How can we build-in ways of finding and fixing transmission errors when using binary codewords?

Create redundancy & conditions for the forms of words so that you are more likely to realize an error has been made & can potentially deduce the real message/data

Definitions:

Codes composed of sequences of binary digits are called binary codes. A parity check digit is an extra digit appended to a message for error-detecting.

Example: Determine the parity check digit that should be appended to each block so that the total number of 1's is even.

(a) 01000111 - even # of 1's so add a 0 to keep an even number of 1's: 010001110

(b) 10111001 - add 1 - had an odd # of 1's: 101110011

(c) 10111010 - add 1: 101110101

Key Assumptions and Formula:

1. The probability of changing a 0 to a 1 and of changing a 1 to a 0 is the same.
2. The probability of an error in each digit is the same and is independent of whether there are errors in other digits (i.e., in probability terms, the transmission of any two digits are independent events).
3. The probability of an error in any digit is small, so the probability of the correct transmission of a block is greater than the probability of a single error in the block and the probability of a single error is greater than the probability of two or more errors.

Formula: $C(n,k)p^k(1-p)^{n-k}$, where p is the probability of error in a single digit, k is the number of errors, and n is the length of the message

Why should this formula make sense?

$C(n,k)$ - looks at places we may have an error

p^k - probability of error for each of the k spots with an error

$(1-p)^{n-k}$ - probability of no error for each of the $(n-k)$ spots without error

together we have the likelihood of each setup $(p^k(1-p)^{n-k})$ & we count the different positions this could happen in ((n,k) part)

Example: Suppose the probability of error in transmission of a single digit is .01 (1%). What is the probability of having exactly 1 error in a message of length 9?

.01 .99 .99 .99 .99 .99 .99 .99 .99 OR .99 .01 .99 .99 .99 .99 .99 .99 .99 etc.

$C(n,k)p^k(1-p)^{n-k}$, $n=9, k=1, p=.01$ so $C(9,1).01^1(1-.01)^{9-1} = 9(.01)(.99)^8 \approx 0.083$

so ~8.3% chance of exactly 1 error

Example: Suppose the probability of error in transmission of a single digit is .01. What is the probability of having exactly 0 errors in a message of length 9?

$n=9, k=0, p=.01$ so $C(9,0).01^0(1-.01)^{9-0} = 1(1)(.99)^9 \approx .914$

aka 91.4% of no error

Example: Suppose the probability of error in transmission of a single digit is .01. What is the probability of having exactly 2 errors in a message of length 9?

$n=9, k=2, p=.01$ so $C(9,2).01^2(1-.01)^{9-2} = 36(.01)^2(.99)^7 \approx .003$

$C(9,2) = \frac{9!}{2!(9-2)!} = \frac{9\cdot 8\cdot 7!}{2\cdot 7!} = 36$

aka .3% chance of 2 errors

Definitions:

Suppose we want to transmit information in blocks of k binary digits. Each block is a messageword and k is the length. We transmit codewords that include more digits to permit error-detection. If each message word has length k and each codeword has length n, the coding scheme is a (k,n)-blockcode. The efficiency of a (k,n)-block code is the ratio k/n.

Example: Suppose we want to transmit messages using ASCII, which uses 8 digit names for each symbol, and we will include a parity check digit for error detection. What are k and n and what is the efficiency of this (k,n)-block code?

$k=8$ (length of messageword)
$n=9$ (length of codeword)

efficiency: $8/9 \approx 88.9\%$

Example: Suppose we have 8 digit message words and we repeat the message for error detection (e.g., message 00000001 00100000 is sent as 00000001 00000001 00100000 00100000). What are k and n and what is the efficiency of this (k,n)-block code?

$k=8$
$n=16$

efficiency: $8/16 = .5 = 50\%$

Example: Suppose we have 8 digit message words and we repeat the message twice for error detection (e.g., message 00000001 00100000 is sent as 00000001 00000001 00000001 00100000 00100000 00100000). What are k and n and what is the efficiency of this (k,n)-block code?

$k=8$
$n=24$

efficiency: $8/24 = 1/3 \approx 33.3\%$ or .333

Definitions:

For this coding scheme, we need a one-to-one function that encodes (call this function E) message words as codewords and an inverse function that decodes (call it D). Thus for message $w_1, w_2, ..., w_m$, we transmit $E(w_1), E(w_2)..., E(w_m)$ and regain the original message by taking $D(E(w_1)) = w_1$, etc. If someone receives a word z that is not a codeword, they know an error happened. Usually the receiver would decode z as the codeword that differs from z by the fewest digits. This is called nearest neighbor decoding

Example: While less efficient, the third option (e.g., message 00000001 00100000 is sent as 00000001 00000001 00000001 00100000 00100000 00100000) permits some error-correction. How could we use the three copy version of the message to determine the likely intended message?

Since 0 errors is the most likely but 1 error is more likely than 2, use the other copies of the word to decide what was intended (e.g. if have 00000001 00000011 00000001, assume 00000001 was intended)

Definition:

For two codewords c_1 and c_2 of the same length, the Hamming distance between c_1 and c_2 is defined to be the number of digits in which c_1 and c_2 differ, denoted $d(c_1, c_2)$.

Example: What is the Hamming distance between the following codewords?

(a) $d(01000111, 01010101) = 2$ (differ in 2 positions)

(b) $d(10111001, 10111011) = 1$ (differ in 1 position)

(c) $d(00000000, 11111111) = 8$ (differ in all 8 positions)

Example: If you add two codewords over $\mathbb{Z}_2$ for each digit, what will happen?

$$\begin{array}{r} 01000111 \\ +\ 01010101 \\ \hline 00010010 \end{array}$$

(adding position-wise, not carrying) — get 1 in positions that differ & 0 in positions that match - alternate way to find Hamming dist

Triangle Inequality (Theorem 3.6):

If c_1, c_2, and c_3 are any codewords of the same length, then $d(c_1, c_3) \leq d(c_1, c_2) + d(c_2, c_3)$.

Why should this make sense?

For each position where c_1 & c_3 differ, either c_1 & c_2 differ or c_2 & c_3 differ so those differences (our measure of distance) for c_1 & c_3 is accounted for in the other differences (& they may have extra differences)

e.g. $c_1 = 0001$, $c_2 = 0011$, $c_3 = 0101$

$d(c_1, c_3) = 1$, $d(c_1, c_2) = 1$, $d(c_2, c_3) = 2$ $\quad 1 \leq 1+2$

Theorem 3.7:

Consider a block code in which m is the minimal Hamming distance between distinct codewords.

(a) This coding scheme can detect r or fewer errors if and only if $m \geq r + 1$.

(b) This coding scheme can correct r or fewer errors if and only if $m \geq 2r + 1$.

Why should this make sense?

To notice a problem, there need to be some impossible words d to fix a problem, we need enough space to know the more probable message (one we're closer to)

Example: Suppose the minimal Hamming distance between codewords in a certain block code is 4. What is the maximum number of errors that can be detected and what is the maximum number of errors that can be corrected?

$m=4$ $4 \geq r+1 \Rightarrow 3 \geq r$ - can detect at most 3 errors

$4 \geq 2r+1 \Rightarrow 3 \geq 2r \Rightarrow \frac{3}{2} \geq r$ - in context, r must be a nonneg. integer so at most 1 error can be corrected

Example: Suppose the minimal Hamming distance between codewords in a certain block code is 16. What is the maximum number of errors that can be detected and what is the maximum number of errors that can be corrected?

$m=16$ $16 \geq r+1 \Rightarrow 15 \geq r$ - 15 errors (at most) detectable

$16 \geq 2r+1 \Rightarrow 15 \geq 2r \Rightarrow \frac{15}{2} \geq r$ - in context, at most 7 errors correctable

Example: Suppose the following set comprises all the possible codewords. What is the minimal Hamming distance between codewords for the set?

(a) {01011, 00110, 00111, 11000, 10101}

two codewords differ in 1 position so minimal Hamming distance is 1

$m=1$ $1 \geq r+1 \Rightarrow 0 \geq r$ - can't guarantee we'll detect errors

(b) {101011, 001101, 100111, 110001, 000001}

dist 2, dist 2, dist 2

codewords that differ by 2 exist (none w/ 0 or 1 differences) so minimal Hamming distance is 2

$m=2$: $2 \geq r+1 \Rightarrow 1 \geq r$ - can guarantee detection of 1 position error (but can't correct)

Example: Suppose we used the parity check digit method as before (length 8 message words based in ASCII, length 9 codewords that must have an even number of 1s). What is the minimal Hamming distance in this context? What does this suggest about error-detection and error-correction? 2 - all options available for first 8 digits but the message words that differ in place (e.g. 00000000 vs 00000001) have different parity so their check digit is different, producing a 2nd difference

$m=2 \Rightarrow$ can guarantee detection of 1 position error ($2 \geq 2r+1 \Rightarrow 1 \geq 2r \Rightarrow \frac{1}{2} \geq r \Rightarrow$ 0 error correction)

Example: If a set of codewords contains the codeword in which each digit is 0, what can be said about the minimal Hamming distance between two codewords?

must be less than or equal to the minimum number of 1's in other codewords