2024 NYC DISCRETE MATH REU
SAMPLE PROJECT

JOHANNA FRANKLIN

This project involves applications of concepts in computability theory, a branch of mathematical logic/theoretical computer science, to the concept of uniform distribution. I'll introduce the basic ideas of each here and then discuss the way this project would connect them.

1. Uniform distribution

A sequence of real numbers $\langle x_i \rangle_i$ that are all between 0 and 1 is said to be *uniformly distributed modulo one* if they appear "in the proportions you'd expect" within that interval. More formally, we say that

Definition 1.1. A sequence $\langle x_i \rangle_i$ of elements of $[0, 1]$ is *uniformly distributed modulo one* if for all a, b such that $0 \leq a < b \leq 1$, we have

$$\lim_{n\to\infty} \frac{|\{i \leq n \mid a \leq x_i \leq b\}|}{n} = b - a.$$

For example, let's take $a = \frac{1}{6}$ and $b = \frac{1}{2}$. We have $b - a = \frac{1}{3}$, so if $\langle x_i \rangle_i$ is uniformly distributed modulo one, as you look at more and more values of x_i, the proportion of them between $\frac{1}{6}$ and $\frac{1}{2}$ will approach $\frac{1}{3}$.

This notion can be generalized to sequences of arbitrary real numbers: rather than asking whether $a \leq x_i \leq b$, we ignore the integer part and ask whether the fractional part of x_i, $[x_i]$, is between a and b instead.

Remark 1.2. It might be interesting at this point to try to come up with a sequence $\langle x_i \rangle_i$ that is uniformly distributed modulo one and a sequence $\langle y_i \rangle_i$ that is not uniformly distributed modulo one. *Hint: The latter is probably easier.*

In 1916, Weyl framed this question in terms of sequences generated by selecting a real x and a sequence of distinct integers $\langle a_i \rangle_i$ to form the sequence $\langle a_i x \rangle_i$. He proved that for almost every x, this sequence is uniformly distributed modulo one.[1]

1.1. Further reading.

L. Kuipers and H. Niederreiter's 1974 book *Uniform Distribution of Sequences* is a classic in this area [1], and Pete Clark has a set of notes on it: `http://alpha.math.uga.edu/~pete/udnotes.pdf`.

Date: January 31, 2024.

[1] If you don't know what "almost every x" means, just take it to mean "all but a negligible amount of xs" and you'll be OK for the moment.

2. Computability theory

In computability theory, it's very easy to get bogged down in technical details. For the moment, let's talk about what you can and can't do with a computer program.[2]

Definition 2.1. A sequence of nonnegative integers $\langle a_i \rangle_i$ is *computable* if there is a computer program that produces a_i when given i as input.

Note 2.2. Not all sequences of (nonnegative) integers are computable! If you've learned about cardinalities, you can figure out that there are uncountably many sequences of integers—but only countably many computer programs, so not every sequence of integers can be produced by a computer program.

Now we'll turn our attention to sets of integers instead of sequences of integers. As you read this next definition, please keep in mind that a computer program might not give an answer for a given input (or, indeed, any input at all).

Definition 2.3. A set A of integers is

(1) *computably enumerable (c.e.)* if there is a computer program P_1 that outputs precisely the elements of A.
(2) *computable* if there is a computer program P_1 that outputs precisely the elements of A and a computer program P_2 that outputs precisely the elements of $\mathbb{N} - A$.

In other words, c.e. sets are defined via positive information only (if a number is in the set, you can eventually find out by running the right computer program), and computable sets are defined via both positive and negative information (you can eventually find out whether any number is in the set or not by running both the "in" computer program and the "out" computer program: eventually, one of them will output the number you're curious about and you'll know which one it is).

We can also talk about using one set to compute another:

Definition 2.4. Let A and B be subsets of $\mathbb{N}$. We say that A is *Turing computable* from B ($A \leq_T B$) if there is a computer program that, for every n, can determine whether $n \in A$ given the additional ability to ask finitely many questions of the sort "Is $m \in B$?".

Turing reducibility ($\leq_T$) is a partial order on subsets of $\mathbb{N}$ and is used to compare the computational strengths of these subsets.

2.1. Further reading.

Rebecca Weber's 2012 book *Computability Theory* is a nice introduction to this area aimed at undergraduates [2].

3. Unification

In 2013, Avigad united computability theory and uniform distribution with the following definition:

Definition 3.1. A real $x \in [0, 1]$ is *UD-random* if and only if for every computable sequence $\langle a_i \rangle_i$ of distinct integers, $\langle a_i x \rangle_i$ is uniformly distributed modulo one.

In this project, we will explore the computability-theoretic aspects of the UD-random reals and, perhaps, other closely related classes of reals.

[2] It really doesn't matter which programming language you're using.

References

[1] L. Kuipers and H. Niederreiter. *Uniform distribution of sequences*. Pure and Applied Mathematics. Wiley-Interscience [John Wiley & Sons], New York-London-Sydney, 1974.

[2] Rebecca Weber. *Computability theory*, volume 62 of *Student Mathematical Library*. American Mathematical Society, Providence, RI, 2012.