# Paradoxes in Probabilistic Programming

KOUSHIK CHOWDHURY, Saarland University, Germany

This document discusses Jules Jacob's [1] work on probabilistic programming. Probabilistic Programming gives programmers a way to create probabilistic models that can be used to draw probabilistic conclusions. This article provides a concise overview of probabilistic programming, paradoxes in probability theory, and the identification of three paradoxes by Jules Jacob in probabilistic programming, as well as his work on events of measure zero and infinitesimal interval which help to eliminate these paradoxes.

Additional Key Words and Phrases: paradox, observe, probabilistic

## 1 INTRODUCTION

### 1.1 Probabilistic Programming

A probabilistic model is created using probabilistic programming in order to make probabilistic inferences that help to make decisions in uncertain circumstances. We can assist in decision-making when there is uncertainty by using probabilistic reasoning. Assuming we want to know the likelihood of an event, probabilistic reasoning determines the likelihood using the relevant evidence. Let us use the world of football as an example [2]. According to the data, 6.01 percent of corner kicks in the English premier league result in goals. Imagine that our striker is over six feet tall and has a specialty in heading. In addition to the fact that it is raining, the opposition goalkeeper is not as tall as a typical goalkeeper. What is the probability that the striker will score a header if the ball is thrown directly in his direction? To determine the solution, we employ a probabilistic reasoning system. In the beginning, we encode our understanding of corner kicks in a football game and take into account pertinent variables. Second, we offer evidence, including the striker's height, the goalkeeper's height, and the weather conditions. Thirdly, we specify that we are interested in learning the probability that the striker will score a goal. The system then employs an inference algorithm to forecast the outcomes. Now we can decide whether to purchase a new tall goalkeeper or not based on the probability of chances for a goal that benefits the team. A probabilistic model has five components: [2],

- **General Knowledge:** General information about your domain that is not specific to a given situation.
- **Probabilistic model:** An expression of general knowledge.
- **Evidence:** Information about the situation includes prior knowledge.
- **Query:** Information that we need to know when making a decision.
- **Inference:** The algorithm that the model uses to provide the outcome.

To run inference algorithms like Metropolis-Hastings, Sequenzielle Monte-Carlo Method, and Hamiltonian Monte Carlo, Jules Jacobs noted that various probabilistic languages like Stan, Church, and Anglican to create a probabilistic model [1]. The probabilistic programming language is a domain-specific language for statistical inference. It has three constructs, including rand, observe, and run. The word "rand" stands for random sampling. "Observe" is for conditioning, and "Run" is for running the simulation. In probabilistic programming, there are two points of view. First, probabilistic programs simplify the process of constructing a likelihood function. The likelihood function

Author's address: Koushik Chowdhury, MSc. Student, Saarland University, Computer Science Department, Saarbrucken, Saarland, 66123 , Germany, s8kochow@stud.uni-saarland.de.

is concerned with the joint probability where random variables are defined in the same space of probabilities. Second, A notation for structured probabilistic models is probabilistic programs.

### 1.2 Introduction to Paradox

Paradoxes are contradictory statements that at first appear to be true. Probability theory frequently contains paradoxes. Let us examine the two most prevalent paradoxes in the field of probability. First, The birthday paradox. Consider the case where you have been invited to a friend's birthday party where at least two of your friends share the same birthday. The paradoxical question asks how many people must attend the party to ensure that there is at least a half percent chance that two of your friends share the same birthday. The objective is to determine the possibility of two of your friends sharing the same birthday, which is P(A). The probability that there are individuals in the room whose birthdates are not the same is P'(A). Therefore, P(A) = 1- P'(A) because A and A' are mutually exclusive in this situation. The threshold size to achieve 50% in the real world is 23 [3]. Therefore, the probability of two of your friends sharing the same birthday at the party if 30 people are invited is over 70%, which is absurd. The other most common paradox is known as Bertrand's box paradox. Three boxes are included in the experiment. One box has 2 silver coins, one box has 2 gold coins, and another has a gold coin and a silver coin. The specific coins that are in each box are unknown to us. Now that we have opened one box and found a gold coin inside, the paradoxical question is: What is the probability that the second coin will also be gold? We typically consider the coin flip experiment result, which is $\frac{1}{2}$, when this type of circumstance arises. In this instance, the actual probability that the second coin is gold is also 0.5 but the actual probability is $\frac{2}{3}$ [4]. The probability is $\frac{0}{3}$ if the box contains both gold and silver coins; otherwise, it is 1/3. Due to the presence of three boxes, the final probability is $\frac{0}{3}+\frac{1}{3}+\frac{1}{3}$ which is $\frac{2}{3}$. As a result, the final probability basically runs counter to what we thought. Probabilistic programming frequently comes across paradoxes of this nature. For instance, in the probabilistic programming language, when we work with a specific unit, such as meters, we obtain a result. However, when we convert meters into centimeters, we obtain a different outcome. This issue was addressed by Julian Jacobs in his study. In existing probabilistic programming languages, Julian Jacobs [1] found three paradoxes that defy the rule that a program should not be dependent on the scale of the parameter. In the following section, we discuss these three paradoxes in the probabilistic programming language.

## 2 IDENTIFIED PARADOXES

### 2.1 Paradox of Type 1

What is the expectation of h conditioned on h' $\sim$ $Normal(1.8, 0.5)$ being equal in the scenario where h $\sim$ $Normal(1.7, 0.5)$ explains a person's height is normally distributed with a mean of 1.7 meters and a probability of 0.5? The program is following [1]. First, we sample h, and then, by enclosing it in a conditional with a Bernoulli distribution, we perform the 'observe' only half of the time. Then we observe h', which is normally distributed to 1.8. The output of this program is 1.72. Now, if we multiply the value by 100 to convert the meter into centimeters, the program returns 170, which is equal to 1.7 meters. The difference when we simply change the unit is 0.02 meters. As a result, the results change depending on whether to use centimeters or meters.

```
h = rand(Normal (1.7, 0.5))
if rand(Bernoulli(0.5)){
        observe(Normal (1.8, 0.5), h)
}
return h
```

Initially, we sample h, then we do the observe only by half of the time by wrapping it in a conditional with Bernoulli distribution. Then we observe h' which is normally distributed to around 1.8. The program yields a result of 1.72. Now, if we multiply the value by 100 to convert the measurement from meters to centimeters, the program returns 170, which is equal to 1.7 meters. When we simply change the unit, there is a 0.02 meter difference. Therefore, whether we use meters or centimeters affects the results. It is unclear whether the program should be or whether the output should be rejected. Additionally, it is stated that the problem is general and not restricted to a specific probabilistic programming language tool.

### 2.2 Paradox of Type 2

Now, we measure weight with height. We want to sample a person's height, h $\sim Normal(1.7, 0.5)$, and weight, w $\sim Normal(60, 10)$. In the program below [1], we first sample the subject's height and weight, and then, based on the Bernoulli distribution, we observe a different height normally distributed to 1.8 and a different weight normally distributed to 70. The BMI is then calculated.

```
h = rand(Normal (1.7, 0.5))
w = rand(Normal (60, 10))
if rand(Bernoulli(0.5)){
        observe(Normal (1.8, 0.5), h)
else
        rand(Normal (60, 10), w)
}
bmi = w/(h*h)
```

Now, similar to paradox type 1, the program shows a change in BMI if we again convert the meter into centimeters by multiplying by hundreds. Although the parameters in this case are different and the Bernoulli distribution has no effect on the number of observers we have, the result still depends on whether to use centimeters or meters..

### 2.3 Paradox of Type 3

The program below [1] assumes that h=h' and that h has a normal distribution of 1.7 with probability 0.5 and h' has a normal distribution of 1.8 with probability 0.5. The program's output is 1.75.

```
h = rand(Normal (1.7, 0.5))
observe(Normal (1.8, 0.5), h)
return h
```

If the parameter were changed to h = log (h), then h $\sim LogNormal(1.7, 0.5)$ and h' $\sim LogNormal(1.8, 0.5)$. The program now returns 1.62 as its output.

```
h = rand(LogNormal (1.7, 0.5))
observe(LogNormal (1.8, 0.5), h)
return log(H)
```

One might believe that the first program and this one are interchangeable whether we use a normal scale or a logarithmic scale. Although the program does not contain any conditions like a Bernoulli distribution, the output still depends on the scale that we choose. The Borel-Kolmogorov paradox, which deals with conditional probability in relation to an event with a probability of zero, is closely related to this paradox.

## 3 MEASURE ZERO EVENT

We now need to figure out how to resolve these paradoxes. We can modify the observe construct to take an interval as a parameter to achieve that. Next, we assume that a rand(D) lies within

that interval. To put that into practice using rejection sampling, we take the sample, determine whether it actually falls within the interval, and set the weight to zero. Interval sampling can also be achieved by multiplying the weight by the probability that a sample lies within the interval. In interval sampling, this probability is nonzero. If we now expand upon the coding example from section 2.1, we arrive at the following.

```
h = rand(Normal (1.7, 0.5))
if rand(Bernoulli(0.5)){
        observe(Normal (1.8, 0.5), Interval(h, 0.1))
}
return h
```

The observer runs this conditionally based on a random Bernoulli distribution. In order to do this in centimeters, we need to adjust the width such as interval(h, 10). As a result, the program will have the same output and will not be affected by meter or centimeter. Now we consider the following example [1],

```
function bmi_example(eps)
        h = rand(Normal(170, 50))
        w = rand(Normal(70, 30))
        if rand(Bernoulli(0.5))
                observe(Normal(200, 10), (h, A*eps))
        else
                observe(Normal(90, 5), (w, B*eps))
        end
        return w / h^2
end
```

Now we took the limit of the variable going to zero and parameterized the probabilistic program by the desired interval's width. We must include a constant A with units of centimeters and a constant B with units of kilograms because the width has no units. Even when the width is zero, the relative sizes of these constants still matter. To see what value it converges to, with eps values of 0.1, 0.01, 0.001, and so forth, with 1000 trials, we could now sample BMI with importance sampling. Now, if we run this program with a different eps value each time, we will get a different result, which means that the randomness will change each time. The introduction of infinitesimal numbers can solve this problem.

## 4 INFINITESIMAL PROBABILITIES

The definition of an infinitesimal number is a pair consisting of a real number and an integer number. Jules Jacob's proposed interval observe is observe(D, (a, w*eps)), where (a, w) are intervals of infinitesimal width, where w yields a number and an infinitesimal symbol called eps. Consider adding this interval to the example from Section 2.1: observe (Normal (1.8, 0.1), (h, w*eps)). In the case that Bernoulli's outcome is valid, we should reject the Bernoulli trial if the Normal (1.8, 0.1) does lie in the interval. If symbolic infinitesimal is zero, then if the result is accurate and h converges to 1.7, the probability of the Bernoulli trial being rejected increases to 1. The BMI example from section 3 is examined in a manner similar to this. If we need to convert from one unit to another, such as from meters to centimeters or from centimeters to meters, we must also change the units for the two constants that we included as well as the constants values, which must be multiplied by 100 or 0.01 depending on the conversion we are taking care. The program behaves consistently when the unit changes thanks to this additional constant factor, along with A and B. Nothing must be altered in order to perform the importance sampling with infinitesimal intervals.

One of the terms mentioned in Jules Jacob's work is "soft computing," which is used to observe statements, especially when the distribution is the normal distribution, and allows the current trial to proceed with the probability density function, assuming it is not rejected. The ability to properly perform parameter transformation, which can be added as a language feature, is one advantage of infinitesimal intervals.

## 5 CONCLUSION

We initially observed, in Jacob's work, that certain probabilistic programs have paradoxical behavior. When the units of a program are changed, we see that similar programs provide different results. By measuring zero events and computing that with an infinitesimal interval, we can solve this problem. These intervals ensure that programs are changeless under parameter transformations and therefore that all paradoxes can be excluded.

## REFERENCES

[1] Jules Jacobs. 2021. Paradoxes of probabilistic programming: and how to condition on events of measure zero with infinitesimal probabilities. Proc. ACM Program. Lang. 5, POPL, Article 58 (January 2021), 26 pages. https://doi.org/10.1145/3434339
[2] Pfeffer, Avi. Practical probabilistic programming. Simon and Schuster, 2016.
[3] Birthday problem. In Wikipedia. https://en.wikipedia.org/wiki/Birthday_problem
[4] Bertrand's box paradox. In Wikipedia. https://en.wikipedia.org/wiki/Bertrand%27s_box_paradox