Place your written answers in this text file

If you worked on this assignment with any other students, please indicate their names here:
Tarun Gulati
Shailaja Kapoor
Sridhar

Q1.

a) For values of N as 100, 1000 and 10,000 , the following were the values of fraction of samples of each flavor:
N = 100
Fraction of times lime was picked: 0.33
Fraction of times lemon was picked: 0.31
Fraction of times cherry was picked: 0.36

N = 1000
Fraction of times lime was picked: 0.284
Fraction of times lemon was picked: 0.329
Fraction of times cherry was picked: 0.387

N = 10000
Fraction of times lime was picked: 0.2944
Fraction of times lemon was picked: 0.3048
Fraction of times cherry was picked: 0.4008

Observation: As the number of observations increases, the fraction of samples of each flavor approach the actual probability of
that flavor as given in the Probability Distribution Table

b) The value En is an indication of the variation in the observed value of the probability and the actual probability of that flavor.
The value of En obtained for n = 100, 1000 and 10,000 is as follows:
[0.6386000000000002, 0.06494800000000002, 0.00545424]

This shows that the value decreases for higher values of number of samples drawn. Hence, the more the number of samples (N), the closer we get to the actual probability of drawing that sample. This is similar to the Weak Law of Large Numbers.

Q2.

The two-node Bayesian Network for this question has been implemented.
Here, we have a causal relationship between the Bin node and the Flavor node.
We are required to calculate
P(Bin | Flavor = 'Lime') *for every bin b(i).*
Using Bayes Theorem, we can convert this problem to:
P(b(i) | Flavor = 'Lime') *= [P(Flavor = 'Lime' | b(i))*P(b(i))] /P(Flavor = 'Lime')*
We can calculate P(Flavor = 'Lime') as
Sum(over all i) [P(Flavor='Lime' | b(i))]*P(b(i))
For the given probability distribution table, I calculated that P(Flavor = 'Lime') = 0.333
Using the formula stated above, I calculated the values for P(Bin | Flavor = 'Lime') for every bin b(i) as follows:

i	P(Flavor = 'Lime' | b(i))	P(b(i))	P(b(i) |Flavor = 'Lime')
1	0.0	0.1	0.0
2	1.0	0.1	0.303
3	0.0	0.1	0.0
4	0.33	0.1	0.1
5	0.67	0.1	0.20303
6	0.0	0.1	0.0
7	0.0	0.1	0.0
8	0.67	0.1	0.20303
9	0.33	0.1	0.1
10	0.33	0.1	0.1

The following is the result of using the enumerate_ask() function on passing the variable as 'Bin' and the evidence as 'Flavor' = 'Lime':

```
{'b10': 0.09909909909909909, 'b4': 0.09909909909909909, 'b5':
0.2012012012012012016, 'b6': 0.0, 'b7': 0.0, 'b1': 0.0, 'b2': 0.30030030030030025,
'b3': 0.0, 'b8': 0.2012012012012012016, 'b9': 0.09909909909909909}
```

My values obtained by hand-calculation are correct and are verified by the above output.

Q3.
The following was the output of the code for the third question:

```
P(Bin|Flavor = 'Lime') for original network:
{'b10': 0.09909909909909909, 'b4': 0.09909909909909909, 'b5': 0.20120120120120116,
'b6': 0.0, 'b7': 0.0, 'b1': 0.0, 'b2': 0.30030030030030025, 'b3': 0.0, 'b8':
0.20120120120120116, 'b9': 0.09909909909909909}
P(Bin|Flavor = 'Lime') for trained network with  100  samples:
{'b10': 0.14705882352941177, 'b4': 0.05882352941176469, 'b5': 0.1764705882352941,
'b6': 0.0, 'b7': 0.0, 'b1': 0.0, 'b2': 0.3529411764705882, 'b3': 0.0, 'b8':
0.20588235294117643, 'b9': 0.05882352941176469}
P(Bin|Flavor = 'Lime') for trained network with  1000  samples:
{'b10': 0.13157894736842105, 'b4': 0.08187134502923975, 'b5': 0.18128654970760233,
'b6': 0.0, 'b7': 0.0, 'b1': 0.0, 'b2': 0.3567251461988304, 'b3': 0.0, 'b8':
0.19883040935672514, 'b9': 0.049707602339181284}
P(Bin|Flavor = 'Lime') for trained network with  10000  samples:
{'b10': 0.13, 'b4': 0.09088235294117647, 'b5': 0.18264705882352938, 'b6': 0.0,
'b7': 0.0, 'b1': 0.0, 'b2': 0.35205882352941176, 'b3': 0.0, 'b8':
0.20088235294117646, 'b9': 0.04352941176470588}
```

As we can see from the values that I obtained, as we increase the value of N, we get closer to the hand-calculated value of the probability of the bin given the flavor.

Q4.
a) A Naïve Bayesian classifier was made.
b) Vote is the cause. Gender, Age, Race, Political Ideological and Population Area all depend on Vote.
c)The network was trained as per the given instructions.
d) The enumerate_ask() method, visits all the nodes of the network. This gives more accuracy as compared to the variable elimination method (which has a directed network). The disadvantage of this property is that visiting all nodes is computationally expensive and hence enumerate_ask takes longer to run (and more space) than variable elimination. Similarly, we see that Monte Carlo estimate produces samples n times, and even computes probability of extremely rare events (like B and E happening simultaneously in the example shown in the class). This makes it very slow as compared to the weighted likelihood method. The advantage of this method is that since we assign a weight to each event (more so for the highly probable events), it does not search the entire network and excludes rare events, which makes it faster.

e)1. The classifier gives the following accuracy on poll_test.txt:

Number of matches for enumerate_ask was: 7518
Accuracy was: 0.7518 (75%)
Number of matches for monte_carlo was: 558
Accuracy was: 0.0558 (5.5%)
Number of matches for weighted likelihood was: 7248
Accuracy was: 0.7248 (72.48%)
Number of matches for variable elimination was: 7518
Accuracy was: 0.7518 (75%)
Number of matches for top down ask was: 5313
Accuracy was: 0.5313 (53%)

2. The classifier gives the following accuracy on poll_train.txt:

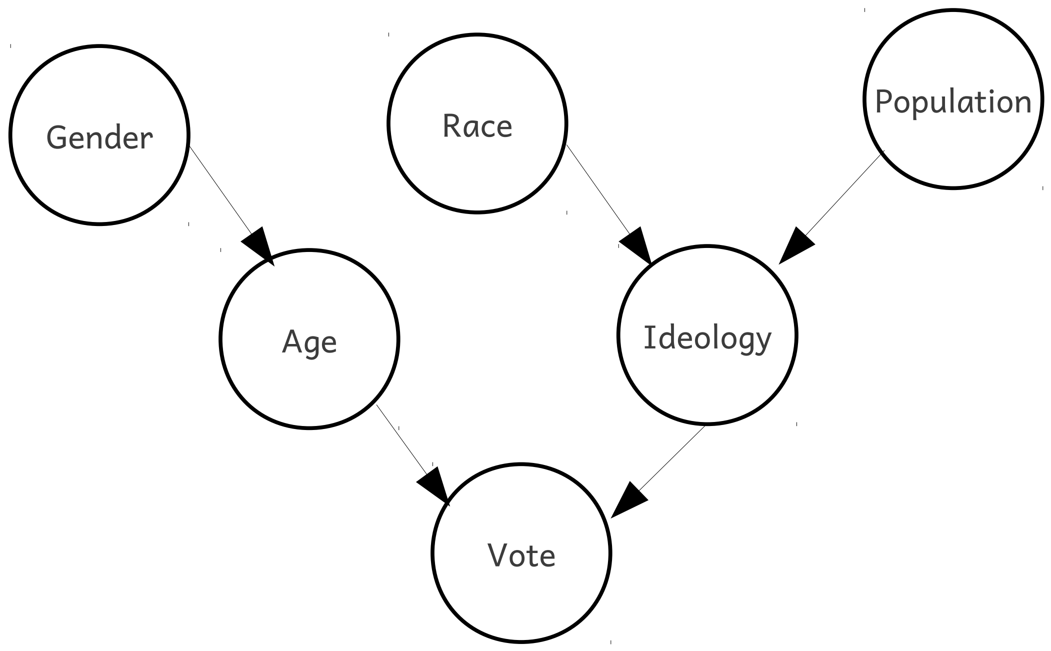
Number of matches for enumerate_ask was: 7515
Accuracy was: 0.7515 (75%)
Number of matches for monte_carlo was: 552
Accuracy was: 0.0552 (5.5%)
Number of matches for weighted likelihood was: 7243
Accuracy was: 0.7243 (72.43%)
Number of matches for variable elimination was: 7515
Accuracy was: 0.7515 (75.15%)
Number of matches for top down ask was: 5333
Accuracy was: 0.5333 (53.33%)

f) Yes it does seem to consider the vote node as the cause for the age, gender, population, race and the ideology. This does not change the dependencies as in a causal network (like this one), changing the order of the nodes does not change the dependencies. This can be explained by Bayes theorem. Consider the vote node (parent) and the age node (child). By Bayes theorem,

$P(V|A) = P(A|V)*P(V)/P(A)$.

Now in this expression, the parent-child relationship is not visible. So, even if we make A the parent of V, the expression would still hold and we would get the same probability values.

g) The following diagram shows the network that I have made.



h) The following independence relationships exist in the network that I have made:
Gender, Race and Population are independent of all other nodes.
Ideology is dependent on Race and Population.
Vote is independent of Gender given Age, and Race and Population given Ideology.

i) By tweaking and tuning the CPT's to improve accuracy (without any sort of training method) , the following is the accuracy that I get by using the different estimation methods.

```
Number of matches for enumerate_ask was:  4051
Accuracy was:  0.4051  (40.51%)
Number of matches for monte_carlo was:  149
Accuracy was:  0.0149  (1.49%)
Number of matches for weighted likelihood was:  3805
Accuracy was:  0.3805  (38.05%)
Number of matches for variable elimination was:  4051
Accuracy was:  0.4051  (40.51%)
```

Changing the CPT values in such a way as to minimize conflicts (when probability is like 0.33,0.33, and 0.34) made a huge difference in the accuracy when the network was not trained (upto 5% for enumerate_ask). After a certain point, the values for weighted likelihood estimate kept increasing, but those for enumerate_ask remained the same.

I encountered difficulties in the sense that, modifying the CPT values, does not always result in better accuracy. I realized that (a bit late I must say), that it is the structure of the network and the dependencies that determine the accuracy more than anything.

j) On training the code using samples from the poll_train file, and then calculating the accuracy, I found that the following were the accuracy values:

```
Number of matches for enumerate_ask was:  7285
Accuracy was:  0.7285
Number of matches for monte_carlo was:  490
Accuracy was:  0.049
Number of matches for weighted likelihood was:  6850
Accuracy was:  0.685
Number of matches for variable elimination was:  7285
Accuracy was:  0.7285
```

As compared to hand-tune, it is pretty obvious that the accuracy improves tremendously. Just to please my curiosity, I tried training the network with the poll_test file and tested it with the poll_train file. Here are the results:

```
Number of matches for enumerate_ask was:  7285
Accuracy was:  0.7285
Number of matches for monte_carlo was:  483
Accuracy was:  0.0483
Number of matches for weighted likelihood was:  6876
Accuracy was:  0.6876
Number of matches for variable elimination was:  7285
Accuracy was:  0.7285
```

This disappointed me!. There was no noticeable difference from the previous version.

k) Just adding edges to the network may or may not increase the accuracy of the network. Logically, knowing the value of another node (adding an edge from a node to another to establish a parent-child relationship) will not increase the uncertainty in the value of the child node (Shannon's Theorem).
The advantage then would be that the accuracy would increase. But if the network is large and complex, the time complexity and space complexity would increase. Hence this

would be a disadvantage.