

CS 228, Winter 2009

Problem Set #2 Solutions

For each problem, a number of error codes describing common mistakes made by students are listed below. If you feel that your homework has been wrongly graded, please come see us. All error codes are tentative in this handout. They will go through modifications as we grade the homeworks.

1. [5 points] MAP Dirichlet

Suppose that a prior on a parameter vector is $p(\boldsymbol{\theta}) \sim \text{Dirichlet}(\alpha_1, \dots, \alpha_k)$. What is the MAP value of the parameters, that is $\text{argmax}_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathcal{D})$? Assume M_1, M_2, \dots, M_k are the sufficient statistics from the data set \mathcal{D} .

Answer:

$$\frac{M_k + \alpha_k - 1}{M + \alpha - k}, \text{ where } \alpha = \sum_{i=1}^k \alpha_i \text{ and } M = \sum_{i=1}^k M_i$$

Error codes:

- (1.1) [3 points] Mis-understood the question as the MLE problem.
- (1.2) [1 point] didn't get that the posterior is also a dirichlet.
- (1.3) [1 point] Minor error.

2. [12 points] Search in Structure Learning

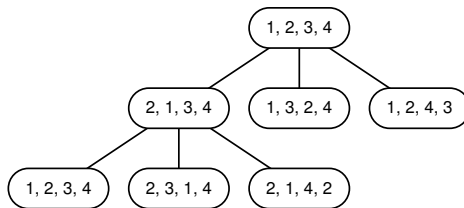


Figure 1: Partial search tree example for orderings over variables X_1, X_2, X_3, X_4 . Successors to $\prec = (1, 2, 3, 4)$ and $\prec' = (2, 1, 3, 4)$ shown.

Consider learning the structure of a Bayesian network for some given ordering, \prec , of the variables, X_1, \dots, X_n . This can be done efficiently as described in Section 14.3.2 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 $i = 1, \dots, n-1$. 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 1). 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 j th 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 $\text{FamScore}_j(\prec'' : \mathcal{D})$ for each variable $X_j, j = 1, \dots, n$.
- If $j > i + 1$, $\text{FamScore}_j(\prec'' : \mathcal{D}) = \text{FamScore}_j(\prec' : \mathcal{D})$.
- If $j < i$, $\text{FamScore}_j(\prec'' : \mathcal{D}) = \text{FamScore}_j(\prec' : \mathcal{D})$.
- If $j = i$ or $j = i + 1$:
 - If $i = h$, $\text{FamScore}_j(\prec'' : \mathcal{D}) = \text{FamScore}_j(\prec : \mathcal{D})$.
 - If $i > h + 1$ or $i < h - 1$, $\text{FamScore}_j(\prec'' : \mathcal{D}) = \text{FamScore}_j(Succ_i(\prec) : \mathcal{D})$.
 - If $i = h + 1$ or $i = h - 1$, compute the family score from scratch: $\text{FamScore}_j(\prec'' : \mathcal{D}) = \text{argmax}_{\mathcal{U} \in \{X_k : X_k \prec'' X_j\}} \text{FamScore}(X_j | \mathcal{U}_j : \mathcal{D})$

Error codes:

- (2.1) [3 points] Incorrectly handling the case of $j < i$ or $j > i + 1$.
- (2.2) [3 points] Incorrectly handling the case of $i = h$.
- (2.3) [3 points] Incorrectly handling the case of $i > h + 1$ or $i < h - 1$.
- (2.4) [3 points] Incorrectly handling the case of $i = h + 1$ or $i = h - 1$.
- (2.5) [2 points] Minor miscellaneous errors: caching too much, recomputing too much, not showing how to actually compute the BIC score (not showing which family scores to recompute)
- (2.6) [8 points] Major miscellaneous errors

3. [15 points] Module Network

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

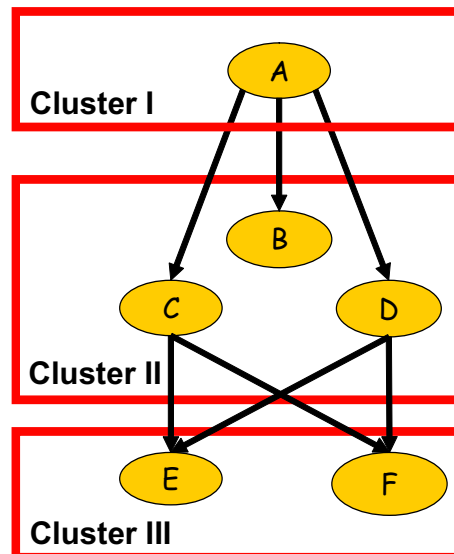


Figure 2: A simple module network

The cluster network structure defines a ground Bayesian network where, for each variable X , we have the parents \mathbf{U}_k for $k = \mathcal{A}(X)$ and the CPD $P_k(X | \mathbf{U}_k)$. Figure 2 shows an example of such a 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.

- [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.
- 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 $o_{k \rightarrow k'}(X)$ 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.

- [2 points] Why did we not include edge reversal in our set of operators?
- [2 points] Describe an efficient implementation for the update associated with the **Node-Move** operator.
- [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.)

Estimate: 1 1/2 pages, 1 1/2 hours

Answer:

- (a) [5 points] To parameterize a module network, we only need one CPD for each cluster, i.e., $P_k(X|U_k)$, $k = 1, 2, \dots, K$. The set of parameters for each $P_k(X|U_k)$ is $\{\theta_{x|u_k} | u_k \in \text{Val}(U_k)\}$. 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 $P_k(X|U_k)$).
- (b) i. [2 points] 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.
- ii. [2 points] 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 subtracting X 's statistics from the sufficient statistics of C_k and adding the sufficient statistics regarding X to the overall sufficient statistics of $C_{k'}$.
- iii. [6 points] 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 $C_{k'}$.
4. [15 points] **Sampling on a Tree**

Suppose we have a distribution $P(\mathbf{X}, \mathbf{E})$ over two sets of variables \mathbf{X} and \mathbf{E} . Our distribution is represented by a nasty Bayes Net with very dense connectivity, and our sets of variables \mathbf{X} and \mathbf{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 $P(\mathbf{X} = \mathbf{x} | \mathbf{E} = \mathbf{e})$. More specifically, we will use a tree-structured Bayes Net as the proposal distribution for use in the importance sampling algorithm.

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

Answer: No, No, Yes

Error codes:

(4.1) [0.5 points] No explanation to one or more parts.

(4.2) [1 points] Answered one or more parts wrong.

- (b) [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 $Q(\mathbf{X}, \mathbf{E})$ defined by this network is “close” to the distribution $P(\mathbf{X}, \mathbf{E})$. You now want to use *the posterior* in Q as your proposal distribution for importance sampling.

- i. 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 $Q(\mathbf{X} | \mathbf{E} = \mathbf{e})$. Describe how to sample from this posterior, once it has been computed.
- ii. 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 $P(\mathbf{X} | \mathbf{E} = \mathbf{e})$. Show precisely how you determine the weights $w[m]$ for the samples.
- iii. Show the form of the final estimator $\hat{P}(\mathbf{X} = \mathbf{x} | \mathbf{E} = \mathbf{e})$ for $P(\mathbf{X} = \mathbf{x} | \mathbf{E} = \mathbf{e})$, in terms of the samples from part i, and the weights from part ii.

Answer: Create a clique tree where each clique is a pair of variables (child and parent). Multiply in the indicator functions for the evidence $\mathbf{E} = \mathbf{e}$. The distribution across the tree now represents $Q(\mathbf{X}, \mathbf{E} | \mathbf{E} = \mathbf{e})$. Calibrate the tree. Now, the belief at a clique over (X, \mathbf{Pa}_X) is proportional to $Q(X, \mathbf{Pa}_X | \mathbf{E} = \mathbf{e})$. From this belief, we can easily compute $Q(X | \mathbf{Pa}_X, \mathbf{E} = \mathbf{e})$ (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).

To get weights:

$$w[m] = \frac{P(\mathbf{x}[m], \mathbf{e}[m])}{Q(\mathbf{x}[m], \mathbf{e}[m] | \mathbf{e})}$$

The estimator:

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

Error codes:

- (4.3) **[1 points]** Imprecise description of clique tree construction and use
 - (4.4) **[2 points]** No mention of how to compute $Q(X | \mathbf{Pa}_X, \mathbf{E} = \mathbf{e})$ from the calibrated clique tree
 - (4.5) **[2 points]** Wrong weight expression
 - (4.6) **[2 points]** Wrong estimator expression
- (c) **[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:

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

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.

Error codes:

- (4.7) **[3 points]** Did not write an expression for w_{ij} in terms of the estimator and the weights.
- (4.8) **[4 points]** Provided a generic answer for the weights involving family score, mutual information, etc., but did not hit on the actual value for the weight.
- (4.9) **[1 points]** Minor Error
- (4.10) **[2 points]** Right initial expression for weight (in terms of family scores), wrong final expression in terms of estimator and weights
- (4.11) **[7 points]** No answer given
- (4.12) **[5 points]** Wrong answer

Estimate: 2 pages, 2 hours

5. **[8 points] EM with Uniform Initialization**

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

Now given a data set $\mathcal{D} = \{\mathbf{x}[1], \dots, \mathbf{x}[M]\}$, where each $\mathbf{x}[m]$ is a complete assignment to the evidence variables, X_1, \dots, 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,

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

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.

Estimate: 1 1/2 pages, 1 hour

Answer:

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

$$\begin{aligned}
\theta_{C=c}^1 &= \frac{\bar{M}_{\theta^0}[c]}{M} = \frac{1}{M} \sum_m P(c \mid \mathbf{x}[m], \theta^0) \\
&= \frac{1}{M} \sum_m \frac{P(\mathbf{x}[m] \mid c, \theta^0) P(c \mid \theta^0)}{P(\mathbf{x}[m] \mid \theta^0)} \\
&= \frac{1}{M} \sum_m \frac{P(\mathbf{x}[m] \mid c, \theta^0) P(c \mid \theta^0)}{\sum_{c' \in \text{Val}(C)} P(\mathbf{x}[m] \mid c', \theta^0) P(c' \mid \theta^0)} \\
&= \frac{1}{M} \sum_m \frac{P(c \mid \theta^0) \prod_{i=1}^n P(\mathbf{x}_i[m] \mid c, \theta^0)}{\sum_{c' \in \text{Val}(C)} P(c' \mid \theta^0) \prod_{i=1}^n P(\mathbf{x}_i[m] \mid c', \theta^0)} \\
&= \frac{1}{M} \sum_m \frac{\frac{1}{|\text{Val}(C)|} \prod_{i=1}^n \frac{1}{|\text{Val}(X_i)|}}{\sum_{c' \in \text{Val}(C)} \frac{1}{|\text{Val}(C)|} \prod_{i=1}^n \frac{1}{|\text{Val}(X_i)|}} \\
&= \frac{1}{|\text{Val}(C)|}
\end{aligned}$$

and

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

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

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

and

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

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

$$\theta_{C=c}^t = \frac{1}{|\text{Val}(C)|} \quad \text{and} \quad \theta_{X_i=x|C=c}^t = \frac{M[x]}{M}.$$

Error Codes:

- (5.1) [3 points] Failed to show that $\theta_c^1 = \frac{1}{|\text{Val}(C)|}$
- (5.2) [3 points] Failed to show that $\theta_{x_i|c}^1 = \frac{M[x_i]}{M}$
- (5.3) [2 points] Failed to show convergence.
- (5.4) [1 points] Minor error.

6. [15 points] Message passing with EM

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 $P(X_i | \mathbf{U}_i)$. 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 15.C on accelerating convergence).

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

- (a) [10 points] Assume we have a clique tree for the network \mathcal{G} and that the CPD $P(X_i | \mathbf{U}_i)$ was assigned to clique \mathbf{C}_j . Analyze which messages change after we update the parameters for $P(X_i | \mathbf{U}_i)$. 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 \mathbf{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 $\mathbf{o}[m]$.

Now supposed we have updated the parameters for $P(X_i | \mathbf{U}_i)$, 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 \mathbf{C}_j will change.

To complete the E-step, we collect the sufficient statistics:

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

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

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

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:

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

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

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

where ψ'_j is the updated initial belief of \mathbf{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 $P(X_i | \mathbf{U}_i)$ is one of these factors for \mathbf{C}_j .

Note that the only messages that are used ($\delta_{i \rightarrow j}$) were unchanged during calibration (since incoming messages to \mathbf{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 \mathcal{C}_j ; therefore the computational cost depends only on the size of \mathcal{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.

- (b) [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.

Error codes:

- (6.1) [8 points] Incorrect/incomplete analysis of the messages
- (6.2) [1 point] Incorrect/missing E-step computation
- (6.3) [1 point] Incorrect/missing M-step computation
- (6.4) [5 points] Incorrect part (b)

7. [15 points] Markov Network Learning

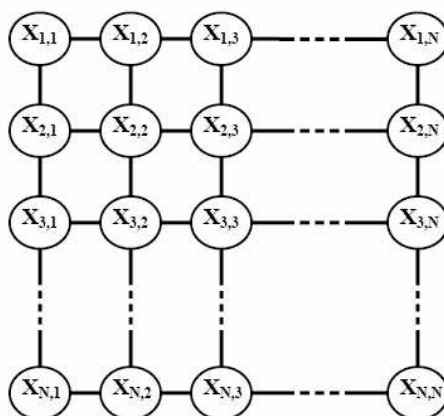


Figure 3: A $N \times N$ grid

Assume that we are learning Markov Network parameters on a $N \times N$ grid (see Figure 3), given a dataset \mathcal{D} of M examples $\mathcal{D} = \{\xi[m] : m = 1, \dots, M\}$. As usual, we consider a log-linear parameterization of the Markov network, using a set of k features $\Phi = \{f_i : i = 1, \dots, k\}$. Each feature is defined over a pair of neighboring variables, e.g. $f_i(X_{2,2}, X_{2,3})$. There is no parameter sharing.

We define the *generalized pseudo-likelihood* objective as:

$$\ell_{gen-pseudo}(\theta : \mathcal{D}) = \frac{1}{M} \sum_m \sum_j \ln P(\mathbf{X}_j[m] \mid \mathbf{X}_{-j}[m], \theta)$$

where $\mathbf{X}_j = \{X_{j,1}, X_{j,2}, \dots, X_{j,N}\}, \forall j = 1, \dots, N$, and \mathbf{X}_{-j} stands for all the variables on other horizontal chains, i.e. $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_{j-1}, \mathbf{X}_{j+1}, \dots, \mathbf{X}_N$.

- (a) [2 points] Write down the probability $P(\mathbf{X}_j \mid \mathbf{X}_{-j})$.
- (b) [10 points] Derive the gradient $\frac{\partial}{\partial \theta_i} \ln P(\mathbf{X}_j \mid \mathbf{X}_{-j})$.
- (c) [3 points] Derive the gradient for the *generalized pseudo-likelihood* objective, that is $\frac{\partial}{\partial \theta_i} \ell_{gen-pseudo}(\boldsymbol{\theta} : \mathcal{D})$.

Answer: Similar to section 16.5.1 on course reader (pp.400-401), but is adapted to the grid structure.

Error codes:

- (7.1) [1 points] didn't mention that $X_j \cap D_i \neq \phi$ only involves $j - 1, j, j + 1$ rows.
- (7.2) [1 points] didn't specify that the derivative in (b) equals 0 when $X_j \cap D_i = \phi$.
- (7.3) [2 points] need more deriving details.
- (7.4) [2 points] didn't specify $X_j \cap D_i$ in the answer of (c).
- (7.5) [2 points] missing (a)
- (7.6) [10 points] missing (b)
- (7.7) [3 points] missing (c)
8. [8 points] **Decomposable Utility Functions**

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:

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

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

- (a) [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.
- (b) [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: 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.

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

Error codes:

- (8.1) [5 points] Assumed the sets of variables in each utility function were disjoint from each other.
- (8.2) [3 points] Provided a non-tractable solution that might save in some cases, but not in general
- (8.3) [3 points] Used a greedy approach that would produce an approximately optimized answer
- (8.4) [3 points] Didn't do, or gave bad explanation for, part (b).
- (8.5) [2 points] Used max-product instead of sum for finding the optimizing assignment.

9. [7 points] Value of Imperfect Information

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 .

Error codes:

- (9.1) [3 points] Correct ideas, but didn't show entirely show to calculate the value of imperfect information.
- (9.2) [4 points] Wrong answer, with some general idea.
- (9.3) [1 points] Minor error