymbol{D}_i \supseteq \boldsymbol{Z}} (-1)^{|\boldsymbol{D}_i - \boldsymbol{Z}|}. \end{aligned}$$

For all $\boldsymbol{Z} \neq \mathcal{X}$, the number of subsets containing $\boldsymbol{Z}$ is even, with exactly half of them having an odd number of additional elements and the other half having an even number of additional elements. Hence, the internal summation is zero except for $\boldsymbol{Z} = \mathcal{X}$, giving:

$$\ln \tilde{P}(\xi) = \ell(\sigma_{\boldsymbol{Z}}[\xi]) = \ln P(\xi),$$

proving the desired result.

Exercise 4.8 — Pairwise to Markov independencies Prove proposition 4.3. More precisely, let P satisfy $\mathcal{I}_\ell(\mathcal{H})$, and assume that X and Y are two nodes in $\mathcal{H}$ that are not connected directly by an edge. Prove that P satisfies $(X \perp Y \mid \mathcal{X} - \{X, Y\})$.

Answer: This result follows easily from the Weak Union property. Let $\boldsymbol{Z} = \mathcal{N}_{\mathcal{H}}(X)$ and let $\boldsymbol{W} = \mathcal{X} - \boldsymbol{Z} - \{X, Y\}$. Then the Markov assumption for X tells us that

$$(X \perp \{Y\} \cup \boldsymbol{W} \mid \boldsymbol{Z}).$$

From the Weak Union property, it follows that

$$(X \perp Y \mid \boldsymbol{Z} \cup \boldsymbol{W}),$$

which is precisely the desired property.

Exercise 4.9 — Positive to global independencies

Complete the proof of theorem 4.4. Assume that equation (4.1) holds for all disjoint sets $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$, with $|\boldsymbol{Z}| \geq k$. Prove that equation (4.1) also holds for any disjoint $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ such that $\boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{Z} \neq \mathcal{X}$ and $|\boldsymbol{Z}| = k - 1$.

Answer: Because $\boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{Z} \neq \mathcal{X}$, there exists at least one node A that is not in $\boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{Z}$. From the monotonicity of graph separation, we have that $\textit{sep}_{\mathcal{H}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z} \cup \{A\})$.

Because $\boldsymbol{X}$ and $\boldsymbol{Y}$ are separated given $\boldsymbol{Z}$, there cannot both be a path between $\boldsymbol{X}$ and A given $\boldsymbol{Z}$ and between $\boldsymbol{Y}$ and A given $\boldsymbol{Z}$. Assume, without loss of generality, that there is no path between $\boldsymbol{X}$ and A given $\boldsymbol{Z}$. By monotonicity of separation, there is also no path between $\boldsymbol{X}$ and A given $\boldsymbol{Z} \cup \boldsymbol{Y}$. We then have that $\textit{sep}_{\mathcal{H}}(\boldsymbol{X}; A \mid \boldsymbol{Z} \cup \boldsymbol{Y})$.

The sets $\boldsymbol{Z} \cup \{A\}$ and $\boldsymbol{Z} \cup \boldsymbol{Y}$ have size at least k. Therefore, the inductive hypothesis equation (4.1) holds, and we conclude

$$(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z} \cup \{A\}) \quad \& (\boldsymbol{X} \perp A \mid \boldsymbol{Z} \cup \boldsymbol{Y}).$$

Because P is positive, we can apply the intersection property (equation (2.11)) and conclude that P satisfies $(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$, as required.

Exercise 4.11 — Minimal I-map [15 points]

In this exercise you will prove theorem 4.6. Consider some specific node X, and let $\mathcal{U}$ be the set of all subsets $\boldsymbol{U}$ satisfying $P \models (X \perp \mathcal{X} - \{X\} - \boldsymbol{U} \mid \boldsymbol{U})$. Define $\boldsymbol{U}^*$ to be the intersection of all $\boldsymbol{U} \in \mathcal{U}$.

1. [**8 points**] Prove that $\boldsymbol{U}^* \in \mathcal{U}$. Conclude that $MB_P(X) = \boldsymbol{U}^*$

   Answer: This follows directly from the intersection property. Specifically, let $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$ be two sets in $\mathcal{U}$, that is, $P \models (X \perp \mathcal{X} - \{X\} - \boldsymbol{U}_i \mid \boldsymbol{U}_i)$. Define $\boldsymbol{Y}_i = \boldsymbol{U}_i - \boldsymbol{U}^*$; it now follows that

   $$\begin{aligned} P &\models (X \perp \boldsymbol{Y}_1 \mid \boldsymbol{U}_2) \\ P &\models (X \perp \boldsymbol{Y}_2 \mid \boldsymbol{U}_1) \end{aligned}$$

   Equivalently,

   $$\begin{aligned} P &\models (X \perp \boldsymbol{Y}_1 \mid \boldsymbol{U}^* \cup \boldsymbol{Y}_2) \\ P &\models (X \perp \boldsymbol{Y}_2 \mid \boldsymbol{U}^* \cup \boldsymbol{Y}_1) \end{aligned}$$

   From the intersection property, it follows that

   $$P \models (X \perp \boldsymbol{U}^* \mid \boldsymbol{Y}_1, \boldsymbol{Y}_2).$$

   All that remains is to show that $\boldsymbol{U}^* = MB_P(X)$. Recall that $MB_P(X)$ is defined to be the minimal set of nodes $\boldsymbol{U}$ such that $P \models (X \perp \mathcal{X} - \{X\} - \boldsymbol{U} \mid \boldsymbol{U})$. This definition implies that $MB_P(X)$ is the minimal set of nodes $\boldsymbol{U} \in \mathcal{U}$. We see that this is indeed $\boldsymbol{U}^*$.

   A common mistake is to use a form of the intersection property which isn't explicitly proven. One such form is $(\boldsymbol{X} \perp \boldsymbol{A}, \boldsymbol{Y} \mid \boldsymbol{W}, \boldsymbol{Z})$ and $(\boldsymbol{X} \perp \boldsymbol{A}, \boldsymbol{W} \mid \boldsymbol{Y}, \boldsymbol{Z}) \longrightarrow (\boldsymbol{X} \perp \boldsymbol{A}, \boldsymbol{Y}, \boldsymbol{W} \mid \boldsymbol{Z})$. This is true, but requires a Weak Union step for completeness. Another common mistake in the first three parts of the problem was to use the monotonic property of conditional independence relations induced by a graph $\mathcal{H}$ to justify adding variables to the right side of a conditional. We must remember, however, that in these parts of the problem we are dealing only with a distribution that may not correspond to a graph.

2. [**2 points**] Prove that if $P \models (X \perp Y \mid \mathcal{X} - \{X, Y\})$ then $Y \notin MB_P(X)$.

   Answer: We see that we can rewrite this independence as $P \models (X \perp \mathcal{X} - \{X\} - \boldsymbol{U}' \mid \boldsymbol{U}')$ where $\boldsymbol{U}' = \mathcal{X} - \{X, Y\}$. But $\boldsymbol{U}' \in \mathcal{U}$ and $Y \notin \boldsymbol{U}'$, thus Y cannot be in the intersection of all $\boldsymbol{U} \in \mathcal{U}$. This means $Y \notin \boldsymbol{U}^*$ or equivalently $Y \notin MB_P(X)$.

3. [**3 points**] Prove that if $Y \notin MB_P(X)$ then $P \models (X \perp Y \mid \mathcal{X} - \{X, Y\})$.

   Answer: We know that $P \models (X \perp \mathcal{X} - \{X\} - MB_P(X) \mid MB_P(X))$. If $Y \notin MB_P(X)$ then we must have $Y \in \mathcal{X} - \{X\} - MB_P(X)$, allowing us to rewrite the independency as $P \models (X \perp \{Y\}, \mathcal{X} - \{X, Y\} - MB_P(X) \mid MB_P(X))$. By weak union we then have $P \models (X \perp \{Y\} \mid \mathcal{X} - \{X, Y\} - MB_P(X), MB_P(X))$ or just $P \models (X \perp Y \mid \mathcal{X} - \{X, Y\})$.

4. [**2 points**] Conclude that $MB_P(X)$ is precisely the set of neighbors of X in the graph defined in theorem 4.5, showing that the construction of theorem 4.6 also produces a minimal I-map.

   Answer: (b) shows that $MB_P(X)$ is a subset of the neighbors defined in the graph of Theorem 4.5, while (c) shows that it is a superset.

Exercise 4.15 — Markov network proof of d-separation [22 points] Prove Proposition 4.10: Let $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ be three disjoint sets of nodes in a Bayesian network $\mathcal{G}$. Let $\boldsymbol{U} = \boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{Z}$, and let $\mathcal{G}' = \mathcal{G}^+[\boldsymbol{U}]$ be the induced Bayesian network over $\mathit{Ancestors}(\boldsymbol{U}) \cup \boldsymbol{U}$. Let $\mathcal{H}$ be the moralized graph $\mathcal{M}[\mathcal{G}']$. Prove that $\textit{d-sep}_{\mathcal{G}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z})$ if and only if $\mathit{sep}_{\mathcal{H}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z})$.

Answer:

[11 points] Suppose not $\textit{d-sep}_{\mathcal{G}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z})$. Then there is an active path in $\mathcal{G}$ between some $X \in \boldsymbol{X}$ and some $Y \in \boldsymbol{Y}$. The active path, like any path, is formed of overlapping unidirectional segments $W_1 \to \ldots \to W_n$. Note that W_n in each segment is either X, Y, or the base of some active v-structure. As such, either W_n or one of its descendants is in $\boldsymbol{U}$, and so all nodes and edges in teh segment $W_1 \to \ldots \to W_n$ are in the induced graph over *Ancestors*$(\boldsymbol{U}) \cup \boldsymbol{U}$. So the active path in $\mathcal{G}$ is a path in $\mathcal{H}$, and in $\mathcal{H}$, the path can only contain members of $\boldsymbol{Z}$ at the bases of v-structures in $\mathcal{G}$ (otherwise, the path would have been active in $\mathcal{G}$). Because $\mathcal{H}$ is the moralized graph $\mathcal{M}[\mathcal{G}']$, we know that those members of $\boldsymbol{Z}$ have been bypassed: their parents in the v-structure must have an edge between them in $\mathcal{H}$. So there is an active path between X and Y in $\mathcal{H}$, so not $\mathit{sep}_{\mathcal{H}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z})$.

[11 points] Now we prove the converse. Suppose that $\boldsymbol{X}$ and $\boldsymbol{Y}$ are d-separated given $\boldsymbol{Z}$. Consider an arbitrary path in $\mathcal{G}$ between some $X \in \boldsymbol{X}$ and some $Y \in \boldsymbol{Y}$. Any path between X and Y in $\mathcal{G}$ must either be blocked by a member of $\boldsymbol{Z}$ or an inactive v-structure in $\mathcal{G}$. First, suppose the path is blocked by a member of $\boldsymbol{Z}$. Then the path in $\mathcal{H}$ (if it exists–it may not because $\mathcal{H}$ is the induced graph) will also be blocked by that member of $\boldsymbol{Z}$.

Of course, if the path between X and Y in $\mathcal{G}$ is not blocked by a member of $\boldsymbol{Z}$, it must be blocked by an inactive v-structure. Because the v-structure is inactive, its base (and any of its descendants) must not be in the induced graph $\mathcal{H}$. As such, the path will not exist in $\mathcal{H}$. Neither the base nor its descendants can be in $\boldsymbol{Z}$, and they cannot be in $\boldsymbol{X}$ or $\boldsymbol{Y}$ either, because then we would have an active path from a member of $\boldsymbol{X}$ to a member of $\boldsymbol{Y}$.

Recall that the edges added in moralization are necessary to create an active path in $\mathcal{H}$ only when paths would be blocked by the observed root node of a v-structure in $\mathcal{G}$. In this case, the segment would have been active in $\mathcal{G}$, so moralization edges cannot effect segments in $\mathcal{H}$ corresponding to inactive segments in $\mathcal{G}$; this is what our proof depends on. Because of all the above, there are no active paths in $\mathcal{H}$ between arbitrary $X \in \boldsymbol{X}$ and $Y \in \boldsymbol{Y}$, so we have $\mathit{sep}_{\mathcal{H}}(\boldsymbol{x}; \boldsymbol{Y} \mid \boldsymbol{Z})$.

Common mistakes:

D-separation to separation

1. [**4 points**]Failed to show that root node of an inactive v-structure in $\mathcal{G}$ will not be in $\mathcal{H}$, or that an active path in H through an inactive v-structure is not possible.
2. [**3 points**] Failed to show modifications necessary to handle moralized edges.
3. [**1 points**] Considered only paths existing in G, failed to consider possibility of entirely new paths in $\mathcal{H}$.
4. [**1 points**] Described only 3-node paths without demonstrating how the cases combine into paths of arbitrary length.
5. [**1 points**] Assumed that for active v-structures, center node must be observed, didn't consider its descendants.
6. [**3 points**] Other significant logical error or omission.
7. [**1 points**] Other minor logical error/omission.

Separation to d-separation

1. [**4 points**] Failed to demonstrate that all the nodes along an active path in G will still appear in $\mathcal{H}$.
2. [**3 points**] Failed to show that an active v-structure in G will not result in blockage in $\mathcal{H}$.
3. [**2 points**] Describe only 3-node paths without describing how these combine into arbitrary paths, in particular, why those nodes will all still appear in $\mathcal{H}$.
4. [**1 points**] Assumed that for active v-structures, center node must be observed, didn't consider its descendants.
5. [**3 points**] Other significant logical error or omission.
6. [**1 points**] Other minor logical error/omission.

Exercise 4.18 — Markov networks and Bayesian networks Let G be a Bayesian network with no immoralities. Let H be a Markov network with the same skeleton as G. Show that H is an I-map of $\mathcal{I}(G)$.

Answer: We need to prove that $\mathcal{I}(H) \subset \mathcal{I}(G)$; that is, if $(X \perp Y \mid \boldsymbol{Z}) \in \mathcal{I}(H)$ then $(X \perp Y \mid \boldsymbol{Z}) \in \mathcal{I}(G)$.

Proving the contrapositive: if $(X \perp Y \mid \boldsymbol{Z}) \notin \mathcal{I}(G)$ then $(X \perp Y \mid \boldsymbol{Z}) \notin \mathcal{I}(H)$. Suppose $(X \perp Y \mid \boldsymbol{Z}) \notin \mathcal{I}(G)$. Then there is an active trail τ from X to Y given $\boldsymbol{Z}$ in G. Consider a triplet $A - B - C$ on t. If it is a v-structure, then by the definition of active trails, B must be observed ($B \in \boldsymbol{Z}$). Then by the definition of active trails, for all triplets $A - B - C$ on t, it must be the case that $A - B - C$ is not a v-structure either it's not a v-structure and $A, B, C \notin \boldsymbol{Z}$ or it is a v-structure and $B \in \boldsymbol{Z}$. If

Let τ be an active trail from some node X to some node Y in G given a set of nodes E. Then we can find an active trail from X to Y in G given E that contains no v-structures by the following recursive process:

1. If τ contains no v-structures, then return T.
2. If τ contains a v-structure, then let $A \rightarrow B \leftarrow C$ be the v-structure on T closest to X. Because G contains no immoralities, there must be an edge from A to C. Let trail τ' be equal to τ.
   (a) If $A = X$, then replace the edges $A \rightarrow B \leftarrow C$ with $A \rightarrow C$ in τ'. τ' has exactly one less v-structure than
   (b) Otherwise, if the incoming edge to A on τ' is $\rightarrow$, then replace the edges $A \rightarrow B \leftarrow C$ with $A \rightarrow C$ in τ'.
   (c) Otherwise, if the incoming edge to A on τ' is $\leftarrow$, then replace the edges $A \rightarrow B \leftarrow C$ with $A \leftarrow C$ in τ'.

Chapter 5

Exercise 5.5 — tree-structured factors in a Markov network [15 points]

Instead of using a table to represent the CPD in a Bayesian network we can also use a CPD-tree.

1. [**3 points**] How can we use a tree in a similar way to represent a factor in a Markov network? What do the values at the leaves of such a tree represent?

   Answer: We use a tree in exactly the same way it is used to represent CPDs. The difference is that the leaves now store only one value — the factor entry associated with the context of the leaf, rather than a distribution.

2. [**6 points**] Given a context $U = u$, define a simple algorithm that takes a tree factor $\phi(Y)$ (from a Markov network) and returns the reduced factor $\phi_{[u]}(Y - U)$ (see definition 5.2.7), also represented as a tree.

   Answer: Traverse the tree in any order (DFS, BFS, inorder). For every node $X \in U$ connect the parent of X with the one child that is consistent with the context u. Delete X, its children that were non consistent with u, and their subtrees.

3. [**6 points**] In some cases it turns out we can further reduce the scope of the resulting tree factor. Give an example and specify a general rule for when a variable that is not in U can be eliminated from the scope of the reduced tree-factor.

   Answer: In the algorithm above, in addition to removing from the tree all the nodes in U, we also deleted entire subtrees. It might be that by doing that we also removed from the tree all instances of some other variable Z which is not in U. If this is the case, we can safely remove Z from the scope of the reduced tree factor, as its value no-longer depends on the instantiation of Z. In general, we can remove from the scope of a tree-factor all the variables that do not appear in it.

   Many students gave a slightly different answer. They said that an additional variable can be removed from the *tree* (and hence from the scope) if in the resulting reduced factor its children sub-trees are equivalent. This answer is correct, but it requires further changes to the tree, and in addition it depends on the actual values of the factor in the leaves. Conversly, in the case for Z above, we can safely change the scope of the reduced factor without making any further changes to the tree, and without considering any specific factor values.

Exercise 5.14 — Simplifying BN2O networks [22 points]

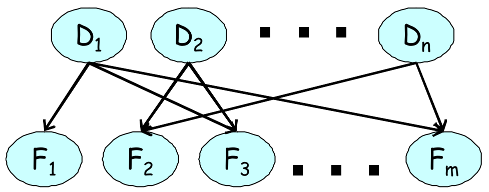


Figure 1: A two-layer noisy or network

Consider the network shown in Figure 1, where we assume that all variables are binary, and that the F_i variables in the second layer all have noisy or CPDs (You can find more details about this network in Concept 4.3 on page 138). Specifically, the CPD of F_i is given by:

$$P(f_i^0 \mid \mathbf{Pa}_{F_i}) \quad = \quad (1-\lambda_{i,0}) \prod_{D_j \in \mathbf{Pa}_{F_i}} (1-\lambda_{i,j})^{d_j}$$

where $\lambda_{i,j}$ is the noise parameter associated with parent D_j of variable F_i. This network architecture, called a *BN2O network* is characteristic of several medical diagnosis applications, where the D_i variables represent diseases (e.g., flu, pneumonia), and the F_i variables represent medical findings (e.g., coughing, sneezing).

Our general task is medical diagnosis: We obtain evidence concerning some of the findings, and we are interested in the resulting posterior probability over some subset of diseases. However, we are only interested in computing the probability of a particular subset of the diseases, so that we wish (for reasons of computational efficiency) to remove from the network those disease variables that are not of interest at the moment.

1. **[15 points]** Begin by considering a particular variable F_i, and assume (without loss of generality) that the parents of F_i are $D_1, \ldots, D_k$, and that we wish to maintain only the parents $D_1, \ldots, D_\ell$ for $\ell < k$. Show how we can construct a new *noisy or* CPD for F_i that preserves the correct joint distribution over $D_1, \ldots, D_\ell, F_i$.

   Answer: This can be done by summing out the D_j variables that we wish to remove and incorporating their effects into the leak noise parameter. We show how to remove a single parent D_k:

$$\begin{aligned} P(F_i = f_i^0 \mid D_1, \ldots D_{k-1}) &= \sum_{D_k} P(F_i = f_i^0, D_k \mid D_1, \ldots, D_{k-1}) \\ &= \sum_{D_k} P(F_i = f_i^0 \mid D_1, \ldots, D_{k-1}, D_k) P(D_k \mid D_2, \ldots D_{k-1}) \\ &= \sum_{D_k} P(F_i = f_i^0 \mid D_1, \ldots D_k) P(D_k) \\ &= \sum_{D_k} (1-\lambda_{i,0}) \left(\prod_{j=1}^{k} (1-\lambda_{i,j})^{D_j} \right) P(D_k) \\ &= (1-\lambda_{i,0}) \left(\prod_{j=1}^{k-1} (1-\lambda_{i,j})^{D_j} \right) \sum_{D_k} (1-\lambda_{i,k})^{D_k} P(D_k) \end{aligned}$$

   Thus, we could replace the previous leak parameter $\lambda_{i,0}$ by a new leak parameter $\lambda'_{i,0} = 1 - (1-\lambda_0) \sum_{D_k} (1-\lambda_{i,k})^{D_k} P(D_k)$ and we have maintained the noisy OR structure of the F_i CPD. This process can then be repeated sequentially for $D_{\ell+1}, \ldots, D_{k-1}$. Some of you noted that $\sum_{D_k} (1-\lambda_{i,k})^{D_k} P(D_k) = (1 - P(d_k^1)) + (1-\lambda_{i,k}) P(d_k^1) = 1 - \lambda_{i,k} P(d_k^1)$ which was fine as well.

   If we repeat this process for all the variables that we wish to remove, we come to the solution:

$$P(F_i = f_i^0 \mid D_1, \ldots D_l) \quad = \quad (1-\lambda_{i,0}) \left(\prod_{j=1}^{l} (1-\lambda_{i,j})^{D_j} \right) \prod_{m=l+1}^{k} \left[\sum_{D_m} (1-\lambda_{i,m})^{D_m} P(D_m) \right]$$

producing a final leak parameter $\lambda'_{i,0} = 1 - (1 - \lambda_{i,0}) \prod_{m=l+1}^{k} \left[\sum_{D_m} (1 - \lambda_{i,m})^{D_m} P(D_m)\right]$.

Common errors:

(a) **[3 points]** Did not show that the transformation results in a noisy-or network. To receive credit for this, you needed to explicitly show the relationship between the λ' and the expressions derived above.
(b) **[3 points]** Minor algebraic error: the final form of the answer for the new leak parameter was not exactly correct but only due to a small algebraic error in the derivation.
(c) **[7 points]** Major algebraic error.
(d) **[12 points]** More major error.

If you had a major algebraic error, you most likely lost 6 to 8 points on this problem. The most common major algebraic errors was the use of the following "identity": $P(F_1 \mid D_1, \ldots, D_{k-1}) = \sum_{D_k} P(F_1 \mid D_1, \ldots, D_k)$

2. [**7 points**] We now remove some fixed set of disease variables D from the network, executing this pruning procedure for all the finding variables F_i, removing all parents $D_j \in D$. Is this transformation exact? In other words, if we compute the posterior probability over some variable $D_i \notin D$, will we get the correct posterior probability (relative to our original model)? Justify your answer.

   Answer: In the general case, this transformation is not exact. As an example, consider a simple network with D_1, D_2 and F_1, F_2 where the parents of F_1 are D_1, D_2 and the parent of F_2 is D_2. Before the transformation, D_1 is not independent of F_2 given F_1. After eliminating D_2 from the network, then F_2 is disconnected from the other variables, giving $(D_1 \perp F_2 \mid F_1)$. In terms of posterior probabilities this means that while for some distributions, $P(D_1 \mid F_1, f_2^0) \neq P(D_1 \mid F_1, f_2^1)$ in the original network, in the transformed network, it is always the case that $P(D_1 \mid F_1, f_2^0) = P(D_1 \mid F_1, f_2^1)$; thus, the results of the posterior probability calculations are not necessarily preserved after variable D_2 is summed out.

   Note, however, that if each of the D_i variables eliminated has exactly one F_i child in the network, then this transformation is exact.

Exercise 5.8 — Soundness of CSI-separation[23 points]

Consider a Bayesian network $\mathcal{B}$ parameterized by a set of tree-CPDs. Recall that in such cases, the network also exhibits context-specific independencies (CSI). In this exercise, we define a simple graph-based procedure for testing for these independencies; your task will be to prove that this procedure is sound.

Consider a particular assignment of evidence $\boldsymbol{Z} = \boldsymbol{z}$. We define an edge $X \rightarrow Y$ to be *spurious* in the context $\boldsymbol{z}$ if, in the tree CPD for Y, all paths down that tree that are consistent with the context $\boldsymbol{z}$ do not involve X. (Example 4.3.6 in the notes provides two examples.) We define $\boldsymbol{X}$ and $\boldsymbol{Y}$ to be *CSI-separated* given $\boldsymbol{z}$ if they are d-separated in the graph where we remove all edges that are spurious in the context $\boldsymbol{z}$.

You will now show that CSI-separation is a sound procedure for detecting independencies. That is: If P is the distribution defined by $\mathcal{B}$, and $\boldsymbol{X}$ and $\boldsymbol{Y}$ are CSI-separated given $\boldsymbol{z}$ in $\mathcal{B}$ (written *CSI-sep*$_{\mathcal{B}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{z})$), then $P \models ((\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{z}))$.

This proof will 4.9. However, the proof of Theorem 4.9 does not provide enough details, so we will provide you with an outline of a proof of the above statement, which you must complete.

1. **[4 points]** Let $\boldsymbol{U} = \boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{Z}$, let $G' = G^+[\boldsymbol{U}]$ be the induced Bayesian network over $\boldsymbol{U} \cup \mathit{Ancestors}(\boldsymbol{U})$, and let $\mathcal{B}'$ be the Bayesian network defined over G' as follows: the CPD for any variable in $\mathcal{B}'$ is the same as in $\mathcal{B}$. You may assume without proof for the rest of this problem that $P_{\mathcal{B}'}(\boldsymbol{U}) = P_{\mathcal{B}}(\boldsymbol{U})$.

   Define a Markov network $\mathcal{H}$ over the variables $\boldsymbol{U} \cup \mathit{Ancestors}(\boldsymbol{U})$ such that $P_{\mathcal{H}} = P_{\mathcal{B}'}$.

   Answer: Let H have structure $\mathcal{M}[\mathcal{B}']$, and have the factors of H be the CPDs of $\mathcal{B}'$. Then $P_{\mathcal{H}} = P_{\mathcal{B}'}$.

2. **[4 points]** Define *spurious edges* and therefore *CSI-separation* for Markov networks. Your definition should be a natural one, but in order to receive credit, parts (3) and (4) must hold.

   Answer: An edge from X to Y is spurious for a Markov net if no factor contains both X and Y. Specifically for the case of tree-factors (as used in Problem 3 of Problem Set 1), an edge from X to Y is spurious for a Markov net if no *reduced* factor contains both X and Y.

   Then we define: $\textit{CSI-sep}_{\mathcal{H}}(\boldsymbol{X};\boldsymbol{Y} \mid \boldsymbol{z})$ if $\textit{sep}_{\mathcal{H}'}(\boldsymbol{X};\boldsymbol{Y} \mid \boldsymbol{z})$, where $\mathcal{H}'$ is $\mathcal{H}$ with all spurious edge removed.

3. **[12 points]** Show that if $\textit{CSI-sep}_{\mathcal{B}}(\boldsymbol{X};\boldsymbol{Y} \mid \boldsymbol{z})$ then $\textit{CSI-sep}_{\mathcal{H}}(\boldsymbol{X};\boldsymbol{Y} \mid \boldsymbol{z})$. (Hint: as one part of this step, use proposition 4.10.)

   Answer: Let $\mathcal{B}_c$ and $\mathcal{H}_c$ be $\mathcal{B}$ and $\mathcal{H}$, respectively, with spurious edges (in the context z) removed, and let $\mathcal{H}_m = \mathcal{M}[\mathcal{B}_c]$. If $\textit{CSI-sep}_{\mathcal{B}}(\boldsymbol{X};\boldsymbol{Y} \mid \boldsymbol{z})$ then $\textit{d-sep}_{\mathcal{B}_c}(\boldsymbol{X};\boldsymbol{Y} \mid \boldsymbol{z})$. Then $\textit{sep}_{\mathcal{H}_m}(\boldsymbol{X};\boldsymbol{Y} \mid \boldsymbol{z})$ due to proposition 4.10.

   Now, $\mathcal{H}_m = \mathcal{H}_c$, because, by construction, spurious edges in $\mathcal{H}$ exactly correspond to edges in $\mathcal{B}$ that are either spurious or are moralizing of spurious edges. Hence, $\textit{sep}_{\mathcal{H}_c}(\boldsymbol{X};\boldsymbol{Y} \mid \boldsymbol{z})$ and therefore, by definition, $\textit{CSI-sep}_{\mathcal{H}}(\boldsymbol{X};\boldsymbol{Y} \mid \boldsymbol{z})$.

4. **[2 points]** Show that if $\textit{CSI-sep}_{\mathcal{H}}(\boldsymbol{X};\boldsymbol{Y} \mid \boldsymbol{z})$ then $P_{\mathcal{H}} \models (\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{z})$. **Answer:** If $\textit{CSI-sep}_{\mathcal{H}}(\boldsymbol{X};\boldsymbol{Y} \mid \boldsymbol{z})$ then $\textit{sep}_{\mathcal{H}_c}(\boldsymbol{X};\boldsymbol{Y} \mid \boldsymbol{z})$. By the soundness of Markov nets, $P_{\mathcal{H}_c} \models ((\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{z}))$. Since $P_{\mathcal{H}_c} = P_{\mathcal{H}}$ (since the reduced factors induce the same probability as the full factors), $P_{\mathcal{H}} \models ((\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{z}))$.

5. **[1 points]** Conclude that if $\boldsymbol{X}$ and $\boldsymbol{Y}$ are CSI-separated given $\boldsymbol{z}$ in $\mathcal{B}$, then $P_{\mathcal{B}} \models (\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{z})$.

   Answer: This follows immediately from (d) and the fact that $P_{\mathcal{B}} = P_{\mathcal{H}}$.

Chapter 9

Exercise 9.10 — Variable Elimination Ordering Heuristics [25 points]

Greedy VE Ordering

- initialize all nodes as unmarked
- for $k = 1 : n$
  - choose the unmarked node that minimizes some greedy function f
  - assign it to be X_k and mark it
  - add edges between all of the unmarked neighbors of X_k
- output $X_1 \dots X_k$

Consider three greedy functions f for the above algorithm:

- $f_A(X_i)$ = number of unmarked neighbors of X_i
- $f_B(X_i)$ = size of the intermediate factor produced by eliminating X_i at this stage
- $f_C(X_i)$ = number of added edges caused by marking X_i

Show that none of the these functions produces an algorithm that dominates the others. That is, give an example (a BN $\mathcal{G}$) where f_A produces a more efficient elimination ordering than f_B. Then give an example where f_B produces a better ordering than f_C. Finally, provide an example where f_C is more efficient than f_A. For each case, define the undirected graph, the factors over it, and the number of values each variable can take. From these three examples, argue that none of the above heuristic functions are optimal.

Answer: We will show an example where f_B is better than both f_A and f_C and an example where f_A is better than f_C.

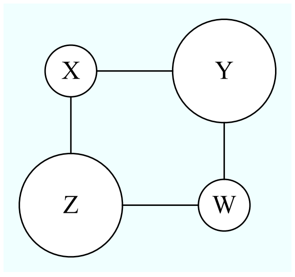


Figure 2: f_B better than f_A and f_C

Consider Figure (2). Suppose pairwise factors $\phi(X, Y)$, $\phi(Y, W)$, $\phi(X, Z)$, and $\phi(Z, W)$. Suppose $|Val(X)| = |Val(W)| = d$, $|Val(Y)| = |Val(Z)| = D$, and $D >> d$.

f_B could choose the ordering Y, Z, X, W (it's ensured to pick one of Z or Y first to avoid creating a large factor over both Z and Y). The cost of variable elimination under this ordering is Dd^2 multiplications and $(D-1)d^2$ additions to eliminate Y, Dd^2 multiplications and $(D-1)d^2$ additions to eliminate Y, d^2 multiplications and $(d-1)d$ additions to eliminate X, and $(d-1)$ additions to eliminate W.

f_A and f_C could each choose the ordering X, Y, Z, W (any possible ordering is equally attractive to these heuristics because they don't consider information about domain sizes). The cost of variable elimination under this ordering is D^2d multiplications and $D^2(d-1)$ additions to eliminate X, D^2d multiplications and $(D-1)Dd$ additions to eliminate Y, Dd multiplications and $(D-1)d$ additions to eliminate Z, and $(d-1)$ additions to eliminate W.

Since $D \gg d$ the $D^2(d-1)$ terms in the cost of variable elimination under an ordering produced by f_C or f_A outweigh the cost of variable elimination under an ordering produced by f_B. So neither f_A nor f_C dominates. The intuition in this example is that f_A and f_C can create unnecessarily large factors because they don't consider the domain sizes of variables.

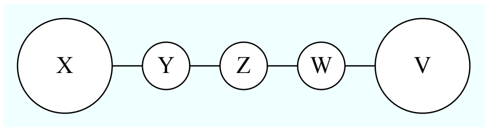


Figure 3: f_A better than f_B

Consider Figure (3). Suppose pairwise factors $\phi(X,Y)$, $\phi(Y,Z)$, $\phi(Z,W)$, and $\phi(W,V)$. Suppose $|Val(Y)| = |Val(Z)| = |Val(W)| = d$, $|Val(X)| = |Val(V)| = D$, $D = 13$, and $d = 2$.

f_A could choose the ordering X, Y, Z, W, V (it's ensured to only pick from the ends of chain). The cost of variable elimination under this ordering is $(D-1)d$ additions to eliminate X, d^2 multiplications and $(d-1)d$ additions to eliminate Y, d^2 multiplications and $(d-1)d$ additions to eliminate Z, Dd multiplications and $(d-1)D$ additions to eliminate W, and $(D-1)$ additions to eliminate V. This is 34 multiplications and 53 additions.

f_B could choose the ordering Z, X, Y, W, V (it's ensured to pick Z first since the size of a factor over X, Z, W is 8, which is less than that for eliminating X or V (26) and less than that for eliminating Y or W (52)). The cost of variable elimination under this ordering is d^3 multiplications and $(d-1)d^2$ additions to eliminate Z, Dd multiplications and $(D-1)d$ additions to eliminate X, d^2 multiplications and $(d-1)d$ additions to eliminate Y, Dd multiplications and $(d-1)D$ additions to eliminate W, and $(D-1)$ additions to eliminate V. This is 64 multiplications and 55 additions.

So f_B doesn't dominate. The intuition in this example is that f_B can avoid dealing with large intermediate factors that it will eventually have to deal with anyways, and in the meantime create a bigger mess for itself.

Chapter 10

Exercise 10.2 (one direction) — clique-tree sepset separation [20 points]

In this problem, you will show that the set of variables in a clique tree separator separates the original Markov network into two conditionally independent pieces.

Let $\boldsymbol{S}$ be a clique separator in the clique tree $\mathcal{T}$. Let $\boldsymbol{X}$ be the set of all variables mentioned in $\mathcal{T}$ on one side of the separator and let $\boldsymbol{Y}$ be the set of all variables mentioned on the other side of the separator. Prove that $sep_I(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{S})$.

Answer: First we prove a short lemma: all nodes that are in both $\boldsymbol{X}$ and $\boldsymbol{Y}$ are in $\boldsymbol{S}$.

Proof of lemma: Suppose $\boldsymbol{S}$ separates C_1 and C_2. Now, consider any node D that is in both $\boldsymbol{X}$ and $\boldsymbol{Y}$. Then there must be at least one clique on each side of the clique separator which each contain D. By the running intersection property, D is contained in C_1 and C_2. Therefore, D is in $\boldsymbol{S}$.

Proof of $sep_I(\boldsymbol{X}, \boldsymbol{Y} \mid \boldsymbol{S})$: We will prove the result by contradiction. Suppose it is not the case that $sep_I(\boldsymbol{X}, \boldsymbol{Y} \mid \boldsymbol{S})$. Then there is some node X in $\boldsymbol{X}$, some node Y in $\boldsymbol{Y}$ and a path π in G between them such that no variable along π is in $\boldsymbol{S}$. Since we never pass through any variable in $\boldsymbol{S}$, the above lemma guarantees that π must pass directly from some variable D which is in $\boldsymbol{X}$ but not in $\boldsymbol{Y}$ to some variable E which is in $\boldsymbol{Y}$ but not in $\boldsymbol{X}$. Since π passes directly from D to E, we know that D and E share an edge in I. So D and E form a clique in I which must be part of some maximal clique in the clique tree. But, by the definition of $\boldsymbol{X}$ and $\boldsymbol{Y}$ we know that all cliques in the tree must be a subset of $\boldsymbol{X}$ or a subset of $\boldsymbol{Y}$. So no clique in the tree can contain D and E, and we have reached a contradiction. Thus $sep_I(\boldsymbol{X}, \boldsymbol{Y} \mid \boldsymbol{S})$.

Exercise 10.6 — modifying a clique tree [12 points] Assume that we have constructed a clique tree $\mathcal{T}$ for a given Bayesian network graph $\mathcal{G}$, and that each of the cliques in $\mathcal{T}$ contains at most k nodes. Now the user decides to add a single edge to the Bayesian network, resulting in a network $\mathcal{G}'$. (The edge can be added between any pair of nodes in the network, as long as it maintains acyclicity.) What is the tightest bound you can provide on the maximum clique size in a clique tree $\mathcal{T}'$ for $\mathcal{G}'$? Justify your response by explaining how to construct such a clique tree. (Note: You do not need to provide the optimal clique tree $\mathcal{T}'$. The question asks for the tightest clique tree that you can construct, using only the fact that $\mathcal{T}$ is a clique tree for $\mathcal{G}$. Hint: Construct an example.)

Answer: The bound on maximum clique size is $k + 1$. Suppose the user decides to add the edge $X \to Y$. We must update our clique tree to satisfy the family preservation and the running intersection property. (Note that any tree satisfying these two properties is a valid clique tree.) Since the original clique satisfies the family preservation property, it must contain a clique $\boldsymbol{C}$ that contains Y and the original parents of Y. Adding X to $\boldsymbol{C}$ restores the family preservation property in the new clique tree. To restore the running intersection property, we simply add X to all the cliques on a path between $\boldsymbol{C}$ and some other clique containing X. Since we added at most one node to each clique, the bound on maximum clique size is $k + 1$.

Exercise 10.15 — Variable Elimination for Pairwise Marginals [10 points] Consider the task of using a calibrated clique tree $\mathcal{T}$ over factors $\mathcal{F}$ to compute all of the pairwise marginals of variables, $P_{\mathcal{F}}(X, Y)$ for all X, Y. Assume that our probabilistic network consists of a chain $X_1 - X_2 - \cdots - X_n$, and that our clique tree has the form $\boldsymbol{C}_1 - \cdots - \boldsymbol{C}_{n-1}$ where $Scope[\boldsymbol{C}_i] = \{X_i, X_{i+1}\}$. Also assume that each variable X_i has $|Val(X_i)| = d$.

1. **[2 points]**

   What is the total cost of doing variable elimination over the tree, as described in the algorithm of Figure 1 of the supplementary handout (see online), for all $\binom{n}{2}$ variable pairs?

Describe the time complexity in term of n and d.

Answer: To compute $P_{\mathcal{F}}(X_i, X_j)$ using CTree-Query in Figure 2, we need to eliminate $j - i - 1 = O(n)$ variables. The largest factor generated has three nodes. Therefore, the complexity for each query is $O(nd^3)$, assuming each of $X_1, \ldots, X_n$ has domain size d. Since we need to run the query $\binom{n}{2} = O(n^2)$ times, the total time complexity is $O(n^3d^3)$.

2. **[8 points]**

   Since we are computing marginals for all variable pairs, we may store any computations done for the previous pairs and use them to save time for the remaining pairs. Construct such a dynamic programming algorithm that achieves a running time which is asymptotically significantly smaller. Describe the time complexity of your algorithm.

   Answer: Due to the conditional independence properties implied by the network, we have that, for $i < j - 1$:

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

   The term $P(X_j \mid X_{j-1})$ can be computed directly from the marginals in the calibrated clique tree, while $P(X_i, X_{j-1})$ is computed and stored from a previous step if we arrange the computation in a proper order. Following is the algorithm:

```
------------------------
// $\pi_j = P(X_j, X_{j+1})$: calibrate potential in clique $C_j$.
// $\mu_{j-1,j} = P(X_j)$: message between clique $C_{j-1}$ and $C_j$.
$\mu_{0,1} = \sum_{X_2} \pi_1(X_1, X_2)$
for j = 1 to n - 1 do
        $\psi(X_j) = \frac{\pi_j}{\mu_{j-1,j}}$
        $\phi(X_j, X_{j+1}) = \pi_j$
for i = 1 to n - 2 do
     for j = i + 2 to n do
          $\phi(X_i, X_j) = \sum_{X_{j-1}} \phi(X_i, X_{j-1}) \times \psi(X_{j-1})$
------------------------
```

   where $\psi(X_j) = P(X_{j+1}|X_j)$ and $\phi(X_i, X_j) = P(X_i, X_j)$.

   The algorithm run through the double loop i, j for $O(n^2)$ times. Each time, it performs a Product-Sum of two factors. The immediate factor has three variables and thus it costs $O(d^3)$ time. Therefore, the total time complexity is $O(n^2d^3)$ which is asymptotically significantly smaller.

New problem — Maximum expected grade [20 points] You are taking the final exam for a course on computational complexity theory. Being somewhat too theoretical, your professor has insidiously snuck in some unsolvable problems, and has told you that exactly K of the N

problems have a solution. Out of generosity, the professor has also given you a probability distribution over the solvability of the N problems.

To formalize the scenario, let $\mathcal{X} = \{X_1, \ldots, X_N\}$ be binary-valued random variables corresponding to the N questions in the exam where $Val(X_i) = \{0(\text{unsolvable}), 1(\text{solvable})\}$. Furthermore, let $\mathcal{B}$ be a Bayesian network parameterizing a probability distribution over $\mathcal{X}$ (i.e., problem i may be easily used to solve problem j so that the probabilities that i and j are solvable are not independent in general).

(Note: Unlike the some of the problems given by professor described above, every part of this problem is solvable!)

1. [**8 points**] We begin by describing a method for computing the probability of a question being solvable. That is we want to compute $P(X_i = 1, Possible(\mathcal{X}) = K)$ where

$$Possible(\mathcal{X}) = \sum_i \mathbf{1}\{X_i = 1\}$$

is the number of solvable problems assigned by the professor.

To this end, we define an *extended factor* ϕ as a "regular" factor ψ and an index so that it defines a function $\phi(\boldsymbol{X}, L) : Val(\boldsymbol{X}) \times \{0, \ldots, N\} \mapsto I\!R$ where $\boldsymbol{X} = Scope[\phi]$. A projection of such a factor $[\phi]_l$ is a regular factor $\psi : Val(\boldsymbol{X}) \mapsto I\!R$, such that $\psi(\boldsymbol{X}) = \phi(\boldsymbol{X}, l)$.

Provide a definition of extended factor multiplication and extended factor marginalization such that

$$P(X_i, Possible(\mathcal{X}) = K) = \left[\sum_{\mathcal{X} - \{X_i\}} \prod_{\phi \in \mathcal{F}} \phi \right]_K \tag{1}$$

where each $\phi \in \mathcal{F}$ is an extended factor corresponding to some CPD of the Bayesian network, defined as follows:

$$\phi_{X_i}(\{X_i\} \cup \mathbf{Pa}_{X_i}, k) = \begin{cases} P(X_i \mid \mathbf{Pa}_{X_i}) & \text{if } X_i = k \\ 0 & \text{otherwise} \end{cases}$$

(Hint: Note the similarity of Eq. (1) to the standard clique tree identities. If you have done this correctly, the algorithm for clique tree calibration (algorithm 10.2) can be used *as is* to compute $P(X_i = 1, Possible(\mathcal{X}) = K)$.)

Answer: Intuitively, what we need to do is to associate the probability mass for each entry in a factor with *the number of problems which are solvable among the variables "seen so far"*. More precisely, we say that a variable X_i has been "seen" in a particular intermediate factor ϕ which arises during variable elimination if its corresponding CPD $P(X_i|\mathbf{Pa}_{X_i})$ was used in creating ϕ. Below, we provide definitions for extended factor product and marginalization analogous to the definitions in section 7.2.1 of the textbook.

Extended factor product

Let $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$ be three disjoint sets of variables, and let $\phi_1(\boldsymbol{X}, \boldsymbol{Y}, k_1)$ and $\phi_2(\boldsymbol{Y}, \boldsymbol{Z}, k_2)$ be two extended factors. We define the *extended factor product* $\phi_1 \times \phi_2$ to be an extended factor $\psi : Val(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}) \times \{0, \ldots, N\} \mapsto I\!R$ such that:

$$\psi(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}, k) = \sum_{k_1 + k_2 = k} \phi_1(\boldsymbol{X}, \boldsymbol{Y}, k_1) \cdot \phi_2(\boldsymbol{Y}, \boldsymbol{Z}, k_2).$$

The intuition behind this definition is that when computing some intermediate factor $\psi(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}, k)$ during variable elimination, we want $\psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, k)$ to correspond with the probability mass for which k of the variables seen so far had a value 1. This can occur if k_1 variables are "seen" in ϕ_1 and $k_2 = k - k_1$ variables are "seen" in ϕ_2. Notice that in this definition, the factor corresponding to any CPD $P(X_i|\mathbf{Pa}_{X_i})$ is never involved in the creation of both ϕ_1 and ϕ_2, so there is no "double-counting" of seen variables. Notice also that there are in general *multiple* ways that seen variables might be partitioned between ϕ_1 and ϕ_2, so the summation is needed.

Extended factor marginalization

> Let $\boldsymbol{X}$ be a set of variables, and $Y \notin \boldsymbol{X}$ a variable. Let $\phi(\boldsymbol{X}, Y, k)$ be an extended factor. We define the *extended factor marginalization* of Y in ϕ to be an extended factor $\psi : Val(\boldsymbol{X}) \times \{0, \ldots, N\} \mapsto I\!R$ such that:
>
> $$\psi(\boldsymbol{X}, k) = \sum_{Y} \phi(\boldsymbol{X}, Y, k).$$

The definition for extended factor marginalization is almost exactly the same as in the case of regular factor marginalization. Here, it is important to note that summing out a variable should not change the value of k associated with some particular probability mass—k acts as a label of how many variables are set to 1 among those "seen" thus far. In the final potential, k gives the total number of variables set to 1 in $\mathcal{X}$.

To show that these definitions actually work (though we didn't require it), one can show by induction (using the definition of initial factors in the problem set) that

$$\psi(X_1, \ldots, X_i, \mathbf{Pa}_{X_1}, \ldots, \mathbf{Pa}_{X_i}, k) \quad = \quad \mathbf{1}\{X_1 + \ldots + X_i = k\} \left[\prod_{j=1}^{i} P(X_j | \mathbf{Pa}_{X_j}) \right]$$

and the result follows since $X_1 + \ldots + X_N = K$ is equivalent to $Possible(\mathcal{X}) = K$. Showing that variable elimination works requires that we show a result for the commutativity of the sum and product operators analogous to that for regular factors.

2. [**6 points**] Realistically, you will have time to work on exactly M problems ($1 \leq M \leq N$). Obviously, your goal is to maximize the expected number of solvable problems that you finish (you neither gain nor lose credit for working on an unsolvable problem). Let $\boldsymbol{Y}$ be a subset of $\mathcal{X}$ indicating exactly M problems you choose to work on and let

$$Correct(\mathcal{X}, \boldsymbol{Y}) \quad = \quad \sum_{X_i \in \boldsymbol{Y}} \mathbf{1}\{X_i = 1\}$$

be the number of solvable problems that you attempt (luckily for you, every solvable problem that you attempt you will solve correctly!). Thus, your goal is to find

$$\text{argmax}_{\boldsymbol{Y}:|\boldsymbol{Y}|=M} E[Correct(\mathcal{X}, \boldsymbol{Y}) \mid Possible(\mathcal{X}) = K].$$

Show that

$$\text{argmax}_{\boldsymbol{Y}:|\boldsymbol{Y}|=M} E[Correct(\mathcal{X}, \boldsymbol{Y})] \neq \text{argmax}_{\boldsymbol{Y}:|\boldsymbol{Y}|=M} E[Correct(\mathcal{X}, \boldsymbol{Y}) \mid Possible(\mathcal{X}) = K]$$

by constructing a simple example in which equality fails to hold.

Answer: Consider a case in which $N = 3$, and $M = 1$, and suppose the Bayesian network $\mathcal{B}$ encodes the following distribution:

- with probability p, X_1 is solvable while X_2 and X_3 are not, and
- with probability $1-p$, X_2 and X_3 are solvable while X_1 is not.

If exactly $K = 2$ problems are solvable, we know automatically that X_2 and X_3 must be solvable and that X_1 is not:

$\boldsymbol{Y}$	$E[Correct(\mathcal{X}, \boldsymbol{Y})]$	$E[Correct(\mathcal{X}, \boldsymbol{Y}) \mid Possible(\mathcal{X}) = K]$
$\{X_1\}$	$1p + 0(1-p) = p$	0
$\{X_2 \text{ or } X_3\}$	$0p + 1(1-p) = 1-p$	1

Thus, the problems chosen by the two methods differ when $p > 1-p$ (i.e., $p > 1/2$).

3. **[6 points]** Using the posterior probabilities calculated in (a), give an efficient algorithm for computing $E[Correct(\mathcal{X}, \boldsymbol{Y}) \mid Possible(\mathcal{X}) = K]$. Based on this, give an efficient algorithm for finding $\text{argmax}_{\boldsymbol{Y}:|\boldsymbol{Y}|=M} E[Correct(\mathcal{X}, \boldsymbol{Y}) \mid Possible(\mathcal{X}) = K]$. (Hint: Use linearity of expectations.)

Answer: Note that

$$\begin{aligned}
E[Correct(\mathcal{X}, \boldsymbol{Y}) \mid Possible(\mathcal{X}) = K] &= E\left[\sum_{X_i \in \mathbf{Y}} \mathbf{1}\{X_i = 1\} \mid Possible(\mathcal{X}) = K\right] \\
&= \sum_{X_i \in \mathbf{Y}} E[\mathbf{1}\{X_i = 1\} \mid Possible(\mathcal{X}) = K] \\
&= \sum_{X_i \in \mathbf{Y}} P(X_i = 1 \mid Possible(\mathcal{X}) = K) \\
&= \frac{1}{Z} \sum_{X_i \in \mathbf{Y}} P(X_i = 1, Possible(\mathcal{X}) = K).
\end{aligned}$$

where $Z = P(Possible(\mathcal{X}) = K)$ is a normalizing constant. To maximize this expression over all sets $\boldsymbol{Y}$ such that $|\boldsymbol{Y}| = M$, note that the contribution of each selected element $X_i \in \boldsymbol{Y}$ is independent of the contributions of all other selected elements. Thus, it suffices to find the M largest values in the set $\{P(X_i = 1, Possible(\mathcal{X}) = K) : 1 \leq i \leq N\}$ and select their corresponding problems.

Chapter 11

Exercise 11.17 — Markov Networks for Image Segmentation [18 points]

In class we mentioned that Markov Networks are commonly used for many image processing tasks. In this problem, we will investigate how to use such a network for image segmentation. The goal of image segmentation is to divide the image into large contiguous regions (segments), such that each segment is internally consistent in some sense.

We begin by considering the case of a small 3×2 image. We define a variable X_i for each node (pixel) in the network, and each can take on the values $1...K$ for K image segments. The resulting Markov Network $\mathcal{M}$ is shown in Figure 4.

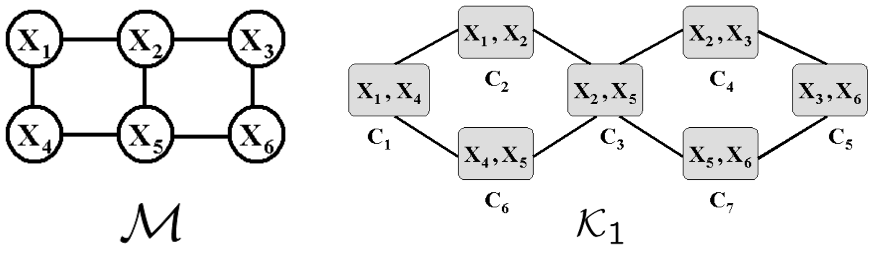


Figure 4: Markov Network for image segmentation.

We will perform our segmentation by assigning factors to this network over pairs of variables. Specifically, we will introduce factors $\phi_{i,j}$ over neighboring pixels i and j that quantify how strong the affinity between the two pixels are (i.e., how strongly the two pixels in question want to be assigned to the same segment).

We now want to perform inference in this network to determine the segmentation. Because we eventually want to scale to larger networks, we have chosen to use loopy belief propagation on a cluster graph. Figure 4 also shows the cluster graph $\mathcal{K}_1$ that we will use for inference.

1. **[1 points]** Write the form of the message $\delta_{3\to 6}$ that cluster C_3 will send to cluster C_6 during the belief update, in terms of the ϕ's and the other messages.

   Answer: The form of the message is:

$$\delta_{3\to 6}(X_5) = \sum_{X_2} \phi_{2,5}(X_2, X_5)\delta_{2\to 3}(X_2)\delta_{4\to 3}(X_2)\delta_{7\to 3}(X_5).$$

2. **[6 points]** Now, consider the form of the initial factors. We are ambivalent to the actual segment labels, but we want to choose factors that make two pixels with high "affinity" (similarity in color, texture, intensity, etc.) more likely to be assigned together. In order to do this, we will choose factors of the form (assume $K = 2$):

$$\phi_{i,j}(X_i, X_j) = \begin{bmatrix} \alpha_{i,j} & 1 \\ 1 & \alpha_{i,j} \end{bmatrix} \qquad (2)$$

   Where $\alpha_{i,j}$ is the affinity between pixels i and j. A large $\alpha_{i,j}$ makes it more likely that $X_i = X_j$ (i.e., pixels i and j are assigned to the same segment), and a small value makes it less likely. With this set of factors, compute the initial message $\delta_{3\to 6}$, assuming that this is the first message sent during loopy belief propagation. Note that you can renormalize the messages at any point, because everything will be rescaled at the end anyway. What will be the final marginal probability, $P(X_4, X_5)$, in cluster C_6? What's wrong with this approach?

Answer: The form of the message is:

$$\delta_{3\to 6}(X_5) = \begin{bmatrix} 1+\alpha_{2,5} \\ 1+\alpha_{2,5} \end{bmatrix} \propto \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Here we see that the initial message (and indeed all subsequent messages) will be equal to the $\mathbf{1}$ message. This means that the cluster beliefs will never be updated. Therefore:

$$P(X_4, X_5) \propto \pi_6(X_4, X_5) = \phi_{4,5}$$

$$P(X_4, X_5) = \begin{cases} \frac{1}{2+2\alpha_{4,5}} & \text{if } X_4 \neq X_5, \\ \frac{\alpha_{4,5}}{2+2\alpha_{4,5}} & \text{if } X_4 = X_5, \end{cases}$$

In this approach, because the initial potentials are symmetric, and the messages are over single variables, no information is ever propagated around the cluster graph. This results in the final beliefs being the same as the initial beliefs. This construction is, as a result, completely useless for this task.

Chapter 12

Exercise 12.13 — Stationary distribution for Gibbs chain [18 points]

Show directly from Eq. (12.21) (without using the detailed balance equation) that the posterior distribution $P(\mathcal{X} \mid \boldsymbol{e})$ is a stationary distribution of the Gibbs chain (Eq. (12.22)). In other words, show that, if a sample $\boldsymbol{x}$ is generated from this posterior distribution, and $\boldsymbol{x}'$ is the next sample generated by the Gibbs chain starting from $\boldsymbol{x}$, then $\boldsymbol{x}'$ is also generated from the posterior distribution.

Answer: The time t distribution is:

$$P^{(t)}(\boldsymbol{x}') \approx P^{(t+1)}(\boldsymbol{x}') = \sum_{\boldsymbol{x} \in Val(\boldsymbol{X})} P^{(t)}(\boldsymbol{x})\mathcal{T}(\boldsymbol{x} \to \boldsymbol{x}').$$

Equation (12.21) refers to the global transition model, which in the case of the Gibbs Chain corresponds to the probability of choosing a variable to flip and then applying the local transition model. All variables are equally likely to be flipped. The varable x_i is sampled from the distribution $P(x_i \mid \boldsymbol{u}_i)$ where $\boldsymbol{u}_i$ is the assignment to all variables other than x_i.

Notice in the case of the Gibbs chain, only one value is changed at a time.

$$\frac{1}{|\mathcal{X}|} \sum_{k \in |\mathcal{X}|} \sum_{\boldsymbol{x}_k \in Val(\boldsymbol{X}_k)} P(x_k, \boldsymbol{u}_k \mid \boldsymbol{e}) P(x'_k \mid \boldsymbol{u}_k, \boldsymbol{e})$$

$$\frac{1}{|\mathcal{X}|} \sum_{k \in |\mathcal{X}|} \sum_{\boldsymbol{x}_k \in Val(\boldsymbol{X}_k)} P(x_k \mid \boldsymbol{u}_k) P(x'_k, \boldsymbol{u}_k \mid \boldsymbol{e})$$

$$P(\boldsymbol{X}' \mid \boldsymbol{e}) \frac{1}{|\mathcal{X}|} \sum_{k \in |\mathcal{X}|} \sum_{\boldsymbol{x}_k \in Val(\boldsymbol{X}_k)} P(x_k \mid \boldsymbol{u}_k)$$

$$P(\boldsymbol{X}' \mid \boldsymbol{e})$$

New exercise— Data Association and Collapsed MCMC [20 points]

Consider a data association problem for an airplane tracking application. We have a set of K sensor measurements blips on a radar screen $\mathcal{U} = \{u_1, \ldots, u_K\}$ and another set of M airplanes $\mathcal{V} = \{v_1, \ldots, v_M\}$, and we wish to map $\mathcal{U}$'s to $\mathcal{V}$'s. We introduce a set of correspondence variables $\mathcal{C} = \{C_1, \ldots, C_K\}$ such that $Val(C_i) = \{1, \ldots, M\}$. Here, $C_i = j$ indicates that u_i is matched to v_j (The fact that each variable C_i takes on only a single value implies that each measurement is derived from only a single object. But the mutual exclusion constraints in the other direction are not forced.). In addition, for each u_i, we have a set of readings denoted as a vector $\boldsymbol{B}_i = (B_{i1}, B_{i2}, \ldots, B_{iL})$; for each v_j, we have a three dimensional random vector $\boldsymbol{A}_j = (A_{j1}, A_{j2}, A_{j3})$ corresponding to the location of the airplane v_j in the space. (Assume that $|Val(A_{jk})| = d$, which is not too large in the sense that summing over all values of $\boldsymbol{A}_j$ is tractable.)

Now suppose that we have a prior distribution over the location of v_j, i.e., we have $P(\boldsymbol{A}_j)$, and we have observed all $\boldsymbol{B}_i$, and a set of $\phi_{ij}(\boldsymbol{A}_j, \boldsymbol{B}_i, C_i)$ such that $\phi_{ij}(\boldsymbol{a}_j, \boldsymbol{b}_i, C_i) = 1$ for all $\boldsymbol{a}_j$, $\boldsymbol{b}_i$ if $C_i \neq j$. The model contains no other potentials.

We wish to compute the posterior over $\boldsymbol{A}_j$ using collapsed Gibbs sampling, where we sample the C_i's but maintain a closed form posterior over the $\boldsymbol{A}_j$'s.

1. [**1 points**] Briefly explain why sampling the C_i's would be a good idea.

   Answer: To compute the posterior over all joint $\boldsymbol{A}_j$'s or a single $\boldsymbol{A}_j$ using exact inference is hard, since it requires exponentially large computation. We use sampling based method to approximate the posterior by sampling C_i's resulting in the conditional independence between $\boldsymbol{A}_j$'s given $\boldsymbol{B}_i$'s and C_i's. This factorization gives us a closed form over each $\boldsymbol{A}_j$, which makes the computation tractable. Moveover, since C_i's serve as selector variables, the context after sampling them further reduces many factors to be uniform (context independence: $\boldsymbol{A}_j$ doesn't change the belief over $\boldsymbol{B}_i$ if $\boldsymbol{C}_i \neq j$). In addition, the space over the joint $\boldsymbol{A}_j$'s (d^{3M}) might be larger than that of C_i's (M^K), since d is usually much larger than M and K, and M is likely to be larger than K. Sampling in a lower dimensional space is better in terms of smaller number of samples and better approximation.

2. [**6 points**] Show clearly the sampling distribution for the C_i variables and the corresponding Markov chain kernels.

   Answer: Let Φ denote the set of all factors (note that $P(\boldsymbol{A}_j)$'s are also the factors in our model), then the distribution over C_i's which we want to sample from is

$$\begin{aligned} P(C_1, C_2, \ldots, C_K | \boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_K) &= P_{\Phi[\bar{\boldsymbol{b}}]}(C_1, C_2, \ldots, C_K) && (3) \\ &= \frac{1}{Z} \sum_{\boldsymbol{A}_j' s} \prod_j P(\boldsymbol{A}_j) \prod_i \phi_{ij}(\boldsymbol{A}_j, \boldsymbol{b}_i, C_i) && (4) \end{aligned}$$

   where Z is the partition function. In order to get samples from the posterior above, we can sample from a Markov chain whose stationary distribution is our target. To do so, we define the Markov chain kernel for each C_k as:

$$P_{\Phi[\bar{\boldsymbol{b}}]}(C_k | \boldsymbol{c}_{-k}) = \frac{P_{\Phi[\bar{\boldsymbol{b}}]}(C_k, \boldsymbol{c}_{-k})}{\sum_{C_k'} P_{\Phi[\bar{\boldsymbol{b}}]}(C_k', \boldsymbol{c}_{-k})}. \quad (5)$$

   For each value of $C_k = c_k, c_k = 1, 2, \ldots, M$, we have

$$\begin{aligned} P_{\Phi[\bar{\boldsymbol{b}}]}(C_k = c_k | \boldsymbol{c}_{-k}) &= \frac{P_{\Phi[\bar{\boldsymbol{b}}]}(c_k, \boldsymbol{c}_{-k})}{P_{\Phi[\bar{\boldsymbol{b}}]}(\boldsymbol{c}_{-k})} && (6) \\ &= \frac{1}{Z} \sum_{\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_M} P_{\Phi[\bar{\boldsymbol{b}}]}(c_k, \boldsymbol{c}_{-k}, \boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_M) && (7) \\ &= \frac{1}{Z} \sum_{\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_M} \prod_{j=1}^{M} P(\boldsymbol{A}_j) \prod_{i:c_i=j} \phi_{ij}(\boldsymbol{A}_j, \boldsymbol{b}_i, j) && (8) \\ &= \frac{1}{Z} \prod_{j=1}^{M} \sum_{\boldsymbol{A}_j} \left(P(\boldsymbol{A}_j) \prod_{i:c_i=j} \phi_{ij}(\boldsymbol{A}_j, \boldsymbol{b}_i, j) \right) && (9) \\ &= \frac{1}{Z} \prod_{j=1}^{M} \sum_{\boldsymbol{A}_j} \Psi_{j,[c_1, c_2, \ldots, c_K]}(\boldsymbol{A}_j) && (10) \end{aligned}$$

   where

$$\Psi_{j,[c_1, c_2, \ldots, c_K]}(\boldsymbol{A}_j) = P(\boldsymbol{A}_j) \prod_{i:c_i=j} \phi_{ij}(\boldsymbol{A}_j, \boldsymbol{b}_i, j) \quad (11)$$

and

$$Z = \sum_{c_k} \prod_{j=1}^{M} \sum_{\boldsymbol{A}_j} \Psi_{j,[c_k,\boldsymbol{c}_{-k}]}(\boldsymbol{A}_j) \tag{12}$$

Note that to the summation over A_j's can be pushed into the product so that when we sum over a specific A_j, the summation is only over a factor $\Psi_{j,[c_1,c_2,\ldots,c_K]}(\boldsymbol{A}_j)$ whose scope only contains $\boldsymbol{A}_j$. And as mentioned in the question, we assume that summing over a single $\boldsymbol{A}_j$ is tractable.

3. [**8 points**] Give a closed form equation for the distribution over the $\boldsymbol{A}_j$ variables given the assignment to the C_i's. (A closed form equation must satisfy two criteria: it must contain only terms whose values we have direct access to, and it must be tractable to compute.)

   Answer: Since A_j's are independent given all C_i's, the joint distribution over A_j's can be factorized as the product of the marginal distributions over each single A_j. Therefore, we only specify the marginal distribution over a single A_j:

$$P(\boldsymbol{A}_j|\bar{\boldsymbol{b}},\bar{c}) = \frac{1}{Z} P(\boldsymbol{A}_j) \prod_{i:c_i=j} \phi_{ij}(\boldsymbol{A}_j,\boldsymbol{b}_i,c_i). \tag{13}$$

   where, $Z = \sum_{\boldsymbol{A}_j} P(\boldsymbol{A}_j)\prod_{i:c_i=j}\phi(\boldsymbol{A}_j,\boldsymbol{b}_i,c_i)$, $\bar{\boldsymbol{b}} = \{\boldsymbol{b}_1,\boldsymbol{b}_2,\ldots,\boldsymbol{b}_K\}$ and $\bar{c} = \{c_1,\ldots,c_K\}$.

4. [**5 points**] Show the equation for computing the posterior over $\boldsymbol{A}_j$ based on the two steps above.

   Answer: After the Markov chain converges to its stationary distribution, i.e., the posterior over C_i's, we can collect M instances each of which is denoted by $\bar{c}[m], m = 1,2,\ldots,M$, and with their distributional parts $P(\boldsymbol{A}_j|\bar{b},\bar{c}[m])$, we can estimate the posterior over $\boldsymbol{A}_j$ as:

$$\begin{aligned}
\hat{P}(\boldsymbol{A}_j = \boldsymbol{a}_j|\bar{\boldsymbol{b}}) &= \frac{1}{M}\sum_{m=1}^{M} \frac{1}{Z[m]} P(\boldsymbol{a}_j) \prod_{i:c_i[m]=j} \phi_{ij}(\boldsymbol{a}_j,\boldsymbol{b}_i,c_i[m]) && (14)\\
&= \frac{1}{M}\sum_{m=1}^{M} \frac{P(\boldsymbol{a}_j)\prod_{i:c_i[m]=j}\phi_{ij}(\boldsymbol{a}_j,\boldsymbol{b}_i,c_i[m])}{\sum_{\boldsymbol{a}'_j} P(\boldsymbol{a}'_j)\prod_{i:c_i[m]=j}\phi_{ij}(\boldsymbol{a}'_j,\boldsymbol{b}_i,c_i[m])} && (15)
\end{aligned}$$

   where $\boldsymbol{a}_j$ is a specific value that $\boldsymbol{A}_j$ takes.

Chapter 15

Exercise 15.3 — Entanglement in DBNs I [6 points]

Prove Proposition 15.1:

Let $\mathcal{I}$ be the influence graph for a 2-TBN $\mathcal{B}_{\rightarrow}$. Then $\mathcal{I}$ contains a directed path from X to Y if and only if, in the unrolled DBN, for every t, there exists a directed path from $X^{(t)}$ to $Y^{(t')}$ for some $t' \geq t$.

Answer: First, we will prove the only if direction. Let the path between X and Y in $\mathcal{I}$ be the set of nodes $(X, X_1, \cdots, Y)$. By definition of $\mathcal{I}$, for every two consecutive nodes X_i — X_{i+1} in this set, for every t, there exists a directed edge from $X_i^{(t)}$ to $X_{i+1}^{(t+1)}$ or $X_{i+1}^{(t)}$. Now, starting at X_t, construct the directed path in the unrolled by joining these directed edges. This construction holds for any t. Hence, we shown the existence of the path in the unrolled DBN.

To prove the if direction, let us look at a path in the unrolled DBN, starting at any $X^{(t)}$ represented by $X, X_1, \cdots, Y$. Consider the directed edge from X_i to X_{i+1} on this path for any i; there must be a directed edge in the influence graph $\mathcal{I}$ from the node corresponding to X_i to the node corresponding to X_{i+1}. This is true otherwise it violates the construction of the influence graph. Now, connect all the directed edges in the influrence graph corresponding to the path in the unrolled DBN. Hence, we have shown that such a path always exists. concludes our proof.

Exercise 15.4 — Entanglement in DBNs II [14 points]

1. **[10 points]** Prove the entanglement theorem, Theorem 15.1:

   Let $\langle \mathcal{G}_0, \mathcal{G}_{\rightarrow} \rangle$ be a fully persistent DBN structure over $\mathcal{X} = \boldsymbol{X} \cup \boldsymbol{O}$, where the state variables $\boldsymbol{X}^{(t)}$ are hidden in every time slice, and the observation variables $\boldsymbol{O}^{(t)}$ are observed in every time slice. Furthermore, assume that, in the influence graph for $\mathcal{G}_{\rightarrow}$:

   - there is a trail (not necessarily a directed path) between every pair of nodes, i.e., the graph is connected;
   - every state variable X has some directed path to some evidence variable in $\boldsymbol{O}$.

   Then there is no persistent independence $(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$ which holds for every DBN $\langle \mathcal{B}_0, \mathcal{B}_{\rightarrow} \rangle$ over this DBN structure.

   Answer: Note that we will need the additional assumption that $\boldsymbol{Z} \cap (\boldsymbol{X} \cup \boldsymbol{Y}) \neq \emptyset$. We first show that $I(\boldsymbol{X}, \boldsymbol{Y} \mid \emptyset)$ cannot hold persistently.

   Assume by contradiction that $I(\boldsymbol{X}, \boldsymbol{Y} \mid \emptyset)$ does indeed hold persistently. By decomposition this implies that $I(X, Y \mid \emptyset)$ must hold persistently for all $X \in \boldsymbol{X}$ and $Y \in \boldsymbol{Y}$. We now pick a particular t, X, and Y and examine the statement we have assumed to hold: $I(X^{(t)}, Y^{(t)} \mid \emptyset)$.

   We first prove the existence of a trail between $X^{(t)}$ and $Y^{(t)}$ in our DBN. As our influence graph is connected it must contain a trail from X to Y. As the influence graph was defined over the 2-TBN this implies a trail in the 2-TBN (using persistence edges to alternate between interface variables and non-interface variables as necessary). This trail implies the existence of a corresponding trail between $X^{(t)}$ and $Y^{(t)}$ in the t and $t+1$ timeslices of the DBN. Call this trail π. For $I(X^{(t)}, Y^{(t)} \mid \emptyset)$ to hold we must have that π is not active. This can occur in two ways:

   (a) π goes through an observed node that is not the center of a v-structure. This is impossible as our only observed nodes are leaves in the DBN.

(b) π goes through a v-structure $A \rightarrow B \leftarrow C$ where neither B nor any of its descendents are observed. This cannot be the case because either $B \in \boldsymbol{O}$ or B is a state variable. However, if B is a state variable we know that a descendant of B is in $\boldsymbol{O}$ by Proposition 19.3.4 and our assumption that every state variable has a directed path to an observation variable in the influence graph.

Thus the trail must be active, giving us a contradiction.

We've thus shown that $I(\boldsymbol{X}, \boldsymbol{Y} \mid \emptyset)$ does not hold persistently and will prove that $I(\boldsymbol{X}, \boldsymbol{Y} \mid \boldsymbol{Z})$. To do this we note that by decomposition we must have $I(X, Y \mid \boldsymbol{Z})$ (and also by our assumption $X, Y \notin \boldsymbol{Z}$). Part (c) will prove that this cannot be the case when we have that $I(X, Y \mid \emptyset)$ does not hold persistently (as we have shown above.)

We now prove that $I(\boldsymbol{X}, \boldsymbol{Y} \mid \boldsymbol{Z})$ does not hold persistently with the additional assumption that $X, Y \notin \boldsymbol{Z}$.

Assume by contradiction that $I(X, Y \mid \boldsymbol{Z})$ holds persistently but that $I(X, Y \mid \emptyset)$ does not. By assumption there must exist some time t such that $I(X^{(t)}, Y^{(t)} \mid \emptyset)$ does not hold. This implies the existence of an active trail π between $X^{(t)}$ and $Y^{(t)}$ in the unrolled DBN. We then consider some time slice t' which has no nodes in common with π. By assumption we must have $I(X^{(t')}, Y^{(t')} \mid \boldsymbol{Z}^{(t')})$. But now we can construct an active trail π' between $X^{(t')}$ and $Y^{(t')}$ given $\boldsymbol{Z}^{(t')}$. We do so by first going from $X^{(t')}$ to $X^{(t)}$ in π via the persistance edges, travelling through π, and then connecting $Y^{(t)}$ and $Y^{(t')}$ via the persistence edges between them. Note that $\boldsymbol{Z}^{(t')}$ cannot interfere by the assumption that timeslice t' contained no nodes in π and that $X, Y \notin \boldsymbol{Z}$. This active trail, however, implies that $I(X, Y \mid \boldsymbol{Z})$ does not hold at time t' and thus is not a persistent independence. Therefore, we have a contradiction.

2. [**4 points**] Is there any 2-TBN (not necessarily fully persistent) whose unrolled DBN is a single connected component, for which $(X \perp Y \mid \boldsymbol{Z})$ holds persistently but $(X \perp Y \mid \emptyset)$ does not? If so, give an example. If not, explain formally why not.

   Answer: There are many counterexamples for the case that the 2-TBN is not fully persistent. One such example has persistent edge $X_2 \rightarrow X_2'$ and other edges $X_1' \leftarrow X_2' \rightarrow X_3'$. Here $X_1^{(t)} \perp X_3^{(t)} \mid X_2^{(t)}$ holds persistently, but $X_1^{(t)} \perp X_3^{(t)} \mid \emptyset$ does not.

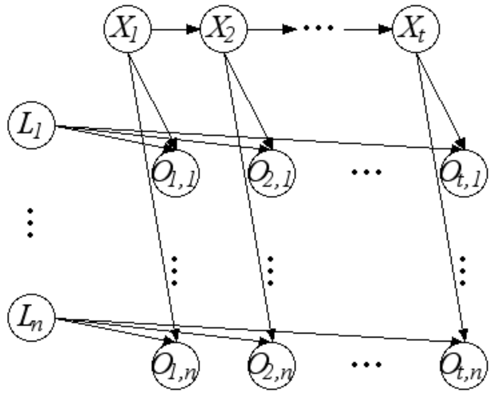
Exercise 15.13 — Collapsed MCMC for Particle Filtering [22 points]

Consider a robot moving around in a space with n stationary landmarks. The positions of the robot and the landmarks are unknown. The position of the robot (also called a *pose*) at time t is denoted $X^{(t)}$. The (fixed) position of landmark i is denoted L_i. At each time t, the robot obtains a (noisy) measurement $O_i^{(t)}$ of its current position relative to each landmark k. The model is parameterized using two components. Robot poses evolve via a *motion model*: $P(X^{(t+1)} \mid X^{(t)})$. Sensor measurements are governed by a *measurement model*: $P(O_i^{(t)} \mid X^{(t)}, L_i)$. Both the motion model and the measurement model are known.

Denote $X^{(1:t)} = \{X^{(1)}, \ldots, X^{(t)}\}$, $\boldsymbol{L} = \{L_1, \ldots, L_n\}$, and $\boldsymbol{O}^{(1:t)} = \{O_1^{(1)}, \ldots, O_n^{(1)}, O_1^{(2)}, \ldots, O_n^{(t)}\}$. We want to use the measurements $\boldsymbol{O}^{(1:t)}$ to localize both the robot and landmarks; i.e., we want to compute $P(X^{(t)}, \boldsymbol{L} \mid \boldsymbol{o}^{(1:t)})$.

1. [**2 points**] Draw a DBN model for this problem. What is the dimension (number of variables) of the belief state for one time slice in this problem? (Assume that the belief state at time t is over all variables about which we are uncertain at time t.)

Answer: The simplest way to represent this is to extend our model for a DBN to allow some static variables, which are not part of each timeslice. We can think of them as existing before the first timeslice. Unrolled, it looks like this:



But to store it compactly, whereas we previously needed only representations of $\mathcal{B}_0$ and $\mathcal{B}_\rightarrow$, representing the initial state and transition model, here we have a third network, representing the static variables, which we will denote $\mathcal{B}_s$. Note that in principle, with this extension to the notion of a DBN, the static variables could actually be a standard Bayes net, with some of them depending on each other; but in our case they will not, so their CPDs have no parents to condition on. However, the CPDs in $\mathcal{B}_\rightarrow$ can then use the variables in $\mathcal{B}_s$ as parents, in addition to any inter-timeslice and intra-timeslice edges.

Of course, we could alternately represent it by including all the static variables in the 2-TBN, each with a deterministic CPD simply propagating its values forward, never changing.

A belief state now corresponds to a set of beliefs for the static variables as well as the state variables of the current timeslice, so in our DBN, the belief state will have a dimension of $n + 1$.

We want to use particle filtering to solve our tracking problem, but particle filtering tends to break down in very high dimensional spaces. To address this issue, we plan to use particle filtering with collapsed particles. For our localization and mapping problem, we have two obvious ways of constructing these collapsed particles. We now consider one such approach in detail, and briefly consider the implications of the other.

1. **[15 points]** We select the robot pose trajectory $X^{(1:t)}$ as the set of sampled variables, and maintain a closed form distribution over $\boldsymbol{L}$. Thus, at time t we have a set of weighted particles, each of which specifies a full sampled trajectory $\boldsymbol{x}^{(1:t)}[m]$, a distribution over landmark locations $P_m^{(t)}(\boldsymbol{L})$, and a weight $w^{(t)}[m]$.

   (a) The distribution over $\boldsymbol{L}$ is still exponentially large in principle. Show how you can represent it compactly in closed form. (Hint: Write out the full form of $P_m^{(t)}(\boldsymbol{L})$ recalling the definition of collapsed particles, and note that our samples represent entire pose trajectories $X^{(1:t)}$.)

   Answer: Since each particle instantiates all of $x^{(1:t)}[m]$ and $\boldsymbol{o}^{(1:t)}[m]$, there is no active trail from L_i to L_j for $i \neq j$, and so the L_i variables are all independent, given a particle. Also, each landmark's location is independent of the observations of the other landmarks, given a trajectory. Thus we can decompose our distribution over the landmarks' locations as the following product of marginals:

$$\begin{aligned}
P_m^{(t)}(\boldsymbol{L}) &= P(\boldsymbol{L} \mid x^{(1:t)}[m], \boldsymbol{o}^{(1:t)}[m]) \\
&= \prod_i P(L_i \mid x^{(1:t)}[m], \boldsymbol{o}^{(1:t)}[m]) \\
&= \prod_i P(L_i \mid x^{(1:t)}[m], o_i^{(1:t)}[m]) \\
&= \prod_i P_m^{(t)}(L_i)
\end{aligned}$$

Note that it is not in closed form yet. We will show how to compute $P_m^{(t)}(L_i)$ in part ii.

(b) Describe how to perform a forward propagation step in our collapsed particle filtering algorithm. Specifically show how to:

- generate a new set of particles for time $t+1$
- compute the weight $w^{(t+1)}[m]$ of each particle;
- compute the distribution $P_m^{(t+1)}(\boldsymbol{L})$.

Answer: To generate new particles, we basically just follow standard particle filtering – normalize the weights of the particles from the previous timeslice; select a particle under that normalized distribution; then sample its forward propagation based on the motion model $P(X^{(t+1)} \mid X^{(t)})$.

To see how to compute our new distribution over $\boldsymbol{L}$, we first make the following observation:

$$\begin{aligned}
P_m^{(t+1)}(L_i) &= P(L_i \mid x^{(1:t+1)}[m], o_i^{(1:t+1)}[m]) \\
&= \frac{1}{Z} P(o_i^{(1:t+1)}[m] \mid L_i, x^{(1:t+1)}[m]) P(L_i \mid x^{(1:t+1)}[m]) \\
&= \frac{1}{Z} P(o_i^{(1:t+1)}[m] \mid L_i, x^{(1:t+1)}[m]) P(L_i) \\
&= \frac{1}{Z} P(L_i) \prod_{t'=1}^{t+1} P(o_i^{(t')}[m] \mid L_i, x^{(t')}[m])
\end{aligned}$$

for some normalizing constant

$$Z = \sum_{l_i} P(l_i) \prod_{t'=1}^{t+1} P(o_i^{(t')}[m] \mid l_i, x^{(t')}[m])$$

which does not depend on L_i. Note that $P(L_i)$ is the prior for the location of the landmark, as encoded by $\mathcal{B}_s$ (or $\mathcal{B}_0$ if using deterministic CPDs to propagate the L_i variables through all timeslices); and each factor of the product is just a lookup in the known measurement model. However, since

$$P_m^{(t)}(L_i) = \frac{1}{Z'} P(L_i) \prod_{t'=1}^{t} P(o_i^{(t')}[m] \mid L_i, x^{(t')}[m])$$

for some other normalizing constant

$$Z' = \sum_{l_i} P(l_i) \prod_{t'=1}^{t} P(o_i^{(t')}[m] \mid l_i, x^{(t')}[m])$$

which similarly does not depend on L_i, then

$$P_m^{(t+1)}(L_i) = \frac{1}{Z''} P_m^{(t)}(L_i) P(o_i^{(t+1)}[m] \mid L_i, x^{(t+1)}[m])$$

for some normalizing constant $Z'' = \frac{Z}{Z'}$ which does not depend on L_i. Thus, to update our beliefs about the location of the landmarks, we just multiply in the probability of the new observations from the measurement model and renormalize.

Having updated our distribution over the locations of the landmarks, we now need to compute the weight for the particle. In standard particle filtering, we use the likelihood of the new observations given the complete particle; here, our particle does not contain the location of the landmarks, so we must compute our weights using the other given information. We can consider the reweighting to be given all of the previously sampled variables together with all observations. Using this insight, we can compute the particles by marginalizing out the landmark locations:

$$\begin{aligned}
w^{(t+1)}[m] &= P(\boldsymbol{o}^{(t+1)}[m] \mid x^{(1:t+1)}[m], \boldsymbol{o}^{(1:t)}[m]) \\
&= \sum_{\boldsymbol{l} \in Val(\boldsymbol{L})} P(\boldsymbol{o}^{(t+1)}[m], \boldsymbol{l} \mid x^{(1:t+1)}[m], \boldsymbol{o}^{(1:t)}[m]) \\
&= \sum_{\boldsymbol{l} \in Val(\boldsymbol{L})} P(\boldsymbol{l} \mid x^{(1:t+1)}[m], \boldsymbol{o}^{(1:t)}[m]) P(\boldsymbol{o}^{(t+1)}[m] \mid x^{(1:t+1)}[m], \boldsymbol{o}^{(1:t)}[m], \boldsymbol{l}) \\
&= \sum_{\boldsymbol{l} \in Val(\boldsymbol{L})} P(\boldsymbol{l} \mid x^{(1:t)}[m], \boldsymbol{o}^{(1:t)}[m]) P(\boldsymbol{o}^{(t+1)}[m] \mid x^{(t+1)}[m], \boldsymbol{l}) \\
&= \sum_{\boldsymbol{l} \in Val(\boldsymbol{L})} \prod_{i=1}^{n} P(l_i \mid x^{(1:t)}[m], o_i^{(1:t)}[m]) P(o_i^{(t+1)}[m] \mid x^{(t+1)}[m], l_i) \\
&= \sum_{\boldsymbol{l} \in Val(\boldsymbol{L})} \prod_{i=1}^{n} P_m^{(t)}(l_i) P(o_i^{(t+1)}[m] \mid x^{(t+1)}[m], l_i)
\end{aligned}$$

Thus we see how to compute the weight for the particle in the new timeslice using the normalized distribution $P_m^{(t)}(\boldsymbol{L})$ from the previous timeslice, together with our measurement model $P(o_i^{(t)}[m] \mid x^{(t)}[m], l_i)$.

2. **[5 points]** As another alternative, we can select the landmark positions $\boldsymbol{L}$ as our set of sampled variables. Thus, for each particle $\boldsymbol{l}[m]$, we maintain a distribution $P_m^{(t)}(X^{(t)})$. Without describing the details of how to perform a forward propagation step in our collapsed particle filtering algorithm, we can think about the effectiveness of such an approach. In this algorithm, what will happen to the particles and their weights eventually (i.e., after a large number of time steps)? What are the necessary conditions for this algorithm to converge to the correct map?

   Answer: Since we never generate any new assignments $\boldsymbol{l}[m]$, on each forward propagation, with some nonzero probability, one or more of the particles will "die off". Thus eventually, we will be left with a single particle $\boldsymbol{l}[m]$ with a weight of 1.

   The single map to which we converge will be the *correct* map only if our initial generation of particles included the correct map among its hypotheses. Since we never generate any new hypotheses, we certainly cannot converge on the right one if we failed to start with it as a hypothesis. However, although this is necessary, it is not sufficient, since even if we start with the correct hypothesis as an available option, it is possible that by chance we might discard it, especially early on when, with few observations, its weight may not be much more than that of the other hypotheses.

As there are exponentially many assignments to $\boldsymbol{L}$, unless we have a very well-specified prior over them, it is highly unlikely the correct solution will be generated as an initial hypothesis.

Chapter 18

Exercise 18.8 — Bayesian score structure equivalence I [5 points]

Show that if $\mathcal{G}$ is I-equivalent to $\mathcal{G}'$, then if we use table CPDs, we have that $\text{score}_L(\mathcal{G} : \mathcal{D}) = \text{score}_L(\mathcal{G}' : \mathcal{D})$ for any choice of $\mathcal{D}$.

Hint: consider the set of distributions that can be represented by parameterization each network structure.

Answer: Using table CPDs for $\mathcal{G}$, we can represent any joint probability distribution that complies with the conditional independence implied by the $\mathcal{G}$. Since $\mathcal{G}'$ implies the same conditional independence as $\mathcal{G}$, they represent exactly the same set of distributions. Therefore, for any parameterization $\boldsymbol{\theta}_{\mathcal{G}}$ of $\mathcal{G}$, there is a corresponding parameterization $\boldsymbol{\theta}_{\mathcal{G}'}$ of $\mathcal{G}'$ which generates the same distribution. Since the likelihood of data set $\mathcal{D}$ only depends on the underlying distribution, we have:

$$\text{score}_L(\mathcal{G} : \mathcal{D}) = \ell(\langle \mathcal{G}, \hat{\boldsymbol{\theta}}_{\mathcal{G}} \rangle : \mathcal{D}) = \ell(\langle \mathcal{G}', \boldsymbol{\theta}_{\mathcal{G}'} \rangle : \mathcal{D}) \leq \text{score}_L(\mathcal{G}' : \mathcal{D})$$

where $\hat{\boldsymbol{\theta}}_{\mathcal{G}}$ are the maximum likelihood parameters for $\mathcal{G}$, and $\boldsymbol{\theta}_{\mathcal{G}'}$ are the corresponding parameters in $\mathcal{G}'$.

By the same reasoning, we have $\text{score}_L(\mathcal{G}' : \mathcal{D}) \leq \text{score}_L(\mathcal{G} : \mathcal{D})$. Therefore:

$$\text{score}_L(\mathcal{G} : \mathcal{D}) = \text{score}_L(\mathcal{G}' : \mathcal{D})$$

Exercise 18.9 — Bayesian score structure equivalence II [10 points]

Show that if $\mathcal{G}$ is I-equivalent to $\mathcal{G}'$, then if we use table CPDs, we have that $\text{score}_{BIC}(\mathcal{G} : \mathcal{D}) = \text{score}_{BIC}(\mathcal{G}' : \mathcal{D})$ for any choice of $\mathcal{D}$.

Answer: The BIC score is defined as: $\text{score}_{BIC}(\mathcal{G} : \mathcal{D}) = \text{score}_L(\mathcal{G} : \mathcal{D}) - \frac{\log M}{2}\text{Dim}[\mathcal{G}]$ and by (6a) we know $\text{score}_L(\mathcal{G} : \mathcal{D}) = \text{score}_L(\mathcal{G}' : \mathcal{D})$. Therefore, to prove $\text{score}_{BIC}(\mathcal{G} : \mathcal{D}) = \text{score}_{BIC}(\mathcal{G}' : \mathcal{D})$, we only need to show $\text{Dim}[\mathcal{G}] = \text{Dim}[\mathcal{G}']$.

We first consider the case where $\mathcal{G}$ and $\mathcal{G}'$ only differs by one safe edge reversal such that $X \to Y \in \mathcal{G}$ and $X \leftarrow Y \in \mathcal{G}'$. All the other variables except X and Y have the same parents in $\mathcal{G}$ and $\mathcal{G}'$ and thus have the same number of parameters. Assume variables $\mathcal{Z}$ are the common parents of X and Y. Denote $\text{SIZE}_{\mathcal{G}}(N)$ to be the number of parameters for variable N in $\mathcal{G}$:

$$\begin{aligned}
& \text{Dim}[\mathcal{G}] \\
= & \sum_{\substack{N \in \mathcal{G} \\ N \neq X, Y}} \text{SIZE}_{\mathcal{G}}(N) + \text{SIZE}_{\mathcal{G}}(X) + \text{SIZE}_{\mathcal{G}}(Y) \\
= & \sum_{\substack{N \in \mathcal{G} \\ N \neq X, Y}} \text{SIZE}_{\mathcal{G}}(N) + (|Val(()X)| - 1)|Val(Z)| + (|Val(Y)| - 1)|Val(Z)||Val(X)| \\
= & \sum_{\substack{N \in \mathcal{G} \\ N \neq X, Y}} \text{SIZE}_{\mathcal{G}}(N) + (|Val(Y)||Val(Z)||Val(X)| - |Val(Z)| \\
= & \sum_{\substack{N \in \mathcal{G}' \\ N \neq X, Y}} \text{SIZE}_{\mathcal{G}'}(N) + (|Val(()Y)| - 1)|Val(Z)| + (|Val(X)| - 1)|Val(Z)||Val(Y)| \\
= & \sum_{\substack{N \in \mathcal{G}' \\ N \neq X, Y}} \text{SIZE}_{\mathcal{G}'}(N) + \text{SIZE}_{\mathcal{G}'}(Y) + \text{SIZE}_{\mathcal{G}'}(X) \\
= & \text{Dim}[\mathcal{G}']
\end{aligned}$$

In general, for any two I-equivalent $\mathcal{G}$ and $\mathcal{G}'$, they can be connected by a sequence of safe edge reversal. Therefore, $\mathcal{G}$ and $\mathcal{G}'$ have the same number of parameters and thus the same BIC score.

Exercise 18.10 — Bayesian score structure equivalence III [10 points]

Show that the Bayesian score with a K2 prior in which we have a Dirichlet prior $Dirichlet(1, 1, \ldots, 1)$ for each set of multinomial parameters is not score equivalent.

Hint: construct a data set for which the score of the network $X \to Y$ differs from the score of the network $X \leftarrow Y$.

Answer: Construct $\mathcal{G}$ to be $X \to Y$ and $\mathcal{G}'$ to be $X \leftarrow Y$. They are I-equivalent. Construct $\mathcal{D}$ to be $\{(x^1, y^1), (x^1, y^0)\}$.

$$P(\mathcal{D}|\mathcal{G}) = \frac{1 \cdot 2}{2 \cdot 3}\frac{1 \cdot 1}{2 \cdot 3} = \frac{1}{18}$$

On the other hand:

$$P(\mathcal{D}|\mathcal{G}') = \frac{1 \cdot 1}{2 \cdot 3}\frac{1}{2}\frac{1}{2} = \frac{1}{24}$$

Therefore, their Bayesian scores are different assuming equal structure prior.

Exercise 18.26 — Ordering-based search in structure learning [12 points]

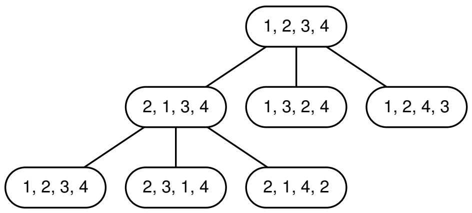


Figure 5: Partial search tree example for orderings over variables X_1, X_2, X_3, X_4. Successors to $\prec = (1, 2, 3, 4)$ and $\prec' = (2, 1, 3, 4)$ shown.

Consider learning the structure of a Bayesian network for some given ordering, $\prec$, of the variables, $X_1, \ldots, X_n$. This can be done efficiently, as described in section 18.4.2 of the course reader. Now assume that we want to perform search over the space of orderings, i.e. we are searching for the network (with bounded in-degree k) that has the highest score. We do this by defining the score of an ordering as the score of the (bounded in-degree) network with the maximum score consistent with that ordering, and then searching for the ordering with the highest score. We bound the in-degree so that we have a smaller and smoother search space.

We will define our search operator, o, to be "Swap X_i and X_{i+1}" for some $i = 1, \ldots, n-1$. Starting from some given ordering, $\prec$, we evaluate the BIC-score of all successor orderings, $\prec'$, where a successor ordering is found by applying o to $\prec$ (see Figure 5). We now choose a particular successor, $\prec'$. Provide an algorithm for computing as efficiently as possible the BIC-score for the successors of the new ordering, $\prec'$, given that we have already computed the scores for successors of $\prec$.

Answer: Notation: Let h be the variable that was swapped for $\prec'$, let i be the variable to be swapped for candidate $\prec''$, and let $Succ_j(\prec)$ be the jth candidate successor of $\prec$.

The BIC-score is decomposable so we only need to consider the family scores. We cache all family scores of $\prec$ as well as all family scores of all candidates for $\prec'$. We can discard all previously cached values prior to those pertaining to $\prec$. We consider the family score $\mathrm{FamScore}_j(\prec'' : \mathcal{D})$ for each variable $X_j, j = 1, \ldots, n$:

- If $j > i+1$, $\mathrm{FamScore}_j(\prec'' : \mathcal{D}) = \mathrm{FamScore}_j(\prec' : \mathcal{D})$.
- If $j < i$, $\mathrm{FamScore}_j(\prec'' : \mathcal{D}) = \mathrm{FamScore}_j(\prec' : \mathcal{D})$.
- If $j = i$ or $j = i+1$:
  - If $i = h$, $\mathrm{FamScore}_j(\prec'' : \mathcal{D}) = \mathrm{FamScore}_j(\prec : \mathcal{D})$.
  - If $i > h+1$ or $i < h-1$, $\mathrm{FamScore}_j(\prec'' : \mathcal{D}) = \mathrm{FamScore}_j(Succ_i(\prec) : \mathcal{D})$.
  - If $i = h+1$ or $i = h-1$, compute the family score from scratch: $\mathrm{FamScore}_j(\prec'' : \mathcal{D}) = \mathrm{argmax}_{\boldsymbol{U} \in \{X_k : X_k \prec'' X_j\}} \mathrm{FamScore}(X_j \mid \boldsymbol{U}_j : \mathcal{D})$

Problem 18.22 — Module network learning [15 points]

In this problem, we will consider the task of learning a generalized type of Bayesian networks that involves shared structure and parameters. Let $\mathcal{X}$ be a set of variables, which we assume are all binary-valued. A *module network* over $\mathcal{X}$ partitions the variables $\mathcal{X}$ into K disjoint clusters, for $K \ll n = |\mathcal{X}|$. All of the variables assigned to the same cluster have precisely the same parents and CPD. More precisely, such a network defines:

- An assignment function $\mathcal{A}$, which defines for each variable X, a cluster assignment $\mathcal{A}(X) \in \{C_1, \ldots, C_K\}$.
- For each cluster $C_k (k = 1, \ldots, K)$, a graph $\mathcal{G}$ which defines a set of parents $\mathbf{Pa}_{C_k} = \boldsymbol{U}_k \subset \mathcal{X}$ and a CPD $P_k(X \mid \boldsymbol{U}_k)$.

The cluster network structure defines a ground Bayesian network where, for each variable X, we have the parents $\boldsymbol{U}_k$ for $k = \mathcal{A}(X)$ and the CPD $P_k(X \mid \boldsymbol{U}_k)$. Figure 6 shows an example of such a network.

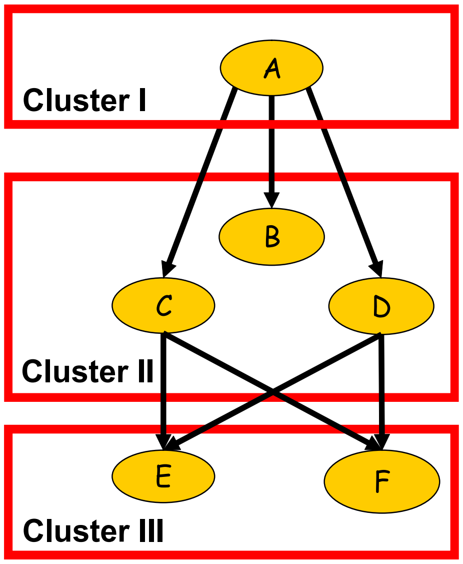


Figure 6: A simple module network

Assume that our goal is to learn a cluster network that maximizes the BIC score given a data set $\mathcal{D}$, where we need to learn both the assignment of variables to clusters and the graph structure.

1. **[5 points]** Define an appropriate set of parameters and an appropriate notion of sufficient statistics for this class of models, and write down a precise formula for the likelihood function of a pair $(\mathcal{A}, \mathcal{G})$ in terms of the parameters and sufficient statistics.

   Answer: To parameterize a module network, we only need one CPD for each cluster, i.e., $P_k(X|U_k), k = 1, 2, \ldots, K$. The set of parameters for each $P_k(X|U_k)$ is $\{\theta_{x^1|u_k}|u_k \in |Val(U_k)|\}$. The sufficient statistics for each CPD are simply aggregated from each family in the cluster. Given the sufficient statistics for each cluster, the likelihood function looks the same as the ordinary BN case (view each cluster as a single family with the CPD $P_k(X|U_k)$).

2. We use greedy local search to learn the structure of the cluster network. We will use the following types of operators (each operation should remain the graph acyclic):

   - **Add** operators that add a parent for a cluster;
   - **Delete** operators that delete a parent for a cluster;
   - **Node-Move** operators $o_{k \to k'(X)}$ that change from $\mathcal{A}(X) = k$ to $\mathcal{A}(X) = k'$. (If $X \in \mathbf{Pa}_{C_{k'}}$, moving X to k' is not allowed.)

   As usual, we want to reduce the computational cost by caching our evaluations of operators and reusing them from step to step.

   (a) **[2 points]** Why did we not include edge reversal in our set of operators? **Answer:** In ordinary BN, an edge represents a one-to-one relationship between two variable, however, in the module networks, an edge from a variable to another variable actually represents a one-to-many relation between the variable and all the variables in the cluster. So the relation is not symmetry, reversing the edge is not well-defined.

   (b) **[2 points]** Describe an efficient implementation for the update associated with the **Node-Move** operator. **Answer:** When X is moved from cluster k to k', only the local scores of these two modules change. So the rescoring of these two modules can be efficiently calculated by substracting X's statistics from the sufficient statistics of C_k and adding the sufficient statistics regarding X to the overall sufficient statistics of $C_{k'}$.

   (c) **[6 points]** For each type of operator, specify which other operators need to be reevaluated once the operator has been taken. Briefly justify your response. (Suppose that we cache and update evaluations of operators and reuse them to save the computation.)

   Answer: For adding and deleting operators, if the operator changes the parents of cluster C_k, then we only need to update the changes in score for those operators that involve C_k. For moving operators, if the operator is to move a variable from cluster k to k', then we only need to update the changes in score for those operators that involve C_k and $C_{k'}$.

New exercise (chapters 12, 18) — Sampling on a Tree [15 points]

Suppose we have a distribution $P(\boldsymbol{X}, \boldsymbol{E})$ over two sets of variables $\boldsymbol{X}$ and $\boldsymbol{E}$. Our distribution is represented by a nasty Bayes Net with very dense connectivity, and our sets of variables $\boldsymbol{X}$ and $\boldsymbol{E}$ are spread arbitrarily throughout the network. In this problem our goal is to use the sampling methods we learned in class to estimate the posterior probability $P(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{E} = \boldsymbol{e})$. More specifically, we will use a tree-structured Bayes Net as the proposal distribution for use in the importance sampling algorithm.

1. [**1 points**] For a particular value of $\boldsymbol{x}$ and $\boldsymbol{e}$, can we compute $P(\boldsymbol{x} \mid \boldsymbol{e})$ exactly, in a tractable way? Can we sample directly from the distribution $P(\boldsymbol{X} \mid \boldsymbol{e})$? Can we compute $\tilde{P}(\boldsymbol{x} \mid \boldsymbol{e}) = P(\boldsymbol{x}, \boldsymbol{e})$ exactly, in a tractable way? For each question, provide a Yes/No answer and a single sentence explanation or description.

   Answer: No, No, Yes

2. [**7 points**] Now, suppose your friendly TAs have given you a tree network, where X_1 is the root and each X_i for $i \neq 1$ has exactly one parent $X_{p(i)}$. They tell you that the distribution $Q(\boldsymbol{X}, \boldsymbol{E})$ defined by this network is "close" to the distribution $P(\boldsymbol{X}, \boldsymbol{E})$. You now want to use *the posterior* in Q as your proposal distribution for importance sampling.

   (a) Show how to sample from the posterior in Q. More specifically, provide an explicit construction for a clique tree over this network, and show how to use it to compute the posterior $Q(\boldsymbol{X} \mid \boldsymbol{E} = \boldsymbol{e})$. Describe how to sample from this posterior, once it has been computed. **Answer:** Create a clique tree where each clique is a pair of variables (child and parent). Multiply in the indicator functions for the evidence $\boldsymbol{E} = \boldsymbol{e}$. The distribution across the tree now represents $Q(\boldsymbol{X}, \boldsymbol{E} \mid \boldsymbol{E} = \boldsymbol{e})$. Calibrate the tree. Now, the belief at a clique over $(X, \mathbf{Pa}_X)$ is proportional to $Q(X, \mathbf{Pa}_X \mid \boldsymbol{E} = \boldsymbol{e})$. From this belief, we can easily compute $Q(X \mid \mathbf{Pa}_X, \boldsymbol{E} = \boldsymbol{e})$ (use Bayes' Rule). Using these CPDs, we can now forward sample directly from the posterior in Q (sample the first variable, then instantiate it in neighboring cliques, repeating to forward sample).

   (b) Now you must reweight the samples according to the rules of importance sampling. You want your weighted samples to accurately represent the actual posterior in the original network $P(\boldsymbol{X} \mid \boldsymbol{E} = \boldsymbol{e})$. Show precisely how you determine the weights $w[m]$ for the samples. **Answer:** The weight of the particle is:

$$w[m] = \frac{P'(\boldsymbol{x}[m], \boldsymbol{e}[m])}{Q(\boldsymbol{x}[m], \boldsymbol{e}[m] \mid \boldsymbol{e})}$$

   (c) Show the form of the final estimator $\hat{P}(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{E} = \boldsymbol{e})$ for $P(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{E} = \boldsymbol{e})$, in terms of the samples from part i, and the weights from part ii. **Answer:** The estimator:

$$\hat{P}(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{E} = \boldsymbol{e}) = \frac{\sum_{m=1}^{M} w[m] 1\{\boldsymbol{x}[m] = \boldsymbol{x}\}}{\sum_{m=1}^{M} w[m]}$$

3. [**7 points**] Examining the results of your sampling, you find that your TAs were wrong, and the original tree from which you sampled did not provide very good estimates for the posterior. You therefore decide to produce a better tree, and rerun the importance sampling again. This better tree should represent a distribution which is "closer" to the posterior from which you are trying to sample. Show how you can use the samples produced in the previous step to learn the best tree (i.e., a tree that best represents the posterior distribution as estimated in part b, according to the likelihood score). Assume you have access to a maximum spanning tree algorithm, and your job is to define the weights for the edges in terms of the estimator computed above. **Answer:** We will use the maximum spanning tree algorithm to select edges where the weight for a candidate edge between X_i and X_j is computed as:

$$\begin{aligned} w_{ij} &= FamScore(X_i \mid X_j : \mathcal{D}) - FamScore(X_i : \mathcal{D}) \\ &= I(X_i; X_j) + H(X_i) \\ &= \sum_{x_i}\sum_{x_j} \bar{M}[x_i, x_j] \log \frac{\bar{M}[x_i, x_j]}{\bar{M}[x_i]} - \sum_{x_i} \bar{M}[x_i] \log \frac{\bar{M}[x_i]}{\bar{M}} \end{aligned}$$

where the $\bar{M}$ terms are the sufficient statistics from the weighted dataset:

$$\bar{M}[x_i, x_j] = \sum_m 1\{x_i[m] = x_i, x_j[m] = x_j\} w[m],$$

etc.

Exercise 18.23 — Optimal reinsertion structure learning [10 points]

More complex search operators are sometimes useful in Bayesian Network structure learning. In the problem we will consider an operator that we will call the *Reinsertion* operator. This operator works as follows:

Assume that we have been searching for awhile, and our current structure is $\mathcal{G}_t$. We now choose a variable X to be our *target variable.* The first step is to remove the variable from the network, by severing all connections into and out of this target variable. We then select the optimal set of parents and children for X, and reinsert it into the network with edges from the selected parents and to the selected children. In addition, consider that we will limit the number of parents to a variable to K_p and the number of children to K_c. This restriction helps us to avoid the need to consider overly complex networks.

Throughout this problem, assume the use of the BIC score for structure evaluation.

1. [**2 points**] Let X_i be our current target variable, and assume for the moment that we have somehow chosen $\boldsymbol{U}_i$ to be optimal parents of X_i. Consider the case of $K_c = 1$, where we want to choose the *single* optimal child for X_i. Candidate children — those that do not introduce a cycle in the graph — are $Y_1, \ldots, Y_\ell$. Write an argmax expression for finding the optimal child C. Explain your answer.

   Answer: Because the BIC score is decomposable by family, we can consider only the family of the proposed child when computing the change in score that would result from adding that child. Thus, we choose the child Y_C such that:

   $$\begin{aligned} Y_C &= \arg\max_{Y_i} FamScore(Y_i \mid \mathbf{Pa}_{Y_i}, X) \\ &= \arg\max_{Y_i} \left[M \left[\boldsymbol{I}_{\hat{P}}(Y_i; \mathbf{Pa}_{Y_i}, X) - \boldsymbol{I}_{\hat{P}}(Y_i; \mathbf{Pa}_{Y_i}) \right] - \frac{\log M}{2} \Delta Dim \right] \end{aligned}$$

   We choose this Y_c only among variables that are not ancestors of any of the variables in $\boldsymbol{U}_i$.

2. [**4 points**] Now consider the case of $K_c = 2$. How do we find the optimal pair of children? Assuming that our family score for any $\{X_k, \boldsymbol{U}_k\}$ can be computed in a constant time f, what is the best asymptotic computational complexity of finding the optimal pair of children? Explain. Extend your analysis to larger values of K_c. What is the computational complexity of this task?

   Answer: For the same reason as above, we can "score" each child independently, and choose the top K_c of them. The total cost for $K_c = 2$ children is $O(fN)$ and for general

K_c we can compute the scores, sort them, and select the top candidates. The complexity of this is $O(fN + N \log N)$. It is also possible to use a better sort if you only need the top K_c candidates, and that the complexity becomes $O(fN + N \log K_c)$.

3. [**3 points**] We now consider the choice of parents for X_i. We now assume that we have already somehow chosen the optimal set of children and will hold them fixed. Can we do the same trick when choosing the parents? If so, show how. If not, argue why not.

   Answer: No we cannot do this. The family score for the child cannot be evaluated independently for each parent. Instead we must consider all combinations of parents. The complexity of this task is exponential in K_p.

New exercise — Structure learning for tree-augmented naive Bayes [**21 points**] Now, consider the problem of learning a classifier network structure which augments the Naive Bayes model. Here, we have a class variable C in addition to the feature variables $X_1, \ldots, X_n$.

1. [**5 points**] We begin by assuming that the class variable C is connected to each of the X_i's, but in addition, each of the X_i's may have at most one additional parent. Modify the algorithm of section 18.4.1 to find the highest-scoring network subject to these constraints. You may assume that we use a structure-equivalent scoring function.

   Answer: In this problem, we again are optimizing with respect to the BIC score, but this time, our baseline graph is the regular Naive Bayes model (which we shall call $\mathcal{G}_{NB}$), which contains an edge from the class variable C to each feature X_i. Thus, we are interested in maximizing

$$\Delta(\mathcal{G}) = \text{score}(\mathcal{G} \ : \ \mathcal{D}) - \text{score}(\mathcal{G}_{NB} \ : \ \mathcal{D}).$$

   To do this, we redefine the weights

$$\begin{aligned} w_{X_j \to X_i} &= \text{FamScore}(X_i \mid X_j, C \ : \ \mathcal{D}) - \text{FamScore}(X_i \mid C \ : \ \mathcal{D}) \\ &= M[\boldsymbol{I}_{\hat{P}}(X_i; X_j, C) - \boldsymbol{I}_{\hat{P}}(X_i; C)] - \frac{\log M}{2}\big(|Val(X_i) - 1||Val(X_j) - 1||C|\big) \\ &= M[\boldsymbol{I}_{\hat{P}}(X_i; X_j, C) - \boldsymbol{I}_{\hat{P}}(X_i; C)] - \log M. \end{aligned}$$

   As before, it is a useful and easy exercise to check that $w_{X_j \to X_i} = w_{X_i \to X_j}$. Thus, picking a graph which maximizes $\Delta(\mathcal{G})$ simply involves using an maximum weight *undirected* spanning tree or forest approach identical to the procedure given in part (a). Using such a method to find edges between the X_i's, we finish by directing all edges (as before) and adding an edge from C to each X_i.

2. [**16 points**] (*) Now, assume that we allow ourselves even more flexibility: for each X_i, we need to decide separately whether to add C as a parent or not. We want to pick the highest scoring network where each node X_i may or may not have C as a parent, and each X_i can have at most one additional parent X_j. (The node C is always a root.) Show how to reduce this problem to one of finding the maximum-weight *directed* spanning tree. (**Hint:** Consider separately each possible family for each node X_i.)

   Note: You do not need to present an algorithm for computing maximum-weight directed spanning trees, but only to show how you could solve the problem of finding an optimal BN of this type if you had a maximum-weight directed spanning tree algorithm to use as a subroutine.

Answer: There are several possible correct answers to this problems; we sketch a few working approaches here.

Method 1

We start with the empty graph ($\mathcal{G}_0$) as our baseline. For each X_i, the possible (non-empty) parent sets of X_i are: (1) $\{X_j\}$ (for some X_j), (2) $\{X_j, C\}$ (for some X_j), and (3) $\{C\}$. Relative to the empty graph, the change in BIC score resulting from using each of the parent sets above is:

$$\begin{aligned}
\Delta_{\{X_j\}} &= \text{FamScore}(X_i \mid X_j \ : \ \mathcal{D}) - \text{FamScore}(X_i \ : \ \mathcal{D}) \\
&= M\boldsymbol{I}_{\hat{P}}(X_i; X_j) - \frac{\log M}{2} \\
\Delta_{\{X_j, C\}} &= \text{FamScore}(X_i \mid X_j, C \ : \ \mathcal{D}) - \text{FamScore}(X_i \ : \ \mathcal{D}) \\
&= M\boldsymbol{I}_{\hat{P}}(X_i; X_j, C) - \frac{3\log M}{2} \\
\Delta_{\{C\}} &= \text{FamScore}(X_i \mid C \ : \ \mathcal{D}) - \text{FamScore}(X_i \ : \ \mathcal{D}) \\
&= M\boldsymbol{I}_{\hat{P}}(X_i; C) - \frac{\log M}{2}
\end{aligned}$$

Our goal is to pick a single parent set (which may be empty) for each X_i while making sure that the generated network is acyclic. To do this, consider a graph over the nodes $C, X_1, \ldots, X_n$, with edges from the class variable C to each feature X_i and from each feature X_j to every other feature X_i. We use the edge weights $w_{C \to X_i} = \Delta_{\{C\}}$ and $w_{X_j \to X_i} = \max(\Delta_{\{X_j, C\}}, \Delta_{\{X_j\}})$. Note that in the latter term, we take a $\max$ since the presence or absence of C in the parent set of X_i does not make a difference with regard to acyclicity of the generated network, as sorted out during the maximum directed spanning forest algorithm; thus, we can simply use the higher scoring option.

Given a maximum directed spanning forest $\mathcal{F}$ for this graph, we define the parent set of each feature X_i in the resulting network as follows:

$$\mathbf{Pa}_i^{\mathcal{G}} = \begin{cases} \{X_j\} & \text{if } X_j \to X_i \in \mathcal{F} \text{ and } \Delta_{\{X_j\}} \geq \Delta_{\{X_j, C\}}, \\ \{X_j, C\} & \text{if } X_j \to X_i \in \mathcal{F} \text{ and } \Delta_{\{X_j\}} < \Delta_{\{X_j, C\}}, \\ \{C\} & \text{if } C \to X_i \in \mathcal{F}, \text{ and} \\ \emptyset & \text{otherwise.} \end{cases}$$

By construction, these four cases are disjoint and hence the parent set of each feature is well-defined. Furthermore, it is easy to verify that any cycle in the generated network implies the existence of a corresponding cycle in $\mathcal{F}$, an impossibility. (To see this, note that $X_j \to X_i$ in the network can only happen if $X_j \to X_i \in \mathcal{F}$ by the definition given above; apply this to all edges in a cycle of the generated network.) Thus, any spanning forest gives a well-formed Bayesian network. Similarly, one may also show that any valid Bayesian network which meets the constraints of the problem statement gives rise to a valid directed spanning forest. Since total weight of any forest is identical to the BIC score of the corresponding network, it follows that this algorithm generates an optimal BIC-scoring network subject to the problem constraints.

As a final note, if we are restricted to using a maximum weighted directed spanning *tree* algorithm, the method of zeroing out negative weights described in part (a) works here as well.

Method 2

We again use $\mathcal{G}_0$ as our baseline, and define $\Delta_{\{X_j\}}$, $\Delta_{\{X_j,C\}}$, and $\Delta_{\{C\}}$ as before for each X_i. For the reduction to the maximum weighted directed spanning forest problem, we define a graph containing C and $X_1,\ldots,X_n$ as before, with directed edges from C to each X_i and from each X_j to X_i. This time, however, we also introduce new nodes $CX_1,\ldots,CX_n$, and create edges from CX_j to X_i for all $i \neq j$, and edges from X_i to CX_i for all i. For each X_i, the weight of the C to X_i edge is $\Delta_{\{C\}}$, the weight of the X_j to X_i edge (for all $j \neq i$) is $\Delta_{\{X_j\}}$, the weight of the CX_j to X_i edge (for all $j \neq i$) is $\Delta_{\{X_j,C\}}$, and the weight of the X_i to CX_i is set to a sufficiently large positive constant (to ensure that the X_i to CX_i is *always* selected by the maximum weighted directed spanning forest algorithm).

Given a maximum directed spanning forest $\mathcal{F}$ for this graph, we define the parent set of each feature X_i in the resulting network as follows:

$$\mathbf{Pa}_i^{\mathcal{G}} = \begin{cases} \{X_j\} & \text{if } X_j \to X_i \in \mathcal{F} \\ \{X_j, C\} & \text{if } CX_j \to X_i \in \mathcal{F} \\ \{C\} & \text{if } C \to X_i \in \mathcal{F}, \text{ and} \\ \emptyset & \text{otherwise.} \end{cases}$$

Clearly, the parent set of each X_i is well-defined. Again, we can check that cycles in the generated network imply cycles in $\mathcal{F}$. (As a simple example, note that if $X_j \to X_i \in \mathcal{F}$, then selecting either $X_i \to X_j$ or $CX_i \to X_j$ for the spanning forest will generate a cycle *because of the required edge* $X_i \to CX_i$; you can generalize this reasoning to cycles of length greater than two.) The remaining arguments for correctness follow similarly to those given in Method 1.

Method 3

In the last method, we do not use $\mathcal{G}_0$ as the baseline; rather, we use a graph $\mathcal{G}_0'$ in which we include the $C \to X_i$ edge for each X_i such that $\Delta_{\{C\}} > 0$ (using the definition from Method 1). We then redefine for each X_i,

$$\begin{aligned} \Delta_{\{X_j\}} &= \mathrm{FamScore}(X_i \mid X_j : \mathcal{D}) - \max(\mathrm{FamScore}(X_i : \mathcal{D}), \mathrm{FamScore}(X_i \mid C : \mathcal{D})) \\ \Delta_{\{X_j,C\}} &= \mathrm{FamScore}(X_i \mid X_j, C : \mathcal{D}) - \max(\mathrm{FamScore}(X_i : \mathcal{D}), \mathrm{FamScore}(X_i \mid C : \mathcal{D})) \end{aligned}$$

and create a graph over only the X_i's such that each edge $X_j \to X_i$ is given weight $\max(\Delta_{\{X_j\}}, \Delta_{\{X_j,C\}})$. Given a maximum directed spanning forest $\mathcal{F}$ for this graph, we define the parent set of each feature X_i in the resulting network as follows:

$$\mathbf{Pa}_i^{\mathcal{G}} = \begin{cases} \{X_j\} & \text{if } X_j \to X_i \in \mathcal{F} \text{ and } \Delta_{\{X_j\}} \geq \Delta_{\{X_j,C\}}, \\ \{X_j, C\} & \text{if } X_j \to X_i \in \mathcal{F} \text{ and } \Delta_{\{X_j\}} < \Delta_{\{X_j,C\}}, \\ \{C\} & \text{if } X_j \to X_i \notin \mathcal{F} \text{ and } C \to X_i \in \mathcal{G}_0', \text{ and} \\ \emptyset & \text{if } X_j \to X_i \notin \mathcal{F} \text{ and } C \to X_i \notin \mathcal{G}_0'. \end{cases}$$

Arguing that the choice of parent set is well-defined and that the resulting Bayesian network must be acyclic is easy (a cycle in the generated network implies a cycle in $\mathcal{F}$ by construction). This time, it is not true that any valid Bayesian network fitting the constraints in the problem corresponds to some directed spanning forest; in particular, it might be the case that for some network where X_i has parent set $\emptyset$, no corresponding directed forest exists since $C \to X_i \in \mathcal{G}_0'$.

However, one can argue that the highest scoring Bayesian network does correspond to some directed spanning forest. To see why, note that in the previous scenario, $C \to X_i \in \mathcal{G}_0'$ implies that $\Delta_{\{C\}} > 0$ for X_i, so changing the parent set of X_i to $\{C\}$ yields an improvement in the total score of the network without generating cycles. Thus, any network which has no corresponding directed spanning forest cannot be optimal.

Finally, it suffices to note that the weight of any directed spanning forest is equal to the score corresponding Bayesian network minus the score of $\mathcal{G}_0'$, so an maximum weight directed spanning forest implies a maximal scoring Bayesian network.

Common errors:

(a) Constructions similar to Method 3, but which used $\mathcal{G}_0$ as the baseline graph and attempted to account for $C \rightarrow X_i$ edges as a postprocessing step. This fails for a simple network $X_1 \leftarrow C \rightarrow X_2$ in which all three variables are strongly correlated (but X_1 and X_2 are conditionally independent given C). The score for adding an $X_1 \rightarrow X_2$ edge (with or without C as another parent of X_2) is certainly higher than that of not including $X_1 \rightarrow X_2$ in the baseline graph $\mathcal{G}_0$ (where X_1 and X_2 are independent of C). Thus, the maximum weighted directed spanning forest algorithm will be forced to select an $X_1 \rightarrow X_2$ edge even though such an edge does not exist in the true network.

(b) Constructions similar to Method 2, but which failed to include the *forced* edges $X_i \rightarrow CX_i$, resulting in possible cycles in the resulting network. Without these forced edges, a maximum weight directed spanning forest might contain both $CX_j \rightarrow X_i$ and $CX_i \rightarrow X_j$, causing the resulting network to have an $X_i \leftrightarrow X_j$ cycle.

(c) **[4 points]** Constructions similar to Method 1, but which omitted the C variable and instead set edge weights $w_{X_j \rightarrow X_i} = \max(\Delta_{\{X_j, C\}}, \Delta_{\{X_j\}}, \Delta_{\{C\}})$. A minor problem here is that for a network with n feature variables, we know that any maximum weight directed spanning tree contains at most $n-1$ edges. Hence, we can make decisions for the parent set of at most $n-1$ feature variables. In the degenerate case of a network with one feature variable X_1, the returned network will always have C disconnected from X_1.

(d) **[6 points]** Constructions which omit the possibility of letting X_i have the parent set $\{C\}$.

Chapter 19

Exercise 19.3 — EM for CPD-trees [6 points]

Consider the problem of applying EM to parameter estimation for a variable X whose local probabilistic model is a CPD-tree. We assume that the network structure $\mathcal{G}$ includes the structure of the CPD-trees in it, so that we have a structure $\mathcal{T}$ for X. We are given a data set $\mathcal{D}$ with some missing values, and we want to run EM to estimate the parameters of $\mathcal{T}$. Explain how we can adapt the EM algorithm in order to accomplish this task.

Answer: Consider the leaves of tree $\mathcal{T}$. Each contains a distribution over the values of X corresponding to a specific set of values $\boldsymbol{q}$ for $\boldsymbol{Q}$, where $\boldsymbol{Q}$ is a subset of the parent nodes $\boldsymbol{U}$. What does this subset of parent nodes represent? It represents the relevant parents in a given situation, specifically whenever $\boldsymbol{Q} = \boldsymbol{q}$. The other parents become irrelevant in this case, as their values do not change the resulting distribution over values for X. In other words, when $\boldsymbol{Q} = \boldsymbol{q}$, it is as if X only has $\boldsymbol{Q}$ as parents. More formally, whenever $\boldsymbol{q} \subseteq \boldsymbol{u}$, $P(X \mid \boldsymbol{U} = \boldsymbol{u}) = P(X \mid \boldsymbol{Q} = \boldsymbol{q})$. Notice that for each leaf $\boldsymbol{Q}$ corresponds to the variables encountered on the path from the root of the tree to that leaf.

For simplicity let us assume that all variables are binary. Consider a leaf in the tree. We determine its relevant subset of parents and the values assigned to this subset ($\boldsymbol{Q}$ and $\boldsymbol{q}$ from the above). Then the parameter that represent that leaf looks like $\theta_{X|\boldsymbol{q}}$. We will get a similar parameter for every other leaf in the tree. If we had complete data, the parameters would be estimated using the sufficient statistics:

$$\theta_{X|\boldsymbol{q}} = \frac{M_{x,\boldsymbol{q}}}{M_{\boldsymbol{q}}} = \frac{\sum_{j=1}^{M} \mathbf{1}(X[j] = x, \boldsymbol{Q}[j] = \boldsymbol{q})}{\sum_{j=1}^{M} \mathbf{1}(\boldsymbol{Q}[j] = \boldsymbol{q})}$$

Similarly, each iteration then updates the parameters as using the expected sufficient statistics:

$$\theta_{X|\boldsymbol{q}} = \frac{\tilde{M}_{x,\boldsymbol{q}}}{\tilde{M}_{\boldsymbol{q}}} = \frac{\sum_{j=1}^{M} P(x, \boldsymbol{q} \mid \boldsymbol{d}[j], \boldsymbol{\theta}_{\text{old}})}{\sum_{j=1}^{M} P(\boldsymbol{q} \mid \boldsymbol{d}[j], \boldsymbol{\theta}_{\text{old}})}$$

Exercise 19.4 — EM for noisy-OR [9 points] Consider the problem of applying EM to parameter estimation for a variable X whose local probabilistic model is a noisy-or. Assume that X has parents $Y_1, \ldots, Y_k$, so that our task for X is to estimate the noise parameters $\lambda_0, \ldots, \lambda_k$. Explain how we can use the EM algorithm to accomplish this task. (Hint: Utilize the structural decomposition of the noisy-or node, as in figure (4.12) of the reader.)

Answer:

Suppose we decompose the noisy-or node into table CPT nodes as done in figure (4.12) of the reader (see Figure 7 above). Since X's CPT becomes a deterministic OR, we leave its CPT fixed in the EM algorithm. We treat the Y_i' as unobserved variables. The only parameters we need to estimate for this expanded noisy-or are the noise parameters λ_i contained in the CPTs for Y_i' and λ_0 contained in the CPT for Y_0'. We have $\lambda_i = \theta_{Y_i'=1|Y_i=1} = P(Y_i' = 1 \mid Y_i = 1)$ and $\lambda_0 = \theta_{Y_0'=1} = P(Y_0' = 1)$. The parameters $\theta_{Y_i'=0|Y_i=0}$ and $\theta_{Y_i'=1|Y_i=0}$ are fixed during EM to 1 and 0, respectively.

So when we use EM we update these parameters as follows:

$$\lambda_i = \frac{\sum_{j=1}^{M} P(Y_i' = 1, Y_i = 1 \mid \boldsymbol{d}[j], \boldsymbol{\theta}_{\text{old}})}{\sum_{j=1}^{M} P(Y_i = 1 \mid \boldsymbol{d}[j], \boldsymbol{\theta}_{\text{old}})}$$

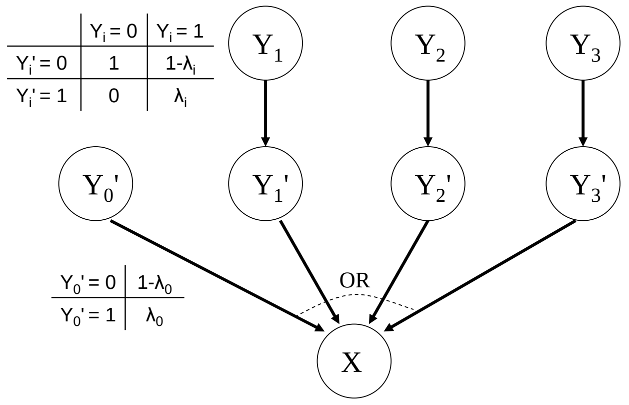


Figure 7: Noisy Or Decomposition

$$\lambda_0 = \frac{\sum_{j=1}^{M} P(Y_0' = 1 \mid \boldsymbol{d}[j], \boldsymbol{\theta}_{\text{old}})}{M}$$

Exercise 19.15 — Message passing with EM [15 points]

Suppose that we have an incomplete data set $\mathcal{D}$, and network structure $\mathcal{G}$ and matching parameters. Moreover, suppose that we are interested in learning the parameters of a single CPD $P(X_i \mid \boldsymbol{U}_i)$. That is, we assume that the parameters we were given for all other families are frozen and do not change during the learning. This scenario can arise for several reasons: we might have good prior knowledge about these parameters; or we might be using an incremental approach (for example, see the section in Box 19.C on accelerating convergence).

We now consider how this scenario can change the computational cost of the EM algorithm.

1. **[10 points]** Assume we have a clique tree for the network $\mathcal{G}$ and that the CPD $P(X_i \mid \boldsymbol{U}_i)$ was assigned to clique $\boldsymbol{C}_j$. Analyze which messages change after we update the parameters for $P(X_i \mid \boldsymbol{U}_i)$. Use this analysis to show how, after an initial pre-computation step, we can perform a single iteration of EM over this single family with a computational cost that depends only on the size of $\boldsymbol{C}_j$ and not the size of the rest of the cluster tree.

   Answer: As an initial pre-computation step, we will fully calibrate M trees, one for each instantiation of the observations $\boldsymbol{o}[m]$.

   Now suppose we have updated the parameters for $P(X_i \mid \boldsymbol{U}_i)$, and now will consider what computation is necessary to perform a full EM step. The E-step involves calibration of the clique trees. During calibration, only the outgoing messages and other messages pointing away from $\boldsymbol{C}_j$ will change. To complete the E-step, we collect the sufficient statistics:

$$\bar{M}[x_i, \boldsymbol{u}_i] = \sum_{m=1}^{M} P(x_i, \boldsymbol{u}_i \mid \boldsymbol{o}[m], \theta)$$

Then to perform the M-step, which consists of updating $P(X_i \mid \boldsymbol{U}_i)$, we calculate:

$$P(X_i \mid \boldsymbol{U}_i) = \frac{\bar{M}[x_i, \boldsymbol{u}_i]}{\bar{M}[\boldsymbol{u}_i]}$$

Now we work backward to see what computation was necessary to perform these calculations. To calculate the probabilities in the E-step, we use part of the result of calibration:

$$P(x_i, \bar{u}_i \mid \bar{o}[m], \theta) = \sum_{\boldsymbol{C}_j - \{X_i, \bar{U}_i\}} \beta'_j(\boldsymbol{C}_j \mid \bar{o}[m], \theta)$$

where β'_j is the updated belief of $\boldsymbol{C}_j$.

$$\beta'_j(\boldsymbol{C}_j \mid \bar{o}[m], \theta) = \psi'_j \prod_{i \in \mathcal{N}(|)} \delta_{i \to j}$$

where ψ'_j is the updated initial belief of $\boldsymbol{C}_j$.

$$\psi'_j = \prod_{k:\alpha(k)=j} \phi_k$$

where α is the function that assigns factors to cliques; we know that the updated $P(X_i \mid \boldsymbol{U}_i)$ is one of these factors for $\boldsymbol{C}_j$.

Note that the only messages that are used $(\delta_{i \to j})$ were unchanged during calibration (since incoming messages to $\boldsymbol{C}_j$ remain unchanged. Therefore, we actually don't need to perform any calibration computation at all. Furthermore, the computation described in each of the equations above only involves the scope of $\boldsymbol{C}_j$; therefore the computational cost depends only on the size of $\boldsymbol{C}_j$ (and also on M, since we do this for each of the M trees), and not on the size of the rest of the tree.

2. **[5 points]** Would this conclusion change if we update the parameters of several families that are all assigned to the same cluster in the cluster tree? Explain.

   Answer: The conclusion does not change. All of the calculations above remain the same; the only difference is that multiple factors used to calculate the initial belief ψ'_j will have been updated when this calculation is performed.

Exercise 19.16 — Incremental EM Suppose we are given a Bayes net $\mathcal{G}$, and a partition of the variables into k subsets, $\mathcal{X} = \boldsymbol{X}_1 \cup \boldsymbol{X}_2 \cup ... \cup \boldsymbol{X}_k$. For each M-step, we choose, according to some strategy, one subset $\boldsymbol{X}_i$, and then instead of optimizing all the parameters of the network at once, we optimize only the parameters $\boldsymbol{\theta}_{X|\mathbf{Pa}_X}$ for $X \in \boldsymbol{X}_i$.

Is this modified version of EM still guaranteed to converge? In other words, does

$$\ell(\boldsymbol{\theta}^t : \mathcal{D}) \leq \ell(\boldsymbol{\theta}^{t+1} : \mathcal{D})$$

still hold? If so, prove the result. If not, explain why not.

Answer: We first use theorem 19.5 to conclude that the improvement in the log-likelihood is lower-bounded by the expected log-likelihood, i.e.,

$$\ell(\boldsymbol{\theta}^{t+1} : \mathcal{D}) - \ell(\boldsymbol{\theta}^t : \mathcal{D}) \geq \boldsymbol{E}_{P(\mathcal{H}|\mathcal{D},\boldsymbol{\theta}^t)}\left[\ell(\boldsymbol{\theta}^{t+1} : \mathcal{D}^+)\right] - \boldsymbol{E}_{P(\mathcal{H}|\mathcal{D},\boldsymbol{\theta}^t)}\left[\ell(\boldsymbol{\theta}^t : \mathcal{D}^+)\right]$$

The above holds for any choice of parameters, $\boldsymbol{\theta}^{t+1}$ and $\boldsymbol{\theta}^{t}$. Now, say in the current iteration, we are maximizing over the parameters in set i. We compute $\boldsymbol{\theta}^{t+1}$ by selecting the parameter that maximizes

$$\boldsymbol{E}_{P(\mathcal{H}|\mathcal{D},\boldsymbol{\theta}^t)}\left[\ell(\boldsymbol{\theta} : \mathcal{D}^+)\right]$$

subject to the constraint that all the parameters not in set i remain the same. Hence, within the constrained space because of the maximization operation

$$\boldsymbol{E}_{P(\mathcal{H}|\mathcal{D},\boldsymbol{\theta}^t)}\left[\ell(\boldsymbol{\theta}^{t+1} : \mathcal{D}^+)\right] - \boldsymbol{E}_{P(\mathcal{H}|\mathcal{D},\boldsymbol{\theta}^t)}\left[\ell(\boldsymbol{\theta}^{t} : \mathcal{D}^+)\right] \geq 0$$

Thus, we conclude that the log-likelihood increases at every iteration of EM.

Note that this is not sufficient to say that EM converges. For EM to converge, an upper-bound to the log-likelihood function must exist. In the case of discrete data, it is at most zero because maximal likelihood assigns 1 to each instance. However, in the case of continuous distributions, this is untrue because the density function may assign arbitrarily large support to a point. If the likelihood function is upper-bounded, then EM will converge to a local maxima, minima or a saddle point.

Exercise 19.27 — Structural EM

Recall that structural EM (SEM) uses the existing candidate Bayesian network to complete the data, and then uses the completed data to evaluate each candidate successor network. One possible scheme to discovering hidden variables in a Bayesian network is to introduce a disconnected hidden variable H into our current candidate, then use structural EM to "hook it up" to the remaining network. Specifically, assume that our current candidate BN structure in the structure search contains some set of known variables $X_1, \ldots, X_n$ (ones that are observed at least sometimes) and a hidden variable H (one which is never observed). Our current structure G has the X_i's connected to each other in some way, but the variable H is not connected to any other variable. As usual, SEM completes the data (including for the hidden variable) using the network G, and uses the completed data to evaluate the potential successor BNs. It then greedily chooses the successor network G' that most improves the score (we are using the BIC (equivalently, MDL) score, so there may be no network improving the score). Show that, in the chosen successor G', H will still necessarily be disconnected from the X_i's.

Answer: Recall that in SEM, we perform structure search by completing the data for our original model $\mathcal{G}$, which we then use to calculate parameters in successor networks $\mathcal{G}'$. In our current candidate network, we know that H is disjoint from the rest of the network and **never** observed. Because H is disjoint, we are asserting that H is independent of all the other variables in the network no matter the observation.

In SEM, we maintain a distribution Q over completed datasets implicitly as a pair $\langle \mathcal{G}^Q, \boldsymbol{\theta}_{\mathcal{G}^Q} \rangle$. An observation $\boldsymbol{o}[i]$'s contribution to the expected sufficient statistic for H is $P(H|\boldsymbol{o}[i], \mathcal{G}^Q, \boldsymbol{\theta}_{\mathcal{G}^Q})$. Because H is independent of every other variable in $\mathcal{G}$, though, this is simply the prior probablity of H in $\boldsymbol{\theta}_{\mathcal{G}^Q}$. Thus, the expected sufficient statistic for H over the entire dataset is $\bar{M}[h] = M \cdot P(h|\mathcal{G}^Q, \boldsymbol{\theta}_{\mathcal{G}^Q})$. If we were running standard parametric EM, H's parameters would never change.

In the following, we will use the shorthand that $P(\boldsymbol{x} \mid \boldsymbol{y}, \mathcal{G}^Q, \boldsymbol{\theta}_{\mathcal{G}^Q}) = P_{\mathcal{G}}(\boldsymbol{x} \mid \boldsymbol{y})$ when it is convenient (to avoid clutter).

Now, let us consider what happens in SEM. In SEM, we use the current completion of the data relative to $\langle \mathcal{G}, \boldsymbol{\theta}_{\mathcal{G}^Q} \rangle$ to compute sufficient statistics. Let $\boldsymbol{x}$ be an instantiation to some subset of the nodes $X_1, \ldots X_n$. Then

$$\bar{M}[\boldsymbol{x}, h] = \sum_{i=1}^{M} P(\boldsymbol{x}, h \mid \boldsymbol{o}[i], \mathcal{G}^Q, \boldsymbol{\theta}_{\mathcal{G}^Q})$$

H is disconnected from the rest of the nodes, so

$$\bar{M}[\boldsymbol{x}, h] = \sum_{i=1}^{M} P(\boldsymbol{x} \mid \boldsymbol{o}[i], \mathcal{G}^Q, \boldsymbol{\theta}_{\mathcal{G}^Q}) P(h \mid \boldsymbol{o}[i], \mathcal{G}^Q, \boldsymbol{\theta}_{\mathcal{G}^Q})$$

Since H is also never observed, it is never a part of $\boldsymbol{o}[i]$. So H is independent from any $\boldsymbol{o}[i]$ and

$$\bar{M}[\boldsymbol{x}, h] = \sum_{i=1}^{M} P_{\mathcal{G}}(\boldsymbol{x} \mid \boldsymbol{o}[i]) P_{\mathcal{G}}(h) = P_{\mathcal{G}}(h) \sum_{i=1}^{M} P_{\mathcal{G}}(\boldsymbol{x} \mid \boldsymbol{o}[i]) = P_{\mathcal{G}}(h) \bar{M}[\boldsymbol{x}].$$

Now, consider what happens if SEM tries to add an edge from H to some node X_i. $\mathbf{Pa}_{X_i}$ is the set of parents of X_i in the current network $\mathcal{G}$. The only family that changes in the new network $\mathcal{G}'$ is the family of X_i, so the difference between the log-likelihood scores of $\mathcal{G}$ and $\mathcal{G}'$ is:

$$\begin{aligned}
&\mathrm{score}_L(\mathcal{G}' \ : \ \mathcal{D}) - \mathrm{score}_L(\mathcal{G} \ : \ \mathcal{D}) \\
&= \ \mathrm{FamScore}_L(X_i \mid \mathbf{Pa}_{X_i} \cup H \ : \ \mathcal{D}) - \mathrm{FamScore}_L(X_i \mid \mathbf{Pa}_{X_i} \ : \ \mathcal{D}) \\
&= \sum_{\boldsymbol{u} \in Val(\mathbf{Pa}_{X_i}), h} \sum_{x_i} \bar{M}[x_i, \boldsymbol{u}, h] \log \frac{\bar{M}[x_i, \boldsymbol{u}, h]}{M[\boldsymbol{u}, h]} - \sum_{\boldsymbol{u} \in Val(\mathbf{Pa}_{X_i})} \sum_{x_i} \bar{M}[x_i, \boldsymbol{u}] \log \frac{\bar{M}[x_i, \boldsymbol{u}]}{\bar{M}[\boldsymbol{u}]} \\
&= \sum_{\boldsymbol{u}} \sum_{x_i} \sum_{h} P_{\mathcal{G}}(h) \bar{M}[x_i, \boldsymbol{u}] \log \frac{\bar{M}[x_i, \boldsymbol{u}]}{\bar{M}[\boldsymbol{u}]} - \sum_{\boldsymbol{u}} \sum_{x_i} \bar{M}[x_i, \boldsymbol{u}] \log \frac{\bar{M}[x_i, \boldsymbol{u}]}{\bar{M}[\boldsymbol{u}]} = 0.
\end{aligned}$$

Similarly, if SEM tries to add an edge from X_i to H the only family that changes is the family of H. We have

$$\begin{aligned}
&\mathrm{score}_L(\mathcal{G}' \ : \ \mathcal{D}) - \mathrm{score}_L(\mathcal{G} \ : \ \mathcal{D}) \\
&= \sum_{h, x_i} \bar{M}[x_i, h] \log \frac{\bar{M}[x_i, h]}{\bar{M}[x_i]} - \sum_{h} \bar{M}[h] \log \frac{\bar{M}[h]}{M} \\
&= M \sum_{h, x_i} P_{\mathcal{G}}(h) \log \frac{P_{\mathcal{G}}(h) \bar{M}[x_i]}{\bar{M}[x_i]} - M \sum_{h} P_{\mathcal{G}}(h) \log P_{\mathcal{G}}(h) = 0.
\end{aligned}$$

So, adding any arc H — X would not increase the likelihood component of the BIC score in SEM. The BIC scoring function penalizes extra parameters, so the sucessor network with an arc between H and X would actually score less than the current candidate network. Thus SEM would never choose the sucessor network with an edge to or from H.

Exercise 19.28 — EM with Uniform Initialization [8 points]

Consider the Naive Bayes model with class variable C and discrete evidence variables $X_1, \ldots, X_n$. The CPDs for the model are parameterized by $P(C = c) = \theta_c$ and $P(X_i = x \mid C = c) = \theta_{x_i|c}$ for $i = 1, \ldots, n$, and for all assignments $x_i \in Val(X_i)$ and classes $c \in Val(C)$.

Now given a data set $\mathcal{D} = \{\boldsymbol{x}[1], \ldots, \boldsymbol{x}[M]\}$, where each $\boldsymbol{x}[m]$ is a complete assignment to the evidence variables, $X_1, \ldots, X_n$, we can use EM to learn the parameters of our model. Note that the class variable, C, is never observed.

Show that if we initialize the parameters uniformly,

$$\theta_c^0 = \frac{1}{|Val(C)|} \qquad \text{and} \qquad \theta_{x_i|c}^0 = \frac{1}{|Val(X_i)|},$$

for all x_i, c, then the EM algorithm converges in one iteration, and give a closed form expression for the parameter values at this convergence point.

Answer: Computing the first iteration of EM (with initialization given above) we get:

$$\begin{aligned}
\theta^1_{C=c} &= \frac{\bar{M}_{\theta^t}[c]}{M} = \frac{1}{M}\sum_m P(c \mid \boldsymbol{x}[m], \boldsymbol{\theta}^0) \\
&= \frac{1}{M}\sum_m \frac{P(\boldsymbol{x}[m] \mid c, \boldsymbol{\theta}^0)P(c \mid \boldsymbol{\theta}^0)}{P(\boldsymbol{x}[m] \mid \boldsymbol{\theta}^0)} \\
&= \frac{1}{M}\sum_m \frac{P(\boldsymbol{x}[m] \mid c, \boldsymbol{\theta}^0)P(c \mid \boldsymbol{\theta}^0)}{\sum_{c' \in \mathit{Val}(C)} P(\boldsymbol{x}[m] \mid c', \boldsymbol{\theta}^0)P(c' \mid \boldsymbol{\theta}^0)} \\
&= \frac{1}{M}\sum_m \frac{P(c \mid \boldsymbol{\theta}^0)\prod_{i=1}^n P(\boldsymbol{x}_i[m] \mid c, \boldsymbol{\theta}^0)}{\sum_{c' \in \mathit{Val}(C)} P(c' \mid \boldsymbol{\theta}^0)\prod_{i=1}^n P(\boldsymbol{x}_i[m] \mid c', \boldsymbol{\theta}^0)} \\
&= \frac{1}{M}\sum_m \frac{\frac{1}{|\mathit{Val}(C)|}\prod_{i=1}^n \frac{1}{|\mathit{Val}(X_i)|}}{\sum_{c' \in \mathit{Val}(C)} \frac{1}{|\mathit{Val}(C)|}\prod_{i=1}^n \frac{1}{|\mathit{Val}(X_i)|}} \\
&= \frac{1}{|\mathit{Val}(C)|}
\end{aligned}$$

and

$$\begin{aligned}
\theta^1_{X_i=x \mid C=c} &= \frac{\bar{M}_{\theta^0}[x,c]}{\bar{M}_{\theta^0}[c]} = \frac{\sum_m P(c \mid \boldsymbol{x}[m], \boldsymbol{\theta}^0)\boldsymbol{1}\{x_i[m]=x\}}{\sum_m P(c \mid \boldsymbol{x}[m], \boldsymbol{\theta}^0)} \\
&= \frac{\sum_m \frac{1}{|\mathit{Val}(C)|}\boldsymbol{1}\{x_i[m]=x\}}{\frac{M}{|\mathit{Val}(C)|}} \qquad \text{from above} \\
&= \frac{1}{M}\sum_m \boldsymbol{1}\{x_i[m]=x\} = \frac{M[x]}{M}
\end{aligned}$$

where $M[x]$ is the number of times $X_i = x$ appears in the data $\mathcal{D}$.

We will now show by induction that for all $t \geq 1$, $\theta^t_{C=c} = \frac{1}{|\mathit{Val}(C)|}$ and $\theta^t_{X_i=x \mid C=c} = \frac{M[x]}{M}$. The second parameter implies that $P(\boldsymbol{x}_i[m] \mid c, \boldsymbol{\theta}^t) = P(\boldsymbol{x}_i[m] \mid c', \boldsymbol{\theta}^t)$ for all $c, c' \in \mathit{Val}(C)$. We have just shown this for $t = 1$, so assuming true for some t, we have for $t + 1$,

$$\begin{aligned}
\theta^{t+1}_{C=c} &= \frac{\bar{M}_{\theta^t}[c]}{M} = \frac{1}{M}\sum_m P(c \mid \boldsymbol{x}[m], \boldsymbol{\theta}^t) \\
&= \frac{1}{M}\sum_m \frac{P(\boldsymbol{x}[m] \mid c, \boldsymbol{\theta}^t)P(c \mid \boldsymbol{\theta}^t)}{P(\boldsymbol{x}[m] \mid \boldsymbol{\theta}^t)} \\
&= \frac{1}{M}\sum_m \frac{P(\boldsymbol{x}[m] \mid c, \boldsymbol{\theta}^t)P(c \mid \boldsymbol{\theta}^t)}{\sum_{c' \in Val(C)} P(\boldsymbol{x}[m] \mid c', \boldsymbol{\theta}^t)P(c' \mid \boldsymbol{\theta}^t)} \\
&= \frac{1}{M}\sum_m \frac{P(c \mid \boldsymbol{\theta}^t)\prod_{i=1}^n P(\boldsymbol{x}_i[m] \mid c, \boldsymbol{\theta}^t)}{\sum_{c' \in Val(C)} P(c' \mid \boldsymbol{\theta}^t)\prod_{i=1}^n P(\boldsymbol{x}_i[m] \mid c', \boldsymbol{\theta}^t)} \\
&= \frac{1}{M}\sum_m \frac{\frac{1}{|Val(C)|}\prod_{i=1}^n P(\boldsymbol{x}_i[m] \mid c, \boldsymbol{\theta}^t)}{\prod_{i=1}^n P(\boldsymbol{x}_i[m] \mid c, \boldsymbol{\theta}^t) \cdot \sum_{c' \in Val(C)} \frac{1}{|Val(C)|}} \\
&= \frac{1}{M}\sum_m \frac{1}{|Val(C)|} = \frac{1}{|Val(C)|}
\end{aligned}$$

and

$$\begin{aligned}
\theta^{t+1}_{X_i=x|C=c} &= \frac{\bar{M}_{\theta^t}[x,c]}{\bar{M}_{\theta^t}[c]} = \frac{\sum_m P(c \mid \boldsymbol{x}[m], \boldsymbol{\theta}^t)\boldsymbol{1}\{x_i[m] = x\}}{\sum_m P(c \mid \boldsymbol{x}[m], \boldsymbol{\theta}^t)} \\
&= \frac{\sum_m \frac{1}{|Val(C)|}\boldsymbol{1}\{x_i[m] = x\}}{\frac{M}{|Val(C)|}} \qquad \text{from above} \\
&= \frac{1}{M}\sum_m \boldsymbol{1}\{x_i[m] = x\} = \frac{M[x]}{M}
\end{aligned}$$

Thus we have shown that initializing the parameters uniformly the EM algorithm converges in one iteration to

$$\theta^t_{C=c} = \frac{1}{|Val(C)|} \qquad \text{and} \qquad \theta^t_{X_i=x|C=c} = \frac{M[x]}{M}.$$

Chapter 20

Exercise 20.11 — Multi-conditional training [12 points] In this problem, we will consider the problem of learning parameters for a Markov network using a specific objective function. In particular assume that we have two sets of variables $\boldsymbol{Y}$ and $\boldsymbol{X}$, and a dataset $\mathcal{D} = \{\langle \boldsymbol{x}[1], \boldsymbol{y}[1]\rangle, \ldots, \langle \boldsymbol{x}[m], \boldsymbol{y}[m]\rangle\}$ We will estimate the model parameters $\boldsymbol{\theta} = [\theta_1 \ldots \theta_n]$ by maximizing the following objective function:

$$f(\boldsymbol{\theta} : \mathcal{D}) = (1-\alpha)\ell_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{\theta} : \mathcal{D}) + \alpha \ell_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{\theta} : \mathcal{D})$$

where $\ell_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{\theta} : \mathcal{D})$ means the conditional log-likelihood of the dataset $\mathcal{D}$ using the distribution $P(\boldsymbol{X} \mid \boldsymbol{Y})$ defined by the Markov network with parameters $\boldsymbol{\theta}$ (similarly for $\ell_{\boldsymbol{Y}|\boldsymbol{X}}$). Thus, our objective is a mixture of two conditional log-likelihoods ($0 < \alpha < 1$). As usual, we consider a log-linear parameterization of a Markov network, using a set of n features $\phi_i[\boldsymbol{X}_i, \boldsymbol{Y}_i]$ where $\boldsymbol{X}_i$ and $\boldsymbol{Y}_i$ are some (possibly empty) subsets of the variables $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively.

1. **[4 points]** Write down the full objective function $f(\boldsymbol{\theta} : \mathcal{D})$ in terms of the features ϕ_i and weights θ_i

   Answer: The objective function can be written as:

$$\begin{aligned} f(\boldsymbol{\theta} : \mathcal{D}) \quad = \quad & \sum_{m=1}^{M} \left(\sum_i \theta_i \phi_i[\boldsymbol{X}_i[m], \boldsymbol{Y}_i[m]] \right) - (1-\alpha) \log \sum_{\boldsymbol{Y}} \exp \sum_i \theta_i \phi_i[\boldsymbol{X}_i[m], \boldsymbol{Y}_i] - \\ & \alpha \log \sum_{\boldsymbol{X}} \exp \sum_i \theta_i \phi_i[\boldsymbol{X}, \boldsymbol{Y}_i[m]] \end{aligned}$$

2. **[8 points]** Derive $\frac{\partial}{\partial \theta_i} f(\theta : \mathcal{D})$: the derivative of the objective with respect to a weight θ_i. Write your final answer in terms of feature expectations $\boldsymbol{E}_Q[\phi_i]$, where Q is either: the empirical distribution of our dataset $\hat{P}$; or a conditional distribution of the form $P_{\boldsymbol{\theta}}(\boldsymbol{W} \mid \boldsymbol{Z} = \boldsymbol{z})$ (for some sets of variables $\boldsymbol{W}, \boldsymbol{Z}$, and assignment $\boldsymbol{z}$.)

   Answer: The derivative has the form:

$$\frac{\partial}{\partial \theta_i} f(\theta : \mathcal{D}) = \sum_{m=1}^{M} \left(\boldsymbol{E}_{\hat{P}}[\phi_i] - (1-\alpha)\boldsymbol{E}_{P_{\boldsymbol{\theta}}(\boldsymbol{Y}_i|\boldsymbol{X}=\boldsymbol{x}[m])}[\phi_i] - \alpha \boldsymbol{E}_{P_{\boldsymbol{\theta}}(\boldsymbol{X}_i|\boldsymbol{Y}=\boldsymbol{y}[m])}[\phi_i] \right)$$

Chapter 21

Exercise 21.1a — Causal query simplification Let $\mathcal{C}$ be a causal model over the graph structure $\mathcal{G}$. Then:

$$P(\boldsymbol{Y} \mid do(\boldsymbol{Z} := \boldsymbol{z}), do(\boldsymbol{X} := \boldsymbol{x}), \boldsymbol{W} = \boldsymbol{w}) = P(\boldsymbol{Y} \mid do(\boldsymbol{Z} := \boldsymbol{z}), \boldsymbol{X} = \boldsymbol{x}, \boldsymbol{W} = \boldsymbol{w})$$

if $\boldsymbol{Y}$ is d-separated from $\boldsymbol{X}$ given $\boldsymbol{Z}, \boldsymbol{W}$ in the graph $\mathcal{G}_{\overline{\boldsymbol{Z}}\underline{\boldsymbol{X}}}$.

Answer: We will prove the contrapositive. Suppose $P(\boldsymbol{Y} \mid do(\boldsymbol{Z} := \boldsymbol{z}), do(\boldsymbol{X} := \boldsymbol{x}), \boldsymbol{W} = \boldsymbol{w}) \neq P(\boldsymbol{Y} \mid do(\boldsymbol{Z} := \boldsymbol{z}), \boldsymbol{X} = \boldsymbol{x}, \boldsymbol{W} = \boldsymbol{w})$. Then there is an active path in $\mathcal{G}_{\overline{\boldsymbol{Z}}}$ from some observed U to some $Y \in \boldsymbol{Y}$ given $\boldsymbol{X}\boldsymbol{Z}\boldsymbol{W} - U$ that contains an edge between some $X \in \boldsymbol{X}$ and its parent. Choose the last such segment along the path to Y and the active path from this $\hat{X}$ to Y. Note that this path cannot contain a segment between an $X' \in \boldsymbol{X}$ and its child; otherwise the path would be inactive since X' is observed and this wouldn't be a v-structure (since the edge is from X' to its child). So the path from $\hat{X}$ to Y would also be active in $\mathcal{G}_{\overline{\boldsymbol{Z}}\underline{\boldsymbol{X}}}$ and so not $\textit{d-sep}_{\mathcal{G}_{\overline{\boldsymbol{Z}}\underline{\boldsymbol{X}}}}(\boldsymbol{Y}; \boldsymbol{X} \mid \boldsymbol{Z}\boldsymbol{W})$.

Exercise 21.3 — Intervention queries

For probabilistic queries, we have that

$$\min_x P(y \mid x) \leq P(y) \leq \max_x P(y \mid x).$$

Show that the same property does not hold for intervention queries. Specifically, provide an example where it is not the case that:

$$\min_x P(y \mid do(x)) \leq P(y) \leq \max_x P(y \mid do(x)).$$

Answer: Consider the case of an XOR network over binary variables, where $P(Z = 1) = 0.5$, $X = Z$, and $Y = X \oplus Z$. In this case, $P(Y = 1) = 0$, but if we intervene at X, we get $P(Y \mid do(X = 1)) = P(Y \mid do(X = 0)) = 0.5$.

Exercise 21.11 — Causal independence

1. [**6 points**] As for probabilistic independence, we can define a notion of causal independence: $(\boldsymbol{X} \perp_C \boldsymbol{Y} \mid \boldsymbol{Z})$ if, for any values $\boldsymbol{x}, \boldsymbol{x}' \in Val(\boldsymbol{X})$, we have that $P(\boldsymbol{Y} \mid do(\boldsymbol{Z}), do(\boldsymbol{x})) = P(\boldsymbol{Y} \mid do(\boldsymbol{Z}), do(\boldsymbol{x}'))$. (Note that, unlike probabilistic independence — $(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$ — causal independence is not symmetric over $\boldsymbol{X}, \boldsymbol{Y}$.) Is causal independence equivalent to the statement: "For any value $\boldsymbol{x} \in Val(\boldsymbol{X})$, we have that $P(\boldsymbol{Y} \mid do(\boldsymbol{Z}), do(\boldsymbol{x})) = P(\boldsymbol{Y} \mid do(\boldsymbol{Z}))$." (Hint: Use exercise 21.3.)

   Answer: No, they are not the same. In the case above, we have that $P(Y = 1 \mid do(X)) = 0.5$ for all values of X, but still $P(Y = 1) \neq P(Y = 1 \mid do(X))$.

2. [**7 points**] Prove that $(\boldsymbol{X} \perp_C \boldsymbol{Y} \mid \boldsymbol{Z}, \boldsymbol{W})$ and $(\boldsymbol{W} \perp_C \boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{Z})$ implies that $(\boldsymbol{X}, \boldsymbol{W} \perp_C \boldsymbol{Y} \mid \boldsymbol{Z})$. Intuitively, this property states that, if changing $\boldsymbol{X}$ cannot affect $P(\boldsymbol{Y})$ when $\boldsymbol{W}$ is fixed, and changing $\boldsymbol{W}$ cannot affect $P(\boldsymbol{Y})$ when $\boldsymbol{X}$ is fixed, then changing $\boldsymbol{X}$ and $\boldsymbol{W}$ together cannot affect $P(\boldsymbol{Y})$.

   Answer: Here is a simplified version of the proof:

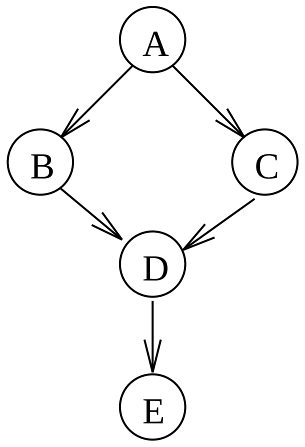
   For any $\boldsymbol{x}, \boldsymbol{x}' \in Val(()\boldsymbol{X})$ and $\boldsymbol{w}, \boldsymbol{w}' \in Val(()\boldsymbol{W})$ we have:

$$\begin{aligned} P(\boldsymbol{Y} \mid do(\boldsymbol{x}, \boldsymbol{w}), do(\boldsymbol{Z})) &= P(\boldsymbol{Y} \mid do(\boldsymbol{x}), do(\boldsymbol{w}), do(\boldsymbol{Z})) && (16)\\ &= P(\boldsymbol{Y} \mid do(\boldsymbol{x}'), do(\boldsymbol{w}), do(\boldsymbol{Z})) && (17)\\ &= P(\boldsymbol{Y} \mid do(\boldsymbol{x}'), do(\boldsymbol{w}'), do(\boldsymbol{Z})) && (18)\\ &= P(\boldsymbol{Y} \mid do(\boldsymbol{x}', \boldsymbol{w}'), do(\boldsymbol{Z})) && (19) \end{aligned}$$

Where line 2 is true from $(\boldsymbol{X} \perp_C \boldsymbol{Y} \mid \boldsymbol{Z}, \boldsymbol{W})$, line 3 is true from $(\boldsymbol{W} \perp_C \boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{Z})$. The result shows that we have $(\boldsymbol{X}, \boldsymbol{W} \perp_C \boldsymbol{Y} \mid \boldsymbol{Z})$.

Exercise 21.12 — Learning Causal Models [17 points]

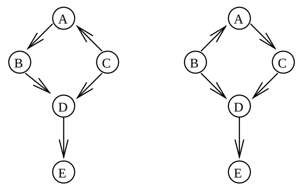
An important question in scientific discovery is using data to extract causal knowledge, i.e., the existence of a causal path between two variables. Note that a causal path exists from X to Y if in any graph that is an I-map of the distribution, there is a causal (i.e., directed) path from X to Y. In this problem, we will consider the interaction between this problem and hidden variables. Assume that our learning algorithm came up with the following network:



which we are willing to assume is a perfect map for the distribution over the variables A, B, C, D, E (i.e., it exactly captures the independencies and dependencies among them). Under this assumption, among which pairs of these variables does there necessarily exist a causal path ...

1. **[7 points]** ... if we assume there are no hidden variables?

   Answer: From the figures below we can see that the paths from A to B and from A to C are not necessarily causal.

   The only way we could break the causal chain from A to D and E would be by making A the bottom of a v-structure (we would have to have B and C as parents. If we do this, however, we lose the independence assumption that $I(B, C \mid A)$. We can refer to the arguments from class to see that the other paths are causal. Thus, we have the following causal paths: A to D, A to E, B to D, B to E, C to D, C to E, and D to E.

2. **[10 points]** ... if we allow the possibility of one or more hidden variables ?

   Answer: We know from part (a) that paths from A to B and A to C are not necessarily causal.

   Now, consider the paths from B and C to D. We know that observing D must activate a path between B and C, which means D must be the bottom of a v-structure which links an active path from B to D and an active path from C to D. Since any new variables we add are never observed, there can be no new v-structures between B and D and no new v-structures between C and D. (If there were such a v-structure, we would now have the property that $I(B, C \mid A, D)$ which does not hold in the original network.)

Since D must be the bottom of a v-structure, we know that the paths which link B and C to D must have their arrows pointing towards D. The only way left to break the causal path from B (or C) to D is to make a hidden variable a common parent of both B and D (or C and D). There are 2 ways we might do this while preserving the arcs from A to B and A to C: we can make H a child of A with 2 children: B (or C) and D, we can just replace the direct connection from B (or C) to D with a path from H to B (or C) and a path from H to D. In either case we lose the property that $I(A, D \mid B, C)$ which holds in our original network.

If we reverse the arc from B (or C) to A as in the figure for part (a), we can replace the link from B (or C) to D by a hidden variable H which is a common parent of B (or C) and D. This network satisfies the same independence assumptions. However, while we lose the direct causal edge from B (or C) to D, a new causal path through A is now present. Note that the only way we were able to do this was by reversing the arc between A and B (or C), so the only way this new causal path from B (or C) to D can be broken is by breaking the causal path from A to D which we consider below.

Thus, the only way of breaking causal paths still open to us is to alter the path from B to C through A. We know that observing A must block an active path between B and C. Clearly, the active path which observing A blocks goes through A. Thus there cannot be any v-structures along the path from B to C through A. If A were at the bottom of such a v-structure then we would activate the path from B to C by instantiating A which means we would lose the property that $I(B, C \mid A)$. If a hidden variable H were at the bottom we would break the active path through A, i.e., we would now have $I(B, C)$ which is not true. (We cannot correct for this new independence by adding a direct path between B and C through a hidden variable because we would then lose the property that $I(B, C \mid A)$.) These remarks hold true whether we are considering modifications to the initial network (with paths from A to B and C) or to either of the modifcations shown in the figures in part (a).

Thus, the paths that are guaranteed to be causal are from A to D, A to E, B to D, B to E, C to D, C to E, and D to E.

Chapter 22

New exercise (chapter 22) — Decomposable Utility Functions [8 points]

Recall that a utility function is a mapping from an outcome (assignment to variables) to a real number. Suppose we have M variables, $X_1 \dots X_M$, each of which has a domain of size $|Val(X_i)| = d$, and a utility function $U(X_1 \dots X_M)$ over these variables. Our goal in this problem is to find the "ideal" outcome (i.e., the one that gives us the highest utility). Concretely, we are searching for:

$$(x_1^* \dots x_M^*) = \arg \max_{x_1 \dots x_M} U(x_1 \dots x_M).$$

Assume that this outcome is unique (that is, there are no ties for first-place outcome).

Suppose that our U is decomposable as a sum of utilities for relatively small subsets of the variables:

$$U(X_1 \dots X_M) = \sum_{i=1}^{k} U(\boldsymbol{Y}_i),$$

where $\boldsymbol{Y}_i \subset \{X_1 \dots X_M\}$.

1. [**5 points**] Describe an algorithm that exactly determines $(x_1^* \dots x_M^*)$ and $U(x_1^* \dots x_M^*)$ and that exploits the structure of the utility function to make it computationally more efficient. You may use as a subroutine any algorithm that we described in class.

   Answer: Let $V(X) = \exp(U(X))$. Perform max-product with the set of factors defined by V. Then $U^* = U(x^*) = \log V(x^*)$, where x^* is the maximizing assignment found by max-product.

2. [**3 points**] Show that your algorithm requires a computational complexity of $O(Md^2)$ for the case of chain decomposability, where:

$$U(X_1 \dots X_M) = \sum_{i=1}^{M-1} U(X_i, X_{i+1}).$$

   Answer: The biggest clique you need is of size 2, so you have $O(d^2)$ operations per clique, giving a total run time of $O(Md^2)$.

Chapter 23

Exercise 23.14 — Value of Perfect Information [16 points]

1. **[10 points]** Let $\mathcal{I}$ be an influence diagram, D a decision variable in $\mathcal{I}$, and X a chance variable which is a non-descendant of D. Prove that $\text{VPI}_{\mathcal{I}}(D \mid X) \geq 0$. Moreover, show that equality is achieved if and only if the decision at D is unchanged for any value x of X and any assignment $\boldsymbol{w}$ to D's other parents $\boldsymbol{W}$. (Assume that ties are broken in some prespecified way when two actions have the same expected utility.)

   Answer: We prove this result for the case where $\mathcal{I}$ contains only a single decision variable. To show that $\text{VPI}_{\mathcal{I}}(D \mid X) \geq 0$, it suffices to show that $\text{MEU}[\mathcal{I}'] \geq \text{MEU}[\mathcal{I}]$ where $\mathcal{I}'$ is the influence diagram obtained by adding an edge from X to D in $\mathcal{I}$.

   Let σ^* be an optimal strategy corresponding to the maximal expected utility in $\mathcal{I}$, and let $\delta_D^{\sigma^*}(D \mid \mathbf{Pa}_D)$ be the probability distribution over choices of D given values of its parents specified by σ^*. Similarly, let γ^* be an optimal strategy for $\mathcal{I}'$, and let $\delta_D^{\gamma^*}(D \mid X, \mathbf{Pa}_D)$ be its corresponding decision rule. We know that

$$\begin{aligned}
\text{MEU}[\mathcal{I}] &= \arg\max_{\sigma}\text{EU}[\mathcal{I}[\sigma]] = \text{EU}[\mathcal{I}[\sigma^*]] \\
&= \sum_{U \in \mathcal{U}} \sum_{u \in Val(U)} u P_{\mathcal{B}_{\mathcal{I}[\sigma^*]}}(U = u) \\
&= \sum_{U \in \mathcal{U}} \sum_{u \in Val(U)} \sum_{\boldsymbol{x} \in Val(\mathcal{X})} \sum_{d \in Val(D)} u P(U = u \mid \boldsymbol{x}, d) P_{\mathcal{B}_{\mathcal{I}[\sigma^*]}}(\boldsymbol{x}, d)
\end{aligned}$$

   where

$$P_{\mathcal{B}_{\mathcal{I}[\sigma^*]}}(\boldsymbol{x}, d) = \delta_D^{\sigma^*}(d \mid \mathbf{pa}_D) \prod_{Y \in \mathcal{X}} P(y \mid \mathbf{pa}_Y).$$

   Now, consider a strategy $\hat{\gamma}$ for ID' in which we "ignore" the value of X in our choice of action at the decision node D and act optimally otherwise; this corresponds to the decision rule $\delta_D^{\hat{\gamma}}(D \mid X, \mathbf{Pa}_D) = \delta_D^{\sigma^*}(D \mid \mathbf{Pa}_D)$. Then,

$$\begin{aligned}
P_{\mathcal{B}_{\mathcal{I}'[\hat{\gamma}]}}(\boldsymbol{x}, d) &= \delta_D^{\hat{\gamma}}(d \mid \mathbf{pa}_D) \prod_{Y \in \mathcal{X}} P(y \mid \mathbf{pa}_Y) \\
&= \delta_D^{\sigma^*}(d \mid \mathbf{pa}_D) \prod_{Y \in \mathcal{X}} P(y \mid \mathbf{pa}_Y) \\
&= P_{\mathcal{B}_{\mathcal{I}[\sigma^*]}}(\boldsymbol{x}, d),
\end{aligned}$$

   so it follows that $\text{MEU}[\mathcal{I}] = \text{EU}[\mathcal{I}[\sigma^*]] = \text{EU}[\mathcal{I}'[\hat{\gamma}]] \leq \text{EU}[\mathcal{I}'[\gamma^*]] = \text{MEU}[\mathcal{I}']$.

   In particular, if the optimal strategy γ^* in $\mathcal{I}'$ does not differ from the optimal strategy σ^* in $\mathcal{I}$ for any value x and assignment $\mathbf{pa}_D$, then $\gamma^* = \hat{\gamma}$, so the equality holds and $\text{VPI}_{\mathcal{I}}(D \mid X) = 0$. On the other hand, if the optimal strategy γ^* in $\mathcal{I}'$ does differ for some value x and assignment $\mathbf{pa}_D$, then strategy γ^* must give strictly higher expected utility that strategy $\hat{\gamma}$ (otherwise, our tie-breaking convention would have ensured that we would stay with $\hat{\gamma}$). This implies that $\text{MEU}[\mathcal{I}'] = \text{EU}[\mathcal{I}'[\gamma^*]] > \text{EU}[\mathcal{I}'[\hat{\gamma}]] = \text{EU}[\mathcal{I}[\sigma^*]] = \text{MEU}[\mathcal{I}]$, or $\text{VPI}_{\mathcal{I}}(D \mid X) > 0$, as desired.

2. **[6 points]** Is the value of learning the values of two variables equal to the sum of the values of learning each of them? I.e., is $\text{VPI}_{\mathcal{I}}(D \mid Y, Z) = \text{VPI}_{\mathcal{I}}(D \mid Y) + \text{VPI}_{\mathcal{I}}(D \mid Z)$. Either prove that this assertion holds, or provide a counterexample.

   Answer: No, the value of learning the values of two variables is not necessarily equal to the sum of the values of learning each of them. We provide a counterexample.

   Consider an influence diagram consisting of two binary-valued chance variables Y and Z, a binary-valued decision node D, and a utility node U. Suppose that Y has a uniform prior over its two possible values, Z is deterministically related to its parent Y by the relationship $Z = Y$, and $U(x, y, d)$ is 1 if $x = y = d$ and 0 otherwise.

   Without additional information, the optimal action at D is to select randomly. Calling this strategy σ, we have $\text{MEU}[\mathcal{I}] = \text{EU}[\mathcal{I}[\sigma]] = \frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 1 = \frac{1}{2}$. Given knowledge of one of the variables (say, Y), an optimal action at D is to pick $D = Y$. Calling this strategy σ_1 and letting the influence diagram $\mathcal{I}_1$ contain an edge from Y to D, we have $\text{MEU}[\mathcal{I}_1] = \text{EU}[\mathcal{I}_1[\sigma_1]] = 1$. If we add another edge from Z to D, however, we don't have any more information than before. Calling this strategy σ_2 and letting the influence diagram $\mathcal{I}_2$ contain edges from both Y and Z to D, it follows that $\text{MEU}[\mathcal{I}_2] = \text{EU}[\mathcal{I}_2[\sigma_2]] = \text{EU}[\mathcal{I}_1[\sigma_1]] = \text{MEU}[\mathcal{I}_1]$.

   Thus, $\text{VPI}_{\mathcal{I}}(D \mid Z) = \text{VPI}_{\mathcal{I}}(D \mid Y) = \text{MEU}[\mathcal{I}_1] - \text{MEU}[\mathcal{I}] = \text{MEU}[\mathcal{I}_2] - \text{MEU}[\mathcal{I}] = \text{VPI}_{\mathcal{I}}(D \mid Z, Y) = \frac{1}{2}$, so $\text{VPI}_{\mathcal{I}}(D \mid Y, Z) = \frac{1}{2} < 1 = \text{VPI}_{\mathcal{I}}(D \mid Y) + \text{VPI}_{\mathcal{I}}(D \mid Z)$.

New (chapter 23) — Value of Imperfect Information [7 points]

In a decision problem, we know how to calculate the value of perfect information of X at decision D. Now imagine that we cannot observe the exact value of X, but we can instead observe a noisy estimate of X.

For this problem, assume X is binary. Also assume the noisy observation has a false positive rate of p and a false negative rate of q. (That is when $X = 0$ we observe 1 with probability p, and when $X = 1$ we observe 0 with probability q.)

Give a simple method by which we can calculate the improvement in MEU from observing this imperfect information. (Your answer should be just a couple lines long, but you should explain exactly how p and q are used.)

Answer: We introduce a new node, X' into our influence diagram, as a child of X, and CPD according to the given noise model - $P(X'|X) = 1 - p, q; p, 1 - q$. Then simply calculate the VPI for X' at decision D.