CS535c Fall 2004: Homework 3

Out Wed 29 Sep, Due Wed 6 Oct

In this assignment, you will implement the variable elimination algorithm discussed in section 7.2 of the book. Download the file HW3.zip from the class homework page, and unzip it. It contains several groups of files:

- You will find a bunch of files with names like factorXXX; these are methods for manipulating tabular factors, as discussed in section 7.2.1 of the book (so you no longer have to worry about repmat and reshape, since I've done it all for you). Factors are implemented as Matlab structures. Look at the function `factorExample.m` to learn how to use them.
- You will find `naiveInfer.m`, which computes $P(X_q|e)$ using naive (brute-force) inference, i.e., it explicitly constructs the joint distribution (represented as a multi-dimensional array), and then marginalizes it down to X_q (the query nodes), and normalizes the result.[1] This is a generalization of the method we have used in homeworks 1 and 2. Note that X_q might be a *set* of variables (possibly empty), so the result is returned as a factor. I have modified the function `hw1-sprinkler.m` online to give an example of how to compute joint distributions over sets of variables (specifically, $P(S, R|W = 2)$ and $P(C, W|S = 2)$).
- You will find `hw3-test-sprinkler.m` and `hw3-test-mnet.m`. These functions call naiveInfer and a function that you must write, called `varElim`, with exactly the same arguments; it then checks that both methods give the same results. (In addition, for the sprinkler case, it checks that the results match those of homework 1.)
- Finally, there are some utility functions: assert, approxeq, ind2subv, subv2ind, and a new one, match.

Your task is to implement the function `varElim.m`. It should have exactly the same input-output behavior as `naiveInfer.m`, but it should be much more efficient. You can pick any elimination ordering you want. We will study ways to find efficient orderings next week.

You should check that your function is correct yourself by running the 2 test scripts mentioned above.

When you are confident your program is working, send it by email to the TA, Mark Crowley (`crowley@cs.ubc.ca`). You should just send one file; if you want to use subfunctions, embed them in the same .m file.

Mark will run the 2 test scripts mentioned above (and possibly some other tests) using your varElim function. Your grade will be proportional to how many of the tests your program passes.

In the event that your program does not get all the numbers correct, Mark will award credit based on the quality of your code. If your code can convince Mark that you have almost implemented the algorithm correctly, you are likely to get more points. Hence you are recommended to write clear code. For hints on good Matlab programming style, see the paper at `http://www.datatool.com/prod02.htm`.

Note: The way I handle evidence is to multiply in extra factors, one per observed node, containing the local evidence. If these local evidences are delta functions (0/1 distributions), then this is equivalent to instantiating the observed nodes to their observed values.

Hint: you may find it helpful to read about the elim-bel algorithm on p10 of Rina Dechter's "Bucket elimination" paper on the class web page.

[1] The normalization constant is called alpha. For a Bayes net with no evidence, $\alpha = 1$; For a Markov net with no evidence, $\alpha = Z$, the partition function. For a Bayes net with evidence, $\alpha = P(e)$, the probability of the evidence.

Hint: You may find the following code fragment helpful (this is the start of my function):

```
function [F, alpha] = varElim(factors, query, elimOrder)
% varElim Variable elimination algorithm
% function [F, alpha] = varElim(factors, query, order)

% Infer number and sizes of variables from the factors
NF = length(factors);
scope = [];
for f=1:NF
  scope = union(scope, factors{f}.scope);
end
N = length(scope);
if ~isequal(scope, 1:N)
  error('variables should be ordered 1,2,...')
end
sizes = ones(1,N);
for f=1:NF
  sizes(match(factors{f}.scope, scope)) = factors{f}.sizes;
end
```