Math Circle: Error Correcting Codes

Prof. Wickerhauser

1 Codes

A *code* is way to convert words into 0s and 1s to send over the internet. What works for words will also work for sounds, pictures, and video, too. Let's use just words for now, to keep down the amount of arithmetic.

1. Base 2 numbers use just 0s and 1s, instead of the ten symbols $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Let's use them to represent the 26 letters of the alphabet:

letter	base 10	base 2	letter	base 10	base 2	letter	base 10	base 2
	0	0	i	9	1001	r	18	10010
a	1	1	j	10	1010	s	19	10011
b	2	10	k	11	1011	t	20	10100
c	3	11	l	12	1100	u	21	10101
d	4	100	m	13	1101	v	22	10110
e	5	101	n	14	1110	w	23	10111
f	6	110	o	15	1111	x	24	11000
g	7	111	p	16	10000	y	25	11001
h	8	1000	q	17	10001	z	26	11010

   Thus "hello" is 8,5,12,12,15 as a base-10 code, and 1000,101,1100,1100,1111 as base-2 code numbers.

   The "letter" represented by 0 is the *space character* which is needed to separate words. Thus "go up" is 7,15,0,21,16 as base 10 code numbers.

* What is "help me" in the base 10 code?

* What is "hey" in the base 2 code?

* You have to separate the code numbers somehow. If we add a leading zero to one-digit code numbers, so for example f=6 becomes 06 (base 10), then every letter will use a 2 digit base 10 code number. Write "hello" using this *zero padded* coding.

* We can likewise add leading zeros to give all the base 2 code numbers five bits. For example f=110 becomes 00110 as a base 2 code number. Write "yes" using this base-2 zero padded coding.

* Write a short secret word using 2-digit base 10 code numbers with zero padding, with no commas or spaces or other separating characters. Exchange with your neighbor and see if you can decode each other's words.

* Write a short secret word using 5-bit base 2 code numbers with zero padding, with no commas or spaces or other separating characters. Exchange with your neighbor and see if you can decode each other's words.

2. Base-2 codes for bigger alphabets, including numbers, capital letters, and punctuation symbols, use more bits per letter. One such code, *ASCII*, uses 7 bits and can have a maximum of 128 symbols. Note that

$$128 = 2^7 = \overbrace{2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2}^{7 \text{ times}}$$

* What is the formula for the number of possible symbols in a base-2 code with b-bit code numbers?

* What is the formula for the number of possible symbols in a base-10 code with d-digit code numbers?

** What is the formula for the number of possible symbols in a base-B code with k-digit code numbers?

2 Error detection

Suppose that there is noise and error and some of the digits (or bits) in a code number are received as a different number. By using bigger code numbers, such errors can be detected by the receiver.

A *parity bit* is an extra bit added to the end of a base-2 code number so that the number of 1 bits is even, which means the sum of the bits will be an even number.

Here is the table of base-2 code numbers padded to 5 bits with leading 0s, listing the extra parity bit needed to make the sum of the digits an even number:

letter	base 2	parity	letter	base 2	parity	letter	base 2	parity
	00000	0	i	01001	0	r	10010	0
a	00001	1	j	01010	0	s	10011	1
b	00010	1	k	01011	1	t	10100	0
c	00011	0	l	01100	0	u	10101	1
d	00100	1	m	01101	1	v	10110	1
e	00101	0	n	01110	1	w	10111	0
f	00110	0	o	01111	0	x	11000	0
g	00111	1	p	10000	1	y	11001	1
h	01000	1	q	10001	0	z	11010	1

A *checksum* is an extra code number which is computed from the other code numbers and sent along with the message. The receiver recomputes the checksum from the received message to increase confidence in the correctness of the message.

1. With the parity bits appended, each code number takes 6 bits. Thus b=000101 and z=110101.

   For this section, we will always use base-2 code words, padded to 6 bits with leading zeros and a parity bit at the end.

* Decode the base-2 message 010001 010010 000000 010001 011110.

* You have been sent the secret password as a base-2 message

$$100001\,010001\,011000\,011010\,110000,$$

but one of the bits has been flipped. Which letter is incorrect?

* Write a six-letter secret word in this 6-bit code, change exactly one bit to a wrong value, and exchange with your neighbor. See who can find the wrong letter first.

2. For this section, use the first table of base 10 code numbers, padded with one leading 0 if necessary to get 2 digits per code number.

   Suppose the checksum is computed by adding the code numbers and then keeping only the last two digits, equivalent to the remainder after division by 100, and written (mod 100).

   For example, the message "hello" or 08 05 12 12 15 has checksum

$$8 + 5 + 12 + 12 + 15 = 52 \equiv 52 \pmod{100},$$

   while "go up to the house" is 07 15 00 21 16 00 20 15 00 20 08 05 00 08 15 21 19 05, with a checksum of

$$7{+}15{+}0{+}21{+}16{+}0{+}20{+}15{+}0{+}20{+}8{+}5{+}0{+}8{+}15{+}21{+}19{+}5 = 195 \equiv 95 \pmod{100}.$$

* Will this checksum detect every single-digit error?

* Change two different digits in the code for "hello" so that the result has the same checksum 52.

* Change as many digits as you like in the code for "hello" so that the result is an English word that has the same checksum 52.

3. If a checksum can take N values, and two or more wrong digits are randomly chosen (not by any design), then the received and computed checksums will falsely agree with probability only $1/N$. Hence the two-digit checksum, where $N = 100$, will detect $1 - 1/100 = 99\%$ of random two-or-more-digit errors.

* Suppose we add the code numbers and keep the last 3 digits as a checksum (reduce mod 1000). What fraction of random two-or-more-digit errors will this detect?

* In this base 10 code, the space character has value 00, so adding extra spaces will not change the checksum. Give an example of adding or subtracting spaces to a phrase that changes its meaning.

* Rearranging the order of letters in a messaage will not change this checksum. Give an example of two words or phrases with the same letters in different order that will have the same checksum.

3 Error correction

By adding enough extra information, it is possible to detect where a limited number of errors occurred. The interesting problem is to do this efficiently, so that the extra information takes up as little space as possible.

1. Suppose that we wish to send only 0s and 1s but with greater confidence that they are received correctly. One way is the *repetition code*: send 000 for 0 and 111 for 1. A single bit error would turn 000 into 001, 010, or 100, and would turn 111 into 110, 101, or 011.

   The *Hamming distance* between two base-2 code numbers is the number of bits that differ between them. For example, the Hamming distance between 000 and 010 is 1, while the Hamming distance between 111 and 010 is 2.

   A *Hamming code* is a code containing a limited number of the possible base-2 code numbers. An error is detected if a received number is not a valid code number, and then it can be corrected to the nearest code number by Hamming distance. For example, in the repetition code the letter f=00110 becomes

   $$000\,000\,111\,111\,000,$$

   and is recoverable from the string below which has had an occasional bit flipped:

   $$001\,000\,111\,101\,000.$$

   Replacing any triple other than 000 or 111 with the nearer of (000,111) by Hamming distance fixes the flipped bits.

   (Note that this repetition code can correct multiple errors, but only if there is no more than one error per repeated bit.)

* Find the correct sequence of 0s and 1s given a received message 011 111 100 000 010 111, which is a single letter from the first table encoded with the repetition code.

* How many times must a bit be repeated if we wish to correct any combination of 2 errors?

* How many times must a bit be repeated if we wish to correct any combination of k errors?

* Base 2 codewords with 3 bits can be visualized as the corners of a unit cube. One corner is the origin 000, the farthest corner from it diagonally has coordinates 111. The nearest corners to 000 are along the coordinate axes and have coordinates 100, 010, and 001. Draw a picture to see that two corners of the cube are connected by an edge if and only if the Hamming distance between them is 1.

** Base 2 codewords with N bits can be visualized as the corners of an N-dimensional unit cube. One corner is the origin $00\ldots0$ (a string of N 0s), the farthest corner from it diagonally is $11\ldots1$ (a string of N 1s). The nearest corners are along the N coordinate axes and have coordinates $10\ldots0$, $010\ldots0$, and so on up to $0\ldots01$. Show that two corners of the N-cube are connected by an edge if and only if the Hamming distance between them is 1.

2. Extra data for error correction can be mixed with the message bits. One clever way to do this is *Hamming's 7,4 code* which uses 7 total bits to send 4 message bits (plus 3 parity bits) so that any single bit error (among the 7) can be detected and fixed.

   The code numbers base 2 consist of 7 bits, named

   $$p_1\ p_2\ d_1\ p_3\ d_2\ d_3\ d_4,$$

   where d_1, d_2, d_3, d_4 are data (or message) bits, while p_1, p_2, p_3 are parity bits added for error correction. These parity bits cover overlapping subsets of the data bits:

   - Bit p_1 is a parity bit for d_1, d_2, d_4, so it is 0 if $d_1 + d_2 + d_4$ is even and 1 if that sum is odd.
   - Bit p_2 is a parity bit for d_1, d_3, d_4, so it is 0 if $d_1 + d_3 + d_4$ is even and 1 if that sum is odd.
   - Bit p_3 is a parity bit for d_2, d_3, d_4, so it is 0 if $d_2 + d_3 + d_4$ is even and 1 if that sum is odd.

   For example, the data quadruplet $d_1d_2d_3d_4 = 0110$ will have parity bits $p_1 = 1$, $p_2 = 1$, and $p_3 = 0$. The resulting Hamming code number base 2 will be $p_1p_2d_1p_3d_2d_3d_4 = 1100110$.

* Find the Hamming 7,4 code number base 2 for the data 0000 and for the data 1111.

* Break up the bit string 0101 0110 1010 into three 4-bit pieces and find the three Hamming 7,4 code numbers base 2 for the pieces.

3. The location of a single bit error in a 7-bit Hamming code number is computed from the parity bits.

   If the received code number is $p_1p_2d_1p_3d_2d_3d_4$, then the computed parity bits p' are:

- $p'_1 = 0$ if $d_1 + d_2 + d_4$ is even and 1 if that sum is odd.
- $p'_2 = 0$ if $d_1 + d_3 + d_4$ is even and 1 if that sum is odd.
- $p'_3 = 0$ if $d_2 + d_3 + d_4$ is even and 1 if that sum is odd.

For example, the 7-bit code number $p_1p_2d_1p_3d_2d_3d_4 = 0100110$ gives $p'_1 = 1$, $p'_2 = 0$, and $p'_3 = 0$. Note that $p'_1 \neq p_1$ and $p'_2 \neq p_2$, although $p'_3 = p_3$. This indicates an error, and indeed there is a flipped bit at location 3=011 (base 2).

* Find p_1, p_2, and p_3 for 1011011.

* Compute p'_1, p'_2, and p'_3 for 1011011.

4. The error bit is at location $q_3q_2q_1$ base 2, where $q_1 = 1$ if $p'_1 \neq p_1$ with $q_1 = 0$ if $p'_1 = 1$, and so on. Notice that the error corrector gives the index of the bad bit in *reverse* base-2 notation.

   For example, the data 0110 which becomes the code number 1100110 has $p_1 = 1$, $p_2 = 1$, $p_3 = 0$. Suppose bit number 1 is flipped during transmission to give 0100110. Then $p'_1 = 0 \neq p_1$, $p'_2 = 1 = p_2$, and $p_3 = 0 = p_3$. Then $q_1 = 1$, $q_2 = 0$, and $q_3 = 0$, so the error corrector says fix the bit at position $q_3q_2q_1 = 001$ (base 2) = 1 (base 10).

* Compute q_1, q_2, and q_3 for 1011011. Which bit is flipped?

* Find the wrong bit in the Hamming 7,4 code string 1010011 and extract the corrected 4 bits of data.

5. Since q_1 is 0 if and only if $p_1 = p'_1$, and p_1 is chosen such that $p_1 + d_1 + d_2 + d_4$ is even, another way to compute q_1 (and similarly q_2 and q_3) is

   - $q_1 = 0$ if $p_1 + d_1 + d_2 + d_4$ is even and 1 if that sum is odd.
   - $q_2 = 0$ if $p_2 + d_1 + d_3 + d_4$ is even and 1 if that sum is odd.
   - $q_3 = 0$ if $p_3 + d_2 + d_3 + d_4$ is even and 1 if that sum is odd.

   Note that $q_3q_2q_1 = 000$ if there is no error. Otherwise, the error is at the location whose base 2 index is $q_3q_2q_1 > 0$.

* Use this method to determine which bit is incorrect in the Hamming code number 1110111

* The sequence 1000011 1000011 1010101 0010010 1001100 consists of five Hamming 7,4 code strings which encode four letters as 5-bit code numbers. One of the bits is wrong. Figure out which bit is wrong and recover the word.

4 Homework

There is no homework. Yay!

But you might want to read the Wikipedia entry for Hamming(7,4).