Error Detection and Correction

This lecture introduces the topic of channel coding. These are codes that allow the receiver to (usually) detect and (often) correct errors introduced by the channel.
After this lecture you should be able to: list some advantages and disadvantages of checksums; compute even and odd parity bits; compute the Hamming distance between two code words; compute the code rate for block, punctured and non-punctured convolutional codes; correct errors in a received block code word by exhaustive search; compute coding gain; and compute the punctured output of a convolutional encoder.

Checksums

A simple way to check for errors in a frame of data is to compute the sum of the byte (or word) values in a frame of data. The sum is computed modulo[1] the maximum value of the check sum. The additive complement[2] of the sum is then appended to the packet to ensure the checksum of an error-free packet will be zero. This is the type of error-detection used by TCP/IP frames used on the Internet.

Checksums are typically used by higher-level protocols since they are easy to compute in software. However, there are many common types of errors, such as insertion of zero words and transposition of values that are not detected by checksums.

Exercise 1: Compute the modulo-4 checksum, C, of a frame with byte values 3, 1, and 2. What values would be transmitted in the packet? What would be the value of the checksum at the receiver if there were no errors? Determine the checksum if the received frame was: 3, 1, 1, C? 3, 1, 2, 0, C? 1, 2, 3, C?

Error Detection

Another technique for detecting errors in received frames is for the transmitter to compute additional bits called "parity bits" and add them to the end of the frame. The receiver computes parity bits itself from the received data and compares them to the received parity bits. If the computed and received parity bits match then either there were no errors or the data was corrupted in such a way the the corrupted data has the same parity value.

[1] "modulo-N" means the remainder after dividing by N.

[2] The "additive complement" is the value that would have to be added to make the result zero (modulo-N).

The probability of the latter event is called the *undetected error probability*. Good error detecting codes try to make this probability as low as possible.

Single Parity Bits

The simplest type of parity is a single parity bit. Typically the parity bit is computed as the modulo-2 sum of all of the bits in the message.

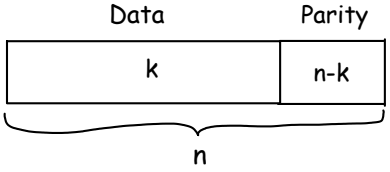
Exercise 2: What is a modulo-2 sum? What is the modulo-2 sum of 1, 0 and 1? What is the modulo-2 sum if the number of 1's is an even number?

The most common example of a single parity bit is a parity bit added to each ASCII character. Most serial interfaces can be configured to compute and append a parity bit to each ASCII character. This bit can be either the sum of all of the bits ("even parity") or it's complement ("odd parity").

The receiver computes the parity bit from the data bits and compares the computed parity bit to the transmitted parity bit. If the computed and received parity bits match then there was either no error or there were an even number of errors.

Block Codes

More complex channel codes use multiple parity bits. A block code where each block of n bits contains k data bits and $n-k$ parity bits is called an (n,k) code:



Exercise 3: How many possible code words are there for an (n, k) code? How many possible parity bit patterns are possible for each code word?

The *Hamming Distance* is the number of bits that differ between two code words. The performance of a code is mainly determined by the minimum (Hamming) distance (D_{min}) between any two code words in the code.

Exercise 4: What is the Hamming distance between the codewords 11100 and 11011? What is the minimum distance of a code with the four codewords 0111, 1011, 1101, 1110?

The *rate* of a code is the ratio of information bits to total bits, or k/n. This is a measure of the efficiency of the code. As we add parity bits the code rate decreases but, for a well-designed code, the minimum distance and thus the error-correcting ability increases.

Forward Error Correction Coding

Certain block codes allow the receiver to correct errors introduced by the channel. These types of codes are called Forward Error Correcting (FEC) because the receiver does not have to ask for erroneous code words to be retransmitted.

Error correction is possible when the code words include enough parity bits that the receiver can decide which codeword was transmitted even though the received codeword does not match any of the codewords that could have been transmitted (in other words, it is known to contain errors).

Minimum Distance Decoding

Conceptually, a receiver can correct errors by choosing the valid codeword that has the smallest Hamming distance from the received codeword. This is because codewords with fewer errors are more likely to happen than those with more errors.

Exercise 5: A block code has two valid codewords, 101 and 010. The receiver receives the codeword 110. What is the Hamming distance between the received codeword and each of the valid codewords? What codeword should the received decide was sent? What bit was most likely in error? Is it possible that the other codeword was sent?

In general, a block code with a minimum Hamming distance between valid codewords of d can detect $d-1$ errors and correct $\lfloor \frac{d-1}{2} \rfloor$ where $\lfloor \cdot \rfloor$ denotes the floor function (round down to the next smallest integer).

Exercise 6: What is the minimum distance for the code in the previous exercise? How many errors can be detected if you use this code? How many can be corrected? What are n, k, and the code rate (k/n)?

However, with large codes it is not possible to do an exhaustive search through all possible codewords to find the one with the minimum distance. There is a large field of study devoted to the design of codes which can be efficiently encoded and decoded.

Most block codes, including the well-known Hamming and Reed-Solomon codes, are encoded and decoded using the properties of polynomials with coefficients from a Galois Field.

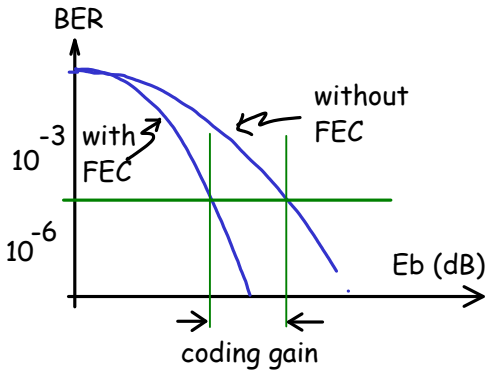
Coding Gain

When the FEC code is well-matched to the error rate and to the types of errors likely to be encountered, the use of FEC results in higher throughput because blocks that contain correctable errors do not need to be retransmitted.

Another possible advantage of using FEC is better power efficiency. This can happen if the same post-correction error rate can be achieved by transmitting less power. Even though the channel will introduce more errors because of the lower signal-to-noise ratio, the use of FEC can correct enough of these errors to reduce the error rate back to the error rate obtained when the higher power was used.

Since FEC requires additional parity bits to be transmitted we should compare power efficiency using only the information bits, not the parity bits. The metric used for this comparison is the energy per *information* bit, E_b. The ratio of the E_b required to achieve the desired error rate with and without coding is called the "coding gain":

$$\text{coding gain} = \frac{E_b \text{ (without coding)}}{E_b \text{ (with coding)}}$$



Exercise 7: What are the units of Energy? Power? Bit Period? How can we compute the energy transmitted during one bit period from the transmit power and bit duration?

Exercise 8: A system needs to operate at an error rate of 10^{-3}. Without FEC it is necessary to transmit at 1W at a rate of 1 Mb/s. When a rate-1/2 code is used together with a data rate of 2 Mb/s the power required to achieve the target BER decreases to 500mW. What is the channel bit rate in each case? What is the information rate in each case? What is E_b in each case? What is the coding gain?

Convolutional Codes

Although it is possible to use block codes to implement FEC, the trend until recently has been to use convolutional codes for communication systems. This was mainly because of the existence of a relatively simple and efficient decoding algorithm for convolutional codes called the Viterbi algorithm.

A rate k/n convolutional code is implemented by reading a certain number of bits into a shift register and outputting a number of modulo-2 sums of these bits. An example is shown below[3]:

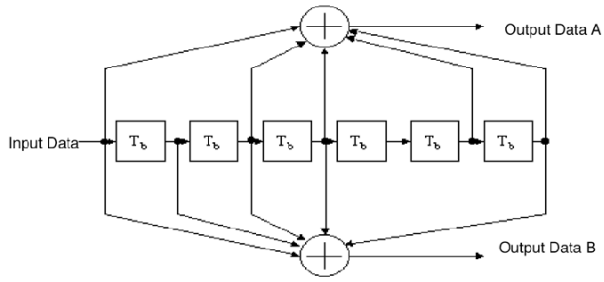


Figure 18-8—Convolutional encoder (k = 7)

[3]Taken from the 802.11 standard.

If there are n output bits for each k input bits the rate of the code is k/n. The "constraint length," (K) of the code is the number of bits that can affect each output bit and is equal to the length of the shift register plus one (because the input bit is also used in computing the output).

Exercise 9: Assuming one bit at a time is input into the encoder in the diagram above, what are k, n, K and the code rate?

Viterbi Algorithm

Most FEC decoders for convolutional codes use an algorithm called the Viterbi algorithm. This algorithm is a "maximum likelihood" decoder because it chooses the bit sequence with the minimum distance from the received sequence (or close to it).

The VA uses the trellis structure of the code (a tree with branches that merge) to avoid exponential increase in decoding complexity with message length. Instead, the complexity is proportional to 2^K where K is the constraint length. Although the algorithm is relatively complex, implementations are readily available.

Although many different convolutional codes are possible, there are certain standard codes that are used by many different systems. Hardware implementations are typically only available for these codes. The most common convolutional code is the rate-1/2 code with a constraint length of 7 shown above.

Higher-rate codes can be derived from the basic rate-1/2 code by not transmitting some of the bits. This is called "puncturing." The same decoder hardware can be used by feeding the decoder a value representing "unknown" (an "erasure") in place of the missing parity bits.

Exercise 10: Consider the encoder above. If the only the bits corresponding to the outputs A, A and B, and B are transmitted corresponding to every three input bits, what is the code rate of this punctured code?

Reed Solomon Codes

The Reed-Solomon code is a block FEC code that is widely used. RS codes operate on non-binary Galois fields, typically GF(64) or GF(256). A GF(256) code

is able to correct a certain number of 8-bit word errors, regardless of the number of bit errors in each word. For this reason Reed-Solomon codes are efficient for channels that have bursty errors because bursts of errors cause multiple errors to fall within the same 8-bit word.

Interleaving

In many cases errors are caused by white noise and appear at random times. However, sometimes errors are caused by short-duration events such as fades or intermittent interference. This causes a burst of errors to happen.

Bursty errors can overwhelm the error-correcting ability of FEC codes and greatly reduce their effectiveness. To avoid this, it is common to 'interleave' the output of the FEC encoder and de-interleave it before FEC decoding. If the interleaver size is sufficiently large this effectively turns bursty errors into random errors.

A typical interleaver is a block interleaver. Bits are written row-wise and read out column-wise. The interleaver writes and reads bits in the opposite way to rearrange the bits back into their original order.

Exercise 11: Give the numbering of the bits coming out of a 4x4 interleaver. If bits 8, 9 and 10 of the interleaved sequence have errors, where would the errors appear in the de-interleaved sequence? If the receiver could correct up to one error per 4-bit word, would it be able to correct all the errors without interleaving? With interleaving?

Exercise 12: If errors on the channel happened in bursts and you were using a RS code using 8-bit words, would you want to interleave bits or bytes?

Modern FEC Codes

Two other FEC codes have become popular in recent years because their error-correcting performance approaches the Shannon Limit.

Turbo Codes use two different codes to encode the data. The decoder uses the information from one code to help with decoding of the other code. Then that information is used to help decode the first code. The procedure is repeated iteratively until there are no errors (determined by a CRC) or a time limit is reached.

Low Density Parity Check (LDPC) codes are block codes with sparse (few 1's) parity check matrices. This means that each parity bit is a function of only a few message bits.

Modern FEC decoders also use "soft-decision" decoding algorithms that operate on the probabilities that particular bits are zeros or ones rather than operating on binary "hard decisions".