THE RANDOMNESS OF PRIME ENCODINGS AND THE COMPUTATIONAL COMPLEXITY OF INTEGER FACTORISATION

Aidan Rocke
aidanrocke@gmail.com

June 25, 2021

1 Motivation

In order to understand the importance of integer factorisation in this analysis, it is important to understand deep neural representations [3] which is what makes deep neural networks so effective. For neural networks to accurately represent a relation between integers and prime numbers with strong generalisation ability, it needs to discover an algorithmic formulation of the unique factorisation theorem. That is to say, an algorithmic version of Euclid's proposition:

> Any number either is prime or is measured by some prime number.

Such an algorithm would be nothing less than a method for integer factorisation. For the remainder of this analysis we shall assume that $P \neq NP$ in computational complexity theory.

2 Refining the original hypothesis

So far the hypothesis that prime encodings are random has been carefully evaluated using a variation of Yao's next-bit test as well as a primality test for all prime numbers under 10^9. To be precise, given a finite number of integers $X := [1, N] \subset \mathbb{N}$ with binary representation mapped to $Y := X_N \in \{0,1\}^*$, a function is approximated using a feedforward neural network(f_θ):

$$f_\theta : X \rightarrow Y \tag{1}$$

whose ability to correctly evaluate the primality of integers is tested on integers $[N+1, 2N]$. In agreement with the hypothesis, the true positive rate of such a function does not exceed 50%. This experimental verification is significant because I am implicitly using the universal approximation property of deep neural networks [6].

Now, we'll note that this provides evidence for the existence of one-way functions in the following manner. Thanks to the Sieve of Eratosthenes, the time complexity for data generation $(X = [1, N], Y = X_N)$ is on the order of $\sim N \log \log N$. However, given an integer M sampled uniformly from $[N+1, 2N]$ there is no known integer factorisation algorithm that is polynomial in the description length of M(i.e. $\sim \log M$), and feedforward neural networks can only simulate algorithms that are polynomial time with respect to their input size.

In principle, this is because prime sieves exploit the multiplicative structure of the integers and integer multiplication is relatively easy from a computational complexity perspective. Given this analysis, the hypothesis ought to be refined as follows:

$$\mathbb{E}[K^{\text{poly}}(X_N)] \sim \pi(N) \cdot \ln(N) \sim N \tag{*}$$

where $K^{\text{poly}}(\cdot)$ denotes a universal compressor that can only simulate Turing Machines with polynomial time complexity.

It is worth noting that the hypothesis (*) is consistent with the assumption that underlies RSA encryption and other cryptographic protocols; integer factorisation can't be solved in polynomial time.

References

[1] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions. In Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science, 1982.

[2] Leonid Levin (2003). "The Tale of One-Way Functions". ACM. arXiv:cs.CR/0012023

[3] Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016). Deep Learning. MIT Press

[4] Lenstra, Arjen K (1988). "Fast and rigorous factorization under the generalized Riemann hypothesis". Indagationes Mathematicae. 50 (4): 443–454.

[5] Yanyi Liu, Rafael Pass. On One-way Functions and Kolmogorov Complexity. Arxiv. 2020.

[6] Hornik, Kurt (1991). "Approximation capabilities of multilayer feedforward networks". Neural Networks.