Programming Assignment 3

By: Shivangi(U35642613)

Introduction

The aim of this assignment is to implement a uniformity test and apply it in two scenarios. The uniformity test is basically used to determine whether a given set of samples is uniformly distributed or not. In this report, I will talk about the process of implementing the uniformity test and how I generated samples from uniform distribution and other types of distributions.

Section 1: Generating Random Samples

To generate random samples, I implemented two subroutines:

1. To generate samples from the uniform distribution
2. To generate samples from a distribution that is epsilon far from uniform in total variation distance.

Answer to Q1 part b

Let's talk about how the second sub-routine:

To generate random samples from a distribution on [n] that is ϵ_{far} from uniform in total variation distance and minimizes the expected number of collisions, I used the following subroutine:

1. Initialize an array p of length n with $p_i = (1 + \epsilon_{far}) / n$ for all i in $[n]$.
2. Repeat the following for k times, where k is the number of samples I want.
    a. Generate a random index i in $[n]$ with probability proportional to p_i.
    b. Output i.
    c. Update p_i to $p_i' = p_i / (1 + \epsilon_{far})$ and update all other elements of p to $p_j' = p_j \times (1 + \epsilon_{far}) / n$ for $j \neq i$.

This subroutine generates samples that are ϵ_{far} from uniform in total variation distance because the initial distribution p is ϵ_{far} from uniform, and each update step reduces the distance by a factor of $(1 + \epsilon_{far}) / n$ for the selected element i and increases the distance by a factor of $(1 + \epsilon_{far}) / n$ for all the other elements. After k iterations, the total variation distance between the distribution of the generated samples and the uniform distribution is at most $(1 + \epsilon_{far})^k / n^k$, which is negligible if k is sufficiently large.

Let's analyze why this algorithm minimizes the expected number of collisions. The expected number of collisions is defined as the sum of probabilities of all elements that are sampled more than once. Thus, to minimize the expected number of collisions, we want to make it less likely for any element to be sampled more than once.

By modifying the probabilities of the elements by adding or subtracting $(1 + \epsilon_{far}) / n$, I am creating a non-uniform distribution that has higher probabilities for some elements and lower probabilities for others. This change in probabilities makes it less likely for any element to be sampled more than once since the samples are now more spread out among the elements. Additionally, by normalizing the distribution so that the probabilities sum up to 1, we ensure that the modified distribution is still a valid probability distribution. Overall, this distribution minimizes the expected number of collisions because it

assigns higher probabilities to elements that have not been selected as often, which reduces the probability of collisions.

Section 2: Finding a sufficient number of samples and threshold

To find the optimal number of samples and threshold, I wrote a subroutine that works with probability 1 - δ. I initialized s to some value (usually 10) and collected s samples from uniform distribution and distribution that is ϵ_{far} from the uniform and observed the number of collisions. I repeated this process $2/\delta$ times. I then calculated the threshold t such that at least 1 - δ fraction of collision for a distribution that is ϵ_{far} from uniform is above t and at least 1 - δ fraction of collision from a uniform distribution is below t. If I couldn't find any such threshold, I increased s by a factor of 1.1 and tried again.

Section 3: Uniformity Testing on Real-world Data

Answer to Q4 part a

For this, I used the dataset from Python's sklearn California Housing Dataset. Specifically, I used the 'Averooms' column of this dataset to generate a distribution of the last two digits of values from this column. It is reasonable to assume that the last few digits of the average number of rooms in a house should be independent and uniformly distributed. This is because the average number of rooms in a house is dependent on various factors and these factors are likely to be independent and not correlated to the last few digits of the metric.

Answer to Q4 part b

I picked up the last two digits from this data and applied my subroutine to find a sufficient number of samples and threshold to this data. I found out that s = 46 and t = 14 are the optimal values for this dataset. I then tested the uniformity by finding the number of collisions for generating s samples and the output was only 7, which proved that this distribution is uniform.

Answer to Q4 part c

Similarly, I paired up the consecutive value from the above data and generated its distribution, and similarly performed the subroutine to find the optimal number of samples and threshold. I found that s = 19 and t = 1 are the optimal values. I then tested the uniformity by finding the number of collisions for generating s samples and the output was only 3, which proved that this distribution is not uniform.

Section 4: Random PIN or Password Generation

Answer to Q5 part a

I generated around 5000 values between 0 and 9 using ChatGPT. Then, I wrote a code to generate a single digit using a value q by adding q values and then taking a modulo 10. I varied the value of q and observed the distribution of digits obtained. I tested each of these distributions using the uniformity test used above with small δ = 0.1. I found that the lowest value of q for which the generated distribution passes the uniformity test is 1.

Answer to Q5 part b

Next, I tested the similarly for pairs and triples. I observed that for q=1, the resulting distribution of pairs of digits was not uniform, indicating that there was some correlation between consecutive digits. I increased the value of q to q=5 and observed that the resulting distribution of pairs of digits was uniform.

I also tried to test the distribution of triples but it was taking more than normal time to run, therefore, I stopped the program.

Overall, we observed that using a combination of q-digits can lead to a nearly uniform distribution of single digits. However, larger values of q may be required to obtain a uniform distribution of consecutive digits.

Conclusion:

Answer to Q6

Overall, I applied the uniformity test to multiple scenarios and it helped me get a better understanding of the concept. This assignment was harder than other programming assignments and hence it took me around 48 to 72 hours.