School of Computer Science
The University of Adelaide

Artificial Intelligence
Assignment 3

Semester 1, 2013
due 11.59, Friday 14 June 2013

---

1 Probabilistic graphical models

Your task is to write a program to implement *likelihood weighted sampling*, as described in lectures, to perform inference on an arbitrary probabilistic graphical model (PGM) of boolean random variables. You will need to parse an input text file that encodes the graph and the conditional probability distributions for each variable (conditioned on its parents). The format of the file is given in section 2 below. Two networks have been defined in this format and can be downloaded, along with two queries each, at:

`https://cs.adelaide.edu.au/users/third/ai/13s1-ai-adelaide/files/ass3.zip`

Your program must be able to parse any file of this format to create and populate an internal data structure of the PGM.

Your program must then prompt the user via the console (or read from redirected standard input) for a single variable query, parse the query correctly and evaluate conditional probability distribution of the query variable, given the evidence. The result must be written as two values `P(QueryVar=`*true*`|...)` and `P(QueryVar=`*false*`|...)` onto the standard output stream, separated by white space.

2 File format

The graphical model will be specified by a text file with format:

```
N

rv0 rv1 ... rvN-1

0 0 1 ... 0
1 0 0 ... 1
...
0 1 1 ... 0

mat0

mat1

...

matN-1
```

Here:

- `N` is the number of random variables in the network;
- `rv` are the random variable names (arbitrary alphanumeric strings);
- `mat` are two dimensional arrays of real numbers (in ASCII) that specify the conditional probability table of each random variable conditioned on its parents;
- The matrix of zeros and ones specifes the directed arcs in the graph; a one (`1`) in the `i,j` entry indicates there is an edge from `i` to `j` (so the ith variable is a parent of the jth variable.

The format of the matrices is a bit subtle. If a node has m parents, then the matrix needs to specify the probability of each outcome (*true, false*) conditioned on 2^m different combinations of parent values, so the matrix will be $2^m \times 2$ (rows × columns). Treating *true* as 1, and *false* as 0, concatenate the values of the parents in their numerical order from most significant bit to least significant bit (left to right) to create a row index r. The entry in the first column, rth row is then the probability that the variable is *true* given the values of the other variables (the entry in the corresponding 2nd column is the probability that the variable is *false*).

For example if A has parents C and F where C is the 3rd variable specified and F is the 6th, then the (2,0) entry represents `P(A=`*`true`*`|C=`*`true`*`,F=`*`false`*`)`, and the (0,1) entry represents `P(A=`*`false`*`|C=`*`false`*`,F=`*`false`*`)`.

The format of the query, entered via the console (or read from redirected standard input) is:

```
P(rvQ | rvE1=val, rvE2=val, ...)
```

where `rvQ` is the name of the query variable, and `rvEx` are the names of the evidence variables with their respective *true*/*false* values specified. As is normal in Bayesian inference, variables not included are unobserved and should be marginalised out.

3 Deliverables

Write your program in Java or C/C++. In the case of Java, name your program `inference.java`. Your program must be able to be compiled and run as follows:

```
$ javac inference.java
$ java inference graphfile.txt
```

In the case of C/C++, you must supply a makefile `Makefile` with a rule called inference to compile your program into a Linux executable binary named `inference.bin`. Your program must be able to be compiled and run as follows:

```
$ make inference
$ ./inference.bin graphfile.txt
```

NOTE1: If a makfile exists, the marking script will assume that the program is written in C/C++, otherwise Java will be assumed.

NOTE2: `graphfile.txt` is of course a placeholder for the name of the file containing the PGM structure.

4 Submission and assessment

You must submit your program on the Computer Science Web Submission System. This means you must create the assignment under your own SVN repository to store the submission files. The SVN key for this submission is `2013/s1/ai/Assignment3`. The link to the Web Submission System used for this assignment is `https://cs.adelaide.edu.au/for/students/automark/`

DO NOT use the link: `https://cs.adelaide.edu.au/services/websubmission/`

For more details on the online submission procedures including SVN visit the home page of the school and look for SVN information under the "For Students" tab. Your code will be compiled and run, testing its outputs for the two networks with two queries each that have been provided to you, and an additional three networks with two queries each. None of the networks will contain more than 16 variables. Since the results are stochastic, the query answers will not be exact and will vary from run to run. The asnwers will therefore be tested against a tolerance of ±0.01 (i.e. your answers must be within 1% of the true values), so you must ensure convergence to this level of precision. If it passes all tests you will get 15% (undergrads) or 12% (postgrads, DEFSCI) of the overall course mark. The objective of the tests is to check for the correct operation of your implementation. Hence, the basis of the assessment is to compare your results against the expected results. You must also ensure that you have an efficient implementation.

5 Using other source code

You may not use other source code for this assignment. You should personally and carefully implement the likelihood weighted sampling to fully understand the concept.

6 Due date and late submission policy

This assignment is due 11.59pm on Friday 14th June 2013. If your submission is late the maximum mark you can obtain will be reduced by 25% per day (or part thereof) past the due date or any extension you are granted.

Prof. Ian Reid, 22 May 2013