

CS 228, Winter 2009

Problem Set #1

1. Partially Directed Acyclic Graphs (PDAGs) [10 points]

All graphs that have the same skeleton and immoralities are I-equivalent. In this problem we will investigate a compact representation for such an equivalence class. Using this representation, we can provide a quick check for edges with required directionality.

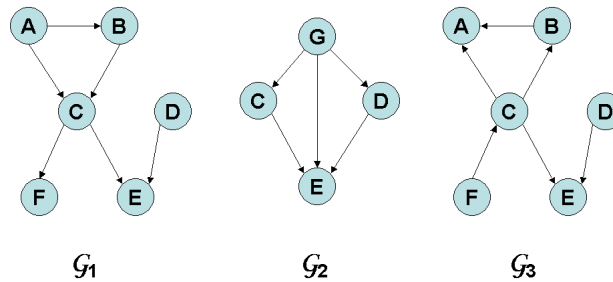


Figure 1: Two Bayesian Networks

- (a) **[3 points]** First consider the case of \mathcal{G}_1 above. For which edges do there exist graphs in the same I-equivalence class in which the edge points in the other direction? Conversely, which edges have fixed direction across all graphs in the I-equivalence class? Based on this example, can you construct a simple but general sufficient condition for some edges to have fixed direction? The condition applies to a local structure of three nodes with fixed skeleton and immorality. If we find this structure in the graph, we can fix the direction of its two edges.
- (b) **[3 points]** Now assume the edge from A to B and the edge from B to C in \mathcal{G}_1 have fixed direction for some reason (i.e., in all I-equivalent graphs, the edge must take the direction from A to B and also from B to C). As a result of this, which additional edge in the graph is forced to have a fixed direction? Now with this additional edge fixed, there is one more edge that is also forced to have fixed direction. Can you find that one? Provide the two (general) sufficient conditions for a fixed edge that you applied in this example. The first condition applies to a local structure of three nodes with three edges, two of which have fixed direction. The second condition applies to a local structure of three nodes with two edges, one of which has fixed direction. If we find either of the above structures in the graph, we can fix the direction of the remaining edge.

(c) [4 points] Now suppose we take the case of \mathcal{G}_2 . Applying the general conditions determined in (a) and (b), we get several edges with fixed directionality. However, there is one more edge that is not covered by the previous conditions, but still has fixed direction in all I-equivalent graphs. Which edge is that? Can you determine the (final) general sufficient condition that will cover this case? The condition applies to a local structure of four nodes with five edges, two of which have fixed direction. If we find this structure in the graph, we can fix the direction of one more edge.

2. **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 $X_1 - X_2 - \dots - X_n$, and that our clique tree has the form $C_1 - \dots - C_{n-1}$ where $Scope[C_i] = \{X_i, X_{i+1}\}$. Also assume that each variable X_i has $|Val(X_i)| = d$.

(a) [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 .

(b) [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.

Estimate: 1.5 pages

3. **Greedy Ordering for Variable Elimination** [20 points]

In this problem we will consider the computational implications of various variable elimination ordering strategies.

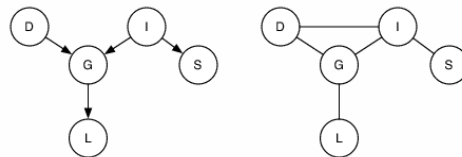


Figure 2: \mathcal{G} and $\mathcal{M}[\mathcal{G}]$

(a) [2 points] Consider the student BN \mathcal{G} above and its undirected moralized graph $\mathcal{M}[\mathcal{G}]$. As discussed in 7.3.2, variable elimination can be understood in terms of operations on an undirected graph (or for a BN, transformations on its undirected moralized graph). For the elimination ordering \prec of G, D, I, L , and S , show the resulting induced graph $\mathcal{I}_{\mathcal{G}, \prec}$.

(b) [18 points] Consider the following greedy algorithm for producing a good elimination ordering:

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 $X_1 \dots X_n$

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 (number of entries, not number of variables) 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 these functions dominates the others. That is, show that no function results in an algorithm that in all cases produces an ordering that is better than the others' orderings with respect to the computational cost of full variable elimination (say to compute the normalization constant for a Markov Net) under the ordering. For instance, you could show an undirected graph \mathcal{H} where an ordering produced by f_C results in less computation in variable elimination than an ordering produced by f_B , another example where f_C performs better than f_A , and another example where f_B performs better than f_C . For each case, define the undirected graph, the factors over it, and the number of values each variable can take. From your examples, argue that none of the above heuristic functions is optimal.

Estimate: 2 pages.

4. 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 $1 \dots K$ for K image segments. The resulting Markov Network \mathcal{M} is shown in Figure 3.

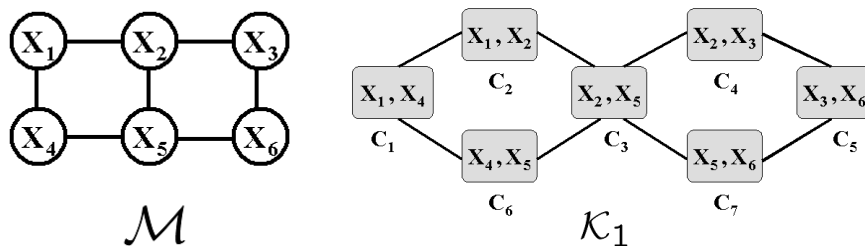


Figure 3: Markov Network for image segmentation.

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 3 also shows the cluster graph \mathcal{K}_1 that we will use for inference.

- (a) **[1 points]** Write the form of the message $\delta_{3 \rightarrow 6}$ that cluster C_3 will send to cluster C_6 during the belief update, in terms of the ϕ 's and the other messages.
- (b) **[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 $K = 2$):

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

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 $\delta_{3 \rightarrow 6}$, 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?

- (c) **[6 points]** In order to overcome this problem, we will augment our cluster graph \mathcal{K}_1 by *adding clusters* to create a new cluster graph \mathcal{K}_2 . In \mathcal{K}_2 , the factors will be assigned to the same clusters that they were in \mathcal{K}_1 , but the graph will also contain clusters over 2×2 neighborhoods of pixels. In addition to adding these clusters and any necessary edges, you may remove edges that appeared in \mathcal{K}_1 . Draw the cluster graph \mathcal{K}_2 that we will use for this round of inference. Compute the fixed-point (converged) messages that the new clusters will send to cluster C_3 , in terms of the α 's.
- (d) **[5 points]** We will now think about how such a construction will scale to larger images. We can start by trying to extend the form of cluster graph \mathcal{K}_2 over an image grid of size $N \times N$. Can we use the same architecture as described above to create a cluster graph \mathcal{K}_3 over such a grid? How many clusters would \mathcal{K}_3 contain in the case of an $N \times N$ image? If this cluster graph will work, prove it. If not, provide a counterexample and explain why loopy belief propagation cannot be applied.

Estimate: 2-3 pages.

5. Data Association and Collapsed MCMC [20 points]

Consider a data association problem with the airplane tracking application:

We have a set of K sensor measurements blips on a radar screen $\mathcal{U} = \{u_1, \dots, u_K\}$ and another set of M airplanes $\mathcal{V} = \{v_1, \dots, v_M\}$, and we wish to map \mathcal{U} 's to \mathcal{V} 's. We introduce a set of correspondence variables $\mathcal{C} = \{C_1, \dots, C_K\}$ such that $Val(C_i) = \{1, \dots, M\}$. 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 $\mathbf{B}_i = (B_{i1}, B_{i2}, \dots, B_{iL})$; for each v_j , we have a three dimensional random vector $\mathbf{A}_j = (A_{j1}, A_{j2}, A_{j3})$ corresponding to the location of the airplane v_j in the space. (Assume that $|\text{Val}(A_{jk})| = d$, which is not too large in the sense that summing over all values of \mathbf{A}_j is tractable.)

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

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

- [1 points] Briefly explain why sampling the C_i 's would be a good idea.
- [6 points] Show clearly the sampling distribution for the C_i variables and the corresponding Markov chain kernels.
- [8 points] Give a closed form equation for the distribution over the \mathbf{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.)
- [5 points] Show the equation for computing the posterior over \mathbf{A}_j based on the two steps above.

Estimate: 1.5 pages

6. 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*: $P(X^{(t+1)} | X^{(t)})$. 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 $P(X^{(t)}, \mathbf{L} | \mathbf{o}^{(1:t)})$.

- [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 .)

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.

- [15 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) [5 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