CS 228, Winter 2007
Final Solutions
Handout #18

You have 24 hours to complete this exam. The exam is given out at noon, and due at noon (12:00 pm) one day after you pick it up. The exam will be handed out and collected in Gates 120 (the Fishbowl). This exam is long and difficult, and we do not expect everyone to finish all of the questions. Be sure to use good test taking skills and attack the easier problems first, spend more time on questions worth more points, and generally pay attention to how you spend your time. Also, you are welcome (and we expect you) to use or refer algorithms from the reader when appropriate, without having to rederive or explain them. Furthermore, please use standard notation from the reader (when possible) and clearly define any terms you introduce. Algorithm answers should be provided in the form of pseudocode (with explanations where necessary), and your answers will be easier to grade (read: you will get higher grades) if you use proper spacing and layout of your answers on the page. We have provided approximate times with each of the problems to give you a rough estimate of how long we think it might take (in time and pages not including diagrams).

1. **[10 points] Representation**

   (a) [**5 points**] 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 $\mathcal{I}(G)$. That is, prove Corollary 5.7.4 — Corollary 5.5.4 in the version of Chapter 5 posted on Coursework — for the case in which G has no immoralities. (You may not simply assume this corollary or use theorems that follow it in the readings.)

   (b) [**5 points**] Let X and Y be variables in a Bayesian network G, and suppose that neither is a parent of the other. Let $W = \mathbf{Pa}_X \cup \mathbf{Pa}_Y$. Prove that $(X \perp Y \mid W) \in \mathcal{I}(G)$.

   Answer: There were many acceptable answers to this question (we counted at least five distinct ones)

   Estimate: 1 page, 1 hour

2. **[20 points] Sampling on a Tree**

   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.

   (a) [**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 $P'(\boldsymbol{x} \mid \boldsymbol{e}) = P(\boldsymbol{x}, \boldsymbol{e})$ exactly, in a tractable way? For each question, provide a Yes/No answer and a single sentence explanation or description.

   Answer: No, No, Yes

   Error codes:

(2.1) **[1 points]** Answered one or more parts wrong.

(b) **[12 points]** Now, suppose your friendly TAs have given you a tree network (each variable besides the root has exactly one parent) that defines a distribution Q. They tell you that $Q(\boldsymbol{X}, \boldsymbol{E})$ is "close" to the distribution $P(\boldsymbol{X}, \boldsymbol{E})$ of the nasty network. You now want to use *the posterior* in Q as your proposal distribution for importance sampling. You now must perform the two steps of importance sampling:

i. Show how to sample from the posterior in Q. More specifically, describe an algorithm for drawing samples $\boldsymbol{x}[m]$ *exactly* from $Q(\boldsymbol{X} \mid \boldsymbol{E} = \boldsymbol{e})$ using only tractable techniques. You answer should be at the level of pseudocode, and should indicate the specific distribution that each variable will be sampled from, and an explanation of how you computed that distribution.

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 $P(\boldsymbol{X} \mid \boldsymbol{E} = \boldsymbol{e})$. Show precisely how you determine the weights $w[m]$ for the samples.

iii. 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: 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).

To get weights:

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

The estimator:

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

Error codes:

(2.2) **[7 points]** Provided a nontractable solution. These included breaking the tree up into subtrees and doing full inference on those or other solutions that could be tractable in some cases, but not in general.

(2.3) **[4 points]** Used forward sampling. These samples are NOT drawn "directly" from the posterior.

(2.4) **[4 points]** Used likelihood weighting. These samples are NOT drawn "directly" from the posterior. After you correct for the weight in part (ii), the result is the same as if you had just done forward sampling.

(2.5) **[2 points]** Included a $P(\boldsymbol{X} \mid \boldsymbol{E})$ in the weight calculation, when this is exactly the (intractable) value we are estimating.

(2.6) **[2 points]** Did not normalize the estimator by $\sum w[m]$. This amounts to trying to use unnormalized importance sampling, when only normalized will work in this case.

(c) **[5 points]** Examining the results of your sampling, you find that 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.

Error codes:

(2.7) **[2 points]** Did not write an expression for w_{ij} in terms of the estimator and the weights.

(2.8) **[2 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.

Estimate: 2 pages, 2 hours

3. **[10 points] Time-Series Models**

Geremy and Ben have gotten bored with their current research on image segmentation and have decided to try some slightly more interesting tasks. Specifically, they want to use a graphical model to classify what activity their friend Steve is doing in a video sequence. They are considering the three models shown in Fig. 1. In each of these models, we assume the state variables $X^{(t)}$ are discrete random variables that specify the task being performed in the current frame of the video sequence (*i.e.* Steve is dancing, running, walking, picking up a ball); these variables are always unobserved. The observations (and features of the observations) are real-valued variables that describe the position and orientation of all the objects as well as Steve's limbs and body in the scene. For each of the time steps, Ben and Geremy get a complete instantiation to all these observed variables.

Geremy and Ben are having some trouble selecting which model to use, though, and would like some help. They know that they will need to solve the following inference problem: $P(X^{(t)} \mid \boldsymbol{o}^{(1:T)})$, for every t in the sequence. They are considering three models: an HMM

(Hidden Markov Model — Fig. 1(a)), an MEMM (Maximum Entropy Markov Model (don't worry about the name, only the structure) — Fig. 1(b)), and a CRF (Conditional Random Field — Fig. 1(c)). Your task is to help them choose a model, keeping in mind the different independence assumptions made by the models.

(a) [**5 points**] Describe one specific advantage the HMM has over the MEMM.

(b) [**5 points**] Describe one specific advantage the CRF has over the HMM.

Guess it looks like they should use a CRF, huh?

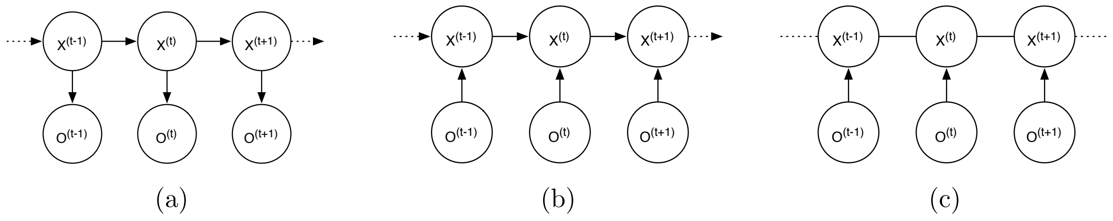

Figure 1: (a) An **HMM**. (b) An **MEMM**. (c) A **CRF**.

Estimate: A couple of sentences, 1/2 hour

Answer:

(a) The MEMM assumes that the observation at any time t' is independent of the state for all $t < t'$. Thus, we cannot perform the smoothing task in an MEMM; we can in an HMM.

Error Codes:

(3.1) **[5 points]** Did not cite any realistic weakness

(3.2) **[3 points]** Miscellaneous error

(b) The HMM assumes that all the observations are independent, and it is very difficult to describe a probability distribution over real-valued observations. The CRF is insensitive to these issues, and it can use features (observations) that are correlated across time and real-valued.

Error Codes:

(3.3) **[5 points]** Did not cite realistic weakness

(3.4) **[3 points]** Miscellaneous error

4. [**10 points**] **Expectation Maximization in a Naive Bayes Model**

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.

Estimate: 1 1/2 pages, 1 hour

Answer:

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

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

and

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

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 $t \geq 1$, $\theta_{C=c}^t = \frac{1}{|Val(C)|}$ and $\theta_{X_i=x|C=c}^t = \frac{M[x]}{M}$. 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 $c, c' \in Val(C)$. 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

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

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

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

Error Codes:

(4.1) **[4 points]** Failed to show that $\theta^1_c = \frac{1}{|Val(C)|}$

(4.2) **[4 points]** Failed to show that $\theta^1_{x_i|c} = \frac{M[x_i]}{M}$

(4.3) **[2 points]** Failed to show convergence.

(4.4) **[1 points]** Minor error.

5. **[20 points] Parameter Learning, Markov Networks** In this problem, we will consider the problem of learning parameters for a Markov network with missing data. As in the case of Bayesian networks, if we assume our data is missing at random, we can perform maximum likelihood parameter estimation by using gradient ascent to optimize the likelihood function. As usual, we assume we have a dataset $\mathcal{D}$ in which some entries are missing, and we denote the variables in our network by $\boldsymbol{X}$. Let $\boldsymbol{o}[m]$ be the observed entries in the m^{th} data instance and let $\mathcal{H}[m]$ be the random variables that are missing entries in that instance, so that for any $\boldsymbol{h}[m] \in Val(\mathcal{H}[m])$, $\langle \boldsymbol{o}[m], \boldsymbol{h}[m] \rangle$ is a complete assignment to $\boldsymbol{X}$. Further, we assume that $|\mathcal{D}| = M$. You should note the correspondence of this problem to question 5 on homework 4.

   We assume we are working with log-linear Markov networks in this problem, so that

$$P(\boldsymbol{x}) = \frac{1}{Z} \prod_{i=1}^{n} \exp(\theta_i \phi_i[\boldsymbol{x}_i])$$

   where $\boldsymbol{x}_i$ is an assignment to the set of variables $\boldsymbol{X}_i \subseteq \boldsymbol{X}$.

   (a) **[4 points]** Write the log-likelihood $\ell(\boldsymbol{\theta} : \mathcal{D})$ of the dataset $\mathcal{D}$ with hidden variables for a log-linear model as described above.

   (b) **[7 points]** Prove that

$$\frac{\partial}{\partial \theta_i} \ell(\boldsymbol{\theta} : \mathcal{D}) = \left[\sum_{m=1}^{M} \boldsymbol{E}_{\boldsymbol{h}[m] \sim P(\mathcal{H}[m] | \boldsymbol{o}[m], \boldsymbol{\theta})}[\phi_i] \right] - M \boldsymbol{E}_{\boldsymbol{\theta}}[\phi_i].$$

   where $\boldsymbol{E}_{\boldsymbol{h}[m] \sim P(\mathcal{H}[m] | \boldsymbol{o}[m], \boldsymbol{\theta})}[\phi_i] = \sum_{\boldsymbol{h}[m]} \phi_i[\boldsymbol{h}_i[m], \boldsymbol{o}_i[m]] P(\boldsymbol{h}[m] | \boldsymbol{o}[m], \boldsymbol{\theta})$. To attempt to find the maximum likelihood parameterization, we would take gradient steps on $\ell(\boldsymbol{\theta} : \mathcal{D})$ of the above form. How many times must we run inference to take one gradient step?

   (c) **[7 points]** Now, instead of doing gradient-based optimization on $\ell(\boldsymbol{\theta} : \mathcal{D})$ we would like to perform EM. Write down the E-step and the M-step for a log-linear Markov network with hidden variables as described above. Specifically, the E-step is estimating the counts for the feature ϕ_is, and the M-step will be optimizing the parameters $\boldsymbol{\theta}$ given using gradient ascent. Write the E-step and the gradient used in the M-step. How many times will you have to perform inference in your model during the E-step? How many times during each gradient step in the M-step?

   (d) **[2 points]** What is the difference between the gradient processes in part (b) and part (c)?

   Estimate: 2 pages, 2 hours

   Answer:

   (a) The likelihood for a single instance takes the form

$$L(\boldsymbol{\theta} : \boldsymbol{o}[m], \mathcal{H}[m]) = \sum_{\boldsymbol{h}[m]} \frac{1}{Z} \exp\left(\sum_{i=1}^{n} \theta_i \phi_i(\boldsymbol{h}_i[m], \boldsymbol{o}_i[m]) \right).$$

   Thus, taking the log-likelihood for the entire dataset, we have

$$\ell(\boldsymbol{\theta} : \mathcal{D}) = \sum_{m=1}^{M} \left[\log \sum_{\boldsymbol{h}[m]} \exp\left(\sum_{i=1}^{n} \theta_i \phi_i[\boldsymbol{h}_i[m], \boldsymbol{o}_i[m]] \right) - \log Z \right].$$

Error codes:

(5.1) **[2 points]** Did not handle case of different sets of $\boldsymbol{h}[m]$.

(5.2) **[4 points]** Totally wrong

(b) The gradient calculation follows closely the gradient in the book for Markov networks, as well as the gradient calculation for Problem Set 4 for a CRF. In the following, let

$$Z(\boldsymbol{o}_[m]) = \sum_{\boldsymbol{h}[m]} \exp\left(\sum_{i=1}^{n} \theta_i \phi_i[\boldsymbol{h}_i[m], \boldsymbol{o}_i[m]]\right).$$

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} \ell(\boldsymbol{\theta} : \mathcal{D}) &= \sum_{m=1}^{M} \frac{\partial}{\partial \theta_j} \log Z(\boldsymbol{o}[m]) - \frac{\partial}{\partial \theta_i} \log Z \\
&= \sum_{m=1}^{M} \frac{1}{Z(\boldsymbol{o}[m])} \sum_{\boldsymbol{h}[m]} \exp\left(\sum_{i=1}^{n} \theta_i \phi_i[\boldsymbol{h}_i[m], \boldsymbol{o}_i[m]]\right) \phi_j[\boldsymbol{h}_j[m], \boldsymbol{o}_j[m]] \\
&\quad - \frac{1}{Z} \sum_{\boldsymbol{x}} \exp\left(\sum_{i=1}^{n} \theta_i \phi_i[\boldsymbol{x}_i]\right) \phi_j[\boldsymbol{x}_j] \\
&= \sum_{m=1}^{M} \sum_{\boldsymbol{h}[m]} \phi_j[\boldsymbol{h}_j[m], \boldsymbol{o}_j[m]] P(\boldsymbol{h}[m] | \boldsymbol{o}[m], \boldsymbol{\theta}) - M \boldsymbol{E}_{\boldsymbol{\theta}}[\phi_j] \\
&= \left[\sum_{m=1}^{M} \boldsymbol{E}_{\boldsymbol{h}[m] \sim P(\mathcal{H}[m] | \boldsymbol{o}[m], \boldsymbol{\theta})}[\phi_j]\right] - M \boldsymbol{E}_{\boldsymbol{\theta}}[\phi_j].
\end{aligned}$$

In every gradient step in the above, we need to run inference $M + 1$ times (once for each instance with hidden variables, once to calculate the expectation of all the features irrespective of evidence).

Error codes:

(5.3) **[1 points]** Mathematical mistake in taking derivative (mulitple mistakes $\Rightarrow$ multiple times this error).

(5.4) **[1 points]** Said inference must be run M times instead of $M + 1$.

(5.5) **[2 points]** Wrong number of inference runs to take a gradient step. Note that running inference gives you the probability of *every* assignment; that is what inference is!

(c) The E-step is easy—we simply use the expected sufficient statistic for the feature ϕ_i:

$$\bar{M}_{\boldsymbol{\theta}^{(t)}}[\phi_i] = \frac{1}{M} \left[\sum_{m=1}^{M} \boldsymbol{E}_{\boldsymbol{h}[m] \sim P(\mathcal{H}[m] | \boldsymbol{o}[m], \boldsymbol{\theta})}[\phi_i]\right].$$

Our gradient simply re-uses the sufficient statistics at every point in the M-step optimization, so that our "gradient" is of the form (where $\boldsymbol{\theta}^{(t,k)}$ are the parameters at iteration k of the M-step gradient optimization, and $\boldsymbol{\theta}^{(t)}$ are the parameters at the t step of the outer EM loop)

$$\bar{M}_{\boldsymbol{\theta}^{(t)}}[\phi_i] - \boldsymbol{E}_{\boldsymbol{\theta}^{(t,k)}}[\phi_i].$$

Thus, for the E-step we need to run inference M times (once for every data instance); the M-step requires that we run inference only once (to calculate $\boldsymbol{E}_{\boldsymbol{\theta}^{(t,k)}}[\phi_i]$) per gradient step.

Error codes:

(5.6) **[2 points]** E-step incorrect.

(5.7) **[2 points]** Confusion or unclear on which parameters $\boldsymbol{\theta}$ to use for the M-step.

(5.8) **[3 points]** M-step incorrect.

(5.9) **[1 points]** Wrong count of inference for E-step.

(5.10) **[1 points]** Wrong count of inference per iteration of the gradient ascent process in the M-step.

(d) The tradeoffs between EM and gradient-based optimization are obvious. In EM, we need inference only once per gradient step inside the M-step, but we re-use old sufficient statistics (which may cost us something when approximating the gradient). The gradient-based method, while requiring more inference, will have more up-to-date sufficient statistics.

Error codes:

(5.11) **[2 points]** Answer did not discuss any of the issues above.

6. [**20 points**] **Factored DBN State Representation**

Recall that when we perform tracking (filtering) in DBNs, in general, we cannot exploit DBN structure to get any factored representation of the exact belief state $\sigma^{(t)}$ due to entanglement. Still, we can enforce our belief state to have a factored representation, and perform *approximate* tracking with it.

In particular, assume that we represent our approximate belief state $\hat{\sigma}^{(t)}(\mathcal{X}^{(t)})$ as a *calibrated clique tree* $\mathcal{T}^{(t)}$ over the variables $\mathcal{X}^{(t)}$. In particular, we have cliques $\boldsymbol{C}_i^{(t)}$, calibrated potentials $\pi_i^{(t)}$, and sepset marginals $\mu_{i,j}^{t}$, for all neighboring cliques i and j ($i < j$). You may wish to review the definitions of calibrated potentials and marginals in the course reader (Sections 9.2.2 and 9.2.3, and in particular Definition 9.2.10).

Our goal is to perform tracking, and express our approximate belief state over the next time step $\hat{\sigma}^{(t+1)}(\mathcal{X}^{(t+1)})$ in the same factored form. In particular $\hat{\sigma}^{(t+1)}(\mathcal{X}^{(t+1)})$ is also a calibrated clique tree $\mathcal{T}^{(t+1)}$ over the variables $\mathcal{X}^{(t+1)}$, with cliques $\boldsymbol{C}_i^{(t+1)}$ (over the same set of variables as $\boldsymbol{C}_i^{(t)}$, only with an incremented time index: $X_j^{(t)} \in \boldsymbol{C}_i^{(t)} \Leftrightarrow X_j^{(t+1)} \in \boldsymbol{C}_i^{(t+1)}$), calibrated potentials $\pi_i^{(t+1)}$, and marginals $\mu_{i,j}^{(t+1)}$.

As explained in class and in the reader, tracking can be performed using a forward pass on a template clique tree of the 2-TBN. In particular, assume we have a template clique tree $\mathcal{S}$ over the variables $\mathcal{X}^{(t)}$ and $\mathcal{X}^{(t+1)}$, with cliques $\boldsymbol{D}_i$, initial potentials ϕ_i, and calibrated potentials ν_i.

(a) [**6 points**] Consider the structure of the template clique tree $\mathcal{S}$. What needs to hold in the relationship between the cliques $\boldsymbol{D}_i$ of the template clique tree $\mathcal{S}$ and the cliques $\boldsymbol{C}_i^{(t)}$, and $\boldsymbol{C}_i^{(t+1)}$ of the approximate belief state clique trees $\mathcal{T}^{(t)}$ and $\mathcal{T}^{(t+1)}$, so that we can perform approximate tracking while maximally preserving the interactions between variables that the tree structure of our approximate belief state can represent. We expect a short answer which relates the scopes of cliques in $\mathcal{S}$, $\mathcal{T}^{(t)}$ and $\mathcal{T}^{(t+1)}$.

Answer:

$$\forall i \exists j \text{ s.t. } \boldsymbol{C}_i^{(t+1)} \subseteq \boldsymbol{D}_j$$

$$\forall i \exists j \text{ s.t. } \boldsymbol{C}_i^{(t)} \subseteq \boldsymbol{D}_j$$

It would be OK not to have the second condition, although the solution to the next part would get a lot more complicated.

Error codes:

(6.1) **[2 points]** Doing it other way around (D to C).

(6.2) **[2 points]** Requiring union of two cliques.

(6.3) **[3 points]** Extra unneeded restrictions.

(6.4) **[5 points]** Major error.

(6.5) **[6 points]** Wrong.

(b) [**14 points**] Describe how you would compute $\hat{\sigma}^{(t+1)}(\mathcal{X}^{(t+1)})$ using $\hat{\sigma}^{(t)}(\mathcal{X}^{(t)})$ and the evidence $o^{(t+1)}$. In particular, show how you would define the initial potentials ϕ_i of the template clique tree $\mathcal{S}$ in terms of the 2-TBN parameters (ϕ'_i) and the belief state $\hat{\sigma}^{(t)}(\mathcal{X}^{(t)})$, and how you would extract the new belief state $\hat{\sigma}^{(t+1)}(\mathcal{X}^{(t+1)})$ as a calibrated clique tree $\mathcal{T}^{(t+1)}$ with cliques $\boldsymbol{C}_i^{(t+1)}$, calibrated potentials $\pi_i^{(t+1)}$, and marginals $\mu_{i,j}^{(t+1)}$. We expect a short answer in pseudocode.

Answer:

1. For each $\boldsymbol{C}_i^{(t)}$ find a $\boldsymbol{D}_j$ s.t. $\boldsymbol{C}_i^{(t)} \subseteq \boldsymbol{D}_j$. Update initial potentials of template clique tree:

$$\phi_j := \phi_j \times \frac{\pi_i^{(t)}}{\prod_k \mu_{i,k}^{(t)}}$$

.

Note division by marginals to avoid over-counting.

2. Calibrate $\mathcal{S}$.
3. Extract calibrated clique potentials:
   For each $\boldsymbol{C}_i^{(t+1)}$ find a $\boldsymbol{D}_j$ s.t. $\boldsymbol{C}_i^{(t+1)} \subseteq \boldsymbol{D}_j$. Letting $\pi_j^{\mathcal{S}}$ be the final belief (posterior) over $\boldsymbol{D}_j$, set

$$\pi_i^{(t+1)} := \sum_{\boldsymbol{D}_j \setminus \boldsymbol{C}_i^{(t+1)}} \pi_j^{\mathcal{S}}$$

4. Now compute marginals over sepsets of neighboring cliques:
   Foreach neighboring $\boldsymbol{C}_i^{(t+1)}$ and $\boldsymbol{C}_i^{(t+1)}$, set

$$\mu_{i,j} := \sum_{\boldsymbol{C}_i^{(t+1)} \setminus \boldsymbol{C}_j^{(t+1)}} \pi_i^{(t+1)}$$

Error codes:

(6.6) **[7 points]** Did not specify *explicitly* how old state is incorporated.

(6.7) **[5 points]** Specified *incorrectly* how old state is incorporated.
(6.8) **[4 points]** Did not account for marginals.
(6.9) **[7 points]** Did not specify extraction.
(6.10) **[5 points]** Specified extraction *incorrectly*.
(6.11) **[2 points]** Minor error.

Estimate: < 1 page, 2 hours

7. [**10 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:

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

(a) [**1 points**] For an arbitrary utility function, what is the computational complexity of finding the maximizing assignment?

Answer: d^M.

Error codes:

(7.1) **[1 points]** Incorrect complexity.

(b) [**9 points**] Now, 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 $\boldsymbol{Y}_i \subset \{X_1 \dots X_M\}$.

i. 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.

ii. 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 ii, 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:

(7.2) **[7 points]** Assumed the sets of variables in each utility function were disjoint from each other.

(7.3) **[4 points]** Provided a nontractable solution that might save in some cases, but not in general

(7.4) **[4 points]** Used a greedy approach that would produce an approximately optimized answer

(7.5) **[4 points]** Didn't do, or gave bad explanation for, part (ii).

(7.6) **[4 points]** Used max-product instead of sum for finding the optimizing assignment.

Estimate: 1/2 page, 1/2 hour