Latin Squares Instructor: Padraic Bartlett

Lecture 8: Error-Correcting Codes and Latin Squares, Part 2/2

Week 3 *Mathcamp 2012*

1 A Quick Recap of Last Lecture

At the end of our last lecture, we defined the mathematical concepts of a **code**, as well as the related concepts of **information rates** and **distance**. We review these definitions here:

Definition. A q-ary code C of length n is a collection C of words of length n, written in base q. In other words, C is just a subset of $(\mathbb{Z}/q\mathbb{Z})^n$.

Definition. Given a q-ary **code** C of length n, we define its **information rate** as the quantity

$$\frac{\log_q(\#\text{ of elements in C})}{n}$$

This, roughly speaking, captures the idea of how "efficient" a code is.

Definition. The **Hamming distance** $d_H(\mathbf{x},\mathbf{y})$ between any two elements $\mathbf{x},\mathbf{y}$ of $(\mathbb{Z}/q\mathbb{Z})^n$ is simply the number of places where these two elements disagree. Given a code C, we say that the minimum distance of C, $d(C)$, is the smallest possible value of $d_H(\mathbf{x},\mathbf{y})$ taken over all distinct $\mathbf{x},\mathbf{y}$ within the code.

Theorem. A code C can correct up to t errors in any received codeword to the correct codeword, as long as $d(C) \geq 2t+1$.

Given these definitions, the question we closed our lecture with yesterday was the following:

Problem. Suppose that you are given a base q, a length n for your codewords, and a minimum distance d that you want your codewords to be from each other (because you want to be able to correct up to $\lceil (d-1)/2 \rceil$ many errors in any codeword, for example.)

What is the maximum size of C — in other words, what is the maximum information rate you can get a code to have with these parameters?

For many values of q, n, d this problem is completely open: for example, we have no idea what the largest code with $n = 10, q = 2, d = 3$ is, for example! This is surprising: we'd really expect that we'd know the largest/most optimal codes for a problem as simple as "find a binary code with words of length 10, such that we can handle up to one error in every word."

We mention this in a Latin squares class because it turns out that (while this problem is open for most values of our parameters) we can use Latin squares to completely resolve this question for a ton of parameters!

We start with a warm-up problem:

Problem. Take any base q. What is the largest code with all codewords of length 4, such that the minimum distance between any two codewords is 3? (In other words, what's the largest q-ary code set of length 4 that can correct ≤ 1 error in any block of 4?)

Without too much effort, we can get a pretty nice upper bound for our possible codes:

Proposition. If C is a q-ary code of length 4 and minimum distance 3, $|C| \leq q^2$.

Proof. Take any two elements $\mathbf{x} = (x_1, x_2, x_3, x_4)$ and $\mathbf{y} = (y_1, y_2, y_3, y_4)$ from our code C. Compare (x_1, x_2) and (y_1, y_2). If this pair agreed with each other, these two codewords would be distance 2 from each other, which is impossible because $d(C)$, the minimum distance in our code, is 3. So they must differ.

So: there are q^2 many such pairs to start off any of our codewords, and we've just shown that we cannot repeat any of these pairs. Therefore, there can be at most q^2-many elements in C.

Surprisingly enough, it turns out that we can use our knowledge of Latin squares to attain this bound!

Proposition 1 *There is a q-ary code of length* 4*, distance* 3*, and containing q^2 many elements, whenever there are a pair of mutually orthogonal Latin squares of order q.*

Proof. Before we begin, we illustrate the proof idea with an example: take two MOLS of order 3.

$$A = \begin{array}{|c|c|c|}\hline 0 & 1 & 2 \\ \hline 1 & 2 & 0 \\ \hline 2 & 0 & 1 \\ \hline \end{array}, B = \begin{array}{|c|c|c|}\hline 0 & 1 & 2 \\ \hline 2 & 0 & 1 \\ \hline 1 & 2 & 0 \\ \hline \end{array}$$

Consider the following construction:

location	s_A	s_B		codewords
$(0,0)$	0	0		0000
$(0,1)$	1	1		0111
$(0,2)$	2	2		0222
$(1,0)$	1	2	$\longrightarrow$	1012
$(1,1)$	2	0		1120
$(1,2)$	0	1		1201
$(2,0)$	2	1		2021
$(2,1)$	0	2		2102
$(2,2)$	1	0		2210

Create a list of all of the 3^2 possible cell locations in a 3×3 grid. For each location, write down the symbol in square A and the symbol in square B: this creates the table at left. If you compress these columns together, you get the list of codewords at the right.

Notice that this table of of codewords has the following property: if you are given any two of the four digits of a codeword, you can uniquely determine which codeword you started with. To see this, simply notice that knowing the row along with either the column, s_A,, or s_B uniquely determines what cell we're talking about (because of the Latin property of both A and B,) and therefore uniquely determines the rest of the values. Similarly, knowing the column and any other position also uniquely determines the cell we're studying, and therefore the codeword. Finally, if you know s_A and s_B, you know all of the other values because this is a pair of mutually orthogonal Latin squares, and therefore (because each pair of symbols shows up exactly once) there is a unique cell corresponding to this pair of symbols.

Therefore, because knowing any two symbols uniquely determines a word, no two words have two locations in common; therefore, the distance between any two words is at least 3. Therefore, $d(C)$ for this code is 3. It is made of base-3 words of length 4, and contains 9 words; therefore, it is exactly what we're looking for in the case that $q = 3$.

In general, we can do the exact same thing: for any q, if we have two mutually orthogonal Latin squares A, B of order q, simply list out all of the q^2 possible cells, along with the symbols in each in A and B. This will be a set of q^2 codewords, none of which overlap at more than one place, by the exact same logic as above; therefore, this is a code with minimum distance 3, and is again exactly what we're looking for.

Furthermore, note that we can easily undo this process: given a code C of length 4, distance 3, with q^2 words, note that if we look at the first two symbols in the words of our code, we get every possible pair of symbols. This is because they must all be distinct (because our code has distance 3, and therefore no pair of symbols overlap,) and there are q^2 many words in total. So, write down these codes in a table. Treat the first two symbols as the cell location, the third as the symbol in that cell in a $q \times q$ Latin square A, and the fourth as the symbol in that cell in a Latin square B. These two squares are both Latin and orthogonal, because no two words overlap in more than one place (and failing to be either Latin or orthogonal would force two words to overlap in two places.)

This settles our question for all values of q where we have a pair of mutually orthogonal Latin squares of order q: in particular, as we discussed earlier, this works for all values of $q \neq 2, 6$. On the HW, you proved that the maximum number of elements in a length 4, distance 3, 2-ary code was 2; so you've resolved that case.

This leaves $q = 6$. As you may have done in past HW sets, you can find a pair of order 6 Latin squares A, B such that when you superimpose A on top of B, you get 34 of the possible 36 pairs. If you turn these two Latin squares into length-4 codewords and just delete the two repeated pair codewords, you are left with 34 different codewords none of which overlap at more than one place. In other words, you've shown that there is a code of size 34, and we know that one does not exist of size 36 (because that would mean that two MOLS of order 6 existed, by our earlier proof.)

The only case remaining is to decide whether it's possible for a 6-ary code of length 4, distance 3 to exist with 35 elements. As you have either shown on the HW or can show now, it's not possible! In particular, having this would be equivalent to having a pair of Latin squares that gave you 35 of 36 possible pairs when superimposed, which you can show is impossible. (Do so!)

So: our knowledge of MOLS gave us complete knowledge of how the length-4 distance-3 codes worked! We can easily generalize the arguments we used above to length-n distance-$n-1$ codes, as we describe here:

Proposition. Let C be a q-ary length n distance d code. Then C contains less than q^{n-d+1} many elements.

Proof. Just like before, take any such code C and look at the first $n-d+1$ elements of every word. If any of these are repeated, then the distance between the corresponding words is at most $d-1$, a contradiction. Therefore none of these are repeated, and therefore there are at most as many codewords as there are such $n-d+1$ strings: i.e. $|C| \leq q^{n-d+1}$.

If $d = n-1$, this is still the claim that any such code contains q^2 many elements at best.

Theorem 2 *There is a q-ary length n, distance $n-1$ code with q^2 many elements if and only if there are $n-2$ MOLS of order q.*

Proof. Just like before, create a table listing these Latin square elements alongside their cell locations

location	s_{L_1}	s_{L_2}	$\dots$	$s_{L_{n-1}}$
$(0,0)$	$L_1(0,0)$	$L_2(0,0)$	$\dots$	$L_{n-2}(0,0)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Turn this table into codewords of length n by just compressing each row into a single word. By the exact same logic as before, knowing any two pieces of information here uniquely determines what cell you're talking about, because of orthogonality and our Latin properties. Therefore, we know that none of these words overlap at more than one place; i.e. that our code has distance C. It contains q^2 cells by construction, and is therefore the example we're looking for.

In particular, we've just shown that a q-ary code of length $q+1$, distance q code with q^2 elements exists if and only if there are $q-1$ MOLS of order q. This is the question for MOLS we spent a lot of time examining, and found that the answer was yes whenever q was a prime power, but did not know the answer to otherwise!

This is a good place to end this class; for our last example, we've managed to connect Latin squares simultaneously with fundamental open problems in their own field (what's the largest set of MOLS for any given order?), another field in mathematics (what's the largest q-ary code of length $q+1$, distance q?) and things that are practically used in reality (this method of error-correction via MOLS is literally how your broadband internet works: you have q-ary codes sent across your cables, and they get corrected for static interference via MOLS. This is actually how your internet works.) Pretty cool, right?