

ASSIGNMENT 4: MEDICAL DIAGNOSIS

Goal: The goal of this assignment is to get experience with learning of Bayesian Networks and understanding their value in the real world.

Scenario: Medical diagnosis. Some medical researchers have created a Bayesian network that models the inter-relationship between (some) diseases and observed symptoms. Our job as computer scientists is to learn parameters for the network based on health records. Unfortunately, as it happens in the real world, certain records have missing values. We need to do our best to compute the parameters for the network, so that it can be used for diagnosis later on.

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

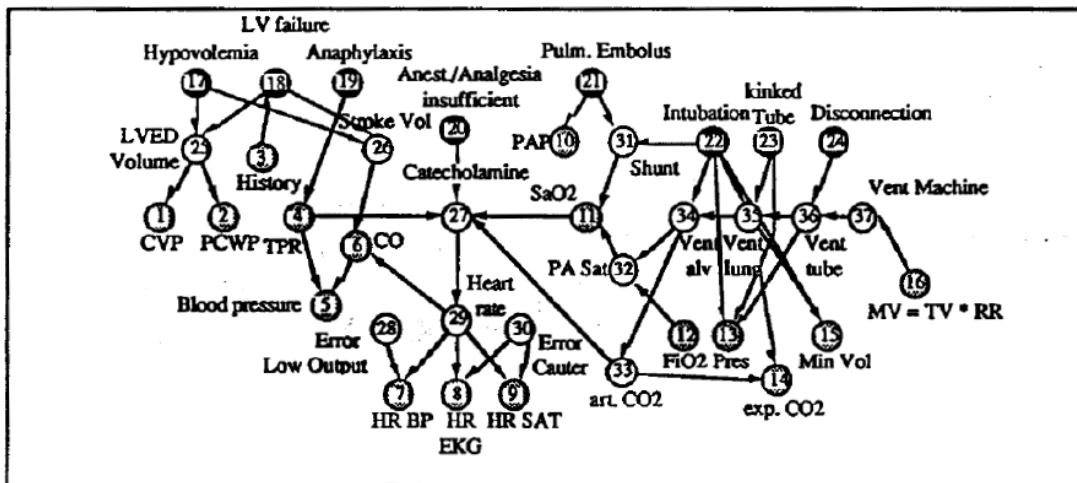


Fig. 1 The ALARM network representing causal relationships is shown with diagnostic (●), intermediate (O) and measurement (⊗) nodes. CO: cardiac output, CVP: central venous pressure, LVED volume: left ventricular end-diastolic volume, LV failure: left ventricular failure, MV: minute ventilation, PA Sat: pulmonary artery oxygen saturation, PAP: pulmonary artery pressure, PCWP: pulmonary capillary wedge pressure, Pres: breathing pressure, RR: respiratory rate, TPR: total peripheral resistance, TV: tidal volume

Notice that eight diagnoses are modeled here: hypovolemia, left ventricular failure, Anaphylaxis, insufficient analgesia, pulmonary embolus, intubation, kinked tube, and disconnection. The observable nodes are CVP, PCWP, History, TPR, Blood Pressure, CO, HR BP, HR EKG, HR SAT, SaO2, PAP, MV, Min Vol, Exp CO2, FiO2 and Pres

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.

The goal of the assignment is to learn the Bayes net from a healthcare dataset.

Input format:

We will work with [alarm.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 will not have 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 patient 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 alarm.bif with most probability values -1 and this datafile. The datafile will have about 10,000 patient 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 alarm.bif network.

What is being provided?

We are providing you with a startup code which parses the alarm.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 patient 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 11000 records.

A [Start up code file](#).

A [format checker](#) to check your output file adheres to alarm.bif format. The format checker assumes that alarm.bif, solved_alarm.bif and gold_alarm.bif are present in current directory and outputs its results. (A next version will also compute the total learning error).

[Alarm.bif](#) whose parameters need to be learned.

[Gold Alarm.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 machine named ‘todi’ or any machine with similar configuration present in GCL.
 - Your **run.sh** should take 2 input files, alarm.bif and records.dat and output a file named solved_alarm.bif file:
./run.sh alarm.bif <sample_data>.dat
Note: Any name could be given to input data file. Output file **should** be named – **solved_alarm.bif**.
 - We will run your code on a new dataset (but with alarm.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 patient records.

The writeup.txt should have two lines as follows

First line should be just a number between 1 and 3. Number 1 means C++. Number 2 means Java and Number 3 means Python.

Second line should mention names of all students 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 two 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 will work alone or in a group of two. Typical rules of repeating a partner in the course apply.
2. You can use C++, Java 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. **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.
4. Please do not search the Web for solutions to the problem.
5. Please do not use ChatGPT or other large language models for solutions 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.
6. Your code will be automatically evaluated against a new dataset generated from a different Bayes net with the same structure. You get a 20% penalty if your output is not automatically parsable. If your code had logical errors, which you fix with a request of second evaluation, you could get as much as a 50% penalty.
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.