Notes 8.370/18.435 Fall 2022

Lecture 27 Prof. Peter Shor

Today, we start our unit on quantum error-correcting codes.

I'm going to start with a digression into classical information theory. In 1948, Claude Shannon published a paper, "A Mathematical Theory of Communication", which started the field of information theory. In it, he showed that noisy communication channels had a *capacity*, and he derived *Shannon's formula* for the capacity of a channel. There is a protocol such that you can send information over a channel at a rate less then its capacity, in the limit as the length of the message goes to infinity, and almost always succeed. On the other hand, if you try to send information over a channel at a rate larger than the capacity, it is overwhelmingly likely that the message the receiver gets has errors—i.e., it is different from the message that the sender transmitted.

Claude Shannon's theory was not constructive. While he showed that an algorithm existed that would succeed in transmitting information at nearly the channel capacity existed, it was a randomized construction that didn't explicitly produce such an algorithm. Worse, the algorithm itself would take exponential time to implement in the most straightforward way. It wasn't until forty-five years after Shannon's paper that somebody experimentally found an efficient coding and decoding procedure that came close to the channel capacity (*turbo codes*), and not until 60 years later until a coding method was found that provably approached Shannon's bound in the limit of long messages (these were *polar codes*).

However, only two years after Shannon published his paper, Richard Hamming discovered the first error-correcting codes, and in the next few decades, computer technology advanced greatly and error correcting codes were developed to the point where they could be used in practice for the reliable transmission of information over noisy channels. Hamming codes are one of a class of codes called *linear codes*, and these have nice properties. They are the class of codes that is most often used in practice, and they have received extensive study.

We will look at the code that Hamming discovered much more closely in future classes, but I want to explain a little bit about how it works right now. You take a four-bit message, and encode it into a seven-bit message in such a way that even if one of the seven bits is wrong, you can still recover the original four bits. The encoding is done by multiplying the message m by a generator matrix G, where

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

The decoding is done by multiplying by a matrix H, where

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

We have $GH = 0 (\bmod 2)$. This is because $G = \begin{pmatrix} I_4 & S \end{pmatrix}$ and $H = \begin{pmatrix} S \\ I_3 \end{pmatrix}$, so $GH = 2S = 0$, where S is the 4×3 matrix

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

Suppose that there is a one-bit error in the received transmission r. Then

$$r = mG + e$$

and

$$rH = mGH + eH = eH,$$

so $rH = eH$ is independent of the message, and tells you what the error is. It is called the *syndrome* (because you can use it to diagnose the error). Computing the error e from the syndrome eH is computationally difficult problem in general; the hard part of designing error-correcting codes is finding codes where there is an efficient algorithm going from the syndrome to the error.

Before Hamming, the only error-correcting codes engineers knew about were *repetition codes*. In these, you just repeat the message bit k times. If $k = 2t + 1$, the code can correct t errors. These codes can be put into the framework of linear codes described above. For a repetition code3, the generator matrix is

$$G = [1, 1, 1, 1, \ldots, 1].$$

Let's look at the three-bit repetition more carefully. Assume that each bit has a p probability of having an error, and a $1 - p$ probability of being correct. Then there will be an error in an encoded bit if and only if at least two of the encoding bits have errors, so instead of a probability p of having an error, the probability is $3p^2(1 - p) + p^3$, which is an improvement as long as $p < \frac{1}{2}$, and is around $3p^2$ if p is small.

Today, we will look at the quantum analog of repetition codes. Since it is impossible to clone a qubit, you can't actually implement a repetition code $|\psi\rangle \rightarrow |\psi\rangle |\psi\rangle |\psi\rangle$. One thing you could do instead is use the unitary:

$$U |0\rangle |00\rangle = |000\rangle$$
$$U |1\rangle |00\rangle = |111\rangle$$

Here, we first adjoint the qubits $|00\rangle$ to the qubit we want to encode and then perform the unitary.

This is a three-qubit code that protects against bit-flip errors. It was first investigated by Asher Peres in 1985. The code maps

$$\alpha |0\rangle + \beta |1\rangle \to \alpha |000\rangle + \beta |111\rangle ,$$

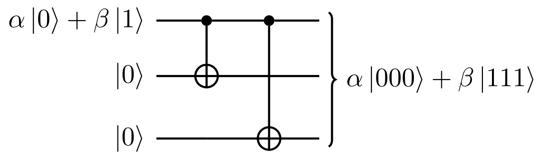
so you can see it does not perform quantum cloning; if you cloned the original qubit, you would get $\big(\alpha |0\rangle + \beta |1\rangle\big)^{\otimes 3}$.

This code can correct one bit-flip, or σ_x, error. How does this work? We measure the answer to the question "which bit is different?" More specifically, we project the three qubits onto one of the following four subspaces:

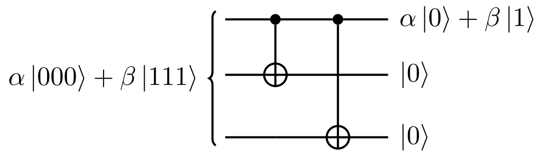
$$\begin{aligned}
&|000\rangle\langle 000| + |111\rangle\langle 111| \\
&|100\rangle\langle 100| + |011\rangle\langle 011| \\
&|010\rangle\langle 010| + |101\rangle\langle 101| \\
&|001\rangle\langle 001| + |110\rangle\langle 110| .
\end{aligned}$$

Once we know which subspace are in, we can correct the error. For example, if the state was projected onto the third subspace above, we would apply a σ_x to the second qubit to correct it.

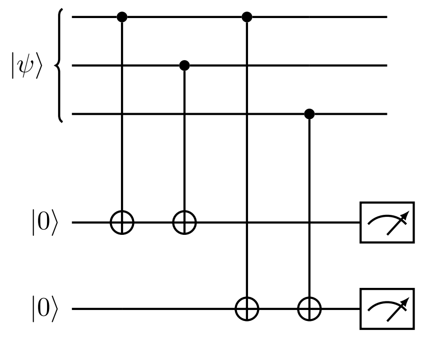
How do we encode a qubit into this code? We use the following circuit:


$\alpha |0\rangle + \beta |1\rangle$
$|0\rangle$
$|0\rangle$
$\alpha |000\rangle + \beta |111\rangle$

How do we decode? We basically reverse the circuit above, which gives exactly the same circuit since the two CNOT gates commute with each other:


$\alpha |000\rangle + \beta |111\rangle$
$\alpha |0\rangle + \beta |1\rangle$
$|0\rangle$
$|0\rangle$

How do we correct errors? We use the following quantum circuit, that projects onto one of the four subspaces described above:



The two measurement results here are called the *syndrome* and tell us what the error is.

If the two measurements are both $|0\rangle$, then we know that $|\psi\rangle$ was in the code subspace. If the first one is $|1\rangle$ and the second is $|0\rangle$, we know that the first and third qubits are qual, but the first and second qubits are different. This means that the second qubit must be different. Thus, if there is only one error, it is on the second qubit, and we can correct the state by applying σ_x to the second qubit. Similarly, if we get the measurement results $(0, 1)$, we know the third qubit is different, and if the measurement results are $(1, 1)$, we know the first qubit is different.

For bit-flip errors, the probabilities work exactly the same way as they did for the classical three-bit code. If the probabililty of a bit-flip error on a single qubit is p, then the probability of a bit-flip error on the encoded qubit is $3p^2 + p^3$, which is approximately $3p^2$ for small p.

But what about phase-flip errors? What happens if we apply a σ_z to one of the three qubits in the code? We will call an encoded $|0\rangle$ and $|1\rangle$ a *logical* $|0\rangle$ and $|1\rangle$, and we will represent them by

$$|0\rangle_L = |000\rangle \qquad\qquad |1\rangle_L = |111\rangle$$

If we apply a phase-flip error in any of three three encoding qubits, it will take $|0\rangle$ to $|0\rangle$ and $|1\rangle$ to $-|1\rangle$. That is, it will apply a phase error to the logical qubit. So if the probability of a phase error on a single qubit is p, the probability of a phase error on the encoded qubit is $3p + p^3$, or approximately $3p$ for small p.

So we can reduce the error rate for bit flip errors at the cost of increasing the error rate for phase flip errors. Is there anything we can do to protect phase-flip errors?

There is. Recall that a Hadamard gate took a σ_x to a σ_z and vice versa. That is, it takes a bit flip error to a phase flip error and vice versa. Thus, we can find a code that interchanges the role of bit-flip and phase-flip errors by applying a Hadamard gate to each of our qubits. This code is simply

$$\begin{aligned}|0\rangle &\to |+++\rangle\\ |1\rangle &\to |---\rangle\,.\end{aligned}$$

There is an equivalent way to represent this code, namely:

$$\begin{aligned}|0\rangle_L &\to \frac{1}{2}\big(\,|000\rangle + |011\rangle + |101\rangle + |110\rangle\,\big),\\ |1\rangle_L &\to \frac{1}{2}\big(\,|100\rangle + |010\rangle + |001\rangle + |111\rangle\,\big).\end{aligned}$$

We obtain this alternate representation by applying a Hadamard gate to the encoded qubit before encoding it. This code encodes a $|0\rangle$ as the superposition of all states with an even number of 0's, and a $|1\rangle$ as the superposition of all states with an odd number of 1s. It protects against phase-flip errors, but any bit-flip error in an encoding qubit results in a bit-flip error on the logical qubit, so bit-flip errors are roughly three times as likely as on unencoded qubits. This can be seen directly from the fact that a bit-flip error takes a bit string with an odd number of 1's to a bit string with an even number of 1's.

So now we have a choice: we can protect against one bit-flip error, but only by making phase-flip errors more likely, or we can protect against one phase-flip errors but only by making bit-flip errors more likely. Is there any way around this problem?

It turns out there is. The answer comes from a technique of classical coding theory: you *concatenate* the two codes. This means you first encode using one code, and then you encode using the other. Let's see how this works.

$$\begin{aligned}|0\rangle &\to |+++\rangle \to (|000\rangle + |111\rangle)^{\otimes 3}\\ |1\rangle &\to |---\rangle \to (|000\rangle - |111\rangle)^{\otimes 3}\end{aligned}$$

Now, what happens? if you have a bit-flip error, it gets corrected immediately by the inner code. If you have a phase-flip error, the inner code turns this into a phase-flip on the logical qubit of the inner code, which gets corrected by the outer code. Thus, any single σ_x or σ_z can be corrected.

How about σ_y errors? They can be corrected as well. A σ_y error can be viewed as a σ_x error and a σ_z error acting on the same qubit, since $\sigma_y = i\sigma_x\sigma_z$. Thus, any single σ_y error can be corrected as well—the inner code will correct the σ_x component and the outer code the σ_z component.

But what about more general errors? It turns out that the 9-qubit code given in this lecture can correct these as well, as long as they are restricted to one qubit. We will see this in the next lecture.