School of Computer Science
The University of Adelaide

Artificial Intelligence
Assignment 3

Semester 1, 2019
due 11.59, Monday 10 June 2019

Probabilistic graphical models

Your task is to perform inference on a probabilistic graphical model (PGM) of boolean (i.e. true/false) random variables.

There are two options available for this assignment, either:

Option 1 Implement code in Java/C/C++/Python to perform approximate inference using one of the methods described in lectures (i.e. one from: rejection sampling; likelihood weighted sampling; or Gibbs sampling). If you choose Option 1, you will need to use the files provided that describe two Bayes Networks and a set of probabilistic queries to solve. Your program must read/parse these files, create the Bayes network, and compute values for each of the queries. You should submit your code and a 2-3 page report of your findings.

Option 2 Perform exact inference on a PGM and also manually conduct approximate inference and compare the results. If you choose Option 2 you only need to deal with one (small) network, but you will need to compute the exact inference and also compute approximate inference manually (i.e. generate random numbers, use these to generate samples, count the samples, etc). You should submit a 2-3 page report of your working and findings.

More details about each option appear below.

Option 1

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

below. Your program must be able to parse any file of this format to create and populate an internal data structure of the PGM. Two networks have been defined in this format and can be downloaded from the Assignment 3 page in the MyUni course pages.

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 the conditional probability distribution of the query variable, given the evidence. The result must be written as two decimal values (corresponding to the values of `P(QueryVar=`*true*`|...)` and `P(QueryVar=`*false*`|...)`) onto the standard output stream. For example:

```
0.872  0.128
```

Write your program in Java, C/C++ or Python. You may not use other source code for this assignment, but you can make use of any libraries you like for reading and writing the files and parsing the input. The sampling and inference procedure must be your own code.

Submission and Assessment for Option 1

For this Option you should submit your code and a pdf report of maximum 2-3 pages via MyUni. The report should briefly describe the algorithm you have chosen to implement, then provide a section with results. For each query, indicate the results obtained (i.e. the conditional probability values for the query) using number of samples N, where N will take values 10, 20, 50 and 100, set out in your report as in the table below.

	n=10	n=20	n=50	n=100
Query 1				
Query 2				
...				
Query M				

Marks will be allocated according to the following rubric:

- Code accuracy: 30 marks
- Report coherence: 30 marks
- Results: 40 marks

There are no marks for coding style per se, but if you code is poorly set out and/or unintelligible (so that its accuracy of implementation is hard to assess) expect to be penalized. Full marks will be awarded for clear, intelligible code, and the completeness and correctness

of the results as reported in the final submission. If you present results in your report that have been generated in some way other than directly from your code, you *must* acknowledge this and in the report set out clearly how you generated the results (e.g. exact inference). If we determine that you have submitted results that cannot be produced using your code, and you have not stated how you generated the results otherwise, you will automatically be given zero marks and referred for academic dishonesty.

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 specifies 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 Conditional Probability Table 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 $\times$ 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*). Thus, the first row of the matrix corresponds to all parent variables taking the value *false* ($r = 000\ldots0$), and the last row has all parents *true* ($r = 111\ldots1$).

For example if the variable A has parents C and F where C is the 3rd variable specified and F is the 6th, then $C, F = 00, 01, 10, 11$ corresponds to row $r = 0, 1, 2, 3$ of the table. The CPT entries for `P(A|C,F)` entries will be:

CPT entry	
P(A=*true*|C=*false*,F=*false*)	P(A=*false*|C=*false*,F=*false*)
P(A=*true*|C=*false*,F=*true*)	P(A=*false*|C=*false*,F=*true*)
P(A=*true*|C=*true*,F=*false*)	P(A=*false*|C=*true*,F=*false*)
P(A=*true*|C=*true*,F=*true*)	P(A=*false*|C=*true*,F=*true*)

For another example, check out the Bayes Net in Figure 1. This network is represented in the format above in the `studentnetwork.txt` file provided for the Assignment.

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.

Option 2

For Option 2 there is no need to code, but you will need to work through the algorithm(s) *manually* on paper. The network below in Figure 1 is a graphical representation of the smaller network used in Option 1. It shows the conditional probability relationships between variables S (Sick), P (Pub), H (Headache), L (Lecture) and D (Doctor) which capture conditional probabilities relating whether or not a student has a Headache to whether they are Sick,

whether they went to the Pub last night, whether they go to the 9am Lecture or not and whether they visit a Doctor.

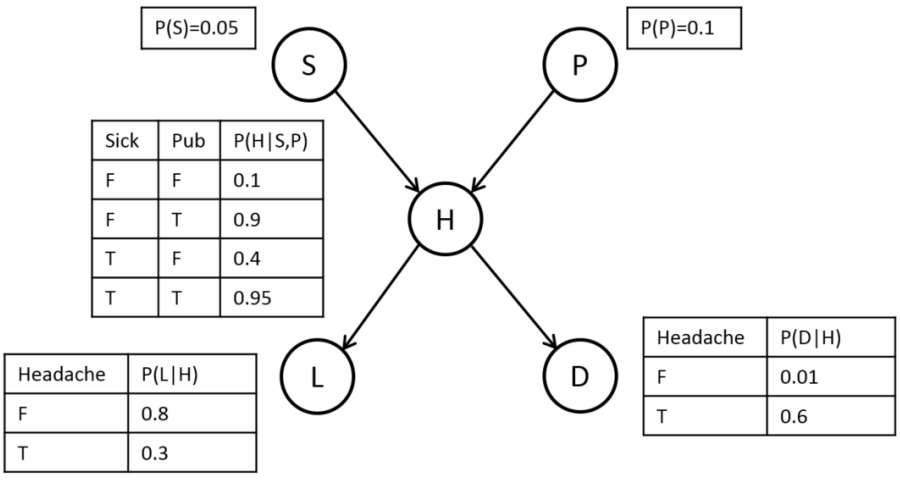

Figure 1: A Bayes Net representing the conditional relationships between illness, lecture attendance, pub and doctor visits

The queries you must solve are:

- P(Sick | Lecture=true, Doctor=true)
- P(Sick | Doctor=false)
- P(Pub | Lecture=false)
- P(Pub | Lecture=false, Doctor=true)

You must produce exact results for the first two, and both exact and approximate results (by sampling) for the third and fourth queries. You will need to conduct manual sampling by any method you choose to generate the approximate results. For the exact results: **Undergrads** can choose any exact inference method; **Postgrads** will be eligible for full marks only if the exact results are generated using variable elimination (see Assessment below).

To "implement" your manual sampling procedure you will need generate a set of random samples. For example, for the first query you will need to generate 3 random true/false values

(for Sick, Headache and Pub) for each sample.[1] You can do this by tossing a coin, but since you'll have to generate 3 values for each sample, this could get dull and time-consuming. Instead I suggest you visit a website such as https://www.random.org/integers/, generate a list of integers between 0 and 100, and treat values greater than 50 as true and less than or equal to 50 as false. For the first query, if you decide to generate 20 samples, you'll need 60 random values (3 per sample). For the second query you'll need 80 (4 per sample).

Submission and Assessment

For this Option you should submit a pdf report of maximum 2-3 pages via MyUni. The report should briefly describe the algorithm you have chosen to implement, provide your exact working for each query, and a table showing the results for each. I recommend you tabulate results for 10 samples and for 20 samples and compare these to the exact results.

Marks will be allocated according to the following rubric:

- Exact results and corresponding working: 30 marks
- Approximate inference results (with working): 40 marks
- Report coherence including description of approximate inference algorithm used: 30 marks

As noted above, **Postgraduates** are expected to use variable elimination to generate their exact results. If a postgrad chooses not to use variable elimination they can achieve up to 15/30 on that component. Full marks will be awarded for complete and correct results, and a coherent description of the sampling method used. If you present results in your report that are inconsistent with the sampling method you have attempted, and you do not acknowledge this in your report, you will automatically be given zero marks for the Assignment and referred for academic dishonesty.

Due date and late submission policy

This assignment is due 11.55pm on Monday 10th June 2019. 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.

[1] If you choose rejection sampling you'll need to generate true/false values for all five random variables, and then reject any sample that does not match Lecture=false and Doctor=true