TÉCNICO LISBOA

Artificial Intelligence and Decision Systems (IASD)
Lab classes, 2015/2016
Assignment #3

Version 1.0 - November 18, 2015

The goal of this lab assignment is to develop using Python a probabilistic reasoner based on Bayesian Networks (BN). A BN allows the representation of the dependencies among random variables and the specification of any full joint probability distribution.

For this lab, consider the following specification of BN:

1. A set of discrete random variables makes up the nodes of the network. The possibles values for each variable is finite.
2. A set of directed links connects pairs of nodes. If there is a link from a node X to node Y, X is said to be parent of Y.
3. Each node X_i has a conditional probability distribution $P(X_i|Parents(X_i))$ that quantifies the effect of the parents on the node.
4. The graph has no directed cycles (and hence is a directed, acyclic graph, or DAG).

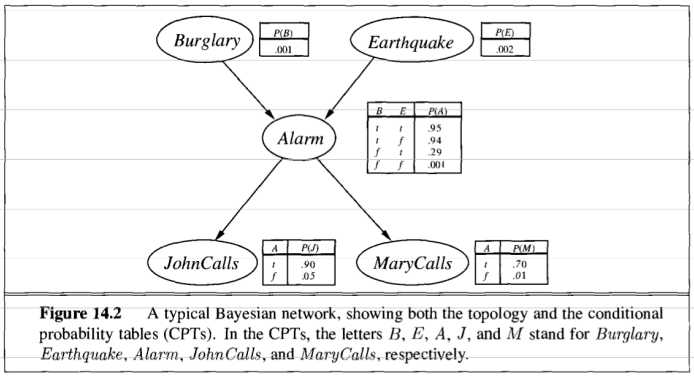

Figure 14.2 A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters B, E, A, J, and M stand for *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, and *MaryCalls*, respectively.

Figure 1: Example of a Bayesian network (extracted from Russel and Norvig, Artificial Intelligence: A Modern Approach, 2nd edition, Prentice Hall, 2003).

The basic task for any probabilistic inference system is to compute the posterior probability distribution for a set of query variables, given some observed event - that is, some assignment of values to a set of evidence variables.

Let X be the query variable; $\mathcal{E}$ the set of evidence variables $E_1, E_2, \ldots, E_m$, and e a particular observed event, and $\mathcal{Y}$ the set of nonevidence variables $Y_1, Y_2, \ldots, Y_l$. Thus the complete set of variables is $\mathcal{U} = \{X\} \cup \mathcal{E} \cup \mathcal{Y}$.

A typical query asks for the posterior probability distribution $P(X|e)$.[1] To answer this question one may use the *sum-product variable elimination algorithm* (VE).

To successfully conclude this lab assignment, you should:

1. develop a Python program to read the BN specification given below, a query variable X, a set of evidence variables and the corresponding assignment (the event e), according to the Query/Evidence specification given below, and

2. calculate using the VE algorithm the probability distribution $P(X|e)$.

The program should also have an option to print out all the steps of the VE algorithm.

BN specification

The Bayesian network specification should be read from an input file with extension **bn**, and follow the format presented next.

This specification involves two main elements: a variable (VAR) and a conditional probability table (CPT). The VAR element allows to describe a random variable (a node) of the BN; it should start with the string "VAR", and then two mandatory keys (**name**, **values**) and two optional keys (**alias**, **parents**), with the following meaning:

name the name of the variable

values the list of possible values (separated by white spaces)

alias an alias name for the variable

parents the list of variables that are parents of the variable (separated by white spaces); here you can use either the variable name or the alias.

[1] You may assume that the query variable is not among the evidence variables.

For the example given in Figure 1, the variable "MaryCalls" is described by:

```
VAR
name MaryCalls
alias M
parents Alarm
values T F
```

The CPT element allows to define the conditional probability table for a variable; it should start with the string "CPT", and then two mandatory keys (**var**, **table**), with the following meaning:

var the name of the variable (here you can use both the variable name or the alias)

table the complete conditional probability distribution; each line of the table starts with a possible value for the variable followed by a combination[2] of possible values for the variable's parents, and it ends with the corresponding probability value. All table elements are separated by white spaces (or newlines)

Again for the example given in Figure 1, the CPT for "MaryCalls" is described by:

```
CPT
var MaryCalls
table t t 0.7 t f 0.01 f t 0.3 f f 0.99
```

The complete specification for the example in Figure 1 is the following:

```
# Variable specification
VAR
name MaryCalls
alias M
parents Alarm
values T F

VAR
name Earthquake
alias E
values T F

VAR
name Burglary
```

[2]If the variable does not have any parents, the line should have only a possible value for the variable and the corresponding probability.

```
alias B
values T F

VAR
name JohnCalls
alias J
parents Alarm
values T F

VAR
name Alarm
alias A
parents Burglary E
values T F

# CPT specification
CPT
var M
table t t 0.7 t f 0.01 f t 0.3 f f 0.99

CPT
var JohnCalls
table t t 0.9 t f 0.05 f t 0.1 f f 0.95

CPT
var B
table t 0.001 f 0.999

CPT
var E
table t 0.002 f 0.998

CPT
var Alarm
table
t t t 0.95
t t f 0.94
t f t 0.29
t f f 0.001
f t t 0.05
f t f 0.06
f f t 0.71
f f f 0.999
```

Any line starting with a '#' should be considered a comment and may be ignored. Your program should perform the necessary tests to ensure that the input file with a BN specification is complete and correct.

Query/Evidence specification

The query variable and evidence variables together with the observed values should be read from an input file with extension **in**, and follow the format presented next.

This specification involves two main elements: a variable (QUERY) and a list of variables (EVIDENCE).

The QUERY element defines the query variable; it should start with the string 'QUERY', and then either the name or the alias of the variable (separated by white spaces).

The EVIDENCE element allows to define the set of variables and corresponding values that constitute the evidence; it should start with the string 'EVIDENCE' followed by an integer n defining the number of evidence variables. After that, the file should have n pairs: variable (name or alias) and value (separated by white spaces).

An example of an input file for the example in Figure 1 is the following:

```
# Query definition
QUERY Burglary
# evidence definition
EVIDENCE 2 JohnCalls T MaryCalls T
```

Program execution

The program should be executed in the following way:

```
$ python bn_inference.py input1.bn input2.in [-verbose]
```

where:

`bn_inference.py` is the python file where execution starts;

`input1.bn` is an input file where the bayesian network is defined.

`input2.in` is an input file where the query and evidence are defined.

`-verbose` is an optional argument to state if the program should print out all the steps of the algorithm.

The input files always have extensions `.bn` for the BN network and `.in` for the query/evidence, but may have any names. Your program **should not assume** that the input files are always in the correct format.

Program output

The result of the program execution (with or without the verbose option) should be saved in an output file with the same name as the `.in` input file, but with the extension `.sol`. For instance, if the **in** file is named `input231.in`, the output file should be named `input231.sol`.

The output file should start with the resulted probability distribution for the query variable, given the evidence. The first line of the solution should be a string with: ten '#' sign characters, a white space, the word 'SOLUTION', a white space, and 10 more #.

The solution should be presented with:

- string 'QUERY' followed by the query variable name (separated by white spaces);
- string 'EVIDENCE' followed by pairs variable/value (all separated by white spaces);
- string 'QUERY_DIST' followed by pairs of values/probabilities; a pair for each possible value for the query variable (all separated by white spaces).

An example of an output for the example in Figure 1 is the following:

```
########## SOLUTION ##########
QUERY Burglary
EVIDENCE JohnCalls T MaryCalls T
QUERY_DIST T 0.284 F 0.716
```

If the verbose option is on, the output file should end with a *clear and readable* presentation of the steps of the algorithm performed to determine the solution. To separate the two parts (solution and steps), insert a line with 10 #, a white space, the string 'STEPS', a white space, and 10 more #.

The deliverable of this Lab Assignment consists of two components:

- the Python source code
- a short report with no more than 2 pages.

Deadline: 24h00 of 12-Dez-2015, by email to lmmc@isr.ist.utl.pt