School of Computer Science
The University of Adelaide

Artificial Intelligence
Assignment 3

Semester 1, 2021

Due 11.59 pm, Friday 11 June 2021

1. Introduction

This assignment has two parts: Probability Graph Model (Section 2) and Part-of-Speech Tagging (Section 3). Both UG and PG students need to finish both parts. In part one, you are expected to perform approximate inference on a given PGM using C++/Java/Python and **submit the codes to the Web Submission system**. In part two, you are expected to provide your exact working for the required computation in a pdf and **submit the pdf in MyUni**, Assignment 3.

2. Probability Graph Model (10 marks)

Your task is to perform approximate inference on a probabilistic graphical model (PGM) of Boolean random variables using ***likelihood weighted sampling***, as described in lecture. You will work on two networks (Burglar and Fire). The networks and example queries are provided in the textual files attached on MyUni and will be described in the later section.

2.1 The task

You need to write a program (C++/Java/Python) to parse the input file of network and to create and populate an internal data structure of the PGM. The program also need to parse the input query file 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, separated by white space. For example:

```
0.872 0.128
```

Note that if you write anything else before these two numbers, automark will interpret that output as the answer, almost certainly resulting in a test failure.

2.2 File Format

The format of the network file is:

```
N

rv0 rv1 ... rvN-1
0 0 1 ... 0
1 0 0 ... 1
...
0 1 1 ... 0

mat0

mat1
...
matN-1
```

- N is the number of random variables in the network;
- rv* are the random variable names (arbitrary alphanumeric strings);
- The matrix of zeros and ones specifies the directed arcs in the graph; a ‘1’ in the (*i, j)* entry indicates there is an edge from *i* to *j*, so the *i*-th variable is a parent of the *j*-th variable.
- mat are two dimensional arrays of real numbers (in ASCII) that specify the conditional probability table of each random variable conditioned on its parents; 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 x 2 (rows x 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, r-th 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 conditioning variables taken the value *false* (r = 000… 0), and the last row has all conditioning variables *true* (r = 111 …1).
  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 correspnd 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*)

The format of the query file 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.

2.3 Deliverables

Implement random forest for wine quality prediction in either C/C++, Java or Python.

C++. 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 named inference.bin. Your program must be able to be compiled and run as follows:

```
$ make inference
$ ./inference.bin [graphfile] [queryfile]
```

JAVA. In the case of JAVA, you must write your program in the file inference.java. Your program must be able to be compiled and run as follows:

```
$ javac inference.java
$ java inference [graphfile] [queryfile]
```

Python. In the case of Python, write you program in the file inference.py. Your program must be able to be run as follows:

```
$ python inference.py [graphfile] [queryfile]
```

At the moment, only an older version of Python (version 2.7.5) is supported on the school servers. If you are not familiar enough with this version of Python, PLEASE DO NOT USE PYTHON for the assignment.

1.3 Expected run time

Your program must be able to terminate within 1 minutes on the server.

1.4 Web Submission Instructions

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
2021/s1/ai/assignment3
The link to the Web Submission System used for this assignment is:
https://cs.adelaide.edu.au/services/websubmission/

After submit your codes through SVN, navigate to 2021, Semester 1, Artificial Intelligence, Assignment 3 (for COMP SCI 3007 7059 students). Then, click Tab "Make Submission" for this assignment and indicate that you agree to the declaration. The automark script will then evaluate your codes.

1.5 Assessment

Your code will be compiled and run, testing its outputs for the two networks with several queries each. Two example queries have been provided to you. Since the results are stochastic, the query answers will not be exact and will vary from run to run. The answers will therefore be tested against a tolerance of ± 1 (i.e. your answers must be within 1% of the true values), so you must ensure convergence to this level of precision (by giving more samples). If it passes all tests you will get 10% 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.

Although it is an auto-assessment, I will randomly pick up some codes to check manually.

1.6 Due date and late submission policy

This assignment is due by **11:59pm Friday 11 June**. 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.

3. Part-of-Speech Tagging – Viterbi (5 marks)

In this part, you will do POS tagging via Viterbi algorithm for the sentence *Janet will back the bill*. This is the example in our lecture and also in the textbook here (https://web.stanford.edu/~jurafsky/slp3/8.pdf).

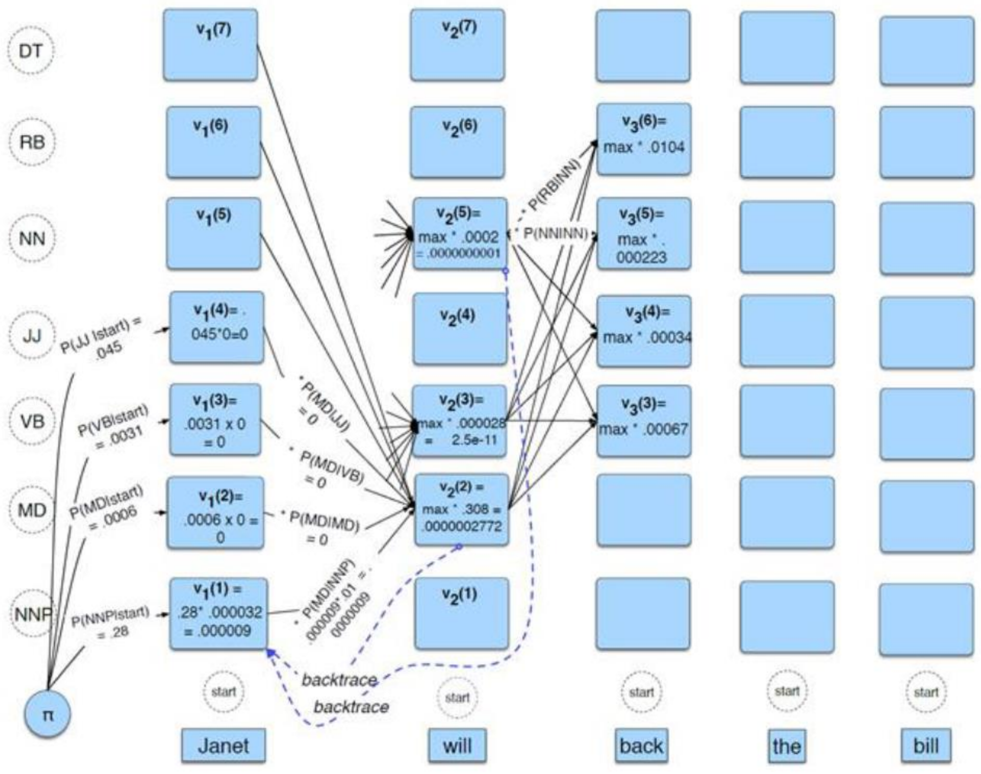
The transition probabilities computed from the WSJ corpus without smoothing are in the table below. Rows are labelled with the conditioning event; e.g., P(VB |MD) is 0.7968.

	NNP	**MD**	**VB**	**JJ**	**NN**	**RB**	**DT**
$<s>$	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

The Observation likelihoods computed from the WSJ corpus without smoothing are in the following table:

	Janet	**will**	**back**	**the**	**bill**
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

The computation for the three columns are shown in the following figure:



3.1 The task

In this task, you need to calculate **$v_3(3)$**. Note that the computation of $v_1(1)$ to $v_1(7)$ and $v_2(2)$ are provided in the lecture slides and the textbook. To compute $v_3(3)$ you first need to compute $v_2(1)$, $v_2(3)$, $v_2(4)$,$v_2(5)$, $v_2(6)$, $v_2(7)$.

For PG students, you need to **further** calculate **$v_4(7)$**. Note that to compute $v_4(7)$, you need to know $v_3(1)$ to $v_3(7)$.

3.2 Deliverable

You will provide your exact working for the computation of $v_2(1)$, $v_2(3)$, $v_2(4)$,$v_2(5)$, $v_2(6)$, $v_2(7)$ and $v_3(3)$ and the final tag sequence into a report in pdf format. You need to write **the detailed process** and the results. Using equations provided by your text editor. **Hand-written equations (e.g., in image format) will not be marked**.

Please submit the pdf to **MyUni, Assignment 3** (https://myuni.adelaide.edu.au/courses/64835/assignments/220569).

3.3 Assessment

Use correct equation, correct value to compute, and get correct results.

UG: 5 marks for $v_3(3)$, include intermediate results

PG: 2 marks for $v_3(3)$, 3 marks for $v_4(7)$, include intermediate results.

3.4 Due date and late submission policy

This assignment is due by **11:59pm Friday 11 June**. 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.