Inference and Representation, Fall 2014

Problem Set 3: Structure learning & exact inference

Due: Friday, October 10, 2014 at 5pm (as a zip file sent to pg1338@nyu.edu. Please make sure the filename is in the format `xyz`-ps3.zip, where `xyz` is your NetID.)

Important: *See problem set policy on the course web site.*

For the following questions, you may use the programming language of your choice. You are allowed to use basic graph packages (e.g., for representing and working with directed or undirected graphs), but are **not** permitted to use any graphical model or probabilistic inference packages.

The zip file should include a PDF file called "solutions.pdf" with your written solutions, separate output files (see question 2), and all of the code that you wrote.

1. The *Alarm* Bayesian network [1], shown in Figure 1, was an alarm message system for patient monitoring that was designed in 1989 as a case study for applying Bayesian networks to medical diagnosis. The Alarm Bayesian network is provided in the file "alarm.bif" (the format should be self-explanatory). Since writing code to parse this input file can be time-consuming, we also provide Python and Matlab code that you can (optionally) use to load the Bayesian network.

   This question will explore variable elimination as applied to the Alarm BN. Implement the sum-product variable elimination algorithm from class (also described in Section 20.3 of Murphy's book). Do not implement pruning of inactive variables. Use the min-fill heuristic to choose an elimination ordering. Note that you do **not** need to make this query-specific, i.e. it should be performed only once on the whole graph and then this elimination order should be used for all queries.

   (a) What is the elimination ordering found by the min-fill heuristic? How many fill edges were added?

   (b) What is the induced width of the graph with respect to the ordering found by min-fill? List the variables in the largest clique.

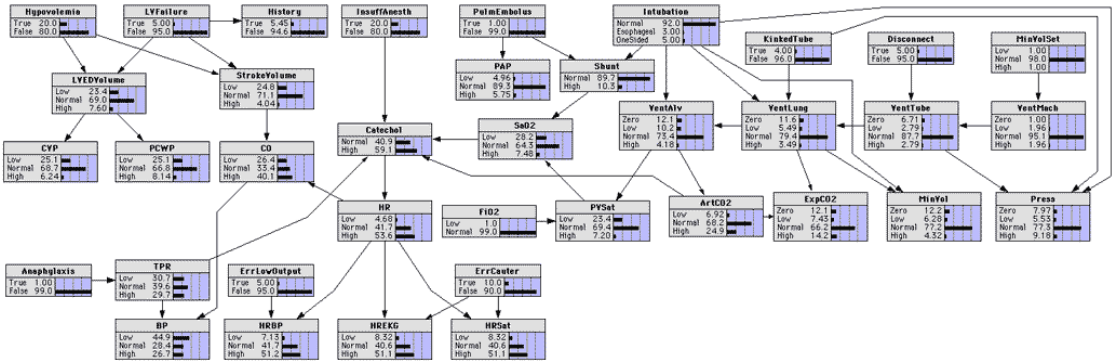

Figure 1: ALARM ("A Logical Alarm Reduction Mechanism") Bayesian network. [1]

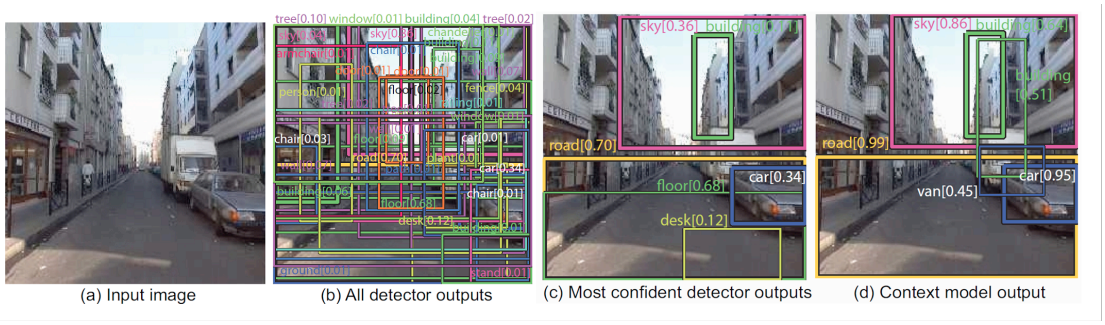

Figure 2: Using context within object detection for computer vision. [2]

Compute the value of the following queries (report up to 4 significant digits):

(c) $p(\text{StrokeVolume} = \text{High} \mid \text{Hypovolemia} = \text{True}, \text{ErrCauter} = \text{True}, \text{PVSat} = \text{Normal}, \text{Disconnect} = \text{True}, \text{MinVolSet} = \text{Low})$

(d) $p(\text{HRBP} = \text{Normal} \mid \text{LVEDVolume} = \text{Normal}, \text{Anaphylaxis} = \text{False}, \text{Press} = \text{Zero}, \text{VentTube} = \text{Zero}, \text{BP} = \text{High})$

(e) $p(\text{LVFailure} = \text{False} \mid \text{Hypovolemia} = \text{True}, \text{MinVolSet} = \text{Low}, \text{VentLung} = \text{Normal}, \text{BP} = \text{Normal})$

(f) $p(\text{PVSAT} = \text{Normal}, \text{CVP} = \text{Normal} \mid \text{LVEDVolume} = \text{High}, \text{Anaphylaxis} = \text{False}, \text{Press} = \text{Zero})$

2. When trying to do object detection from computer images, *context* can be very helpful. For example, if "car" and "road" are present in an image, then it is likely that "building" and "sky" are present as well (see Figure 2). In recent work, a tree-structured Markov random field (see Figure 3) was shown to be particularly useful for modeling the prior distribution of what objects are present in images and using this to improve object detection [2].

   In question, you will replicate some of the results from [2]. It is not necessary to read this paper to complete this assignment.[1]

[1] That said, please see `http://people.csail.mit.edu/myungjin/HContext.html` if you are curious for more details. We omit the spatial prior and the global image features, and use only the co-occurences prior and the local detector outputs (b_i, c_{ik}, and s_{ik} from [2]'s Figure 3).

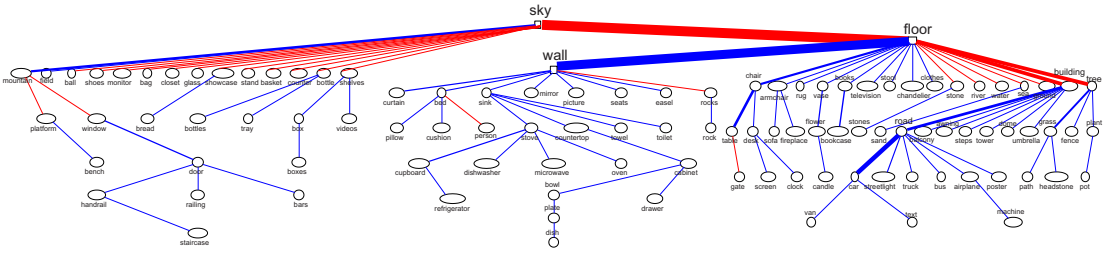

Figure 3: Pairwise MRF of object class presences in images [2]. Red edges denote negative correlations between classes. The thickness of each edge represents the strength of the link. You will be learning this MRF in question 2(a).

(a) **Structure Learning of tree-structured MRFs: the Chow-Liu Algorithm**. In this question you will implement the Chow-Liu algorithm (1968) for maximum likelihood learning of tree-structured Markov random fields [3]. See also Murphy's book Section 26.3 for a brief overview.

Let T denote the edges of a tree-structured pairwise Markov random field with vertices V. For the special case of trees, it can be shown *any* distribution $p_T(\mathbf{x})$ corresponding to a Markov random field over T admits a factorization of the form:

$$p_T(\mathbf{x}) = \prod_{(i,j)\in T} \frac{p_T(x_i, x_j)}{p_T(x_i)p_T(x_j)} \prod_{j\in V} p_T(x_j), \tag{1}$$

where $p_T(x_i, x_j)$ and $p_T(x_i)$ denote pairwise and singleton marginals of the distribution p_T, respectively.

The goal of learning is to find the tree-structured distribution $p_T(\mathbf{x})$ that maximizes the log-likelihood of the training data $\mathcal{D} = \{\mathbf{x}\}$:

$$\max_T \max_{\theta_T} \sum_{\mathbf{x}\in\mathcal{D}} \log p_T(\mathbf{x}; \theta_T).$$

We will show in a later lecture that, for a fixed structure T, the maximum likelihood parameters for a MRF will have a property called *moment matching*, meaning that the learned distribution will have marginals $p_T(x_i, x_j)$ equal to the empirical marginals $\hat{p}(x_i, x_j)$ computed from the data $\mathcal{D}$, i.e. $\hat{p}(x_i, x_j) = \mathit{count}(x_i, x_j)/|\mathcal{D}|$ where $\mathit{count}(x_i, x_j)$ is the number of data points in $\mathcal{D}$ with $X_i = x_i$ and $X_j = x_j$. Thus, using the factorization from Eq. 1, the learning task is reduced to solving

$$\max_T \sum_{\mathbf{x}\in\mathcal{D}} \log \left[\prod_{(i,j)\in T} \frac{\hat{p}(x_i, x_j)}{\hat{p}(x_i)\hat{p}(x_j)} \prod_{j\in V} \hat{p}(x_j) \right].$$

We can simplify the quantity being maximized over T as follows (let $N = |\mathcal{D}|$):

$$\begin{aligned}
&= \sum_{\mathbf{x}\in\mathcal{D}} \Big(\sum_{(i,j)\in T} \log \left[\frac{\hat{p}(x_i, x_j)}{\hat{p}(x_i)\hat{p}(x_j)} \right] + \sum_{j\in V} \log \left[\hat{p}(x_j)\right] \Big) \\
&= \sum_{(i,j)\in T} \sum_{\mathbf{x}\in\mathcal{D}} \log \left[\frac{\hat{p}(x_i, x_j)}{\hat{p}(x_i)\hat{p}(x_j)} \right] + \sum_{j\in V} \sum_{\mathbf{x}\in\mathcal{D}} \log \left[\hat{p}(x_j)\right] \\
&= \sum_{(i,j)\in T} \sum_{x_i, x_j} N\hat{p}(x_i, x_j) \log \left[\frac{\hat{p}(x_i, x_j)}{\hat{p}(x_i)\hat{p}(x_j)} \right] + \sum_{j\in V} \sum_{x_i} N\hat{p}(x_i) \log \left[\hat{p}(x_j)\right] \\
&= N\Big(\sum_{(i,j)\in T} I_{\hat{p}}(X_i, X_j) - \sum_{j\in V} H_{\hat{p}}(X_j) \Big),
\end{aligned}$$

where $I_{\hat{p}}(X_i, X_j) = \sum_{x_i, x_j} \hat{p}(x_i, x_j) \log \frac{\hat{p}(x_i, x_j)}{\hat{p}(x_i)\hat{p}(x_j)}$ is the empirical *mutual information* of variables X_i and X_j, and $H_{\hat{p}}(X_i)$ is the empirical *entropy* of variable X_i. Since the entropy terms are not a function of T, these can be ignored for the purpose of finding the maximum likelihood tree structure. **We conclude that the maximum likelihood tree can be obtained by finding the maximum-weight spanning tree in a complete graph with edge weights $I_{\hat{p}}(X_i, X_j)$ for each edge (i, j).**

Figure 4: Kitchen

Figure 5: Office

The Chow-Liu algorithm then consists of the following two steps:

i. Compute each edge weight based on the empirical mutual information.
ii. Find a maximum spanning tree (MST) via Kruskal or Prim's Algorithm.
iii. Output a pairwise MRF with edge potentials $\phi_{ij}(x_i, x_j) = \frac{\hat{p}(x_i,x_j)}{\hat{p}(x_i)\hat{p}(x_j)}$ for each $(i, j) \in T$ and node potentials $\phi_i(x_i) = \hat{p}(x_i)$.

We have one random variable $X_i \in \{0, 1\}$ for each object type (e.g., "car" or "road") specifying whether this object is present in a given image. For this problem, you are provided with a matrix of dimension $N \times M$ where $N = 4367$ is the number of images in the training set and $M = 111$ is the number of object types. This data is in the file "chowliu-input.txt", and the file "names.txt" specifies the object names corresponding to each column.

Implement the Chow-Liu algorithm described above to learn the maximum likelihood tree-structured MRF from the data provided. Your code should output the MRF in the standard UAI format described here:

`http://www.cs.huji.ac.il/project/PASCAL/fileFormat.php`

(see "kitchen.uai" and "office.uai" for two examples of files in this format).

(b) In this question, you will implement both the sum-product and max-product belief propagation algorithms for exact inference in tree-structured Markov random fields. It is OK to special-case your code for (tree-structured) binary pairwise MRFs.

You will use your algorithms to do inference in two CRFs corresponding to two images, one of a kitchen scene (see Figure 4) and the other of an office scene (see Figure 5). The two input files ("kitchen.uai" and "office.uai") are in UAI format (see above).

Each CRF describes the conditional distribution $p(b_1, \ldots, b_{111}, \mathbf{c} \mid \mathbf{s})$, where $b_i \in \{0, 1\}$ denotes the presence or absence of objects of type i in the image (the corresponding object names are given in the file "names.txt"), the variables $\mathbf{c} = \{c_{ik}\}$ where $c_{ik} \in \{0, 1\}$ specify whether location k in the image contains object i, and $\mathbf{s}$ are features of the image. The evidence (i.e. the $\mathbf{s}$ variables) is already subsumed into the edge and node potentials, and so only the $\mathbf{b}$ and $\mathbf{c}$ variables are represented in the provided MRFs (that is, you can treat this as a regular MRF).

You can sanity check your implementation of sum-product belief propagation by running it on the UAI file that you output in part (a). The single-node marginals should be precisely the same as the empirical marginals $\hat{p}(x_i)$ that you computed from the data. Note that this corresponds to the *context prior*, $p(b_1, \ldots, b_{111})$, which makes up only part of the CRF used for object recognition.

For each of the below questions, report the answer only for variables 1 through 111 (the object presence variables $\mathbf{b}$), and use the names (e.g., "wall") that are provided.

i. For each of the two images, what is the MAP assignment, i.e. $\arg\max_{\mathbf{b},\mathbf{c}} p(\mathbf{b}, \mathbf{c} \mid \mathbf{s})$? Just report which objects are present in the image (i.e., names(i) for which $b_i = 1$) according to the MAP assignment.

ii. Use the sum-product algorithm to compute the single-node marginals. For each of the two images, what objects are present with probability greater than 0.8 (i.e., names(i) for which $p(b_i = 1 \mid \mathbf{s}) \geq 0.8$)?

iii. For the two images, what objects are present with probability greater than 0.6?

Note: It is recommended to implement both sum-product and max-product in log-space, to prevent numerical overflow. See Section 3.5.3 of Murphy's book for the `log-sum-exp` trick which you will need to use to prevent numerical underflow. Also, the total running time of your algorithm should be $O(N)$, where N is the number of variables, *not* $O(N^2)$ which is what it would be had you ran variable elimination once for each variable.

References

[1] I.A. Beinlich, H.J. Suermondt, R.M. Chavez, and G.F. Cooper. The alarm monitoring system: a case study with two probabilistic inference techniques for belief networks. Technical Report KSL-88-84, Stanford University. Computer Science Dept., Knowledge Systems Laboratory, 1989.

[2] Myung Jin Choi, Joseph J. Lim, Antonio Torralba, and Alan S. Willsky. Exploiting hierarchical context on a large database of object categories. In *IEEE Conference on Computer VIsion and Pattern Recognition (CVPR)*, 2010.

[3] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.