Algebraic Coding Theory

Abstract:

When transmitting data through any medium, the data will usually be susceptible to corruption. In many applications, this corruption can be detrimental to the usability of the data. For example, when transmitting encrypted data, a single bit error could propagate to corrupt the decrypted text. The goal of our project was to explore the Modern Algebra concepts used when creating codes for data transmission. These error-control codes are methods of adding redundancy and repetition to data in order to make up for the errors in transmission. Our main goal is to discover the role of group theory, specifically finite fields, in some of the basic error-code implementations.

When creating these error-control codes, we first start by determining the type of problem we are trying to solve. Some problems only require that we find out whether our received data is valid or not. This is called error-detection. To explore this, we will look at ISBN numbers for books. On the other hand, it is often necessary to not only detect errors in the data but also to correct them. To explore this idea, we will implement a simple 2-D Parity Matrix. This is a method for encoding data before transmission that allows for single bit error-correction.

Finite Fields play a large role in the study of error-control codes. Recall that a Field is a commutative ring in which every non-zero element has an inverse and the order of a field is always a prime power. Therefore, when constructing codes, we will use elements in *GF(q)* where *q* is a prime power. That is, $q = p^k$, where p is prime and k is a positive integer (GF stands for Galois Field). With this, we will now explore our 2 examples of error- control codes, seeing how they use these properties of finite fields.

The 10-digit ISBN numbers are a set of n-tuples over the field *GF(11)*. The elements of *GF(11)* are *{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, X}*, where X is just the number 10 (used to distinguish between a 1 followed by a 0). We can see that the order of GF(11) is 11 which is prime. The first 9 digits of the 10 digit ISBN number contains data about the book (author, book number, etc), the 10th digit is a number generated so that the following relationship is satisfied:

$$\sum_{i=1}^{10} ix_i \equiv 0 \pmod{11}.$$

ISBN codes are of the type know as *linear block codes*. Briefly, this means that the code-words (n-tuple elements) are closed under addition and that the all-zero code-word is a part of the code. It also means that the minimum number of ways 2 elements differ (*Hamming distance*) is equal to the minimum *Hamming weight* of any non-zero element. This is equal to 2 for the ISBN example. Hamming distance Hamming weight are defined more clearly in our presentation. Using these properties we can conclude that the ISBN coding scheme can detect a single error. This is because, if 2 errors occur, it is possible that these errors result in a valid, but incorrect, ISBN number.

We will now briefly cover a simple error-correcting scheme using 2-D parity. Elements of a 2-D parity code come from GF(2) meaning they are binary. To construct the parity matrix, the original data is formed into a matrix. The rows and columns are then appended with a 0 or 1, creating another row and column on the original matrix. This is our codeword to be sent. The 0 or 1 is appended to make the corresponding row and column have an even number of 1's. When the message is received, lets assume a single bit error has occurred. The decoder sees which row

and column has an odd number of 1's and switches the corresponding bit. Generally, the more dimensions added to a parity matrix, the more errors it can correct.

An understanding of finite fields is necessary to properly study error-control codes. Figure 1 can be used to visualize the sending of data as an isomorphism from finite field S to finite field R. With S being a subring of R. The closure and existence of inverse properties of finite fields are used to make decisions during decoding. We can also see that any code consists of a prime number of code-words. Without basic knowledge of the properties of finite fields, it would be easy to make a mistake trying to construct an efficient code.