CS5340 ASSIGNMENT 4: MONTE CARLO INFERENCE

1. OVERVIEW

In this assignment, you will write code to perform Monte Carlo inference i.e. Importance sampling and Gibbs sampling. Often, marginalization over large/continuous probability distributions can be intractable. In Monte Carlo inference, we circumvent this problem by sampling from simpler proposal distributions to approximate our target distribution, making the problem much more tractable for complex distributions.

References: Lecture 9

Honour Code. This coding assignment constitutes **15%** of your final grade in CS5340. Note that plagiarism will not be condoned! You may discuss with your classmates and check the internet for references, but you MUST NOT submit code/report that is copied directly from other sources!

2. SUBMISSION INSTRUCTIONS

Items to be submitted:

- **Source code** (zip file containing folders part1 and part2):
  - part1/main.py– code for importance sampling.
  - part2/main.py – code for Gibbs sampling.
- **Report (report.pdf).** This should describe your implementation and be no more than one page.

Please indicate clearly your name and student number (the one that looks like A1234567X) in the report as well as the top of your source code. Zip the two files together and name it in the following format: **A1234567X_lab4.zip** (replace with your student number).

Submit your assignment by **4 November 2020, 2359HRS** to LumiNUS. 25% of the total score will be deducted for each day of late submission.

3. GETTING STARTED

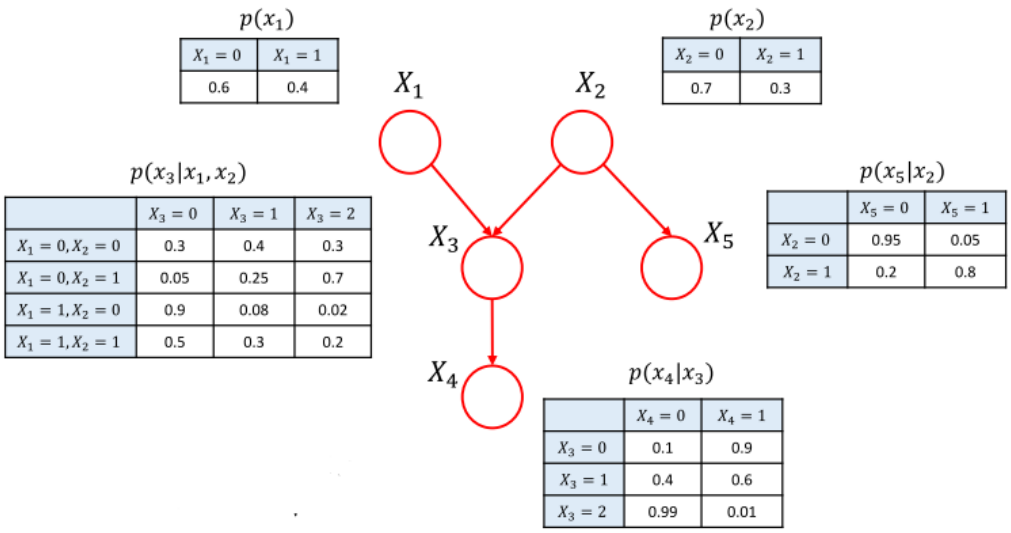
This assignment as well as the subsequent ones require Python 3.5, or later. You need certain python packages, which can be installed using the following command:

```
pip install -r requirements.txt
```

If you have any issues with the installation, please post them in the forum, so that other students or the instructors can help accordingly.

4. TEST CASES

To help with your implementation, we have provided a few sample inputs. They can be found in the **data/inputs** folder. The ground-truth files can be found in **data/ground-truth**. Your predictions will be stored in **data/predictions**. For test cases 1 and 2, the inputs will be on the example given in Lecture 9:



For cases 3 and 4, we will use a slightly larger graph. Note that the ground-truth might be slightly different from your answers. We will assume a tolerance up to 1 decimal place. During grading, your code will be evaluated on hidden test cases on top of the validation test cases we have provided.

Additionally, we expect your code to run in a reasonable amount of time, any time within 2-3x of our timings below is reasonable:

Case	1	2	3	4
Importance Sampling	13s	10s	37s	33s
Gibbs Sampling	12s	8s	39s	32s

Table 1: Timings on i7-6500U @ 2.50GHz.

5. IMPORTANCE SAMPLING

In part 1 i.e. importance sampling, you will be provided with local target conditional distribution $p_u(x_u|x_{\pi_u})$ and local proposal conditional probabilities i.e. $q_u(x_u|x_{\pi_u})$ where π_u is are the parents of node u. In this assignment, we will assume our proposal distribution has the form i.e. $q(X) = \prod q_u(x_u|x_{\pi_u})$ and the target distribution has the form $p(X) = \prod p_u(x_u|x_{\pi_u})$. Each sample is then weighted by the ratio between $\frac{p(X)}{q(X)}$ to obtain the approximate distribution. You are expected to return the conditional probability distribution $p(X_F|X_E)$ where the nodes in the graph are $X = X_F \cup X_E$, with X_F being the query nodes and X_E being the evidence nodes.

Note that during sampling, students are expected to perform sampling in a topological order i.e. parent nodes should be sampled before child nodes, and the input samples should be updated with the new parent samples before sampling from the child node.

You will implement the following functions:

- Performs sampling for a single iteration (all nodes sampled once): **`_sample_step()[3 points]`**
    - Students are expected to sample from the local proposal distributions for each node. Do not sample from a joint proposal distribution.
    - Sampling should be done in topological order. For example, in the example from Lecture 9, sampling in the order X_1, X_2, X_3, X_4, X_5 would be one way to sample topologically.
    - Once the parent node is sampled, use the sample from the parent node as the observation of the parent into the child node's probability distribution to sample from the child's distribution.
- Performs sampling for N iterations, weight the samples and return the approximate conditional probability $p(X_F|X_E)$: **`_get_conditional_probability()[4 points]`**
    - Do not compute the joint proposal distribution or joint target distribution to get the importance. You are expected to take the output these values by taking the scalar product of the slices of the local proposal/conditional distribution values.

Note that we will also provide evidence variables and evidence should be observed for the local proposal distribution, similar to the lecture notes.

Hint: It might be useful to create additional functions for this part. Place these functions between the designated comment blocks for each file.

6. GIBBS SAMPLING

In part 2 i.e. Gibbs sampling, you will be provided with conditional probabilities i.e. $q_u(x_u|X - \{x_u\})$ for each node x_u. You will sample from the conditional distributions for each node, by holding other nodes fixed. The samples are then weighted equally to obtain the approximate distribution $p(X_F|X_E)$ where X_F are the query nodes and X_E are the evidence nodes.

- Performs sampling for a single iteration (all nodes sampled once): **`_sample_step()[3 points]`**
    - Refer to lecture 9.
- Performs sampling for N iterations, and return the approximate distribution **`_get_conditional_probability()[5 points]`**
    - Students are expected to reduce the proposal distributions for each node to its Markov Blanket.

Note that we will also provide evidence variables and the node distributions should be updated with the evidence variables.

Hint: It might be useful to create additional functions for this part. Place these functions between the designated comment blocks for each file. Some points are attributable to more efficient implementations.