Math 110 HW 4 solutions

June 9, 2013

1 Preliminary vocab and facts

First, just to say what lingo and vocabulary I'm using, or remind you in case this is not standard. I may use "xor" to refer to adding words in a binary code in the way usually defined, where bits are added separately; as opposed to thinking of those words as numbers written in binary and adding the numbers themselves.

The word "parity", as in "the parity of a number", refers to the number being odd or even. The computer scientists' lingo "flip a bit" means, in a string of 0's and 1's, i.e. a word in a binary code, change one letter (bit) from 0 to 1 or from 1 to 0. This can also be thought of as an xor operation; for example to flip the first bit, xor with the word 1000..0, or to flip the second, xor with 0100...0.

I define distance between two words w_1, w_2 of the same length as the number of bits where they are different. But also, each such bit shows up as a 1 in $w_1 + w_2$, which has a 0 wherever the two bits are the same, so you could also say it's the number of 1s in the word $w_1 + w_2$.

2 Problems

1. Show that if there exists a code with word length N, M words and minimum distance d with d even, then there exists a code with word length N, M words, and minimum distance d such that every word has an even number of 1's.

   Let us call the original code C. First of all, if $d = 0$ the problem is not true as stated. Indeed in that case C could consist of every word of length N, then any code made of words with an even number of 1's would have to have at most half as many words as C does. So from now on assume $d > 0$, which for this problem means $d \geq 2$.

   Notice that the distance between any two words is odd if one word has an even number of 1's and the other has an odd number of 1's. If both words have an even number of 1's, the distance between them is even; if they have an odd number of 1's, the distance is even. This is because each time you flip one bit the parity of the number of 1's changes, and because the number of wasted bitflips, i.e. the difference between the number of bitflips and the distance moved, is always even.

The procedure relies on the following lemma:
Lemma 1. *If binary words w_1 and w_2 are length N and distance k apart, then given any word z, the distance between $w_1 + z$ and $w_2 + z$ is k.*

Proof. A proof is clumsier to write down than simply to think about. Nevertheless: for each bit where w_1 and w_2 are the same, that contributes 0 to measuring their distance. Then $w_1 + z$ and $w_2 + z$ are still the same in that bit (regardless of whether the corresponding bit in z is 0 or 1), which contributes 0 to the distance between the new words. For a bit where w_1 and w_2 differ, contributing 1 to the distance between the words, again, after xoring with z that bit in the new two words is also different, contributing 1 to the distance between the words. When each bit is accounted for in this way, the lemma holds. $\square$

In code C, let S be the set of all words with an even number of 1's, and let T be its complement. Let z be the word of length N where the first bit is 1 and the rest are 0. For every word in C that is not in S, xor it with z; call the new set of words $T + z$. Now, the distances between words in $T + z$ is at least d, because that is true of T, and because of the lemma above. Also, every word in $T + z$ has an even number of 1's.

The distance between a word in T and a word in S must be an odd number that is at least d. Since d is even, that means it must be at least $d + 1$. So the distance between any word in $T + z$ and S must be at least d. Notice that since $d > 0$, this means no word in S is the same as a word in $T + z$; otherwise the code would be smaller than we intended, as discussed already.

Since the distance between pairs of words within S, within $T + z$, and between them, is always at least z, we can combine the two sets $S, T + z$ to create a code C as required.

I did not need to say so explicitly, but it's good to note that this proof works even if S is the empty set, or if S is all of C.

2. What is the largest possible number of words in a code with length 6 and minimum distance 3?

The first bound we can find is a covering bound. If two words w, x in the code are at least 3 apart, then the set of {words at most 1 away from w} and the set of {words at most 1 away from x} do not overlap. For any word, the set of words at most 1 away from it includes the word itself, and six words each obtained by flipping exactly 1 bit.

So each word in the code produces a new, disjoint set of 7 points. Since there are 64 words of length 6 altogether, there are at most $\lfloor 64/7 \rfloor = 9$ words in the code.

Assume there are at least 9 words in the code. Then either at least 5 words start with 0, or at least 5 words start with 1. Cutting off the first bit we get a code of length 5, and minimum distance 3, and at least 5 words in it. Then at least 3 words start with the same bit, so similarly we get a code of length 4, minimum distance 3, and at least 3 words in it.

Let's check this by hand: without loss of generality assume 0000 is in this code, and then any other word must have at least three 1s. There are five such words: 0111, 1011, 1101, 1110, and 1111. Unfortunately no two of them are distance 3 apart, so we cannot get the desired code.

Thus the bound of 9 cannot be attained. On the other hand, if we look at an example of a Hamming code of length 7, distance 3, with 16 words in it, there are 8 that share the first bit 0; cutting that bit off we get a code of length 6, distance 3, with 8 words in it. In such a way we obtain a linear code generated by

$$\begin{matrix} 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{matrix}$$

which gives us a total list of

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{matrix}$$

Because the code is linear, to see that it is minimum distance 3, as in a previous problem it is enough to note that every nonzero element has at least three 1s. Thus the answer to the question posed in the problem is 8, as attained by this code.

3. Let C be a binary code of length n and minimum distance d. Define a new code C' of length $n+1$ by
$$C' = \{x_1x_2\dots x_{n+1} | x_1x_2\dots x_n \in C, x_1\dots x_{n+1} \equiv 0 \pmod 2\}.$$
Suppose that d is odd. Prove that the minimum distance of C' is $d+1$.

   We can read this as saying that each word in C' is formed by taking a word in c for the first n bits, and the last bit is the sum of the first n bits.

   Suppose d is odd as given in the problem. If two numbers in C differ by exactly d, then if the bits of one word sum to 1, the bits of the other sum to 0, and vice versa. Thus, in C' the words will have $n+1$st bit different, and the distance between them is $d+1$. Otherwise, if two words differ in C by $d+1$ or more, then they already differ in C' by $d+1$ or more. Thus the minimum distance of C' is $d+1$ as was to be shown.

4. Let C be the binary linear code with generator matrix
$$G = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

   (a) Give a complete list of all the codewords of C.

   Recall that G being a generator matrix for C means that the rows of G form a basis for the words in C. We can keep track by writing the codewords in order. Since the first three columns form the identity matrix, the first three bits of any element in the linear code will represent the linear combination of rows taken. In particular listing them from least to greatest, the first three bits will be $001, 010, 011, 100, 101, 110, 111$. Then writing this out we get eight elements of the code:

$$\begin{array}{ccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & 0
\end{array}$$

(b) What is the minimum distance of C? How many errors can it correct?

The minimum distance of C is 4, as the norm of each nonzero word (the distance between that word and 0) is 4, so by linearity the difference between any two code elements is a word of norm 4 (or 0 if the two elements are the same). Note that it is not enough to say the three generators each had norm 4; what makes this work is that linear combinations of them also had norm 4, thus they were originally 4 apart.

The string 0000101 is two away from 1100101 and 0011101 and 0000000 in the code, so it cannot be corrected. But since the minimum distance is 4, any word at most 1 away from a code element can be unambiguously corrected: thus, it can correct 1 error.

(c) Decode the words $1100111, 0010110$. The first is closest to 1100101 and the second to 0110110. Thus we decode to those.

5. Consider the binary code with four codewords $\{(0,0,1),(1,1,1),(1,0,0),(0,1,0)\}$. Show that this code is not linear and compute its minimum distance.

If we add $(0,0,1)$ and $(1,1,1)$ we get $(1,1,0)$ which is not in the code; thus the code is not linear. Its minimum distance is 2 by explicit checking of all six pairs of words.

6. Suppose $k < n + 1 - \log_2(\sum_{i=0}^{d-1} \binom{n}{i})$. Prove that there exists a linear $[n, k, d]$ code.

To unpack the problem, we want to show there is a linear binary code where the words have length n, the number of generators is k, and the minimum distance is d.

Fix d and n. Let l be the greatest integer for which there exists a linear $[n, l, d]$ code. Let C be such a code. Suppose $l + 1 < n + 1 - \log_2(\sum_{i=0}^{d-1} \binom{n}{i})$ (otherwise there is nothing to prove). We will construct a $[n, l+1, d]$ code.

The inequality is equivalent to $2^l(\sum_{i=0}^{d-1} \binom{n}{i}) < 2^n$. The sum $\sum_{i=0}^{d-1} \binom{n}{i}$ is equal to the number of words of distance less than d from a fixed codeword. Thus the quantity $2^l(\sum_{i=0}^{d-1} \binom{n}{i})$ is an upper bound for the number of words of distance less than d from a word in C. Therefore there exists a word w for which the Hamming distance satisfies $d(w, c) \geq d$ for all $c \in C$.

Let $C' = C \cup \{w + c \mid c \in C\}$. We will show that C' is a linear $[n, l+1, d]$ code.

Suppose $x, y \in C$. If $x \in C$ and $y \in C$ then $d(x, y) \geq d$ as C has minimum distance d. If $x \in C$ and $y \notin C$ then $y = w + z$ for some $z \in C$ and $d(x, y) = d(x, w + z) = d(x - z, w) \geq d$

as $x - z \in C$ and using the construction of w. If $x \notin C$ and $y \notin C$ then $x = w + x'$ and $y = w + y'$ for some $x', y' \in C$ and $d(x, y) = d(x', y') \geq d$. Therefore the minimum distance of the code C' is d.

To check the other required properties of the code C' is routine. For example to show C' is linear, break into cases as in the previous paragraph and use linearity of C and the fact that $w + w = 0$.

7. Let k be an integer. There are $2^k - 1$ prisoners. A black or white hat is to be placed on the head of each prisoner. Each prisoner will be able to see the color of the hats worn by every other prisoner, but not the color of his or her own hat. From this point in time, no communication between the prisoners is possible. Each prisoner is then simultaneously asked: "What color is your hat?" There are three allowed responses, white, black and no response. If at least one person guesses their hat color correctly and no-one guesses the wrong color, the prisoners are set free. Otherwise they are all exectuted.

   The prisoners may meet beforehand to decide on their strategy. What is their best course of action? Assume that their common goal is to be freed.

   Solution: This is a famous problem involving Hamming codes. I sent out a class email on how to read up about this problem.

   In the meeting beforehand, the prisoners assign themselves an order. They choose a binary code C of words of length $2^k - 1$ that can correct 1 error, and that has as few elements as possible. There are linear codes meeting this constraints called Hamming codes, each of which has 2^d elements and minimum distance d.

   Now each prisoner knows $2^k - 2$ bits of information. If the bits could correspond to an element of the code, the prisoner guesses the hat color that would make the string *not* an element of the code. Otherwise, the prisoner does not guess.

   In this way, since the code can correct 1 error, the prisoner corresponding to the incorrect bit makes the guess that is not the code element. So for each of the 2^d words in the code, there are d words 1 away from that word. If the arrangement is the codeword, the prisoners all guess incorrectly to make it one of these d words; if it is one of these d words, then the prisoner corresponding to the affected bit guesses correctly, and everyone else passes. Thus there is a $d/(d + 1)$ chance of survival.

   For example, the case of 3, the code is chosen as $\{000, 111\}$. If the first prisoner perceives $x00$, they guess 1; if $x11$, they guess 0; otherwise they pass. It is similar for other prisoners and can be summed up as saying "if a prisoner sees 2 hats of the same color, they guess their own hat to be the other color; if they perceive 2 hats of different colors, they make no guess". In that way, only the cases 000 and 111 lead to execution, where everyone guesses incorrectly; in the other 6 cases, one person guesses correctly and the other two pass. Thus they have a 3/4 chance of survival.