Classification of Natural Numbers, into Composite, Prime and Gaussian Prime Using a Multi-Layered Perceptron

Nathan Hall  
Computer Science  
Missouri State University  
Springfield, MO USA  
Hall257@live.missouristate.edu

Rohan Saha  
Computer Science  
Missouri State University  
Springfield, MO USA  
line 5: email address

*Abstract*—Prime numbers are an area of interest to both mathematics and information security. This paper will attempt to demonstrate how a multi-layer perceptron (MLP) can be implemented to predict whether a number, given its results to certain primality/composite tests, is prime, with an accuracy of, or near, one-hundred percent.

Keywords—prime numbers, multi-layer perceptron, primality

# Introduction

Prime numbers are not just a mathematical curiosity. They can be represented in the real-world. Take for example 7 people who want to play a game. There is no way to split the seven people into equal size teams, except all on 1 team, or 7 teams of 1 player each.

Many think of prime numbers as only being members of the natural numbers, but mathematicians hold that their negative counterparts are also primes. For many cases, the sign of the prime is irrelevant, so for this paper we will only be dealing with the non-negative primes.

Prime numbers have a long history. One of the earliest methods to determine prime numbers is the Sieve of Eratosthenese [1], where given an index *n*, the sieve method will find all prime numbers up to the given index. The method is deterministic and has an approximate run-time of . Since it is sub-quadratic runtime, this is pretty good. The drawback is that it starts from scratch each time, and to reach record large primes will take a large amount of computer resources and/or time. This has led mathematicians to derive many formulas and tests, some based on unique properties of primes, to either find or predict them. Although, these formulas and tests can lead to false outcomes, known as pseudoprimes.

Fast forwarding through time, we find the modern world depending on prime numbers for a range of security needs. From online transactions and banking to secure communication, prime numbers are crucial to preventing unwanted and/or unauthorized access to personal and government information.

This paper attempts to capitalize on the work done by many people, across time, to form an ensemble of tests, take those results, feed them into an MLP, and try to accurately identify whether the number used in those tests, is a prime or composite, in constant runtime.

# Primality Tests

## General Algebraic Property of Primes

The first attribute we will discuss is a simple property that all prime numbers have. All primes greater than 3, can be written in the form , where k is any integer. The proof of this will be left to the reader. For our purposes, this implies that given a prime , . Therefore, we will be using mod 6 as an attribute.

## Selfridge Conjecture

The Selfridge Conjecture [2], also known as the PSW Conjecture, has two conditions that need to be met, 1) the number is odd, and 2) it is congruent to prior to being subjected to the algorithm of primality testing. For our purposes, we have ignored the first conditions, but include the second condition, of as an attribute. The algorithms used to test are and , where is the kth Fibonacci number.

The first test is a simple use of Fermat’s Little Theorem, , using base 2.

A note of warning. There is a closed form formula to calculate the nth Fibonacci number, which is , where , and . As you can see, it is an exponential function, which can lead to overflow. With a closer look, we found that the python 3 language, due to rounding, the formula stops accurately returning Fibonacci numbers at . Therefore, generating the recursive sequence is preferred. This shows it suffers from the same defect of that the Sieve of Eratosthenese does. For this reason, in future analysis, it will be desired to be removed from the list of attributes of the MLP.

## Baillie-PSW

The Baillie-PSW [3] test is akin to the Selfridge Conjecture, with the primary difference of using Lucas numbers instead of Fibonacci numbers. The Lucas numbers has the same implementation drawbacks as mentioned before, concerning Fibonacci numbers, and this too will be analyzed in context of attribute removal.

## Gaussian Primes

Gaussian primes are a proper subset of real primes. There are three properties for Gaussian primes[7], which only one needs to be satisfied. Two are essentially the same. Given , where either or is 0, then the non-zero term must be itself a prime. This is a bit trivial, but the remaining property is the one of interest to our dataset. Given , where and are both non-zero, then must be a prime in the reals.

The given property descriptions are intended for starting from the complex number and determining if it is a Gaussian prime. To start from the reals, and see if it can be factored into complex integers is what we’re interested in. It turns out that given a real prime and if , then is not a Gaussian prime. If it is congruent to 1, using modulo 4, then it can always be expressed in terms of , which a sum of squares can be factored into . Take 5 as an example. It is a prime in the reals, but 5 can be expressed as . Therefore 5 is a Gaussian composite. The proof of this relationship is left to the reader. Thus, we use mod 4 as an attribute in our dataset.

# Multi-layer Perceptron

An MLP is a class of feedforward artificial neural network. It consists of at least three layers of nodes. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLPs utilizes a supervised learning technique called backpropagation for training.[4][5] Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.[6]

Feedforward networks such as MLPs are like tennis, or ping pong, where they’re mainly involved in two motions, a constant back and forth. You can think of this ping pong of guesses and answers as a kind of accelerated science, since each guess is a test of what we think we know, and each response is feedback letting us know how wrong we are.

In the forward pass, the signal flow moves from the input layer through the hidden layers to the output layer, and the decision of the output layer is measured against the ground truth labels.

In the backward pass, using backpropagation and the chain rule of calculus, partial derivatives of the error function w.r.t. the various weights and biases are back-propagated through the MLP. That act of differentiation gives us a gradient, or a landscape of error, along which the parameters may be adjusted as they move the MLP one step closer to the error minimum. This can be done with any gradient-based optimization algorithm such as stochastic gradient descent. The network keeps playing that game of tennis until the error can go no lower. This state is known as convergence.

Figure 1 shows the basic architecture of the MLP. The left most layer, known as input layer consists of set of neurons {xi | x1 + x2 + x3 +… xn}, each neuron in the input layer receives the input data and passes to the next layer which is known as the hidden layer. The neurons in the hidden layer is associated with a weight value. This weight value shows gives an importance to each neuron, for example if neuron x3 is important as an input attribute, then the weight value for node a3 in the hidden layer will have much greater weight than the other neurons. Each neuron in the hidden layer is also associated with an activation function, these functions does some processing to the input node. The activation function is applied to the dot product of the input value and the weights into a node to produce an output. A bias value is usually added to indicate that the activation function is applied or the node fires if the value of

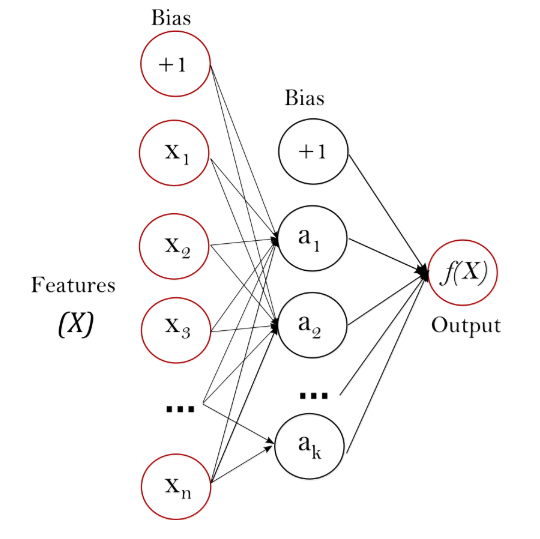


Fig. 1

# Data Results

## Hidden Layers

In the first table, we ran the MLP with varying number of hidden layers. The activation function and the number of maximum iterations were held constant. The accuracy of each experiment is recorded below.

|  |  |  |  |
| --- | --- | --- | --- |
| Hidden Layers | Activation | Max Iterations | Accuracy |
| 1 | relu | 100 | 0.828804348 |
| 2 | relu | 100 | 0.828804348 |
| 3 | relu | 100 | 0.913043478 |
| 4 | relu | 100 | 0.064673913 |
| 5 | relu | 100 | 1.00 |
| 6 | relu | 100 | 1.00 |
| 7 | relu | 100 | 1.00 |
| 8 | relu | 100 | 1.00 |
| 9 | relu | 100 | 0.99728609 |
| 10 | relu | 100 | 0.972826087 |

## Maximum Iterations

In the second table, the number of maximum iterations is varied, while holding the number of hidden layers and the activation function constant. The accuracy of each experiment is recorded below.

|  |  |  |  |
| --- | --- | --- | --- |
| Hidden Layers | Activation | Max Iterations | Accuracy |
| 5 | relu | 10 | 0.894021739 |
| 5 | relu | 20 | 0.89673913 |
| 5 | relu | 30 | 0.836956522 |
| 5 | relu | 40 | 0.872282609 |
| 5 | relu | 50 | 0.828804348 |
| 5 | relu | 60 | 0.828804348 |
| 5 | relu | 70 | 0.994565217 |
| 5 | relu | 80 | 0.997282609 |
| 5 | relu | 90 | 0.997282609 |
| 5 | relu | 100 | 1.00 |
| 5 | relu | 110 | 0.994565217 |
| 5 | relu | 120 | 0.994565217 |

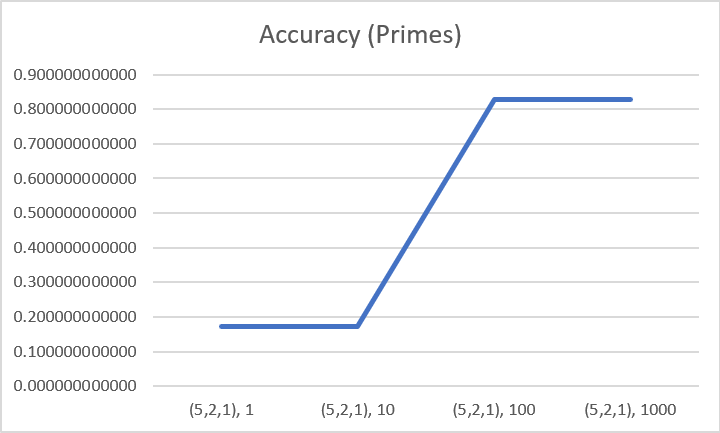
## Activation Functions

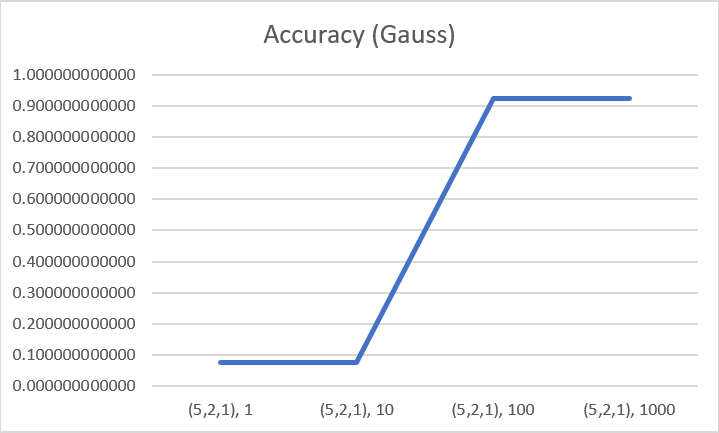
In the third, and final, table, the variable of interest is the activation functions. The number of hidden layers and maximum iterations were held constant. The accuracy of each experiment is recorded below.

|  |  |  |  |
| --- | --- | --- | --- |
| Hidden Layers | Activation | Max Iterations | Accuracy |
| 5 | relu | 100 | 1.00 |
| 5 | tanh | 100 | 0.991847826 |
| 5 | logistic | 100 | 0.828804348 |
| 5 | identity | 100 | 0.997282609 |

## Primes and Gaussian Primes Accuracy

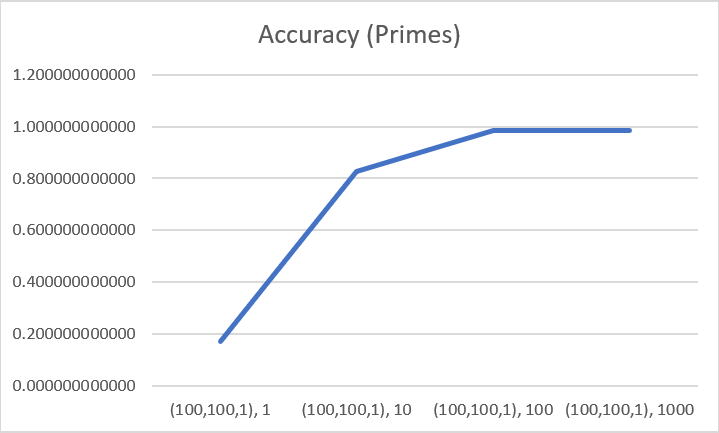
In the following two graphs, the independent variable is number of iterations, and the dependent variable is the accuracy. Accuracy in the first graph is pertaining to the MLP’s ability to correctly classify if the given data is that of a prime number, or a composite. The second graph’s accuracy pertains to the MLP’s ability to correctly classify if the given data is that of a Gaussian prime, or a Gaussian composite.



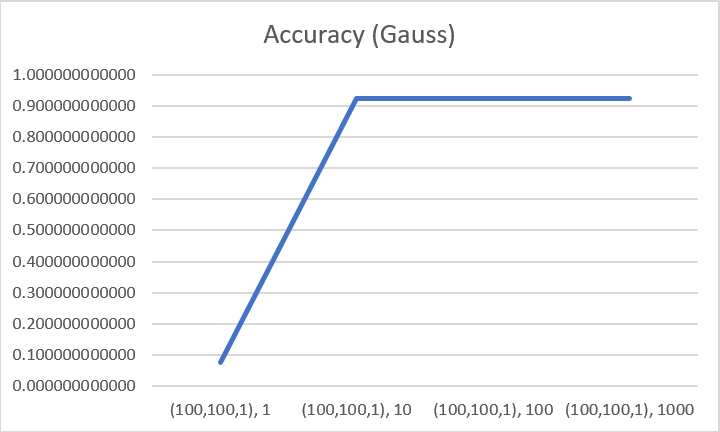


As you can see, the MLP of a (5,2,1) node structure is more adept at correctly classifying Gaussian primes, which is a proper subset of primes.

In the next two graphs, the independent and dependent variables are the same as before, but with a (100, 100, 1) node structure. The accuracy of the first graph, as before, pertains to accurately classifying whether the given data is that of a prime, or a composite.



The second graph’s accuracy represents the same node structure’s performance at accurately classifying Gaussian primes.



In this case, the MLP node structure of (100,100,1) is more adept at classifying prime numbers, than at classifying Gaussian primes.

The effect of node structure on accuracy was expected, but in respect to being more or less accurate with one set of primes, than a subset of them, was unexpected. Further investigation will be conducted in future research on this.

##### References

1. K. Rosen, Elementary Number Theory, 5th ed., Pearson, 2005, pp. 70.
2. Wikipedia, [https://en.wikipedia.org/wiki/John\_Selfridge#Selfridge%27s\_Conjecture\_about\_Primality\_Testing](https://en.wikipedia.org/wiki/John_Selfridge#Selfridge's_Conjecture_about_Primality_Testing)
3. Wolfram MathWorld, <http://mathworld.wolfram.com/Baillie-PSWPrimalityTest.html>
4. Rosenblatt, Frank. x. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC, 1961
5. Rumelhart, David E., Geoffrey E. Hinton, and R. J. Williams. "Learning Internal Representations by Error Propagation". David E. Rumelhart, James L. McClelland, and the PDP research group. (editors), Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundation. MIT Press, 1986.
6. Cybenko, G. 1989. Approximation by superpositions of a sigmoidal function Mathematics of Control, Signals, and Systems, 2(4), 303–314.
7. Wolfram MathWorld <http://mathworld.wolfram.com/GaussianPrime.html>