Error Correcting Codes: Combinatorics, Algorithms and Applications **(Fall 2007)**

Lecture 2: Error Correction and Channel Noise

August 29, 2007

Lecturer: Atri Rudra *Scribe: Yang Wang & Atri Rudra*

As was mentioned in the last lecture, the fundamental tradeoff we are interested in for this course is the one between the amount of redundancy in the code vs. the number of errors that it can correct. We defined the notion of rate of a code to capture the amount of redundancy. However, before we embark on a systematic study of the tradeoff above, we need to formally define what it means to correct errors. We do so next.

1 Error correction

Before we define what we mean by error correction, we formally define the notion of *encoding*.

Definition 1.1 (Encoding function). *Let $C \subseteq \Sigma^n$. An equivalent description of the code C is by an injective mapping $E : [|C|] \to \Sigma^n$ called encoding function.*

Next we move to error correction. Intuitively, we can correct a received word if we can recover the transmitted codeword (or equivalently the corresponding message). This "reverse" process is achieved by *decoding*.

Definition 1.2 (Decoding function). *Let $C \subseteq \Sigma^n$ be a code. A mapping $D : \Sigma^n \to [|C|]$ is called a decoding function for C.*

The definition of a decoding function by itself does not give anything interesting. What we really need from a decoding function is that it recovers the transmitted message. This notion is captured next.

Definition 1.3 (Error Correction). *Let $C \subseteq \Sigma^n$ and let $t \geq 1$ be an integer. C is said to be t-*error-correcting *if there exists a decoding function D such that for every error message $m \in [|C|]$ and error pattern e with at most t errors, $D(C(m) + e) = m$.*

Figure 1 illustrates how the definitions we have examined so far interact.

We will also very briefly look at a weaker form of error recovery called *error detection*.

Definition 1.4 (Error detection). *Let $C \subseteq \Sigma^n$ and let $t \geq 1$ be an integer. C is said to be t-*error-detecting *if for every message m and every error pattern e with at most t errors, there is a procedure that can decide if $(C(m) + e) \in C$.*

Note that a t-error correcting code is also a t-error detecting code (but not necessarily the other way round). Although error detection might seem like a weak error recovery model, it is useful in settings where the receiver can ask the sender to re-send the message. For example, error detection is used quite heavily in the Internet.

With the above definitions in place, we are now ready to look at the error correcting capabilities of the codes we looked at in the previous lecture.

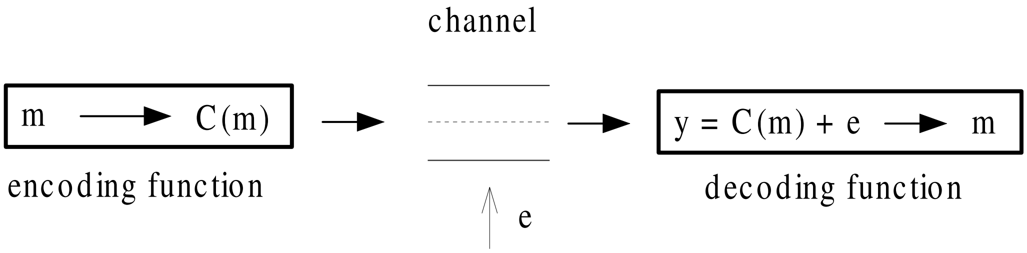

Figure 1: Coding process

2 Parity and Repetition codes

In the last lecture, we looked at examples of parity code and repetition code, with the following properties:

$$C_{\oplus} : q = 2, k = 4, n = 5, R = 4/5.$$

$$C_{3,rep} : q = 2, k = 4, n = 12, R = 1/3.$$

We will start with the repetition code. To study its error correcting capabilities, we will consider the following natural decoding function. Given a received word $y \in \{0, 1\}^{12}$, divide it up into four consecutive blocks (y_1, y_2, y_3, y_4), where every block consists of three bits. Then for every block y_i ($1 \leq i \leq 4$), output the majority bit as the message bit. Now we claim that this decoding function can correct any error pattern with at most 1 error. This is easy to verify. For example, if a block of 010 is received, we know the original message bit should be 0 because there are two 0s. In other words, we have argued that

Proposition 2.1. *$C_{3,rep}$ is a 1-error correcting code.*

However, it is not too hard to see that $C_{3,rep}$ cannot correct 2 errors. For example, if both the errors happen in the same block and a block in the received word is 010 then the original block in the codeword could have been either 111 or 000. Thus no decoder can successfully recover the transmitted message in this case. (Recall we are assuming that the decoder has no side information about the transmitted message.)

Thus, we have pin-pointed the error correcting capabilities of the $C_{3,rep}$ code: it can correct 1 error but no more. However, note that the argument assumed that the error positions can be located arbitrarily. In other words, we are assuming that the channel noise behaves arbitrarily (subject to a bound on the total number of errors). Obviously, we can model the noise differently. We now briefly digress to look at this issue in slightly more detail.

2.1 Digression: Channel Noise

As was mentioned above, until now we have been assuming the following noise model that was studied by Hamming [2]:

> Any error pattern can occur during transmission as long as the total number of errors is bounded. Note that this means that the location as well as the nature[1] of the errors are arbitrary.

We will frequently refer to Hamming's model as the Adversarial Noise Model.

We could also have following error model.

> No more than 1 error can happen in any contiguous 3 bit block.

First note that for the channel model above, no more than 4 errors can occur when a codeword in $C_{3,rep}$ is transmitted. Second, note that the decoding function that takes a majority vote in each block, can always successfully recover the transmitted codeword for *any* error pattern (while in the worst case noise model it could only correct at most 1 error). This channel model is admittedly a bit contrived but it illustrates the point that the error correcting capabilities of a code (and a decoding function) crucially depends on the noise model.

An alternate way to model the noise than Hamming's way is to model the channel as a stochastic process. As a concrete example, we briefly mention the *binary symmetric channel with crossover probability* $0 \leq p \leq 1$, denoted by BSC_p which was first studied by Shannon [1]. In this model when a (binary) codeword is transferred through the channel, every bit flips independently with probability p. We note that we only need to consider p in the range $[0, 1/2]$.[2]

Note that the two noise models proposed by Hamming and Shannon are in some sense two extremes: Hamming's model assumes *no* knowledge about the channel (except that a bound on the total number of errors is known) while Shannon's noise model (for example BSC_p) assumes *complete* knowledge about how noise is produced in the channel. In this course, we will consider only these two extreme noise models. In real life, the situation sometimes is in the "middle." If you are interested in such channels, one such hybrid model called *Arbitrary Varying Channel* is one of the suggested project topics.

We now return back to $C_\oplus$ and look at its error correcting capabilities in the worst case noise model. We claim that $C_\oplus$ cannot correct even one error. Suppose 01000 is the received word, then we know that an error has occurred but we do not know which bit was flipped. The two codewords 00000 and 01001 can both give rise to the received word 01000 when exactly one bit has been flipped. As we are assuming that the receiver has no side information about the transmitted codeword, no decoder can know what the transmitted codeword was.

References

[1] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(2):379–423 and 623–656, July and October, 1948.

[1] For binary codes, there is only one kind of error: a bit flip. However, for codes over a larger alphabet, say $\{0, 1, 2\}$, 0 being converted to a 1 or 2 are both errors but are different kinds of errors.

[2] Assume we could do error-correction on BSC_p for every $0 \leq p \leq 1/2$. Then for any $p > 1/2$, just flip the bits of the received word and then we can assume that the channel is the BSC_{1-p}, which by our hypothesis, we can already handle (as $1 - p < 1/2$).

[2] R. W. Hamming. Error Detecting and Error Correcting Codes. *Bell System Technical Journal*, 29(2):147-160, April, 1950.