CS 422 - Assignment 1

Bharat Gangwani

September 25, 2023

1 Question 1

X_3 : Sum of the outcomes of 3 fair six-sided dice
W : Winnings from the game
$A : X_3 \in \{2, 7, 12, 15\}$
$B : X_3 \in \{9, 18\}$
$C : X_3 \in \{5, 9, 10, 13\}$

$$\begin{aligned}
P(A) &= \frac{1}{216}(0 + 15 + 18 + 10) = \frac{43}{216} \\
P(B) &= \frac{1}{216}(25 + 1) = \frac{26}{216} \\
P(C) &= \frac{1}{216}(6 + 25 + 27 + 21) = \frac{79}{216}
\end{aligned}$$

$$\begin{aligned}
P(W = 1) &= P(A) \\
&= 0.19907 \\
P(W = -1) &= P(B) + P(C')P((A \cup B)') \\
&= 0.55202 \\
P(W = 2) &= P((A \cup B)')P(C) \\
&= (1 - P(A \cup B))P(C) \\
&= 0.24891
\end{aligned}$$

$$E[W] = 1(0.19907) + (-1)(0.55202) + 2(0.24891) = 0.14487$$
$$P(W \in \{1, 2\}) = 0.24891 + 0.19907 = 0.44798$$

I will play this game because the $E[W] > 0$.

2 Question 2

$$\begin{aligned}
&\min_{L,S} 800L + 600S \\
&\text{s.t. } L \leq 11, S \leq 8 \\
&40L + 30S \geq 450 \\
&L + S \leq 14
\end{aligned}$$

I used the Simplex method within Excel solver to solve this problem. The optimal solution is $L = 11, S = 3$ with a cost of \$9000. A linear programming problem may be infeasible if the constraints are too restrictive or unbounded (not a well defined problem). Simplex may also have to visit a very large number of nodes for problems with many constraints before it terminates.

3 Question 3

$$\#d = 3$$

$$\#c = 3$$

$$\#t = 4$$

$P(X = \widehat{x \mid G} = g) = \frac{\#(X=x,G=g)}{\#(G=g)}$ where X represents either Bob or Sonia's rating, $g \in \{d, c, t\}$ and $x \in \{1, 2, 3, 4, 5\}$.

s	$P(S = s \mid G = d)$	$P(S = s \mid G = c)$	$P(S = s \mid G = t)$
1	0.0	0.0	0.0
2	0.0	0.0	0.0
3	0.0	0.666	0.0
4	0.333	0.333	0.25
5	0.666	0.0	0.75

b	$P(B = b \mid G = d)$	$P(B = b \mid G = c)$	$P(B = b \mid G = t)$
1	0.0	0.0	0.0
2	0.0	0.0	0.25
3	0.0	0.0	0.5
4	0.666	0.333	0.25
5	0.333	0.666	0.0

4 Question 4

Nodes (All variables are binary)
T: Travelling
F: Fraudulent transaction
P: Foreign purchase
I: Online purchase
C: Computer owner
A: Purchased computer related accessory

```python
from pgmpy.models import BayesianNetwork
from pgmpy.factors.discrete import TabularCPD
from pgmpy.sampling import BayesianModelSampling
from pgmpy.inference import VariableElimination

# Define the Bayesian Network
fraudNet = BayesianNetwork([("T", "F"), ("T", "P"), ("F", "P"), ("F", "I"), ("C",
    "I"), ("C", "A")])
```

```python
# Define the CPDs
cpd_T = TabularCPD('T', 2, [[0.92], [0.08]])
cpd_C = TabularCPD('C', 2, [[0.40], [0.60]])

cpd_F = TabularCPD('F', 2, [[0.998, 0.98], [0.002, 0.02]], ["T"], [2])
cpd_A = TabularCPD('A', 2, [[0.999, 0.9], [0.001, 0.1]], ["C"], [2])

cpd_P = TabularCPD('P', 2, [[0.99, 0.9, 0.1, 0.1], [0.01, 0.1, 0.9, 0.9]], ['T',
    'F'], [2, 2])
cpd_I = TabularCPD('I', 2, [[0.999, 0.99, 0.989, 0.98], [0.001, 0.01, 0.011,
    0.02]], ['F', 'C'], [2, 2])

# Add CPDs to the model
fraudNet.add_cpds(cpd_T, cpd_C, cpd_F, cpd_A, cpd_P, cpd_I)

# Prior Probability of Fraud
samples = BayesianModelSampling(fraudNet).likelihood_weighted_sample(evidence = [],
    size = 100000, seed = 123)
print(samples["F"].value_counts(normalize = True))

inference = VariableElimination(fraudNet)
print(inference.query(['F'], evidence = {}))

samples = BayesianModelSampling(fraudNet).likelihood_weighted_sample(evidence =
    [("P", 0), ("I", 1), ("A", 0)], size = 100000, seed = 123)
total_weight = samples._weight.sum()
print(samples[samples["F"] == 1]["_weight"].sum()/total_weight)

inference = VariableElimination(fraudNet)
print(inference.query(['F'], evidence = {"P":0, "I":1, "A":0}))

samples = BayesianModelSampling(fraudNet).likelihood_weighted_sample(evidence =
    [("P", 0), ("I", 1), ("A", 0), ("T", 1)], size = 100000, seed = 123)
total_weight = samples._weight.sum()
print(samples[samples["F"] == 1]["_weight"].sum()/total_weight)

inference = VariableElimination(fraudNet)
print(inference.query(['F'], evidence = {"P":0, "I":1, "A":0, "T":1}))
```

The Bayesian network is shown in Figure 1. The likelihood weighted samples are shown in Table 1. The answers to questions 2 and 3, the unconditional and conditional probabilities, are shown in Table 2. Answers between the likelihood weighting approach and variable elimination are very similar. For part 3, the probability increases by almost 10-fold, suggesting travel is a strong indicator of fraudulent transactions.

Figure 1: Fraudulent transaction network

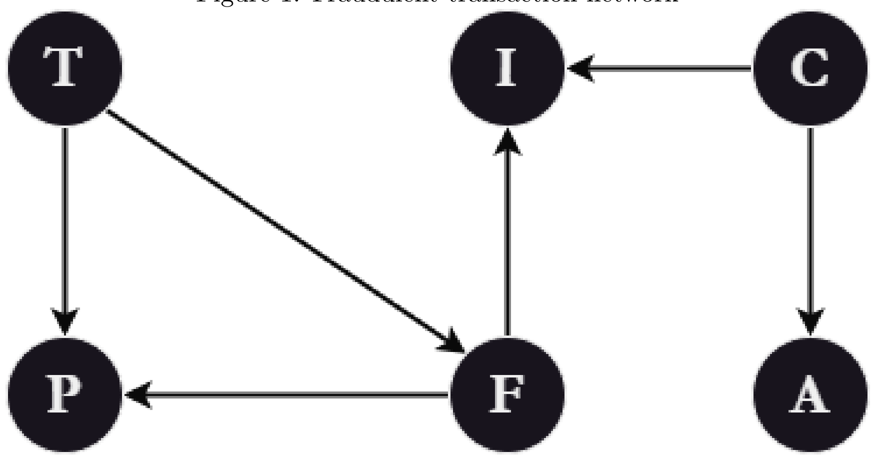

Table 1: 10 samples from likelihood weighting

T	F	P	I	C	A	_weight
0	0	0	1	0	0	0.000989
0	0	0	1	0	0	0.000989
0	0	0	1	0	0	0.000989
0	0	0	1	1	0	0.008910
0	0	0	1	0	0	0.000989
0	0	0	1	1	0	0.008910
1	0	0	1	0	0	0.000100
0	0	0	1	1	0	0.008910
0	0	0	1	1	0	0.008910
0	0	0	1	1	0	0.008910

5 Question 5

I assume $g0$ mentioned in the question stands for $g1$. Hence, evidence given: $S = s0$ and $G = g1$.
Forward Pass

$$\begin{aligned}
\max_{D,I,L} P(L, I, D, S = s0, G = g1) &= \max_{D,I,L} P(L|G = g1)P(G = g1|D, I)P(S = s0|I)P(D)P(I) \\
&= 0.9 * \max_{I} P(S = s0|I)P(I) \max_{D} P(G = g1|D, I)P(D) \\
&= 0.9 * \max_{I} P(S = s0|I)P(I) * \begin{bmatrix} 0.18 & 0.54 \end{bmatrix}' \\
&= 0.9 * \max_{I} \begin{bmatrix} 0.7 * 0.95 * 0.18 \\ 0.3 * 0.2 * 0.54 \end{bmatrix} \\
&= 0.9 * \max_{I} \begin{bmatrix} 0.1197 \\ 0.0324 \end{bmatrix} \\
&= 0.9 * 0.1197
\end{aligned}$$

Backward Pass

Table 2: Answers for parts 1-3

Probability	Likelihood weighting	Variable Elimination
$P(F=1)$	0.00333	0.0034
$P(F=1 \mid P=0, I=1, A=0)$	0.00518	0.0052
$P(F=1 \mid P=0, I=1, A=0, T=1)$	0.05113	0.0508

$I^* = i0, L^* = l1$

$$\begin{aligned}
\arg\max_D P(L=l1, I=i0, D, S=s0, G=g1) &= 0.9P(I=i0)P(S=s0|I=i0)\arg\max_D P(G=g1|D, I=i0)P(D) \\
&= \arg\max_D P(G=g1|D, I=i0)P(I=i0)P(D) \\
&= \arg\max_D \begin{bmatrix} 0.7 & 0.3 \end{bmatrix} \begin{bmatrix} 0.3*0.6 & 0.4*0.05 \\ 0.9*0.6 & 0.4*0.5 \end{bmatrix} \\
&= \arg\max_D \begin{bmatrix} 0.288 \\ 0.074 \end{bmatrix} \\
&= d0
\end{aligned}$$

Therefore, $(L=l1, I=i0, D=d0)$ are the MPE for $S=s0, G=g1$

6 Question 6

(a) States: $L1, L2, L3, L4$
Actions: $L1, L2, L3, L4$ (If state == action, then driver attempts to pick up customer)
Transition Probabilities:

$$\begin{bmatrix}
P(L'=j \mid L=i, A=\text{“}pickup\text{”}) & L1 & L2 & L3 & L4 \\
L1 & 0 & 0.4 & 0.2 & 0.4 \\
L2 & 0.6 & 0 & 0.4 & 0 \\
L3 & 0.6 & 0 & 0 & 0.4 \\
L4 & 0.4 & 0.6 & 0 & 0
\end{bmatrix}$$

$P(L'=j \mid L=i, A=\text{“}move\text{”}) = 1, \forall\, i, j$

Reward Matrix:

$$\begin{bmatrix}
R(L=i, L'=j, A=\text{“}pickup\text{”}) & L1 & L2 & L3 & L4 \\
L1 & 0 & 24 & 10.5 & 4.75 \\
L2 & 9 & 0 & 6.25 & -1 \\
L3 & 13.5 & -1 & 0 & 7.8 \\
L4 & 9.75 & 4 & -1 & 0
\end{bmatrix}$$

$$\begin{bmatrix}
R(L=i, L'=j, A=\text{“}move\text{”}) & L1 & L2 & L3 & L4 \\
L1 & 0 & -1 & -1.5 & -1.25 \\
L2 & -1 & 0 & -1.75 & -1 \\
L3 & -1.5 & -1 & 0 & -1.2 \\
L4 & -1.25 & -1 & -1 & 0
\end{bmatrix}$$

(b)

```
import numpy as np
findp = np.array([0.2, 0.6, 0.4, 0.7])
transProb = np.array([[0, 0.4, 0.2, 0.4],
                      [0.6, 0, 0.4, 0],
```

```
                  [0.6, 0, 0, 0.4],
                  [0.4, 0.6, 0, 0]])
fares = np.array([[0, 25, 12, 6],
              [10, 0, 8, 0],
              [15, 0, 0, 9],
              [11, 5, 0, 0]])
costs = np.array([[0, 1, 1.5, 1.25],
              [1, 0, 1.75, 1],
              [1.5, 1, 0, 1.2],
              [1.25, 1, 1, 0]])
profits = fares - costs

v = np.zeros(4)
q = np.zeros([4, 4])
gamma = 0.95

for i in range(100000):
    pickup = np.diag(findp*np.sum(transProb*(profits + gamma*v), axis = 1) +
        (1-findp)*gamma*v)
    move = -costs + gamma*v.reshape(4,1)
    np.fill_diagonal(move, 0)
    q = pickup + move
    v = np.max(q, axis=1)

# q(1)
# [[ 2.72 -1.   -1.5  -1.25 ]
# [-1.    4.74  -1.75 -1.   ]
# [-1.5  -1.    4.488 -1.2 ]
# [-1.25 -1.   -1.    4.41 ]]

# v(1)
# [2.72 4.74 4.488 4.41 ]

# q(2)
# [[5.653144 1.584  1.084   1.334  ]
# [3.503   8.494704 2.753   3.503  ]
# [2.7636  3.2636  8.33664 3.0636 ]
# [2.9395  3.1895  3.1895  8.28163 ]]

# v(2)
# [5.653144 8.494704 8.33664 8.28163 ]

print(np.argmax(q, axis=1))

# [0 1 2 3]
```

(c)

```
from pulp import *

gamma = 0.95

model = LpProblem("Taxi", LpMinimize)
v1 = LpVariable("V(L1)", 0, None, LpContinuous)
v2 = LpVariable("V(L2)", 0, None, LpContinuous)
```

```
v3 = LpVariable("V(L3)", 0, None, LpContinuous)
v4 = LpVariable("V(L4)", 0, None, LpContinuous)

model += v1 + v2 + v3 + v4

model += v1 >= 0.2*(0.4*(24 + gamma*v2) + 0.2*(10.5+gamma*v3) + 0.4*(4.75 +
    gamma*v4)) + 0.8*(gamma*v1)
model += v1 >= -1 + gamma*v2
model += v1 >= -1.5 + gamma*v3
model += v1 >= -1.25 + gamma*v4
model += v2 >= -1 + gamma*v1
model += v2 >= 0.6*(0.6*(9 + gamma*v1) + 0.4*(6.25 + gamma*v3)) + 0.4*(gamma*v2)
model += v2 >= -1.75 + gamma*v3
model += v2 >= -1 + gamma*v4
model += v3 >= -1.5 + gamma*v1
model += v3 >= -1 + gamma*v2
model += v3 >= 0.4*(0.6*(13.5 + gamma*v1) + 0.4*(7.8 + gamma*v4)) + 0.6*(gamma*v3)
model += v3 >= -1.2 + gamma*v4
model += v4 >= -1.25 + gamma*v1
model += v4 >= -1 + gamma*v2
model += v4 >= -1 + gamma*v3
model += v4 >= 0.7*(0.4*(9.75 + gamma*v1) + 0.6*(4 + gamma*v2)) + 0.3*(gamma*v4)

model.solve()

v = np.array([vi.varValue for vi in model.variables()])
print(v)

q = np.zeros([4, 4])
gamma = 0.95

for i in range(1):
    pickup = np.diag(findp*np.sum(transProb*(profits + gamma*v), axis = 1) +
        (1-findp)*gamma*v)
    move = -costs + gamma*v.reshape(4,1)
    np.fill_diagonal(move, 0)
    q = pickup + move

print(np.argmax(q, axis=1))

# [0 1 2 3]
```