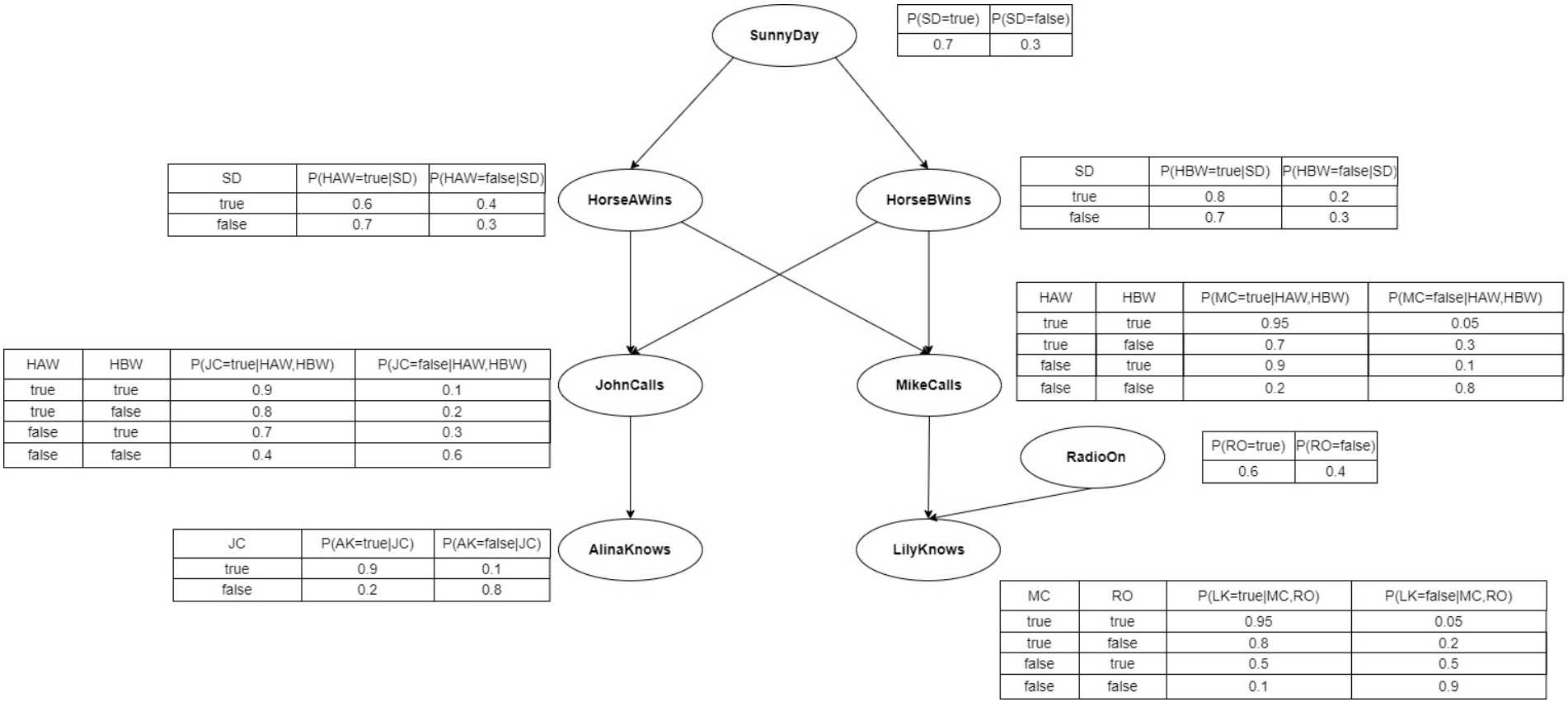
Project 3: Probability

In this Probability Project, we use Python to do the programming.

1 Bayesian Network Design

As shown in Figure1.1, there are 8 random boolean variables in our customized Bayesian Network.



P(SD=true)	P(SD=false)
0.7	0.3

SD	P(HAW=true|SD)	P(HAW=false|SD)
true	0.6	0.4
false	0.7	0.3

SD	P(HBW=true|SD)	P(HBW=false|SD)
true	0.8	0.2
false	0.7	0.3

HAW	HBW	P(JC=true|HAW,HBW)	P(JC=false|HAW,HBW)
true	true	0.9	0.1
true	false	0.8	0.2
false	true	0.7	0.3
false	false	0.4	0.6

HAW	HBW	P(MC=true|HAW,HBW)	P(MC=false|HAW,HBW)
true	true	0.95	0.05
true	false	0.7	0.3
false	true	0.9	0.1
false	false	0.2	0.8

P(RO=true)	P(RO=false)
0.6	0.4

JC	P(AK=true|JC)	P(AK=false|JC)
true	0.9	0.1
false	0.2	0.8

MC	RO	P(LK=true|MC,RO)	P(LK=false|MC,RO)
true	true	0.95	0.05
true	false	0.8	0.2
false	true	0.5	0.5
false	false	0.1	0.9

Figure1.1 A customized Bayesian Network and the conditional probability tables (CPTs)

Suppose there will be a horse competition tomorrow. The weather will be either sunny or not sunny, which is represented by the variable *SunnyDay*. There will be two horses in this competition. *HorseAWins* represents whether horse A will win and *HorseBWins* represents whether horse B will win. Two audiences John and Mike will presumably make a phone call about the result of this competition to their wife Alina and Lily respectively, which is shown by the variables *JohnCalls* and *MikeCalls*. There is a chance that Alina will pick up the phone and know the result. However, we cannot rule out the probability that Alina will miss the phone call. This situation also applies to Lily. As for Lily, she usually listens to the radio and may know the result of this horse competition from radio news. The variables *AlinaKnows*, *LilyKnows*, and *RadioOn* describe these circumstances.

2 Bayesian Network Representation

2.1 Bayes Nets

In our project, we have a *Node* class to initialize each node of variable in the Bayesian Network and a *BayesNet* class to initialize a Bayesian Network by combining all the node objects.

Each node object contains several properties: the variable name, parent variables, child nodes, and conditional probability table (CPT) of this node. The variable name of this node is a *string* data type. The parent variables of this node are a *list* of *string*. Child nodes of this node are a *list* of node objects. CPT is a *dictionary* data type where value is the probability of one event and key is the values of that event (either True or False in our project).

2.2 Queries and Evidence

To test our custom Bayesian Network, we create three queries including one causal query, one diagnostic query, and one for the sanity check. Each query is a *dictionary* data type and it has two key-value pairs: one for the nonevidence variable, another for the observed variables. The observed variables are also a *dictionary* data type including their names and their observed values. The three queries are nested in one dictionary as shown in Figure2.1.

```
q_horse = {
    'causal': {'X': 'AlianKnows', 'e': {'SunnyDay': T}},
    'diagnostic': {'X': 'HorseBWins', 'e': {'LilyKnows': T}},
    'sanity': {'X': 'LilyKnows', 'e': {'HorseBWins': F, 'HorseAWins': F}}
}
```

Figure2.1 Our Custom Query

3 Inference and Evaluation

3.1 Inference

Our exact inference, rejection sampling, and Gibbs sampling algorithms are based on pseudo-code in Figure13.11, 13.17, and 13.20 respectively in the textbook AIMA.

3.2 Evaluation

3.2.1 Convergence

The results of rejection sampling and Gibbs sampling depend on the sample size. As the number of samples increases, it converges on a value. In our project, we use a **threshold** to test stability. Specifically, we first append the result of each sampling into a list (*rec*) as the number of samples increases. Then we take out the **last five** values in

the list to do the convergence test. We compare each value with the **mean** of these five values. If the gap between two numbers is larger than the threshold (the threshold is **0.01** in our custom example), then it doesn't pass the convergence test and the sampling output is not converging. If all these values in the list pass the convergence test, then the sampling output is converging. The algorithm of this convergence test is shown in Figure3.1.

```
rec_mean = sum(rec)/(len(rec))
for num in rec:
    if abs(num - rec_mean) > threshold:
        return False
return True
```

Figure3.1 Convergence/Stability test

Figure3.2, 3.3, and 3.4 show convergence graphs for rejection sampling and Gibbs sampling using our custom query. The green horizontal line shows the value from exact inference.

We can see that in Figure3.2 the curve of rejection sampling converges on over 2000 samples while the curve of Gibbs sampling stabilizes when the sample size is 6610. In Figure3.3, the curve of rejection sampling converges on just over 1000 samples while the curve of Gibbs sampling stabilizes on the sample size of 2910. However, in Figure3.4, the curve of rejection sampling converges on around 6000 samples while the curve of Gibbs sampling stabilizes when the sample size is about 2300.

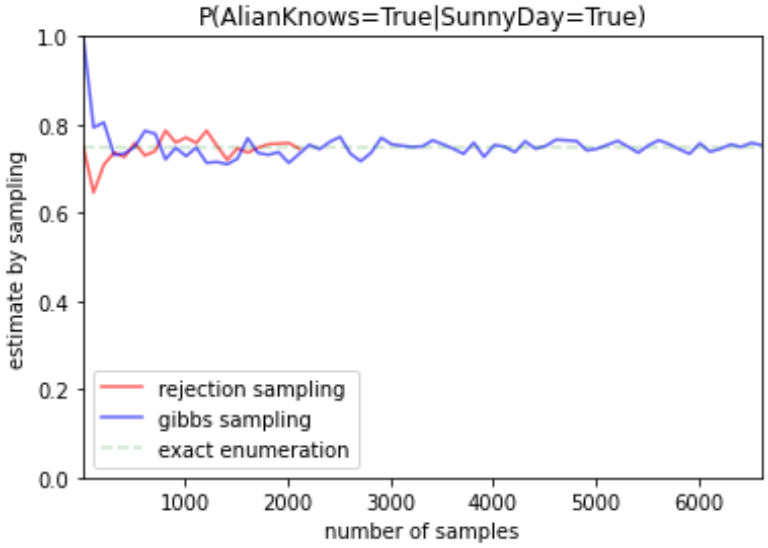


Figure3.2 Convergence graph using a causal query

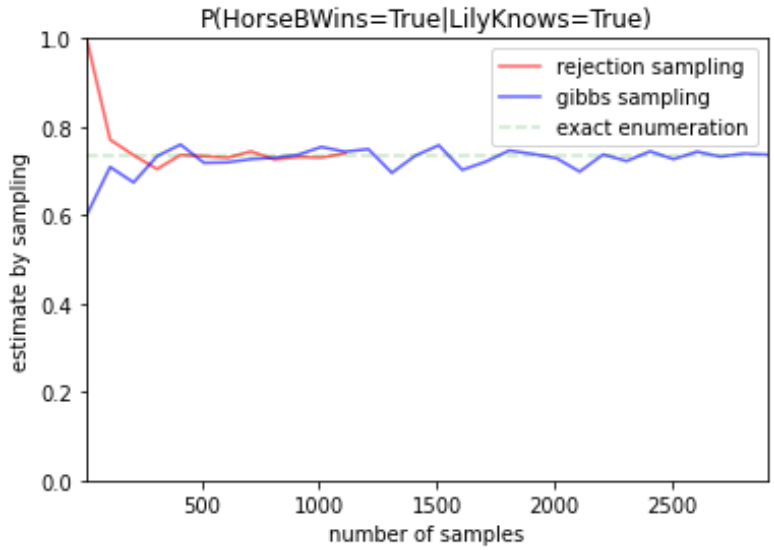


Figure3.3 Convergence graph using a diagnostic query

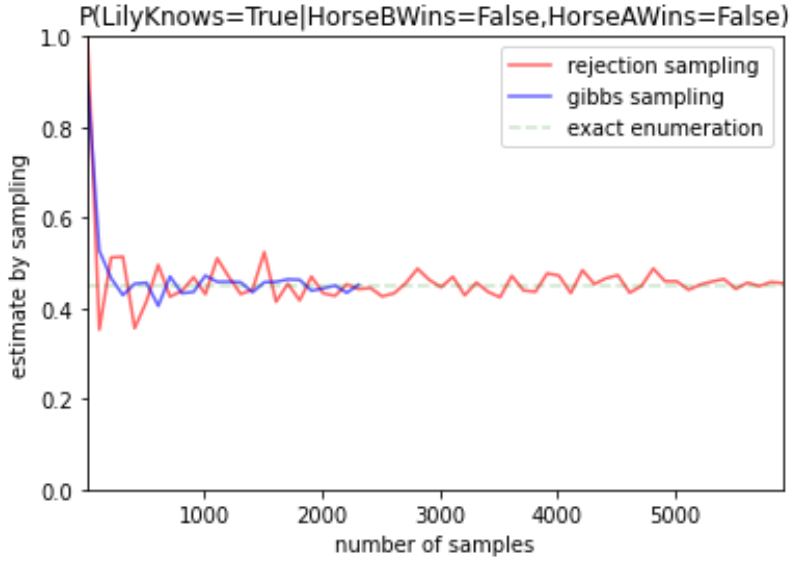


Figure3.4 Convergence graph using a sanity check

The sample size needed for rejection sampling to become stable largely depends on the probability of its evidence event. If this probability is small enough, then rejection sampling usually needs more samples to achieve stability.
However, the sample size for rejection sampling to achieve stability may be hard to predict since it is wild and the value varies from 1000 to 6000 in our three queries. This conjecture still needs to be confirmed.
In terms of **time efficiency**, rejection sampling converges faster than Gibbs sampling in all our queries even though fewer samples are needed for Gibbs sampling in our sanity check (more details in evaluation.ipynb).
[answers to questions 6(a),6(b),6(c)]

3.2.2 Accuracy

We use relative error to estimate the accuracy of these two sampling methods. The relative error can be calculated as: $|sampledValue - exactValue|/exactValue$. We also set three **thresholds** (**0.01, 0.005, 0.001**) to determine if the relative error is approaching

zero. If the relative error of each of **three consecutive sample values** approaches zero, then we say that this sampling is now accurate on this sample size. The results of estimating accuracy using our custom query are shown in Figure3.5, 3.6, 3.7 (other results are in our Jupyter Notebook).

In Figure3.5, Gibbs sampling reaches zero relative error on under 2000 samples while the corresponding sample size for rejection sampling is about 10000. In Figure3.6, Gibbs sampling achieves zero relative error on around 3000 samples while the corresponding sample size for rejection sampling is about 5000. In Figure3.7, Gibbs sampling achieves zero relative error on under 10000 samples while the corresponding sample size for rejection sampling is over 20000.

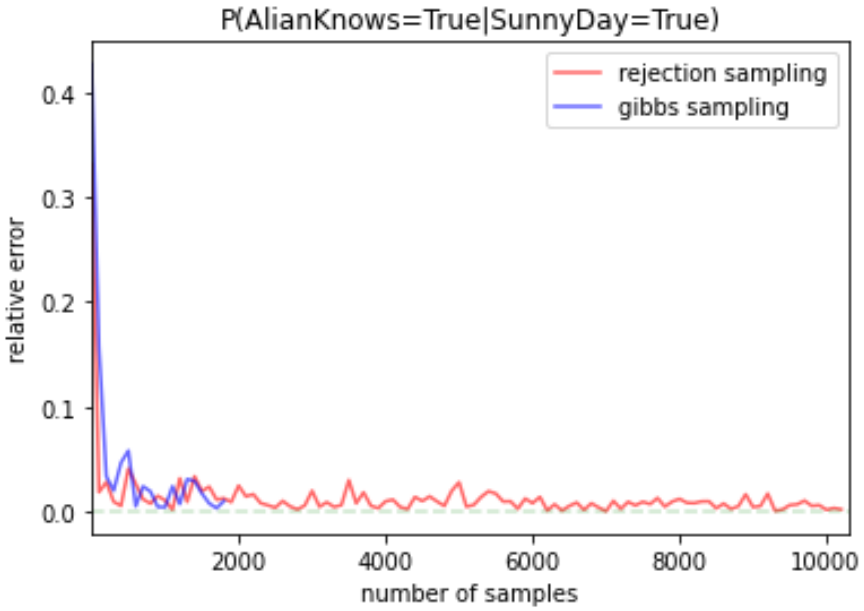


Figure3.5 Accuracy graph using a casual query (threshold = 0.005)

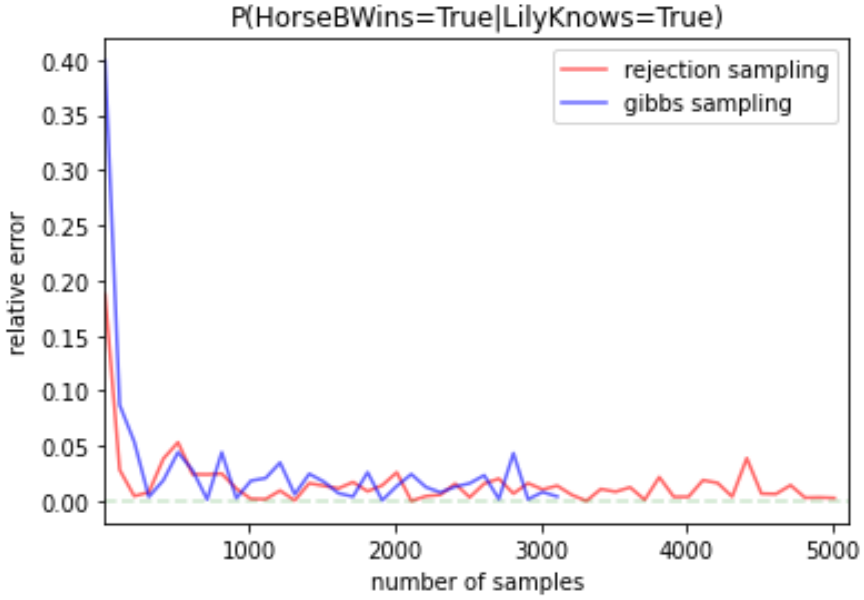


Figure3.6 Accuracy graph using a diagnostic query (threshold = 0.005)

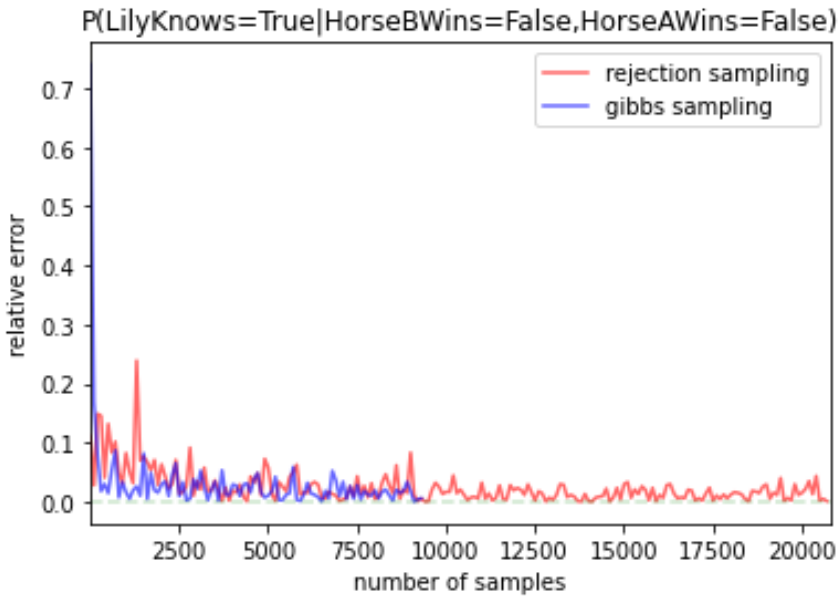


Figure3.7 Accuracy graph using a sanity check (threshold = 0.005)

We can see from these pictures that the accuracy is getting better as the number of samples increase. As the threshold becomes smaller, it seems that Gibbs sampling is more likely to achieve zero relative error on smaller sample size.
[answers to questions 6(b),6(c)]

4 Discussion

From Section3, we can infer that convergence, accuracy, and time-efficiency of rejection sampling and Gibbs sampling varies from query to query. Based on the results of our custom Bayesian Network and AIMA-ALARM example, we believe that it is likely that these values depend more on the query than the structure of the network. For example, in Figure4.1, when the probability of the evidence event (Alarm = True) is very small (10^{-3}), rejection sampling often takes a much longer time to converge regardless of the network structure.

As Gibbs sampling algorithm is more complex than rejection sampling algorithm, it usually takes more time to complete a task. But if you want a more accurate result that is close to the enumeration value and the sample size is limited, Gibbs sampling may be a better choice.

If the probability of an evidence event is very low and the sample size is limited, rejection sampling usually requires more samples, and it may be better to use Gibbs sampling. If you can try a large sample size, rejection sampling may be a better option since it is faster, but the proper sample size is hard to predict.
[answers to questions 6(d),6(e)]

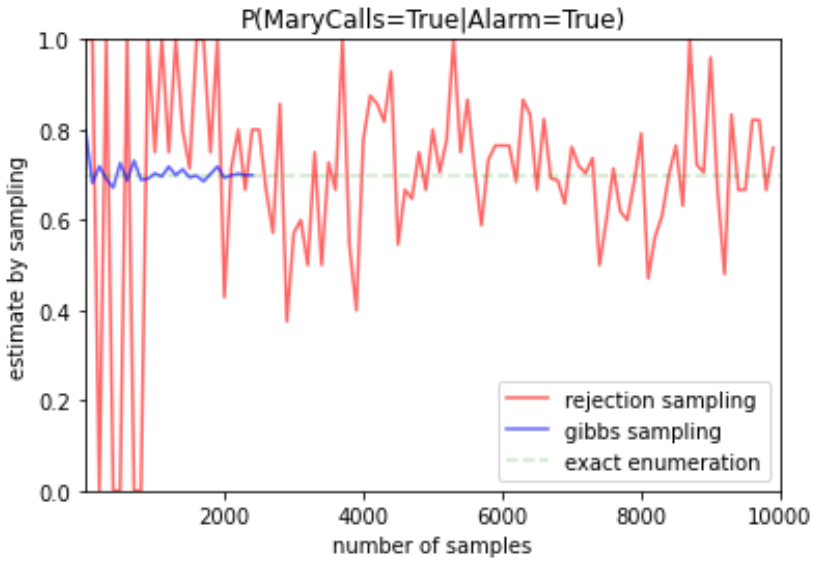


Figure4.1 Convergence graph using a sanity check (AIMA-ALARM)

5 Collaboration

Contributions		Working time	
Coding	BayesianNetwork.py	2h	1h
	ExactInference.py	2h	1h
	ApproximateInference.py	1h	4h
	evaluation.py	1h	1.5h
Writeup	Bayesian Network design	1.5h	
	Queries and evidence	40min	10min
	Inference and evaluation	2.5h	0.5h
	Discussion	20min	5min
	Collaboration	5min	2min