CHANNEL CODING THEOREM

For a discrete memoryless channel, represents the maximum amount of information that can be transmitted per channel use in a reliable manner.

The *channel coding theorem* states:

If a discrete memoryless channel has capacity *C* and a source generates information at a rate less than *C*, then there exists a coding technique such that the output of the source may be transmitted over the channel with an arbitrarily low probability of symbol error.

For the special case of a binary symmetric channel, the theorem teaches us that if the code rate *r* is less than the channel capacity *C*, then it is possible to find a code that achieves error-free transmission over the channel. Conversely, it is not possible to find such a code if the code rate *r* is greater than the channel capacity *C*. Thus, the channel coding theorem specifies the channel capacity *C* as a *fundamental limit* on the rate at which the transmission of reliable (error-free) messages can take place over a discrete memoryless channel. The issue that matters here is not the SNR, so long as it is large enough, but how the channel input is encoded.

The most unsatisfactory feature of the channel coding theorem, however, is its nonconstructive nature. The theorem asserts the existence of good codes but does not tell us how to find them. By *good codes* we mean families of channel codes that are capable of providing reliable transmission of information (i.e., at arbitrarily small probability of symbol error) over a noisy channel of interest at bit rates up to a maximum value less than the capacity of that channel. The error-control coding techniques described in this chapter provide different methods of designing good codes

Notation

Many of the codes described in this chapter are *binary codes*, for which the alphabet consists only of binary symbols 0 and 1. In such a code, the encoding and decoding functions involve the binary arithmetic operations of *modulo-2 addition and multiplication*

performed on codewords in the code.

Throughout this chapter, we use the ordinary plus sign (+) to denote modulo-2 addition. The use of this terminology will not lead to confusion because the whole chapter relies on binary arithmetic.

Thus, according to the notation used in this chapter, the rules for modulo-2 addition are as follows: Because 1 + 1 = 0, it follows that 1 = –1. Hence, in binary arithmetic, subtraction is the same as addition. The rules for modulo-2 multiplication are as follows:

Division is trivial, in that we have and division by 0 is not permitted. Modulo-2 addition is the EXCLUSIVE-OR operation in logic and modulo-2 multiplication is the AND operation.

Linear Block Codes

By definition:

A code is said to be linear if any two codewords in the code can be added in modulo-2 arithmetic to produce a third codeword in the code.

Consider, then, an (*n*,*k*) linear block code, in which *k* bits of the *n* code bits are always Identical to the message sequence to be transmitted. The (*n* – *k*) bits in the remaining portion are computed from the message bits in accordance with a prescribed encoding rule that determines the mathematical structure of the code. Accordingly, these (*n* – *k*) bits are referred to as *parity-check bits*. Block codes in which the message bits are transmitted in unaltered form are called *systematic codes*. For applications requiring *both* error detection and error correction, the use of systematic block codes simplifies implementation of the decoder.

In the linear block codes, the parity bits and message bits have a linear combination, which means that the resultant code word is the linear combination of any two code words.

Let us consider some blocks of data, which contains **k** bits in each block. These bits are mapped with the blocks which has **n** bits in each block. Here **n** is greater than **k**. The

transmitter adds redundant bits which are n−kn−k bits. The ratio **k/n** is the **code rate**. It is denoted by **r** and the value of **r** is **r < 1**.

The n−kn−k bits added here, are **parity bits**. Parity bits help in error detection and error correction, and also in locating the data. In the data being transmitted, the left most bits of the code word correspond to the message bits, and the right most bits of the code word correspond to the parity bits.

Block codes operate on a block of bits. Block codes are referred to as (n, k) codes. A block of k information bits are coded to become a block of n bits. But before we go any further with the details, lets look at an important concept in coding called Hamming distance. Lets say that we want to code the 10 integers, 0 to 9 by a digital sequence. Sixteen unique sequences can be obtained from four bit words. We assign the first ten of these, one to each integer. Each integer is now identified by its own unique sequence of bits.

Let *m*0, *m*1, □, *mk* – 1 constitute a block of *k* arbitrary message bits. Thus, we have 2*k* distinct message blocks. Let this sequence of message bits be applied to a linear block

0 + 0 = 0

1 + 0 = 1

0 + 1 = 1

1 + 1 = 0

Because 1 + 1 = 0, it follows that 1 = –1. Hence, in binary arithmetic, subtraction is the same as addition. The rules for modulo-2 multiplication are as follows:

Division is trivial, in that we have

and division by 0 is not permitted. Modulo-2 addition is the EXCLUSIVE-OR operation in logic and modulo-2 multiplication is the AND operation.

Creating block codes: The block codes are specified by (n.k). The code takes k information bits and computes (n-k) parity bits from the code generator matrix. Most block codes are systematic in that the information bits remain unchanged with parity bits

attached either to the front or to the back of the information sequence.

* Hamming code, a simple linear block code

* Hamming codes are most widely used linear block codes.

* A Hamming code is generally specified as (2n- 1, 2n-n-1).

* The size of the block is equal to 2n-1.

The number of information bits in the block is equal to 2n-n-1 and the number of overhead bits is equal to n. All Hamming codes are able to detect three errors and correct one.