Class 1 Exercises

CS250/EE387, Winter 2025

1. Let $\mathcal{C} \subset \{0,1\}^n$ be a code over the alphabet $\{0,1\}$ with block length n. What distance does $\mathcal{C}$ need to have to correct up to two errors?

   Come up with a code $\mathcal{C} \subset \{0,1\}^n$ that can correct two errors and that has $n \leq 10$. There is a straightforward construction of such a $n = 10$ and with message length $k = 2$. Once you find that construction, can you do better? (For example, can you find a code that can correct up to two errors, with $k = 2$ and $n < 10$?)

   Bonus. What's the best you can do? Can you prove that it's the best?

   Extra Bonus. What's the best you can do if $k = 3$? $k = 4$? Does your solution generalize?

   Solution

   To correct up to two errors, we should have distance $d = 5$. The straightforward construction is to repeat each bit 5 times. So $\mathcal{C} = \{0000000000, 0000011111, 1111100000, 1111111111\}$. But there are better constructions. For example, this one has $n = 8$: $\{00000000, 00011111, 11100011, 11111100\}$.

   For the bonus, you can't do better than $n = 8$. Suppose towards a contradiction that we could get away with $n = 7$. WLOG, the first two of the four codewords are

   $$c = 0000000, c' = 1111100$$

   (assuming that there are two codewords of distance exactly 5). Now consider the third codeword c''. It must have weight at least 5, so that $\Delta(c, c'') \geq 5$. Thus, at least 3 of the first 5 positions should be 1. On the other hand, we also need $\Delta(c', c'') \geq 5$, aka, that c' and c'' agree on at most 2 positions. This implies that at most 2 of the first 5 posiitions should be be 1, or else c' and c'' would agree too much. But this is a contradiction.

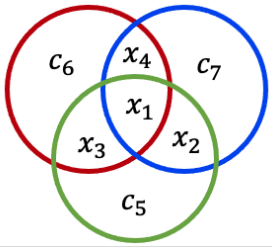
   For the extra bonus, this is actually an open question in general!!

   [Break for a bit of lecture before moving on]

2. In the lecture, we saw a binary code that had message length $k = 4$, codeword length $n = 7$, and distance $d = 3$. The encoding map was:

   $$(x_1, x_2, x_3, x_4) \mapsto (x_1, x_2, x_3, x_4, x_1 + x_2 + x_3, x_1 + x_3 + x_4, x_1 + x_2 + x_4),$$

   (all mod 2), and we had this picture of circles:



(a) We asserted that this code has distance 3. Convince yourself of this. (You don't need to give a formal proof, just stare at it until you are convinced and/or can convince each other).
Hint. It suffices to show that there are no codewords with fewer than 3 ones. (Do you see why?)

Solution

I'm convinced. (See the bonus below for more).

(b) It turns out that this is optimal – for example, there is no binary code with $k = 5, d = 3$ and $n = 7$. Prove this!
Hint. Suppose that there were such a code. Consider the *Hamming balls* of radius 1 given by

$$B(c, 1) = \{x \in \{0, 1\}^7 : \Delta(x, c) \leq 1\}$$

for each $c \in C$. Do any of these Hamming balls overlap? How many points do they cover in total?

Solution

Following the hint, these Hamming balls are disjoint. Each of them contains 8 points (the center and the $n = 7$ things that are one bit-flip away). Altogether, that's $2^5 \cdot 8 = 256$ distinct points in $\{0, 1\}^7$ covered by these Hamming balls. But there are only $2^7 = 128$ points in $\{0, 1\}^7$. Contradiction.

(c) **(Bonus).** Generalize your logic on the previous problem to give an upper bound on k, in terms of n and d, if a code $C \subset \{0, 1\}^n$ with message length k and distance d exists.

Solution

This is the Hamming bound. See the lecture notes.

3. **(Bonus.)** How would you show formally that the distance of the Hamming code in the previous problem has distance 3. (There are several ways – try to find the most general way you can! What abstractions might be useful?)

Solution

There are several ways to see that this code has distance 3. One of them is by looking at the *parity-check* matrix, as discussed in Class 2. However, here are some other ways of doing it:

- We can view this code as three overlapping circles, like we did in lecture. Because of the Hint from 2(a), we just need to show that if at least one of message bits is nonzero, then there are actually three nonzero codeword bits. We can break this into three cases:
  - If exactly one message bit is nonzero, then the two circles that touch it are nonzero. So at least three codeword bits are nonzero.
  - If exactly two message bits are nonzero, then at least one circle hits only one of those bits. So again at least three codeword bits are nonzero.
  - If three or more message bits are nonzero, we are done.
- We can view the encoding map as a matrix-vector multiplication, with the matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

That is, the encoding of the vector $x \in \{0,1\}^4$ is given by Gx, mod 2. Again, we need to show that any non-trivial combination of the columns of G has weight at least 3. Let's separate G into the top 4 rows (the identity-matrix-part) and the bottom 3 rows. Again, we have the same three cases:

- If we include exactly one column, it should have weight at least 3. By inspection, all the columns have weight at least 3. Hooray!
- If we include two columns, then their sum should have weight at least 3. Since the identity-matrix part of their sum will have weight 2, that means that the sum of any two columns in the bottom part of the matrix (that is, the vectors 011, 101, 110, 111) should have weight at least 1. So it's enough for those four vectors to all be distinct. And indeed they are!
- If we include 3 or more columns, then we have weight 3 from just the identity-matrix part.

4. **(More bonus).** The code in the previous problem suggests a general recipe for creating codes (with $k = 4$ and $n = 7$):

$$(x_1, x_2, x_3, x_4) \mapsto (x_1, x_2, x_3, x_4, f_5(\vec{x}), f_6(\vec{x}), f_7(\vec{x})),$$

where f_5, f_6, f_7 are some linear functions mod 2. (That is, $f_i(x_1, x_2, x_3, x_4)$ is the sum of some of the message bits, mod 2.)

What properties should f_5, f_6, f_7 have in order to make sure that the code we get has distance 3? How many possibilities are there?

Solution

We can use the intuition from the previous exercise. Let's use the linear-algebraic way of looking at it. So we have some matrix, with a top part that's the identity, and a bottom part that corresponds to the functions f_i. Consider the 3×4 matrix that is the bottom part. Each row corresponds to one of the f_i's. Let's call the columns v_1, v_2, v_3, v_4. We saw from the previous exercise that:

- Each of the v_i's needs to have weight at least 2.
- Each of the v_i's needs to be distinct.

How many ways are there to come up with distinct $v_1, v_2, v_3, v_4 \in \{0,1\}^3$ that have weight at least 2? There are only four such vectors $(110, 101, 011, 111)$ so if they all have to be distinct, we have to have these four vectors...but we could put them in any order. Each order gives a different way of creating the parities f_i. So there are $4! = 24$ ways to pick these parities.

If you want to use the circles way to visualize it, imagine that we draw four dots, one for each message symbol, and we get to draw three "circles" (or weird blobby shapes) capturing different subsets of the dots. The "circles" correpond to the parity bits. Again looking at our cases from before, we need it to be the case that:

- Each dot needs to belong to at least two "circles"
- Each pair of dots needs to have at least one "circle" that contains only one of them. (That is, we can't have a pair of dots that's either both contained or both not contained in every circle).