Comparison of execution time and precision between approximate and exact inference in Bayesian Networks

FAIKR module 3 project report

Edoardo Procino

Academic Year 2021-2022

Contents

1 Introduction **2**

2 The dataset **2**

3 The network **3**
3.1 Structure learning . . . 3
3.1.1 Constraint-based approach . . . 3
3.1.2 Score-based approach . . . 3
3.2 Hand-made graph . . . 3

4 Parameter Estimation **5**

5 Exact Inference **5**
5.1 Exact inference with variable elimination . . . 5
5.2 Querying the Network . . . 5

6 Approximate Inference **6**

7 Comparison **6**
7.1 MyInference class . . . 6
7.2 Exact inference vs Maximum likelihood . . . 8
7.3 Exact inference vs Rejection Sampling . . . 9
7.4 Likelihood Weighting vs Rejection Sampling . . . 11

8 Conclusion **12**
8.1 Outliers . . . 12
8.2 Concluding analysis . . . 12

1 Introduction

A Bayesian network is a *directed acyclic graph* that represents a set of variables and their conditional dependencies.

In this report I will explain my project that I did for the module 3 of the subject *Fundamental of Artificial Intelligence and Knowledge Representation* during my master in AI.

My idea was to take a dataset, build a Bayesian network from it and make a comparison between some types of inference.

The dataset that I chose is about asthma for the reason that I like applying my work to something helpful for the people like medicine. I know that this project is very small and I used the dataset only to do theoretical stuff, but this topic is anyway interesting for me.

In particular, I compared the execution time between the *exact inference with Variable Elimination* and two different types of approximate inference methods: *Likelihood Weighting* and *Rejection Sampling*, making a distinction between the time needed for the sampling and the time needed to calculate the probabilities. Finally, I computed the percentage error between the exact inference end the two methods for the approximate.

I repeated all this procedures three times using three different sizes for the sampling in order to understand in which situation is worthier using the exact inference rather than the approximate inference and vice versa.

2 The dataset

The dataset that I used is very simple, it has only nine columns and each attribute has a domain of dimension two or three. I did the choice to use a little dataset like this one due to execution time because, running my code, even with a such small dataset (and so with a small network), required me more than 5 hours (I did a lot of queries in order to have a 'right' number to make an analysis).

As written before, the dataset has nine attributes, that are:

- $sex \in ['female', 'male']$
- $age \in ['young', 'adult', 'old']$
- $urbanization \in ['low', 'medium', 'high']$
- $education \in ['low', 'high']$
- $geo\_area \in ['north', 'centre', 'south/islands']$
- $allergy \in ['yes', 'no']$
- $smoke \in ['yes', 'no']$
- $sedentary \in ['yes', 'no']$
- $asthma \in ['yes', 'no']$

3 The network

In this section I will explain how I built the network and how I reached the final shape.

3.1 Structure learning

As first approach, since I saw in the documentation of pgmpy [6] (the library that I used) a section about *structure learning*, I tried this way to build the network.

Structure learning is a task that consists to learn the structure of the graph from data and there exist two approaches.

3.1.1 Constraint-based approach

The first approach is the so called *constraint-based* that does some independence test to identify a set of edge constraints for the graph and then find the best graph that satisfies the constraints. Since this technique works well with some prior structure knowledge and requires a lots of data samples, but I am not an asthma expert and the dataset is small (less than 3000 entries), I did not test this method.

3.1.2 Score-based approach

The second approach is the *score-based approach* that consists in defining a metric to evaluate how well the network fits the data. Basically, we compute a score for each graph and we take the greatest, so this is a search maximization problem.

I tested the *Hill climb search* with all the possible score functions available on pgmpy, but all the results did not convince me. All the computed graph had the problem that they did not respect an ordering that lead to a causal-effect network, the networks generated could be right but it was impossible to assert (at least only looking at the structure) if a graph was correct. The main problems were the following ones:

- Some graphs had an edge from **asthma** to **allergy**. This is wrong from the causal-effect point of view because it is have an allergy that could lead to develop asthma [1];
- Some graphs had a lot of edges in the 'wrong' order like from **education** to **age**, intuitively the edge should be in the opposite direction;
- Some graphs had nonsense edges like from **education** to **asthma**, it is quite obvious that your level of education do not influence if you have or not asthma. In addiction, in this case also the edge from asthma to education has nonsense, so for a human is impossible to understand if this edge is correct.

So, since the small number of nodes, for all the reasons that I explained above and to have a more readable graph, I decided to build my network from scratch.

3.2 Hand-made graph

To build my network I used good sense and the Internet to search information about asthma.

Now I will explain the edges of the graph:

- **age** has an edge to **sedentary**, **smoke**, **allergy** and **education** because the older you are, the more likely is to be sedentary (an old man/woman normally do less activity than a younger one), to smoke, to have an allergy and to have an higher instruction level;
- **education** influences **sedentary** because, probably, if you have studied, you know from high school that be sedentary is not worth for you health;
- **allergy** influences **asthma** because according to this [1] article, some allergies con cause asthma symptoms;
- **smoke** influences **asthma** because when a person inhales tobacco smoke, irritating substances settle in the moist lining of the airways and can set off asthma episodes [9];
- **sedentary** influences **asthma** according to the study *Low Fitness and Increased Sedentary Time Are Associated With Worse Asthma–The National Youth Fitness Survey* [8];
- **urbanization** influences: **education** because in a urbanized city people have more opportunity to study, **sedentary** because in an urbanized context people have more opportunity to use public transports and **asthma** because according to EPA (United States Environmental Protection Agency) air pollution, typical of urbanized cities, can trigger asthma attacks [11];
- **geo_area** influences **urbanization** because, on average, the north of the world is more urbanized than the south, but also influences **asthma** and **allergy** according to the paper *Geographic factors can cause allergies, asthma: People living close to the equator are at higher risk, study finds* [5];
- **sex** influences **asthma** according to the paper *Sex Differences in Asthma: A Key Role of Androgen-Signaling in Group 2 Innate Lymphoid Cells* [3].

The resulting Bayesian network is the one in the Figure 1.

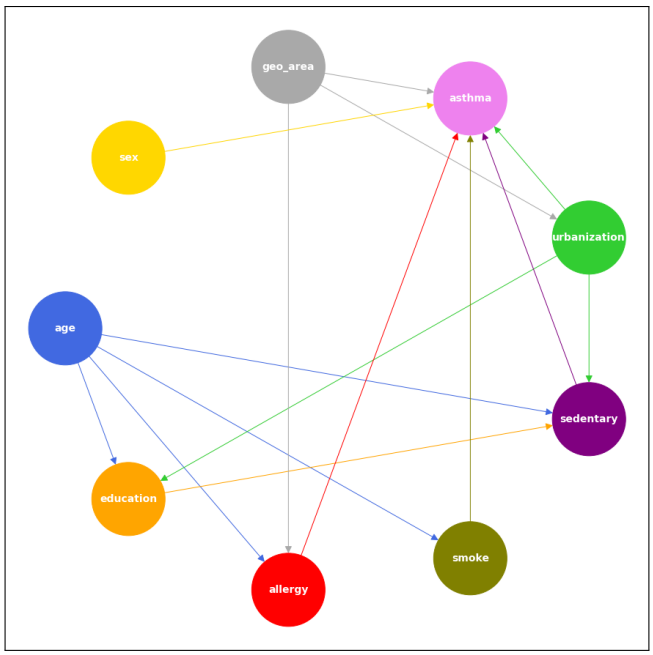

Figure 1: *My Bayesian network*

4 Parameter Estimation

At this point I had the network, and, since I wanted it to be a model for the asthma data, I needed to do the so called *Parameter Estimation* to enable the network to learn the *Conditional Probability Tables (CPTs)*.

There are a lot of algorithms to achieve this task, but the two most popular are the *Maximum Likelihood Estimator (MLE)* and the *Bayesian Estimator* (pgmpy implements both of them). I chose MLE because I had not prior knowledge on the problem so, even if the dataset had not a lot of entries, this was the best choice [4].

5 Exact Inference

Starting from this section, I will explain how I did the inference on my network and then I will compare all the tested method on execution time and precision.

5.1 Exact inference with variable elimination

The first inference that I tested was the exact inference with the algorithm *Variable Elimination*.

Two of the main approaches for the exact inference are: *Inference by enumeration* and *Variable Elimination*. I chose the latter because according to the book *Artificial Intelligence A modern Approach* [2] the Variable Elimination algorithm is an evolution of the Enumeration algorithm since the former uses dynamic programming to avoid recomputing.

Regarding the elimination order (which could determines the efficiency of the algorithm), the above mentioned book [2] says that use heuristic is a good way to determine this order since determining it is an intractable problem.

Looking at the documentation of pgmpy regarding the variable elimination [7], and also looking at the source code of the pgmpy VariableElimination class [10], I discovered that if you call the function *query* — that returns the probabilities of the specified queries give, or not, some evidences — without specifying the "elimination_order" parameter, its default value is the string "MinFill" that says to the class to use the MinFill heuristic for computing the elimination order. For this reason I did not spend time to try different elimination orders.

5.2 Querying the Network

Before going on with the approximate inference and the comparisons, I would like to write a little bit about some queries that I did both for curiosity and to see if the results make sense. I did the following queries with exact inference because is more accurate that approximate inference and with a small network like this one the execution time is not a problem.

To explore a little bit my network I did the following queries: P(asthma | smoke, sedentary) and P(asthma | age, allergy).

I chose these queries for two reasons:

- I am interested to see how two things that you can control (smoking and be sedentary) and two things that you can not control (your age and if you have an allergy) influence if you have or not asthma;

- Smoking, have an allergy and be sedentary directly influence have asthma, so it is easier understand if the results make sense.

The results was those expected:

- P(asthma | smoke, sedentary): we have the highest probability to not have asthma with smoke='yes' and sedentary='no' and it is very close to the one with smoke='no' and sedentary='no'. On the other hand, the highest probability to have asthma is with both the evidences equal to 'yes';
- P(asthma | age, allergy): The results show that having or not an allergy influence a lot having or not asthma while the age influences it very little.

6 Approximate Inference

For the approximate inference part I chose two algorithms — the *Likelihood Weighting* and the *Rejection Sampling* — in order to understand which is the best in which situation.

According to *Artificial Intelligence A modern Approach* [2], in general, approximate inference algorithms are faster than the exact inference ones, but, as their name suggests, they perform an approximation, so they produce a probability distribution slightly different than the distribution produced by exact inference algorithms.

In the following section I will compare the exact inference with variable elimination and the two, above mentioned, algorithms for the approximate inference.

7 Comparison

7.1 MyInference class

To make the comparison easier to do I built a python class called **MyInference**, the main methods are:

- The constructor, that takes in input a model (a fitted network) and initializes two variables to perform the exact and the approximate inference;
- **my_exact_inference**: this methods takes in input a query and a list of evidences, performs the exact inference using the Variable Elimination algorithm and returns the probability values and the execution time;
- **my_rejection_sample** and **my_likelihood_weighted**: they are the equivalent of the my_exact_inference method but to perform approximate inference. The only two differences are that they also take in input the size for the sampling and they return also the time needed to generate the samples;
- **get_evidences**: it takes in input a list of queries, a data object (a pandas dataframe) and return a list of lists of evidences that contains all the possible permutations (i.e: if I pass ['smoke','sedentary'] the output will be something like [[smoke:'yes'],[smoke:'no'], [sedentary:'yes'],[sedentary:'no'],[smoke:'no',sedentary:'no'],[smoke:'no',sedentary:'yes'], [smoke:'yes',sedentary:'no'],[smoke:'yes',sedentary:'yes']]). This method is useful to generates a lot of evidences to perform a lot of queries;

- **comparison**: it takes in input a list of queries, a list of evidences and a size parameter for the approximate inference and perform all the three kinds of inference for all the queries using all the evidences. For each approximate inference query, the method also computes the percentage error in relation to the exact inference value. This method returns 7 lists, each of them contains a value for each performed query. The lists are:
  - time_exact_inference
  - time_approx_inference_likelihood
  - time_approx_inference_rejection
  - time_approx_generation_likelihood
  - time_approx_generation_rejection
  - error_rejection
  - error_likelihood

At this point I called the method **get_evidences** with the list of queries ["age","urbanization","smoke", "sedentary","asthma"] (if you call it with all the possible values then it requires too much time to perform the following part) to have a list of 431 possible evidences, then I called the method **comparison** three times (all the times with all the possible queries and with the 431 evidences) changing the size parameter (I used 10^2,10^3,10^4).

In the following sections I will compare the results using some plots.

7.2 Exact inference vs Maximum likelihood

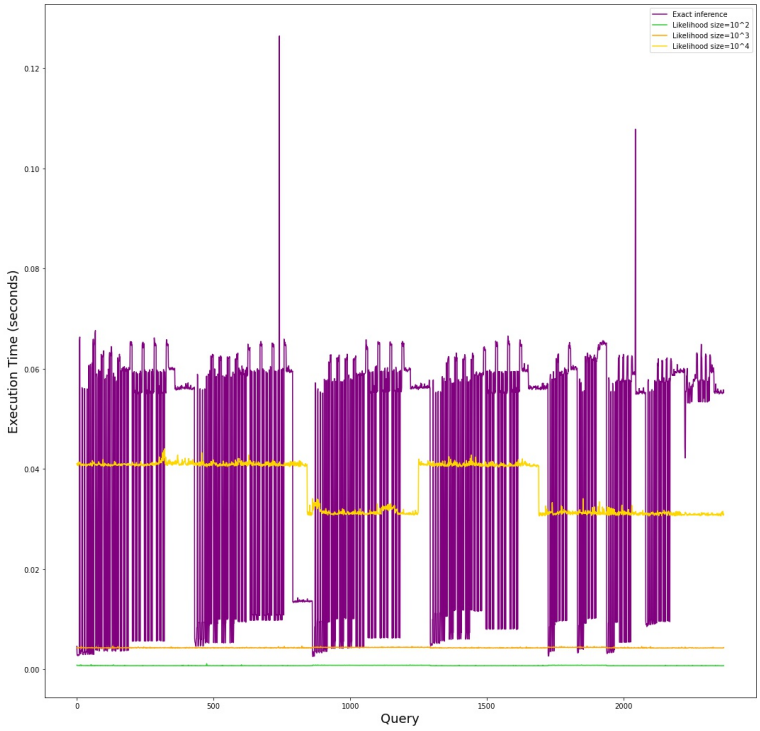

Figure 2: *Comparison of execution time between exact inference and approximate inference with Likelihood Weighting*

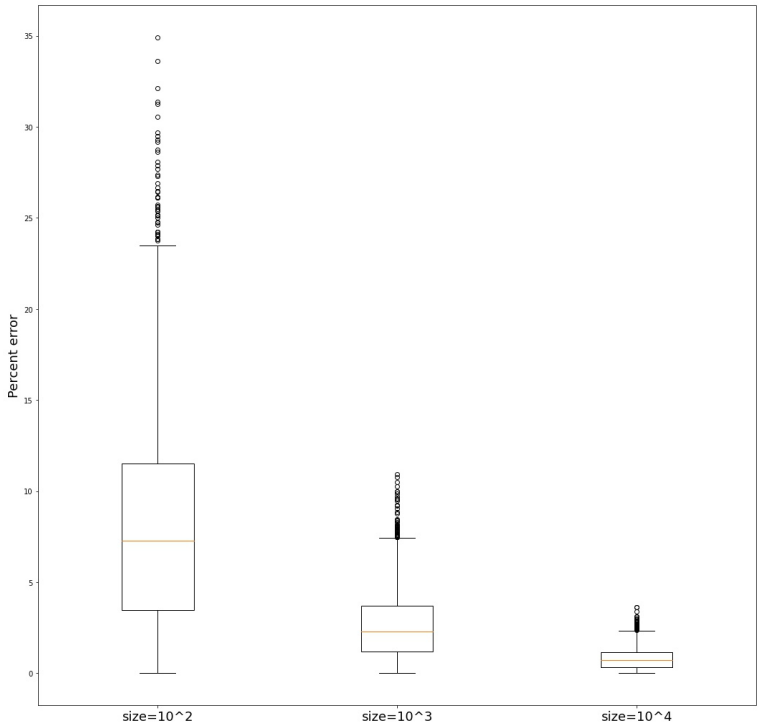

Figure 3: *Comparison between errors Likelihood Weighting*

0.040744

Table 1: *Average execution time (seconds) of Variable Elimination algorithm*

Size 10^2	Size 10^3	Size 10^4
0.000735	0.004305	0.0365008

Table 2: *Average execution time (seconds) of Likelihood Weighting algorithm*

Size 10^2	Size 10^3	Size 10^4
9.08	2.90	0.90

Table 3: *Average percentage error of Likelihood Weighting algorithm*

Looking at the tables and the graphs above, it is interesting that using a size equal to 10^4 the average execution time is practically the same, so, at least in this case, it seems better to use the exact inference because with the approximate one you have to pay the generation time. Anyway, if someone wants to use the approximate inference, using a size of 10^3 seems very good since the execution time is ten times lower and the error is only 2.9% as the table 3 shows.

7.3 Exact inference vs Rejection Sampling

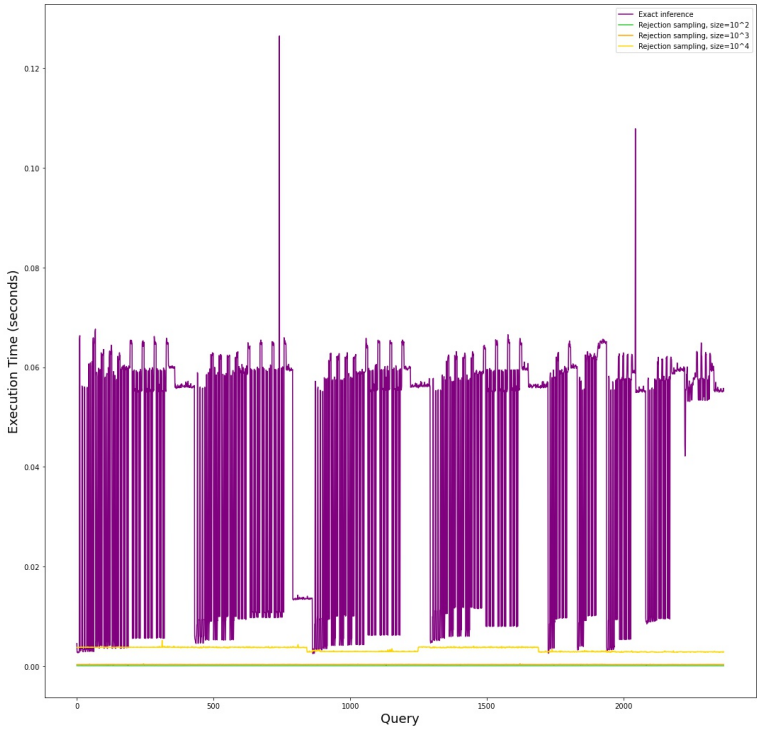

Figure 4: *Comparison of execution time between exact inference and approximate inference with Rejection Sampling*

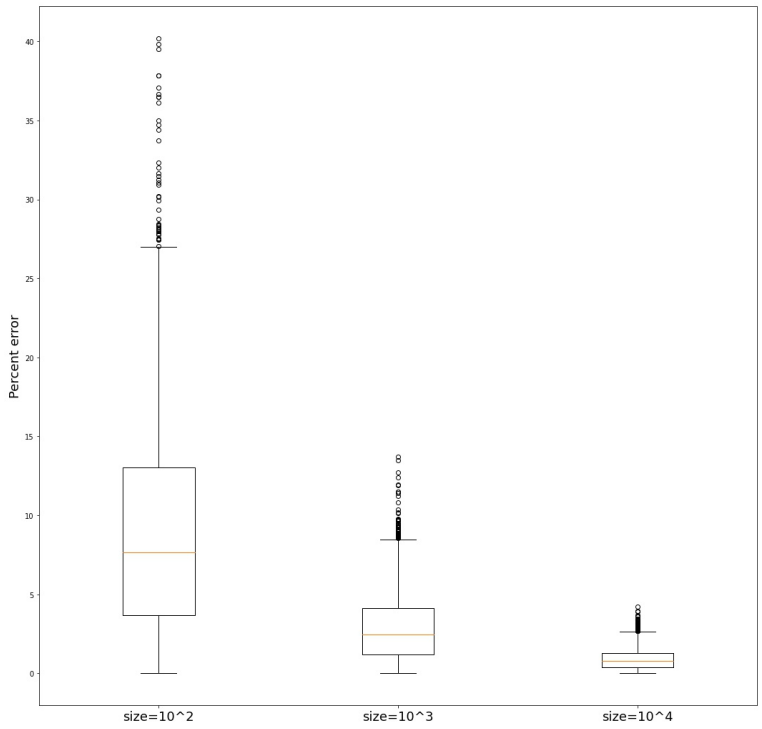

Figure 5: *Comparison between errors Rejection Sampling*

Size 10^2	Size 10^3	Size 10^4
0.000109	0.000367	0.003409

Table 4: *Average execution time (seconds) of Rejection Sampling algorithm*

Size 10^2	Size 10^3	Size 10^4
8.25	2.62	0.81

Table 5: *Average percentage error of Rejection Sampling algorithm*

With Rejection Sampling I obtained results even better than the ones with Likelihood Weighting, the errors (with the same size) are very close and the execution is even faster. The reason why the Likelihood Weighting is considered better than the Rejection Sampling is that it can generates the samples in a faster way than Rejection Sampling and this happens because Likelihood Weighting generates only events that are consistent with the evidence. In the following section I will compare the generation times of the two methods.

7.4 Likelihood Weighting vs Rejection Sampling

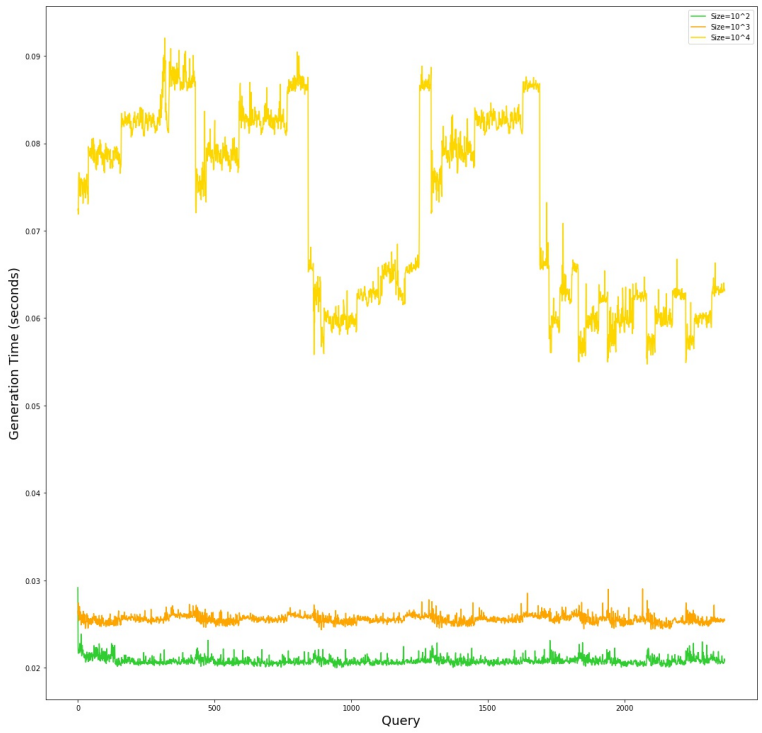

Figure 6: *Time to generates the samples Likelihood Weighting*

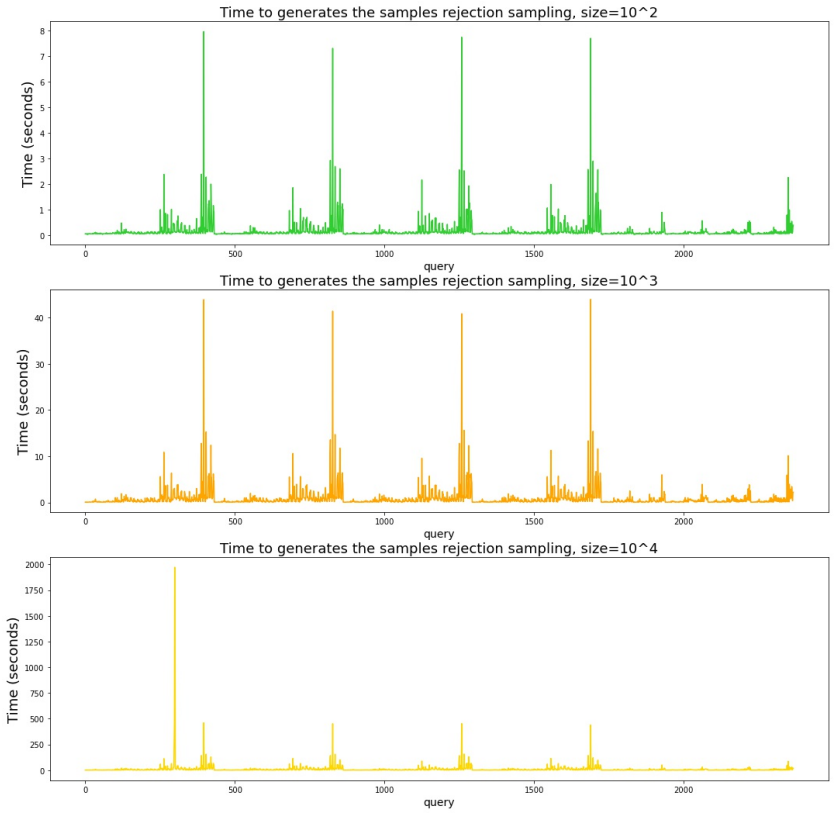

Figure 7: *Time to generates the samples Rejection Sampling*

Size 10^2	Size 10^3	Size 10^4
0.020793	0.025562	0.072575

Table 6: *Average generation time (seconds) of Likelihood Weighting algorithm*

Size 10^2	Size 10^3	Size 10^4
0.159203	0.829491	10.032359

Table 7: *Average generation time (seconds) of Rejection Sampling algorithm*

It can be easily seen that there is a very huge difference between the generation times of the two methods. In both algorithms the sampling time grows a lot increasing the size parameter, but, while using the Likelihood Weighting the graph reaches a maximum time slightly greater than 0.09 seconds, with Rejection Sampling the graph reaches some peaks very close to 500 and a maximum near 2000 seconds to generate the samples for a single query making the Rejection Sampling an unfeasible solution, especially for bigger networks.

8 Conclusion

8.1 Outliers

In the figures 3 and 5 there are a lot of outlier data. Looking at them, only four or five per chart are related to the class (i.e.: the outlier error was generated by a query of the type P(asthma | E) where E is a set of evidences).

These outliers are probably generated by the small number of entries in the dataset. Despite this, I think that at the end of the day these outliers do not compromise the results of my work because everything works like expected:

- Increasing the size the error goes down while the precision goes up;
- Likelihood Weighting seems to be better than Rejection Sampling;
- The approximate inference is faster than the exact.

8.2 Concluding analysis

My results reflect what the theory says. The exact inference is slower than the approximate inference, but it is a feasible solution when you have a network with few nodes or when you have to do only a very restricted number of queries since you do not have to pay the generation time for the sampling. As tables 1 and 2 show the fact that with a size of 10^4 the Likelihood Weighting algorithm is as fast as the variable Elimination algorithm is probably for the small size of my network, even though some tests with a bigger network could be very interesting.

Regard the approximate inference methods the best one (between the two examined) is the Likelihood Weighting because even if the execution time of the Rejection Sampling takes less time, the generation time of the latter method makes it not usable in real world situation. To my mind, looking at the results that I had, Rejection Sampling is not worth either if you sample only one time and then you perform a lot of queries, the generation time is also in this case too high.

References

[1] *Allergy and Asthma correlation.* URL: https://www.mayoclinic.org/diseases-conditions/asthma/in-depth/allergies-and-asthma/art-20047458.

[2] *Artificial Intelligence A Modern Approach, Third Edition.* URL: https://zoo.cs.yale.edu/classes/cs470/materials/aima2010.pdf.

[3] *Biological sex and Asthma correlation.* URL: https://www.frontiersin.org/articles/10.3389/fimmu.2017.01069/full.

[4] *Comparison between MLE and Bayesian estimator.* URL: https://towardsdatascience.com/maximum-likelihood-vs-bayesian-estimation-dd2eb4dfda8a.

[5] *Geographic area and Asthma/Allergy correlation.* URL: https://www.sciencedaily.com/releases/2013/02/130204095926.htm.

[6] *pgmpy documentation.* URL: https://pgmpy.org/.

[7] *pgmpy documentation of Variable Elimination algorithm.* URL: https://pgmpy.org/exact_infer/ve.html.

[8] *Sedentary and Asthma correlation.* URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7187732.

[9] *Smoke and Asthma correlation.* URL: https://my.clevelandclinic.org/health/articles/4584-smoking--asthma.

[10] *Source code of pgmpy's Variable Elimination class.* URL: https://pgmpy.org/_modules/pgmpy/inference/ExactInference.html#VariableElimination.

[11] *Urbanization and Asthma correlation.* URL: https://www.epa.gov/sciencematters/links-between-air-pollution-and-childhood-asthma.