Probabilistic Graphical Models
Spring 2023

Homework Assignment 2
Exact Inference
Due Date: 31.5.2023

May 10, 2023

1 (60 pts.) Programming with PGMPy

In this section, you will implement a "sampling-based version" of the junction tree algorithm for computing the partition function of a Markov network. The main idea is to compute the probability of the partition function by combining an exact inference algorithm with sampling.

Algorithm ComputePartitionFunction()

Input: A Markov network $G(V, E)$ representing the joint probability distribution P_V; Integers w and N
Output: An estimate of the partition function

1: $Z \leftarrow 0$
2: Construct a junction tree T for the Markov Network G.
3: Heuristically remove variables from the junction tree T until its largest cluster (bag) has most w variables. Let $\mathbf{X}$ be the removed variables.
4: $\mathbf{X} \leftarrow$ `wCutset`$(\mathbf{T})$ {You need to implement this function. The heuristic is: remove the variable that appears in the largest number of bags in the tree $\mathbf{T}$.}
5: Let Q be a uniform distribution over $\mathbf{X}$ {i.e., Q is uniform over all assignments $\mathbf{X} = \mathbf{x}$}
6: **for** $i = 1$ to N **do**
7:     Generate a sample to the variables in $\mathbf{X}$. That is, generate a value assignment to all variables in $\mathbf{X}$ from Q. Let the sampled assignment be $\mathbf{X} = \mathbf{x}$
8:     $\mathbf{x} \leftarrow$ `GenerateSample`$(\mathbf{Q})$ {You need to implement this function}
9:     $part_{\mathbf{x}} \leftarrow$ `computePartitionFunctionWithEvidence`$(T, G, \mathbf{x})$ {You are provided the implementation of this function}
10:     $t_{\mathbf{x}} \leftarrow \frac{part_{\mathbf{x}}}{Q(\mathbf{x})}$
11:     $Z \leftarrow Z + t_{\mathbf{x}}$
12: **return** $\frac{Z}{N}$

Figure 1: Computing the partition function with conditioning and sampling

Since you are provided with an implementation of the function `computePartitionFunctionWithEvidence`, which computes the partition function with evidence, the main challenge is to implement the `wCutset` and `GenerateSample` functions. For more details, see `skeleton.py` file.

What to do and turn in for Part 1 ?

1. Write a program that implements Algorithm ComputePartitionFunction. The program should take the following inputs:

   - A Markov network in the UAI format (hint: use the `UAIReader` class of PGMPy).
   - An integer w, which denotes the bound on the largest cluster of the junction tree.
   - An integer N, which denotes the number of samples.

   The program should output an estimate of the partition function as in Figure 1.

2. Replace the uniform distribution Q with the distribution Q^{RB} that returns the actual probability that $\mathbf{X} = \mathbf{x}$. In other words, the sample $\mathbf{x}$ is sampled from Q^{RB}, where $Q^{RB}(\mathbf{x}) = \sum_{V \setminus \mathbf{X}} P_V(\mathbf{x}, V - \mathbf{X})$.

For parts 1 and 2, turn in your source code and a readme file that describes how to use your software. You are provided with the file `skeleton.py` that contains an implementation to the function `computePartitionFunctionWithEvidence`, and the input file `grid4x4.uai`. Your code should contain two functions, the first corresponding to item 1 (`ExperimentsDistributionQUniform`), and the second to item 2 (`ExperimentsDistributionQRB`).

3. Execute your two programs on the PGM provided (e.g., `grid4x4.uai`). Try the following values for $N = \{50, 100, 1000, 5000\}$ and $w = \{1, 2, 3, 4, 5\}$, and execute each algorithm at least 10 times (using different random seeds). For each run, compute the log-relative error between the exact partition function and the approximated one computed by your algorithm. That is, compute:

$$\frac{\left|\log Z - \log(\hat{Z})\right|}{\log Z} \tag{1}$$

where Z is the exact answer and $\hat{Z}$ is the approximate answer your algorithm computed.

Report your results in a table such as the one given below. Describe your findings in a few sentences. Which sampling distribution leads to better results in terms of accuracy and runtime? which method is better ? How do N and w affect the accuracy, runtime, variance etc. ?

Problem Name			**Uniform Sampling** (Q)				**Weighted Uniform** (Q^{RB})			
		$N \rightarrow$	50	100	1000	5000	50	100	1000	5000
Grids 4 (grid4x4.uai)	$w = 1$	Time	$t \pm t_s$							
	$w = 1$	Error	$e \pm e_s$							
Grids 4 (grid4x4.uai)	$w = 2$	Time	$t \pm t_s$							
	$w = 2$	Error	$e \pm e_s$							
Grids 4 (grid4x4.uai)			$\cdots$							
Grids 4 (grid4x4.uai)	$w = 5$	Time	$t \pm t_s$							
	$w = 5$	Error	$e \pm e_s$							

In the table above, the runtime is in seconds. The quantity before the $\pm$ is the average (e.g., e and t), and the quantity after $\pm$ is the standard deviation (e.g., t_s and e_s) over 10 executions (recall that you run every algorithm 10 times).
For part 3, turn in a PDF or a word file.

2 (40 pts.) Variable Elimination

We consider a Bayesian Network in Figure 2 with the following random variables (RVs): X_1, X_2, $\boldsymbol{Y} = (Y_1, \ldots, Y_N)$, $\boldsymbol{Z} = (Z_1, \ldots, Z_N)$, whose joint distribution $P_{X_1,X_2,\boldsymbol{Y},\boldsymbol{Z}}$ is the following:

$$P_{X_1,X_2,\boldsymbol{Y},\boldsymbol{Z}}(x_1, x_2, \boldsymbol{y}, \boldsymbol{z}) = P_{X_1}(x_1) P_{X_2}(x_2) \prod_{n=1}^{N} P_{Y_n|X_1}(y_n|x_1) P_{Z_n|Y_n,X_2}(z_n|y_n, x_2) \tag{2}$$

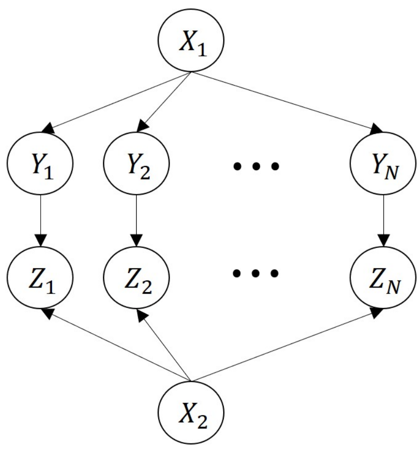

Figure 2: The Bayesian Network

The RVs X_1, $Y_1, \ldots, Y_N$, $Z_1, \ldots, Z_N$ take on values in $\{1, 2 \ldots, K\}$, and X_2 takes on values in $\{1, 2, \ldots, N\}$. A minimal directed I-Map of the distribution is shown in Figure 2.
In what follows, assume that the complexity of looking up the values $P_{X_1}(x_1)$ (for $x_1 \in \{1, 2 \ldots, K\}$), $P_{X_2}(x_2)$, $P_{Y_n|X_1}$, and $P_{Z_n|Y_n, X_2}$ is in $O(1)$.

1. (5 pts.) Draw the moral graph over the RVs $X_1, X_2, Y_1, \ldots, Y_N$, after conditioning on $\boldsymbol{Z}$.
2. (5 pts.) Provide a good elimination ordering for the graph from the previous item. Draw the graph with the fill-edges that result from your proposed elimination order.
3. (10 pts.) Determine α and β such that the complexity of computing $P_{X_1|\boldsymbol{Z}}$ using the Variable Elimination algorithm along the order you proposed in the previous item is $O(N^\alpha K^\beta)$.

   For parts (4) and (5), suppose that you also have the following conditional independence relation: $Y_i \perp Z_i | X_2 = c$ for all $i \in [1, N]$ where $c \neq i$. That is, the conditional independence relation holds only for certain values of c. This type of conditional independence relation is called *context-specific* conditional independence.
4. (12 pts.) For fixed values of $\boldsymbol{Z} = \boldsymbol{z}$, $X_1 = x_1$, and $X_2 = c$ (i.e., where $c \in [1, N]$), show that:

$$P_{\boldsymbol{Z}|X_1, X_2}(\boldsymbol{z}|x_1, c) = f(x_1, c, z_c) g(c, \boldsymbol{z})$$

   for some function $f(x_1, c, z_c)$ that can be computed in $O(K)$ operations, and some function $g(c, \boldsymbol{z})$ that can be computed in $O(N)$ operations. Express $f(x_1, c, z_c)$ in terms of $P_{Y|X_1}$ and $P_{Z|Y, X_2}$, and $g(c, \boldsymbol{z})$ in terms of $P_{Z|X_2}$. (Hint: draw the network in the scenario described:)
5. (8 pts.) Provide an expression for $P_{X_1, \boldsymbol{Z}}(x_1, \boldsymbol{z})$ in terms of P_{X_1}, $P(X_2)$, f and g (from the previous item). Use your expression to explain how $P_{X_1|\boldsymbol{Z}}(x_1|\boldsymbol{z})$ can be computed in $O(N^2 + NK^2)$ operations for a fixed $\boldsymbol{Z} = \boldsymbol{z}$.

Good Luck!