ASSIGNMENT 4: WEATHER FORECASTING

Goal: The goal of this assignment is to gain experience with parameter learning and inference in Bayesian Networks, and to understand their usefulness in a real-world scientific domain.

Scenario: Meteorological researchers have created a Bayesian Network that models the inter-relationships between atmospheric conditions, storm development factors, and observable weather indicators. Your job as computer scientists is to learn the parameters (conditional probabilities) of this network from historical weather data.

As often happens in real-world data, some weather records have missing measurements — due to instrument failures, unreported variables, or sensor limitations. Your task is to use statistical learning techniques to estimate the missing parameters of the network, so that it can later be used for storm severity and hail forecasting.

Problem Statement: We are given the Bayesian Network created by the atmospheric scientists. The network is shown below:

The network models relationships among about 56 variables, including atmospheric dynamics, radar and satellite measurements, and severe weather outcomes such as hail and storm type. The network includes variables like:

- Upper-air conditions: LLIW (low-level wind shear), RELHUM (relative humidity), TEMP (temperature), WNDMOD (wind modulation)
- Storm development factors: MCONV (moist convection), STAB (stability), VORTEX (vorticity), LIFT (updraft strength)
- Observable indicators: ECHO_TOP, RADAR_REF, VIL, CLOUD_TOP, TEMP_OBS, HUM_OBS
- Outcome nodes: HAIL, HAIL_SIZE, SEVERE_STORM, TORNADO

Such networks can be represented in many formats. We will use the .bif format. BIF stands for Bayesian Interchange Format. The details about the format are here. We are also providing a .bif parser so that you can start directly from a parsed Bayesian network represented as a graph.

Your task is to learn the missing conditional probabilities in this network based on the provided dataset. Once learned, the network can be used to infer the likelihood of severe weather events (e.g., P(HAIL=True | evidence)).

Input format:

We will work with hailfinder.bif network. Please have a look at this file to get a basic understanding of how this information relates to the Bayes net image above. A sample Bayes net is as follows

```
variable “X” {
        type discrete[2] { “True” “False” };
}
variable “Y” {
        type discrete[2] { “True” “False” };
}
variable “Z” {
        type discrete[2] { “True” “False” };
}
probability(“X”) { table 0.2 0.8 ; }
probability(“Y”) { table 0.4 0.6 ; }
probability(“Z” “X” “Y”) { table 0.2 0.4 0.3 0.5 0.8 0.6 0.7 0.5; }
```

This says that X, Y, and Z all have two values each. X and Y has no parents and prior P(X=True)=0.2, P(X=False)=0.8, and so on. Z has both X and Y as parents. Its probability table says P(Z=True|X=True, Y=True) = 0.2, P(Z=True|X=True, Y=False) = 0.4 and so on.

Our input network will have the Bayes net structure including variables and parents, but may not have some (or all) probability values. We will use -1 to represent that the probability value is unknown.

probability(“X”) { table -1 -1 ; } will represent that prior probability of X is unknown and needs to be computed via learning.

To learn these values we will provide a data file. Each line will be a historical weather record. All features will be listed in exactly the same order as in the .bif network and will be comma-separated. If a feature value is unknown we will use the special symbol “?” for it. There will be no more than 1 unknown value per row. Example:

“True”, “False”, “True”
“?”, “False”, “False”

Here the first row says that X=True, Y=False and Z=True. The second row says that X is not known, Y and Z are both False.

Overall your input will be hailfinder.bif with most probability values -1 and this datafile. The datafile will have about 10,000 historical weather records.

Output format:

Your output will be the result of learning each probability value in the conditional probability tables. In other words, you need to replace all -1s with a probability value upto **four decimal places**. Thus, your output is a complete hailfinder.bif network.

What is being provided?

We are providing you with a startup code which parses the hailfinder.bif file and stores the relevant information about the graph in a data structure. We have provided you with some functions which may be of help to you like querying for children of a node, parents of a node etc. but if you require any additional functions, you are free to play with this file. Additionally, since it is simple parser you can parse the file yourself and create your own parser if you feel that would be more helpful.

The following files are provided:

A Dataset file: records.dat file where a single line is a single weather record and each variable in the record is separated by spaces. The unknown record is marked by "?". Each line contains at max 1 missing record. The file contains more than 10000 records.

A Start up code file.

A format checker to check your output file adheres to hailfinder.bif format. The format checker assumes that hailfinder.bif, solved_hailfinder.bif and gold_hailfinder.bif are present in current directory and outputs its results. It also outputs the total learning error.

Hailfinder.bif whose parameters need to be learned.

Gold_Hailfinder.bif has the true parameters.

What to submit?

1. Submit your code in a .zip file named in the format **<EntryNo>.zip.** If there are two members in your team it should be called <EntryNo1>_<EntryNo2>.zip. Make sure that when we run "unzip yourfile.zip" it contains a directory with the same name as the zip file and the following files are present in that directory:
    - **compile.sh**
    - **run.sh**
    - **Writeup.txt**
    - You will be **penalized** for any submissions that do not conform to this requirement.
    - Your code must compile and run on our VMs. They run amd64 Linux version ubuntu 20.04. You are already provided information on the compilers on those VMs. They have RAM of 8 GB.
    - Your **run.sh** should take 2 input files, hailfinder.bif and records.dat and output a file named solved_hailfinder.bif file:
      ./run.sh hailfinder.bif <sample_data>.dat
      Note: Any name could be given to input data file. Output file **should** be named – **solved_hailfinder.bif.**
    - We will run your code on a new dataset (but with hailfinder.bif structure) and verify the ability of your code to find conditional probabilities. Your code needs to finish learning in 2 mins. The dataset will have about 10,000 weather records.

    The writeup.txt should have three lines as follows

    First line should be just a number between 1 and 2. Number 1 means C++. Number 2 means Python.

    Second line should be this honor code. "Even though I/we have taken help from the following students and LLMs in terms of discussing ideas and coding practices, all my/our code is written by me/us."
    Third line should mention names of all students and LLMs you discussed/collaborated with (see guidelines on collaboration vs. cheating on the course home page). If you never discussed the assignment with anyone else say None.
    After these first three lines you are welcome to write something about your code, though this is not necessary.

Code verification before submission

Your submission will be auto-graded. This means that it is absolutely essential to make sure that your code follows the input/output specifications of the assignment. Failure to follow any instruction will incur significant penalty (20%). Please doublecheck your final submission with the given model checker so that you are sure it can be autograded.

Evaluation Criteria

1. Accuracy of learned values. For each parameter value we will compute the absolute value of the difference from the gold value. We will sum these error terms for all parameters to compute the total learning error.
2. Extra credit may be awarded to standout performers.

What is allowed? What is not?

1. You may work in teams of two or by yourself. Typical rules of repeating a partner in the course apply. We do not expect a different quality of assignment for 2 people teams. At the same time, please spare us the details in case your team cannot function smoothly.
2. You can use C++ or python this assignment. But you cannot use built-in libraries for Bayes nets or expectation maximization.
3. You must not discuss this assignment with anyone outside the class. You cannot ask LLMs to write code for you. However, you are allowed to give the LLM your code in case you are stuck in debugging, so that you can learn about your mistakes fast, and do not waste time in debugging. **Make sure you mention the names in your write-up in case you discuss with anyone from within the class outside your team.** Please read academic integrity guidelines on the course home page and follow them carefully.Please do not search the Web for solutions to the problem.
4. Please do not use ChatGPT or other large language models for creating direct solutions (code) to the problem. Our TAs will ask language models for solutions to this problem and add its generated code in plagiarism software. If plagiarism detection software can match with TA code, you will be caught.
5. Your code will be automatically evaluated. You get a *minimum* of 20% penalty if your output is not automatically parsable.
6. Please do not use ChatGPT or other large language models for creating direct solutions (code) to the problem. Our TAs will ask language models for solutions to this problem and add its generated code in plagiarism software. If plagiarism detection software can match with TA code, you will be caught.
7. We will run plagiarism detection software. Any team found guilty of either (1) sharing code with another team, (2) copying code from another team, (3) using code found on the Web, will be awarded a suitable strict penalty as per IIT and course rules.