Advanced AI Programming Assignment 2: Bayesian Networks

Agrima Khanna

April 2, 2024

Contents

1 Coding Question

1.1 Implementation of the variable elimination algorithm

I implemented the 5 functions as instructed. The restrict, normalize, sumout, multiply, and inference methods.

a. **restrict:** The restrict function narrows down the factor to a specific value of a given variable.

```
def restrict(factor, variable, value):
    newShape = np.array(factor.shape)
    newShape[variable] = 1
    sliceList = [slice(None)] * factor.ndim
    sliceList[variable] = value

    return factor[tuple(sliceList)].reshape(newShape)
```

Listing 1: Restrict Method

b. **normalize:** The normalize function adjusts the factor's values so that they sum to 1, turning it into a proper probability distribution.

```
def normalize(factor):
    return factor / np.sum(factor.flatten())
```

Listing 2: Normalize Method

c. **sumout:** The sumout function marginalizes a variable out of the factor by aggregating (summing) the probabilities across all states of the variable being removed. This operation simplifies the factor to only those variables of interest.

```
def sumout(factor,variable):
    return np.sum(factor,axis=variable,keepdims=True)
```

Listing 3: Sumout Method

d. **multiply:** The multiply function combines two factors into a single factor by multiplying their corresponding probabilities. The result is a new factor that represents the joint probability distribution over the variables present in both input factors.

```
def multiply(factor1,factor2):
    return factor1*factor2
```

Listing 4: Multiply Method

e. **inference:** The inference function employs the restrict, multiply, and sumout functions to compute the conditional probability distribution of a set of query variables given some evidence. This function first applies the evidence to the relevant factors through restriction, then combines all factors by multiplication. It proceeds to eliminate all irrelevant variables (those not part of the query or evidence) by summing them out in a specified order. The final step involves normalizing the resultant factor to ensure it represents a valid probability distribution.

```
# inference function
def inference(factorList,queryVariables,orderedListOfHiddenVariables,
    evidenceList):
    # restrict factors
    for index in np.arange(len(factorList)):
        shape = np.array(factorList[index].shape)
        for evidence in evidenceList:
            if shape[evidence[0]] > 1:
                factorList[index] = restrict(factorList[index],
    evidence[0],evidence[1])
        shape = np.array(factorList[index].shape)

    # eliminate each hidden variable
    print ("Eliminating hidden variables\n")
    hiddenId = 5
    for variable in orderedListOfHiddenVariables:
        print("Eliminating {}".format(variables[variable]))

        # find factors that contain the variable to be eliminated
        factorsToBeMultiplied = []
        for index in np.arange(len(factorList)-1,-1,-1):
            shape = np.array(factorList[index].shape)
            if shape[variable] > 1:
                factorsToBeMultiplied.append(factorList.pop(index))

        # multiply factors
        product = factorsToBeMultiplied[0]
        for factor in factorsToBeMultiplied[1:]:
            product = multiply(product,factor)

        # sumout variable
        newFactor = sumout(product,variable)
        factorList.append(newFactor)
        shape = np.array(newFactor.shape)
        hiddenId = hiddenId + 1
        print("New factor: f{}({})={}\n".format(hiddenId,variables[
    shape>1],np.squeeze(newFactor)))

    # multiply remaining factors
    print ("Multiplying factors")
    answer = factorList[0]
    for factor in factorList[1:]:
        answer = multiply(answer,factor)
        shape = np.array(answer.shape)
        print("Answer pre-normalization: f{}({})={}\n".format(hiddenId
    +1,variables[shape>1],np.squeeze(answer)))

    # normalize answer
    print ("Normalizing ...")
    answer = normalize(answer)
    shape = np.array(answer.shape)
    print("Normalized answer: f{}({})={}\n".format(hiddenId+2,
    variables[shape>1],np.squeeze(answer)))
    return answer
```

Listing 5: Inference Method

I also have a helper function that prints the result of an inference.

```
def print_helper(query_variables, evidence_list, inference_result):
    print('Result:')
    print('P(', end='')

    # Print the query variables
    for i, query_variable in enumerate(query_variables):
        print(query_variable, end='' if i == len(query_variables) - 1 else
    ',')

    # Print the evidence list if it's not empty
    if evidence_list:
        print(' | ', end='')
        for evidence in evidence_list:
            variable, value = evidence
            # The value is expected to be a boolean, so print accordingly
            value_str = variable if value else f'~{variable}'
            print(value_str, end=',')
        print('\b', end='')  # Remove the last comma

    # Print the result
    print(') = {}'.format(np.squeeze(inference_result)))
```

Listing 6: Print Helper Function

1.2 Instructions for using the program

1.2.1 Requirements

- I used Python version 3.11.5.
- These necessary packages are listed in the `requirements.txt` file included with the program. To install these packages, use the command `pip install -r requirements.txt` in your terminal or command prompt.

1.2.2 Starting the Program

To start the program, navigate to the directory containing the program files in your terminal and run the command `python BayesianNetwork.py`. The program will then prompt you for input to begin the query process.

1.2.3 Entering Queries

The program offers two modes of operation: default queries and custom queries. Follow these steps to enter your query:

1. **Default Query**
   Enter `1` if you want to compute a default query. You will then be asked to select from a list of pre-defined queries(Figure 1).

2. **Custom Query**
   Enter `2` to input a custom query (Figure 2). You will need to:

```
(venv) agrimakhanna@MacBook-Pro-180 PA_2 % python BayesianNetwork.py
Welcome to the Bayesian Network Inference Program

Pr(OC)=[0.2 0.8]

Pr(Trav)=[0.95 0.05]

Pr(Fraud|Trav)=[[0.996 0.004]
 [0.99  0.01 ]]

Pr(FP|Fraud,Trav)=[[[0.99 0.01]
  [0.9  0.1 ]]

 [[0.1  0.9 ]
  [0.1  0.9 ]]]

Pr(IP|OC,Fraud)=[[[0.999 0.001]
  [0.949 0.051]]

 [[0.9   0.1  ]
  [0.85  0.15 ]]]

Pr(CRP|OC)=[[0.99 0.01]
 [0.9  0.1 ]]

Enter 1 for default query or 2 for custom query: 1
1. P(Fraud)
2. P(Fraud|FP,~IP,CRP)
3. P(Fraud|FP,~IP,CRP,Trav)
4. P(Fraud|IP)
5. P(Fraud|IP,CRP)
Enter query number: 3
Restricting factors

f0(['OC'])=[0.2 0.8]

f1([])=0.05

f2(['Fraud'])=[0.99 0.01]

f3(['Fraud'])=[0.9 0.9]
```

Figure 1: Default Query

```
Pr(OC)=[0.2 0.8]

Pr(Trav)=[0.95 0.05]

Pr(Fraud|Trav)=[[0.996 0.004]
 [0.99  0.01 ]]

Pr(FP|Fraud,Trav)=[[[0.99 0.01]
  [0.9  0.1 ]]

 [[0.1  0.9 ]
  [0.1  0.9 ]]]

Pr(IP|OC,Fraud)=[[[0.999 0.001]
  [0.949 0.051]]

 [[0.9   0.1  ]
  [0.85  0.15 ]]]

Pr(CRP|OC)=[[0.99 0.01]
 [0.9  0.1 ]]

Enter 1 for default query or 2 for custom query: 2
```

Figure 2: Custom Query

```
Enter 1 for default query or 2 for custom query: 2
Enter query variables separated by commas: Fraud
```

Figure 3: Enter Query Variables

```
Enter 1 for default query or 2 for custom query: 2
Enter query variables separated by commas: Fraud
Enter evidence variables separated by commas: IP,true,CRP,false,Trav,true,FP,false
```

Figure 4: Enter Evidence Variables

(a) Enter the query variables separated by commas (Figure 3). Variable names are case senstive and must be entered as Fraud, Trav, FP, OC,CRP, IP.

(b) Provide the evidence variables, also separated by commas. When prompted for evidence variables, enter each variable's name followed by its state. States are represented by either `true` or `True` for 'true' or `false` or `False` for 'false'. Separate each variable-state pair with a comma. Do not insert spaces between the variable, state, or comma (Figure 4). For example, to indicate that variable `FP` is true and variable `CRP` is false, you would enter `FP,true,CRP,false`. Variable names are case senstive and must be entered as Fraud, Trav, FP, OC,CRP, IP.

1.2.4 Reviewing Output

After entering your query, the program will process the information using the variable elimination algorithm and output the results, which include the probability distribution of the query variables given the evidence (Figure 5).

1.2.5 Running Additional Queries

Upon completion of a query, you can run additional queries by responding `Y` (Figure 6) when asked if you want to do another query. To exit the program, respond with `N` (Figure 7).

2 Fraud Detection System

2.1 Question (a): Bayes Network

Based on the given information, I have designed the resulting Bayes Network (Figure 8).

2.2 Question (b)

1. We need compute the value of P(Fraud). I have verified my answer with the output from my program (Figure 9).

$$\begin{aligned} \text{P(Fraud)} &= \text{P(Fraud|Trav)P(Trav)} + \text{P(Fraud|}\neg\text{Trav)P(}\neg\text{Trav)} \\ &= 0.01 \times 0.05 + 0.004 \times 0.95 \\ &= 0.0043 \end{aligned}$$

```
Enter 1 for default query or 2 for custom query: 2
Enter query variables separated by commas: Fraud
Enter evidence variables separated by commas: IP,true,CRP,false,Trav,true,FP,false
Eliminating hidden variables

Eliminating OC
New factor: f6(['Fraud'])=[0.072198 0.118098]

Multiplying factors
Answer pre-normalization: f7(['Fraud'])=[0.0495 0.0005]

Answer pre-normalization: f7(['Fraud'])=[4.95e-03 5.00e-05]

Answer pre-normalization: f7(['Fraud'])=[3.573801e-04 5.904900e-06]

Normalizing ...
Normalized answer: f8(['Fraud'])=[0.98374582 0.01625418]

Result:
P(Fraud | IP,~CRP,Trav,~FP) = [0.98374582 0.01625418]
Do you want to do another query? (Y/N):
```

Figure 5: Custom Query Output

```
Do you want to do another query? (Y/N): Y
```

Figure 6: Additional Queries

```
Do you want to do another query? (Y/N): N
Thank you for using the program. Goodbye!
```

Figure 7: Exiting the Program

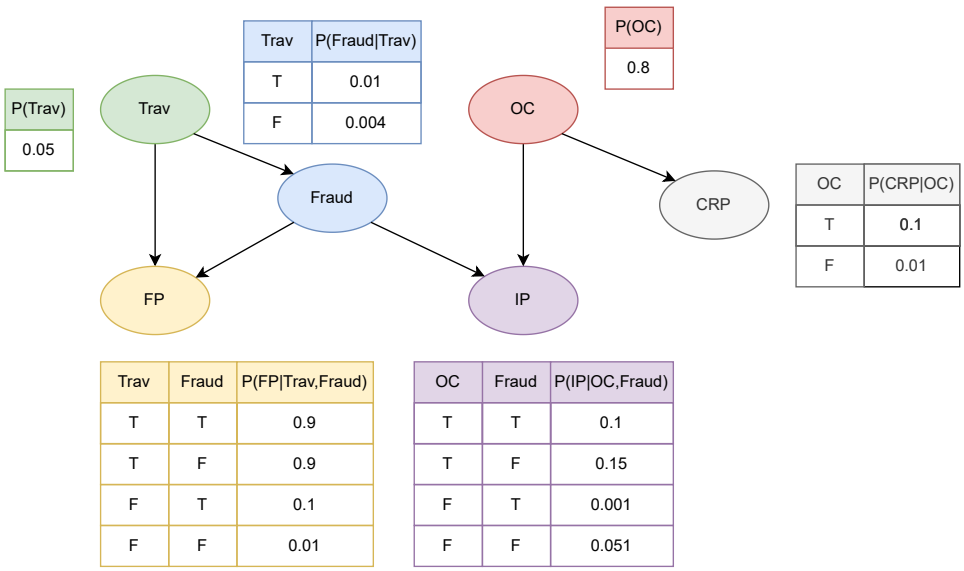

Figure 8: Bayes Network

```
Eliminating Trav
New factor: f6(['Fraud' 'FP'])=[[9.41688e-01 5.40120e-02]
 [3.47000e-03 8.30000e-04]]

Eliminating FP
New factor: f7(['Fraud'])=[0.9957 0.0043]

Eliminating IP
New factor: f8(['OC' 'Fraud'])=[[1. 1.]
 [1. 1.]]

Eliminating OC
New factor: f9(['Fraud' 'CRP'])=[[0.918 0.082]
 [0.918 0.082]]

Eliminating CRP
New factor: f10(['Fraud'])=[1. 1.]

Multiplying factors
Answer pre-normalization: f11(['Fraud'])=[0.9957 0.0043]

Normalizing ...
Normalized answer: f12(['Fraud'])=[0.9957 0.0043]

Result:
P(Fraud) = [0.9957 0.0043]
Do you want to do another query? (Y/N):
```

Figure 9: Prior Probability: P(Fraud)

2. Given the conditions that the transaction is foreign (FP), not an internet purchase (¬IP), and that the cardholder has recently purchased computer-related accessories (CRP), the query formed is $P(\text{Fraud} \mid \text{FP}, \neg\text{IP}, \text{CRP})$ and the probability that the current transaction is fraudulent is approximately 0.01431 (Figure 10).

   Given:

$$f_1(OC) = P(OC) = \begin{cases} 0.8 & \text{if true} \\ 0.2 & \text{if false} \end{cases}$$

$$f_2(Fraud, Trav) = P(Fraud|Trav) = \begin{cases} 0.01 & \text{if both true} \\ 0.004 & \text{if Trav is true and Fraud is false} \\ 0.99 & \text{if Trav is false and Fraud is true} \\ 0.996 & \text{if both false} \end{cases}$$

$$f_3(Trav) = P(Trav) = \begin{cases} 0.05 & \text{if true} \\ 0.95 & \text{if false} \end{cases}$$

$$f_4(OC) = P(CRP|OC) = \begin{cases} 0.1 & \text{if true} \\ 0.01 & \text{if false} \end{cases}$$

$$f_5(OC, Fraud) = P(\neg ip|OC, Fraud) = \begin{cases} 0.9 & \text{if OC and Fraud are true} \\ 0.85 & \text{if OC is true and Fraud is false} \\ 0.999 & \text{if OC is false and Fraud is true} \\ 0.949 & \text{if both false} \end{cases}$$

$$f_6(Trav, Fraud) = P(fp|Trav, Fraud) = \begin{cases} 0.9 & \text{if Trav and Fraud are true} \\ 0.9 & \text{if Trav is true and Fraud is false} \\ 0.1 & \text{if Trav is false and Fraud is true} \\ 0.01 & \text{if both false} \end{cases}$$

To eliminate the variable $Trav$:

$$\begin{aligned} f_7(Fraud) &= \text{sumout}_{Trav}\left[f_2(Fraud, Trav) \times f_3(Trav) \times f_6(Trav, Fraud)\right] \\ &= \begin{cases} 0.00083 & \text{if Fraud is true} \\ 0.05401 & \text{if Fraud is false} \end{cases} \end{aligned}$$

To eliminate the variable OC:

$$\begin{aligned} f_8(Fraud) &= \text{sumout}_{OC}\left[f_1(OC) \times f_4(OC) \times f_5(OC, Fraud)\right] \\ &= \begin{cases} 0.07400 & \text{if Fraud is true} \\ 0.06990 & \text{if Fraud is false} \end{cases} \end{aligned}$$

Then, the probability of Fraud given fp, $\neg ip$, and crp is calculated as:

$$P(\text{Fraud}|fp, \neg ip, crp) = \alpha \times f_7(\text{Fraud}) \times f_8(\text{Fraud}) = 0.01431$$

where α is a normalizing constant defined as:

$$\alpha = \frac{1}{f_7(\text{Fraud}) \times f_8(\text{Fraud}) + f_7(\neg\text{Fraud}) \times f_8(\neg\text{Fraud})}$$

```
Eliminating hidden variables

Eliminating Trav
New factor: f6(['Fraud'])=[0.054012 0.00083 ]

Eliminating OC
New factor: f7(['Fraud'])=[0.073998 0.069898]

Multiplying factors
Answer pre-normalization: f8(['Fraud'])=[3.99677998e-03 5.80153400e-05]

Normalizing ...
Normalized answer: f9(['Fraud'])=[0.98569217 0.01430783]

P(Fraud|FP,~IP,CRP)=[0.98569217 0.01430783]

Do you want to do another query? (Y/N): 
```

Figure 10: Posterior Probability: $P(\text{Fraud} \mid \text{FP}, \neg\text{IP}, \text{CRP})$

3. The probability that the transaction is fraudulent, given it is a foreign transaction, not an internet purchase, the cardholder purchased computer-related accessories in the past week, and is currently out of town on business, is found to be 0.00945. This new piece of information about the cardholder's travel status slightly decreases the likelihood of fraud compared to the initial assessment (Figure 11).

$$f_1(OC) = P(OC) = \begin{cases} 0.8 & \text{if true (t)} \\ 0.2 & \text{if false (f)} \end{cases}$$

$$f_2(Fraud) = P(Fraud|trav) = \begin{cases} 0.01 & \text{if trav is true (t)} \\ 0.99 & \text{if trav is false (f)} \end{cases}$$

$$f_3() = P(trav) = \begin{cases} 0.05 & \text{if trav true (t)} \\ 0.95 & \text{if trav false (f)} \end{cases}$$

$$f_4(OC) = P(crp|OC) = \begin{cases} 0.1 & \text{if OC is true (t)} \\ 0.001 & \text{if OC is false (f)} \end{cases}$$

$$f_5(OC, Fraud) = P(\neg ip|OC, Fraud) = \begin{cases} 0.9 & \text{if OC and Fraud are both true (tt)} \\ 0.85 & \text{if OC is true and Fraud is false (tf)} \\ 0.999 & \text{if OC is false and Fraud is true (ft)} \\ 0.949 & \text{if both are false (ff)} \end{cases}$$

$$f_6(Fraud) = P(fp|trav, Fraud) = \begin{cases} 0.9 & \text{if both trav and Fraud true (tt)} \\ 0.1 & \text{if both false (ff)} \end{cases}$$

To eliminate the variable OC, we sum over OC:

$$\begin{aligned} f_7(Fraud) &= \text{sumout}_{OC}[f_1(OC) \times f_4(OC) \times f_5(OC, Fraud)] \\ &= \begin{cases} 0.06990 & \text{if Fraud is true (t)} \\ 0.07400 & \text{if Fraud is false (f)} \end{cases} \end{aligned}$$

```
Eliminating hidden variables

Eliminating OC
New factor: f6(['Fraud'])=[0.073998 0.069898]

Multiplying factors
Answer pre-normalization: f7(['Fraud'])=[0.0495 0.0005]

Answer pre-normalization: f7(['Fraud'])=[0.04455 0.00045]

Answer pre-normalization: f7(['Fraud'])=[3.2966109e-03 3.1454100e-05]

Normalizing ...
Normalized answer: f8(['Fraud'])=[0.99054883 0.00945117]

P(Fraud|FP,~IP,CRP,Trav)=[0.99054883 0.00945117]

Do you want to do another query? (Y/N):
```

Figure 11: Posterior Probability: $P(\text{Fraud} \mid \text{FP}, \neg\text{IP}, \text{CRP}, \text{Trav})$

The posterior probability is then given by:

$$P(\text{Fraud}|fp, \neg ip, crp, trav) = \alpha \times f_2(\text{Fraud}) \times f_3() \times f_6(\text{Fraud}) \times f_7(\text{Fraud}) = 0.00945$$

where α is the normalizing constant:

$$\alpha = \frac{1}{\sum_{Fraud} f_2(\text{Fraud}) \times f_3() \times f_6(\text{Fraud}) \times f_7(\text{Fraud})}$$

4. To minimize the chance of an online purchase being flagged as fraud after stealing a credit card, a smart move would be to initially make a computer related purchase. This can create an impression in the fraud detection system that the cardholder possesses a computer, which in turn can lower the suspicion of fraud for subsequent online transactions. It's demonstrable that the probability of a transaction being marked as fraudulent declines when a computer-related purchase is recorded; specifically, the probability of fraud given an internet purchase drops from P(Fraud | IP) = 0.00696 (Figure 12) to P(Fraud | IP,CRP) = 0.00649 (Figure 13) if a computer-related purchase is also noted.

3 Scaling Up the Bayes Network

Bayesian networks offer a more sophisticated and well-organized method of modeling probabilistic relationships instead of preserving full Conditional Probability Tables . The key point is modularity, as above systems can be modified without serious changes. However, they are still partly scalable when new variables are added or the relationships between existing ones are changed. On the other hand, with the increase in the number of variables in the network and its complexity, the execution of exact inference becomes more difficult. Joint probability evaluation, i.e. answering the question, is more and more resources-intensive – this problem is known as 'curse of dimensionality' . Although

```
Eliminating hidden variables

Eliminating Trav
New factor: f6(['Fraud' 'FP'])=[[9.41688e-01 5.40120e-02]
 [3.47000e-03 8.30000e-04]]

Eliminating FP
New factor: f7(['Fraud'])=[0.9957 0.0043]

Eliminating OC
New factor: f8(['Fraud' 'CRP'])=[[0.072198 0.008002]
 [0.118098 0.012102]]

Eliminating CRP
New factor: f9(['Fraud'])=[0.0802 0.1302]

Multiplying factors
Answer pre-normalization: f10(['Fraud'])=[0.07985514 0.00055986]

Normalizing ...
Normalized answer: f11(['Fraud'])=[0.99303787 0.00696213]

P(Fraud|IP)=[0.99303787 0.00696213]

Do you want to do another query? (Y/N): 
```

Figure 12: Posterior Probability: $P(\text{Fraud} \mid \text{IP})$

```
Eliminating hidden variables

Eliminating Trav
New factor: f6(['Fraud' 'FP'])=[[9.41688e-01 5.40120e-02]
 [3.47000e-03 8.30000e-04]]

Eliminating FP
New factor: f7(['Fraud'])=[0.9957 0.0043]

Eliminating OC
New factor: f8(['Fraud'])=[0.008002 0.012102]

Multiplying factors
Answer pre-normalization: f9(['Fraud'])=[7.9675914e-03 5.2038600e-05]

Normalizing ...
Normalized answer: f10(['Fraud'])=[0.9935111 0.0064889]

P(Fraud|IP,CRP)=[0.9935111 0.0064889]

Do you want to do another query? (Y/N): 
```

Figure 13: Posterior Probability: $P(\text{Fraud} \mid \text{IP}, \text{CRP})$

Bayesian networks require less storage space compared to full CPT, the "curse" remains, and this overhead can be quite substantial for a large problem-scaling . In contrast, storing an entire CPT for a probabilistic model is much simpler conceptually — it is just a direct tabulation of joint probabilities for all combinations of variables . This exactly allows one to quickly answer queries by directly consulting the table, assuming it is possible to fit such a table into memory. However, the approach clearly does not scale: as the number of variables increases, the CPT grows exponentially to the number of variables, and even a small number of variables make the CPT impossibly big for storage or for simple table lookups . Besides, CPTs are not modular: retraining the model means recomputing the whole table, which is expensive and inconvenient.