University of Leeds – School of Computing

COMP2611 **COURSEWORK 2** (10% of module)

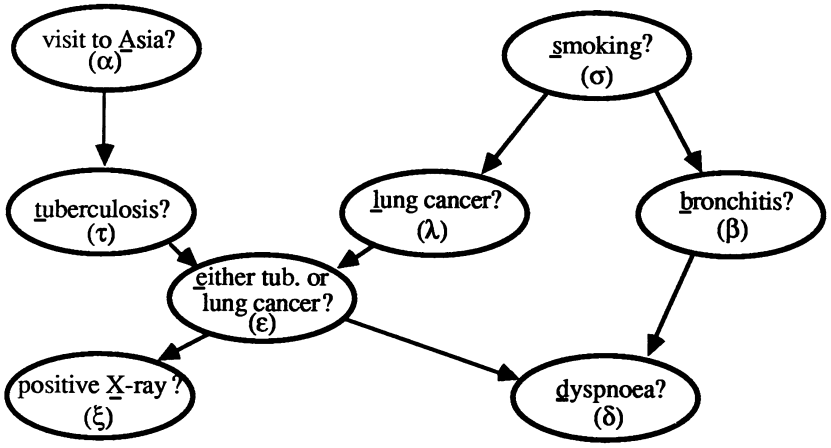
Coursework issued:	Friday 22th March 2019
Submission deadline:	Monday 15th April 2019

To be submitted via Minerva (the VLE)

General Overview

In this coursework, you will be developing a Python program to implement the fictitious 'Asia' network that models the following qualitative medical 'Knowledge':

"Shortness-of-breath (dyspnoea) may be due to tuberculosis, lung cancer or bronchitis, or none of them, or more than one of them. A recent visit to Asia increases the chances of tuberculosis, while smoking is known to be a risk factor for both lung cancer and bronchitis. The results of a single chest X-ray do not discriminate between lung cancer and tuberculosis, as neither does the presence or absence of dyspnoea."



You will be helping clinicians to decide the probability of each disease given some evidence. The CPD values for this model, made of binary variables, are given in the Table below, where any binary variable denoted as x is equivalent to saying $x = 1$, furthermore, $\overline{x} \equiv \neg x \equiv (x = 0)$.

$p(asia) = 0.01$	$p(smoke) = 0.5$
$p(tub\|asia) = 0.05$	$p(lung\|smoke) = 0.1$
$p(tub\|\overline{asia}) = 0.01$	$p(lung\|\overline{smoke}) = 0.01$
$p(bron\|smoke) = 0.6$	$p(xray\|either) = 0.98$
$p(bron\|\overline{smoke}) = 0.3$	$p(xray\|\overline{either}) = 0.05$
$p(either\|lung, tub) = 0.999$	$p(dysp\|either, bron) = 0.9$
$p(either\|lung, \overline{tub}) = 0.999$	$p(dysp\|either, \overline{bron}) = 0.7$
$p(either\|\overline{lung}, tub) = 0.999$	$p(dysp\|\overline{either}, bron) = 0.8$
$p(either\|\overline{llung}, \overline{tub}) = 0.001$	$p(dysp\|\overline{either}, \overline{bron}) = 0.1$

The Problem

In the lectures, we have seen how to use *pgmpy* (`http://pgmpy.org`) to implement Bayesian networks, define their nodes, edges (showing directions of dependencies), CPDs, and given some evidence, compute the posterior probabilities of the query variables. Using these skills, write a Python program, `asia.py` that:

- Constructs the Asia network using variable names:

  `asia, smoke, tub, bron, lung, either, dysp, xray`

  and the parameters for the CPDs shown above (Avoid fetching the network from `http://www.bnlearn.com/bnrepository/asia/asia.bif.gz` as the values for CPDs may be different).

- Takes evidence variables, computes, prints the exact (using `VariableEliminiation` pgmpy class) and approximate posterior probabilities (using the provided `GibbsSampling` class) for each of the query variables. The program should accept various input options using the following switches:

  `--evidence` A list of space separated variable:value pairs used as evidence

  `--query` A list of space separated query variables

  `--exact` Used to perform variable elimination

  `--gibbs` Used to perform Gibbs sampling

  `-N` Number of samples generated using Gibbs sampling

  `--ent` Used to compute the cross-entropy between the exact and approximate distributions

  Refer to the `test_asia.py` to see examples of runs that your program should pass.

- Computes the total cross-entropy between the approximated (q) and exact (p) posterior probability distributions for all query variables using the following equation:

$$\mathrm{H(p,q)} = -\sum_{x \in \{\text{query set}\}} [q(\overline{x}) \log p(\overline{x}) + q(x) \log p(x)]$$

The cross entropy indicates how good our approximated posterior is: the smaller cross entropy, the closer approximation to the exact distribution. This is a useful measure to ensure that your Gibbs sampling works fine, as longer Markov chains should converge to exact posterior distribution (according to the law of large numbers). This means that the larger `N`, the smaller cross entropy should be obtained.

Software dependencies

Refer to Documentation in `pgmpy` website and examples thereafter to study how you could implement your own code. In particular, have a look at: `http://pgmpy.org/_modules/pgmpy/sampling/Sampling.html` to see how you can generate samples from a Bayesian model for approximate inference according to MCMC method (see Lecture 10 for the principle of computing the posteriors). Note that the `GibbsSampling.py` class implemented by `pgmpy` does not support sampling with given evidence.

You will be provided with a python file, `GibbsSamplingWithEvidence.py`, that performs Gibbs sampling with a given evidence. You will need to copy this file into your local directory and import `GibbsSampling` class in `asia.py` from the provided code to enable sampling. Note that the samples from this class are not obtained subsequently right after each other, as more than one variable may change states between two subsequent samples. However, this does not make a difference here and it serves the purpose for computing the approximate posteriors.

This coursework has been designed using Python 3 (as the official support for Python 2.7 is ending soon) which should be installed in your own computer before installing `pgmpy` and developing your code. We have also requested IT department to install `pgmpy` in DEC10 Lab room and we will be testing your coursework under Python 3 versions.

You hand in two Python files and nothing else

- A file called `asia.py`, as described above.
- A file called `test_asia.py` the test file that calls `asia.py` with certain configurations of input options (see the test file for examples). This file will be provided but you can comment out the lines that your program fails to pass with no error in less than 60 seconds. We will consider longer run times (per line) as unreasonable but your code will not be marked on how fast the tests run provided that it is not unreasonable.
- Do not submit the provided `GibbsSamplingWithEvidence.py` as a part of your coursework; you should not change this file.
- Submission is via Minerva (the VLE); no submission using email will be accepted.

Marking scheme (total available: 100)

- Code prints brief auxiliary information to specify which output is printed (5)

- Code runs and produces correct solutions for exact inference using the variable elimination technique (2×15)
- Code runs and makes reasonable approximate posterior probabilities for query variables using Gibbs sampling (2×20)
- Cross entropy is computed and printed out (15)
- Cross entropy is shown to be smaller for larger number of samples (5)
- To get full marks there should be a few appropriate comments in the code but not excessively long (5)