Computer Science 384
St. George Campus

February 26, 2016
University of Toronto

Homework Assignment #3: Bayesian Networks
Due: March 22, 2016 by 11:59 PM

Silent Policy: *A silent policy will take effect 24 hours before this assignment is due, i.e. no question about this assignment will be answered, whether it is asked on the discussion board, via email or in person.*

Late Policy: 10% per day after the use of 2 grace days.

Total Marks: This part of the assignment represents 10% of the course grade.

Handing in this Assignment

What to hand in on paper: Nothing.

What to hand in electronically: You must submit your assignment electronically. Download `BayesianNetwork.py`, `VariableElimination.py`, `MedicalBayesianNetwork.py` and `DecisionSupport.py` from the A3 web page. Modify `VariableElimination.py` and `DecisionSupport.py` so that they solve the problems specified in this document. Submit your modified files `VariableElimination.py` and `DecisionSupport.py`.

How to submit: If you submit before you have used all of your grace days, you will submit your assignment using MarkUs. Your login to MarkUs is your CDF username and password. It is your responsibility to include all necessary files in your submission. You can make as many submissions as you like while you still have grace days; the number of grace days you use will be determined by the time of your final submission.

If you are submitting your assignment late and have used all your grace days, you must email it your assignment files to `csc384w16-late@cs.toronto.edu` with '[CSC384] Late Assignment Submission' in the subject header. ONLY submit your assignment via email AFTER you have used all your grace days. Note that the lateness penalty (10%/day) will apply to these submissions.

Warning: marks will be deducted for incorrect submissions.

We will test your code electronically. You will be supplied with a testing script that will run a **subset** of the tests. If your code fails all of the tests performed by the script (using Python version 3.4.3), you will receive zero marks. It's up to you to figure out further test cases to further test your code – that's part of the assignment!

When your code is submitted, we will run a more extensive set of tests which will include the tests run in the provided testing script and a number of other tests. You have to pass all of these more elaborate tests to obtain full marks on the assignment.

Your code will not be evaluated for partial correctness, it either works or it doesn't. It is your responsibility to hand in something that passes at least some of the tests in the provided testing script.

- *make certain that your code runs on CDF using python3 (version 3.4.3) using only standard imports.* This version is installed as "python3" on CDF. Your code will be tested using this version and you will receive zero marks if it does not run using this version.

- Do not add any non-standard imports from within the python file you submit (the imports that are already in the template files must remain). Once again, non-standard imports will cause your code to fail the testing and you will receive zero marks.
- Do not change the supplied starter code. Your code will be tested using the original starter code, and if it relies on changes you made to the starter code, you will receive zero marks.

Clarification Page: Important corrections (hopefully few or none) and clarifications to the assignment will be posted on the Assignment 3 Clarification page, linked from the CSC384 A3 web page, also found at: `http://www.cdf.toronto.edu/~csc384h/winter/Assignments/A3/a3_faq.html`. You are responsible for monitoring the A3 Clarification page.

Questions: Questions about the assignment should be asked on Piazza:
`https://piazza.com/utoronto.ca/winter2016/csc384/home`.

Introduction

There are two parts to this assignment

1. The implementation of the variable elimination algorithm. Starter code is provided which you must complete.
2. Adapting the variable elimination algorithm to map possible treatments to outcomes in a medical decision support system application.

What is supplied and if/how it is to be modified:

- **BayesianNetwork.py** – class definitions for `Variable`, `Factor`, and `BayesianNetwork`. Do not modify this file.
- **VariableElimination.py** – starter code which you must complete. Write the functions `multiply_factors`, `restrict_factor`, `sum_out_variable` and `VariableElimination`.
- **MedicalBayesianNetwork.py** – class definitions for `MedicalBayesianNetwork` and `Patient`. Do not modify this file.
- **DecisionSupport.py** – starter code which you must complete. Write the function `DecisionSupport`
- **Sample test cases** for testing your code. *These will be released on the A3 web page on Tuesday, March 1.*

Question 1: Variable Elimination

You will implement the variable elimination algorithm `VariableElimination` in the provided file `VariableElimination.py`. The functions `multiply_factors`, `restrict_factor`, and `sum_out_variable` must also be implemented in the same file. These factor operations will be necessary to complete the variable elimination algorithm.

When choosing a variable elimination ordering, use the `min_fill_ordering` function provided in `BayesianNetwork.py`. Deviating from the provided ordering may result in deducted marks.

The file `VariableElimination.py` provides the complete input/output specification of the functions you are to implement. The following is an overview:

- `multiply_factors(factors)` – returns a new `Factor` object that is the product of all the factors in `factors`.

- `restrict_factor(factor, variable, value)` – returns a new `Factor` object that is the restriction of factor `factor` with respect to the variable `variable` that has been assigned the value `value`.
- `sum_out_variable(factor, variable)` – returns a new `Factor` object that is the factor `factor` wherein the variable `variable` has been summed out.
- `VariableElimination(net, queryVar, evidenceVars)` – Returns a probability distribution over the variable `querVar`. This should be the result of performing variable elimination on the Bayesian network `net` subject to variables in `evidenceVars` that have had evidence assigned to them.

In all cases you should not modify any of the input parameters. Each function must create a new object to return. Do not try and simply alter an existing object.

Your implementation will depend on code provided in `BayesianNetwork.py`. Do not modify this file. Your code will be tested using our version of the file with your different functions tested individually. The following is an overview of the features in `BayesianNetwork.py`. More complete documentation is provided in the file's comments.

- Class `Variable`: A single instance of a Bayes net variable
  - Note the `set_evidence` function which must be used to assign a value to the variable when it is used as evidence in the variable elimination algorithm
  - Contains the variables' possible domain values
- Class `Factor`: defines a function over an **ordered** sequence of variables. All conditional probability tables in a Bayes net are represented by factors.
  - Variables in the factor are in the list `scope`
  - All possible assignments of the variables have an associated value in the factor
- Class `BayesianNetwork`: A small class containing a list of factors and a list of relevant variables.
- Class `AssignmentIterator`: A helper function for iterating over all possible variable assignments for a factor.
  - Using this is optional, see the `print_table` function for an example of how it is used
- Function `min_fill_ordering(Factors,QueryVar)`: returns the min fill ordering given a list of factors and a query variable. You must use this function to fix the ordering of variables that you eliminate in your Variable Elimination implementation.

Keep in mind that the test cases provided are not exhaustive. Consider all possible situations when implementing your code.

Question 2: Medical Decision Support System

You will implement a Medical Decision Support System by utilizing your code from Question 1 together with the implementation of a function `DecisionSupport` in the provided file `DecisionSupport.py`. Details follow.

Background: In class we saw how to use Bayesian inference to compute the probability of a query given some evidence. An interesting application of Bayesian inference is in *Medical Decision Support*, which can aid patients and medical practioners in understanding the likelihood of different medical outcomes predicated on observations regarding the patient's health, and potential treatment choices. For example, consider a patient who has been diagnosed with stage 2 colon cancer. The patient's medical information may also include observations regarding their age, whether this is a new or recurring cancer, genetic disposition, etc. The patient and medical practitioner wish to understand the probability of eradicating the cancer

as well as experiencing certain treatment "side effects" based on the observations they have on hand and as a consequence of selecting different treatment options such as surgery, chemotherapy, or radiation therapy. In this question we will explore such a simple medical decision support system, mapping actions related to medical treatments onto medical outcomes. More specifically, given patient-specific health information, we will use a Bayesian network to predict patient outcomes conditioned on possible treatment choices.

You will be provided with a Bayesian network detailing possible medical observations, conditions, treatments and outcomes for a patient. This introduces two new types of variables:

- **Treatment Variables** are variables that represent the treatment choices a patient may make. For example, choosing between medication or surgery.
- **Outcome Variables** are random variables which represent future results of particular interest to the patient such as recovery time, side effects, pain, or degree of success.

Note that Treatment and Outcome variables are mutually exclusive, meaning that a variable in the net cannot be *both* a Treatment and Outcome variable.

Given such a Bayesian network and patient-specific evidence as input, your task will be to **output a conditional probability table relating the treatment variables to the outcome variables**. This will allow the patient to make a more informed decision as to which treatment he or she should select.

To do this you must complete and submit `DecisionSupport.py`, which contains the single function `DecisionSupport`. *For this question, this is the only function you need to implement.* This function will be given as input a `MedicalBayesianNetwork` object and a `Patient` object. It should return a conditional probability table that represents the likelihood of outcomes of given treatments in the form of a `Factor` object.

We expect that `DecisionSupport.py` will exploit code written in Question 1.

The classes defined in `MedicalBayesianNetwork.py` are:

- Class `MedicalBayesianNetwork` – A `BayesianNetwork` object with lists added to identify the treatment variables and the outcome variables that are in the network.
- Class `Patient` – Contains a set of evidence variables corresponding to patient information (i.e. symptoms, conditions, pertinent clinical demographics).

Example `MedicalBayesianNetwork` and `Patient` objects will be supplied with the test script.

Code from Question 1 should be reused for Question 2, but note that the TA's copy of `VariableElimination.py` will be used to mark Question 2 rather than the one you submit.

Additional Notes

- Time inefficient solutions may be deducted marks.
- As noted previously, non-standard imports are *not* permitted. Standard imports available on the CDF machines may be used, but none are required to complete the assignment. Do not use the function `copy.deepcopy()`; it will cause problems.
- Floating point arithmetic can be imprecise. Your calculations do not have to exactly match the expected answer, but must be within 1% of the answer.

HAVE FUN and GOOD LUCK!