Using a Feedforward Neural Network (FNN) to Classify Polynomial Roots

Yagna Patel
yagnapatel@berkeley.edu

March 23th 2017

1 Introduction

Univariate polynomials are polynomials of one indeterminate (variable). A degree-n univariate polynomial is of the form

$$P_n(z) = a_n z^n + a_{n-1} z^{n-1} + \ldots + a_1 z + a_0 \quad (a_n \in \mathbb{C})$$

Now, the fundamental theorem of algebra states that

Theorem 1. *Every polynomial function of degree-n, such that $n > 0$, has exactly n complex roots.*

These roots can be integers ($\mathbb{Z}$), non-integer reals ($\mathbb{R} \setminus \mathbb{Z}$), or complex ($\mathbb{C}$). For example, consider the polynomial

$$f(z) = z^4 - \frac{3}{2}z^3 + \frac{3}{2}z^2 - \frac{3}{2}z + \frac{1}{2}$$

Notice that

$$\begin{aligned} f(z) &= z^4 - \frac{3}{2}z^3 + \frac{3}{2}z^2 - \frac{3}{2}z + \frac{1}{2} \\ &= (z-1)(z-0.5)(z-i)(z+i) \end{aligned}$$

Therefore $f(z)$ has four roots: two complex roots, one integer root, and one non-integer real root.

2 Problem

There exist multiple methods of finding the number of roots of a polynomial. For example, one can use Rational Root Theorem to find the number of rational roots of a polynomial, Descartes' rule of signs to find the number of real roots of a polynomial, etc. In this project, we will train a Feedforward Neural Network (FNN) to classify a polynomial's roots into three types: integers ($\mathbb{Z}$), non-integer reals ($\mathbb{R} \setminus \mathbb{Z}$), and complex ($\mathbb{C}$) roots.

To most people, this problem that is posed doesn't seem too interesting. However, I am quite interested in seeing if this can provide a new and different way of viewing polynomials.

To my knowledge, there has been substantial research done in finding polynomial roots using neural networks, however, classification of polynomial roots, or finding the number of roots in general remains fairly untouched. There exists a method of using a FNN to find the number of real roots of a polynomial [1], however, there are many holes in this paper that I will attempt to fill in with my project.

3 FNN Design

A neural network is a universal approximator. It can approximate a continuous function in any compact set. Since polynomials are continuous functions, we have no problem there. However, it may be a challenge to find the set. The optimal set would be the one that contains all the roots of the polynomial. However, there do exist methods which we can utilize to compute the set. Additionally, we also have *a priori* information that we can encode into the FNN. This includes information such as knowing that there exists a relation between polynomial roots and the coefficients of the polynomial. Finally, WLOG we will assume that the coefficient of the highest degree term in the polynomial is 1.

*Note: This may possibly change. See **my GitHub** for updates.*

4 Generating Dataset

To train my FNN, we will need a dataset of polynomials and its classified roots. However, we don't have to go looking for a dataset. Instead, we can generate our own! A general overview of the method is as follows

1. `Generate a random number n, the degree of our polynomial`

2. `Generate n complex numbers, the roots of our polynomial.`
3. `Compute the polynomial using the n roots.`
4. `Classify the roots into the three categories: integer, non-integer real, and complex roots.`
5. `Repeat.`

A detailed approach is provided **here**.

References

[1] B. Mourrain, N.G. Pavlidis, D.K. Tasoulis, M.N. Vrahatis, Determining the number of real roots of polynomials through neural networks, *Computers and Mathematics with Applications 51* 527-536 (2006).