CS 880: Quantum Information Processing 11/09/10

Lecture 26: Error Correcting Codes

Instructor: Dieter van Melkebeek Scribe: John Gamble

Last class we began talking about quantum error correcting codes, and the difficulties that arise when trying to apply classical error correcting techniques to a quantum setting. At first, the task seems rather daunting since in a classical setting each bit has only two possible values, while on a quantum computer each qubit can take a continuum of values. Nonetheless, we were able to show that it is sufficient to be able to correct only bit and phase flips and their combination. Linearity then enabled us to correct an arbitrary error. We used this to develop a code that represented one logical qubit as nine physical qubits, and could correct an arbitrary, single-qubit error.

Today, we will examine the Calderbank-Shor-Steane (CSS) procedure for generating quantum codes based on classical codes. Using this, we will be able to find a seven physical qubit code to represent one logical qubit and correct an arbitrary, single-qubit error (an improvement over our previous nine-qubit code). Using the stabalizer formalism, it is possible to generate a five-qubit code, which is optimal, but we will not cover that here.

1 Background on Classical Codes

A *code* is a mapping $C : \{0,1\}^k \to \{0,1\}^n$ that maps information words in the set $\{0,1\}^k$ to (longer) codewords in the set $\{0,1\}^n$. The basic idea is that we are adding some redundancy to become robust to some errors. Two key properties of codes are their *rate* $\rho = k/n$ and their *relative distance* $\delta = d/n$, where d is the minimum Hamming distance between any two valid codewords. Since we do not want the encodings to be too expensive, good codes should have high rates. Also, as we would like to be able to correct many errors, we would also like high relative distances. Codes are typically denoted by (n, k) or (n, k, d).

As before, if the absolute distance $d \geq 2t + 1$, where t is the maximum number of errors that can occur, received words that come from different codewords cannot collide and we can correct the errors. More specifically, if we consider some received codeword $r = C(x) + e$, which is an encoding of the information word x with some error e, as long as $2 \cdot \text{weight}(e) + 1 \leq d$ then we can recover $C(x)$ from r simply by choosing the closest valid codeword to r.

1.1 Linear Codes

Many codes are linear, meaning that C is a linear mapping. In this case, the absolute distance d of the code is just the minimum weight of any nonzero codeword. To see this, first note that by linearity the zero vector must map to the zero vector: $C(0^k) = 0^n$, so d is at most the minimum weight of a nonzero codeword. Now, suppose that two codewords, α and β, are the closest in the code. But then, by linearity, $\gamma = \alpha - \beta$ is also a codeword. Since the distance between α and β is the smallest in the code and is also the same as the distance between γ and 0^n, which is the weight of γ, we have that d is at least the minimum weight.

Another interesting feature of linear codes is that the code generation and decoding can be done in a generic way:

1. For encoding, use $x \rightarrow Gx$, where G is an $n \times k$ generating matrix.

2. For decoding, use the $(n-k) \times n$ parity check matrix P:

$$y \in C \leftrightarrow Py = 0. \tag{1}$$

We can view P as a set of homogeneous linear equations that exactly characterize the codewords. Also, we can think of P^T as the generator matrix of the orthogonal complement of C, and its *dual code* $C^\perp$. Vectors in this dual code are orthogonal to all codewords in C, and are taken from information words of length $n-k$.

The parity check matrix cannot only be used to detect errors, but also to correct them. For instance, suppose that we had a received word $r = C(x) + e$ with error e, generated from an information word x. Then, applying the parity check matrix to r gives us $Pr = 0 + Pe$. Here, Pe is called the *error syndrome*, as it will allow us to diagnose the error. Note that for any two $e_1 \neq e_2$, as long as neither have weight more than $(d-1)/2$, we know that $Pe_1 \neq Pe_2$. If they were the same, then we would have $P(e_1 - e_2) = 0$, so $e_1 - e_2$ would be a nonzero codeword (by linearity) of weight less than d, which is a contradiction. Hence, if no more than $(d-1)/2$ errors, our syndrome tells us the location of any errors that occurred.

The distance d can also be interpreted in terms of P. Specifically, d is the smallest weight of string y such that $Py = 0$. In other words, d is the smallest nonzero number of columns of P that add up to the all zero column. Linear codes are typically denoted with square brackets: $[n, k]$ or $[n, k, d]$.

1.2 Examples of linear codes

The trivial example we mentioned last time is the repetition code, where we just repeat one bit a number of times. In that case, we are trying to encode one bit, so $k = 1$. n is the number of times we repeat. The relative distance is just $\delta = 1$, which is as good as it can be, since the only two valid codewords are all zeros or all ones. However, the rate is $\rho = 1/n$, as bad as it can be. As we mentioned in an earlier lecture, there are also families of codes where both δ and ρ are positive constants. One example is the Justesen code, but we will not develop it here.

Next, consider the Hamming codes, a class of codes defined by their parity check matrices. Take P to have length s, with columns consisting of all nonzero strings of length s:

$$P = \begin{pmatrix} 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & \vdots & \vdots & \cdots & 1 \\ \vdots & 1 & 1 & \cdots & \vdots \\ 1 & 0 & 1 & \cdots & 1 \end{pmatrix}. \tag{2}$$

P has dimension $s \times (2^s - 1)$. Since P operates on codewords of length n, we know $n = 2^s - 1$. Also, since P has $n - k$ rows, $k = 2^s - 1 - s$. Since it takes a linear combination of three columns of P to form the zero vector, the code has distance 3. Hence, this is a $[2^s - 1, 2^s - 1 - s, 3]$ code, and can correct a single error. Further, determining where the error occurred is very easy. This is because for any e of weight one, $Pe = i$, where i is the binary representation of the location of the error.

A common instantiation of this is for $s = 3$, where we obtain $H_3 = [7, 4, 3]$ with dual code $H_3^\perp[7, 3]$. One can also show that in this case $H_3^\perp \subseteq H_3$.

Exercise 1. *Verify that $H_3^\perp \subseteq H_3$.*

2 CSS Codes

CSS codes are a family of codes generated by a procedure for translating classical codes to quantum codes.

Theorem 1. *Suppose we have a (classical) $[n, k_1]$-code C_1 with distance $d(C_1) \geq 2t - 1$ and a classical $[n, k_2]$-code $C_2 \subseteq C_1$ such that the distance of the dual code $C_2^\perp$ satisfies $d\left(C_2^\perp\right) \geq 2t - 1$. Then, there exists a quantum code $\mathrm{CSS}(C_1, C_2)$ that maps $k_1 - k_2$ logical qubits to n physical qubits that can correct arbitrary errors on t qubits.*

As a specific example of this, pick $C_1 = C_2^\perp = H_3$. Then, $k_1 - k_2 = 4 - 3 =$, $t = 1$, and $n = 7$, which is a seven-qubit code that encodes a single logical qubit and can correct an arbitrary error on one qubit.

Proof. First, note that it is enough to specify the encoding of base states, as we can then generate an arbitrary information word from a linear combination of base states. The basis for the codewords will all be of the form

$$|C_2 + x\rangle = \frac{1}{\sqrt{|C_2|}} \sum_{y \in C_2} |y + x\rangle \,, \tag{3}$$

where $x \in C_1$. So, our codewords are coset states of C_2. Two of these states $|C_2 + x_1\rangle$ and $|C_2 + x_2\rangle$ are distinct when x_1 and x_2 belong to distinct cosets of C_1 in C_2, i.e., iff their difference $x_1 - x_2 \in C_2$. Thus, the number of available codewords is $2^{k_1}/2^{k_2} = 2^{k_1 - k_2}$. The question now is how we correct errors on these encodings.

We will proceed as we did before, by showing that we can correct bit flips, phase flips, and their combination, which is sufficient to correct arbitrary errors by linearity. We begin with bit flips, supposing that we have a state $|C_2 + x + e\rangle$. where e is a bit flip error. Note that since we required that $C_2 \subseteq C_1$, for each $y \in C_2$, $y + x$ is also a codeword in in C_1. Hence, on each component of the superposition of our coset state, we can correct for e as long as $\mathrm{weight}(e) < t$, since C_1 can correct t errors. By linearity, this means that we can correct up to t bit-flip errors on $|C_2 + x\rangle$. Note that, as before, this procedure does not affect phase flips.

Next, we look at phase flip errors. Recall that in the Hadamard domain, phase flips become bit flips. Referring to our discussion of the Fourier transform of a coset state and the fact that $H^{\otimes n}$ is the Fourier transform over $\mathbb{Z}_2^n$, we obtain the following for the Fourier transform of a valid codeword:

$$H^{\otimes n} |C_2 + x\rangle = F_{\mathbb{Z}_2^{\otimes n}} |C_2 + x\rangle = \frac{1}{\sqrt{\left|C_2^\perp\right|}} \sum_{y \in C_2^\perp} \chi_y(x) |y\rangle \,, \tag{4}$$

where in this case the character $\chi_y(x) = (-1)^{x \cdot y}$. If we apply this to a superposition of encoded states, we have

$$H^{\otimes n} \left(\sum_{x \in C_1} \alpha_x |C_2 + x\rangle \right) = \sum_{y \in C_2^\perp} \left(\sum_{x \in C_1} \frac{\alpha_x}{\sqrt{\left|C_2^\perp\right|}} \chi_y(x) \right) |y\rangle \,. \tag{5}$$

If we have some phase flips before the Hadamard transform, then they map to bit flips after the Hadamard transform, i.e., we replace $|y\rangle$ on the right-hand side of (5) by $|y+e\rangle$, where e indicates the positions of the phase flips. As long as e has weight at most t, then we can correct the errors, since $C_2^\perp$ was required to be able to correct t errors. Next, we apply the Hadamard transform again, which brings us back to a valid encoding.

Since phase flips are not affected by the bit flip correction, applying the bit flip error correction procudure followed by the phase flip error correction procedure allows us to correct bit flips, phase flips, and combined bit/phase flips on t qubits. By the linearity argument from last lecture, this means the procedure corrects arbitrary errors on up to t qubits. □

As it turns out, the simple nine-qubit code we developed last time can also be viewed as a CSS code through use of classical repetition codes.

Exercise 2. *Show that the nine-qubit code we developed last time can be expressed using the CSS formalism.*