

CS 228, Winter 2008

Problem Set #2

We have provided approximate lengths with each of the problems to give you a rough estimate of how long we think each answer might be not including diagrams. These are meant to be guidelines only to make sure that answers are concise and readable (for diagrams, the larger the better).

1. D-Separation [22 points]

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

- (a) [7 points] For an undirected graph H , recall that we say $sep_H(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z})$ (i.e., \mathbf{X} and \mathbf{Y} are separated given evidence \mathbf{Z} in H) whenever all paths between variables in \mathbf{X} and variables in \mathbf{Y} are blocked by some variables in \mathbf{Z} .

Prove that if we have a moralized graph H derived from an original Bayesian network G then $sep_H(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z})$ implies $d-sep_G(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z})$. This is a generalization of your result from Problem 5 on the previous problem set, and will be used in the next part.

- (b) [15 points] Now, assume that we perform variable elimination on our Bayesian network G , using the Sum-Product-Variable-Elimination algorithm (see section 10.2.1.2). This defines an induced graph I where each intermediate factor ψ during the elimination process defines a clique (see section 11.1). Furthermore, as described in class this defines a clique tree \mathcal{T} where each intermediate factor ψ is represented as a clique in the tree and therefore as a subset of a maximal clique in I .

Let \mathbf{S} be a clique separator in the clique tree \mathcal{T} . Let \mathbf{X} be the set of all variables mentioned in \mathcal{T} on one side of the separator and let \mathbf{Y} be the set of all variables mentioned on the other side of the separator. Prove that $sep_I(\mathbf{X}; \mathbf{Y} \mid \mathbf{S})$ and conclude (using the above result) that $d-sep_G(\mathbf{X}; \mathbf{Y} \mid \mathbf{S})$. (Hint: Remember the running intersection property.)

Estimate: 2 pages

2. Message Passing [14 points]

Let \mathcal{B} be a Bayesian Network over a set of n random variables \mathcal{X} and \mathcal{F} be the set of factors corresponding to the CPDs of \mathcal{B} . Suppose that we then construct a clique tree \mathcal{T} by choosing a variable elimination ordering \prec and finding the maximal cliques in the induced graph $\mathcal{I}_{\mathcal{F}, \prec}$. Our task is to optimize the message passing that takes place as we calibrate \mathcal{T} . For simplicity you may assume $|Val(X)| = d$ for all $X \in \mathcal{X}$ for the following question.

Recall that the message passing operation in SP-Message (page 357 of the reader) works by multiplying the incoming messages (except for one) with the product of the factors assigned to the clique, and then eliminating all of the variables not in the scope of the outgoing message. For example, in the figure below, suppose the pictured clique (called \mathcal{C}_1) contains initial factors $\phi_1(A, B)$, $\phi_2(B, C)$, and $\phi_3(C, D)$ and it receives message $\delta_{1 \rightarrow 2}(A, D)$ from \mathcal{C}_0 . Then an outgoing message $\delta_{2 \rightarrow 3}(B, D)$ is generated by multiplying all initial factors and incoming messages, and then summing out over all variables except B and D .

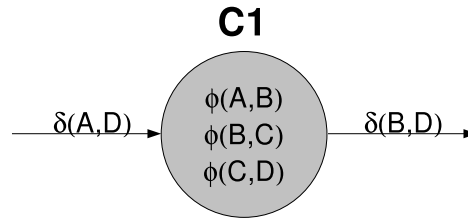


Figure 1: Message Passing Example

Now we wish to improve upon the efficiency of our message passing, but at the cost of the exactness of our algorithm. We would like to force the incoming message into the following factored form: $\tilde{\delta}_{1 \rightarrow 2}(A, D) = \tilde{\delta}_{1 \rightarrow 2}^1(A) \cdot \tilde{\delta}_{1 \rightarrow 2}^2(D)$. Similarly, we would like force to the outgoing message into the following form: $\tilde{\delta}_{2 \rightarrow 3}(B, D) = \tilde{\delta}_{2 \rightarrow 3}^1(B) \cdot \tilde{\delta}_{2 \rightarrow 3}^2(D)$.

- (a) **[6 points]** Show how the outgoing message $\tilde{\delta}_{2 \rightarrow 3}(B, D)$ can be computed more efficiently using this representation. Specifically, you should show precisely how the factored messages $\tilde{\delta}_{1 \rightarrow 2}^1(A)$ and $\tilde{\delta}_{1 \rightarrow 2}^2(D)$ are used along with the initial factors in \mathbf{C}_1 to compute $\tilde{\delta}_{2 \rightarrow 3}(B, D)$. Also, show how this is asymptotically more efficient than standard message passing.
- (b) **[8 points]** Briefly but precisely describe the algorithm underlying this example (no need for pseudo-code), assuming that incoming messages may be forced into the following factored form: $\tilde{\delta}_{i \rightarrow j}(\mathbf{S}_{i,j}) = \prod_k \tilde{\delta}_{i \rightarrow j}^k(\mathbf{A}_{i,j}^k)$, where $\mathbf{A}_{i,j}^k$ is a *disjoint* partition of $\mathbf{S}_{i,j}$. Describe how factored messages are computed and how they are used to compute outgoing messages. Briefly explain why the factored messages are not exact.

Estimate: 1 page

3. **Variable Elimination[22 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 $X_1 - X_2 - \dots - X_n$, and that our clique tree has the form $\mathbf{C}_1 - \dots - \mathbf{C}_{n-1}$ where $\text{Scope}[\mathbf{C}_i] = \{X_i, X_{i+1}\}$. Also assume that each variable X_i has $|\text{Val}(X_i)| = d$.

- (a) **[4 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 .

- (b) **[18 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.

Estimate: 1.5 pages

4. **Optimal Clique Trees[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.)

Estimate: .5 pages

5. Distributional Particle Filtering [30 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*: $P(X^{(t+1)} | X^{(t)})$. Sensor measurements are governed by a *measurement model*: $P(O_i^{(t)} | X^{(t)}, L_i)$. Both the motion model and the measurement model are known.

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

- (a) [4 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 .)

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 distributional particles. For our localization and mapping problem, we have two obvious ways of constructing these distributional particles. We now consider one such approach in detail, and briefly consider the implications of the other.

- (b) [18 points] We select the robot pose trajectory $X^{(1:t)}$ as the set of sampled variables, and maintain a closed form distribution over \mathbf{L} . Thus, at time t we have a set of weighted particles, each of which specifies a full sampled trajectory $\mathbf{x}^{(1:t)}[m]$, a distribution over landmark locations $P_m^{(t)}(\mathbf{L})$, and a weight $w^{(t)}[m]$.
- i. The distribution over \mathbf{L} is still exponentially large in principle. Show how you can represent it compactly in closed form. (Hint: Write out the full form of $P_m^{(t)}(\mathbf{L})$ recalling the definition of distributional particles, and note that our samples represent entire pose trajectories $X^{(1:t)}$.)
 - ii. Describe how to perform a forward propagation step in our distributional 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 $P_m^{(t+1)}(\mathbf{L})$.
- (c) [8 points] As another alternative, we can select the landmark positions \mathbf{L} as our set of sampled variables. Thus, for each particle $\mathbf{l}[m]$, we maintain a distribution $P_m^{(t)}(X^{(t)})$. Without describing the details of how to perform a forward propagation step in our

distributional 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?

Estimate: 2-3 pages