Smart Codes

UCLA Math Circle
09/30/2012

1 Error Detection

Sequences of numbers, or codes, are ubiquitous in our lives. Some of them, such as credit card number and bank account numbers, are very important. They are also very sensitive to man-made and mechanical errors. So it is important that some of these numbers have the ability to detect, and even correct possible errors.

A nice example of codes with the ability to detect errors is the ISBN, a number associated to every book. There are two versions: one with 10 digits, one with 13 digits. For example, the 10-digit ISBN the first *Twilight* novel is 0316160172. The first digit 0 tells us that the book is in English. The next eight digits contains the publishing information of the book, and the last digit is used to to check that this is a *valid* ISBN number. Suppose the first 9 numbers in a sequence of 10 digits looks like

$$\overline{a_1 a_2 \cdots a_9}.$$

Calculate the following sum modulo 11

$$\sum_{k=1}^{9} k \cdot a_k, \tag{1}$$

i.e. find its remainder when it is divided by 11. The sequence is a valid ISBN number if and only if the remainder agrees with the last digit, where we use the digit X to represent the remainder 10.

In the *Twilight* example above, we have

$$\overline{a_1 a_2 \cdots a_9} = 031616017.$$

The number in the expression (1) is

$$\sum_{k=1}^{9} k \cdot a_k = 1 \cdot 0 + 2 \cdot 3 + 3 \cdot 1 + 4 \cdot 6 + 5 \cdot 1 + 6 \cdot 6 + 7 \cdot 0 + 8 \cdot 1 + 9 \cdot 7 \tag{2}$$

$$= 145 \equiv 2 \pmod{11}. \tag{3}$$

Since the remainder agrees the last digit 2, this is a valid ISBN number.

Exercise 1. Calculate the sum (1) for the following sequences of numbers to see if it could be a valid ISBN number.

(a) 0439023521

(b) 311001436X

(c) 0439784542

Exercise 2. Given a sequence $\overline{a_1a_2\dots a_{10}}$, consider the sum

$$10 \cdot a_1 + 9 \cdot a_2 + \cdots + 2 \cdot a_9 + a_{10}. \tag{4}$$

What is its remainder when divided by 11 if $\overline{a_1a_2\dots a_{10}}$ is the number in *Exercise 1(a), 1(b), 1(c)* respectively.

Exercise 3. Show that a sequence $\overline{a_1a_2\dots a_{10}}$ could be a valid ISBN number if the sum in (4) divisible by 11.

Exercise 4. In *Exercise 1*, is it possible to change a single digit in a valid ISBN number such that it is still valid? Vice versa, is it possible to make an invalid ISBN number into a valid ISBN number? What about swapping two adjacent digits?

Besides the 10-digit ISBN number, the 13-digit ISBN number is also in use. For example, the 13-digit ISBN of the fourth *Harry Potter* book is 9780439139601, compare to the 10-digit ISBN number 0439139600. The first three digits of the 13-digit ISBN number provide more information about the book, and the last digit is still used for an error-detection. However, its error-detection mechanism is different from the one we described above. Given a sequence of twelve digits between 0 and 9, say $\overline{a_1a_2\dots a_{11}a_{12}}$, we need to consider the following sum modulo 10

$$\sum_{k=1}^{12}(2+(-1)^k)a_k = a_1 + 3 \cdot a_2 + a_3 + 3 \cdot a_4 + \cdots + a_{11} + 3 \cdot a_{12}. \tag{5}$$

It needs to agree with the last digit for $\overline{a_1a_2\dots a_{11}a_{12}}$ to be a valid 13-digit ISBN number.

Exercise 5. Check whether the following sequence is a valid 13-digit ISBN number

(a) 9780439139601

(b) 9780439784542

(c) 9781178050233

(d) 9783110014635

Exercise 6. Try *Exercise 4* for 13-digit ISBN numbers. Compare the results with those of *Exercise 4*.

Exercise 7. For the 13-digit ISBN number, modify the error-detection mechanism, as best as you could, so that it can detect transposition of neighboring digits, in addition to detecting other simple errors.

2 Error Correction

From the exercises above, we see that both the 10-digit and 13-digit ISBN numbers can detect single digit error. However, it cannot correct the error automatically upon discovering it, since any digit could be the one with error. In fact, early computers do not have such capability and would stop if it runs into an error while reading codes. This was an annoying problem and people tried to create smarter codes that could correct simple errors. For simplicity, we only use the digits, or bits, 0 and 1 in a string from now on. Also, we assume all codewords in a code have the same length.

The simplest idea is to repeat each bit several times. For example, if we have the digits 1010110 and encode it by repeating each digit 3 times, then we have the codeword

$$111000111000111111000$$

Suppose we see 111000110100111111000 instead, then we can immediately fix the error and recover the original string of digits 1010110 Not only does this method detect the existence of single digit and transposition errors, but also their locations and hence fix them.

Suppose we fix an integer k to be the length of the string and r the number of times a digit is to be repeated Then the total length of the codeword is $n = k \cdot r$. All such possible

codewords together is called a ***repeating*** $[n, k]$***-code***. For example, a repeating $[4, 2]$-code contains the codewords $0000, 0011, 1100$ and 1111.

To measure the efficiency of codes, we can use the quantity information rate defined by

$$R = \frac{\log_2(w)}{n}, \tag{6}$$

where w is the total number of codewords in the code, and n is the length of each codeword. For example, the information rate of the repeating $[4, 2]$-code is $\frac{\log_2(4)}{4} = \frac{1}{2}$.

Exercise 8. Write down all the codewords in a repeating $[6, 2]$-code and calculate its information rate.

Exercise 9. What is the information rate of a repeating $[n, k]$-code? Can it correct single-digit error?

An example of a more complicated coding scheme is Hamming's square code. Begin with a message with 4 bits. First, write this message in a 2×2 square. Then compute the sum of each row, respectively each column, and write it at the end of the row, respectively the bottom of the column. Finally compute the sum of all entries and write it in the lower right corner to complete a 3×3 square. Reading out the 9 bits then gives the codeword. Since we only have 0's and 1's in the alphabet, the addition rules will be

$$0 + 0 = 0, 1 + 0 = 1, 1 + 1 = 0.$$

For example, if the message is 1011, then the codeword is 101110011.

$$1011 \longrightarrow \begin{array}{|cc|} \hline 1 & 0 \\ 1 & 1 \\ \hline \end{array} \longrightarrow \begin{array}{cc|c} 1 & 0 & 1 \\ 1 & 1 & 0 \\ \hline 0 & 1 & \end{array} \longrightarrow \begin{array}{cc|c} 1 & 0 & 1 \\ 1 & 1 & 0 \\ \hline 0 & 1 & 1 \end{array} \longrightarrow 101110011$$

Exercise 10. The following codewords are encoded using the method above. Correct any single-digit or transposition error if there is any.

(a) 110110011

(b) 100101011

(c) 001010110

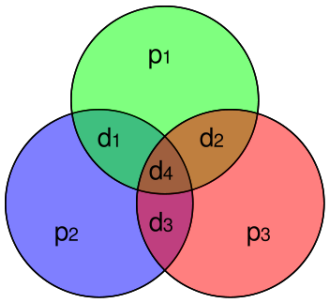
Exercise 11. If a single-digit errors is present in the codeword, can it be discovered and fixed? What about a transposition?

Exercise 12. Show that the information rate of Hamming's 2×2 square code is $\frac{4}{9}$. Generalize this coding method to $m \times n$ rectangular code and find its information rate. If $m \cdot n$ is fixed, when is the information rate maximum? Compare it to the maximal information rate of the repeating code.

Exercise 13. Is the $m \times n$ rectangular code capable of correcting single-digit errors or transposition errors?

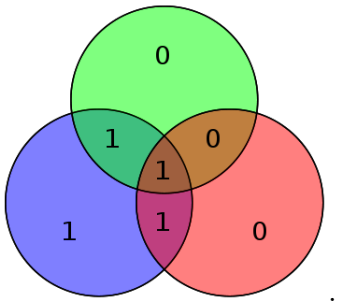
Exercise 14. If the last digit in the codeword is removed, can we still decode the message? What about fixing single-digit error or transposition error? What is the information rate?

From *Exercise 12*, we know that the $m \times n$ rectangular code is much more efficient at encoding information than the simple repeating code. In fact, the information rate increases as the length of the message increases. For message word with 4-bits, we will describe another encoding mechanism, which has higher information rate than Hamming's 2×2 square code. It is called Hamming's $[7, 4]$-code. Suppose we have the 4-bit word $\overline{d_1 d_2 d_3 d_4}$. Define three bits $p_1, p_2, p_3 \in \{0, 1\}$ such that in the diagram below, four bits in the same circle add up to 0.



The codeword is then $\overline{p_1 p_2 d_1 p_3 d_2 d_3 d_4}$.

For example, if the bits are 1011, then the diagram above becomes



So $p_1 = 0, p_2 = 1, p_3 = 0$ and the codeword is 0110011.

Suppose you are given a 7-bit message $\overline{a_1 a_2 \dots a_7}$. Define the check bits c_1, c_2, c_4 by

$$\begin{aligned} c_1 &= a_1 + a_3 + a_5 + a_7, \\ c_2 &= a_2 + a_3 + a_6 + a_7, \\ c_4 &= a_4 + a_5 + a_6 + a_7. \end{aligned}$$

Then $\overline{a_1 a_2 \dots a_7}$ is a codeword if $c_1 = c_2 = c_4 = 0$. Amazingly if it is different from a codeword by a single digit, the binary number $c_4 c_2 c_1$ gives the location of error digit. That is why the order $\overline{p_1 p_2 d_1 p_3 d_2 d_3 d_4}$ is not arbitrary.

Exercise 15 List all the codewords in Hamming's $[7, 4]$-code. What is the information rate? How does it compare to the information rate of Hamming's 2×2 square code?

Exercise 16 Pick your favorite 3-digit decimal number. Convert it to binary number and break it into blocks of 4 bits (add appropriate number 0 to the front to make the number of bits divisible by 4). Then encode each block using Hamming's $[7, 4]$-code.

Exercise 17 Pick a codeword and change a single digit. Calculate the check bits c_1, c_2 and c_4. Treat $c_4 c_2 c_1$ as a 3-digit binary number and convert it to decimal. Do this a few times and what pattern do you notice? Can you prove it?

Exercise 18 Can you generalize Hamming's $[7, 4]$-code to encoding messages whose numbers of bits are powers of 2? (*Hint*: In general, it is Hamming's $[2^k + k + 1, 2^k]$-code for any integer k).