ECE 7670
Lecture 1 – Introduction

Objective: To introduce fundamental concepts in digital communications and motivate the need for error correction. Also, to introduce a simple error correction code.

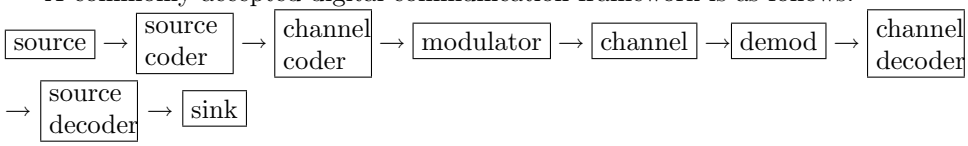
Introduction

Digital communication is becoming increasingly important in the new communications infrastructures that are being developed. It is interesting to note some of the advantages that digital offers over conventional analog communications:

1. Channel coding theorem (reliability)
2. Flexibility (encryption, interleaving, etc.)
3. Compression
4. Spectral shaping
5. Performance assessment

There is substantially more to digital communication than simply connecting two digital chips together! "Bits" are an abstract concept, and bits must be represented by voltages. Every signal must ultimately be a continuous-time waveform, and we want to design our system so that the waveform "fits" well with the channel.

A commonly-accepted digital communication framework is as follows:



- The **source** is the source of (digital) data.
- The **source encoder** serves the purpose of removing as much redundancy as possible from the data. This is the **data compression** portion.
- The **channel coder** puts a modest amount of redundancy back in order to do **error detection or correction.**
- The **channel** is what the data passes through, possibly becoming corrupted along the way. There are a variety of channels of interest, including:
  - The magnetic recording channel
  - The telephone channel
  - Other bandlimited channels
  - The multi-user channel
  - Deep-space channels
  - Fading and/or jamming and/or interference channels
  - The genetic representation channel
  - Any place where there is the possibility of corruption in the data
- The **channel decoder** performs error correction or detection

- The **source decoder** undoes what is necessary to get the data back.

There are also other possible blocks that could be inserted into this model:

- There is always the need to perform some kind of synchronization, on carrier, on symbols, on frames, etc.
- A block to enforce channel-constraints. Some channels (e.g., the magnetic recording channel) have constraints on how long a run of zeros or ones can be. The constraints are enforced in what is often known as a *line coder*.
- A block to perform encryption/decryption.
- A block to perform lossy compression.

In this class we focus on the **channel encoder** and the **channel decoder**. The goal is to obtain coders which are effective (correct lots of errors) with a reasonable complexity.

The motivation for this area of study comes from Shannon's **channel coding theorem**, which states (roughly): every channel has a capacity C (bits/sec) such that whenever the transmission rate R is less than C, the probability of error can be made arbitrarily small. However, the proof does not explain how to construct the codes. For transmission over a channel with bandwidth W with Gaussian noise at a SNR of E_s/N_0, the capacity is

$$C = W \log_2(1 + \frac{E_s}{N_0}) \text{ bits/second.}$$

Shannon's theorem was a major step forward. Prior to Shannon, it was believed that either infinite power or infinitesimal rate would be required for arbitrarily reliable performance.

Example 1 An example of code performance: BPSK.

$$P_b = Q(\sqrt{2E_b/N_0})$$

Improvement: repetition code, or increase E_b/N_0. □

Shannon's theorem sparked a great deal of research into finding codes.

1. 1950: Hamming codes (single-error correcting)
2. 1959–1960: BCH and Reed-Solomon codes
3. (Late 50's: convolutional codes)

It took 10 years to bridge the gap! This is non-trivial stuff.

Some motivation

Why is coding used? It can provide the difference between an operating communications system and a disfunctional system! Every modern digital communications link (including magnetic storage, cellular phones, the internet) employs some kind of error control. (Networks usually do not have forward error correction.) In some instances, the gains are significant

DSN: "When viewed from a purely economic standpoint, the impact can be impressive. It was estimated in the late 1960's that each dB of coding gain was worth $1,000,000 in development and launch costs. The current value for the Deep Space network is $80,000,000 per dB of gain. In the *Galileo* mission, coding may have made the difference between success and failure in a billion dollar effort." [Heegard and Wicker, p. 5].

A little bit of digital communications

For most of the semester, we will assume binary digital communications using BPSK. This is modeled (from a bit point of view) using the BSC. From a signal point of view, we send a normalized signal $\psi(t)$ with either an amplitude $+\sqrt{E_b}$ or $-\sqrt{E_b}$, where E_b is the energy/bit (in Joules/bit). The performance of BPSK is determined by

$$P_b = p = Q(\sqrt{2E_b/N_0})$$

(Show slide.)

Let us suppose that we are sending R_S bits/second. Then the power (energy/time) is equal to $P = R_S E_b$ Watts. Now, what if we *increase* the rate of transmission (send more bits per second), while retaining the same transmission power? Suppose that we have a quantity R, the **code rate**, such that $R < 1$. If we need to send R_S/R bits second (a higher rate). Then, fixing the power will require a new energy level E_2 satisfying

$$P = (R_S/R)E_2 = R_S E_b$$

so that

$$E_2 = RE_b.$$

That is, we have *less* energy per bit, because the bits are coming faster.

Now, what does this have to do with error correction? Suppose that we have a desired bit throughput rate of R_S bits/second, and a fixed level of power P. Error correction introduces redundancy. We might get (for example) 15 bits of coded information for every 11 bits input. We say this is a rate $R = 11/15$ code. To keep the original date throughput the same, we must send the coded bits at a rate 15/11 faster. This means that the energy/bit that can be expended on the coded information is 11/15 of the energy/bit that can be expended on the uncoded information: we are losing ground.

Suppose we have a system where $E_b/N_0 = 10$ dB. For BPSK, this corresponds to a BER of 3.9×10^{-6}. When a (15,11) code is used, the SNR is 8.65 dB, a drop of $10\log_{10}(11/15) = -1.34$ dB.

But, the (15,11) is able to correct a single error in every block. It can be shown that the output of the decoder has a BER of 5.6×10^{-8}, better than it was before it was coded. To achieve this BER without coding, the transmitter would have had to produce an E_b/N_0 of 11.5 dB. The difference between this value, and the value actually sent, is the coding gain. (Show slide.)

Distance between codewords

Error correction is obtained by adding redundant symbols to a codeword. If we send k symbols into the coder, we get n out. The **rate** of a code is

$$R = k/n.$$

It is always the case for error correction that $R < 1$.

Example 2 Binary repetition code. □

We can represent codewords as points in some geometrical space. The distance measure that we use is the **Hamming distance**. If $\mathbf{x}$ and $\mathbf{y}$ are codewords, then the Hamming distance is

$$d(\mathbf{x}, \mathbf{y}) = \text{number of places that } \mathbf{x} \text{ and } \mathbf{y} \text{ differ}$$

We also define the **weight** of a vector $\mathbf{x}$ as the number of elements of $\mathbf{x}$ which are nonzero. That is,

$$w(\mathbf{x}) = d(\mathbf{x}, \mathbf{0}).$$

Definition 1 Let

$$\mathcal{C} = \{\mathbf{c}_i, i = 0, 1, \ldots, M-1\}$$

be a code. The **minimum distance** of the code is the Hamming distance between pairs of codewords of smallest distance,

$$d^* = \min_{c_i, c_j \in \mathcal{C}, 1 \neq j} d(\mathbf{c}_i, \mathbf{c}_j).$$

□

For a code with k input symbols and n output symbols and minimum distance d^* we may write that this is a (n, k, d^*) code.

Think of each codeword as being a point in n-dimensional space. (Draw picture) Suppose that $\mathbf{c}_0$ is sent, and $\mathbf{r}$ is received, and that $d(\mathbf{c}_0, \mathbf{r}) = 1$. If the distance to all other codewords is larger than 1, we may guess that $\mathbf{c}_0$ is the transmitted code. The design goal for error correction coding, essentially, is how to choose 2^k points out of a space of 2^n points ($n > k$) to maximize the distance between all pairs of codewords.

Suppose that t errors occur between transmission and reception. If

$$d^* \geq 2t + 1,$$

then the closest codeword is correct. Thus we can correct up to t errors.

We can also check if an error has occurred. We have both correction capability and detection capability. The coding design problem, in a nutshell, is to place points in an n-dimensional space to maximize the distance.

Example 3 Parity check codes.

$$00 \rightarrow 000$$
$$01 \rightarrow 011$$
$$10 \rightarrow 101$$
$$11 \rightarrow 110$$

What is d^*? How many errors can we correct? How many errors can we detect? □

Classes of codes

(Picture) We will focus throughout the semester on linear codes.

We will first work through a code known as the **Hamming code** to introduce some of the ideas. The intent here is to introduce the concepts; we will return to these ideas later to clarify some of the details.

Linear codes begin with a generator matrix G. Let $\mathbf{m}$ be an input vector of k symbols. Then the codeword is computed by

$$\mathbf{c} = \mathbf{m}G.$$

For example,

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

If $\mathbf{m} = [m_0, m_1, m_2, m_3] = [1, 0, 0, 0]$ then

$$\mathbf{c} = [1,0,0,0] \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} = [1,0,0,0,1,0,1]$$

Note that **in this case** the input word appears explicitly in the output code. Such a code is called a **systematic code**. The other bits are called **parity bits**.

The use of matrices to describe these codes motivates a close examination of linear algebra.

For each linear code there is also a **parity check** matrix H with the following property: An n-tuple $\mathbf{c}$ is a codeword if and only if it is orthogonal to every row vector of H:

$$\mathbf{c}H^T = 0 \Leftrightarrow \mathbf{c} \text{ is a codeword.}$$

(Explain "orthogonal to every row vector").

Now we observe that the **rows** of G are possible codewords. (why?) Then we have

$$GH^T = 0.$$

Let us use this property about the parity check matrix. Suppose we send $\mathbf{c}$, and an error occurs:

$$\begin{aligned} \mathbf{r} &= \mathbf{c} + \mathbf{e} \\ &= \mathbf{m}G + \mathbf{e}. \end{aligned}$$

At the receiver, we do not know $\mathbf{e}$. If we multiply $\mathbf{r}$ by H^T we obtain the **syndrome**:

$$\mathbf{s} = \mathbf{r}H^T = (\mathbf{m}G + \mathbf{e})H^T = \mathbf{m}GH^T + \mathbf{e}H^T = \mathbf{e}H^T.$$

Thus the syndrome does not depend upon the transmitted codeword. Now, given the syndrome we want to determine our best guess of the error $\mathbf{e}$.

Let us look at the syndrome for the code above. We have

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

The syndrome is given by

$$\mathbf{s} = \mathbf{r}H^T = [r_1, r_2, r_3, \ldots, r_7] \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= r_1[1\ 0\ 1] + r_2[1\ 1\ 1] + r_3[1\ 1\ 0] + \cdots + r_7[0\ 0\ 1]$$

The syndrome is the sum of the columns of H (sideways).

Now suppose $\mathbf{m} = [1, 0, 0, 0]$. Then

$$\mathbf{c} = [1,0,0,0,1,0,1]$$

Suppose that

$$\mathbf{r} = [1, \underline{1}, 0, 0, 1, 0, 1]$$

(Note the error). The syndrome is

$$\mathbf{s} = \mathbf{r}H^T = [1, 1, 1].$$

This is the second column of H. Hence, since the syndrome is not zero, we conclude that an error occurred in the second position of $\mathbf{r}$.

Note that this code cannot correct more than one error: if there were two errors, say at r_2 and r_4, then the syndrome is the sum of the syndromes, which gives us just another column of H. Extension of this idea to be able to correct more than one error leads us to a careful examination of the way we do arithmetic. We will take the mathematical properties we are familiar with and use them to create other structures that have sufficient power to do what we need.