Notes 8.370/18.435 Fall 2021

Lecture 29 Prof. Peter Shor

Today, we're talking about the seven-qubit Hamming code. This Hamming code is probably the simplest CSS code; it's certainly the shortest CSS code which corrects one error. In lecture, I tried to present both the Hamming code and CSS codes at the same time, but thinking about my presentation, I think it would work better to do the quantum Hamming code first and the more general CSS code construction later, so I'm doing that in these notes.

First, we need to explain the classical Hamming code. The codewords of the Hamming code are the binary linear combinations of the generator matrix G:

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

There are 2^4 of these combinations, since there are 4 rows.

We consider a code C to be the set of codewords of the code. There are many different generator matrices that give essentially the same code.

The Hamming code encodes four bits into seven bits. To encode, we take a four-bit message m and multiply G by it to get the corresponding codeword:

$$c = mG\,.$$

We've been working with quantum mechanics, where you use left multiplication, so it may seem strange to be using right multiplication (the vector is on the left and the matrix is on the right). However, this is the standard convention in coding theory, and if I try to reverse it I suspect I would get very confused.

The Hamming code is just the simplest non-trivial example of a large class of classical error-correcting codes, called *linear codes*. In these codes, the codewords are binary linear combinations of some generator matrix G. In this lecture, we will talk about classical linear codes, and then discuss the 7-qubit quantum Hamming code, which encodes one qubit into seven qubits and corrects one error. In the next lecture, we will talk about quantum CSS codes, which are the simplest generalization of classical linear codes to quantum codes. A broader generalization, *stabilizer codes*, exists, but we won't discuss them in these notes.

How do we correct errors for the Hamming code? We use the *parity check* matrix H, which is a generator matrix for the dual space of the code. If you have a vector space V, then the dual space is

$$V^\perp = \{w | w \cdot v = 0 \; \forall v \in V\}.$$

If you've used to working with vector spaces over $\mathbb{R}$ or $\mathbb{C}$, one thing that is confusing about binary vector spaces is that the dual can overlap with the original, so some vectors can be in both V and $V^\perp$. However, even though this is true, it is still the case that $\dim V + \dim V^\perp = n$, as happens in vector spaces over $\mathbb{R}$.

Why can the Hamming correct one error? It's because the minimum non-zero codeword has Hamming weight 3. Here, the *Hamming weight* of a codeword is the number of non-zero coordinates of a codeword (for binary codewords, the number of 1's). The *Hamming distance* $d_H(c_1, c_2)$ between two codewords c_1 and c_2 is the Hamming weight of $c_1 - c_2$, in other words, the number of coordinates where c_1 and c_2 differ.

We now prove

Theorem 1. *Suppose that the minimum non-zero weight of a codeword in code C is d. Then the code C can correct $t = \lfloor \frac{d-1}{2} \rfloor$ errors and detect $d-1$ errors.*

Proof: We claim that the a word w cannot be within Hamming distance t of two different codewords c_1 and c_2. Suppose it was, then

$$d_H(c_1, c_2) \leq d_H(c_1, w) + d_H(w, c_2) \leq 2 \left\lfloor \frac{d-1}{2} \right\rfloor \leq d - 1,$$

but $c_1 - c_2$ is a codeword, and has Hamming weight $d_H(c_1, c_2)$, which must be at least d. Since we just showed that $d_H(c_1, c_2) < d$, this is a contradiction. Thus, for any word, there is at most one codeword within distance t of it. So if a codeword has t or fewer errors, we can correct it to the unique codeword within Hamming distance t.

If a codeword has fewer than $d - 1$ errors, then the errors cannot take it to another codeword, so we can detect that there is an error (although we cannot necessarily correct it to the original codeword). $\square$.

The dual of a code C , is $C^\perp$, the set of words that are perpendicular to every codeword in C. That is,

$$C^\perp = \{x | x \cdot c = 0 \forall c \in C\}.$$

A generator matrix for $C^\perp$ is a parity check matrix for C. The standard variables to use is G for generator matrices and H for parity check matrices.

A parity check matrix for the Hamming code is H^T, where

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

Note that the rows of H are the same as the first three rows of G, so for the Hamming code, $C^\perp \subset C$. Since all the rows of H are perpendicular to the rows of G, the parity check matrix has the property that $GH^T = 0$, which is important for error correction.

The Hamming code is called a $(7, 4, 3)$ code because it maps 4 bits into 3 bits and has minimum weight 3. The code whose generator matrix is H is a $(7, 3, 4)$ code. In fact, every non-zero codeword has Hamming weight 4; this is not hard to see. And knowing this, when we add the last row of G, we get a codeword where every word except the zero codeword and the all-ones codeword has Hamming weight either 3 or 4.

How do we correct an error in the Hamming code? The Hamming code can correct one error, which we will represent as a vector such as $e = (0, 0, 1, 0, 0, 0, 0)$. If the

error is in a single bit, then only one coordinate of e will be 1 and the rest will be 0; here, the error is in the third bit. Now, let us suppose we receive a coded message with an error. This received message can be represented as $r = mG + e$. To correct the error, we multiply by the matrix H^T:

$$rH^T = (mG + e)H^T = eH^T,$$

since $GH^T = 0$. The string eH^T is called the *syndrome*, because this is what we use to diagnose the error. Since e contained a single 1 in (say) the k'th position, eH will be the k'th row of H^T.

Let's do an example.

Suppose we receive the noisy codeword $(1, 0, 1, 1, 0, 1, 0)$. How do we find the error? We multiply by H^T, so

$$\Big(1,\ 0,\ 0,\ 1,\ 0,\ 1,\ 0\Big) \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} = \Big(0,\ 1,\ 1\Big)$$

Now, $rH^T = \big(0,\ 1,\ 1\big)$, which is the third row o f H^T. Thus, the third bit of our received codeword is in error. and the correct codeword is $\big(1,\ 0,\ 1,\ 1,\ 0,\ 1,\ 0\big)$. Linear algebra shows that this the encoding of message $\big(1,\ 0,\ 1,\ 0\big)$

Recall that for the Hamming code, $C^\perp \subset C$. A code with $C \subseteq C^\perp$ is called *weakly self-dual*, and these codes are important for building quantum error-correcting codes.

So what is the quantum Hamming code? It will encode two bits into one bit. The logical $|0\rangle$ is the superposition of all eight of the codewords in H

$$|0\rangle_L = \frac{1}{\sqrt{8}} \sum_{c \in H} |c\rangle$$

The logical $|1\rangle$ is the superposition of the eight codewords that are in G and not in H. These are the codewords of H XORed with the all 1's vector:

$$|1\rangle_L = \frac{1}{\sqrt{8}} \sum_{c \in H} |c \oplus 1111111\rangle$$

.

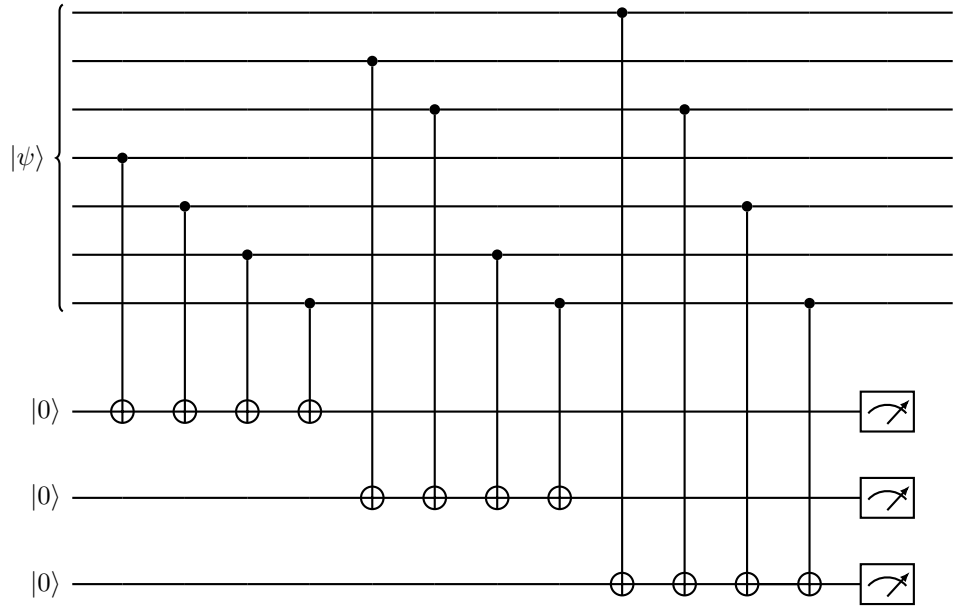
Why does this correct an arbitrary one-bit error. To show this, it turns out that we only need to show that it corrects a single σ_x error and a single σ_z error. Because this code treats σ_x errors and σ_z errors separately, it will automatically correct a single σ_y error, and from this we can deduce that it will correct an arbitrary single-qubit error. (This was proven in the lecture notes on the nine-qubit code.)

Why does it correct one bit-flip error? This state is a superposition of elements of the classical Hamming code, and we can apply the classical Hamming error correction

algorithm to each of these elements so as to correct a single bit-flip error. We do this by computing the syndrome. If b_i is the i'th bit, then the syndrome is:

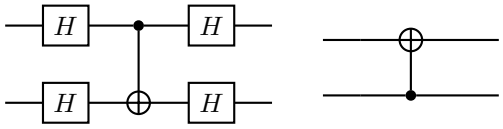
$$(b_4 \oplus b_5 \oplus b_6 \oplus b_7,\ \ b_2 \oplus b_3 \oplus b_6 \oplus b_7,\ \ b_1 \oplus b_3 \oplus b_5 \oplus b_7).$$

This can be computed by the following quantum circuit:



Once the syndrome has been computed, the error needs to be determined and then corrected. The determination of the error from the syndrome can be done classically.

Why does the Hamming code correct one phase-flip error? Recall that the Hadamard gate H takes bit-flip errors to phase-flip errors and vice versa. So we need to compute what happens when we apply a Hadamard gate to every qubit of the code. It turns out that we get the same quantum code back. Thus, the circuit for measuring the syndrome for phase flips is the same as the circuit for measuring the syndrome for bit flips, except that we put seven Hadamard gates on the qubits in the quantum code at the front of the circuit, and seven more at the back of the circuit. Alternatively, we can just use the identity



to turn all the CNOT gates upside-down.

We haven't actually done the calculations to show that $H^{\otimes 7}$ gives the same quantum code back. Let's see what happens when we apply the Hadamard transform to the quantum code. We have

$$|0\rangle_L = \frac{1}{\sqrt{8}} \sum_{x \in H} |x\rangle .$$

Applying the Hadamard transform gives

$$H^{\otimes 7} |0\rangle_L = \frac{1}{\sqrt{8}} \frac{1}{\sqrt{2^7}} \sum_{x \in H} \sum_{y \in Z_2^7} (-1)^{x \cdot y} |y\rangle .$$

Now, we need to figure out what $\sum_{x \in H} (-1)^{x \cdot y} |y\rangle$ gives. If $y \in G$, then since every element of G is orthogonal to every element of H, all the phases are $+1$, and we get $8\,|y\rangle$. If $y \notin G$, then half of the phases will be -1 and half will be $+1$, so they cancel out. Thus,

$$H^{\otimes 7} |0\rangle_L = \frac{8}{\sqrt{8}} \frac{1}{\sqrt{2^7}} \sum_{y \in G} |y\rangle = \frac{1}{4} \left(\sum_{y \in H} |y\rangle + \sum_{y \in G \setminus H} |y\rangle \right) = \frac{1}{\sqrt{2}} \big(|0\rangle_L + |1\rangle_L \big)$$

What about $H^{\otimes 7} |1\rangle_L$? Here,

$$H^{\otimes 7} |1\rangle_L = \frac{1}{\sqrt{8}} \frac{1}{\sqrt{2^7}} \sum x \in G \setminus H \sum_{y \in Z_2^7} (-1)^{x \cdot y} |y\rangle .$$

If $y \notin G$, then the sum is 0 for the same reason. If $y \in H$, then the sum is $8\,|y\rangle$, and if $y \in G \setminus H$, the sum is $-8\,|y\rangle$. This is just because the inner product of two vectors in $G \setminus H$ is -1. Thus, we get

$$H^{\otimes 7} |1\rangle_L = \frac{1}{\sqrt{2}} \big(|0\rangle_L - |1\rangle_L \big)$$

So both $H^{\otimes 7} |0\rangle_L$ and $H^{\otimes 7} |1\rangle_L$ are states in our quantum Hamming code, and thus bit-flip errors in $H^{\otimes 7} |0\rangle_L$ and $H^{\otimes 7} |1\rangle_L$ can be corrected by the error correction procedure. This means that phase-flip errors of $|0\rangle_L$ and $|1\rangle_L$ can be corrected by applying $H^{\otimes 7}$, applying the bit-flip error correction procedure, and applying $H^{\otimes 7}$ again, and we have shown that the quantum Hamming code is a error correcting code that can correct any single-bit error.

There is one last thing I want to say about the quantum Hamming code. How do we apply a logical σ_x? You can turn any codeword in H into a codeword in $G \setminus H$ and vice versa by adding the vector 1111111 (actually, adding any codeword in $G \setminus H$ will do). So the operation corresponding to a logical $|0\rangle$ is just $\sigma_x^{\otimes 7}$.

How do you apply a logical σ_z? It turns out that you can apply a $\sigma_z^{\otimes 7}$. One way to see this is using the fact that a Hadamard get applied to each qubit applies a Hadamard gate to the logical qubit, and similarly a σ_x gate applied to each qubit also applies a σ_x gate to the logical qubit. But you can also see it by realizing that $\sigma_z^{\otimes 7}$ applies a $+1$ phase to any $|y\rangle$ for $y \in H$ and a -1 phase to any $|y\rangle$ for $y \in G \setminus H$.