The University of Calgary
Department of Electrical and Computer Engineering

ENCM 509 - Introduction to Biometric Systems Design

Laboratory Experiment #8
Bayesian Networks

1 Introduction

This laboratory exercise is an introduction to Bayesian or belief networks that offer causal interpretation of decision making in terms of probability. We will investigate the simulation and inference on the Bayesian Networks.

Kevin Murphy developed a Matlab toolbox called BNT, which can be found at `http://code.google.com/p/bnt/`

The package `FullBNT` can be found in the Windows directory on the "N:" drive *\ENCM\509\lab8*.

2 Bayesian Belief Network toolbox

A Bayesian Network is a directed acyclic graph derived from a causal network of nodes corresponding to some variables represented by conditional probability of their occurrence. The nodes corresponding to causes are the parent nodes, and the effects of the causes are represented by child nodes. The edges between nodes are directed from a parent to a child node. In this laboratory, we will use the package to encode a Bayesian network and perform inference (computation of probabilities given some evidences on the variables).

To install the BNT package, follow these steps:

1. Open Matlab
2. Change your working directory to *\ENCM\509\lab8\FullBNT*.
3. Type in the Matlab screen:

   ```
   addpath(genpathKPM(pwd))
   ```

4. Type in 'test_BNT'
5. Type in 'clear all'

Some errors may appear in the output after running the test but these should not affect the rest of the lab and can be ignored.

3 Example 1

First let us consider a network shown in Fig. 1. The nodes of the network represent 4 variables:

- Winter - The probability that there is a winter season.
- Influenza - The probability that a person got an influenza.
- Cold - The probability that a person got a cold.
- Fever - The probability that a person has fever.

Each node (variable) is assigned with a table which gives the conditional probabilities for child node variable in columns, given the value of the parent node variable in rows. To define the structure of the network, you can use the following code.

```
N = 4; %the total number of nodes or random variables
W = 1; %the fiirt node called W (Winter)
I = 2;
C = 3;
F = 4; %each node is assigned a number
dag = zeros(N,N); %define a NxN matrix which  specifies the graph structure
dag(W,[I C]) = 1; %W is a parent of both I and C
dag(I, F) = 1; %I is a parent to F
dag(C, F) = 1; %C is a parent to F
discrete_nodes = 1:N; %all node take discrete values.
node_sizes = [2 2 2 2]; % each node  takes 2 values (e.g. True and False).
```

Next, the Bayesian network can be created as follows.

```
onodes=[]; %Specifying which nodes are the observed ones (here none)
bnet=mk_bnet(dag,node_sizes,'discrete',discrete_nodes,'observed',onodes);
```

Now, the conditional probability tables (CPTs) must be specified. These tables must be reordered into a vector, column by column. The example is shown below (here, $False = 1$ and $True = 2$).

```
bnet.CPD{W} = tabular_CPD(bnet, W, [0.5 0.5]);
bnet.CPD{C}= tabular_CPD(bnet, C, [0.8 0.2 0.2 0.8]);
bnet.CPD{I} = tabular_CPD(bnet, I, [0.5 0.9 0.5 0.1]);
bnet.CPD{F} = tabular_CPD(bnet, F, [1 0.1 0.1 0.01 0 0.9 0.9 0.99]);
```

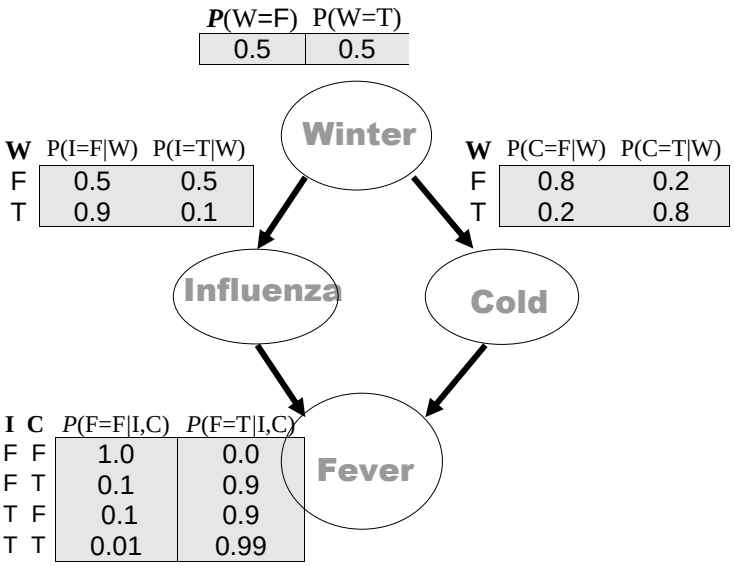

Figure 1: Bayesian Network with 4 nodes.

Exercise 1: Inference in the network

Having created the Bayesian Network, we can now use it for inference. For this exercise, we will use the junction tree engine, the exact inference algorithms. Consider example below: let the evidence of Fever be True.

```
 engine = jtree_inf_engine(bnet);
% you can also use:   engin =  var_elim_inf_engine(bnet);
 evidence = cell(1,N);% the size of the evidence vector
 evidence{F} = 2;% the evidence of Fever is True
```

```
[engine, loglik] = enter_evidence(engine, evidence);
marg = marginal_nodes(engine, I);
%calculate the updated value of probability for Influenza
```

The result will be an array of probabilities for Influenza. To narrow the values down, you can type:

```
marg.T
```

which returns the value $P(Influenza|Fever = True)$.

You can also and check for

```
marg.T(2)
```

and it returns the result of $P(Influenza = True|Fever = True)$.

Perform calculations (by hand) of the probabilities that Influenza is True, and then that it is False, given the evidence that fever is True. Compare the results to the ones provided by Matlab.

Exercise 2: more inference

Let us add evidence for the node Cold as well:

```
evidence{C} = 2;
[engine, loglik] = enter_evidence(engine,evidence);
```

Then

```
marg = marginal_nodes(engine, I);
```

returns the updated values of probabilities. For example,

```
marg.T
```

returns the value $P(Influenza|Cold = True, Fever = True)$, and

```
marg.T(2)
```

returns the result of $P(Influenza = True|Cold = True, Fever = True)$.

Perform calculations (by hand) of the above probabilities that Influenza is True, and then that it is False, given the evidence that Fever and Cold are both True. Compare the result to the one derived by Matlab.

Exercise 3: Computing joint probability

We can compute the joint probability on a set of nodes. Calculate the joint probability of variables I, C and F after updating of probabilities upon an evidence. Set, for example, $W = False$, and then use $m = marginalnodes(engine, [CIF])$ as shown in the code below:

```
evidence = cell(1,N);
[engine, ll] =enter_evidence(engine, evidence); % no evidence
m = marginal_nodes(engine, [I C F]); %it returns m as a structure.
%Its T field is a 3-dimensional array that contains
%joint probability on the specified nodes.
```

Here, $m.T$ returns $T(i, j, k)$, where i is the value of I, j is the value of C, and k is the value of F. So $T(1, 1, 2)$ will be the result of $P(I = 1, C = 1, F = 2)$. Note that $P(I = 1, C = 1, F = 2) = 0$, since it is impossible to have Fever if both Influenza and Cold are false.

Perform calculations (on paper using the formulas) of the above probabilities that Influenza is True, and then that it is False, given the evidence that Fever and Cold are both True. Compare the result to the one returned by Matlab.

4 Example 2

Consider the Bayesian Network as shown in Figure 2.

Exercise 4: Defining the network

Create a Matlab code that define this network: define the number of nods, indexes, graph structure, node sizes, construct the networks and set the CPT for each node.

Exercise 5: inference

Find the joint probability (on paper): P(Season=High,Flight=False,SARS =False, Fever=True, Cough=True)

Perform the adequate calculations using Matlab. Set the inference engine to

```
engine = var_elim_inf_engine(bnet);
```

define the size of the evidence vector, but do not define the evidence of nodes. Use

```
[engine, loglik] = enter_evidence(engine, evidence);
marg = marginal_nodes(engine, [Season Flight SARS Fever Cough])
```

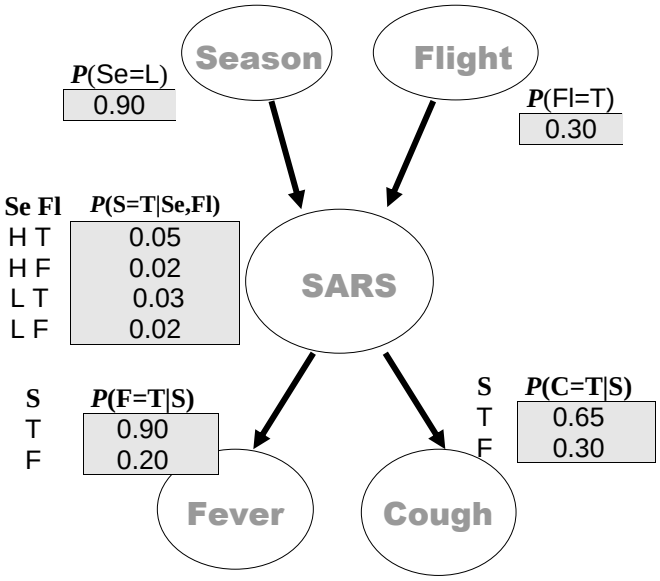

Figure 2: Sample Bayesian Network.

This returns all joint probabilities. To return a particular value, for example, for $Season = High(2)$, $Flight = False(2)$, $SARS = False(2)$, $Fever = True(1)$, and $Cough = True(1)$, use:

```
[engine, loglik] = enter_evidence(engine, evidence);
marg.T(2,2,2,1,1)
```

Compare the results calculated on paper, and the Matlab results.

Exercise 6: more inference

Perform inference on the network, given the evidence for one of the root nodes, the middle node and one of the leaf node. For example, given the evidence that $W = False$, find the probability that $Cough = True$. Do same for node $Flight$, $SARS$, $Fever$ and $Cold$. Perform calculations on paper, and then using Matlab.

In Matlab, set the the inference engine, assuming a variable X (any of the given five) is being an evidence, given its value x, and the variable Y for which the marginal distribution is being calculated:

```
engine = var_elim_inf_engine(bnet);
evidence = cell(1,N);
evidence{X} = x;
[engine, loglik] = enter_evidence(engine, evidence);
marg = marginal_nodes(engine, Y);
format short e
marg.T
```

5 Learning/Training Bayesian Network

Here we will discuss how to train/learn both the structure of the network, and the parameters of the probability distribution. We will generate a dataset using a predetermined Bayesian network (use the original example with 4 nodes W, I, C, F) so we can learn another network. The network obtained through the training should be close enough or the same as in the original example.

Thus, first, you have to define the network *bnet* with nodes W, I, C, F, by reusing some code from the first exercise: define the number of nodes ($N = 4$), the structure ($dag = zeros(N, N)$), the nodes and provide the CPDs and structure.

Next, we have to have a dataset which will be used by the BNT to find a network structure. The fragment below generates the dataset.

```
nsamples=1000;
samples = cell(N2, nsamples);
for i=1:nsamples
 samples(:,i) = sample_bnet(bnet);
end
data = cell2num(samples);
```

Here, "samples" is the matrix which will contain the dataset. "nsamples" is the number of samples we will generate. Thus, we are sampling the Bnet to obtain "nsamples" (number of samples); "data" is a variable that contain the dataset and is ready for using in learning.

Now, define the new network, $bnet2$ with nodes $W2, I2, C2, F2$, by reusing some code from the first exercise:

```
N2 = 4;
C2 = 1;
S2 = 2;
R2 = 3;
W2 = 4;
```

We will now discuss how to learn the structure of a Bayesian network. The code below is setting the properties of the network we will generate.

```
node_sizes = ones(1,N);
node_sizes(1,1) = 2;
node_sizes(1,2)= 3
node_sizes(1,3) = 2;
node_sizes(1,4) = 3;
```

This information can be obtained from the dataset as follows.

```
data = cell2num(samples);
order = [W2 I2 C2 F2];
max_fan_in = 2;
dag2 = learn_struct_K2(data, node_sizes, order, 'max_fan_in', max_fan_in);
```

Here, the array "data" contains the dataset; the order of the nodes is set and it controls the structure of the graph; max_fan_in means the maximum number of parents that node is allowed to have. We set it to 2, i.e. any node can only have a maximum of 2 parents. We learn the structure of the Bayesian network using K2 algorithm. There are other algorithms in the BNT.

Next, we initialize the network.

```
bnet2 = mk_bnet(dag2, node_sizes);
seed = 4;
rand('state', seed);
bnet2.CPD{W2} = tabular_CPD(bnet2, W2);
bnet2.CPD{C2} = tabular_CPD(bnet2, C2);
bnet2.CPD{I2} = tabular_CPD(bnet2, I2);
bnet2.CPD{F2} = tabular_CPD(bnet2, F2);
```

The code above sets the seed to initialize the parameters or CPD of each node. It uses random function, and also initialize tabular CPD for $W2, C2, I2$ and $F2$. Next, we learn the parameter of the new Bayesian network and its CPT called CPT3:

```
bnet2 = learn_params(bnet2, data);
CPT3 = cell(1,N2);
for
i=1:N2
  S=struct(bnet2.CPD{i});
  CPT3{i}=S.CPT;
end
```

Exercise 7: generate the network

Set the number of samples at $nsamples = 30$. Generate the network of 4 nodes W, I, C, F as instructed above, and compare with the original one. You can print out the network structure using $bnet.dag$ and $bnet2.dag$ and their CPTs using $bnet.CPD$ and $bnet2.CPD$.

Next, set $nsamples = 1000$ and then $nsamples = 5000$. Retrain the network and compare the results again. Draw conclusions.

6 Laboratory Report and Code (10 marks)

In this report, include:

- The results of calculations of the probabilities by hand and using Matlab for each exercise (Exercises 1 - 7 : 1 mark each).
- Answers to the questions in all Exercises and make conclusions (1 mark).
- Listing of the `.m` files and data files you created (2 marks).

7 Acknowledgments

The Bayesian Network Toolbox (BNT) was developed by Kevin Murphy (UBC) and is available as a freeware.

Svetlana Yanushkevich
March 13, 2019