

Lecture of January 9th, 2007

Scribe: Frank Kschischang

## 1 Motivation

All information is *encoded* in some fashion. The words on this page, the sound of speech, the colour of traffic signals all encode information in different ways. Such encodings may serve a variety of purposes; for example, encryption or compression. In this course, the purpose of encoding will be for reliable transmission, to permit the decoder to make reliable decisions about the transmitted messages, even after reception via a noisy channel. For example, even though some letters in the previous word have been corrupted by noise, the natural redundancy of written English provides a means for error correction. The central aim of coding theory is to discover similar encoding schemes for digital messages, and effective decoding schemes that recover the message from the noise-corrupted received signal.

Coding theory finds a theoretical motivation in information theory. According to Shannon's celebrated channel coding theorem, reliable (i.e., virtually error-free) information transfer is possible over a channel at rates up to the channel *capacity*, and, conversely, only unreliable information transfer is possible at rates larger than capacity. In general it is necessary to *encode* the transmitted information to approach the capacity of a given channel.

The field of coding theory began with the work of Hamming in the late 1940s. Hamming worked with early error-prone computers, and once observed:

If the machine can *detect* an error, why can't it locate the position of the error and *correct* it?

Thus, the motivation for coding theory comes from the need to protect data transmission systems against channel noise, enabling the detection and correction of *errors* that inevitably arise in transmission. (We will refine this motivation below.)

## 2 Some Basic Definitions

Let  $A$  be a finite set, and denote the number of elements in  $A$  by  $|A|$ . In coding theory, a finite set is often referred to as an *alphabet*. A *binary* alphabet has just two elements, typically  $\{0, 1\}$ . Likewise a *ternary* alphabet has 3 elements, a *quaternary* alphabet has 4 elements; in general, an  $M$ -ary alphabet has  $M$  elements. The set of  $n$ -tuples over  $A$ , i.e., the  $n$ -fold Cartesian product of  $A$  with itself, is denoted by  $A^n$ . Thus

$$A^n = \{(a_1, a_2, \dots, a_n) : a_1, a_2, \dots, a_n \in A\}.$$

- ★ **Definition:** A *code of block length  $n$*  over an alphabet  $A$  is a nonempty subset of  $A^n$ . The elements of a code are referred to as *codewords*.

For example the code  $\mathbf{C} = \{(0, 0, 0), (1, 1, 1)\}$  is a code over the binary alphabet  $\{0, 1\}$  with two codewords.

### 3 Channel Models and Capacity

A general block diagram for a digital transmission system is shown in Fig. 1, showing the path of a message  $m$  through an encoder, a channel, and a decoder, to the destination.

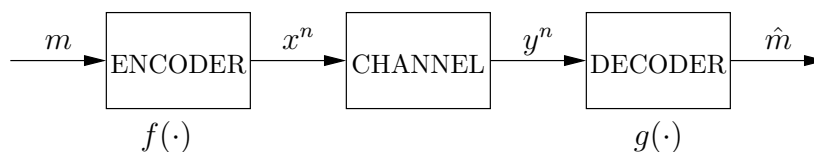


Figure 1: A communication system showing a message  $m$  which is encoded to a codeword  $x^n = f(m)$ , transmitted over a channel resulting in received word  $y^n$ , which is decoded to  $\hat{m} = g(y^n)$ .

#### 3.1 The Encoder

In digital transmission systems, the transmitted message  $m$  is drawn from some finite set  $\Omega = \{1, 2, \dots, M\}$  of possible messages.

The encoder implements a mapping  $f$  (called an *encoding function*) from  $\Omega$  to a set of inputs compatible with the channel. We will assume that the channel has input alphabet  $\mathcal{X}$ . In a block-oriented scheme, the encoding function  $f$  is from  $\Omega$  to a vector  $x^n$  of input symbols of a fixed length  $n$ , i.e., an element of  $\mathcal{X}^n$ . The range of  $f$ , i.e., the image of  $\Omega$  under  $f$ , is the set  $\mathbf{C} = \{f(1), f(2), \dots, f(M)\}$ ; clearly this is a code of length  $n$  over  $\mathcal{X}$ . Provided that  $f$  is one-to-one (which is always the case in practice), then  $\mathbf{C}$  has  $M$  codewords.

An important parameter of a code is its rate. If, for example,  $M = 4$ , each message could represent one of the four possible combinations of 2 bits, i.e., 00, 01, 10, or 11; each codeword requires  $n$  channel uses, hence the code rate is  $2/n$  bits per channel use. Similarly if  $M = 8$ , each message could represent one of 000, 001,  $\dots$ , 111; hence the code rate is  $3/n$  bits per channel use. In general, we have the following definition.

- ★ **Definition:** The *rate*, measured in bits per channel use, of a code  $\mathbf{C}$  of length  $n$  is defined as

$$R(\mathbf{C}) = \frac{1}{n} \log_2 |\mathbf{C}| \text{ bits per channel use.}$$

Sometimes, if  $|\mathcal{X}| = q > 2$ , instead of taking a logarithm to base 2 in the rate, a logarithm to the base  $q$  is used instead. In this case, only the units of rate change, i.e.,  $R(C)$  may be expressed in terms of  $q$ -ary symbols per channel use.

★ **Definition:** The *rate*, measured in  $q$ -ary symbols per channel use, of a code  $\mathbf{C}$  of length  $n$  is defined as

$$R(\mathbf{C}) = \frac{1}{n} \log_q |\mathbf{C}| \text{ } q\text{-ary symbols per channel use.}$$

(Here it is assumed that  $q$  is an integer with value at least 2.)

### 3.2 The Channel

A communication channel is modelled mathematically via a probability law that associates the channel inputs  $x \in \mathcal{X}$  to the channel outputs  $y \in \mathcal{Y}$ . If  $\mathcal{Y}$  is a discrete alphabet, then, for each possible channel input,  $x$ , we may specify the probability,  $p(y|x)$ , that a channel output,  $y$ , is observed by the receiver given that  $x$  is transmitted. Likewise if  $\mathcal{Y}$  is continuous then, for each possible channel input,  $x$ , we may specify the conditional probability density function  $p_{y|x}(y|x)$  that governs the distribution of the channel output  $y$  given that  $x$  is transmitted. Each transmission of a symbol  $x$  is regarded as a *use* of the channel.

If the channel is used repeatedly (e.g.,  $n$  times as in the transmission of a codeword), then for each input  $n$ -tuple  $x^n = (x_1, \dots, x_n)$ , we must specify the probability  $p(y^n|x^n)$  that a channel output  $y^n = (y_1, \dots, y_n)$  is observed by the receiver.

A channel is *memoryless* if the  $i$ th output symbol  $y_i$  depends on the  $i$ th channel input, but not on other inputs and outputs of the channel, i.e., a channel is memoryless if, for all  $i$ ,

$$p(y_i|x_1, \dots, x_n, y_1, \dots, y_{i-1}, y_{i+1}, \dots, y_n) = p(y_i|x_i).$$

It follows that

$$p(y^n|x^n) = \prod_{i=1}^n p(y_i|x_i)$$

for a memoryless channel. Though not every communication channel is well-modelled as being memoryless, we will study only memoryless channels in this course.

### 3.3 The Decoder

The decoder implements a mapping  $g$  (called a *decoding function*) from the set  $\mathcal{Y}^n$  of possible received vectors to an element  $\hat{m}$  of  $\Omega' = \{0, 1, 2, \dots, M\}$ . Notice that, compared with  $\Omega$ ,  $\Omega'$  has the extra element “0,” which is used to designate a decoding failure, i.e., the event that the decoder cannot or chooses not to make a

decision about the received word. For example, an error-detecting decoder would output “0” whenever the received vector  $y$  is not a valid codeword.

An *error* occurs whenever the output  $\hat{m}$  of the decoder is not equal to  $m$ . An important figure of merit for such a system is the probability of error, i.e.,  $P[\hat{m} \neq m]$ . In many situations, we would like to implement a system for which  $P[\hat{m} \neq m]$  is as small as practically possible.

### 3.4 Capacity

It is interesting to know what conditions should be satisfied in order for the probability of error to approach zero. For example, should the code rate approach zero in order to have an arbitrarily small probability of error? The surprising answer to this question is no. Shannon’s famous 1948 paper, “A Mathematical Theory of Communication,” (which founded the field of information theory) established fundamental limits on the rate of communication. In this paper, Shannon showed that there is a specific upper rate (called the *channel capacity*) associated with every discrete memoryless channel. Channel capacity is usually denoted with the letter  $C$  (not to be confused with the boldface  $\mathbf{C}$  used to denote a code). Provided only that the code rate  $R$  satisfies the condition that  $R < C$ , Shannon proved the existence of a sequence of codes of increasing block length  $n$  which can be decoded with an error probability that approaches zero as  $n \rightarrow \infty$ . Furthermore, Shannon proved that every family of codes whose error probability approaches zero must have  $R < C$ .

For example, in the *binary symmetric channel* (BSC) model, the channel input,  $x$ , and channel output,  $y$ , are drawn from a binary alphabet, typically  $\{0, 1\}$ . The BSC’s probability law is given by

$$p(y|x) = \begin{cases} 1 - p & \text{if } y = x, \\ p & \text{if } y \neq x, \end{cases}$$

where the parameter  $p$  is known as the *crossover probability* associated with the channel.

The capacity of the BSC is given by

$$C_{\text{BSC}}(p) = 1 + p \log_2(p) + (1 - p) \log_2(1 - p).$$

This function is plotted in Fig. 2.

For example,  $C_{\text{BSC}}(0.11) \approx 0.5$ . This means that, in order to achieve error-free transmission over a binary symmetric channel with an 11% error rate, the code rate cannot be larger than  $1/2$ . The goal of the coding theorist is to find families of codes that can be decoded with practical algorithms, achieving small error probabilities when the channel error probability is  $p$ , and whose rate approaches  $C_{\text{BSC}}(p)$ .

This gives us a more refined goal for coding theory. Rather than simply “to protect data transmission systems against channel noise” as stated earlier, we would

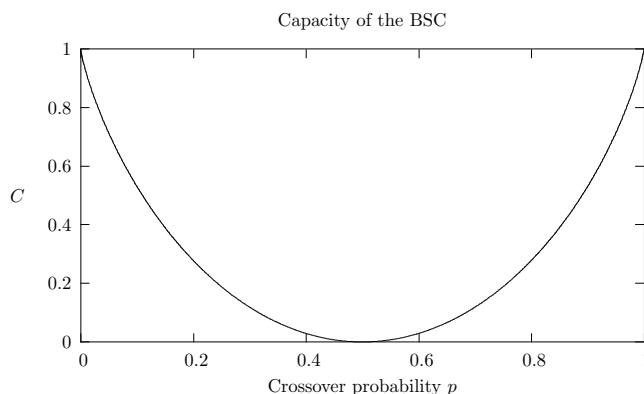


Figure 2: Channel capacity of the binary symmetric channel.

like to do so *efficiently*, i.e., to design data transmission systems with practical decoding algorithms that come close to achieving channel capacity.

The BSC may be generalized to the  $Q$ -ary symmetric channel for any  $Q \geq 2$ . The channel inputs and outputs are drawn from a finite alphabet of  $Q$  elements, for example  $\{0, 1, \dots, Q - 1\}$ , and the channel's probability law is given by

$$p(y|x) = \begin{cases} 1 - p & \text{if } y = x, \\ p/(Q - 1) & \text{if } y \neq x. \end{cases}$$

Two other channels of interest in this course are the binary erasure channel and the additive white Gaussian noise channel. For reference, these channel models are defined in the Appendices.

## 4 The (7,4) Binary Hamming Code

To get a sense of one of the directions we will be pursuing in this course, consider Fig. 3. The three overlapping circles in the figure generate seven regions, labelled with the symbols  $u_1, \dots, u_4$  and  $p_1, \dots, p_3$ , where each label stands for a bit, i.e., an element of the set  $\{0, 1\}$ . A valid *codeword* in the (7,4) Hamming code is any configuration of bits  $(u_1, \dots, u_4, p_1, \dots, p_3)$  with the property that the *parity* (i.e., the number of ones) in each circle of Fig. 3 is even. There are many ways to determine such configurations; in fact,  $u_1, \dots, u_4$  can be chosen arbitrarily, but once values have been assigned to them,  $p_1, p_2$ , and  $p_3$  are determined. Thus there are  $2^4 = 16$  valid configurations, three of which are shown in Fig. 4. The code is referred to as a (7,4) code, since each valid codeword has 7 bits, out of which 4 bits can be chosen freely (while the remaining 3 bits are functions of these 4).

Suppose a Hamming codeword is transmitted over a binary channel, which we

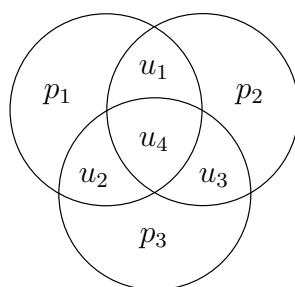


Figure 3: The parity bits  $p_1, p_2, p_3$  are chosen to make the *parity* (the number of ones) in each circle an even number.

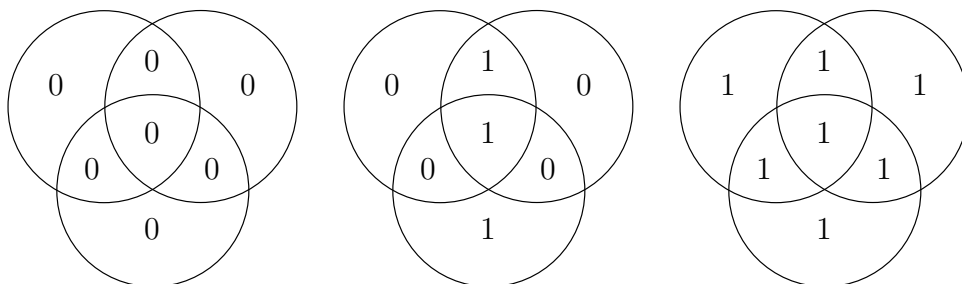


Figure 4: Three different Hamming codewords.

model as an *adversary* who is permitted to alter *at most* one of the seven bits in the codeword. Can the action of the adversary be detected and corrected? Yes! The receiver just needs to check the parity in each circle. To see this consider the following cases:

1. the adversary does not flip any bits. In this case, the receiver will find even parity in each circle and accept the received word as a valid codeword.
2. the adversary flips  $p_1$  or  $p_2$  or  $p_3$ . In this case, the receiver will find odd parity in exactly one circle, and the error can be corrected by changing just one of  $p_1$  or  $p_2$  or  $p_3$  (the one in the odd-parity circle).
3. the adversary flips  $u_1$  or  $u_2$  or  $u_3$ . In this case, the receiver will find odd parity in exactly two circles, and the error can be corrected by changing just one of  $u_1$  or  $u_2$  or  $u_3$  (the one in the intersection of the two odd-parity circles).
4. the adversary flips  $u_4$ . In this case, the receiver will find odd parity in all three circles, and the error can be corrected by changing  $u_4$ .

Since every error pattern that can be created by the adversary can be corrected, we see that the (7,4) Hamming code is single-error-correcting.

The code is not double-error-correcting however. If we permit the adversary to make two errors, the decoding algorithm given above will fail. For example, suppose the adversary flips  $u_1$  and  $p_3$ . In this case, the decoder will find odd parity in all three circles. According to the algorithm given above, the decoder will change  $u_4$  to correct this situation, thus introducing a third error! In fact, as the reader may verify, the decoder will *always* introduce a third error when presented with a codeword that has been corrupted in two distinct positions. This tells us that error-correction does not always help in reducing the error rate of the channel: if the channel error rate is too high, the decoder may become “confused” and produce an output that is even “noisier” than the output of the channel!

## 5 Hamming Distance

Recall that a *metric* (or distance function) defined on a set  $\mathcal{X}$  is a function  $d : \mathcal{X} \times \mathcal{X} \rightarrow [0, \infty)$  that satisfies for all  $x, y, z \in \mathcal{X}$

1. non-negativity:  $d(x, y) \geq 0$ , with  $d(x, y) = 0$  if and only if  $x = y$ ;
2. symmetry:  $d(x, y) = d(y, x)$ ;
3. the triangle inequality:  $d(x, y) \leq d(x, z) + d(z, y)$ .

In this course, the so-called *Hamming distance*, defined on  $n$ -tuples over alphabets, plays a pivotal role.

- ★ **Definition:** The *Hamming distance* between two  $n$ -tuples  $x$  and  $y$  over the alphabet  $A$ , i.e., between two elements  $x, y \in A^n$ , is equal to the number of coordinate positions in which  $x$  and  $y$  are different.

For example, if  $A = \{0, 1\}$ , we have

$$d((0, 0, 0), (0, 1, 1)) = 2, \quad d((0, 1, 0), (1, 0, 1)) = 3, \quad d((1, 1, 0), (1, 1, 0)) = 0.$$

- ★ **Theorem:** Hamming distance is a metric.

*Proof:* left as an exercise.

If  $x$  is transmitted and  $y$  is received, then  $d(x, y)$  is equal to the number of errors that occurred in transmission. In a sense, Hamming distance provides a measure of dissimilarity between  $n$ -tuples: the greater the Hamming distance between  $x$  and  $y$ , the greater their dissimilarity. In many channels (particularly the binary symmetric channel with crossover probability  $p < 1/2$ ), the greater is the Hamming distance between  $x$  and  $y$ , the less likely is  $y$  to be received when  $x$  is transmitted.

An important parameter of a code is the minimum Hamming distance between its codewords.

- ★ **Definition:** The *minimum Hamming distance* of a code  $\mathbf{C}$  is

$$d_{\min}(\mathbf{C}) = \min\{d(x, y) : x, y \in \mathbf{C}, x \neq y\}.$$

For example,

$$\begin{aligned} d_{\min}(\{(0, 0, 0), (1, 1, 1)\}) &= 3, \\ d_{\min}(\{(0, 0, 0), (0, 1, 1), (1, 0, 1), (1, 1, 0)\}) &= 2. \end{aligned}$$

Suppose we are given a channel whose input alphabet and output alphabet are the same. In this case, there is a well-defined notion of Hamming distance between channel inputs and channel outputs, and the concept of a minimum distance decoder is well defined.

- ★ **Definition:** Let a distance measure  $d(x, y)$  between channel input  $n$ -tuples  $x$  and channel output  $n$ -tuples  $y$  be given. Given a channel output  $y$  a *minimum distance decoder* for a code  $\mathbf{C}$  returns a codeword  $x \in \mathbf{C}$  such that  $d(x, y) \leq d(z, y)$  for all  $z \in \mathbf{C}$ .

More plainly, given an  $n$ -tuple  $y$ , a *minimum distance decoder* returns a valid codeword *nearest* to  $y$ . Thus, one way to specify a decoding function  $g$  is to specify a minimum distance decoder. A minimum distance decoder is always complete.

For example, if  $\mathbf{C} = \{(0, 0, 0), (1, 1, 1)\}$ , then  $g(0, 0, 0) = g(0, 0, 1) = g(0, 1, 0) = g(1, 0, 0) = (0, 0, 0)$  and  $g(1, 1, 1) = g(1, 1, 0) = g(1, 0, 1) = g(0, 1, 1) = (1, 1, 1)$ .

- ★ **Definition:** Let a distance measure  $d(x, y)$  between channel input  $n$ -tuples  $x$  and channel output  $n$ -tuples  $y$  be given. Given a channel output  $y$  a *bounded distance*

*decoder* with error-correcting radius  $t$  returns a codeword  $x \in \mathbf{C}$  such that  $d(x, y) \leq t$  if such an  $x$  exists; otherwise it declares a decoding failure.

For example, setting  $t = 0$  yields an error-detecting decoder: the decoder will declare a decoding failure whenever the received word  $y$  is not a valid codeword. In general, a bounded distance decoder is incomplete.

## Appendices

### A The binary erasure channel

In the *binary erasure channel* (BEC) model, the channel input,  $x$ , is drawn from the binary set  $\{0, 1\}$ , and the channel output,  $y$ , is drawn from the ternary set  $\{0, 1, *\}$ . The BEC's probability law is given by

$$p(y|x) = \begin{cases} 1 - \epsilon & \text{if } y = x, \\ \epsilon & \text{if } y = *, \\ 0 & \text{otherwise.} \end{cases}$$

The symbol  $*$  is called an *erasure*, and the parameter  $\epsilon$  is the *erasure probability* of the channel. In this channel model, no bit “errors” (i.e., bit flips from 0 to 1 or *vice versa*) are possible; the only ambiguity in the channel output arises from the presence of erased symbols.

The capacity of the BEC is given by

$$C_{\text{BEC}}(\epsilon) = 1 - \epsilon \text{ bits per channel use,}$$

as plotted in Fig. 5.

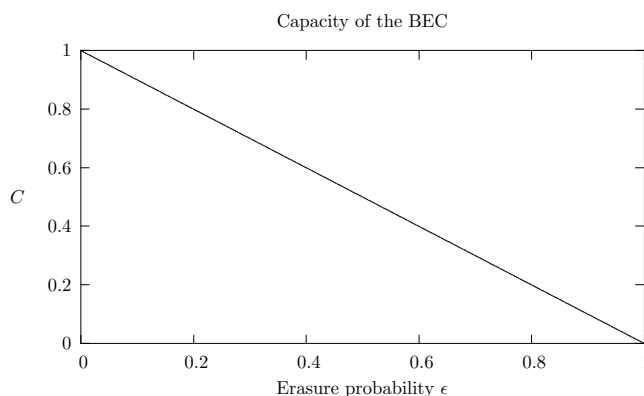


Figure 5: Channel capacity of the binary erasure channel.

The obvious  $2^m$ -ary generalization of this channel often represents a suitable model for transmission of  $m$ -bit packets in a data network: packets that arrive at their destination are taken to be correct, while packets that do not arrive are taken to be erased.

## B The additive white Gaussian noise channel

In the *additive white Gaussian noise* (AWGN) channel model, the channel input,  $x$ , and output,  $y$ , are drawn from the set  $\mathbb{R}$  of real numbers, with  $y = x + n$ , where  $n$  is a zero-mean Gaussian random variable with variance  $\sigma^2$ . It follows that in the AWGN channel's probability law is given as

$$p_{y|x}(y|x) = (2\pi\sigma^2)^{-1/2} \exp(-(y-x)^2/2\sigma^2).$$

Usually the channel input  $x$  (regarded as a random variable) must satisfy the average power constraint  $E[x^2] \leq P$ . The capacity of the AWGN channel under this constraint is given by

$$C_{\text{AWGN}}(\text{SNR}) = \frac{1}{2} \log_2(1 + \text{SNR}) \text{ bits per channel use,}$$

where the SNR (the “signal-to-noise ratio”) is defined as  $P/\sigma^2$ . This function is plotted in Fig. 6.

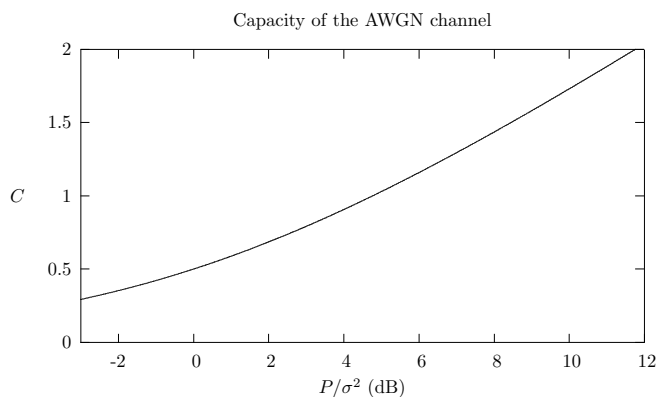


Figure 6: Channel capacity of the binary erasure channel.