Quantum Information Processing

Quantum Information Theory (II)

Richard Cleve

Institute for Quantum Computing & Cheriton School of Computer Science

University of Waterloo

November 15, 2022

Abstract

The goal of these notes is to explain the basics of quantum information processing, with intuition and technical definitions, in a manner that is accessible to anyone with a solid understanding of linear algebra and probability theory.

These are lecture notes for the third part of a course entitled "Quantum Information Processing" (with numberings QIC 710, CS 768, PHYS 767, CO 681, AM 871, PM 871 at the University of Waterloo). The other parts of the course are: a primer for beginners, quantum algorithms, and quantum cryptography. The course web site `http://cleve.iqc.uwaterloo.ca/qic710` contains other course materials, including video lectures.

I welcome feedback about errors or any other comments. This can be sent to `cleve@uwaterloo.ca` (with "Lecture notes" in subject, if at all possible).

Contents

1 Distance measures between states

This section is about distance measures between quantum states. We'll see various ways of quantifying how different two quantum states are, including the *fidelity* and the *trace distance*. I'll also show you the Holevo-Helstrom Theorem, which relates trace distance to the operational problem of distinguishing between a pair of states by a measurement.

Recall that we consider two quantum states to be *indistinguishable* if, for any measurement procedure, the probability distribution of outcomes is identical between the two states. For example, $|0\rangle$ and $-|0\rangle$ are indistinguishable. In [*Part 2, section 1*] we saw different probabilistic mixtures that resulted in the same density matrix. In all such cases, the two states are indistinguishable; we don't even consider them to be different states.

Definition 1.1 (distinguishable states)**.** *We say two states are* distinguishable *if they're not indistinguishable. In other words, if for some measurement procedure, the outcome probabilities are different for the two states.*

For example, the $|0\rangle$ and the $|+\rangle$ state are distinguishable. Note that our definition of distinguishable does not require us to be able to perfectly tell the two states apart. Another example is $\frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1|$ and $|+\rangle\langle +|$.

Definition 1.2 (perfectly distinguishable states)**.** *Define two states to be* perfectly distinguishable *if there is a measurement procedure that perfectly tells them apart.*

For example, $|+\rangle$ and $|-\rangle$ are perfectly distinguishable.

We have three qualitative categories: indistinguishable, distinguishable, and perfectly distinguishable. Can we quantify how different two states are?

1.1 Operational distance measure

An operational way of quantifying the difference between two states is based on the guess-the-state game that we've seen several times.

For any two states, which in general can be mixed states, imagine the game where Alice flips a fair coin to decide which of the two states to set a quantum system to, and then she sends the quantum system to Bob. Bob knows what the two possible states are, but Alice does not tell him which one she chose. Bob's goal is to apply a measurement procedure measurement to the state that he received and to use the classical outcome to guess which state it was. We can write the success probability of Bob's optimal measurement procedure as $(1+\delta)/2$, where $\delta \in [0,1]$.

The trivial strategy for Bob is to just guess a random bit (ignoring the system that he receives from Alice). That strategy succeeds with probability $\frac{1}{2}$. We can think of δ as how much better one can do than that baseline. Another way of viewing δ is as the success probability minus the failure probability. It's sometimes useful to look at δ that way.

For a given pair of quantum states, what's the best δ attainable? If the two states are indistinguishable then $\delta = 0$ is the best possible. At the other extreme, if the two states are perfectly distinguishable then the success probability can be 1, so $\delta = 1$.

An in-between case is when the two states are $|0\rangle$ and $|+\rangle$. These are not perfectly distinguishable, but the distinguishing probability can be as high as

$$\cos^2(\tfrac{\pi}{8}) = \frac{1+\cos(\frac{\pi}{4})}{2} = \frac{1+\frac{1}{\sqrt{2}}}{2}, \tag{1}$$

so $\delta = \frac{1}{\sqrt{2}} \approx 0.707$. Another in-between case is $\frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1|$ vs. $|+\rangle\langle +|$, for which the best possible distinguishing probability is $\frac{3}{4} = \left(1+\frac{1}{2}\right)/2$, so in that case $\delta = \frac{1}{2}$.

This is one natural way of quantifying the distance between two states, in terms of the highest distinguishing probability possible. A natural question is: given two density matrices ρ_0 and ρ_1, what is the highest distinguishing probability possible? In section 1.5, we'll see a systematic way of addressing such questions.

1.2 Geometric distance measures

Now, let's look at the distance between two quantum states form a different perspective: a geometric perspective.

1.2.1 Euclidean distance

If we have two d-dimensional pure states, $|\psi_0\rangle$ and $|\psi_1\rangle$ then it seems natural to take the Euclidean distance between their state vectors $\big||\psi_0\rangle - |\psi_1\rangle\big|_2$.

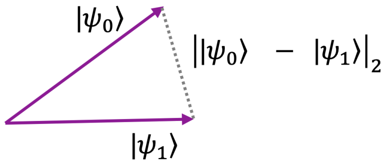

Figure 1: Euclidean distance between pure states $|\psi_0\rangle$ and $|\psi_1\rangle$.

If their Euclidean distance is small then it's intuitively natural to think of the states as "close". But, if their Euclidean distance is large, should we think of the states as "far apart"? Not necessarily, because of global phases. Note that $-|\psi\rangle$ is the unit vector farthest away from $|\psi\rangle$ (the Euclidean distance being 2), even though these are the same state.

Also, it is not so clear how this notion of Euclidean distance can be extended to mixed-states.

1.2.2 Fidelity

Another notion of distance is called the *fidelity*, which for pure states is the absolute value of the inner product between the state vectors $\big|\langle\psi_0|\psi_1\rangle\big|$.

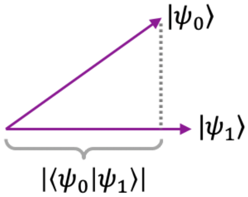

Figure 2: Fidelity between pure states $|\psi_0\rangle$ and $|\psi_1\rangle$.

This distance measure is calibrated in reverse in the sense that *large* fidelity means "close" and *small* fidelity means "far". Clearly, fidelity 1 means indistinguishable and fidelity 0 means perfectly distinguishable (since orthogonal states are perfectly distinguishable). Notice how the absolute value takes care of any distinctions between vectors due to global phases.

For the in-between values of fidelity, if the fidelity is close to 1 means that the states are "close".

What about the fidelity between mixed states? It turns out that there is a definition of fidelity for mixed states. If ρ_0 and ρ_1 are the density matrices of the two mixed states, then the fidelity is given by the formula

$$F(\rho_0, \rho_1) = \mathrm{Tr}\Big(\sqrt{\sqrt{\rho_0}\,\rho_1\sqrt{\rho_0}}\Big). \tag{2}$$

I'm not going to explain this formula, but I want you to see it. You cannot simplify this formula by cyclically permuting one of the $\sqrt{\rho_0}$ factors to the other side because of the square root within the trace. One reasonable property that this has is that, it agrees with the definition $\big|\langle\psi_0|\psi_1\rangle\big|$ for the very special case of pure states. This is easy to verify, which I'll leave as an exercise.

Exercise 1.1. *Prove that, if* $\rho_0 = |\psi_0\rangle\langle\psi_0|$ *and* $\rho_1 = |\psi_1\rangle\langle\psi_1|$ *then it holds that* $F(\rho_0, \rho_1) = \big|\langle\psi_0|\psi_1\rangle\big|$.

After looking at that expression for fidelity involving all those square roots of matrices, let's think about what it means to take the square root of a matrix. This is part of a more general *functional calculus* on square matrices.

1.3 Functional calculus for linear operators

Suppose that M is a normal matrix and $f : \mathbb{C} \to \mathbb{C}$. Then we can define f applied to the matrix M as follows. Since M is normal, we can diagonalize M in some orthonormal basis (the columns of some unitary U) as

$$M = U^* \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_d \end{bmatrix} U. \tag{3}$$

Define $f(M)$ as the matrix where f is applied to each eigenvalue, namely,

$$f(M) = U^* \begin{bmatrix} f(\lambda_1) & 0 & \cdots & 0 \\ 0 & f(\lambda_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f(\lambda_d) \end{bmatrix} U. \tag{4}$$

Square root of a positive matrix

Every $x \in \mathbb{C}$ has at least one square root, and if $x \neq 0$ then x has two square roots. However, if $x \in \mathbb{R}_+ = \{x \in \mathbb{R} : x \geq 0\}$ then x has a unique square root in $\mathbb{R}_+$. Whenever $M \geq 0$, there is a natural definition of $\sqrt{M}$ since then the eigenvalues of M are in $\mathbb{R}_+$ and have unique square roots in $\mathbb{R}_+$.

⚠ A word of caution: in general, for positive matrices L and M, it does *not* hold that $\sqrt{LM} = \sqrt{L}\sqrt{M}$. This is because L and M may not be simultaneously diagonalizable. They are each diagonalizable, but not necessarily with respect to the same orthonormal basis.

Von Neumann entropy of a positive matrix

Whenever $M \geq 0$, it makes sense to define $M \log M$ (which is related to the von Neumann Entropy of a quantum state, defined as $\mathrm{Tr}\big(M \log M\big)$), that will come up in a later part of the course.

Absolute value of any matrix

We can also define the absolute value of a normal matrix M using the functional calculus, as the absolute values of all the eigenvalues

$$|M| = U^* \begin{bmatrix} |\lambda_1| & 0 & \cdots & 0 \\ 0 & |\lambda_2| & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & |\lambda_d| \end{bmatrix} U. \tag{5}$$

Notice that

$$|M| = \sqrt{M^* M}, \tag{6}$$

and this interesting because for *any* (not necessarily normal) matrix M, it holds that $M^* M \geq 0$. Therefore, we have a definition of the absolute value that extends to any matrix (not necessarily diagonalizable).

1.4 Trace norm and trace distance

Now, I'd like to show you a very interesting distance measure between quantum states, called the trace distance. First, I'll show you the definition of the trace norm of a matrix, which is the trace of the absolute value of the matrix.

Definition 1.3 (trace norm). *For any $M \in \mathbb{C}^{d \times d}$, the* trace norm *of M is defined as*

$$\|M\|_1 = \mathrm{Tr}\big(|M|\big) = \mathrm{Tr}\big(\sqrt{M^* M}\,\big). \tag{7}$$

The notation is as a norm with subscript 1 (sometimes this alternative notation is used, with the subscript "tr").

It's not too hard to show that, if M is normal then the trace norm of M is the 1-norm of the vector of eigenvalues of M. In other words, if the eigenvalues of M are $\lambda_1, \lambda_2, \ldots, \lambda_d$ then

$$\|M\|_1 = |\lambda_1| + |\lambda_2| + \cdots + |\lambda_d|. \tag{8}$$

Now we are ready to define the trace distance between two states. It's the trace norm of the difference between their density matrices.

Definition 1.4 (trace distance)**.** *For any two d-dimensional states, whose density matrices are ρ_0 and ρ_1, the* trace distance *between them is*

$$\|\rho_0 - \rho_1\|_1. \tag{9}$$

Of all the different matrix norms on which we could base a distance measure, what's so special about the trace norm? Why is this a meaningful measure of distance between states? The answer is given by the amazing Holevo-Helstrom Theorem.

1.5 The Holevo-Helstrom Theorem

Remember the δ that arose in our discussion of state distinguishability? We defined δ to be the advantage over random guessing in the guess-the-state game. In fact, that δ is exactly the trace norm multiplied by $\frac{1}{2}$. So the trace norm coincides with how well the states can be distinguished in the guess-the-state game!

Theorem 1.1 (Holevo-Helstrom Theorem)**.** *Let ρ_0 and ρ_1 be the density matrices of two d-dimensional states. If one of these two states is prepared by the flip of a fair coin and then the best distinguishing procedure succeeds with probability*

$$\frac{1 + \frac{1}{2}\|\rho_0 - \rho_1\|_1}{2}. \tag{10}$$

(If the trace distance had been defined as $\frac{1}{2}\|\rho_0 - \rho_1\|_1$ instead of $\|\rho_0 - \rho_1\|_1$ then the factor of $\frac{1}{2}$ would not apear in Eq. (10). But we're stuck with the standard definition.)

To prove the Holevo-Helstrom Theorem, we need to show:

- There is a measurement whose success probability is $\frac{1}{2} + \frac{1}{4}\|\rho_0 - \rho_1\|_1$.
- No measurement can perform better than $\frac{1}{2} + \frac{1}{4}\|\rho_0 - \rho_1\|_1$.

1.5.1 Attainability of success probability $\frac{1}{2} + \frac{1}{4}\|\rho_0 - \rho_1\|_1$

In this section, we prove the attainability part of Theorem 1.1. Namely, that there exists a measurement that attains success probability $\frac{1}{2} + \frac{1}{4}\|\rho_0 - \rho_1\|_1$, where ρ_0 and ρ_1 are the density matrices of the states that we wish to distinguish between.

Note that, since ρ_0 and ρ_1 are Hermitian, $\rho_0 - \rho_1$ is also Hermitian. But notice that, because of the minus sign, $\rho_0 - \rho_1$ need not be positive. In general, $\rho_0 - \rho_1$ has some negative eigenvalues.

Consider the two projectors, Π_0 and Π_1, defined as follows. Let Π_0 be the projector onto the space of all eigenvectors of $\rho_0 - \rho_1$ whose eigenvalues are ≥ 0. Let Π_1 be the projector onto the space of all eigenvectors of $\rho_0 - \rho_1$ whose eigenvalues are < 0. Then Π_0 and Π_1 are orthogonal projectors and $\Pi_0 + \Pi_1 = I$. Therefore Π_0 and Π_1 are the elements of a POVM measurement. We'll show that this measurement succeeds with probability $\frac{1}{2} + \frac{1}{4}\|\rho_0 - \rho_1\|_1$.

First note that

$$\big(\Pi_0 - \Pi_1\big)\big(\rho_0 - \rho_1\big) = \big|\rho_0 - \rho_1\big|. \tag{11}$$

To see why this is so, think of $\Pi_0 - \Pi_1$ and $\rho_0 - \rho_1$ in diagonal form (they are simultaneously diagonalizable). $\Pi_0 - \Pi_1$ has a $+1$ eigenvalue in all the positions where the corresponding eigenvalue of $\rho_0 - \rho_1$ is non-negative. $\Pi_0 - \Pi_1$ has a -1 eigenvalue in all the positions where the corresponding eigenvalue of $\rho_0 - \rho_1$ is negative. Therefore, $\Pi_0 - \Pi_1$ flips the sign of all the negative eigenvalues of $\rho_0 - \rho_1$, resulting in $\big|\rho_0 - \rho_1\big|$.

Note that Eq. (11) implies that

$$\mathrm{Tr}\big((\Pi_0 - \Pi_1)(\rho_0 - \rho_1)\big) = \|\rho_0 - \rho_1\|_1. \tag{12}$$

We can also expand

$$\begin{aligned} \mathrm{Tr}\big((\Pi_0 - \Pi_1)(\rho_0 - \rho_1)\big) &= \mathrm{Tr}(\Pi_0\rho_0) - \mathrm{Tr}\,\Pi_0\rho_1) - \mathrm{Tr}(\Pi_1\rho_0) + \mathrm{Tr}(\Pi_1\rho_1) & (13)\\ &= \big(\mathrm{Tr}(\Pi_0\rho_0) + \mathrm{Tr}(\Pi_1\rho_1)\big) - \big(\mathrm{Tr}(\Pi_0\rho_1) + \mathrm{Tr}(\Pi_1\rho_0)\big), & (14) \end{aligned}$$

where $\frac{1}{2}\big(\mathrm{Tr}(\Pi_0\rho_0) + \mathrm{Tr}(\Pi_1\rho_1)\big)$ is the success probability,[1] and $\frac{1}{2}\big(\mathrm{Tr}(\Pi_0\rho_1) + \mathrm{Tr}(\Pi_1\rho_0)\big)$ is the failure probability.[1] So the success probability minus the failure probability is equal to $\frac{1}{2}$ times the trace distance.

Note that our proof of this part is constructive. It actually gives us a recipe to determine the optimal measurement for distinguishing between two mixed states: consider the matrix that is one density matrix subtracted from the other density matrix; take the projector to the positive eigenspace of this matrix and the projector to the negative eigenspace of this matrix; that's the POVM measurement that solves the distinguishing problem with success probability $\frac{1}{2} + \frac{1}{4}\|\rho_0 - \rho_1\|_1$.

[1] Averaged over the random choice of the state.

1.5.2 Optimality of success probability $\frac{1}{2} + \frac{1}{4}\|\rho_0 - \rho_1\|_1$

So far, we've proved that a particular measurement attains the proposed success probability. But how do we know that there isn't an even better measurement? In this section, we prove the optimality part of Theorem 1.1. Namely, that there does not exist a measurement whose success probability exceeds $\frac{1}{2} + \frac{1}{4}\|\rho_0 - \rho_1\|_1$.

Let E_0 and E_1 be the POVM elements of any 2-outcome POVM measurement for distinguishing between ρ_0 and ρ_1. We'll prove an upper bound on it's performance.

First, notice that E_0 and E_1 are simultaneously diagonalizable.[2] Why? Because $E_1 = I - E_0$. So if E_0 is diagonal is some coordinate system then so is E_1. Therefore, we can write

$$E_0 = U^* \begin{bmatrix} p_0 & 0 & \cdots & 0 \\ 0 & p_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p_{d-1} \end{bmatrix} U \quad \text{and} \quad E_1 = U^* \begin{bmatrix} q_0 & 0 & \cdots & 0 \\ 0 & q_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & q_{d-1} \end{bmatrix} U. \qquad (15)$$

The eigenvalues of E_0 and E_1 are between 0 and 1 and corresponding eigenvalues sum to 1. It follows that the largest eigenvalue of $E_0 - E_1 \leq 1$.

Definition 1.5 (infinity norm)**.** *For a normal matrix M, the* infinity norm[3] $\|M\|_\infty$ *is defined as the absolute value of the largest eigenvalue of M.*

In this language, we have that $\|E_0 - E_1\|_\infty \leq 1$. We will make use of the following lemma, which is an instance of Hölder's inequality for matrices.

Lemma 1.1. *For any Hermitian L and M,* $\mathrm{Tr}(LM) \leq \|L\|_\infty \|M\|_1$.

Now, let's continue proving that any POVM measurement for distinguishing between ρ_0 and ρ_1 does not outperform the measurement that we constructed. Define

$$A = \frac{\rho_0 + \rho_1}{2} \quad \text{and} \quad B = \frac{\rho_0 - \rho_1}{2}. \qquad (16)$$

Note that $\rho_0 = A + B$, $\rho_1 = A - B$, and $\mathrm{Tr}(A) = 1$.

[2] Please note that this only holds because it's a 2-outcome POVM measurement. For POVM measurements with 3 or more outcomes, the matrices might not be simultaneously diagonalizable.

[3] This is equivalent to the *spectral norm*, which is defined for all matrices.

The success probability (averaged over inputs) is

$$\begin{aligned}
\tfrac{1}{2}\operatorname{Tr}(E_0\rho_0) + \tfrac{1}{2}\operatorname{Tr}(E_1\rho_1) &= \tfrac{1}{2}\operatorname{Tr}(E_0(A+B)) + \tfrac{1}{2}\operatorname{Tr}(E_1(A-B)) && (17)\\
&= \tfrac{1}{2}\operatorname{Tr}((E_0+E_1)A) + \tfrac{1}{2}\operatorname{Tr}((E_0-E_1)B) && (18)\\
&\leq \tfrac{1}{2}\operatorname{Tr}(A) + \tfrac{1}{2}\|E_0-E_1\|_\infty\|B\|_1 && (19)\\
&= \tfrac{1}{2} + \tfrac{1}{4}\|\rho_0-\rho_1\|_1. && (20)
\end{aligned}$$

This proves optimality and we have now completed the proof of the Holevo-Helstrom Theorem.

1.6 Purifications and Ulhmann's Theorem

Next, I'd like to show you what a *purification* of a quantum state is. Any mixed state can be viewed as a pure state on some larger system with part of that larger system traced out.

Let ρ be any density matrix of a d-dimensional system. Suppose ρ can be written as a probabilistic mixture of m pure states, as

$$\rho = \sum_{k=0}^{m-1} p_k \left|\psi_k\right\rangle\!\left\langle\psi_k\right|. \tag{21}$$

Now, consider the pure state on a d-dimensional register and an m-dimensional register

$$\left|\phi\right\rangle = \sum_{k=0}^{m-1} \sqrt{p_k}\left|\psi_k\right\rangle \otimes \left|k\right\rangle. \tag{22}$$

It's easy to see that $\operatorname{Tr}_2\left|\phi\right\rangle\!\left\langle\phi\right| = \rho$. The pure state $\left|\phi\right\rangle$ is called a *purification* of ρ. This is one way to purify ρ, but the purification of a state is not unique.

Now, let's recall the previous strange-looking definition of fidelity between general mixed states that I showed you earlier—but which I didn't explain. Using our language of the trace norm, we can rewrite the expression for fidelity as

$$F(\rho_0,\rho_1) = \left\|\sqrt{\rho_0}\sqrt{\rho_1}\right\|_1. \tag{23}$$

For any two density matrices, we can take purifications of them and then take the inner product of these purifications. The result will depend on which purification we choose. The following theorem relates the fidelity between mixed states with inner products of their purifications.

Theorem 1.2 (Ulhmann's Theorem)**.** *For any two mixed states ρ_0 and ρ_1, the fidelity between them is the maximum $\langle\phi_0|\phi_1\rangle$ taken over all purifications $|\phi_0\rangle$ and $|\phi_1\rangle$.*

For further details, please see [Nielsen and Chuang, *Quantum Computation and Quantum Information*, pp. 410–411].

1.7 Fidelity vs. trace distance

We've discussed fidelity and trace distance, mostly the latter. Known relationships between them are

$$1 - F(\rho_0, \rho_1) \leq \|\rho_0 - \rho_1\|_1 \leq \sqrt{1 - F(\rho_0, \rho_1)^2}. \tag{24}$$

In particular, suppose that we have two mixed states with density matrices ρ_0 and ρ_1, and we want to show that their trace distance is small. One way to do this is to construct purifications of them whose inner products are close to 1. By Ulhmann's Theorem and the second inequality above, for any purifications $|\phi_0\rangle$ and $|\phi_1\rangle$ we have

$$\|\rho_0 - \rho_1\|_1 \leq \sqrt{1 - \langle\phi_0|\phi_1\rangle^2}. \tag{25}$$

2 Simple quantum error-correcting codes

In this section, we begin the subject of quantum error-correcting codes, which can protect quantum states from noise. We'll first briefly review some results about error-correcting codes for *classical* information. Then we'll consider *quantum* error-correcting codes and I'll explain Shor's nine-qubit quantum error-correcting code.

Let's start by discussing what noise is. Broadly speaking, noise is when information gets disturbed. Imagine that Alice wants to send some bits to Bob, but their communication channel is flawed.

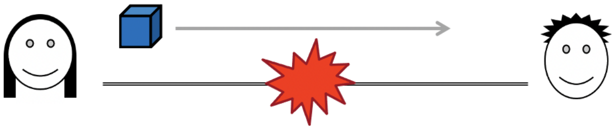

Figure 3: Alice sending Bob a bit over a noisy communication channel.

Suppose that, for each bit $b \in \{0,1\}$ that Alice sends via the channel, what Bob receives is

$$\begin{cases} b & \text{with prob. } 1-\epsilon \\ \neg b & \text{with prob. } \epsilon, \end{cases} \tag{26}$$

for some parameter $\epsilon \in [0, \frac{1}{2}]$. So each bit gets flipped with probability ϵ. This mapping from states of bits to states of bits is called a *binary symmetric channel*, and we refer to it as BSC (or as BSC_ϵ, to specify the parameter ϵ).

This channel can be viewed as a classical analogue of the depolarizing channel. Recall that the depolarizing channel takes a qubit as input and produces as output a probabilistic mixture of that state and the maximally mixed state. The binary symmetric channel outputs a probabilistic mixture of the input bit and a classical maximally mixed state (which is a uniformly distributed random bit). If we identify bit 0 with the probability vector $\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$ and bit 1 with $\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$ then the binary symmetric channel BSC_ϵ is a linear mapping such that

$$\mathsf{BSC}_\epsilon \begin{bmatrix}1\\0\end{bmatrix} = (1-2\epsilon)\begin{bmatrix}1\\0\end{bmatrix} + 2\epsilon\begin{bmatrix}\frac{1}{2}\\ \frac{1}{2}\end{bmatrix} \tag{27}$$

$$\mathsf{BSC}_\epsilon \begin{bmatrix}0\\1\end{bmatrix} = (1-2\epsilon)\begin{bmatrix}0\\1\end{bmatrix} + 2\epsilon\begin{bmatrix}\frac{1}{2}\\ \frac{1}{2}\end{bmatrix}. \tag{28}$$

Now suppose that Alice wants to communicate a bit $b \in \{0,1\}$ to Bob and their communication channel is a binary symmetric channel BSC_ϵ. Can Alice and Bob reduce the noise level to a smaller ϵ? The obvious way is for Alice and Bob to get a better communication hardware, with a smaller parameter ϵ. But Alice and Bob have an alternative to investing in better hardware: they can use an error-correcting code.

2.1 Classical 3-bit repetition code

Perhaps the simplest error-correcting code is the 3-bit repetition code, which works as follows. To transmit bit $b \in \{0,1\}$, Alice encodes b into three copies of b. Then Alice sends each of these three bits through the channel. Then Bob takes the majority value of the three bits that he receives (which might not all be the same, because some bits might get flipped by the channel).

How well does this perform? The system succeeds if no more than one bit is flipped (because that doesn't change the majority) and it fails if two or more bits are flipped. Assume that the channel behaves independently for each bit that passes through it.

Then the failure probability can be calculated as follows. There are three ways that one bit can be flipped, each occurring with probability $\epsilon^2(1-\epsilon)$. And there is one way that all three bits can flip, occurring with probability ϵ^3. So the failure probability is

$$3\epsilon^2(1-\epsilon) + \epsilon^3 = 3\epsilon^2 - 2\epsilon^3. \tag{29}$$

Here's a plot of the failure probability resulting from this scheme as a function of the original failure probability of the channel.

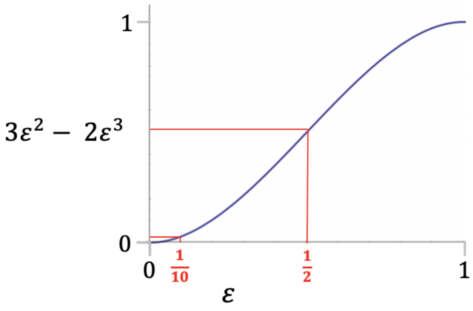

Figure 4: Failure probability of the 3-bit repetition code as a function of ϵ.

If $\epsilon = \frac{1}{2}$ then there is no improvement: the success probability remains at $\frac{1}{2}$. But this should not be surprising, because, if $\epsilon = \frac{1}{2}$ then the channel sends no information: it just outputs a random bit uncorrelated with the bit that Alice is sending. But when ϵ is smaller there is an advantage. For example, when $\epsilon = \frac{1}{10}$ the failure probability from using the code is around $\frac{1}{35}$. And the smaller ϵ is the more pronounced the error reduction is. If $\epsilon = \frac{1}{1000}$ then the failure probability from using the codes is around $\frac{1}{300000}$ (a three-hundred-fold decrease).

What price are we paying for this improvement? The main cost is that that three bits have to be sent instead of one.

Definition 2.1 (rate of a code)**.** *The* rate *of a code is the inverse of the expansion in message length due to the encoding.*

The rate of this code is $\frac{1}{3}$. Each bit of the encoding conveys $\frac{1}{3}$ of a bit of the data to be transmitted.

If the data is a long string of bits then there will be errors, but fewer errors using the code. With no code, the expected fraction of errors is ϵ. With the code, the expected fraction of errors is $3\epsilon^2 - 2\epsilon^3$. Suppose that Alice wants to send a Gigabyte to Bob (that's around 8.5 billion bits) and their communication channel has noise parameter $\epsilon = \frac{1}{100}$. Without the code, the fraction of errors will be around 85 million. With the code, the fraction of errors will be about 24 thousand. That's significantly fewer errors. Achieved at the cost of sending three Gigabytes instead of one.

But suppose the we don't want *any* errors. Can this be achieved? One approach is to use a larger repetition code than 3-bits. That reduces the error probability for each bit. But this also reduces the rate—so, in the above example, many more Gigabytes would have to be sent. But there are much better error-correcting codes than repetition codes.

2.2 Brief remarks about the existence of good classical codes

An error-correcting code need not separately encode each bit. Rather, each block of n bits (a message) can be encoded into a block of m bits (a codeword). The rate of such a code is $\frac{n}{m}$. Note that a small rate means a large message expansion (an inefficiency); whereas, a rate close to 1 means a small message expansion.

A fundamental result about the existence of good classical error-correcting codes can be informally stated as:

> For a binary symmetric channel with any error parameter $\epsilon < \frac{1}{2}$, the success probability for encodings of long strings can be made arbitrarily close to 1, *while maintaining a constant rate.*

So, in fact, Alice doesn't have to send many Gigabytes in order to get every single bit through to Bob correctly.

I'm going to state the result about good error-correcting codes more precisely. Please note that I'm not going to give the details of the construction or the analysis. The theory of error-correcting codes is a large field of study, that could easily take a course its own course to explain.

In order to state the result, we need some basic definitions for block codes. Such a code consists of an *encoding* function $E : \{0,1\}^n \to \{0,1\}^m$ and a *decoding* function $D : \{0,1\}^m \to \{0,1\}^n$. The rate of such a code is $\frac{n}{m}$.

Assuming that the communication channel is a binary symmetric channel with error parameter ϵ, the error probability of a code of the above form is defined as the maximum, for all $a_1 a_2 \dots a_n \in \{0,1\}^n$, of

$$\Pr\big[D(\mathsf{BSC}_\epsilon(E(a_1 a_2 \dots a_n)))\big] \neq a_1 a_2 \dots a_n. \tag{30}$$

Just to be clear: success means all the bits are successfully received at the other end; that there are no errors.

Definition 2.2 (Shannon entropy)**.** *The* Shannon entropy *of a probability vector* $(p_1, p_2, \ldots, p_d)$ *is defined as*

$$H(p_1, p_2, \ldots, p_d) = -\sum_{k=1}^{d} p_k \log p_k \,. \tag{31}$$

Define $R : [0, \frac{1}{2}] \rightarrow [0, 1]$ as $R(\epsilon) = 1 - H(\epsilon, 1 - \epsilon)$. Here is a plot of this function.

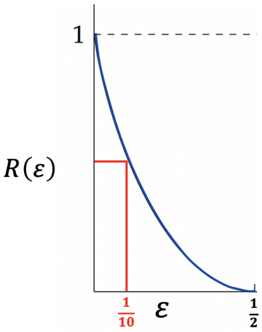

Figure 5: The rate function $R(\epsilon)$.

I'm now going to state the result about good multi-bit error-correcting codes. Let the noise level be any $\epsilon < 1/2$. Think of that as a property of the communication channel that you're stuck with using.

Then you can select any rate r, as long as $r < R(\epsilon)$. And you can select an arbitrarily small $\delta > 0$, which is your desired error probability bound. Then there exists an error-correcting code $E : \{0,1\}^n \rightarrow \{0,1\}^m$, $D : \{0,1\}^m \rightarrow \{0,1\}^n$ with rate $\frac{n}{m} > r$ and whose failure probability is less than δ.

There are additional considerations, that I'd like to mention, even though I won't go in the details:

- One is the block-length n. The smaller δ is, the larger n has to be.
- Another is the computational cost of computing E and D. The bottom line is that there are codes for which this can be done efficiently.

OK, so that was a very brief overview of error-correcting codes for classical information. The question is what happens with quantum error-correcting codes.

2.3 Shor's 9-qubit quantum error-correcting code

We started with a simple classical error-correcting code, the 3-bit repetition code. So we might be tempted to start with a quantum repetition code, where a qubit is encoded as three copies of itself.

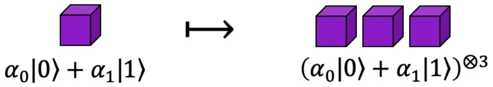
$\alpha_0|0\rangle + \alpha_1|1\rangle$
$\longmapsto$
$(\alpha_0|0\rangle + \alpha_1|1\rangle)^{\otimes 3}$

Figure 6: A naïve first attempt at a 3-qubit quantum repetition code.

Of course, this fails for multiple reasons, starting with the fact that a general quantum state cannot be copied (the no-cloning theorem).

It's easy to copy classical information, and all error-correcting codes for classical information are based on some sort of redundancy. But the no-cloning theorem kind of suggests that redundancy for quantum information might not be possible. That was the thinking shortly after Shor's algorithms for factoring and discrete log came out. But the underlying intuition that no-cloning implies no-redundancy was wrong.

I'm going to show you the first error-correcting code, that was discovered by Peter Shor in the mid-1990s. It's a 9-qubit code that is constructed by combining two 3-qubit codes that protect against very limited error types.

2.3.1 3-qubit code that protects against one X error

Let's start with a simple 3-qubit code that protects against a very restricted error-set. Suppose the only error possible is a Pauli X, a bit flip. Then these encoding and decoding circuits protect against up to one X-error.

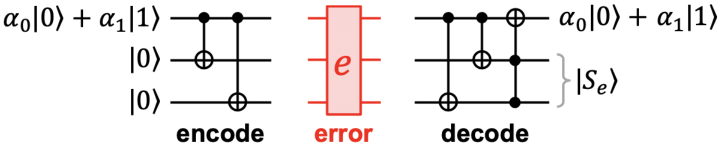
$\alpha_0|0\rangle + \alpha_1|1\rangle$
$|0\rangle$
$|0\rangle$
e
$\alpha_0|0\rangle + \alpha_1|1\rangle$
$|S_e\rangle$
encode
error
decode

Figure 7: 3-qubit code that protects against one X error.

The *encoding* circuit takes a qubit as input and produces three qubits as output. Then the encoded data is affected by an error e, where e can be any one of these four unitary operations:

$$\begin{array}{cccc} I \otimes I \otimes I & X \otimes I \otimes I & I \otimes X \otimes I & I \otimes I \otimes X \\ \text{(no error)} & \text{(flip 1st qubit)} & \text{(flip 2nd qubit)} & \text{(flip 3rd qubit)} \end{array} \tag{32}$$

Then the three qubits are input to the *decoding* circuit.

It turns out that, in all four cases of e, the data is correctly recovered. The final state of the first qubit is the same as its initial state. It's a straightforward exercise to verify these, but I recommend that you work through some of these cases to convince yourself.

If you work out the final states, you will see that the second and third qubits end up in a state that depends on what the error operation e is. We'll refer to these two qubits as the *syndrome of the error*, denoted as $|s_e\rangle$. So, not only does the encoding/decoding protect the data against the errors, but the decoding process also reveals what the error e is.

What about other errors? Specifically, what happens if there is a Z-error on one of the three encoded qubits? A Z-error is not corrected; instead it's passed through. By this, I mean that if the data is in state $\alpha_0 |0\rangle + \alpha_1 |1\rangle$ then applying a Z-error to one of the three encoded qubits causes the output of the decoding circuit to be $\alpha_0 |0\rangle - \alpha_1 |1\rangle$, which is the same as applying Z directly to the original data. So this encoding/decoding does *not* protect against an Z-error, but passes it through.

2.3.2 3-qubit code that protects against one Z error

Now, here's another 3-qubit code which protects against up to one Z-error.

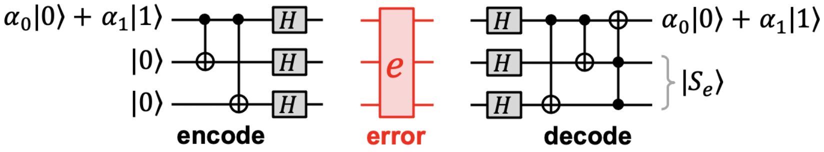
$\alpha_0|0\rangle + \alpha_1|1\rangle$
$|0\rangle$
$|0\rangle$
H
encode
e
error
decode
$\alpha_0|0\rangle + \alpha_1|1\rangle$
$|S_e\rangle$

Figure 8: 3-qubit code that protects against one Z error.

The encoding/decoding is like the code in figure 7 for protecting against X-errors, but with a layer of Hadamard gates added to the end of the encoding and the beginning of the decoding. This is essentially a re-purposing of our code for X-errors into a code for Z-errors. Think of a Z-error and the H gates. Since $HZH = X$ and $HH = I$, any Z-errors effectively become X-errors and then are handled as the previous encoding/decoding in figure 7.

2.3.3 9-qubit code that protects against one Pauli error

By combining the 3-qubit codes from figures 7 and 8, we can obtain a 9-qubit code that protects against any Pauli error (I, X, Y, or Z) in any one of the nine qubit positions. The encoding/decoding circuits are the following.

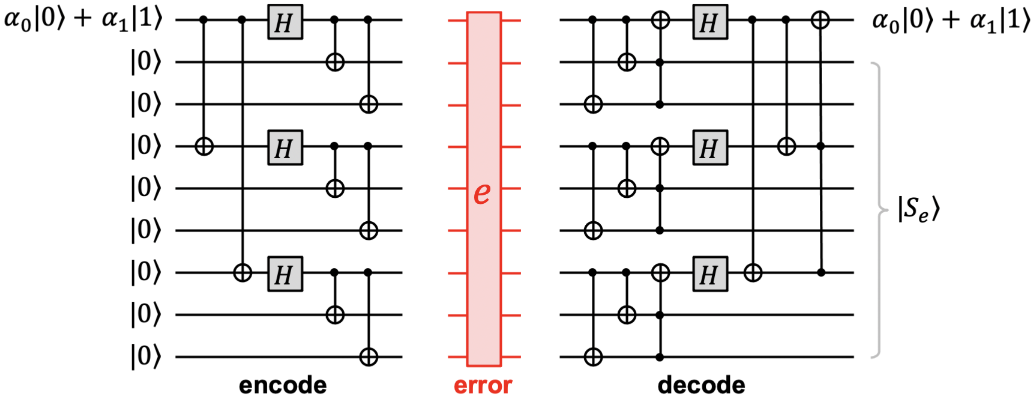

Figure 9: Shor's 9-qubit code.

A nice way of understanding how this code works is in terms of it's inner part (the last two layers of gates in the encoding part and first three layers of gates in the decoding part) and an outer part (the other gates). The inner part consists of three blocks, where each block is a copy of our code for X-errors from figure 7. And the outer part is our code for Z-errors from figure 8.

What happens if there's an X-error? It's corrected by the inner part. What happens if there's a Z-error? A Z-error is passed through by the inner part and then corrected by the outer part. What happens if there's a Y-error? Since $Y = iXZ$ (and the phase i makes no difference in this context), this can be viewed as a Z-error and an X-error. The inner part corrects the X-error and the outer part corrects the Z-error.

So if the error e is any Pauli error in one qubit position then it is corrected by this code; the data is recovered in the final state of the first qubit. There are 28 different one-qubit Pauli errors e (including $I^{\otimes 9}$) that this code protects against. The final state of eight qubits after the first qubit is the *error syndrome* $|s_e\rangle$, and contains information about which error was corrected.[4]

In fact, this codes corrects against an *arbitrary* error in any one qubit position and it is also effective for communicating through a channel with depolarizing noise.

[4] A curiosity is that some of the 28 possible one-qubit Pauli errors have the error syndrome.

2.4 Quantum error models

Let's step back and consider some of the error models that the Shor code protects against. There are several error models, but let's focus on two that are the most closely related to our discussion.

Worst-case unitary noise

By *worst-case unitary noise* noise, we mean that there is a set of possible unitary errors, and the error can be any one of them. For example, the Shor code is resilient against an arbitrary one-qubit Pauli error on a 9-qubit encoding.

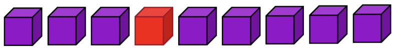

Figure 10: Arbitrary unitary error on one single qubit.

In fact, if a code is resilient against any 1-qubit Pauli error then it is resilient against *any* 1-qubit unitary operation U. To see why this is so, note that any 2×2 unitary U can be expressed as

$$U = \eta_0 I + \eta_x X + \eta_y Y + \eta_z Z, \tag{33}$$

for some $\eta_0, \eta_x, \eta_y, \eta_z \in \mathbb{C}$. Then it's a straightforeward exercise to deduce from figure 9 that, if the error is set to $e = I^{\otimes j-1} \otimes U \otimes I^{\otimes 8-j}$ (that is, applying U to the j-th qubit), then the output will state is

$$\big(\alpha_0 |0\rangle + \alpha_1 |1\rangle\big) \otimes \big(\eta_0 |s_0\rangle + \eta_x |s_{x,j}\rangle + \eta_y |s_{y,j}\rangle + \eta_z |s_{z,j}\rangle\big), \tag{34}$$

where $|s_0\rangle$, $|s_{x,j}\rangle$, $|s_{y,j}\rangle$, $|s_{z,j}\rangle$ are the respective error syndromes of I, X, Y, Z in position j.

More generally, there exist other codes, with m-qubit encodings which have the property there, for some threshold k, the error can be an arbitrary unitray operation acting on any subset of the qubits of size k.

Depolarizing noise

Our discussion of classical error-correcting codes was based on the binary symmetric channel, which flips any bit passing through it with probability ϵ. A quantum analogue of the binary symmetric channel is the depolarizing channel, which can be defined as the mixed unitary channel of the form

$$\begin{cases} I & \text{with prob. } 1-\epsilon \qquad \text{(no error)} \\ X & \text{with prob. } \epsilon/3 \qquad \text{(bit flip)} \\ Y & \text{with prob. } \epsilon/3 \qquad \text{(bit+phase flip)} \\ Z & \text{with prob. } \epsilon/3 \qquad \text{(phase flip)} \end{cases} \tag{35}$$

for a parameter $\epsilon \leq \frac{3}{4}$. This is like the binary symmetric channel, but allowing for the fact that a qubit can be flipped in more than one way (bit-flip, phase-flip, or both).

Exercise 2.1. *Show that the definition of the depolarizing channel in Eq.* (35) *is equivalent to our earlier definition of the depolarizing channel, as mapping each state* ρ *to a convex combination of that state and the maximally mixed state, namely*

$$p\rho + (1-p)\big(\tfrac{1}{2}\,|0\rangle\langle 0| + \tfrac{1}{2}\,|1\rangle\langle 1|\big) \qquad \textit{(for some } p\textit{).} \tag{36}$$

In the depolarizing noise model, we assume that every qubit of the encoded state is independently affected by a depolarizing channel. So all of the qubits incur an error; however, there is a bound ϵ on the severity of that error.

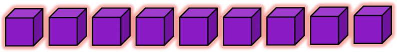

Figure 11: Depolarizing error on every qubit.

How does the Shor code perform in this model? For the depolarizing channel with parameter ϵ, let's think of ϵ as the *effective error probability*, since the qubit passes through the channel is undisturbed with probability $1-\epsilon$. If a qubit is encoded by the Shor code and each of the nine qubits of the encoding incurs a depolarizing error with parameter ϵ then we can analyze the effective error probability resulting from the encoding/decoding process. It turns out that this effective error probability is upper bounded $c\epsilon^2$, for some constant[5] c. So, if ϵ is small enough to begin with, then the reduction due to squaring ϵ is more than the effect of multiplying by c, so the effective error is decreased. The cost is that Alice has to send nine qubits to convey just one qubit. So the rate of this code is $\frac{1}{9}$.

Are there better codes than this? And are there good multi-qubit quantum error-correcting codes? This will be discussed in section 3.

[5] I don't know the exact constant, but it is less than 36.

2.5 Redundancy vs. cloning

The Shor code encodes one qubit in state $\alpha_0 |0\rangle + \alpha_1 |1\rangle$ into nine qubits, in state $\alpha_0 |0\rangle_L + \alpha_1 |1\rangle_L$ where

$$|0\rangle_L = \left(\tfrac{1}{\sqrt{2}}|000\rangle + \tfrac{1}{\sqrt{2}}|111\rangle\right)^{\otimes 3} \tag{37}$$

$$|1\rangle_L = \left(\tfrac{1}{\sqrt{2}}|000\rangle - \tfrac{1}{\sqrt{2}}|111\rangle\right)^{\otimes 3}. \tag{38}$$

Sometimes, $|0\rangle_L$ and $|1\rangle_L$ are referred to as the *logical zero* and *logical one* of the code.

The encoded state has the property that, if any error is inflicted on any one of its qubits then the data can still be recovered. To be clear, note that the recovery procedure is not provided with any information about *which* qubit has been affected by an error.

Which of the nine qubits contains the data? The answer is that no individual qubit contains any information about the qubit. The information is somehow "spread out" among the nine qubits.

A natural question is whether there is a more efficient code that requires fewer than nine qubits and protects against any one-qubit error. This question will be answered in section 3.

I'd like to briefly mention a different type of error, called an *erasure error*. For this type of error, the positions of the affected qubits are known. One way of viewing this is that some of the qubits are "lost," but we know the positions of the lost qubits—and the remaining qubits are undisturbed. For example, suppose that a qubit is encoded via the Shor code into nine qubits and then qubit 2 and qubit 6 go missing.

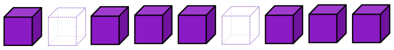

Figure 12: Erasure error on 2 qubits.

It turns out that the Shor code can handle any two erasure errors. From any seven of the nine encoded qubits, the data can be recovered.

Finally, here's a theorem that's very easy to prove but it helps clarify the distinction between redundancy and copying.

Theorem 2.1 (non-existence of a 4-qubit code protecting against two erasure errors)**.** *There does not exist a 4-qubit code that protects against two erasure errors.*

Proof. Suppose that we had such a code. Then, take the first two qubits and think of them as an encoding with two missing qubits. Same with the last two qubits.

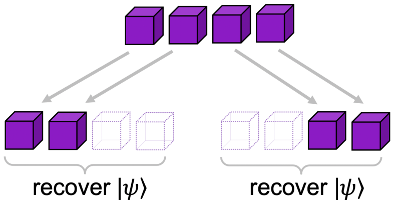

Figure 13: 4-qubit code protecting against two erasure errors violates the no-cloning theorem.

If the code was resilient against two erasure errors then we could recover the data from each of these, thereby producing two copies of the data. This would contradict the no-cloning theorem. □

3 Calderbank-Shor-Steane codes

In this section, we will begin with a quick overview of classical linear error-correcting codes, and then I'll explain how to construct good *quantum* error-correcting codes from certain classical linear codes using a method due to Calderbank, Shor and Steane. These are commonly called CSS codes.

In section 2.2, I stated results about multi-bit classical error-correcting codes, without saying anything about how these codes work. We're going to begin by looking at some of the structure of classical linear codes.

3.1 Classical linear codes

Here we consider certain block codes that encode n-bit data as m-bit codewords.

Definition 3.1 (linear code)**.** *An error-correcting code is* linear *if the the mapping from data to codewords is a linear mappping from $\{0,1\}^n$ to $\{0,1\}^m$, with respect to the field $\mathbb{Z}_2$. In particular, the set of codewords is a linear space.*

The 3-bit repetition code from section 2.1 is a linear code. In particular, the set of its codewords is $\{000, 111\}$, which is a 1-dimensional subspace of $\{0,1\}^3$.

Another example of a linear code is the code given by the table in figure 14.

data	codeword
000	0000000
001	0001111
010	0110011
011	0111100
100	1010101
101	1011010
110	1100110
111	1101001

Figure 14: A 7-bit classical error-correcting code.

This code linearly maps 3-bit strings of data into 7-bit codewords. The codewords are a 3-dimensional subspace of $\{0,1\}^7$. In fact, the three strings 1010101, 0110011, 0001111 (codewords for 001, 010, 100) are a basis for the 3-dimensional subspace. In section 3.3, I'll show you how codes that have this linear structure (and additional properties) can be converted into quantum error-correcting codes in a systematic way. And the 7-bit code in figure 14 will serve as a running example to illustrate the construction.

Although an error-correcting code is a mapping from $\{0,1\}^n$ to $\{0,1\}^m$, its error-correcting properties can be deduced from properties of it set of codewords, and we frequently refer to a code by its set of codewords.

Definition 3.2 (Hamming distance)**.** *For any two binary m-bit strings, their* Hamming distance *is defined as the number of bit positions in which they are different. That's how many bits you need to flip to convert between the two strings.*

Definition 3.3 (distance of a code)**.** *The* distance of a code *is the minimum Hamming distance between any two codewords. For linear codes, this is equivalent to the minimum distance from any non-zero codeword to the zero codeword.*

What's the minimum distance of the 7-bit code in figure 14? Since it's a linear code, it suffices to check the minimum Hamming weight of all the non-zero codewords, which is 4. So the minimum distance is 4.

The minimum distance of a classical code is closely related to its error-correcting properties.

Theorem 3.1. *If the minimum distance of a code is d then $\lfloor \frac{d-1}{2} \rfloor$ is the number of errors that can be corrected. In other words, as long as the number of bits flipped is strictly less than $\frac{d}{2}$, they can be corrected.*

Proof. First think of the subset of the m-bit strings that are the codewords, and associate with each codeword a neighbourhood that consists of all m-bit strings whose distance from that codeword is strictly less than $\frac{d}{2}$. Figure 15 is a schematic illustration of the codewords (in blue) and their associated neighborhoods (in grey).

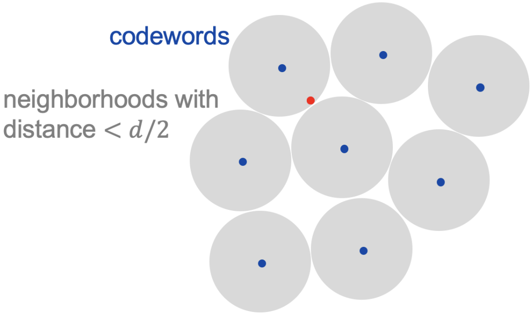

Figure 15: Neighborhoods with associated with associated with codewords.

Note that none of the neighborhoods intersect (because if they did then there would be an m-bit string whose distance from two different codewords is strictly less than $\frac{d}{2}$, violating the fact that the minimum distance is d). If fewer than $\frac{d}{2}$ bits of any codeword are flipped then the perturbed codeword stays within the same neighborhood (e.g., the red point in figure 15). Therefore, there is a unique codeword whose distance is less than $\frac{d}{2}$ from the perturbed codeword. So there is an unambiguous decoding. □

As an example, suppose that, for the code in figure 14, one bit of a codeword is flipped, resulting in the string 10001010. Can you find the unique codeword whose distance is 1 from this string?

3.1.1 Dual of a linear code

You may recall the *dot product* between binary strings $a, b \in \{0,1\}^m$ that we previously saw in the context of Simon's algorithm, which is

$$a \cdot b = a_1 b_1 + a_2 b_2 + \cdots + a_m b_m \bmod 2. \tag{39}$$

Remember that this this is not an inner product because the dot product of a nonzero vector with itself can be zero. But it has some nice properties and we can very loosely think of two strings as being "orthogonal" if their dot product is zero.

Definition 3.4 (dual code). *For any linear code whose codewords are $C \subseteq \{0,1\}^m$, define the dual code as*

$$C^\perp = \big\{a \in \{0,1\}^m \;\big|\; \text{for all } b \in C,\ a \cdot b = 0\big\}. \tag{40}$$

The set $C^\perp$ can be loosely thought as all codewords that are orthogonal to C.

The codewords of the 7-bit code in figure 14 are

$$\begin{aligned} C = \big\{&0000000, 1010101, 0110011, 1100110,\\ &0001111, 1011010, 0111100, 1101001\big\}, \end{aligned} \tag{41}$$

and the dual of that code is

$$\begin{aligned} C^\perp = \big\{&0000000, 1010101, 0110011, 1100110,\\ &0001111, 1011010, 0111100, 1101001,\\ &1111111, 0101010, 1001100, 0011001,\\ &1110000, 0100101, 1000011, 0010110\big\}. \end{aligned} \tag{42}$$

Every vector in C has dot product zero with every vector in $C^\perp$.

Note that $C^\perp$ is a superset of C (it contains all of C, plus additional strings). This can happen because the dot product is not an inner product. What's the minimum distance of $C^\perp$ in this example? The minimum distance of $C^\perp$ is 3. Note that this means that $C^\perp$ can correct against a 1-bit error.

3.1.2 Generator matrix and parity check matrix

For every linear code with n-bit data and m-bit codewords, we associate two matrices.

One matrix is the $n \times m$ *generator matrix*, G, which expresses the encoding process as a linear operator. Namely, for all $a \in \{0,1\}^n$,

$$E(a) = aG. \tag{43}$$

The convention in coding theory is that binary strings are row vectors and the matrix multiplication is on the right side.

For our running 7-bit code example, the generator matrix is

$$G = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1\\ 0 & 1 & 1 & 0 & 0 & 1 & 1\\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{44}$$

and to encode a three-bit string $a \in \{0,1\}^3$, we right multiply by the generator matrix

$$E(a) = \begin{bmatrix} a_0 & a_1 & a_2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} b_0 & b_1 & b_2 & b_3 & b_4 & b_5 & b_6 \end{bmatrix}. \tag{45}$$

It's not hard to see that the rows of G are a basis for the set of codewords.

Another interesting matrix associated with a linear code is called the $m \times (m-n)$ *parity check matrix*, H, which can be used to check whether a given string is a codeword or not. For any string $b \in \{0,1\}^m$, $b \in C$ if and only if $bH = 0^{m-n}$, the zero vector. The columns of H are a basis for $C^\perp$, the dual of C.

For our 7-bit code example, a parity check matrix is

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{46}$$

Notice that the space generated by the rows of G is orthogonal to the space generated by the columns of H. This is expressed succinctly by the fact that $GH = 0^{n \times (m-n)}$, the $n \times (m-n)$ zero matrix.

3.1.3 Error-correcting via parity-check matrix

Let $C \subset \{0,1\}^m$ be a linear code with with distance d and let H be a parity check matrix of C. Here's how to correct errors using H.

For any codeword $b \in \{0,1\}^m$, let b' be the perturbed codeword after an error has been applied. It's useful to think of an m-bit *error vector*, e, that has a 1 in each position where a bit is flipped, and a 0 in the other positions. Then we can write $b' = b + e$, where the vectors are added bitwise (mod 2). If fewer than $\frac{d}{2}$ bits of b are flipped then the Hamming weight of e is less than $\frac{d}{2}$.

Now, consider what happens if we multiply b' by the parity check matrix H

$$\begin{aligned} b'H &= (b+e)H && (47) \\ &= bH + eH && (48) \\ &= eH && (49) \end{aligned}$$

(where we are using the fact that $bH = 0$ because b is a codeword).

We call bH the *syndrome of the error* e. For linear codes, the syndrome depends only on the error, and not on the codeword on which the error occurred. This property will be especially valuable when we construct quantum error-correcting codes based on classical linear codes.

Referring back to our running example 7-bit code, here's a table of the syndromes of all the error under consideration. That is, where at most one bit is flipped.

e	eH
0000000	0000
1000000	1001
0100000	1010
0010000	1011
0001000	1100
0000100	1101
0000010	1110
0000001	1111

Figure 16: Syndromes associated with errors.

In general, all errors e that are of Hamming weight less than $\frac{d}{2}$ have unique syndromes.

Let's go through am example of an error-correction procedure using the syndrome table in figure 16. Suppose that we receive the string 1001010 from the channel (and we're not told what the error is). Multiplying by the parity check matrix, we obtain $\begin{bmatrix}1\,0\,0\,1\,0\,1\,0\end{bmatrix}H = \begin{bmatrix}1\,0\,1\,1\end{bmatrix}$, so the syndrome of the error is 1011. Looking at the syndrome table, we can see that this syndrome corresponds to the error 0010000 (i.e., the third bit is flipped). So we can flip the third bit again to correct the error, obtaining the original codeword $b = 1011010$. To get the data $a \in \{0,1\}^3$ from the codeword b, we can use the fact that $\begin{bmatrix}a_0\,a_1\,a_2\end{bmatrix}G = b$ (where G is a generator matrix for the code) and solve the system of linear equations to get a.

As an aside, for large m and where d grows as a function of m, the syndrome table can be of size exponential in m. So it is not efficient to explicitly construct the entire table of errors and syndromes. Good error-correcting codes with large block sizes are designed with special additional structural properties which enable the error as a function of the syndrome to be computed efficiently.

All this information about classical linear codes is useful for understanding the methodology of CSS codes.

3.2 $H \otimes H \otimes \cdots \otimes H$ revisited

Before discussing the CSS code construction, I'd like to show you some more nice properties of the m-fold tensor product of Hadamard gates. We've already seen in the notes [*Part II: Quantum algorithms (I)*, section 6.3.1] that

$$H^{\otimes m} \left|0^m\right\rangle = \tfrac{1}{\sqrt{2^m}} \sum_{b \in \{0,1\}^m} \left|b\right\rangle, \tag{50}$$

and, more generally, for any $w \in \{0,1\}^m$,

$$H^{\otimes m} \left|w\right\rangle = \tfrac{1}{\sqrt{2^m}} \sum_{b \in \{0,1\}^m} (-1)^{b \cdot w} \left|b\right\rangle, \tag{51}$$

where $b \cdot w$ denotes the dot-product $b_0 w_0 + b_1 w_1 + \cdots + b_{m-1} w_{m-1}$.

Now, here's a generalization of the above two equations related to states that are uniform superpositions over the elements of a linear subspace $C \subseteq \{0,1\}^m$. Applying $H^{\otimes m}$ to such a state results in an equally weighted superposition of the elements of $C^{\perp}$. Namely,

$$H^{\otimes m} \Big(\tfrac{1}{\sqrt{|C|}} \sum_{a \in C} \left|a\right\rangle \Big) = \tfrac{1}{\sqrt{|C^{\perp}|}} \sum_{b \in C^{\perp}} \left|b\right\rangle. \tag{52}$$

Also, for a uniform superposition over the elements of a linear subspace $C \subseteq \{0,1\}^m$ offset by some $w \in \{0,1\}^m$, there is a similar expression with phases,

$$H^{\otimes m} \Big(\tfrac{1}{\sqrt{|C|}} \sum_{a \in C} \left|a + w\right\rangle \Big) = \tfrac{1}{\sqrt{|C^{\perp}|}} \sum_{b \in C^{\perp}} (-1)^{b \cdot w} \left|b\right\rangle. \tag{53}$$

Notice that Eq. (52) generalizes Eq. (50), since $\{0^m\}$ is the zero-dimensional linear subspace of $\{0,1\}^m$ and $\{0^m\}^{\perp} = \{0,1\}^m$. Similarly, Eq. (53) generalizes Eq. (51).

Exercise 3.1. *Prove Eqns.* (52) *and* (53).

Hint: if G is the $n \times m$ generator matrix of C then

$$\tfrac{1}{\sqrt{|C|}} \sum_{a \in C} \left|a\right\rangle = \tfrac{1}{\sqrt{2^n}} \sum_{b \in \{0,1\}^n} \left|bG\right\rangle. \tag{54}$$

3.3 CSS codes

A CSS code is based on two classical linear codes, $C_0, C_1 \subseteq \{0,1\}^m$, that are related by these properties:

- $C_0 \subsetneq C_1$
- $C_0^\perp \subseteq C_1$.

Two relevant parameters are $k = \dim(C_1) - \dim(C_0)$ and d, the distance of code C_1. From this, we will construct a quantum error-correcting code that encodes k qubits as m qubits, and protects against any errors in fewer than $\frac{d}{2}$ qubits.

From our running example, we can take

$$\begin{aligned} C_0 = \big\{&0000000,\ 1010101,\ 0110011,\ 1100110,\\ &0001111,\ 1011010,\ 0111100,\ 1101001\big\}, \end{aligned} \tag{55}$$

$$\begin{aligned} C_1 = \big\{&0000000,\ 1010101,\ 0110011,\ 1100110,\\ &0001111,\ 1011010,\ 0111100,\ 1101001,\\ &1111111,\ 0101010,\ 1001100,\ 0011001,\\ &1110000,\ 0100101,\ 1000011,\ 0010110\big\}. \end{aligned} \tag{56}$$

Clearly $C_0 \subsetneq C_1$ and $C_0^\perp = C_1$. For this example, $k = \dim(C_1) - \dim(C_0) = 4-3 = 1$, and $d = 3$ (the distance of code C_1).

In general, let G_0 and G_1 be generator matrices for C_0 and C_1, respectively. Suppose that G_0 is an $n \times m$ matrix and G_1 is an $(n+k) \times m$ matrix. We can express

$$G_1 = \left[\begin{array}{c} G_0 \\ \hline W \end{array}\right], \tag{57}$$

where W is a $k \times m$ matrix representing the additional rows to add to G_0 in order to extend the span from C_0 to C_1.

In our running example, we have

$$G_0 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \qquad W = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{58}$$

$$G_1 = \left[\begin{array}{ccccccc} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right]. \tag{59}$$

3.3.1 CSS encoding

For linear codes $C_0 \subset C_1 \subseteq \{0,1\}^m$ such that $C_0^\perp \subseteq C_1$, let $k = \dim(C_1) - \dim(C_0)$. The related CSS code encodes a k-qubit state as an m-qubit state as follows.

For all $v \in \{0,1\}^k$, define the *logical basis state* $|v\rangle_L$ as the m-qubit state

$$|v\rangle_L = \tfrac{1}{\sqrt{|C_0|}} \sum_{a \in C_0} |a + vW\rangle \, . \tag{60}$$

Then an arbitrary k-qubit (pure) state

$$\sum_{v \in \{0,1\}^k} \alpha_v \, |v\rangle \tag{61}$$

is encoded as the m-qubit state

$$\sum_{v \in \{0,1\}^k} \alpha_v \, |v\rangle_L = \sum_{v \in \{0,1\}^k} \alpha_v \Big(\tfrac{1}{\sqrt{|C_0|}} \sum_{a \in C_0} |a + vW\rangle \Big). \tag{62}$$

Returning to our running example, we have logical qubits

$$\begin{aligned} |0\rangle_L = {} & |0000000\rangle + |1010101\rangle + |0110011\rangle + |1100110\rangle \\ & |0001111\rangle + |1011010\rangle + |0111100\rangle + |1101001\rangle \end{aligned} \tag{63}$$

$$\begin{aligned} |1\rangle_L = {} & |1111111\rangle + |0101010\rangle + |1001100\rangle + |0011001\rangle \\ & |1110000\rangle + |0100101\rangle + |1000011\rangle + |0010110\rangle \, . \end{aligned} \tag{64}$$

The state $|0\rangle_L$ is a uniform superposition of the elements of C_0. The state $|1\rangle_L$ is a uniform superposition of the elements of C_0+1111111. Any 1-qubit state $\alpha_0 |0\rangle + \alpha_1 |1\rangle$ is encoded as the 7-qubit state $\alpha_0 |0\rangle_L + \alpha_1 |1\rangle_L$. This is called the *Steane code* and we'll show that it protects against any 1-qubit error.

3.3.2 CSS error-correcting

Let $C_0 \subset C_1 \subseteq \{0,1\}^m$ be linear such that $C_0^\perp \subseteq C_1$ and let $k = \dim(C_1) - \dim(C_0)$. We will show how to perform error-correction for the related CSS code, correcting any Pauli error acting on fewer than $\frac{d}{2}$ qubits, where d is the distance of code C_1.

We begin by showing how to correct X-errors acting on fewer than $\frac{d}{2}$ qubits. The encoding of a k-qubit state is of the form of Eq. (62). This state is a superposition of basis states from the larger code C_1, whose minimum distance is d.

We can write the X-error operator as $E_e = X^{e_0} \otimes X^{e_1} \otimes \cdots \otimes X^{e_{m-1}}$, where $e \in \{0,1\}^m$ has Hamming weight less than $\frac{d}{2}$. For encoded data $|\psi\rangle$, the state $E_e |\psi\rangle$ is the encoding subjected to the error.

Based on the parity check matrix H_1 of C_1, we can create a circuit than produces the error syndrome $|S_e\rangle$ in an ancillary register.

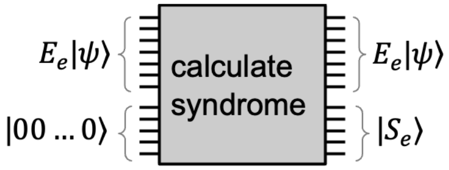

Figure 17: Computing the X-error syndrome.

Since the syndrome depends only on the error vector e and not any computational basis state from C_1, the output of the circuit is the product state $\big(E_e|\psi\rangle\big) \otimes |S_e\rangle$.

In our running example, this syndrome is computed using the parity check matrix

$$H_1 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{65}$$

and, based the entries of H_1, a circuit for producing the X-error syndrome is this.

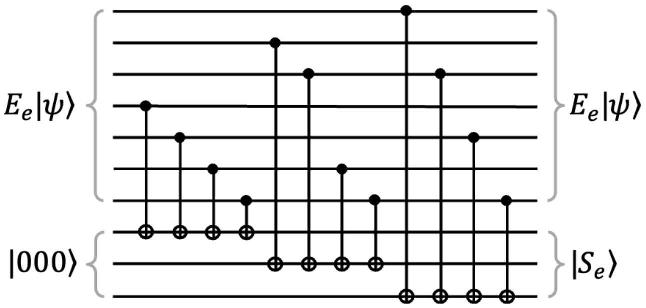

Figure 18: Explicit circuit computing the X-error syndrome for the Steane code.

It's easy to confirm that, for any computational basis state in the code word, this computes the syndrome $|S_e\rangle$ associated with error e. Once the error syndrome S_e has been computed, the error e can be deduced and undone. So this procedure corrects X-errors, as long as there are fewer than $\frac{d}{2}$ of them.

What about Z-errors? To correct against Z-errors, we apply an H operation to each qubit of the encoded data. This converts the encoding to the Hadamard basis, where Z-errors are X-errors.

What does the encoding look like in the Hadamard basis? Using the results from section 3.2, this is

$$H^{\otimes m}\left(\sum_{v\in\{0,1\}^k} \alpha_v \left|v\right\rangle_L\right) = \sum_{v\in\{0,1\}^k} \alpha_v H^{\otimes}\left(\frac{1}{\sqrt{|C_0|}}\sum_{a\in C_0}\left|a+vW\right\rangle\right) \tag{66}$$

$$= \sum_{v\in\{0,1\}^k} \alpha_v \left(\frac{1}{\sqrt{|C_0^{\perp}|}}\sum_{b\in C_0^{\perp}}(-1)^{b\cdot(vW)}\left|b\right\rangle\right). \tag{67}$$

Since $C_0^{\perp} \subseteq C_1$, this state is also a superposition of computational basis states from C_1. Therefore, we can apply the procedure in figure 17 again in the Hadamard basis.

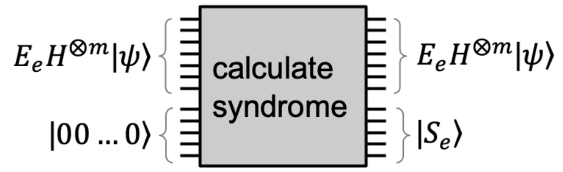
$E_eH^{\otimes m}|\psi\rangle$
$|00\ldots0\rangle$
calculate syndrome
$E_eH^{\otimes m}|\psi\rangle$
$|S_e\rangle$

Figure 19: Computing the Z-error syndrome.

From the syndrome, the Z-errors (which are X-errors in the Hadamard basis) can be undone. Then $H^{\otimes m}$ is applied again to return to the computational basis.

So far, we can correct X-errors and Z-errors. Since $Y = iXZ$, each Y-error is like an X-error and a Z-error, and each of those is corrected by the two aforementioned procedures.

3.3.3 CSS code summary

The Steane 7-qubit CSS code and the Shor 9-qubit code both protect against a 1-qubit error; however, note that the Steane code has better rate.

In general, the performance of a CSS code depends on the performance of the classical linear codes $C_0 \subset C_1 \subseteq \{0,1\}^m$ (with $C_0^{\perp} \subseteq C_1$) on which it is based. Since good classical codes of the above form exist, there exist qualitatively good quantum error-correctng codes. It turns out that there exists a threshold $\epsilon_0 = 0.055...$

such that, for the depolarizing channel with error parameter $\epsilon < \epsilon_0$, there exist CSS codes of constant rate and arbitrarily small failure probability. This is qualitatively equivalent to what's achievable with classical error-correcting codes, although the specific constants are smaller.

This gives an idea of what quantum error-correcting codes can accomplish. But this is just the beginning. There are many other quantum error-correcting codes, some of which are better in certain respects than CSS codes. Also, the depolarizing channel is a kind of standard noise model, but it's not the only one. Quantum error-correcting codes that perform well against this model tend to perform well against variations of this error model. And there is some fine-tuning possible if one knows the exact error model.

4 Very brief remarks about fault-tolerance

The error-correcting codes that I've described assume that the noise is restricted to the communication channel. They assume that the encoding and decoding processes are not subject to any noise. What about noise during the execution of a quantum circuit?

Suppose that we want to execute a quantum circuit, but there is noise *during the computation*. One simple way of modeling this is to assume that there is a depolarizing channel at each qubit applied at each time step.

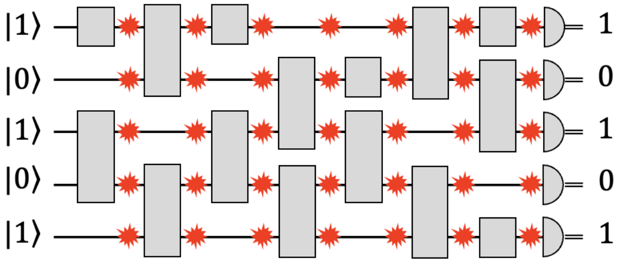

Figure 20: Noisy gates modeled by a depolorizing channel at each qubit at each time step.

How can we cope with this kind of noise? If the error parameter of the depolorizing channel ϵ is very small then this is OK. Suppose that the size of the computation is less than $\frac{1}{10\epsilon}$. Then, with good probability, none of the qubits incurs a flip (by which I mean a bit-flip, phase-flip, or both), so the computation succeeds.

But, if ϵ is a constant (dictated by the precision of our hardware), then size of the largest circuit possible will also be bounded by a constant. If we want to execute a larger circuit then we need a smaller ϵ, which means better precision in our quantum gates. For very large circuits we would need very high precision components.

But we can do much better than this, due to the celebrated *Threshold Theorem.*

Theorem 4.1 (Threshold Theorem, rough statement)**.** *There exists a constant* $\epsilon_0 > 0$ *such that if the precision per gate per time step is below* ϵ_0 *then we can perform arbitrarily large computations without having to further increase the precision.*

The rough idea is to convert the circuit that we want to perform into a another circuit that is fault-tolerant. The fault-tolerant circuit uses error-correcting codes in place of each qubit, and performs additional operations that correct errors at regular time intervals, so they don't accumulate. The known fault-tolerant constructions can be quite elaborate, and use several clever ideas. But what's nice is that the fault-tolerant circuits are not that much larger asymptotically than the original circuits. In some formulations, the size increase is by a logarithmic factor; and in some formulations, by a constant factor. The fault tolerant circuit will not be exactly of the form of a larger version of the kind of circuit that appears in Fig. 20. For the details to work out, fresh ancilla qubits must be injected into the circuit at intermediate times and also qubits must also be measured at intermediate times to perform the corrections.

This result is quite impressive, if you consider that there is noise during any encoding and decoding operation within the fault-tolerant circuit.