

Foreword

This chapter is based on lecture notes from the course CSE 545– Error-Correcting Codes: Combinatorics, Algorithms and Applications taught by Atri Rudra at University at Buffalo, SUNY.

This version is dated **February 19, 2012**. For the latest version, please go to

<http://www.cse.buffalo.edu/atri/courses/coding-theory/book/>

The material in this chapter is supported in part by the National Science Foundation under CAREER grant CCF-0844796. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF).



©Atri Rudra, 2012.

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-nd/3.0/> or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

Chapter 2

A Look at Some Nicely Behaved Codes: Linear Codes

Let us now pause for a bit and think about how we can represent a code. In general, a code $C : \{0, 1\}^k \rightarrow \{0, 1\}^n$ can be stored using $n2^k$ bits (n bits for each of the 2^k codewords). For constant rate codes, this is exponential space, which is prohibitive even for modest values of k like $k = 100$. A natural question is whether we can do better. Intuitively, the code must have some extra structure that would facilitate a succinct representation of the code. We will now look at a class of codes called *linear codes* that have more structure than general codes which leads to some other nice properties. We have already seen binary linear codes in Section 1.5, that is, $C \subseteq \{0, 1\}^n$ is linear code if for all $\mathbf{c}_1, \mathbf{c}_2 \in C$, $\mathbf{c}_1 + \mathbf{c}_2 \in C$, where the “+” denotes bit-wise EXOR.

Definition 2.0.4 (Linear Codes). *Let q be a prime power (i.e. $q = p^s$ for some prime p and integer $s \geq 1$). $C \subseteq \{0, 1, \dots, q - 1\}^n$ is a linear code if it is a linear subspace of $\{0, 1, \dots, q - 1\}^n$. If C has dimension k and distance d then it will be referred to as an $[n, k, d]_q$ or just an $[n, k]_q$ code.*

Of course the above definition is not complete because we have not defined a linear subspace yet. We do that next.

2.1 Finite Fields

To define linear subspaces, we will need to work with (finite) fields. We begin with a quick overview of fields. For a more thorough treatment refer to any standard text on algebra or the book on finite fields by Lidl and Niederreiter [30].

Informally speaking, a field is a set of elements on which one can do addition, subtraction, multiplication and division and still stay in the set. More formally,

Definition 2.1.1. *A field \mathbb{F} is given by a triple $(S, +, \cdot)$, where S is the set of elements containing special elements 0 and 1 and $+, \cdot$ are functions $\mathbb{F} \times \mathbb{F} \rightarrow \mathbb{F}$ with the following properties:*

- Closure: For every $a, b \in S$, we have both $a + b \in S$ and $a \cdot b \in S$.

- Associativity: $+$ and \cdot are associative, that is, for every $a, b, c \in S$, $a + (b + c) = (a + b) + c$ and $a \cdot (b \cdot c) = (a \cdot b) \cdot c$.
- Commutativity: $+$ and \cdot are commutative, that is, for every $a, b \in S$, $a + b = b + a$ and $a \cdot b = b \cdot a$.
- Distributivity: \cdot distributes over $+$, that is for every $a, b, c \in S$, $a \cdot (b + c) = a \cdot b + a \cdot c$.
- Identities: For every $a \in S$, $a + 0 = a$ and $a \cdot 1 = a$.
- Inverses: For every $a \in S$, there exists its unique additive inverse $-a$ such that $a + (-a) = 0$. Also for every $a \in S \setminus \{0\}$, there exists its unique multiplicative inverse a^{-1} such that $a \cdot a^{-1} = 1$.

With the usual semantics for $+$ and \cdot , \mathbb{R} (set of real number) is a field but \mathbb{Z} (set of integers) is not a field as division of two integers can give rise to a rational number. In this course, we will exclusively deal with *finite fields*. As the name suggests these are fields with a finite size set of elements. (We will overload notation and denote the size of a field $|\mathbb{F}| = |S|$.) The following is a well known result.

Theorem 2.1.2 (Size of Finite Fields). *The size of any finite field is p^s for prime p and integer $s \geq 1$.*

One example of finite fields that we have seen is the field of two elements $\{0, 1\}$, which we will denote by \mathbb{F}_2 (we have seen this field in the context of binary linear codes). For \mathbb{F}_2 , addition is the XOR operation, while multiplication is the AND operation. The additive inverse of an element in \mathbb{F}_2 is the number itself while the multiplicative inverse of 1 is 1 itself.

Let p be a prime number. Then the integers modulo p form a field, denoted by \mathbb{F}_p (and also by \mathbb{Z}_p), where the addition and multiplication are carried out $\text{mod } p$. For example, consider \mathbb{F}_7 , where the elements are $\{0, 1, 2, 3, 4, 5, 6\}$. So we have $4 + 3 \text{ mod } 7 = 1$ and $4 \cdot 4 \text{ mod } 7 = 2$. Further, the additive inverse of 4 is 3 as $3 + 4 \text{ mod } 7 = 0$ and the multiplicative inverse of 4 is 2 as $4 \cdot 2 \text{ mod } 7 = 1$.

More formally, we prove the following result.

Lemma 2.1.3. *Let p be a prime. Then $\mathbb{F}_p = (\{0, 1, \dots, p-1\}, +_p, \cdot_p, 0, 1)$ is a field.*

Proof. The properties of associativity, commutativity, distributivity and identities hold for integers and hence, they hold for \mathbb{F}_p . The closure property follows since both the “addition” and “multiplication” are done $\text{mod } p$, which implies that for any $a, b \in \{0, \dots, p-1\}$, $a +_p b, a \cdot_p b \in \{0, \dots, p-1\}$. Thus, to complete the proof, we need to prove the existence of unique additive and multiplicative inverses.

Fix an arbitrary $a \in \{0, \dots, p-1\}$. Then we claim that its additive inverse is $p - a \text{ mod } p$. It is easy to check that $a + p - a = 0 \text{ mod } p$. Next we argue that this is the unique additive inverse. To see this note that the sequence $a, a + 1, a + 2, \dots, a + p - 1$ are p consecutive numbers and thus, exactly one of them is a multiple of p , which happens for $b = p - a \text{ mod } p$, as desired.

Now fix an $a \in \{1, \dots, p-1\}$. Next we argue for the existence of a unique multiplicative inverse a^{-1} . Consider the set of numbers $\{a \cdot b\}_{b \in \{1, \dots, p-1\}}$. We claim that all these numbers are unique. To see this, note that if this is not the case, then there exist $b_1 \neq b_2 \in \{0, 1, \dots, p-1\}$ such that $a \cdot b_1 = a \cdot b_2 \pmod p$, which in turn implies that $a \cdot (b_1 - b_2) = 0 \pmod p$. Since a and $b_1 - b_2$ are non-zero numbers, which implies that p divides $a \cdot (b_1 - b_2)$. Further, since a and $|b_1 - b_2|$ are both at most $p-1$, this implies that factors of a and $(b_1 - b_2) \pmod p$ when multiplied together results in p , which is a contradiction since p is prime. Thus, this implies that there exists a unique element b such that $a \cdot b = 1 \pmod p$ and thus, b is the required a^{-1} . \square

One might think that there could be different fields with the same number of elements. However, this is not the case:

Theorem 2.1.4. *For every prime power q there is a unique finite field with q elements (up to isomorphism¹).*

Thus, we are justified in just using \mathbb{F}_q to denote a finite field on q elements.

2.2 Linear Subspaces

We are finally ready to define the notion of linear subspace.

Definition 2.2.1 (Linear Subspace). $S \subseteq \mathbb{F}_q^n$ is a linear subspace if the following properties hold:

1. For every $\mathbf{x}, \mathbf{y} \in S$, $\mathbf{x} + \mathbf{y} \in S$ where the addition is vector addition over \mathbb{F}_q (that is, do addition component wise over \mathbb{F}_q).
2. For every $a \in \mathbb{F}_q$ and $\mathbf{x} \in S$, $a \cdot \mathbf{x} \in S$ where the multiplication is over \mathbb{F}_q .

Here is a (trivial) example of a linear subspace of \mathbb{F}_5^3 :

$$S_1 = \{(0, 0, 0), (1, 1, 1), (2, 2, 2), (3, 3, 3), (4, 4, 4)\}. \quad (2.1)$$

Note that e.g. $(1, 1, 1) + (3, 3, 3) = (4, 4, 4) \in S_1$ and $2 \cdot (4, 4, 4) = (3, 3, 3) \in S_1$ as required by the definition. Here is another somewhat less trivial example of a linear subspace over \mathbb{F}_3^3 :

$$S_2 = \{(0, 0, 0), (1, 0, 1), (2, 0, 2), (0, 1, 1), (0, 2, 2), (1, 1, 2), (1, 2, 0), (2, 1, 0), (2, 2, 1)\}. \quad (2.2)$$

Note that $(1, 0, 1) + (0, 2, 2) = (1, 2, 0) \in S_2$ and $2 \cdot (2, 0, 2) = (1, 0, 1) \in S_2$ as required.

Remark 2.2.2. *Note that the second property implies that $\mathbf{0}$ is contained in every linear subspace. Further for any subspace over \mathbb{F}_2 , the second property is redundant (note that as $\mathbf{c} + \mathbf{c} = \mathbf{0}$ over \mathbb{F}_2 , the first property guarantees that $\mathbf{0} \in S$).*

Before we state some properties of linear subspaces, we state some relevant definitions.

¹An isomorphism $\phi : S \rightarrow S'$ is a map (such that $\mathbb{F} = (S, +, \cdot)$ and $\mathbb{F}' = (S', \oplus, \circ)$ are fields) where for every $a_1, a_2 \in S$, we have $\phi(a_1 + a_2) = \phi(a_1) \oplus \phi(a_2)$ and $\phi(a_1 \cdot a_2) = \phi(a_1) \circ \phi(a_2)$.

Definition 2.2.3 (Span). Given a set $B = \{\mathbf{v}_1, \dots, \mathbf{v}_\ell\}$. The span of B is the set of vectors

$$\left\{ \sum_{i=1}^{\ell} a_i \cdot \mathbf{v}_i \mid a_i \in \mathbb{F}_q \text{ for every } i \in [\ell] \right\}.$$

Definition 2.2.4 (Linear independence of vectors). We say that $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k$ are linearly independent if for every $1 \leq i \leq k$ and for every $k-1$ -tuple $(a_1, a_2, \dots, a_{i-1}, a_{i+1}, \dots, a_k) \in \mathbb{F}_q^{k-1}$,

$$\mathbf{v}_i \neq a_1 \mathbf{v}_1 + \dots + a_{i-1} \mathbf{v}_{i-1} + a_{i+1} \mathbf{v}_{i+1} + \dots + a_k \mathbf{v}_k.$$

In other words, \mathbf{v}_i is not in the span of the set $\{\mathbf{v}_1, \dots, \mathbf{v}_{i-1}, \mathbf{v}_{i+1}, \dots, \mathbf{v}_n\}$.

For example the vectors $(1, 0, 1), (1, 1, 1) \in S_2$ are linearly independent.

Definition 2.2.5 (Rank of a matrix). The rank of matrix in $\mathbb{F}_q^{k \times k}$ is the maximum number of linearly independent rows (or columns). A matrix in $\mathbb{F}_q^{k \times n}$ with rank $\min(k, n)$ is said to have full rank.

One can define the row (column) rank of a matrix as the maximum number of linearly independent rows (columns). However, it is a well-known theorem that the row rank of a matrix is the same as its column rank. For example, the matrix below over \mathbb{F}_3 has full rank:

$$G_2 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

Any linear subspace satisfies the following properties (the full proof can be found in any standard linear algebra textbook).

Theorem 2.2.6. If $S \subseteq \mathbb{F}_q^n$ is a linear subspace then

1. $|S| = q^k$ for some $k \geq 0$. The parameter k is called the dimension of S .
2. There exists $\mathbf{v}_1, \dots, \mathbf{v}_k \in S$ called basis elements (which need not be unique) such that every $\mathbf{x} \in S$ can be expressed as $\mathbf{x} = a_1 \mathbf{v}_1 + a_2 \mathbf{v}_2 + \dots + a_n \mathbf{v}_n$ where $a_i \in \mathbb{F}_q$ for $1 \leq i \leq k$. In other words, there exists a full rank $k \times n$ matrix G (also known as a generator matrix) with entries from \mathbb{F}_q such that every $\mathbf{x} \in S$, $\mathbf{x} = (a_1, a_2, \dots, a_k) \cdot G$ where

$$G = \begin{pmatrix} \leftarrow \mathbf{v}_1 \rightarrow \\ \leftarrow \mathbf{v}_2 \rightarrow \\ \vdots \\ \leftarrow \mathbf{v}_k \rightarrow \end{pmatrix}$$

3. There exists a full rank $(n-k) \times n$ matrix H (called a parity check matrix) such that for every $\mathbf{x} \in S$, $H\mathbf{x}^T = \mathbf{0}$.
4. G and H are orthogonal, that is, $G \cdot H^T = \mathbf{0}$.

Proof Sketch.

Property 1. We begin with the proof of the first property. For the sake of contradiction, let us assume that $q^k < |S| < q^{k+1}$, for some $k \geq 0$. Iteratively, we will construct a set of linearly independent vectors $B \subseteq S$ such that $|B| \geq k + 1$. Note that by the definition of a linear subspace the span of B should be contained in S . However, this is a contradiction as the size of the span of B is at least $q^{k+1} > |S|$.

To complete the proof, we show how to construct the set B in a greedy fashion. In the first step pick \mathbf{v}_1 to be any non-zero vector in S and set $B \leftarrow \{\mathbf{v}_1\}$ (we can find such a vector as $|S| > q^k \geq 1$). Now say after the step t (for some $t \leq k$), $|B| = t$. Now the size of the span of the current B is $q^t \leq q^k < |S|$. Thus there exists a vector $\mathbf{v}_{t+1} \in S \setminus B$ that is linearly independent of vectors in B . Set $B \leftarrow B \cup \{\mathbf{v}_{t+1}\}$. Thus, we can continue building B till $|B| = k + 1$, as desired.

Property 2. We first note that we can pick $B = \{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ to be any set of k linearly independent vectors. (We will argue soon that such a set exists.) This is because the span of B is contained in S . However, since $|S| = q^k$ and the span of B has q^k vectors, the two have to be the same. The argument for the existence of B follows from the greedy algorithm in the above paragraph.

Property 3. Property 3 above follows from another fact that every linear subspace S has a null space $N \subseteq \mathbb{F}^n$ such that for every $\mathbf{x} \in S$ and $\mathbf{y} \in N$, $\langle \mathbf{x}, \mathbf{y} \rangle = 0$. Further, it is known that N itself is a linear subspace of dimension $n - k$. (The claim that N is also a linear subspace follows from the following two facts: for every $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{F}_q^n$, (i) $\langle \mathbf{x}, \mathbf{y} + \mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle + \langle \mathbf{x}, \mathbf{z} \rangle$ and (ii) for any $a \in \mathbb{F}_q$, $\langle \mathbf{x}, a\mathbf{y} \rangle = a \cdot \langle \mathbf{x}, \mathbf{y} \rangle$.) In other words, there exists a generator matrix H for it. This matrix H is called the parity check matrix of S .

Property 4. The proof of this is left as an exercise. □

As examples, the linear subspace S_1 in (2.1) has as one of its generator matrices

$$G_1 = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$$

and as one of its parity check matrices

$$H_1 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}.$$

Further, the linear subspace S_2 in (2.2) has G_2 as one of its generator matrices and has the following as one of its parity check matrices

$$H_2 = \begin{pmatrix} 1 & 1 & 2 \end{pmatrix}.$$

Finally, we state another property of linear subspaces that is useful.

Lemma 2.2.7. *Given matrix G of dimension $k \times n$ that is a generator matrix of subspace S_1 and matrix H of dimension $(n - k) \times n$ that is a parity check matrix of subspace S_2 such that $GH^T = \mathbf{0}$, then $S_1 = S_2$.*

Proof. We first prove that $S_1 \subseteq S_2$. Given any $\mathbf{c} \in S_1$, there exists \mathbf{x} such that $\mathbf{c} = \mathbf{x}G$. Then,

$$\mathbf{c}H^T = \mathbf{x}GH^T = \mathbf{0},$$

which implies that $\mathbf{c} \in S_2$, as desired.

To complete the proof note that as H has full rank, its null space (or S_2) has dimension $n - (n - k) = k$ (this follows from a well known fact from linear algebra). Now as G has full rank, the dimension of S_1 is also k . Thus, as $S_1 \subseteq S_2$, it has to be the case that $S_1 = S_2$.² \square

2.3 Properties of Linear Codes

The above theorem gives two alternate characterizations of an $[n, k]_q$ linear code C :

- C is generated by its $k \times n$ generator matrix G . As an example that we have already seen, the $[7, 4, 3]_2$ Hamming code has the following generator matrix:

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

- C is also characterized by an $(n - k) \times n$ parity check matrix H . We claim that the following matrix is a parity check matrix of the $[7, 4, 3]_2$ Hamming code:

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

Indeed, it can be easily verified that $G \cdot H^T = \mathbf{0}$. Then Lemma 2.2.7 proves that H is indeed a parity check matrix of the $[7, 4, 3]_2$ Hamming code.

We now look at some consequences of the above characterizations of an $[n, k]_q$ linear code C . We started this chapter with a quest for succinct representation of a code. Note that both the generator matrix and the parity check matrix can be represented using $O(n^2)$ symbols from \mathbb{F}_q (which is much smaller than the exponential representation of a general code). More precisely,

Proposition 2.3.1. *Any $[n, k]_q$ linear code can be represented with $\min(nk, n(n - k))$ symbols from \mathbb{F}_q .*

²If not, $S_1 \subset S_2$ which implies that that $|S_2| \geq |S_1| + 1$. The latter is not possible if both S_1 and S_2 have the same dimension.

There is an encoding algorithm for C that runs in $O(n^2)$ (in particular $O(kn)$) time— given a message $\mathbf{m} \in \mathbb{F}_q^k$, the corresponding codeword $C(\mathbf{m}) = \mathbf{m} \cdot G$, where G is the generator matrix of C .

Proposition 2.3.2. *For any $[n, k]_q$ linear code, given its generator matrix, encoding can be done with $O(nk)$ operations over \mathbb{F}_q .*

There is an error-detecting algorithm for C that runs in $O(n^2)$ (in particular $O(n^2 - nk)$) time— given a received word $\mathbf{y} \in \mathbb{F}_q^n$, check if $H \cdot \mathbf{y}^T = \mathbf{0}$. This is a big improvement over the naive brute force exponential time algorithm (that goes through all possible codewords $\mathbf{c} \in C$ and checks if $\mathbf{y} = \mathbf{c}$).

Proposition 2.3.3. *For any $[n, k]_q$ linear code, given its parity check matrix, error detection can be performed in $O(n(n - k))$ operations over \mathbb{F}_q .*

Next, we look at some alternate characterizations of the distance of a linear code.

2.3.1 On the Distance of a Linear Code

We start with the following property, which we have seen for the special case of binary linear codes (Proposition 1.5.4).

Proposition 2.3.4. *For a $[n, k, d]_q$ code C ,*

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

Proof. To show that d is the same as the minimum weight we show that d is no more than the minimum weight and d is no less than the minimum weight.

First, we show that d is no more than the minimum weight. We can see this by considering $\Delta(\mathbf{0}, \mathbf{c}')$ where \mathbf{c}' is the non-zero codeword in C with minimum weight; its distance from $\mathbf{0}$ is equal to its weight. Thus, we have $d \leq wt(\mathbf{c}')$, as desired.

Now, to show that d is no less than the minimum weight, consider $\mathbf{c}_1 \neq \mathbf{c}_2 \in C$ such that $\Delta(\mathbf{c}_1, \mathbf{c}_2) = d$. Note that $\mathbf{c}_1 - \mathbf{c}_2 \in C$ (this is because $-\mathbf{c}_2 = -1 \cdot \mathbf{c}_2 \in C$, where -1 is the additive inverse of 1 in \mathbb{F}_q and $\mathbf{c}_1 - \mathbf{c}_2 = \mathbf{c}_1 + (-\mathbf{c}_2)$, which by the definition of linear codes is in C). Now note that $wt(\mathbf{c}_1 - \mathbf{c}_2) = \Delta(\mathbf{c}_1, \mathbf{c}_2) = d$, since the non-zero symbols in $\mathbf{c}_1 - \mathbf{c}_2$ occur exactly in the positions where the two codewords differ. Further, since $\mathbf{c}_1 \neq \mathbf{c}_2$, $\mathbf{c}_1 - \mathbf{c}_2 \neq \mathbf{0}$, which implies that the minimum Hamming weight of any non-zero codeword in C is at most d . \square

Next, we look at another property implied by the parity check matrix of a linear code.

Proposition 2.3.5. *For any $[n, k, d]_q$ code C with parity check matrix H , d is the minimum number of linearly dependent columns in H .*

Proof. By Proposition 2.3.4, we need to show that the minimum weight of a non-zero codeword in C is the minimum number of linearly dependent columns. Let t be the minimum number of linearly dependent columns in H . To prove the claim we will show that $t \leq d$ and $t \geq d$.

For the first direction, Let $\mathbf{c} \neq \mathbf{0} \in C$ be a codeword with $wt(\mathbf{c}) = d$. Now note that, by the definition of the parity check matrix, $H \cdot \mathbf{c}^T = \mathbf{0}$. Working through the matrix multiplication, this gives us that $\sum_{i=1}^n c_i H^i$, where

$$H = \begin{pmatrix} \uparrow & \uparrow & \dots & \uparrow & \dots & \uparrow \\ H^1 & H^2 & \dots & H^i & \dots & H^n \\ \downarrow & \downarrow & \dots & \downarrow & \dots & \downarrow \end{pmatrix}$$

and $\mathbf{c} = (c_1, \dots, c_n)$. Note that we can skip multiplication for those columns for which the corresponding bit c_i is zero, so for this to be zero, those H^i with $c_i \neq 0$ are linearly dependent. This means that $d \geq t$, as the columns corresponding to non-zero entries in \mathbf{c} are one instance of linearly dependent columns.

For the other direction, consider the minimum set of columns from $H, H^{i_1}, H^{i_2}, \dots, H^{i_t}$ that are linearly dependent. This implies that there exists non-zero elements $c'_{i_1}, \dots, c'_{i_t} \in \mathbb{F}_q$ such that $c'_{i_1} H^{i_1} + \dots + c'_{i_t} H^{i_t} = \mathbf{0}$. (Note that all the c'_{i_j} are non-zero as no set of less than t columns are linearly dependent.) Now extend $c'_{i_1}, \dots, c'_{i_t}$ to the vector \mathbf{c}' such that $c'_j = 0$ for $j \notin \{i_1, \dots, i_t\}$. Note that $\mathbf{c}' \in C$ and thus, $d \leq wt(\mathbf{c}') = t$ (where recall t is the minimum number of linearly independent columns in H). \square

2.4 Hamming Codes

We now change gears and look at the general family of linear codes due to Hamming. So far we have seen the $[7, 4, 3]_2$ Hamming code. In fact for any $r \geq 2$, there is a $[2^r - 1, 2^r - r - 1, 3]_2$ Hamming code. Thus in Section 1.5, we have seen this code for $r = 3$.

Consider the $r \times (2^r - 1)$ matrix \mathbf{H}_r over \mathbb{F}_2 , where the i th column \mathbf{H}_r^i , $1 \leq i \leq 2^r - 1$, is the binary representation of i (note that such a representation is a vector in $\{0, 1\}^r$). For example, for the case we have seen ($r = 3$),

$$\mathbf{H}_3 = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

Note that by its definition, the code that has \mathbf{H}_r as its parity check matrix has block length $2^r - 1$ and dimension $2^r - r - 1$. This leads to the formal definition of the general Hamming code.

Definition 2.4.1. *The $[2^r - 1, 2^r - r - 1]_2$ Hamming code has parity check matrix \mathbf{H}_r .*

In other words, the general $[2^r - 1, 2^r - r - 1]_2$ Hamming code is the code $\{\mathbf{c} \in \{0, 1\}^{2^r - 1} \mid \mathbf{H}_r \cdot \mathbf{c}^T = \mathbf{0}\}$.

Next we argue that the above Hamming code has distance 3 (in Proposition 1.5.2 we argued this for $r = 3$).

Proposition 2.4.2. *The Hamming code $[2^r - 1, 2^r - r - 1, 3]_2$ has distance 3.*

Proof. No two columns in \mathbf{H}_r are linearly dependent. If they were, we would have $\mathbf{H}_r^i + \mathbf{H}_r^j = \mathbf{0}$, but this is impossible since they differ in at least one bit (being binary representations of integers, $i \neq j$). Thus, by Proposition 2.3.5, the distance is at least 3. It is at most 3, since (e.g.) $\mathbf{H}_r^1 + \mathbf{H}_r^2 + \mathbf{H}_r^3 = \mathbf{0}$. \square

Now note that under the Hamming bound for $d = 3$ (Theorem 1.6.2), $k \leq n - \log_2(n + 1)$, so for $n = 2^r - 1$, $k \leq 2^r - r - 1$. Hence, the Hamming code is a perfect code. (See Definition 1.7.3.)

In Question 1.7.1, we asked which codes are perfect codes. Interestingly, the only perfect binary codes are the following:

- The Hamming codes which we just studied.
- The trivial $[n, 1, n]_2$ codes for odd n (which have 0^n and 1^n as the only codewords)³,
- Two codes due to Golay [13].

The above result was proved by van Lint [44] and Tietavainen [43].

2.5 Family of codes

Till now, we have mostly studied specific codes, that is, codes with *fixed* block lengths and dimension. The only exception was the “family” of $[2^r - 1, 2^r - r - 1, 3]_2$ Hamming codes (for $r \geq 2$) that we studied in the last section. We will see shortly that when we do an asymptotic study of codes (which is what we will do), it makes more sense to talk about a family of codes. First, we define the notion of family of codes:

Definition 2.5.1 (Family of codes). $C = \{C_i\}_{i \geq 1}$ is a family of codes where C_i is a $(n_i, k_i, d_i)_q$ code for each i (and we assume $n_{i+1} > n_i$). The rate of C is defined as

$$R(C) = \lim_{i \rightarrow \infty} \left\{ \frac{k_i}{n_i} \right\}.$$

The relative distance of C is defined as

$$\delta(C) = \lim_{i \rightarrow \infty} \left\{ \frac{d_i}{n_i} \right\}.$$

For example, C_H the family of Hamming code is a family of codes with $n_i = 2^i - 1$, $k_i = 2^i - i - 1$, $d_i = 3$ and thus,

$$R(C_H) = \lim_{i \rightarrow \infty} 1 - \frac{i}{2^i - 1} = 1,$$

and

$$\delta(C_H) = \lim_{i \rightarrow \infty} \frac{3}{2^i - 1} = 0.$$

³The proof that this code is perfect is left as an exercise.

We will mostly work with family of codes from now on. This is necessary as we will study the asymptotic behavior of algorithms for codes, which does not make sense for a fixed code. For example, when we say we say that a decoding algorithm for a code C takes $O(n^2)$ time, we would be implicitly assuming that C is a family of codes and that the algorithm has an $O(n^2)$ running time when the block length is large enough. From now on, unless mentioned otherwise, whenever we talk about a code, we will be implicitly assuming that we are talking about a family of codes.

Given that we can only formally talk about asymptotic run time of algorithms, we now also state our formal notion of efficient algorithms:

We'll call an algorithm related to a code of block length n to be efficient, if it runs in time polynomial in n .

For all the specific codes that we will study in this book, the corresponding family of codes will be a "family" in a more natural sense. In other words, all the specific codes in a family of codes will be the "same" code except with different parameters. A bit more formally, we will consider families $\{C_i\}_i$, where given i , one can compute a sufficient description of C_i efficiently.⁴

Finally, the definition of a family of code allows us to present the final version of the the big motivating question for the book. The last formal version of the main question we considered was Question 1.4.1, where we were interested in the tradeoff of rate R and distance d . The comparison was somewhat "unfair" because R was a ratio while d was an integer. A more appropriate comparison should be between rate R and the relative distance δ . Further, we would be interested in tackling in the main motivating question for families of codes, which results in the following final version:

Question 2.5.1. *What is the optimal tradeoff between $R(C)$ and $\delta(C)$ that can be achieved by some code family C ?*

2.6 Efficient Decoding of Hamming codes

We have shown that Hamming code has distance of 3 and thus, by Proposition 1.4.3 can correct one error. However, this is a *combinatorial* result and does not give us an efficient algorithm. One obvious candidate for decoding is the MLD function. Unfortunately, the only implementation of MLD that we know is the one in Algorithm 1, which will take time $2^{\Theta(n)}$, where n is the block length of the Hamming code. However, we can do much better. Consider the following simple algorithm: given the received word \mathbf{y} , first check if it is indeed a valid codeword. If so, we are done. Otherwise, we flip each of the n bits and check if the resulting vector is a valid codeword. If so, we have successfully decoded from one error. (If neither happens, then

⁴We stress that this is not *always* going to be the case. In particular, we will consider "random" codes where this efficient constructibility will not be true.

we declare a decoding failure.) Algorithm 2 formally presents this algorithm (where $C_{H,r}$ is the $[2^r - 1, 2^r - r - 1, 3]_2$ Hamming code).⁵

Algorithm 2 Naive Decoder for Hamming Code

INPUT: Received word \mathbf{y}

OUTPUT: \mathbf{c} if $\Delta(\mathbf{y}, \mathbf{c}) \leq 1$ else Fail

```

1: IF  $\mathbf{y} \in C_{H,r}$  THEN
2:   RETURN  $\mathbf{y}$ 
3: FOR  $i = 1 \dots n$  DO
4:    $\mathbf{y}' \leftarrow \mathbf{y} + \mathbf{e}_i$                                 ▷  $\mathbf{e}_i$  is the  $i$ th standard vector
5:   IF  $\mathbf{y}' \in C_{H,r}$  THEN
6:     RETURN  $\mathbf{y}'$ 
7: RETURN Fail

```

It is easy to check that Algorithm 2 can correct up to 1 error. If each of the checks $\mathbf{y}' \in C_{H,r}$ can be done in $T(n)$ time, then the time complexity of the proposed algorithm will be $O(nT(n))$. Note that since $C_{H,r}$ is a linear code (and dimension $k = n - O(\log n)$) by Proposition 2.3.3, we have $T(n) = O(n \log n)$. Thus, the proposed algorithm has running time $O(n^2 \log n)$.

Note that Algorithm 2 can be generalized to work for any linear code C with distance $2t + 1$ (and hence, can correct up to t errors): go through all possible error vector \mathbf{z} (with $wt(\mathbf{z}) \leq t$) and check if $\mathbf{y} - \mathbf{z}$ is in the code or not. Algorithm 3 presents the formal algorithm (where C is an $[n, k, 2t + 1]_q$ code). The number of error patterns \mathbf{z} considered by Algorithm 3 is⁶

Algorithm 3 Decoder for Any Linear Code

INPUT: Received word \mathbf{y}

OUTPUT: $\mathbf{c} \in C$ if $\Delta(\mathbf{y}, \mathbf{c}) \leq t$ else Fail

```

1: FOR  $i = 0 \dots t$  DO
2:   FOR  $S \subseteq [n]$  such that  $|S| = i$  DO
3:     FOR  $\mathbf{z} \in \mathbb{F}_q^n$  such that  $wt(\mathbf{z}_S) = wt(\mathbf{z}) = i$  DO
4:       IF  $\mathbf{y} - \mathbf{z} \in C$  THEN
5:         RETURN  $\mathbf{y} - \mathbf{z}$ 
6: RETURN Fail

```

$\sum_{i=0}^t \binom{n}{i} (q-1)^i \leq O((nq)^t)$. Further by Proposition 2.3.3, Step 4 can be performed with $O(n^2)$ operations over \mathbb{F}_q . Thus, Algorithm 3 runs in with $O(n^{t+2} q^t)$ operations over \mathbb{F}_q , which for q

⁵Formally speaking, a decoding algorithm should return the transmitted message \mathbf{x} but Algorithm 2 actually returns $C_{H,r}(\mathbf{x})$. However, since $C_{H,r}$ is a linear code, it is not too hard to see that one can obtain \mathbf{x} from $C_{H,r}(\mathbf{x})$ in $O(n^2)$ time—the details are left as an exercise.

⁶Recall (1.14).

being polynomial small in n , is $n^{O(t)}$ operations. In other words, the algorithm will have polynomial running time for codes with constant distance (though the running time would not be practical even for moderate values of t).

However, it turns out that for Hamming codes there exists a decoding algorithm with an $O(n^2)$ running time. To see this first note that if the received word \mathbf{y} has no errors then $H_r \cdot \mathbf{y}^T = \mathbf{0}$. If not, $\mathbf{y} = \mathbf{c} + \mathbf{e}_i$, where $\mathbf{c} \in C$ and \mathbf{e}_i which is the unit vector with the only nonzero element at the i -th position. Thus, if H_r^i stands for the i -th column of H_r ,

$$H_r \cdot \mathbf{y}^T = H_r \cdot \mathbf{c}^T + H_r \cdot (\mathbf{e}_i)^T = H_r \cdot (\mathbf{e}_i)^T = H_r^i,$$

where the second equality follows as $H_r \cdot \mathbf{c}^T = \mathbf{0}$, which in turn follows from the fact that $\mathbf{c} \in C$. In other words, $H_r \cdot \mathbf{y}^T$ gives the *location* of the error. This leads to Algorithm 4.

Algorithm 4 Efficient Decoder for Hamming Code

INPUT: Received word \mathbf{y}

OUTPUT: \mathbf{c} if $\Delta(\mathbf{y}, \mathbf{c}) \leq 1$ else Fail

- 1: $\mathbf{b} \leftarrow H_r \cdot \mathbf{y}^T$.
 - 2: Let $i \in [n]$ be the number whose binary representation is \mathbf{b}
 - 3: IF $\mathbf{y} - \mathbf{e}_i \in C_H$ THEN
 - 4: RETURN $\mathbf{y} - \mathbf{e}_i$
 - 5: RETURN Fail
-

Since Step 1 in Algorithm 4 is a matrix vector multiplication (which can be done in $O(n \log n)$ time as the matrix is $O(\log n) \times n$) and Step 3 by Proposition 2.3.3 can be performed in $O(n \log n)$ time, Algorithm 4 runs in $O(n \log n)$ time. Thus,

Theorem 2.6.1. *The $[n = 2^r - 1, 2^r - r - 1, 3]_2$ Hamming code is 1-error correctable. Further, decoding can be performed in time $O(n \log n)$.*

2.7 Dual of a Linear Code

Till now, we have thought of the parity check matrix as defining a code via its null space. However, we are not beholden to think of the parity check matrix in this way. A very natural question to ask is what happens if we think of the parity check matrix as a generator matrix? The following definition addresses this question.

Definition 2.7.1 (Dual of a code). *Let H be the parity check matrix of C , then the code generated by H is called the dual of C and is denoted by C^\perp .*

It is obvious from the definition that if C is an $[n, k]_q$ code then C^\perp is an $[n, n - k]_q$ code. The first example that might come to mind is $C_{H,r}^\perp$, which is also known as the *Simplex code* (we will denote it by $C_{Sim,r}$). Adding an all 0's column to H_r and using the resulting matrix as

a generating matrix, we get the *Hadamard* code (we will denote it by $C_{Had,r}$). We claim that $C_{Sim,r}$ and $C_{Had,r}$ are $[2^r - 1, r, 2^{r-1}]_2$ and $[2^r, r, 2^{r-1}]_2$ codes respectively. The claimed block length and dimension follow from the definition of the codes, while the distance follows from the following result.

Proposition 2.7.2. $C_{Sim,r}$ and $C_{Had,r}$ both have a distance of 2^{r-1} .

Proof. We first show the result for $C_{Had,r}$. In fact, we will show something stronger: every non-zero codeword in $C_{Had,r}$ has weight exactly 2^{r-1} (the claimed distance follows from Proposition 2.3.4). Consider a message $\mathbf{x} \neq \mathbf{0}$. Let its i th entry is $x_i = 1$. \mathbf{x} is encoded as

$$\mathbf{c} = (x_1, x_2, \dots, x_r)(H_r^0, H_r^1, \dots, H_r^{2^r-1}),$$

where H_r^j is the binary representation of $0 \leq j \leq 2^r - 1$ (that is, it contains all the vectors in $\{0, 1\}^r$). Further note that the j th bit of the codeword \mathbf{c} is $\langle \mathbf{x}, H_r^j \rangle$. Group all the columns of the generating matrix into pairs (\mathbf{u}, \mathbf{v}) such that $\mathbf{v} = \mathbf{u} + \mathbf{e}_i$ (i.e. \mathbf{v} and \mathbf{u} are the same except in the i th position). Notice that this partitions all the columns in 2^{r-1} disjoint pairs. Then,

$$\langle \mathbf{x}, \mathbf{v} \rangle = \langle \mathbf{x}, \mathbf{u} + \mathbf{e}_i \rangle = \langle \mathbf{x}, \mathbf{u} \rangle + \langle \mathbf{x}, \mathbf{e}_i \rangle = \langle \mathbf{x}, \mathbf{u} \rangle + x_i = \langle \mathbf{x}, \mathbf{u} \rangle + 1.$$

Thus we have that exactly one of $\langle \mathbf{x}, \mathbf{v} \rangle$ and $\langle \mathbf{x}, \mathbf{u} \rangle$ is 1. As the choice of the pair (\mathbf{v}, \mathbf{u}) was arbitrary, we proved that for any non-zero codeword $\mathbf{c} \in C_{Had}$, $wt(\mathbf{c}) = 2^{r-1}$.

For the simplex code, we observe that all codewords of $C_{Had,3}$ are obtained by padding a 0 to the codewords in $C_{Sim,r}$, which implies that all non-zero codewords in $C_{Sim,r}$ also have a weight of 2^{r-1} , which completes the proof. \square

We remark that the family of Hamming code has a rate of $1/2$ and a (relative) distance of $1/2$ while the family of Simplex/Hadamard codes have a rate of 0 and a relative distance of $1/2$. Notice that both code families either have rate or relative distance equal to 0 . Given this, the following question is natural special case of Question 2.5.1:

Question 2.7.1. *Does there exist a code family C such that $R(C) > 0$ and $\delta(C) > 0$ hold simultaneously?*

Codes that have the property above are called *asymptotically good*.