Introduction to Coding Theory **CMU: Spring 2010**

Notes 1: Introduction, linear codes

January 2010

Lecturer: Venkatesan Guruswami *Scribe: Venkatesan Guruswami*

The theory of error-correcting codes and more broadly, information theory, originated in Claude Shannon's monumental work A mathematical theory of communication, published over 60 years ago in 1948. Shannon's work gave a precise measure of the information content in the output of a random source in terms of its *entropy*. The *noiseless coding theorem* or the source coding theorem informally states that n i.i.d. random variables each with entropy $H(X)$ can be compressed into $n(H(X)+\epsilon)$ bits with negligible probability of information loss, and conversely compression into $n(H(X)-\epsilon)$ bits would entail almost certain information loss.

More directly relevant to this course is Shannon's *noisy coding theorem* which considered communication of a message (say consisting of k bits that are output by a source coder) on a noisy communication channel whose behavior is given by a stochastic channel law. The noisy coding theorem states that every such channel has a precisely defined real number called *capacity* that quantifies the maximum rate at which reliable communication is possible on that channel. More precisely, given a noisy channel with capacity C, if information is transmitted at rate R (which means $k = nR$ message bits are communicated in n uses of the channel), then if $R < C$ then *exist* coding schemes (comprising an encoder/decoder pair) that guarantee negligible probability of miscommunication, whereas if $R > C$, then regardless of the coding scheme, the probability of error at the receiver is bounded below by some constant (which increased as R increases). (Later, a strong converse to the Shannon coding theorem was proved, which shows that when $R > C$, the probability of miscommunication goes exponentially (in k) to 1.) Shannon's theorem was one of the early uses of the probabilistic method; it asserted the existence of good coding schemes at all rates below capacity, but did not give any efficient method to construct a good code or for that matter to verify that a certain code was good.

We will return to Shannon's probabilistic viewpoint and in particular his noisy coding theorem in a couple of lectures, but we will begin by introducing error-correcting codes from a more combinatorial/geometric viewpoint, focusing on aspects such as the minimum distance of the code. This viewpoint was pioneered by Richard Hamming in his celebrated 1950 paper Error detecting and error correcting codes. The Hamming approach is more suitable for tackling worst-case/adversarial errors, whereas the Shannon theory handles stochastic/probabilistic errors. This corresponds to a rough dichotomy in coding theory results – while the two approaches have somewhat different goals and face somewhat different limits and challenges, they share many common constructions, tools, and techniques. Further, by considering meaningful relaxations of the adversarial noise model or the requirement on the encoder/decoder, it is possible to bridge the gap between the Shannon and Hamming approaches. (We will see some results in this vein during the course.)

The course will be roughly divided into the following interrelated parts. We will begin by results on the existence and limitations of codes, both in the Hamming and Shannon approaches. This will highlight some criteria to judge when a code is good, and we will follow up with several explicit constructions of "good" codes (we will encounter basic finite field algebra during these

constructions). While this part will mostly have a combinatorial flavor, we will keep track of important algorithmic challenges that arise. This will set the stage for the algorithmic component of the course, which will deal with efficient (polynomial time) algorithms to decode some important classes of codes. This in turn will enable us to approach the absolute limits of error-correction "constructively," via explicit coding schemes and efficient algorithms (for both worst-case and probabilistic error models).

Codes, and ideas behind some of the good constructions, have also found many exciting "extraneous" applications such as in complexity theory, cryptography, pseudorandomness and explicit combinatorial constructions. (For example, in the spring 2009 offering of 15-855 (the graduate complexity theory course), we covered in detail the Sudan-Trevisan-Vadhan proof of the Impagliazzo-Wigderson theorem that P = BPP under a exponential circuit lower bound for E, based on a highly efficient decoding algorithm for Reed-Muller codes.) Depending on time, we may mention/discuss some of these applications of coding theory towards the end of the course, though given that there is plenty to discuss even restricting ourselves to primarily coding-theoretic motivations, this could be unlikely.

We now look at some simple codes and give the basic definitions concerning codes. But before that, we will digress with some recreational mathematics and pose a famous "Hat" puzzle, which happens to have close connections to the codes we will soon introduce (that's your hint, if you haven't seen the puzzle before!)

Guessing hat colors

The following puzzle made the New York Times in 2001.

15 players enter a room and a red or blue hat is placed on each person's head. The color of each hat is determined by a coin toss, with the outcome of one coin toss having no effect on the others. Each person can see the other players' hats but not his own.

No communication of any sort is allowed, except for an initial strategy session before the game begins. Once they have had a chance to look at the other hats, the players must simultaneously guess the color of their own hats or pass. The group wins the game if at least one player guesses correctly and no players guess incorrectly.

One obvious strategy for the players, for instance, would be for one player to always guess "red" while the other players pass. This would give the group a 50 percent chance of winning the prize. Can the group achieve a higher probability of winning (probability being taken over the initial random assignment of hat colors)? If so, how high a probability can they achieve?

(The same game can be played with any number of players. The general problem is to find a strategy for the group that maximizes its chances of winning the prize.)

1 A simple code

Suppose we need to store 64 bit words in such a way that they can be correctly recovered even if a single bit per word gets flipped. One way is to store each information bit by duplicating it

three times. We can thus store 21 bits of information in the word. This would permit only about a fraction $\frac{1}{3}$ of information to be stored per bit of the word. However, it would allow us to correct any single bit flip since the majority value of the three copies of the bit gives the correct value of the bit, even if one of the copies is flipped.

Hamming in 1950 introduced a code, now called the "Hamming code," which could also correct 1-bit errors using fewer redundant (or extra) bits. The code is defined in such a way that a chunk of 4 information bits x_1, x_2, x_3, x_4 gets mapped (or "encoded") to a "codeword" of 7 bits as

$$x_1,\ x_2,\ x_3,\ x_4,\ x_2 \oplus x_3 \oplus x_4,\ x_1 \oplus x_3 \oplus x_4\, x_1 \oplus x_2 \oplus x_4,$$

This transformation can equivalently be represented as a mapping from x to Gx (the operations are done modulo 2) where x is the column vector $[x_1\, x_2\, x_3\, x_4]^T$ and G is the matrix

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

It is easy to verify that two distinct 4-bit vectors x and y get mapped to codewords Gx and Gy which differ in at least 3 bits. (Define $w = x - y$ so that $w \neq 0$. Now check that for each non-zero w, Gw has at least 3 bits which are 1.) It follows that the above code can correct all single bit flips, since for any 7-bit vector there is always at most one codeword which can be obtained by a single bit flip. (As we will see soon, these codes also have the remarkable property that for $y \in \{0,1\}^7$ which is not a codeword, there is always a codeword which can be obtained by a single bit flip.)

2 Some basic definitions

Let us get a few simple definitions out of the way.

Definition 1 (Hamming distance) *The Hamming distance between two strings x and y of the same length over a finite alphabet Σ, denoted $\Delta(x, y)$, is defined as the number of positions at which the two strings differ, i.e., $\Delta(x, y) = |\{i | x_i \neq y_i\}|$. The fractional Hamming distance or relative distance between $x, y \in \Sigma^n$ is given by $\delta(x, y) = \frac{\Delta(x,y)}{n}$.*

It is trivial to check that the Hamming distance defines a metric on Σ^n.

Definition 2 (Hamming weight) *The Hamming weight of a string x over alphabet Σ is defined as the number of non-zero symbols in the string. More formally, the Hamming weight of a string $\mathrm{wt}(x) = |\{i | x_i \neq 0\}|$. Note that $\mathrm{wt}(x - y) = \Delta(x, y)$.*

Given a string $x \in \Sigma^n$, the *Hamming ball* or radius r around x is the set $\{y \in \Sigma^n \mid \Delta(x, y) \leq r\}$.

Definition 3 (Code) *An error correcting code or block code C of length n over a finite alphabet Σ is a subset of Σ^n. The elements of C are called the codewords in C, and the collection of all codewords is sometimes called a codebook.*

The alphabet of C is Σ, and if $|\Sigma| = q$, we say that C is a q-ary code. When $q = 2$, we say that C is a binary code. The length n of the codewords of C is called the block length of C.

Associated with a code is also an encoding map E which maps the message set $\mathcal{M}$, identified in some canonical way with $\{1, 2, \ldots, |C|\}$ say, to codewords belonging to Σ^n. The code is then the image of the encoding map.

We now define two crucial parameters concerning a code: its *rate* which measures the amount of redundancy introduced by the code, and its *minimum distance* which measures the error-resilience of a code quantified in terms of how many errors need to be introduced to confuse one codeword for another.

Definition 4 (Rate) *The rate of a code $C \subseteq \Sigma^n$, denoted $R(C)$, is defined by*

$$R(C) = \frac{\log |C|}{n \log |\Sigma|} \ .$$

Thus, $R(C)$ is the amount of non-redundant information per bit in codewords of C.

The dimension of C is defined to $\frac{\log |C|}{\log |\Sigma|}$; this terminology will make sense once we define linear codes shortly. Note that a q-ary code of dimension ℓ has q^ℓ codewords.

Definition 5 (Distance) *The minimum distance, or simply distance, of a code C, denoted $\Delta(C)$, is defined to be the minimum Hamming distance between two distinct codewords of C. That is,*

$$\Delta(C) = \min_{\substack{c_1, c_2 \in C \\ c_1 \neq c_2}} \Delta(c_1, c_2) \ .$$

In particular, for every pair of distinct codewords in C the Hamming distance between them is at least $\Delta(C)$.

The relative distance of C, denoted $\delta(C)$, is the normalized quantity $\frac{\Delta(C)}{n}$, where n is the block length of C. Thus, any two codewords of C differ in at least a fraction $\delta(C)$ of positions.

Example 1 *The parity check code, which maps k bits to $k + 1$ bits by appending the parity of the message bits, is an example of distance 2 code. Its rate is $k/(k + 1)$.*

Example 2 *The Hamming code discussed earlier is an example of distance 3 code, and has rate $4/7$.*

The following simple fact highlights the importance of distance of a code for correcting (worst-case) errors. The proof follows readily from the definition of minimum distance.

Lemma 6 *For a code, the following statements are equivalent:*

1. *C has minimum distance $2t+1$.*
2. *C can be used to correct all t symbol errors.*
3. *C can be used to detect all $2t$ symbol errors.*
4. *C can be used to correct all $2t$ symbol erasures. (In the erasure model, some symbols are erased and the rest are intact, and we know the locations of the erasures. The goal is to fill in the values of the erased positions, using the values of the unerased positions and the redundancy of the code.)*

3 Linear codes

A general code might have no structure and not admit any representation other than listing the entire codebook. We now focus on an important subclass of codes with additional structure called linear codes. Many of the important and widely used codes are linear.

Linear codes are defined over alphabets Σ which are finite fields. Throughout, we will denote by $\mathbb{F}_q$ the finite field with q elements, where q is a prime power. (Later on in the course, it is valuable to have a good grasp of the basic properties of finite fields and field extensions. For now, we can safely think of q as a prime, in which case $\mathbb{F}_q$ is just $\{0, 1, \ldots, q-1\}$ with addition and multiplication defined modulo q.)

Definition 7 (Linear code) *If Σ is a field and $C \subset \Sigma^n$ is a subspace of Σ^n then C is said to be a linear code.*

As C is a subspace, there exists a basis $c_1, c_2, \ldots, c_k$ where k is the dimension of the subspace. Any codeword can be expressed as the linear combination of these basis vectors. We can write these vectors in matrix form as the columns of a $n \times k$ matrix. Such a matrix is called a generator matrix.

Definition 8 (Generator matrix and encoding) *Let $C \subseteq \mathbb{F}_q^n$ be a linear code of dimension k. A matrix $G \in \mathbb{F}_q^{n \times k}$ is said to be a generator matrix for C if its k columns span C.*
The generator matrix G provides a way to encode a message $x \in \mathbb{F}_q^k$ (thought of as a column vector) as the codeword $Gx \in C \subseteq \mathbb{F}_q^n$. Thus a linear code has an encoding map $E : \mathbb{F}_q^k \to \mathbb{F}_q^n$ which is a linear transformation $x \to Gx$.

Comment: Many coding texts define the "transpose" version, where the rows of the $k \times n$ generator matrix span the code. We prefer the above definition since it is customary to treat vectors as column vectors (even in these coding texts) and it is therefore nice to multiply by vectors on the right and avoid taking transposes of vectors.

Note that a linear code admits many different generator matrices, corresponding to the different choices of basis for the code as a vector space.

Notation: A q-ary linear code of block length n and dimension k will be referred to as an $[n, k]_q$ code. Further, it the code has minimum distance d, it will be referred to as an $[n, k, d]_q$ code. When the alphabet size q is clear from the context, or not very relevant to the discussion, we omit the subscript.

Example 3 *Some simple examples of binary linear codes:*

- *The binary parity check code: This is an* $[n, n-1, 2]_2$ *code consisting of all vectors in* $\mathbb{F}_2^n$ *of even Hamming weight.*
- *The binary repetition code: This is an* $[n, 1, n]_2$ *code consisting of the two vectors* 0^n *and* 1^n*.*
- *The Hamming code discussed above is a* $[7, 4, 3]_2$ *linear code.*

Exercise 1 *Show that for a linear code* C*, its minimum distance equals the minimum Hamming weight of a nonzero codewords of* C*, i.e.,*

$$\Delta(C) = \min_{\substack{c \in C \\ c \neq \mathbf{0}}} \mathrm{wt}(c) \ .$$

Exercise 2 (Systematic or standard form) *Let* C *be an* $[n, k]_q$ *linear code. Prove that after permuting coordinates if necessary,* C *has a generator matrix of the form* $[I_k \mid G']^T$ *where* I_k *is the* $k \times k$ *identity matrix and* G' *is some* $k \times (n-k)$ *matrix.*

A generator matrix in the form $[I_k \mid G']^T$ is said to be in *systematic form*. When such a generator matrix is used for encoding, the encoding is called *systematic*: the first k symbols of the codeword are just the message symbols, and the remaining $n-k$ symbols comprise redundant check symbols. Thus, after permuting coordinates if needed, every linear code admits a systematic encoding. The above-mentioned encoding map for the $[7, 4, 3]$ Hamming code was systematic.

3.1 Parity check matrices

The following is a good way to flex your basic linear algebra muscles (Exercise 2 is a useful way to proceed):

Exercise 3 *Prove that* C *is an* $[n, k]_q$ *code if and only if there is a matrix* $H \in \mathbb{F}_q^{(n-k)\times n}$ *of full row rank such that*

$$C = \{c \in \mathbb{F}_q^n \mid Hc = \mathbf{0}\} \ .$$

In other words, C is the nullspace of H. Such a matrix H is called a *parity check matrix* for C. A linear code can thus be compactly represented by either its generator matrix or its parity check matrix. The minimum distance of a code has the following characterization in terms of the parity check matrix.

Lemma 9 *If* H *is the parity check matrix of a linear code* C*, then* $\Delta(C)$ *equals the minimum number of columns of* H *that are linearly dependent.*

3.2 Hamming code revisited

The Hamming code is best understood by the structure of its parity check matrix. This will also allow us to generalize Hamming codes to larger lengths.

We defined the $C_{\text{Ham}} = [7, 4, 3]_2$ Hamming code using generator matrix

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

If we define the matrix

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix},$$

then one can check that $HG = \mathbf{0}$ and that H is a parity check matrix for C_{Ham}. Note that H has a rather nice structure: its columns are the integers 1 to 7 written in binary.

Correcting single errors with the Hamming code: Suppose that y is a corrupted version of some (unknown) codeword $c \in C_{\text{Ham}}$, with a single bit flipped. We know by the distance property of C_{Ham} that c is uniquely determined by y. In particular, a naive method to determine c would be to flip each bit of y and check if the resulting vector is in the null space of H.

A more slick (and faster) way to correct y is as follows. We know that $y = c + e_i$ where e_i is the column vector of all zeros except a single 1 in the i'th position. Note that $Hy = H(c + e_i) = Hc + He_i = He_i =$ the ith column of H. The ith column of H is the binary representation of i, and thus this method recovers the location i of the error.

Definition 10 (Syndrome) *The vector Hy is said to be the syndrome of y.*

Generalized Hamming codes: Define H_r to be the $r \times (2^r - 1)$ matrix where column i of H_r is the binary representation of i. This matrix must contain e_1 through e_r, which are the binary representations of all powers of two from 1 to 2^{r-1}, and thus has full row rank.

Now we can define the r'th generalized Hamming code

$$C^{(r)}_{\text{Ham}} = \{c \in \mathbb{F}_2^{2^r-1} \mid H_r c = \mathbf{0}\}.$$

to be the binary code with parity check matrix H_r.

Lemma 11 *$C^{(r)}_{\text{Ham}}$ is an $[2^r - 1, 2^r - 1 - r, 3]_2$ code.*

PROOF: Since H_r has rank r, it follows that the dimension of $C^{(r)}_{\text{Ham}}$ equals r. By Lemma 9, we need to check that no two columns of H_r are linearly dependent, and there are 3 linearly dependent

columns in H_r. The former follows since the columns of H_r are all distinct. For the latter, note that the first 3 columns of H_r, being the binary representations of $1, 2, 3$, add up to 0. $\square$

Despite its simplicity, the Hamming code is amazing in that it is optimal in the following sense.

Lemma 12 *If C is a binary code of block length n and minimum distance 3, then*

$$|C| \leq \frac{2^n}{n+1} \ . \tag{1}$$

It follows that the Hamming code $C^{(r)}_{\text{Ham}}$ has the maximum possible number of codewords (and thus has largest rate) amongst all binary codes of block length $2^r - 1$ and minimum distance at least 3.

PROOF: This is a special case of the "Hamming volume" (upper) bound on the size of codes. For each $c \in C$, define its neighborhood $N(c) = \{y \in \{0,1\}^n \mid \Delta(y, c) \leq 1\}$ of all strings that differ in at most one bit from c. Since C has distance 3, by the triangle inequality we have $N(c) \cap N(c') = \emptyset$ for $c \neq c' \in C$. Therefore

$$2^n \geq |\bigcup_{c \in C} N(c)| = \sum_{c \in C} |N(c)| = |C| \cdot (n+1)$$

giving the desired upper bound on $|C|$. Note that $C^{(r)}_{\text{Ham}}$ has dimension $2^r - 1 - r$, and therefore size $2^{2^r-1}/2^r$ which meets the upper bound (1) for block length $n = 2^r - 1$. $\square$

The obvious generalization of the above argument to arbitrary distances d gives the following bound. This is called the *Hamming bound* (also sometimes called the *Volume bound*). We will meet this bound again and also discuss some more sophisticated combinatorial bounds a few lectures down.

Lemma 13 (Hamming bound) *Let C be a binary code of block length n and distance d. Then*

$$|C| \leq \frac{2^n}{\sum_{i=0}^{\lfloor \frac{d-1}{2} \rfloor} \binom{n}{i}} \ . \tag{2}$$

Note that for equality to hold in (2) something remarkable must happen. Hamming balls of radius $\lfloor \frac{d-1}{2} \rfloor$ around the codewords must cover each point in $\{0,1\}^n$ *exactly once*. Thus we would have a *perfect packing* of non-overlapping Hamming spheres that cover the full space. (There is no way to pack balls in Euclidean space in such a way, with no "holes" in between!) Codes which meet the bound (2) are therefore called *perfect codes*.

By Lemma 12, Hamming codes are perfect. It turns out that perfect codes are a rare phenomenon. The following theorem (which is beyond the scope of this course) enumerates all (binary) perfect codes:

Theorem 14 (Tietavainen and van Lint) *The following are all the binary perfect codes:*

- *The Hamming codes $C^{(r)}_{\text{Ham}}$*

- *The* $[23, 12, 7]_2$ *Golay code*
- *The trivial codes consisting of just one codeword, or the whole space, or the repetition code* $\{0^n, 1^n\}$ *for odd* n

The Golay code Gol_{23} is a remarkable object with many equivalent definitions. The following defines it as a so-called "cyclic" code: Gol_{23} consists of $(c_0, c_1, \ldots, c_{22}) \in \{0,1\}^{23}$ when the polynomial $c(X) = c_0 + c_1 X + \cdots + c_{22} X^{22}$ is divisible by $g(X) = 1 + X + X^5 + X^6 + X^7 + X^9 + X^{11}$ in the ring $\mathbb{F}_2[X]/(X^{23} - 1)$.

3.3 Dual of a code

Since a linear code is a subspace of $\mathbb{F}_q^n$ we can define its dual or orthogonal space.

Definition 15 (Dual code) *If* $C \subseteq \mathbb{F}_q^n$ *is a linear code, then its dual, denoted* $C^\perp$*, is a linear code over* $\mathbb{F}_q$ *defined as*

$$C^\perp = \{z \in \mathbb{F}_q^n \mid \langle z \cdot c \rangle = 0 \ \forall \ c \in C\}.$$

where $\langle z, c \rangle = \sum_{i=1}^n z_i c_i$ *is the dot product over* $\mathbb{F}_q$ *of vectors* z *and* c*.*

Using Exercise 3, verify the following.

Exercise 4
1. *If* C *is an* $[n, k]_q$ *linear code, then* $C^\perp$ *is an* $[n, n-k]_q$ *linear code.*
2. $(C^\perp)^\perp = C$*.*
3. *If* H *is a parity check matrix for* C*, then* H^T *is a generator matrix for* $C^\perp$*. Equivalently, if* G *is a generator matrix for* C*, then* G^T *is a parity check matrix for* $C^\perp$*.*

Unlike vector spaces over $\mathbb{R}$, where the dual (or orthogonal complement) $W^\perp$ of a subspace $W \subseteq \mathbb{R}^n$ satisfies $W \cap W^\perp = \{\mathbf{0}\}$ (and $W + W^\perp = \mathbb{R}^n$), for subspaces of $\mathbb{F}_q^n$, C and $C^\perp$ can intersect non-trivially. One can have $C^\perp \subseteq C$ (such a code C is called *self-orthogonal*) or even $C = C^\perp$ (called *self-dual codes*).

Exercise 5 *For every even* n*, give an example of a self-dual binary linear code of block length* n*.*

A rather mysterious phenomenon in coding theory is that for many constructions of good codes, their dual also has some nice properties. We will now discuss the dual of the Hamming code, a code called the *Hadamard code* which has been highly influential and played a fundamental role in the computer-scientists' study of coding theory.

3.4 Hadamard code

The dual of the Hamming code $C_{\mathrm{Ham}}^{(r)}$ has as generator matrix $G_r = (H_r)^T$, which is a $(2^r - 1) \times r$ matrix whose rows are all non-zero bit vectors of length r. This is a $[2^r - 1, r]_2$ code and is called the *simplex code*. The *Hadamard code* is obtained by adding the all-zeroes row to G_r.

Definition 16 (Hadamard code) *The binary Hadamard code* Had_r *is a* $[2^r, r]_2$ *linear code whose* $2^r \times r$ *generator matrix has all* r*-bit vectors as its rows. Thus the encoding map for the Hadamard code encodes* $x \in \mathbb{F}_2^r$ *by a string in* $\mathbb{F}_2^{2^r}$ *consisting of the dot product* $\langle x, a \rangle$ *for every* $a \in \mathbb{F}_2^r$*.*

The Hadamard code can also be defined over $\mathbb{F}_q$*, but encoding a message in* $\mathbb{F}_q^k$ *with its dot product with every vector in* $\mathbb{F}_q^k$*.*

We note that the Hadamard code is the most redundant linear code in which no two codeword symbols are equal in every codeword. Hadamard codes have excellent distance property:

Lemma 17 *The Hadamard code* Had_r *(as well as the Simplex code) has minimum distance* 2^{r-1}*. The* q*-ary Hadamard code of dimension* r *has distance* $(1 - 1/q)q^r$*.*

PROOF: We prove that for $x \neq 0$, $\langle x, a \rangle \neq 0$ for exactly 2^{r-1} (i.e., half of the) elements $a \in \mathbb{F}_2^r$. Assume for definiteness that $x_1 = 1$. Then for every a, $\langle x, a \rangle + \langle x, a + e_1 \rangle = x_1 = 1$, and therefore exactly one of $\langle x, a \rangle$ and $\langle x, a + e_1 \rangle$ equals 1. The proof for the q-ary case is similar. □

We will later see that binary codes cannot have relative distance more than $1/2$ (unless they only have a fixed constant number of codewords). Thus the relative distance of Hadamard codes is optimal, but their rate is (necessarily) rather poor.

Comment: The *first order Reed-Muller code* is a code that is closely related to the Hadamard code. Linear algebraically, it is simply the subspace spanned by the Hadamard code and the all 1's vector (i.e., the union of the Hadamard code H and its coset $H + \mathbf{1}$). It maps a message $m_1, m_2, \ldots, m_r, m_{r+1}$ to $(m_1 a_1 + \cdots + m_r a_r + m_{r+1})_{a \in \mathbb{F}_2^r}$, or equivalently the evaluations $(M(a))_{a \in \mathbb{F}_2^r}$ of the r-variate polynomial $M(X_1, X_2, \ldots, X_r) = \sum_{i=1}^{r} m_i X_i + m_{r+1}$. It is a $[2^r, r+1, 2^{r-1}]_2$ code. We will later see that no binary code of block length n and relative distance $1/2$ can have more than $2n$ codewords, so the first order Reed-Muller code is optimal in this sense.

Code families and Asymptotically good codes

The Hamming and Hadamard codes exhibit two extremes in the trade-off between rate and distance. Hamming codes have rate approaching 1 (and in fact optimal rate) but their distance is only 3. Hadamard codes have (optimal) relative distance $1/2$, but their rate approaches 0. A natural question this raises is whether there are codes which have both good rate and good relative distance (say, with neither of them approaching 0 for large block lengths).

To formulate this question formally, and also because our focus in this course is on the asymptotic behavior of codes for large block lengths, we consider code families. Specifically, we define a family of codes to be an infinite collection $\mathcal{C} = \{C_i | i \in \mathbb{N}\}$ where C_i is a q_i-ary code of block length n_i with $n_i > n_{i-1}$ and $q_i \geq q_{i-1}$. Most of the constructions of codes we will encounter in this book will naturally belong to an infinite family of codes that share the same general structure and properties (and it is usually these asymptotic properties that will often guide our constructions).

We have defined the alphabet sizes q_i of the codes in the family to also grow with i (and n_i). Code families where $q_i = q$ for all i for some fixed q will be of special interest (and turn out to be more

challenging to understand and construct). We will call such families as *code families over a fixed alphabet* or more specifically as q-ary code families.

The notions of rate and relative distance naturally extend to code families. The *rate* of an infinite family of codes $\mathcal{C}$ is defined as

$$R(\mathcal{C}) = \liminf_i \left\{ \frac{k_i}{n_i} \right\} .$$

The *(relative) distance* of a family of codes $\mathcal{C}$ equals

$$\delta(\mathcal{C}) = \liminf_i \left\{ \frac{\Delta(C_i)}{n_i} \right\} .$$

A q-ary family of codes is said to be *asymptotically good* if both its rate and relative distance are bounded away from zero, i.e., if there exist constants $R_0 > 0$ and $\delta_0 > 0$ such that $R(\mathcal{C}) \geq R_0$ and $\delta(\mathcal{C}) \geq \delta_0$.

In this terminology, the question raised above concerning (binary) codes with both good rate and good distance, can be phrased as: "Are there asymptotically good binary code families?"

For a code family, achieving a large rate and large relative distance are naturally conflicting codes, and there are fundamental trade-offs between these. We will study some of these in the course. A good deal is known about the existence (or even explicit construction) of good codes and as well as limitations of codes, but the best possible asymptotic trade-off between rate and relative distance for binary codes remains a fundamental open question. In fact, the asymptotic bounds have seen no improvement since 1977!