ELEX 7860 : Wireless System Design
2021 Winter Term

Error Detection and Correction

This lecture introduces the topic of channel coding. These are codes that allow the receiver to (sometimes) detect and correct errors introduced by the channel.

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 most likely to have been 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

In principle, a receiver can determine the transmitted codeword with maximum likelihood 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 be received than those with more errors.

Exercise 1: 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 receiver decide was sent? What bit was most likely in error? Is it possible that the other codeword was sent?

A block code with a minimum Hamming distance of d can detect $d-1$ errors. This is because if a code has a minimum distance of d, any codeword a distance $d-1$ away will be different than a valid codeword and will be detected as an invalid codeword (i.e. an error).

A block code with a minimum distance of d can correct $\lfloor\frac{d-1}{2}\rfloor$ where $\lfloor\cdot\rfloor$ denotes the floor function (round down to the next smallest integer). This is because if a code was minimum distance of d then less than $(d)/2$ (if d odd) or less than $(d-1)/2$ (if d even) errors will not move a codeword closer to another valid codeword and by choosing the valid codeword closest to the received one we will correct the error.

Exercise 2: 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.

Block FEC Codes

A simple approach is to repeat each bit N times. This (Nk, k) repetition code has a minimum distance N, which happens when two codewords differ in one data bit.

Exercise 3: How many errors can an N-fold repetition code detect? Correct? What is the code rate?

Another approach is to arrange the data bits in a rectangular array and compute one modulo-2 sum for each column and one for each row. These parity bits are transmitted along with the data bits. At the receiver the parity bits are re-computed and any discrepancies point to an error in a particular row and column.

For example, for 9 data bits we would use a 4×4 matrix where the last column and row contain the modulo-2 sums of the respective rows and columns:

$$\begin{array}{ccc|c}
0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 \\
\hline
0 & 0 & 1 & 1
\end{array}$$

Exercise 4: If the rows of bits we received were 00111, 01010, 11110, 01001, 01010 which bit is likely in error?

Although such horizontal-and-vertical parity check codes[1] are more efficient than repetition

[1]Sometimes known as transverse and longitudinal parity checks since they were originally used to protect information stored on tape.

codes, the most efficient single-error-correcting codes are Hamming codes.

For an integer value m, a Hamming code has a codeword size of $n = 2^m - 1$, $k = 2^m - 1 - m$ data bits and $n - k = m$ parity bits. For example, for $m = 6$ each codeword has $n = 63$ bits and $k = 64-6-1 = 57$ data bits for a rate of $\frac{57}{63} \approx 0.90$.

Generator and Parity Check Matrices

A linear block code is one where each n-bit codeword can be generated by multiplying a $1 \times k$ data vector d by a $k \times n$-bit generator matrix G. These contain 1's and 0's and the sums are computed modulo-2.

A systematic code is one where the data bits are transmitted unchanged. For a systematic code the first (or last[2]) k columns are an identity matrix:

$$G = \left[I_k \,|\, P \right]$$

and the columns of P contain a '1' for each data bit included in the calculation of each parity bit.

Exercise 5: What is the generator matrix for the (5,3) code that computes two parity bits as: $p_0 = d_0 \oplus d_1$ and $p_1 = d_1 \oplus d_2$ where d_i is the i'th data bit?

Note that permuting the rows is equivalent to changing the order of the data bits and permuting the columns changes the order of the transmitted bits. In both cases the minimum distance and thus the error-correcting performance of the code is unaffected.

A parity check matrix, H, is an $(n-k) \times n$-bit matrix that is multiplied by the received $n \times 1$ codeword vector to check the parity equations:

$$H = \left[P^T \,|\, I_{n-k} \right]$$

If the product is zero then there were no errors detected.

Exercise 6: What is the parity check matrix for the code above? If data vector $[101]$ is to be transmitted, what is the codeword? If there are no errors, what is the result of multiplying the received codeword by H? If the channel introduces an error into the second bit?

Syndrome Decoding

If we can correct up to t errors in n bits, the number of possible correctable error patterns, $\binom{n}{t}$, will be much smaller than the number of possible received codewords, 2^n. It is much more efficient to search through the possible correctable error patterns than to search through all possible received codewords.

Exercise 7: How many possible correctable error patterns are there for a $(31, 26)$ Hamming code? How many possible received bit patterns?

If the received codeword is the transmitted vector x added (xor'ed) with an error vector e, then the result of the parity check is $H(x + e) = Hx + He = 0 + He$. He is called the syndrome. One possible decoding method is to generate all possible syndromes and map each one to an error vector. By adding the corresponding error vector to the received message we can correct the error.

Exercise 8: What are the possible syndromes for the code above? What was the syndrome when the second bit was in error? Would the code correct the right bit?

Exercise 9: Does a syndrome of zero indicate an error? What is the largest value of n for which a k-bit syndrome can correct a single error?

Cyclic Codes

An important subset of linear block codes are cyclic codes in which each codeword is a multiple of an $n-k$-bit generator polynomial. As in the case of CRCs, this allows for efficient implementation of encoders and decoders.

The receiver can divide the received codeword by the generator polynomial to obtain a syndrome. If the result is non-zero then there are error(s). Decoding algorithms search to find the error pattern that would result in that syndrome.

Reed Solomon Codes

A Reed-Solomon code is a block FEC code that is widely used. RS codes typically operate on non-binary Galois fields, for example GF(64) or GF(256). A RS code uses blocks of length $n = 2^m - 1$ where m is the number of bits per symbol. A RS code can correct $\frac{(n-k)}{2}$ symbol errors.

Exercise 10: What is the block size for a RS code using symbols from GF(64) in bit? In symbols?

Exercise 11: How many parity symbols would we need if we wanted to correct 8 8-bit symbol errors? What are (n, k) for this code?

A GF(256) code uses 256 symbols and so each symbol can represent an 8-bit word. When a symbol error is corrected, all of the corresponding bits are correct.

[2] Unfortunately, there are different conventions for both the matrix shapes and the order of the data and parity bits in G.

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.

Exercise 12: A block FEC code uses values from GF(4). The 4 possible elements are represented using the letters A through D. The valid code words are: ABC, DAB, CDA, and BCD.

What is the minimum distance of this code? How many errors can be detected? Corrected?

If the codeword ADA is received, was an error made? Can it be corrected? If so, what codeword should the decoder decide was transmitted?

If each codeword represents two bits, how many bit errors were corrected?

Repeat if the codeword received was AAA.

Reed-Solomon codes belong to a class of cyclic codes known as BCH codes.

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]:

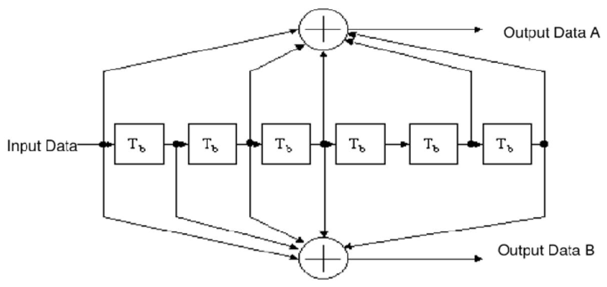


Figure 18-8—Convolutional encoder (k = 7)

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 delay of the oldest bit that can affect the output and is equal to the length of the shift register plus one (because the input bit is also used in computing the output).

[3]Taken from the 802.11 standard.

Exercise 13: Assuming one bit at a time is input into the encoder in the diagram above, what are k, n, K and the code rate?

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

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.

Practical Aspects

Puncturing, Shortening, Erasures

Higher-rate codes can be derived from the basic rate-1/2 convolutional code by not transmitting some of the generated bits. This is called "puncturing." The same decoder hardware can be used by feeding the decoder a value representing "unknown."

We can also reduce the rate of a block code by setting certain bits in the block to known values (e.g. zero) before encoding and not transmitting them. This is called "shortening." At the receiver the known values are re-inserted and decoding is done as usual.

If the channel detects an unreliable or missing bit (an "erasure"), the decoder can ignore that bit. This improves performance.

Exercise 14: Consider the convolutional 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?

For example, RS codes can correct up to $n-k$ erasures[4].

[4]And various combinations of erasures and errors.

Exercise 15: A code uses symbols drawn from GF(8) and is implemented using modulo-8 arithmetic with digits 0 to 7. If we implement parity sum and the three symbols are 2, 4 and 5, what is the value of the parity symbol? If we receive the sequence 1, 2, 3, 4 was there an error? If we know the error was in the first symbol, what was the correct value?

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.

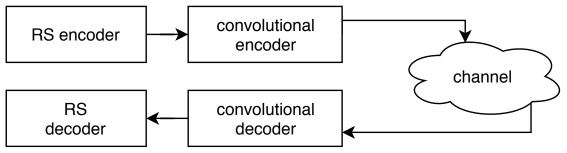
Bursty errors can overwhelm the error-correcting ability of FEC codes and greatly reduce their effectiveness. To avoid this, it is common to 'interleave' the output of the FEC encoder and de-interleave it before FEC decoding. If the interleaver size is sufficiently large this effectively turns bursty 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 16: 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 17: 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?

Concatenated Codes



It's common to apply a second FEC code to the result of a first FEC code. This is typically done because different codes perform differently for different error patterns. For example, the errors from a Viterbi decoder tend to be bursty with a burst duration of approximately the constraint length of the code. By following a RS encoder with a convolutional code (so that the RS decoder follows the Viterbi decoder) the RS code can efficiently "clean up" after the Viterbi decoder. The net result is better performance than using a single code at a lower rate.

Soft-Decision Decoding

In practice FEC decoders use "soft-decision" decoding algorithms that operate on the probabilities that particular bits are ones (or zeros) rather than on binary "hard decisions." For example, received bits may be less reliable when the signal is faded and these bits can be given less weight when computing which codeword was most likely to have been transmitted ("maximum likelihood decoding"). Soft-decision ML decoders have significantly better performance than hard-decision decoders.

It's also possible to declare that some bits were not received at all. As with puncturing (where the bits are not transmitted at all), these "erasure" bits then are assigned a zero reliability (in fact, a "log-likelihood" of zero) and have no effect on the decoding.

Modern FEC Codes

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

Turbo Codes use two codes to encode the data. The decoder uses the information obtained from decoding one code to help with decoding of the other code. That information is then used to help decode the first code. The iterative procedure is repeated 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. As above, decoding of LDPC codes is iterative with each of the parity check equations being used successively to update the log-likelihoods of the received bits.

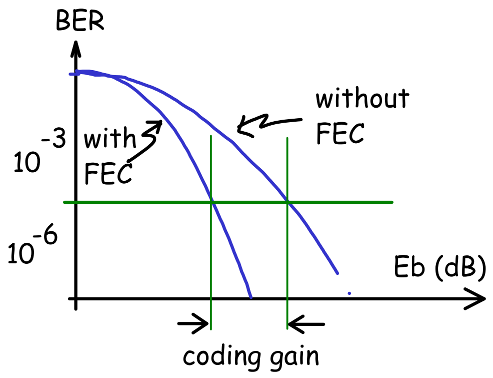
Coding Gain

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 18: 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 19: 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?