ERROR-CORRECTING CODES - WEEK 1

Introduction

Error-detection and correction are needed for most data, particularly compressed or scrambled/encrypted data where a single error can make all nonsense. Examples: in computers on telecommunication channels etc.

Data in blocks give rise to **block codes**, of n symbols. If there are q symbols in the 'alphabet' then there are q^n possible values for a block. Typically q=2, binary. A subset of all these blocks forms the code – the valid set of, say, M 'messages' or **codewords**. The codewords are created from raw source data using some rule. For example, in linear binary codes, M=2^k (the codewords are k bits in length) and are mapped into the n-bit vectors, using a matrix (see below) giving an (n,k) code: n-bit vectors carrying k-bits of information. Appearance of any other vector, not in the code, shows it to be an error. Maybe it can be corrected to a valid codeword if it is 'near enough' to one.

The 'd**istance**' between two vectors is the number of corresponding positions (symbols or bits) in which they differ. The distance between 11010 and 10001 is 3, and is found by XORing them together and counting the resultant 1's in 01011. If the alphabet is 0,1,2 then the distance between 12102 and 21112 is 3 (subtract mod 3 and count the non-zero symbols). The **distance of a code** is the minimum distance between any pair of codewords. If errors are deemed to occur randomly and the distance = d, then any number of errors up to and including (d-1) are detectable. Any number of errors less than half d are potentially correctable, by finding the valid codeword nearest to the received erroneous vector. (Could there be more than one at the same nearest distance?) Typically we work with d=2t+1 so t errors are in principle correctable.

Example: Suppose there are four 8-bit codewords 00000000, 11111000, 00111111, 11000111; the distance of the code is 5, so t=2. The received vector 11000000 is an obvious error, which real codeword was corrupted? Suppose 00000000 is not a valid codeword, what effect does that have?

If the probability of a bit-error = p, then the probability of an erroneous vector being detectable is greater or equal to nCj.[(p^j).((1-p)^(n-j)] summed from j = 1 to (d-1), where nCj is a binomial coefficient = n!/(j!(n-j)!). The probability of an erroneous vector being potentially correctable is the same expression summed from j = 1 to t. Plot a graph of the probabilities of potentially undetectable or un-correctable errors for decreasing values of p and you get 'waterfall curves'. (See Annexe 1E)

Given the block length n, it is obvious that the distance of any code is controlled by the number of codewords. If there are lots of them there is little space between them. Conversely, if we want good error-correcting capabilities we need a large distance – and consequently fewer codewords. For binary linear codes we may say that a small **coderate k/n** should give good error-correcting capabilities, and a large one nearing unity will not. A small coderate means inefficiency; n bits used to carry only k. A high coderate means a smaller distance and poor error-correction. What is the relation between coderate k/n and distance d? Answer: It depends on the code, but some general bounds can be established.

Exercise: How many binary codewords of length n=5 can be found so that the distance =2? Or d=3? Or n=7 and d=3?

The Sphere-Packing Bound - Annexe 1A

Imagine each codeword surrounded by a 'sphere' of correctable corruptions of – all those vectors at a distance less than or equal to t from it. If there are M codewords, all these spheres, which must not overlap occupy a volume M.(1+S(n,t)) where S(n,t) = Sum [j=1 to t]nCj where nCj is a binomial coefficient, enumerating all vectors at distance j from the codeword. It can be shown (see Annexe 1F) that (1+S(n,t)) can be replaced by S(n,t)=< 2^(n.H(t/n)) where H() is the **entropy function**. So if the non-overlapping spheres are to fit in the total space available (2^n) then we require

M .2^(n.H(t/n)) =< 2^n

Assuming M = 2^k, and taking logs we get:

The coderate k/n =< (1 – H(t/n)). This is a bound on code-rate in terms of distance. [H(x) is a function rising rapidly from 0 with x, reaching a maximum of 1 when x=1/2 and then falling symmetrically to zero when x=1. H(x) = x(log 1/x) + (1-x)(log 1/(1-x)), the logarithms being to base 2.] This is the sphere-packing bound.

Capacity

Shannon's Theorem (Annexe 1B) on Capacity, of a channel, is not unrelated. If p is the bit-error probability then (for a Binary Symmetric Channel (BSC)) information theory shows that Capacity, C = 1 – H(p). Capacity is defined as the maximum mutual information between input to and output from a channel – for example the difference between the *a posteriori* information about the input (after observing the output) and the *a priori* information about the input before observing the output. (See Annexe 1C) . Shannon's theorem, attached, shows that as long as the code-rate is less than the capacity, k/n < C, it is always possible to find a code providing error-free communication. The proof assumes that decoding consists in drawing a sphere of radius slightly more than n.p around the received vector to be decoded. If a single codeword is found in this sphere decoding is to it. If several or no codewords are encountered decoding has failed. The number of such spheres that will fit in the space is limited by k/n. Such decoding can always be successful given a suitable code and a large enough n.

Linear Codes

An (n,k) linear code is a k-dimensional subspace in an n-dimensional vector space over GF(q). If q=2 it is a binary linear code. The all-zero vector (00....000) is obviously a member of the code.The sum of any two codewords is a codeword.

The **weight** of the code is defined as the minimum weight of all codewords, with the weight of a codeword being equal to the number of non-zero elements in it. Students to show the weight of a linear code equals its distance.

The code, as a subspace, has a **basis of k linearly independent n-vectors**. Without loss of generality these can be assumed to be in **systematic form**. (Students to show). That is: the basis vectors are the k

row vectors of the **(k x n) matrix G = (I,P)** where I is the (k x k) identity matrix and P is a (k x n-k) parity check matrix. **Any k-vector m can be turned into a codeword c by forming c = m.G.** The result is that the first k elements of the codeword are simply the vector m repeated and the remaining (n-k) are check elements used for error detection and correction.

The code as a subspace has a **null-space** of all the vectors orthogonal to the code. (Beware. Over a finite field vectors can be orthogonal to themselves!) A basis for the nullspace of the code defined by **G** is **H = (-P*,I**) where P* is the (n-k x k) transpose of P and I is (n-k x n-k). **H is the Null-Matrix. G.H* = 0, students to check. Any codeword must satisfy c.H* = 0.**

The code can be presented as a **Standard Array**. The top row is a list of all the codewords. The next row is a coset formed by adding an error-vector to the code, starting with the coset-leader added to the codeword (000...00). The following row is another coset with a new leader, selected from vectors not already written down. Et cetera. The process ends when all vectors feature in the standard array (q^(n-k) by q^k). Attached (Annexe 1D) is the standard array for a (7 x 4) code – which happens to be a perfect Hamming code.

A received vector r can be checked to see if it is a codeword, by using **r.H* = 0? If r is corrupted by an error-vector e so that r = c+ e, then r.H* = e.H* = s, where s is called the syndrome.** If the weight of the error vector is less than or equal to t (when the distance d = 2t + 1), then the error is a correctable error**. There is a one-to-one correspondence between correctable errors and their syndromes.** (Students to show.)

The **weight distribution**, listing the number of codewords of each possible weight, also lists the distance distribution. In the (7,4) example it is symmetric.

Since for a codeword c.H* = 0 the distance of a code in the minimum number of linearly dependent columns of H.

A code, such as the (7,4) code whose distance is odd (=3) can be turned into an even distance code by adding a parity bit to all codewords. Students to show how this produces an (8,4) code with distance =4. What are G and H for this new code? What is the standard array?

Non-binary linear codes are readily constructed. For example, over GF(3) suppose the three rows of the Null Matrix H are (1111111100100), (2221110011010) and (2102102121001) what are n and k? What is G? What is the distance? What is the weight distribution?

Another example: Over GF(4) a linear code has as generator matrix G = [1 w^2 w] where w is a root of x^2 + x + 1 so that GF(4) has elements (0, 1, w, w^2 = 1+w). What are n and k? What is the null-matrix H? What is the distance of the code? List the valid codewords. If we represent the four elements of GF(4) by 0= 00, 1= 01, w=10, w^2=11, what is the distance of the code as viewed as a binary code?