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.
After this lecture you should be able to: compute the minimum number of detectable and correctable errors from the code's (minimum) Hamming distance; correct errors in a received block code word by exhaustive search; convert between the parity check equations, generator matrix and parity check matrix for a systematic block code; compute the syndrome for a received word using the parity check matrix; compute the block length and number of correctable symbol errors for a Reed Solomon code of a given symbol size; draw the Tanner graph for a parity check matrix; determine n, k and the code rate for convolutional codes ; compute the output bits from a standard K=7 rate-$\frac{1}{2}$ convolutional encoder given the starting state and input; interleave a bit sequence using a block interleaver of a given depth and word size; compute the punctured output of a convolutional encoder; compute E_b/N_0 from SNR, channel bit rate, code rate and sampling rate (or bandwidth); determine coding gain from a BER vs Eb/No curve.

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

Repetition and Hamming 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?

For any integer value $m \geq 2$, a Hamming code has $n-k=m$ parity bits, a codeword size of $n=2^m-1$, and $k = 2^m - 1 - m$ data bits. Hamming codes have a minimum distance of 3 and can thus correct one error.

"Perfect" codes have the highest possible code rate for a given minimum distance. Hamming codes are perfect codes for a minimum distance of 3.

Exercise 4: What are n, k, and the code rate for a Hamming code with $m = 3$? $m = 6$? $m = 2$?

Generator and Parity Check Matrices

$1 \times n$-bit codewords, c, can be generated by multiplying a $1\times k$ data vector d by a $k\times n$-bit generator matrix G: $c = dG$. 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[1]) 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:

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

that is multiplied by the received $n\times 1$ codeword vector c^T to obtain $(n-k)\times 1$ parity check vector $p = Hc^T = H(dG+e)^T = H((dG)^T + e^T) = HG^T d + He^T$ where e is the $1\times n$=bit vector of errors introduced by the channel. Since HG^T is zero, p will be zero only if e, the vector of errors introduced by the channel, is zero.

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?

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

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 c added (xor'ed) with an error vector e, then the result of the parity check is He^T. He^T is called the syndrome. When the number of possible syndromes is small we can 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?

Exercise 9: Does a syndrome of zero correspond to an error? If each syndrome corresponds to a different error vector, what is the largest value of n for which a $n-k$-bit syndrome can correct a single error? What are n and k for Hamming code with $m = 3$?

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, g. As in the case of CRCs, this allows for efficient implementation of encoders and decoders.

Block codes including Reed-Solomon codes, are encoded and decoded using the properties of polynomials with coefficients from a Galois Field (to be covered later).

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

Reed-Solomon codes are widely-used block FEC codes that belong to a class of cyclic codes known as BCH codes. RS codes use non-binary symbols such as GF(64) or GF(256). A RS code uses blocks of length

$n = 2^m - 1$ symbols 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 corrected. For this reason Reed-Solomon codes are efficient for correcting bursty errors because each corrected symbol corrects all bit errors 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.

Convolutional Codes

For many years convolutional codes were the most widely used FEC codes for communication systems due to the existence of a relatively simple and efficient decoding algorithm called the Viterbi algorithm.

A rate k/n convolutional code is implemented by reading a one (or more) bits into a shift register and outputting two (or more) modulo-2 sums of these bits. An example is shown below[2]:

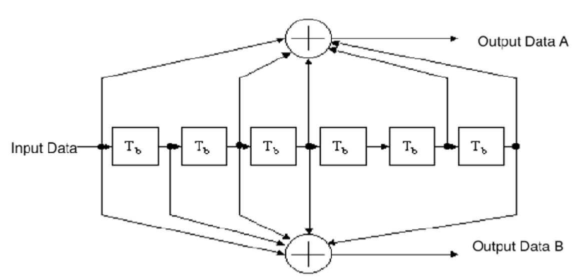


Figure 18-8—Convolutional encoder (k = 7)

[2]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 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).

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, constraint length of 7 code shown above with generator polynomial coefficients 133 and 171 (in octal notation).

Viterbi Algorithm

Most FEC decoders for convolutional codes use an algorithm called the Viterbi algorithm (VA). 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 exploits the state machine structure of the encoder to avoid exponential increase in decoding complexity. The decoder keeps track of the likelihood (actually, the log-likelihood) of the input sequence that is most likely to have resulted in each of the 2^K encoder states. At the end of the message, or when the length of stored input sequences are a few times longer than the constraint length, the sequence (or the older bit(s) of the sequence) corresponding to the encoder state with the highest likelihood are output by the decoder. Thus the decoder complexity is proportional to 2^K instead of 2^n.

Practical Aspects

Puncturing, Shortening, Erasures

We can increase the rate of a code by not transmitting some of the bits output from the encoder. This is called "puncturing." The same decoder hardware can be used by feeding the decoder a value representing "unknown" for the punctured bits.

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?

We can *reduce* the rate of a code by setting certain bits 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 the decoder can ignore that bit. This improves performance. The missing bit is called an "erasure."

For example, RS codes can correct up to $n-k$ erasures instead of $(n-k)/2$ errors[3].

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.

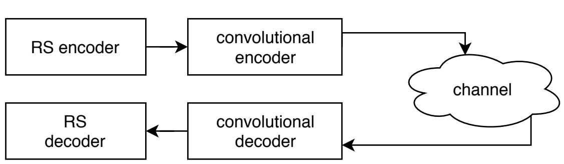
A burst of errors can exceed the error-correcting ability of an FEC code and greatly reduce its effectiveness. To avoid this, it is common to 'interleave' the output of the FEC encoder and de-interleave it before FEC decoding. If the interleaver size is sufficiently large this effectively turns bursty 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 15: 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 16: 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

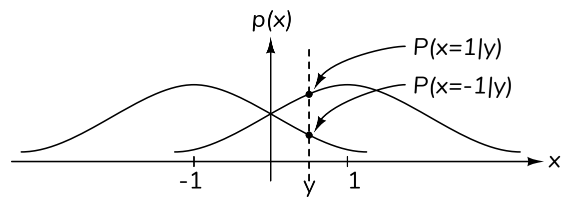


It's common to apply a second FEC code to the result of a first FEC code. This is typically done because different codes perform differently for different error patterns. For example, the errors from a Viterbi decoder tend to be bursty with a burst duration of approximately the constraint length of the 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.

Likelihood Ratios and Soft-Decision Decoding

If a receiver knows the statistics of the channel, such as the noise variance for an AWGN channel, it can compute the probability that a particular value x_i was transmitted based on the received value y_i. This probability is called a likelihood.

For example, if we transmit $x_i = \pm 1$ with equal probability over an AWGN channel with noise variance σ^2 then we can compute the probability that a $+1$ or a -1 was transmitted from y_i:



This is different than the probability of computing the probability of obtaining a value (or range of values) for a given value of x_i.

FEC decoders typically use "soft-decision" decoding algorithms that operate on likelihoods rather than on binary "hard decisions." Soft-decision decoders have significantly better performance than hard-decision decoders.

Soft-decision algorithms often use "log likelihoods" or log-likelihood ratios:

$$\log\left(\frac{P(x_i=0)}{P(x_i=1)}\right)$$

Exercise 17: If the likelihood of a bit being 1 is 0.5 what is the log-likelihood? What if the probability of a bit being 0 is 0.25? What is the product of the two probabilities? The ratio? The sum and difference of the two log-likelihoods?

Exercise 18: What are the log-likelihood ratios for: A punctured bit? An erasure? A shortened bit equal to 0? If 0 and 1 are equally likely to have been transmitted?

[3] And various combinations of erasures and errors.

Turbo Codes

"Turbo" Codes use iterative decoding of concatenated codes. The structure is similar to a concatenated code but with an interleaver placed between the two encoders and a de-interleaver between the decoders. 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 (e.g., as determined by a CRC) or an iteration limit is reached.

Turbo Codes were the first practical codes to approach the Shannon capacity bound and were used in 3G and 4G cellular systems.

LDPC codes

LDPC codes are widely used block codes. They are (almost) optimum and have low implementation complexity.

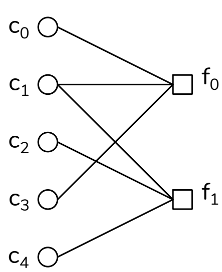
Low density (sparse) parity check matrices make it possible to implement codes whose codewords are thousands of bits long.

Exercise 19: What is k for a rate 0.9 code with n=10000? If d = n-k+1 how many errors could this code correct? [This is actually an upper bound on d.]

We can draw a parity check matrix as a bipartite (two part) "Tanner graph" where the nodes in one part are the codewords bits (variable nodes) and the nodes in the other are the parity check bits (check nodes). The parity check matrix defines the connections between the variable and check nodes.

For example, for the (5,3) code described above, the parity check matrix and the Tanner graph would be:

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



A simple decoding algorithm is to "flip" the bit(s) for which the number of failed parity checks exceed a threshold. The decoder repeats this process until all check nodes are zero or it reaches an iteration limit.

More effective decoding algorithms use "soft-decision" iterative "belief propagation" between codeword nodes and checksum nodes where a "belief" is a likelihood (or LLR).

LDPC codes are used in 4G and 5G cellular and several WLAN standards.

Polar Codes

A Polar code is a block code whose generator matrix has a structure that "polarizes" the channel so that some decoded bits have a lower error rates and others have a higher error rate although the overall capacity of the channel is unchanged. Data is transmitted over the k bits that have a low error rate while zeros – known as "frozen" bits – are transmitted over the $n - k$ bits that have a high error rate.

Typical decoding algorithms use "successive cancellation" algorithms that sequentially use more-reliable bits to correct errors in less-reliable bits.

Polar codes are used in 5G cellular standards (along with LDPC codes).

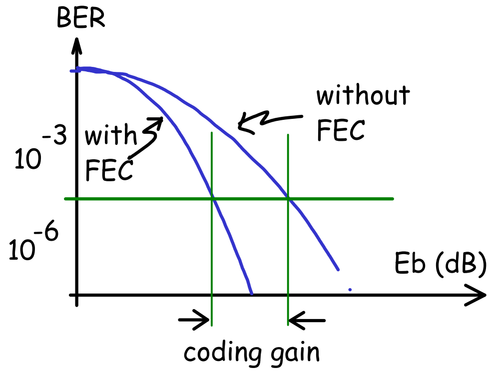
Coding Gain

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.

FEC can result in better power efficiency. This happens when 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 20: 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 21: 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?