CS 486/686 Assignment 2
Fall 2022
(86 marks)

Blake VanBerlo

Due Date: 11:59 PM ET on Tuesday, October 25, 2022

Instructions

- Submit the signed academic integrity statement any written solutions in a file to the Q0 box in the A2 project on Crowdmark. **(5 marks)**.
- Submit your written answers o questions 1 and 2 as PDF files to the Q1 and Q2 boxes respectively in the A2 project on Crowdmark. I strongly encourage you to complete your write-up in LaTeX, using this source file. If you do, in your submission, please replace the author with your name and student number. Please also remove the due date, the Instructions section, and the Learning goals section. Thanks!
- Submit any code to `Marmoset` at https://marmoset.student.cs.uwaterloo.ca/. Be sure to submit your code to the project named `Final`.
- No late assignment will be accepted. This assignment is to be done individually.
- Lead TAs:
  - Christopher Risi (cjrisi@uwaterloo.ca)
  - Dake Zhang (dake.zhang@uwaterloo.ca)

  The TAs' office hours will be scheduled and posted on LEARN and Piazza.

Learning goals

Bayesian Networks

- Understand independence and conditional independence relationships in Bayesian networks.

Variable Elimination Algorithm

- Define factors. Manipulate factors using the operations restrict, sum out, multiply and normalize.
- Trace the execution of and implement the variable elimination algorithm for calculating a prior or a posterior probability given a Bayesian network.

1 Three Proofs (26 marks)

(a) Recall that random variables X and Y are conditionally independent, given a third variable Z if and only if:

$$P(X|Y \wedge Z) = P(X|Z) \tag{1}$$

$$P(Y|X \wedge Z) = P(Y|Z) \tag{2}$$

$$P(X \wedge Y|Z) = P(X|Z)P(Y|Z) \tag{3}$$

Show that Equation 3 follows from Equations 1 and 2. Justify each step of your proof.

Hint: you may find some of the probability rules from Lecture 6 to be useful.

Marking Scheme: (6 marks)

- (4 marks) Correct proof
- (2 marks) Each step is justified

Solutions:

$P(X|Y \wedge Z) = P(X \wedge Y \wedge Z)/P(Y \wedge Z) = P(X|Z)$ - Conditional Probability from equation 2

$P(X \wedge Y|Z) = P(X \wedge Y \wedge Z)/P(Z)$ - Conditional Probability

$= P(X \wedge Y \wedge Z)/P(Y \wedge Z) * P(Y \wedge Z)/P(Z)$ - Alegebra

$= P(X|Z) * P(Y \wedge Z)/P(Z)$ - Substitute Equation 2

$= P(X|Z)P(Y|Z)$ - Conditional Probability

(b) The *Markov blanket* of random variable X in a Bayesian network consists of the set of random variables that are parents, children, or parents of children of X. For a Bayesian network with random variables $\mathcal{S}$, let $B = \{B_1, B_2, ..., B_n\} \subset \mathcal{S}$ be the Markov blanket for $X \in \mathcal{S}$. See Figure 1 for an example.

Show that $X \perp\!\!\!\perp Y \,|\, B, \forall\, Y \in \mathcal{S} \setminus B$. In other words, show that X is conditionally independent of every other variable in the network, conditioned on its Markov blanket.

Hint: Consider using the fundamental conditional independence relationships discussed in Lecture 8.

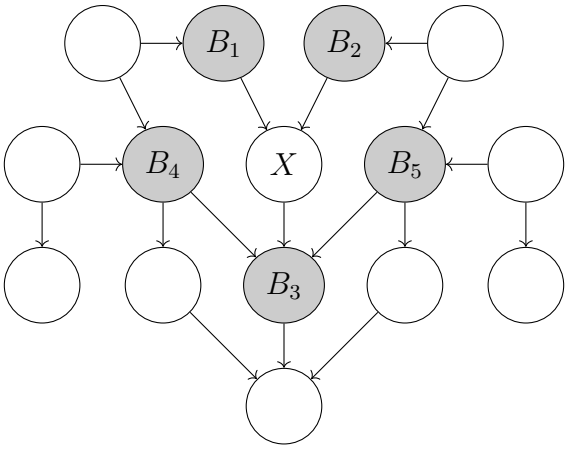


FIGURE 1: An example of a Bayesian network with its Markov blanket shaded grey.

Marking Scheme: (10 marks)

- (8 marks) Correct proof
- (2 marks) Proof is succinct and easy to understand

Solutions: For X to be conditionally independent with any node N given B , there should not be a path from X to N that does not go through B. Since B includes all children and parents of X, this holds for all N and thus X is conditionally independent of every variable in the network,conditioned on its Markov blanket

(c) Not all the random variables in a Bayesian networks are always required to answer a probabilistic query. In fact, all variables that are not ancestors of query variables or evidence variables are irrelevant to the query. Let $Q = \{Q_1, \ldots, Q_m\}$ be the set of query variables and $e = \{e_1, \ldots, e_n\}$ be the set of evidence variables. Prove that the Variable Elimination Algorithm (VEA) returns the same distribution for some query $P(Q|E)$ if all irrelevant variables are pruned from the Bayesian network.

Hint: try a direct proof. Show that a particular ordering of hidden variable elimination results in the same final distribution returned by the normalization operation.

Marking Scheme: (10 marks)

- (8 marks) Correct proof
- (2 marks) Proof is succinct and easy to understand

Solutions: TODO

2 The Variable Elimination Algorithm (55 marks)

You will implement the variable elimination algorithm to perform inference in Bayesian networks with discrete random variables.

We have provided four Python files. Please read the detailed comments in the provided files carefully.

1. `factor.py` Defines a factor class for VEA. **Do not change this file.**
2. `vea.py` Contains empty functions for VEA operations. You will implement all of the functions in this file. **Do not change the function signatures.**
3. `example.py` Provides a demonstration of how to define the factors for a Bayesian network (for the Holmes scenario) and execute the variable elimination algorithm.

Here are some tips for implementing the variable elimination algorithm:

1. `restrict()`: Consider using splicing operations or `take`.
2. `multiply()`: Consider using the `numpy` broadcasting rules to multiply multidimensional arrays of different shapes.
3. `sum_out()`: Consider using the `sum` operation.
4. You may find it helpful to print the VEA operations and intermediate factors. See Appendix A for tips on printing the output of VEA. Note that this is optional.

Please complete the following tasks.

(a) Implement the empty functions in `vea.py`. Zip and submit this file only on Marmoset.

Marking Scheme: (49 marks)

Unit tests:

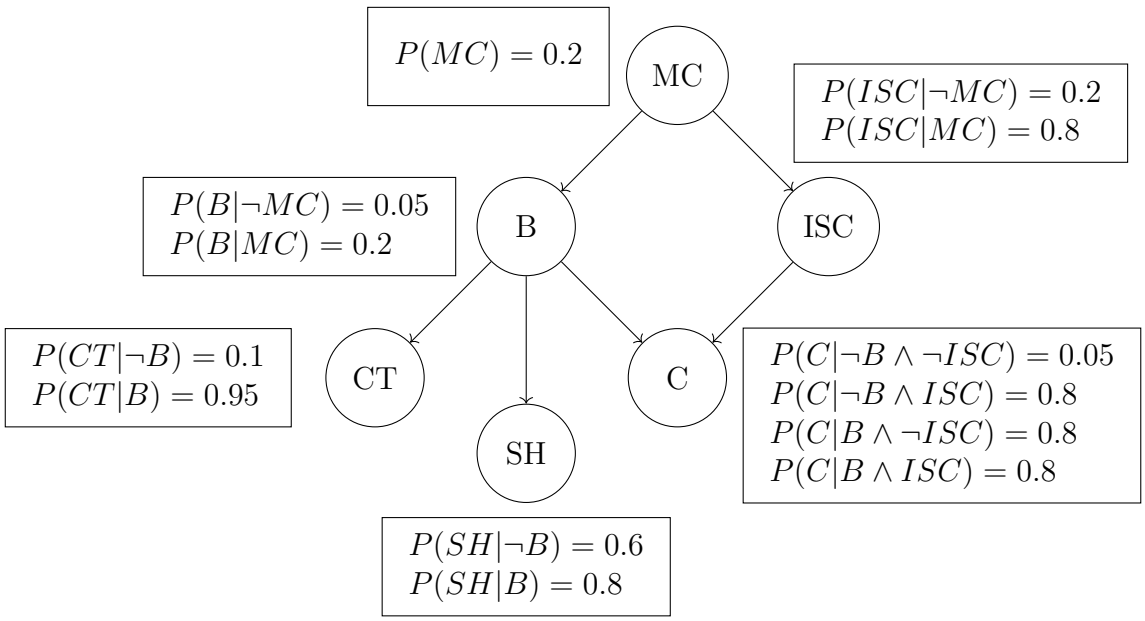
- `normalize`
  (1 public test + 2 secret tests) × 1 mark = 3 marks
- `restrict`
  (1 public test + 2 secret tests) × 2 marks = 6 marks
- `sum_out`
  (1 public test + 2 secret tests) × 2 marks = 6 marks

- `multiply`
  (1 public test + 7 secret tests) × 2 marks = 16 marks
- `vea`
  (1 public test + 5 secret tests) × 3 marks = 18 marks

Solutions: Code submiited.

(b) Below is a Bayesian network that appears in a recent review of uncertainties in Bayesian networks with discrete variables (Rohmer, (2020)). The network is a probabilistic model of brain cancer diagnosis. Note that the example is meant to be illustrative, as the network is simple and the conditional probability tables are fictional. The random variables are defined below:

Random Variable	Definition
MC	Metastatic cancer
B	Brain tumour
CT	CT scan is positive for brain tumour
SH	Severe headache
ISC	Increased serum calcium
C	Patient falls into coma



Using your implementation from the previous question, execute the Variable Elimination Algorithm to determine the probability that a patient has a brain tumour, given that their blood work shows an increased calcium concentration, and they do not report having severe headaches. Eliminate hidden variables in reverse alphabetical order. For example, you would eliminate MC before ISC and CT before C. For each random variable's domain, consider `False` as 0 and `True` as 1.

Marking Scheme: (2 marks)

- (2 marks) Correct answer

Solutions: $P(B|ISC \wedge \neg SH) = 0.0667$

(c) The complexity of VEA depends on the *treewidth* of the Bayesian network, which is the maximum number of random variables in a factor after summing out a variable. The order in which hidden variables are eliminated impacts the treewidth, thereby impacting the complexity.

Use your implementation of VEA to evaluate the probability of metastatic cancer given that a patient's CT scan is positive for a brain tumour (i.e., $P(MC = 1|CT = 1)$). Do not prune the network. Execute VEA for each possible order of eliminating the random variables and note the treewidth in each case. Produce a histogram of the treewidths for each execution of VEA.

Hint: You might find `itertools.permutation()` useful in determining the possible permutations of a list of strings.

Marking Scheme: (4 marks)

- (3 marks) Plot is correct
- (1 mark) Plot is clear

Appendix A Printing the output of VEA

You might find it helpful to print the output of the intermediate factors created during the execution of VEA. In our solution code, we print out each factor along with the steps of the algorithm when the `verbose` argument in `vea()` is set to `True`. Below we provide guidance on how you could print the algorithm's intermediate steps to understand it and debug. Note that if you are asked to manually perform VEA on an exam, you will be asked to list your steps in a similar format.

Note that we are not marking your code for its console output — only for the values returned by each function.

Print out a factor as `f(A B E)`. Use `f` to denote the name of every factor. Print out the variables in **lexicographical order**. For example,

```
f(A B E)
```

After each operation (restrict, multiply, sum out, normalize), print out the resulting factor in a table. You can print out the columns and the rows in any order. For example,

```
B,   E,   Prob
True,  True,  0.9600
True,  False,  0.9500
False,  True,  0.2000
False,  False,  0.0100
```

Print out the **define** operation as shown below. Show the factors in any order.

```
Define factors f(A G) f(A W)
```

Print out the **restrict** operation as shown below.

```
Restrict f(B E) to B = True to produce f(E)
 E,   Prob
 True,  0.9600
 False,  0.9500
```

If you need to perform the restrict operation on multiple factors and on multiple variables for each factor, go through the factors in **lexicographical order**. For each factor, go through the variables in **lexicographical order**. Print out one operation per line.

The example below performs two restrict operation on `f(A B E)`, one restrict operation on `f(A G)`, and one restrict operation on `f(A W)`.

```
Restrict f(A B E) to A = True to produce f(B E)
 B,   E,   Prob
 True,  True,  0.9600
 True,  False,  0.9500
 False,  True,  0.2000
 False,  False,  0.0100

Restrict f(B E) to B = True to produce f(E)
 E,   Prob
 True,  0.9600
 False,  0.9500

Restrict f(A G) to A = True to produce f(G)
 G,   Prob
 True,  0.4000
 False,  0.6000

Restrict f(A W) to A = True to produce f(W)
 W,   Prob
 True,  0.8000
 False,  0.2000
```

Print out the **multiply** operation as shown below.

```
Multiply f(A B E) f(B) to produce f(A B E)
```

If you need to multiply more than two factors together, print out the operation on a single line. Show the factors in any order. For example,

```
Multiply f(E) f() f(E) f() f() to produce f(E)
 E,   Prob
 True,  0.0000
 False,  0.0001
```

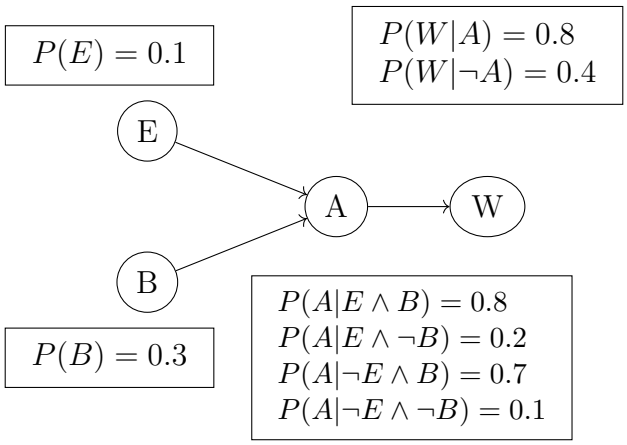
Print out the **sum out** operation as shown below.

```
Sum out W from f(W) to produce f()
Prob
1.0
```

Print out the **normalize** operation as shown below.

```
Normalize f(E) to produce f(E)
 E,   Prob
 True,  0.0003
 False,  0.9997
```

See below for an example output for the simplified version of the Holmes network given in Lecture 9, slide 25:



Suppose we would like to calculate the probability of a burglary occurring given that the alarm did not go off (i.e., $P(B|\neg A)$).

```
Define factors f(E) f(B) f(B E A) f(A W)
E Value
0 0.9000
```

```
1 0.1000

B Value
0 0.7000
1 0.3000

B E A Value
0 0 0 0.9000
0 0 1 0.1000
0 1 0 0.8000
0 1 1 0.2000
1 0 0 0.3000
1 0 1 0.7000
1 1 0 0.2000
1 1 1 0.8000

A W Value
0 0 0.6000
0 1 0.4000
1 0 0.2000
1 1 0.8000


Restrict f(B E A) to A = 0 to produce f(B E)
B E Value
0 0 0.9000
0 1 0.8000
1 0 0.3000
1 1 0.2000

Restrict f(A W) to A = 0 to produce f(W)
W Value
0 0.6000
1 0.4000

Sum out W from f(W) to produce f()
Value
1.0000

Multiply f(E) f(B E) to produce f(B E)
B E Value
0 0 0.8100
0 1 0.0800
1 0 0.2700
1 1 0.0200
```

```
Sum out E from f(B E) to produce f(B)
B Value
0 0.8900
1 0.2900

Multiply f(B) f() f(B) to produce f(B)
B Value
0 0.6230
1 0.0870

Normalize: f(B)
B Value
0 0.8775
1 0.1225
```

From the last factor, we see that $P(B|\neg A) = 0.1225$.