ECE 7670
Lecture 2 – Linear block codes

Objective: To deepen understanding of the concepts of linear block codes previously introduced.

1 Introduction to linear block codes

Recall the example from the last time of a code: we had some generator matrix, some parity check matrix, and a means of doing some decoding. We will generalize these ideas.

A block code is a code in which k bits (or, more generally, symbols) are input and n bits (or, more generally symbols) are output. We designate the code as an (n, k) code. We will start with bits, elements from the field $GF(2)$; later we will consider elements from a field $GF(q)$ (after we know what this means).

If we input k bits, then there are 2^k distinct messages (or, more generally q^k). Each message of n symbols associated with a with each input block is called a *codeword.* We could, in general, simply have a lookup table with k inputs and n outputs. However, as k gets large, this quickly becomes infeasible. (Try $k = 255$, for example.) We therefore restrict our attention to linear codes.

Definition 1 A block code C of length n with 2^k codewords is called a **linear** (n, k) code if and only if its 2^k code words form a k-dimensional subspace of the vector space of all n-tuples over the field $GF(2)$.

More generally, with a bigger field, a block code C of length n with q^k is called a **linear** (n, k) code if and only if its q^k code words form a k-dimensional subspace of the vector space of all n-tuples over the field $GF(q)$. □

We remind ourselves of what a vector space is: we have an addition defined that is commutative and closed; we have scalar multiplication that is closed, distributive, and associative. We will formalize these properties a little further, but this suffices for the present purposes. We will see (later) that we have a *group structure* on the addition operation.

So what does this mean for codewords: the sum of any two codewords is a codeword. Being a linear vector space, there is some *basis*, and all codewords can be obtained as linear combinations of the basis. We can designate $\{\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_{k-1}\}$ as the basis vectors. In a nutshell, it means that we can represent the coding operation as matrix multiplication, as we have already seen. We can formulate a *generator matrix* as

$$G = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{k-1} \end{bmatrix}$$

G is a $k \times n$ matrix. If $\mathbf{m} = (m_0, m_1, \ldots, m_{k-1})$ is an input sequence, then the output is the codeword

$$\mathbf{m}G = m_0\mathbf{g}_0 + m_1\mathbf{g}_1 + \cdots + \mathbf{m}_{k-1}\mathbf{g}_{k-1}.$$

We observe that the all-zero sequence must be a codeword. Therefore, *the minimum distance of the code C is the codeword of smallest weight.*

Comment on circuits to implement encoding.

We have a vector space of dimension k embedded in a vector space of dimension n, the set of all n-tuples. Associated with every linear block code generator G is a matrix H called the parity check matrix whose rows span the nullspace of G. Then if $\mathbf{c}$ is a codeword, then

$$\mathbf{c}H^T = 0.$$

That is, a codeword is orthogonal to each row of H. From this we observe that

$$GH^T = \mathbf{0}.$$

There is also associated with each code a **dual code** that has H as its generator matrix. The dual code is denoted as $C^\perp$. If G is the generator for an (n, k) code, then H is the generator for an $(n, n-k)$ code.

Example 1 A $(7, 4)$ Hamming code can be generated by

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

The 16 codewords are

$$\begin{array}{c} 0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,1\,1\,1\,1 \\ 0\,0\,1\,0\,1\,1\,0 \\ 0\,0\,1\,1\,0\,0\,1 \\ 0\,1\,0\,0\,1\,0\,1 \\ 0\,1\,0\,1\,0\,1\,0 \\ 0\,1\,1\,0\,0\,1\,1 \\ 0\,1\,1\,1\,1\,0\,0 \\ 1\,0\,0\,0\,0\,1\,1 \\ 1\,0\,0\,1\,1\,0\,0 \\ 1\,0\,1\,0\,1\,0\,1 \\ 1\,0\,1\,1\,0\,1\,0 \\ 1\,1\,0\,0\,1\,1\,0 \\ 1\,1\,0\,1\,0\,0\,1 \\ 1\,1\,1\,0\,0\,0\,0 \\ 1\,1\,1\,1\,1\,1\,1 \end{array}$$

The parity check matrix is

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

When regarded as a generator of an $(7, 3)$ code, the codewords of this code, the dual code has the codewords

$$\begin{array}{c} 0\,0\,0\,0\,0\,0\,0 \\ 1\,1\,0\,1\,0\,0\,1 \\ 1\,0\,1\,1\,0\,1\,0 \\ 0\,1\,1\,0\,0\,1\,1 \\ 0\,1\,1\,1\,1\,0\,0 \\ 1\,0\,1\,0\,1\,0\,1 \\ 1\,1\,0\,0\,1\,1\,0 \\ 0\,0\,0\,1\,1\,1\,1 \end{array}$$

It may be verified that every codeword in C is orthogonal to every codeword in $C^{\perp}$ □

When we want to do the encoding, it is often convenient to have the original data explicitly evident in the codeword. Coding of this sort is called *systematic encoding*. For the codes that we are to talk about, it will always be possible to determine a generator matrix in such a way the encoding is systematic: simply perform row reductions and column reordering on G until an identity matrix is revealed. We can thus write G as

$$G = [P|I_k]$$

where I_k is the $k \times k$ identity matrix and P is $k \times n-k$. . Then

$$\mathbf{c} = \mathbf{m}G = \mathbf{m}[P|I_k] = [c_0\ c_1\ \ldots\ c_{n-k-1}|m_0\ m_1 \ldots\ m_k]$$

When G is systematic, it is easy to determine the parity check matrix H. It is simply

$$H = [I_{n-k}| - P^T].$$

Note: in $GF(2)$ (binary operations) the negative of a number is simply the number. We could write (for binary codes)

$$H = [I_{n-k}\ P^T].$$

The parity check matrix (whether systematic or not) can be used to get some useful information about the code.

Theorem 1 *Let a linear block code C have a parity check matrix H. The minimum distance of C is equal to the smallest positive number of columns of H which are linearly dependent.*

This concept should be distinguished from that of *rank*, which is the *largest* number of columns of H which are linearly independent.

Proof Let the columns of H be designated as $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{n-1}$. Then since $\mathbf{c}H^T = 0$ for any codeword $\mathbf{c}$, we have

$$0 = c_0\mathbf{d}_0 + c_1\mathbf{d}_1 + \cdots + c_{n-1}\mathbf{d}_{n-1}$$

Let $\mathbf{c}$ be the codeword of smallest weight, $w = w(\mathbf{c}) = d_{\min}$. Then the columns of H corresponding to the elements of $\mathbf{c}$ are linearly dependent. □

Based on this, we can determine a bound on the distance of a code:

$$d_{\min} \leq n - k + 1. \qquad \text{The Singleton bound}$$

This follows since H has $n-k$ linearly independent rows. (The row rank = the column rank.) So *any* combination of $n-k+1$ columns of H must be linearly dependent.

For a received vector $\mathbf{r}$, the **syndrome** is

$$\mathbf{s} = \mathbf{r}H^T.$$

Obviously, for a codeword the syndrome is equal to zero. We can determine if a received vector is a codeword. Furthermore, the syndrome is independent of the transmitted codeword. If $\mathbf{r} = \mathbf{c} + \mathbf{e}$,

$$\mathbf{s} = (\mathbf{c} + \mathbf{e})H^T = \mathbf{e}H^T.$$

Furthermore, if two error vectors $\mathbf{e}$ and $\mathbf{e}'$ have the same syndrome, then the error vectors must differ by a nonzero codeword. That is, if

$$\mathbf{e}H^T = \mathbf{e}'H^T$$

then

$$(\mathbf{e} - \mathbf{e}')H^T = 0$$

which means they must be a codeword.

2 Maximum likelihood detection

Before talking about decoding, we should introduce a probabilistic criterion for decoding, and show that it is equivalent to finding the closest codeword. Given a received vector $\mathbf{r}$, the decision rule that minimizes the probability of error is to find that codeword $\mathbf{c}_i$ which maximizes $P(\mathbf{c} = \mathbf{c}_i|\mathbf{r})$. This is called the *maximum a posteriori* decision rule. (Proof that this minimizes probability of error is shown in the communications class.) We note by Bayes rule that

$$P(\mathbf{c}|\mathbf{r}) = \frac{P(\mathbf{c})P(\mathbf{r}|\mathbf{c})}{P(\mathbf{r})},$$

where, for example, $P(\mathbf{r})$ is the probability of observing the vector $\mathbf{r}$. Now, since $P(\mathbf{r})$ is independent of $\mathbf{c}$, maximizing $P(\mathbf{c}|\mathbf{r})$ is equivalent to maximizing

$$P(\mathbf{c})P(\mathbf{r}|\mathbf{c}).$$

If we now assume that *each codeword is chosen with equal probability*, then maximizing $P(\mathbf{c})P(\mathbf{r}|\mathbf{c})$ is equivalent to maximizing

$$P(\mathbf{r}|\mathbf{c}).$$

A codeword which is selected on the basis of maximizing $P(\mathbf{r}|\mathbf{c})$ is said to be selected according to the *maximum likelihood* criterion. We shall assume throughout the text a maximum likelihood criterion.

Let us see what this means for us.

$$P(\mathbf{r}|\mathbf{c}) = \prod_{i=1}^{n} P(r_i|c_i)$$

Assuming a BSC channel with crossover probability p, we have

$$P(r_i|c_i) = \begin{cases} 1-p & \text{if} c_i = r_i \\ p & \text{if} c_i \neq r_i \end{cases}$$

Then

$$\begin{aligned} P(\mathbf{r}|\mathbf{c}) &= \prod_{i=1}^{n} P(r_i|c_i) = (1-p)^{\#(p_i = c_i)} p^{\#(p_i \neq c_i)} \\ &= (1-p)^{n - \#(p_i \neq c_i)} p^{\#(p_i \neq c_i)} = (1-p)^n \left(\frac{p}{1-p}\right)^{d(\mathbf{c},\mathbf{r})}. \end{aligned}$$

Then if we want to maximize $P(\mathbf{r}|\mathbf{c})$, we should choose that $\mathbf{c}$ which is **closest** to $\mathbf{r}$, since $0 \leq (p/(1-p)) \leq 1$. Thus, under our assumptions, the ML criterion is the minimum distance criterion. In every case, we should choose the error vector of lowest weight.

3 The standard array and syndrome table decoding

Suppose we send $\mathbf{c}$ and we receive

$$\mathbf{r} = \mathbf{c} + \mathbf{e}.$$

Assuming that error sequences with lower weight are more probable than error sequences with higher weight (the maximum likelihood criterion), we want to determine our decoded word $\mathbf{c}'$ such that the error sequence $\mathbf{e}'$ satisfying

$$\mathbf{r} = \mathbf{c}' + \mathbf{e}'$$

has minimum weight.

One way to do this is to create a *standard array*. We form it the following way:

1. Write down a list of all possible codewords in a row, with the all-zero codeword first.
2. From the remaining n-tuples which have not already been used in the standard array, select one of smallest weight. Write this down as the *coset leader* under the all-zero codeword. On this row, add the coset leader to each codeword at the top of the column.
3. Repeat step 2 until all the n-tuples have been listed.

An example standard array for a $(7, 3)$ code is shown here, where

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

We make the following observations:

1. There are 2^k codewords (columns) and 2^n possible vectors, so there are 2^{n-k} rows in the standard array.

2. The sum of any two vectors in the same row of the standard array is a code vector.

3. No two vectors in the same row of a standard array are identical. Because otherwise we have

$$\mathbf{e}_i + \mathbf{c}_i = \mathbf{e}_i + \mathbf{c}_j, \text{ with } i \neq j$$

   which means $\mathbf{c}_i = \mathbf{c}_j$, which is impossible.

4. Every vector appears exactly once in the standard array. We know every vector must appear at least once, by the construction. If a vector appears in both the lth row and the mth row we must have

$$\mathbf{e}_l + \mathbf{c}_i = \mathbf{e}_m + \mathbf{c}_j$$

   for some i and j. Let us take $l < m$. We have

$$\mathbf{e}_m = \mathbf{e}_l + \mathbf{c}_i - \mathbf{c}_j = \mathbf{e}_l + \mathbf{c}_k$$

   for some k. This means that $\mathbf{e}_m$ is on the lth row of the array, which is a contradiction.

Each row of the standard array is called a *coset*; we will encounter the term coset in a more formal setting soon.

To decode with the standard array, we locate the received vector $\mathbf{r}$ in the standard array. Then the error sequence is the coset leader; the best guess of the transmitted word is the codeword at the top of the column. For example, if

$$\mathbf{r} = 0011011$$

then

$$\mathbf{c}' = 0011101.$$

Since we designed the standard array with the smallest error patterns as coset leaders, this is the ML decision.

As observed before, there are 2^{n-k} coset leaders. These are called the *correctable error patterns*. Fact: an (n, k) code is capable of correcting 2^{n-k} error patterns.

The standard array can be used to decode linear codes, but suffers from a major problem: the memory requirements quickly become excessive. We want to look for easier approaches.

A first step (which doesn't go far enough), is to use the syndrome in the decoding. Based on the properties of the syndrome above, all elements in a row of the standard array have the same syndrome. We therefore only need to store syndromes and their associated error patterns.

For the code whose standard array was given before, we have

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

Steps to decoding

1. Compute the syndrome, $\mathbf{s} = \mathbf{r}H^T$.
2. Look up the error pattern $\mathbf{e}$ using $\mathbf{s}$.
3. Then $\mathbf{c} = \mathbf{r} + \mathbf{e}$.

Example 2 We provide another example of the standard array, because it raises some interesting issues. For the $(5, 2)$ code with

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

the standard array is .

This code is capable of correcting all errors of one bit. In addition, there are two other errors of two bits that can be corrected. Note that the minimum distance of this code is 3. The parity check matrix is

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

The standard array using syndromes is

□

4 Hamming codes

Hamming codes are the earliest and simples example of linear block codes. The parameters are as follows, for $m \geq 2$:

code length:	$n = 2^m - 1$
Number of information symbols:	$k = 2^m - m - 1$
Number of parity symbols:	$n - k = m$
Error correcting capability:	$t = 1$

Examples are: $(3, 1), (7, 4), (15, 11)$ and $(31, 26)$ codes.

The parity check matrix of a Hamming code consists of all nonzero binary m-tuples. The columns may be ordered to correspond to a systematic code. Syndrome decoding of Hamming codes using the standard array is straightforward: the syndrome indicates which column of H corresponds to the error position.

5 Weight distributions and performance of linear codes

We pause in our development of decoding algorithms to address the question of how well linear codes can perform. We recall that we can correct up to t errors if $d^* = 2t + 1$. We can also *detect* up to $d^* - 1$ errors. It may be possible to detect some errors up to d^*, but it cannot detect all of them, because there is at least one codeword with weight d^*. We say that the *random error detecting capability* of a code is d^*.

However, it may be possible to detect a large number of error patterns with weight $\geq d^*$. Consider the following: there are 2^n possible vectors, of which 2^k are codewords. There are thus $2^n - 2^k$ error patterns which are distinct from codewords. It is possible that an error sequence is exactly equal to a codeword, in which case the error is undetectable. There are $2^k - 1$ undetectable error patterns (the all zero error does not matter). The $2^n - 2^k$ patterns which are detectable are called *detectable error patterns*. In many codes, $2^k - 1$ is much smaller than 2^n, so that only a small fraction of the error patterns are undetectable.

In carefully characterizing the performance of codes, the *weight distribution* of the code is important. Consider the codewords of a linear (n, k) code C. One of the codewords has weight 0. There may be codewords with weight 1, weight 2, and so forth. Let A_i be the number of code vectors of weight i. The numbers $A_0, A_1, \ldots, A_n$ are called the *weight distribution* of C.

Example 3 For the $(7, 4)$ Hamming code presented above,

$$A_0 = 1 \qquad A_1 = 0 \qquad A_2 = 0 \qquad A_3 = 7 \qquad A_4 = 7 \qquad A_7 = 1.$$

□

If we want to use a code for error detection, we can determine the probability of an undetected error using the weight distribution. The probability of an undetected error is the probability that an error pattern is equal to a (nonzero) code vector. Then

$$P_u(E) = \sum_{i=1}^{n} A_i p^i (1-p)^{n-i},$$

where p is the crossover probability. For the Hamming example,

$$P_u(E) = 7p^3(1-p)4 + 7p^4(1-p)^3 + p^7.$$

If $p = .01$ then $P_u(E) \approx 7 \times 10^{-6}$.

We saw in example 2 that it may be possible to correct more than the minimum number of errors in some cases. The number

$$t = \lfloor (d^* - 1)/2 \rfloor$$

is called the *random-error-correcting capability* of the code. Based upon this number, the probability of erroneous decoding is upper bounded by

$$P(E) \leq \sum_{i=t+1}^{n} \binom{n}{i} p^i (1-p)^{n-i}$$

Example 4 For the $(7, 4)$ Hamming code,

$$P(E) \leq 21p^2(1-p)^5 + 35p^3(1-p)^4 + 35p^4(1-p)^3 + 21p^5(1-p)^2 + 7p^6(1-p) + p^7.$$

When $p = 0.01$ we get $P(E) \leq 0.0020$ □

In most cases we can correct many error patterns of more than t errors. There are a total of 2^{n-k} correctable error patterns (the number of rows in the standard array).

When we know the weight distribution, we can get a more precise statement of the probability of error. We make a decoding error if and only if the error pattern is not a coset leader. Let α_i be the number of coset leaders of weight i. The numbers $\alpha_0, \alpha_1, \ldots, \alpha_n$ are called the weight distribution of the coset leaders. Using these numbers,

$$P(E) = 1 - \sum_{i=0}^{n} \alpha_i p^i (1-p)^{n-i},$$

Example 5 For the $(7, 3)$ code presented above,

$$\alpha_0 = 1 \qquad a_1 = 7 \qquad \alpha_2 = 7 \qquad \alpha_3 = 1.$$

Then

$$P(E) = 1 - (1-p)^7 - 7p(1-p)^6 - 7p^2(1-p)^5 - p^3(1-p)^4.$$

If $p = 0.01$, then $P(E) = 1.4 \times 10^{-3}$. □

There is an interesting relationship between the weight distribution of a code and a dual code. Let $A_0, A_1, \ldots, A_n$ be the weight distribution of a code C, and let $B_0, B_1, \ldots, B_n$ be the weight distribution of the dual code $C^\perp$. Also let

$$A(z) = A_0 + A_1 z + \cdots + A_n z^n.$$

This polynomial is called the *weight enumerator* for C. (Think of the Z transform of a finite series.) Let

$$B(z) = B_0 + B_1 z + \cdots + B_n z^n.$$

The following identity is known as the *MacWilliams identity*:

$$A(z) = 2^{-(n-k)}(1+z)^n B\left(\frac{1-z}{1+z}\right).$$

The MacWilliams identity can be used to compute the probability of an undetected error from a linear code from the weight distribution of its dual. It can be shown that

$$P_u(E) = 2^{-(n-k)} B(1-2p) - (1-p)^n.$$

6 Modifications to linear codes

We introduce some minor modifications to linear codes.

Definition 2 A code is **punctured** by deleting one of its parity symbols. An (n, k) code becomes an $(n-1, k)$ code. □

Definition 3 A code is **shortened** by deleting a message symbol. An (n, k) code becomes an $(n, k-1)$ code. □

Definition 4 A code is **expurgated** by deleting some of its codewords (while still maintaining the linear code properties.) □

Definition 5 A code is **extended** by adding an additional redundant coordinate, producing an $(n+1, k)$ code. □

7 Bounds on linear block codes

Thinking geometrically, around each code point is a cloud of points corresponding to non-codewords. These form a "sphere" around each of the code vectors. The **Hamming sphere** is the sphere of radius t that contains vectors a distance $\leq t$ from a codeword. The number of vectors in the Hamming sphere is denoted $V_q(n, t)$ (where $q = 2$ for binary codes). We can see that

$$V_q(n,t) = \sum_{j=0}^{t} \binom{n}{j} (q-1)^j.$$

From a decoding point of view, if a received vector $\mathbf{r}$ falls inside the Hamming sphere of a codeword, then that codeword is selected.

The **redundancy** of a code is essentially the number of parity symbols in a codeword. More precisely we have

$$r = n - \log_q M$$

where M is the number of codewords. For the codewords we have seen to this point, $r = n - k$.

Theorem 2 *(***The Hamming Bound) A t-error correcting q-ary code must have redundancy r satisfying**

$$r \geq \log_q V_q(n,t)$$

Proof Each of M spheres in C has radius t. The spheres do not overlap. The total number of points enclosed by the spheres must be $\leq q^n$. We must have

$$MV_q(n,t) \leq q^n$$

so

$$q^n/M \geq V_q(n,t)$$

from which the result follows. □

A code that satisfies the Hamming bound with equality is said to be a **perfect code.** In terms of the standard array, it means that the random error correcting capability is the best it gets: there are no leftover codewords. Hamming codes are perfect codes. (Actually, being perfect codes does not mean the codes are the best possible codes.) The set of perfect codes is actually quite limited, since (it can be shown that) the number of codewords of a q-ary code must be $M = q^k$. Possible perfect codes:

1. The set of all n-tuples, with minimum distance = 1 and $t = 0$.
2. Odd-length binary repetition codes.
3. Hamming codes (linear) or other nonlinear codes with equivalent parameters.
4. The Golay G_{23} code.
5. The G_{11} and G_{23} codes, related to quadratic residue codes.

An upper bound on the redundancy is the Gilbert bound:

Theorem 3 *There exists a t-error correcting q-ary code of length n and redundancy r that satisfies*

$$r \leq \log_q V_q(n,2t).$$

The proof is interesting, because it demonstrates a power technique in coding, the random code. It also points out that the code need not be linear.

Proof There are q^n n-tuples possible. Begin by selecting one of these at random, then delete from further consideration all vectors that are at Hamming distance less than or equal to $2t$ from the selected codeword. Repeat, selecting another codeword at random from those still available. The selection of each code word results in the deletion of at most $V_q(n,2t)$ vectors from the set of q^n possible. Continuing, at least M code vectors may be selected, where

$$M = \lceil q^n/V_q(n,2t) \rceil \geq \frac{q^n}{V_q(n,2t)}.$$

□

Of considerable theoretical interest is how families of codes perform as their block length becomes long. Consider the ratio

$$\delta = d_{\min}/n.$$

We would hope that as n gets longer, $d_{\min}$ might grow correspondingly. In exploring this behavior, let $A_q(n, d_{\min})$ be the maximum possible number of codewords for a q-ary code of length n and minimum distance $d_{\min}$. Then the number of codewords is $\log_q A_q(n, d_{\min})$, and the rate is

$$\frac{\log_q A_q(n, d_{\min})}{n}$$

We say that the *asymptotic rate* of the code is the limiting value of this:

$$a(\delta) = \limsup_{n\to\infty} \frac{\log_q A_q(n, d_{\min})}{n}$$

We will now examine a bound on $a(\delta)$.

Definition 6 Let the q-ary entropy function be defined as

$$H_q(x) = x \log_q(q-1) - x \log_q x - (1-x) \log_q(1-x)$$

for $0 < x < (q-1)/q$. □

This function has the following property, which we will not prove here (see the book)

Lemma 4 *For* $0 \le \delta \le (q-1)/q)$,

$$\lim_{n\to\infty} \frac{\log_q V_q(n, \lfloor \delta n \rfloor)}{n} = H_q(\delta).$$

We now employ this in a bound:

Theorem 5 *(Gilbert-Varsharmov bound) If* $0 \le \delta \le (q-1)/q$*, then* $a(\delta) \ge 1 - H_q(\delta)$*.*

Proof The error correction capability is

$$t = \lfloor (\delta n - 1)/2 \rfloor,$$

so that

$$V_q(n, 2t) \le V_q(n, \lfloor \delta n \rfloor).$$

Then

$$\begin{aligned} a(\delta) &= \limsup_{n\to\infty} \frac{\log_q A_q(n, \lfloor \delta n \rfloor)}{n} \\ &\ge \lim_{n\to\infty} \frac{1}{n} \log_q \left(\frac{q^n}{V_q(n, \lfloor \delta n \rfloor)} \right) \qquad \text{Gilber bound} \\ &= \lim_{n\to\infty} \left[1 - \frac{\log_q V_q(n, \lfloor \delta n \rfloor)}{n} \right] \\ &= 1 - H_q(\delta) \qquad \text{previous lemma} \end{aligned}$$

□

We also can state a lower bound:

Theorem 6 *(McEliece-Rodemich-Rumsey-Welch bound)*

$$a(\delta) \le H_2(\frac{1}{2} - \sqrt{\delta(1-\delta)}).$$

Plots of these bounds for binary codes are shown.