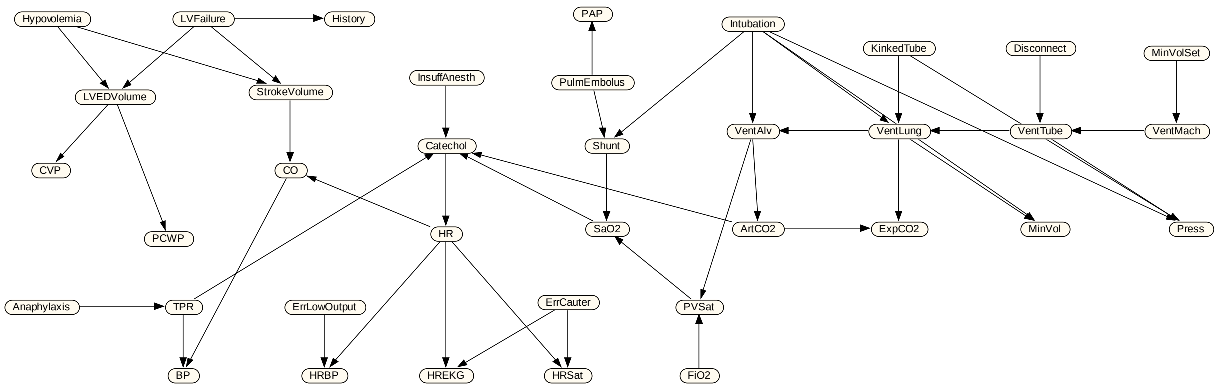
COMS 6998-01 Advanced Machine Learning
Assignment 3
March 14, 2002
Prof. Tony Jebara

The assignment is due on **April 1st 2002, before midnight** either in my office CEPSR 605 or via email to jebara@cs.columbia.edu. If you email me the assignment, please use standard formats, i.e. send me plain text, latex, postscript, pdf or word.

1. Moralization and Undirected Graph Cliques

The directed graph below is of the ALARM network, which stands for 'A Logical Alarm Reduction Mechanism'. This is used in medical diagnostics for patient monitoring in emergency room situations. All nodes are discrete, but can have more than 2 states (not just binary). For example, PAP is Pulmonary Artery Pressure and is split into Low, Normal and High (i.e. 3 states: 0,1,2). Many continuous nodes often get discretized in this way because this generates simpler algorithms and probability distributions than dealing directly with continuous and discrete nodes mixed in a single graph.



As usual, the graph below represents a distribution in terms of each node and its parents:

$$p(X) = \prod_i p\left(X_i \mid X_{\pi_i}\right)$$

Convert the above directed graph into its corresponding undirected graph using moralization and draw the resulting moral graph.

Find all the maximal cliques in the undirected graph and trace them out over the moral graph (or write them out as sets). These then specify the joint distribution as a product of potential functions on undirected cliques (i.e. write out the specific sets of nodes in each potential function).

$$p(X) = \frac{1}{Z} \prod_c \Psi(X_c)$$

2. Directed Graph Conditional Probability Tables Maximum Likelihood Parameter Learning

We want to estimate the numerical values of the probability tables in graphs such as the one above by looking at real data (i.e. from real patients and what happened in previous emergency room situations). The maximum likelihood estimate for each conditional probability function is as follows (as in class and in the text in Chapter 8).

Where the m() function counts the number of times a particular configuration of the node and its parents appeared in the training data.

$$p\left(X_i \mid X_{\pi_i}\right) = \frac{m\left(X_i, X_{\pi_i}\right)}{\sum_{X_i} m\left(X_i, X_{\pi_i}\right)}$$

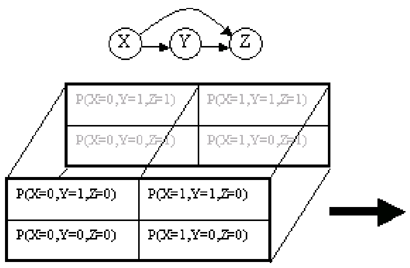
Implement in Matlab the above recipe for maximum likelihood learning from the counts for an arbitrary directed graph (acyclic). Assume that the nodes and their parents appear in topological order and the parent-child relationships are specified by a simple matrix. Also, assume that the nodes can handle an arbitrary number of discrete states. However, in this assignment, we will only deal with binary states for now. Your algorithm should take IID samples of the graph and generate the entries of the tables (make sure that they are positive, normalized, etc.)

To store the tables, a matrix format will be used (Matlab is not the most flexible language for datastructures, unfortunately). Each row corresponds to the conditional probability table (cpt) for a single node:

$$p\left(X_i \mid X_{\pi_i}\right)$$

That row of numbers is a long vector of the cpt hypercube rasterized where the earlier parents (i.e. lower number) increase slowest while the largest number parents increase fastest and the node's.

To understand what the above means, consider the following toy example over 3 binary variables. The directed graph indicates we need to store P(X), P(Y|X) and P(Z|X,Y). There are 3 cpt's therefore, and each can be represented by a row of a matrix as follows (below we show the case for P(Z|X,Y). The P(Z|X,Y) is shown as a probability cube is then expanded into a table that enumerates all configurations and then finally we read off the vector of a cpt for p(Z|X,Y) at the bottom from the table:



X	Y	Z	P(Z|X,Y)
0	0	0	P(Z=0|X=0,Y=0)
0	0	1	P(Z=1|X=0,Y=0)
0	1	0	P(Z=0|X=0,Y=1)
0	1	1	P(Z=1|X=0,Y=1)
1	0	0	P(Z=0|X=1,Y=0)
1	0	1	P(Z=1|X=1,Y=0)
1	1	0	P(Z=0|X=1,Y=1)
1	1	1	P(Z=1|X=1,Y=1)

P(Z=0|X=0,Y=0)	P(Z=1|X=0,Y=0)	P(Z=0|X=0,Y=1)	P(Z=1|X=0,Y=1)	P(Z=0|X=1,Y=0)	P(Z=1|X=1,Y=0)	P(Z=0|X=1,Y=1)	P(Z=1|X=1,Y=1)

Since each cpt will be different in size for each node (i.e. each has different number of parents and cardinality of parent states and self-states), the sizes of each of the rows of the matrix will be different for each row number (or node number). For example, the row above has 8 entries and would require a matrix of 8 columns to contain it. But the graph for X has no parents and would require only a 2-element row P(X=0) and P(X=1). To contain them all, the big matrix of all cpts should have as many columns as required for the biggest cpt and the number of rows is the number of nodes in the graph. You can pad the big matrix with –1.0 values at the empty locations.

Three pieces of code are given to get you started. The Matlab function likeNet.m computes the log-likelihood of a directed graph, the function sampleNet.m gives out IID data samples from an existing graph structure and its numerical tables, and the function learnNet.m actually learns the graph from the IID samples and an existing graph structure. The last function, learnNet.m is what you need to complete. You will need to write only on the order of 30 lines of Matlab code to complete it. Study the code that is given to get more details of the datastructures work and how to implement directed graphs so that you can easily write the learnNet.m function from there.

Test your algorithm on the following topological graph:

$$\begin{tikzcd}
1 \arrow[r] \arrow[rr, bend left] & 2 \arrow[r] \arrow[rr, bend left] & 3 \arrow[r] \arrow[rr, bend left] & 4 \arrow[r] & 5
\end{tikzcd}$$

The parent-child relationship is captured with the following matrix (row number is the child node and column number is the parent number, note diagonals are zero since children can't be their own parents):

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0
\end{bmatrix}$$

The data set to train on is dataset5.txt which contains 120 datapoints represented as 5-element row vectors with binary (0,1) values. The first column is node 1's values, and the last one is node 5's values.

Hand in the resulting cpts you estimate in the format specified by the above Matlab functions. Hand in your learnNet.m function and also hand in the log-likelihood you obtained with your estimates. Provide a brief discussion of your results and code.

3. Cross-Validation for Graph Structure Learning

Here, you will use the learnNet.m function you have built to hunt for the best graph structure by learning different ones and seeing which gets the best likelihood on test data. Train your graphs on dataset6.txt with your learnNet.m function and test them on dataset7.txt (each is 120 points). Try the following graphs and report their log-likelihoods on training and testing. The nodes are again all binary. By finding the graph with the highest log-likelihood test score, you will isolate the best structure and that is the graph that was originally used to create dataset6.txt and dataset7.txt.

Graph 1:

$$\begin{tikzcd}
1 & 2 & 3 & 4 & 5
\end{tikzcd}$$

Graph 2:

$$\begin{tikzcd}
1 \arrow[r] & 2 \arrow[r] & 3 \arrow[r] & 4 \arrow[r] & 5
\end{tikzcd}$$

Graph 3:

$$\begin{tikzcd}
1 \arrow[r] \arrow[rrrr, bend left] & 2 \arrow[r] & 3 \arrow[r] & 4 \arrow[r] & 5
\end{tikzcd}$$

Graph 4:

$$\begin{tikzcd}
1 \arrow[r] \arrow[rrrr, bend left] & 2 \arrow[r] \arrow[rrr, bend right] & 3 \arrow[r] & 4 \arrow[r] & 5
\end{tikzcd}$$

Show your parents-child matrix for each of the above graphs. Discuss the relationship between log-likelihood on testing and on training with the number of free parameters in the cpt's. You are free to try out different graphs as well if you like (just remember that for the above code pieces, the parents-children matrix has to remain topological, i.e. only the lower triangle can be non-zero).