1 Definitions

Code A code of **block length** n over an **alphabet** Σ is a subset of Σ^n [1].

Ex:

- alphabet $\Sigma = \{0, 1\}$ and block length $n = 2$ a code could be $C = \{00, 11\}$.
- alphabet $\Sigma = \{0, 1, 2\}$ and block length $n = 1$ a code could be $C = \{1, 2\}$.

Alternative view: Mapping from messages to codewords. Code C is the image of the mapping.

message space: $\{0, 1\}^3$ *codeword space:* $\{0, 1\}^4$

Alphabet An *alphabet* Σ is a finite non-empty set of symbols.

Block Length The *block length* refers to the *length* of code words in a code

Dimension The *Dimension* of a code $C \subset \Sigma^n$ is $k = \log_q |C|$ where $q = |\Sigma|$

Ex: We have the image of a mapping $\{0, 1\}^3 \to \{0, 1\}^4$

- **# of codewords:** 2^3
- **Dimension:** $\log_{q=2} 2^3 = 3$

Rate The *Rate* of a code with *dimension* k and *block length* n is $R = \frac{k}{n}$ [2]

Note: higher *Rate* $\implies$ lower *Redundancy*

Hamming Distance [3] Let $u, v \in \Sigma^n$. The *Hamming Distance* between $u\&v$ is $\triangle(u, v) = |\{i \in [n] : u_i \neq v_i\}$

Error Correction Let $C \in \Sigma^n$ be a code, and let $t \geq 1$. C is said to be *t-error-correcting* if $\exists$ deconding algorithm D s.t

$$\forall c \in C \quad \forall y \in \Sigma^n \text{ s.t } \triangle(c, y) \leq t \quad D(y) = c$$

Basically all codewords $y \in \Sigma^n$ with distance t to some codeword in C are mapped to that codeword. Here error is the *Hamming Distance*

Error Detection $C \subset \Sigma^n$ is *t-error-detecting* if $\exists$ detector alogrithm D s.t

$$\forall c \in C \quad \forall y \in \Sigma^n \text{ s.t } \triangle(c, y) \leq t \quad D(y) = \begin{cases} 0 & \text{if } c = y \\ 1 & \text{if } c \neq y \end{cases}$$

Basically we can detect if the distance bewtween recieved and sent code is less t that some error occured

Distance of C The *distance* of code C is

$$d = min_{c \neq c' \in C} \triangle(c, c')$$

[1] We define $q = |\Sigma|$

[2] R is always between 0&1 (measure of *density* of information within codewards)

[3] *Relative Hamming Distance* defined as $\delta(u, v) = \frac{1}{n}\triangle(u, v)$

2 Linear Codes

In this chapter, we introduce structure to our codes by requiring them to be linear subspaces. This allows for more compact representations and often leads to more efficient encoding and decoding algorithms.

2.1 Finite Fields and Linear Algebra Basics

To define linear codes, we first need an alphabet that supports arithmetic operations.

Finite Field A **finite field**, denoted $\mathbb{F}_q$, is a finite set of q elements that comes with addition, subtraction, multiplication, and division operations satisfying standard arithmetic rules (associativity, commutativity, distributivity, identities, and inverses).

- A common example is the field $\mathbb{F}_p$ of integers modulo a prime number p.
- The size q of any finite field must be a prime power, i.e., $q = p^r$ for some prime p and integer $r \geq 1$.

Vector Space over $\mathbb{F}_q$ The set of all n-tuples of elements from $\mathbb{F}_q$ is denoted $\mathbb{F}_q^n$. This forms a vector space where vector addition and scalar multiplication are performed component-wise over $\mathbb{F}_q$.

Linear Subspace A subset $C \subseteq \mathbb{F}_q^n$ is a **linear subspace** if it is closed under addition and scalar multiplication. That is:

- For any $\vec{c}_1, \vec{c}_2 \in C$, their sum $\vec{c}_1 + \vec{c}_2$ is also in C.
- For any $\vec{c} \in C$ and any scalar $\alpha \in \mathbb{F}_q$, the product $\alpha\vec{c}$ is also in C.
- A consequence is that the all-zero vector $\vec{0}$ is always in any linear subspace.

2.2 Definition and Properties of Linear Codes

Linear Code A **linear code** C is a linear subspace of $\mathbb{F}_q^n$.

- If the subspace has dimension k, we say it is an $[n, k]$ code.
- If we also know its minimum distance is d, we call it an $[n, k, d]$ code. The square brackets distinguish linear codes from general $(n, |C|, d)$ codes.

Generator Matrix (G) An $[n, k]$ linear code C can be described as the row span of a $k \times n$ matrix G whose rows are linearly independent. This is called a **generator matrix**.

$$C = \{\vec{m}G \mid \vec{m} \in \mathbb{F}_q^k\}$$

Any message $\vec{m}$ of length k can be encoded into a codeword $\vec{c}$ of length n by the matrix multiplication $\vec{c} = \vec{m}G$.

Parity-Check Matrix (H) An $[n, k]$ linear code C can also be described as the null space of an $(n-k) \times n$ matrix H of full rank. This is called a **parity-check matrix**.

$$C = \{\vec{c} \in \mathbb{F}_q^n \mid H\vec{c}^T = \vec{0}\}$$

This means a vector $\vec{c}$ is a valid codeword if and only if it satisfies the $n-k$ linear constraints (parity checks) defined by the rows of H. For any generator matrix G and parity-check matrix H for the same code, it holds that $GH^T = \mathbf{0}$.

2.3 Distance of Linear Codes

The linearity of the codes simplifies the calculation of their distance.

Hamming Weight The **Hamming weight** of a vector $\vec{c}$, denoted $wt(\vec{c})$, is the number of non-zero components in $\vec{c}$. Note that $wt(\vec{c}) = \triangle(\vec{c}, \vec{0})$.

Theorem 1. *The minimum distance d of a non-trivial linear code C is equal to the minimum Hamming weight of any non-zero codeword in C.*

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

Proof. Let $\vec{c}_1, \vec{c}_2 \in C$ be two distinct codewords. Since C is a linear subspace, their difference $\vec{c}_1 - \vec{c}_2$ is also a non-zero codeword in C. The distance is $\triangle(\vec{c}_1, \vec{c}_2) = wt(\vec{c}_1 - \vec{c}_2)$. Therefore, the minimum distance between any two distinct codewords is the same as the minimum weight of any non-zero codeword. □

Theorem 2. *The minimum distance d of a linear code C with parity-check matrix H is the smallest integer d such that there exist d linearly dependent columns in H.*

Proof. A codeword $\vec{c}$ exists if and only if $H\vec{c}^T = \vec{0}$. Let the columns of H be $\vec{h}_1, \ldots, \vec{h}_n$. Then $H\vec{c}^T = \sum_{i=1}^{n} c_i \vec{h}_i$. A non-zero codeword $\vec{c}$ of weight d has exactly d non-zero components, say at indices $i_1, \ldots, i_d$. The condition $H\vec{c}^T = \vec{0}$ becomes $\sum_{j=1}^{d} c_{i_j} \vec{h}_{i_j} = \vec{0}$. This is a linear dependency among the columns $\vec{h}_{i_1}, \ldots, \vec{h}_{i_d}$. The minimum distance d corresponds to the smallest set of columns that are linearly dependent. □

2.4 Dual Codes

Dual Code For an $[n, k]$ linear code C, its **dual code**, denoted $C^\perp$, is the set of all vectors in $\mathbb{F}_q^n$ that are orthogonal to every codeword in C.

$$C^\perp = \{\vec{v} \in \mathbb{F}_q^n \mid \vec{v} \cdot \vec{c} = 0 \text{ for all } \vec{c} \in C\}$$

If C has generator matrix G and parity-check matrix H, then $C^\perp$ is an $[n, n-k]$ code with generator matrix H and parity-check matrix G.

2.5 Families of Codes & Asymptotics

A key goal in coding theory is to find families of codes that have both a high rate ($R = k/n$) and a high relative distance ($\delta = d/n$) as the block length n goes to infinity. We seek families of $[n_i, k_i, d_i]_q$ codes where $n_i \to \infty$ and the ratios $R_i = k_i/n_i$ and $\delta_i = d_i/n_i$ approach desirable limits.

2.6 Hamming Codes

Hamming codes are a famous family of linear codes that are "perfect," meaning they achieve the maximum possible number of codewords for their distance.

Hamming Code For any integer $r \geq 2$, the binary **Hamming code** $\text{Ham}(r, 2)$ is a code with parameters:

- Block length: $n = 2^r - 1$
- Dimension: $k = 2^r - 1 - r$
- Minimum distance: $d = 3$

The parity-check matrix H for this code is an $r \times (2^r - 1)$ matrix whose columns consist of all non-zero binary vectors of length r. Since any two columns are distinct and non-zero, no two columns are linearly dependent, but many sets of three columns are (e.g., if $\vec{h}_i + \vec{h}_j = \vec{h}_k$), so the distance is exactly 3.

Decoding Hamming Codes Hamming codes have a very efficient decoding algorithm. Given a received vector $\vec{y}$, we compute the **syndrome** $\vec{s} = H\vec{y}^T$.

- If $\vec{s} = \vec{0}$, then $\vec{y}$ is a codeword and we assume no error occurred.
- If a single error occurred at position i, so that $\vec{y} = \vec{c} + \vec{e}_i$, then the syndrome is $\vec{s} = H(\vec{c} + \vec{e}_i)^T = H\vec{c}^T + H\vec{e}_i^T = H\vec{e}_i^T$. This result is precisely the i-th column of H.

By computing the syndrome, we can directly identify the column corresponding to the error position and correct it by flipping the bit.

3 Probability & Entropy

So far, we have studied the deterministic properties of codes. However, coding theory is fundamentally about reliable communication over unreliable channels, which requires a probabilistic perspective. In this chapter, we introduce essential tools from probability and information theory that allow us to analyze the performance of codes and prove the existence of "good" codes.

3.1 Essential Probabilistic Tools

We begin with two fundamental inequalities that are used throughout coding theory to bound the probabilities of error events.

Union Bound This is a simple but incredibly useful principle. It states that the probability of at least one of several events occurring is no greater than the sum of their individual probabilities.

Formally, for any set of events $A_1, A_2, \ldots, A_m$,

$$P(A_1 \cup A_2 \cup \cdots \cup A_m) \leq \sum_{i=1}^{m} P(A_i)$$

In coding theory, we often use this to bound the overall probability of a decoding error by summing the probabilities of individual error events (e.g., the probability that a random noise vector transforms the sent codeword into another specific codeword).

Chernoff Bound The Chernoff bound provides an exponentially decreasing upper bound on the probability that a sum of independent random variables deviates significantly from its expected value. It is much stronger than looser bounds like Markov's or Chebyshev's inequality.

A common form used in coding theory relates to the sum of independent Bernoulli trials. Let $X_1, \ldots, X_n$ be independent random variables where $P(X_i = 1) = p$ and $P(X_i = 0) = 1 - p$. Let $X = \sum_{i=1}^{n} X_i$. The expected value is $E[X] = np$. The Chernoff bound states that for any $\delta > 0$,

$$P(X \geq (1 + \delta)np) \leq e^{-D(p(1+\delta)||p)n}$$

and

$$P(X \leq (1 - \delta)np) \leq e^{-D(p(1-\delta)||p)n}$$

where $D(a||p) = a \log \frac{a}{p} + (1 - a) \log \frac{1-a}{1-p}$ is the Kullback-Leibler (KL) divergence. This bound is crucial for proving that random codes are, with high probability, good codes.

3.2 Entropy: A Measure of Uncertainty

The concept of entropy, borrowed from statistical mechanics, was adapted by Claude Shannon to quantify the uncertainty or "information content" of a random variable.

Intuition Imagine a random event. If the event is highly predictable (e.g., a loaded coin that almost always comes up heads), its outcome provides very little new information. If the event is highly unpredictable (e.g., a fair coin flip), its outcome is surprising and provides more information. Entropy measures this average uncertainty.

Binary Entropy For a binary random variable that takes the value 1 with probability p and 0 with probability $1-p$, the **binary entropy function**, $H_2(p)$, measures its uncertainty in bits.

$$H_2(p) = -p\log_2(p) - (1-p)\log_2(1-p)$$

- If $p = 0$ or $p = 1$, the outcome is certain, and $H_2(p) = 0$. There is no uncertainty.
- If $p = 0.5$, the outcome is maximally uncertain, and $H_2(0.5) = 1$ bit.

The base of the logarithm determines the units; for coding theory, we almost always use base 2.

q-ary Entropy The concept generalizes to an alphabet of size q. If a random variable can take q different values with probabilities $p_1, \ldots, p_q$, its entropy is $H_q(p_1, \ldots, p_q) = -\sum_{i=1}^{q} p_i \log_q(p_i)$.

A special case relevant to us is when we are interested in the probability of a symbol being one specific value (with probability p) versus any of the other $q-1$ values (with total probability $1-p$). The entropy in this context is often written as:

$$H_q(p) = -p\log_q(p) - (1-p)\log_q(1-p) + p\log_q(q-1)$$

This form will be instrumental in estimating the size of Hamming balls.

3.3 The Volume of a Hamming Ball

The Hamming ball is a central concept for understanding error correction. The ability of a code to correct errors is directly related to whether the Hamming balls around its codewords are disjoint.

Definition The **Hamming ball** of radius r around a vector $\vec{c} \in \mathbb{F}_q^n$, denoted $B(\vec{c}, r)$, is the set of all vectors $\vec{y} \in \mathbb{F}_q^n$ such that the Hamming distance $\triangle(\vec{c}, \vec{y}) \leq r$.

The **volume** of this ball, denoted $Vol_q(r, n)$, is the number of vectors it contains. Since the space is symmetric, this volume does not depend on the center $\vec{c}$. We can calculate it by counting the number of vectors with weight at most r:

$$Vol_q(r, n) = |B(\vec{0}, r)| = \sum_{i=0}^{r} \binom{n}{i} (q-1)^i$$

The term $\binom{n}{i}$ chooses i positions to have errors, and $(q-1)^i$ accounts for the $q-1$ possible non-zero symbols that can appear in each of those positions.

Approximating the Volume Calculating the sum above is cumbersome for large n. Fortunately, information theory provides a beautiful and powerful approximation. The largest term in the sum dominates the value, and using Stirling's approximation for the binomial coefficient, we arrive at the following theorem.

Theorem 3 (Volume of a Hamming Ball)**.** *For an alphabet of size $q \geq 2$, an error probability $0 \leq p \leq 1 - 1/q$, and large block length n, the volume of a Hamming ball of radius pn is approximately:*

$$Vol_q(pn, n) \approx q^{nH_q(p)}$$

where $H_q(p)$ is the q-ary entropy function.

This result is profound. It connects a geometric property (the volume of a ball) to an information-theoretic one (entropy). It tells us that the number of typical error patterns of a certain density p occupies a "volume" whose size is determined by the entropy. This approximation is the cornerstone of the Gilbert-Varshamov bound and proofs about the capacity of noisy channels.

4 Upper & Lower Bounds on Codes

A central question in coding theory is: for a given block length n and alphabet size q, what is the best trade-off between the number of codewords (or the rate, R) and the error-correcting capability (the distance, d)? This chapter explores the fundamental limits on these parameters.

We will study two types of bounds:

- **Upper Bounds:** These establish a ceiling on how good a code can be. They tell us that no code with parameters (n, d) can have more than a certain number of codewords. These are impossibility results.
- **Lower Bounds:** These establish a floor on how good a code can be. They guarantee the existence of a code with at least a certain number of codewords for a given (n, d). These are existence results, often proven by showing a construction method.

4.1 Upper Bounds (Limits on Performance)

4.1.1 The Singleton Bound

The Singleton bound is one of the simplest upper bounds. It connects the dimension, block length, and distance of any code, not just linear ones.

Theorem 4 (Singleton Bound)**.** *For any $(n, M, d)_q$ code,*

$$M \leq q^{n-d+1}$$

For any linear $[n, k, d]_q$ code, this is equivalent to:

$$d \leq n - k + 1 \quad or \quad k \leq n - d + 1$$

Proof. The proof is surprisingly simple. Take any code C with M codewords. Now, "puncture" the code by deleting the first $d-1$ coordinates from every codeword. Let the original codewords be $\vec{c} = (c_1, c_2, \ldots, c_n)$. Let the new, shortened codewords be $\vec{c}' = (c_d, c_{d+1}, \ldots, c_n)$.

Let's consider two distinct original codewords, $\vec{c}_i$ and $\vec{c}_j$. By definition, their Hamming distance is $\triangle(\vec{c}_i, \vec{c}_j) \geq d$. This means they differ in at least d positions.

When we delete the first $d-1$ coordinates, they can differ in at most $d-1$ of these deleted positions. Therefore, they must still differ in at least one of the remaining $n - (d-1)$ positions. This means the new, shortened codewords $\vec{c}_i'$ and $\vec{c}_j'$ must still be distinct.

So, after puncturing, we have M distinct codewords of length $n-d+1$. The total number of possible vectors of this new length is q^{n-d+1}. Since all our new codewords must fit into this space, we must have:

$$M \leq q^{n-d+1}$$

For a linear code, $M = q^k$, so $q^k \leq q^{n-d+1}$, which simplifies to $k \leq n-d+1$. □

Codes that meet the Singleton bound with equality are called **Maximum Distance Separable (MDS)** codes.

4.1.2 The Hamming Bound (Sphere Packing Bound)

This bound comes from a simple geometric idea: if a code is to correct t errors, the Hamming balls of radius t around each codeword must be disjoint.

Theorem 5 (Hamming Bound)**.** *Let C be an $(n, M, d)_q$ code. Let $t = \lfloor \frac{d-1}{2} \rfloor$ be the number of errors it can correct. Then:*

$$M \cdot Vol_q(t, n) \leq q^n$$

where $Vol_q(t, n) = \sum_{i=0}^{t} \binom{n}{i}(q-1)^i$ is the volume of a Hamming ball of radius t.

Proof. Consider the M codewords in C. Around each codeword $\vec{c}_i$, draw a Hamming ball $B(\vec{c}_i, t)$ of radius t. If a received word $\vec{y}$ is in $B(\vec{c}_i, t)$, it will be decoded to $\vec{c}_i$. For this decoding to be unambiguous, all these balls must be disjoint. If a vector $\vec{y}$ were in two balls, say $B(\vec{c}_i, t)$ and $B(\vec{c}_j, t)$, then $\triangle(\vec{y}, \vec{c}_i) \leq t$ and $\triangle(\vec{y}, \vec{c}_j) \leq t$. By the triangle inequality, this would mean $\triangle(\vec{c}_i, \vec{c}_j) \leq \triangle(\vec{c}_i, \vec{y}) + \triangle(\vec{y}, \vec{c}_j) \leq 2t$. But we know $d \geq 2t + 1$, so this is a contradiction. Thus, the balls are disjoint.

The volume of each ball is $Vol_q(t, n)$. Since there are M such disjoint balls, their total volume is $M \cdot Vol_q(t, n)$. This total volume cannot exceed the volume of the entire space $\mathbb{F}_q^n$, which is q^n. The inequality follows. □

Asymptotic Form: For large n, using our approximation $Vol_q(t, n) \approx q^{nH_q(t/n)}$, and setting $t/n \approx d/(2n) = \delta/2$, the Hamming bound becomes:

$$q^k \cdot q^{nH_q(\delta/2)} \lesssim q^n$$

Taking $\log_q$ of both sides and dividing by n gives an upper bound on the rate $R = k/n$:

$$R \leq 1 - H_q(\delta/2)$$

Codes that meet the Hamming bound with equality are called **perfect codes**. Hamming codes are an example.

4.1.3 The Plotkin Bound

The Plotkin bound is particularly effective for codes with a very large minimum distance.

Theorem 6 (Plotkin Bound, Binary Case)**.** *For a binary $(n, M, d)_2$ code where $d > n/2$,*

$$M \leq \frac{2d}{2d - n}$$

Proof. Let the codewords be $\vec{c}_1, \ldots, \vec{c}_M$. Consider the sum of all pairwise Hamming distances:

$$S = \sum_{1 \leq i < j \leq M} \triangle(\vec{c}_i, \vec{c}_j)$$

Since every pairwise distance is at least d, we have a lower bound on S:

$$S \geq \binom{M}{2} d = \frac{M(M-1)}{2} d$$

Now, let's find an upper bound on S. Let's write the M codewords as rows of a matrix. Let w_j be the number of 1s in column j. Then the number of 0s in column j is $M - w_j$. The number of pairs that differ in column j is $w_j(M - w_j)$. This quantity is maximized when $w_j = M/2$, giving a maximum value of $M^2/4$. Summing over all columns:

$$S = \sum_{j=1}^{n} w_j(M - w_j) \leq \sum_{j=1}^{n} \frac{M^2}{4} = \frac{nM^2}{4}$$

Combining the lower and upper bounds on S:

$$\frac{M(M-1)}{2} d \leq \frac{nM^2}{4}$$

$$2(M-1)d \leq nM$$

$$2Md - 2d \leq nM$$

$$M(2d - n) \leq 2d$$

Since we assumed $d > n/2$, the term $(2d - n)$ is positive, so we can divide by it:

$$M \leq \frac{2d}{2d - n}$$

□

4.2 Lower Bounds (Guarantees of Existence)

4.2.1 The Gilbert-Varshamov Bound

This is the most famous lower bound and provides a constructive proof for the existence of good codes.

Theorem 7 (Gilbert-Varshamov Bound)**.** *There exists an* $(n, M, d)_q$ *code with*

$$M \geq \frac{q^n}{Vol_q(d-1, n)}$$

Proof. We prove this by a greedy construction.

1. **Initialize:** Start with an empty code, $C = \emptyset$.
2. **Step 1:** Choose any vector $\vec{c}_1$ from $\mathbb{F}_q^n$ and add it to C.
3. **Step 2:** Choose any vector $\vec{c}_2$ such that $\triangle(\vec{c}_2, \vec{c}_1) \geq d$. Add $\vec{c}_2$ to C.
4. **Step i:** Choose a vector $\vec{c}_i$ such that $\triangle(\vec{c}_i, \vec{c}_j) \geq d$ for all previously chosen codewords $\vec{c}_j \in C$ $(j < i)$. Add $\vec{c}_i$ to C.
5. **Termination:** Continue this process until no more such vectors can be found.

The final code C has minimum distance at least d by construction. Let's find a lower bound on its size, $M = |C|$.

The process terminates when every vector $\vec{v} \in \mathbb{F}_q^n$ is within a distance of $d-1$ of at least one codeword in our constructed code C. In other words, the union of all Hamming balls of radius $d-1$ centered at our codewords covers the entire space:

$$\bigcup_{\vec{c} \in C} B(\vec{c}, d-1) = \mathbb{F}_q^n$$

The volume of the union is less than or equal to the sum of the individual volumes:

$$q^n = \left| \bigcup_{\vec{c} \in C} B(\vec{c}, d-1) \right| \leq \sum_{\vec{c} \in C} |B(\vec{c}, d-1)| = M \cdot Vol_q(d-1, n)$$

Rearranging this gives the desired bound: $M \geq q^n / Vol_q(d-1, n)$. □

Asymptotic Form: Using the volume approximation $Vol_q(d-1, n) \approx q^{nH_q((d-1)/n)}$, and setting $(d-1)/n \approx \delta$, the GV bound becomes:

$$q^k \gtrsim \frac{q^n}{q^{nH_q(\delta)}}$$

Taking $\log_q$ and dividing by n gives a lower bound on the rate R:

$$R \geq 1 - H_q(\delta)$$

For Linear Codes: A similar argument shows the existence of a linear $[n, k, d]_q$ code satisfying the same asymptotic bound. One can construct the parity-check matrix H greedily, adding columns one by one, ensuring that no $d-1$ columns are linearly dependent.

4.3 Summary of Bounds

The bounds define a region in the space of (R, δ) where good codes can exist.

Bound	**Type**	**Asymptotic Form (R vs. δ)**
Singleton	Upper	$R \leq 1 - \delta$
Hamming	Upper	$R \leq 1 - H_q(\delta/2)$
Plotkin	Upper	$R \leq 1 - 2\delta$ (for $q = 2, \delta > 1/2$)
Gilbert-Varshamov	Lower	$R \geq 1 - H_q(\delta)$

Which bound is best?

- The **Singleton bound** is the most general but often the weakest.
- The **Hamming bound** is better than the Singleton bound for small δ.
- The **Plotkin bound** is very strong for large δ (it's better than Singleton for $\delta > 1 - 1/q$), showing that high-rate codes cannot have very large distances.
- The **Gilbert-Varshamov bound** is the best known lower bound for general codes over a wide range of parameters. For many decades, it was the benchmark for existence proofs.

5 Stochastic Channels & Channel Capacity

So far, we have designed codes to combat a fixed number of errors. In reality, errors occur randomly. To understand the ultimate limits of communication, we must model the communication channel itself as a probabilistic system. This chapter introduces the foundational models for noisy channels and the concept of channel capacity, which defines the maximum rate of reliable communication.

5.1 Modeling Communication Channels

A communication channel is a system that takes an input symbol from an alphabet $\mathcal{X}$ and produces an output symbol from an alphabet $\mathcal{Y}$. Due to noise, the output may be different from the input. We model this relationship with conditional probabilities.

Discrete Memoryless Channel A channel is **discrete** if the alphabets $\mathcal{X}$ and $\mathcal{Y}$ are finite. It is **memoryless** if the probability of receiving a certain output symbol depends only on the corresponding input symbol, not on any previous inputs or outputs. Mathematically, for an input sequence $\vec{x} = (x_1, \ldots, x_n)$ and output sequence $\vec{y} = (y_1, \ldots, y_n)$:

$$P(\vec{y}|\vec{x}) = \prod_{i=1}^{n} P(y_i|x_i)$$

We will focus on these simple but powerful models.

5.1.1 The Binary Symmetric Channel (BSC)

The most fundamental model for a noisy channel is the Binary Symmetric Channel, or BSC.

Definition The **Binary Symmetric Channel**, denoted BSC_p, has a binary input and output alphabet ($\mathcal{X} = \mathcal{Y} = \{0, 1\}$). It is defined by a single parameter, the **crossover probability** p.

- With probability $1 - p$, the input bit is transmitted correctly.
- With probability p, the input bit is flipped (a "crossover" occurs).

We assume $0 \leq p \leq 1/2$. If $p > 1/2$, we could simply flip all received bits to get an equivalent channel with crossover probability $1 - p < 1/2$.

The transition probabilities are:

$$P(Y = 0|X = 0) = 1 - p \qquad P(Y = 1|X = 0) = p$$
$$P(Y = 1|X = 1) = 1 - p \qquad P(Y = 0|X = 1) = p$$

5.1.2 The q-ary Symmetric Channel (qSC)

This is a direct generalization of the BSC to a non-binary alphabet.

Definition The **q-ary Symmetric Channel**, denoted qSC_p, has an input and output alphabet of size q, i.e., $\mathcal{X} = \mathcal{Y} = \{0, 1, \ldots, q - 1\}$.

- With probability $1-p$, the input symbol is transmitted correctly.
- With probability p, an error occurs. When an error occurs, the input symbol is changed to one of the other $q-1$ symbols with equal probability. Thus, the probability of it changing to any specific incorrect symbol is $p/(q-1)$.

5.2 Channel Capacity

The central question of information theory, answered by Claude Shannon, is: what is the maximum rate at which we can communicate over a noisy channel with an arbitrarily low probability of error? This rate is the channel capacity.

Mutual Information To define capacity, we first need the concept of **mutual information**, $I(X;Y)$. It measures the amount of information that the output Y provides about the input X. It is the reduction in uncertainty about X gained by observing Y.

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

Where:

- $H(Y)$ is the entropy (total uncertainty) of the output.
- $H(Y|X)$ is the conditional entropy, representing the uncertainty that remains about the output Y even when we know the input X. This is the uncertainty caused purely by the channel's noise.

So, $I(X;Y)$ is what's left when we subtract the noise from the total output uncertainty.

Channel Capacity The channel capacity, C, is the maximum possible mutual information between the input and output, where the maximization is over all possible input distributions $p(x)$.

$$C = \max_{p(x)} I(X;Y)$$

Capacity is a property of the channel itself, not of any specific code. It is the "speed limit" for reliable communication over that channel, measured in bits per channel use.

5.2.1 Capacity of the BSC

Let's compute the capacity of the BSC_p. We want to maximize $I(X;Y) = H(Y) - H(Y|X)$.

1. **Calculate $H(Y|X)$:** This is the entropy of the noise. If we know the input $X = x$, the output Y is a random variable that is flipped with probability p. The entropy of this process is simply the binary entropy function $H_2(p)$. Since this is true for any input x, the average conditional entropy is $H(Y|X) = H_2(p)$.

2. **Maximize $H(Y)$:** Our expression is now $I(X;Y) = H(Y) - H_2(p)$. To maximize this, we must maximize $H(Y)$. The output Y is a binary variable, so its entropy is at most 1. This maximum is achieved when the output is uniformly distributed, i.e., $P(Y = 0) = P(Y = 1) = 1/2$.

3. **Find the Maximizing Input Distribution:** An output is uniform if the input is uniform. If we choose $P(X = 0) = P(X = 1) = 1/2$, then $P(Y = 1) = P(Y = 1|X = 0)P(X = 0) + P(Y = 1|X = 1)P(X = 1) = p(1/2) + (1 - p)(1/2) = 1/2$. So a uniform input gives a uniform output.

4. **Result:** With a uniform input, $H(Y) = 1$. Therefore, the capacity is:

$$C_{BSC} = 1 - H_2(p)$$

This result is beautiful. It says the capacity is 1 (the max possible for a binary channel) minus a penalty term that is exactly the entropy of the noise process. If $p = 0$, $C = 1$. If $p = 0.5$ (total noise), $H_2(0.5) = 1$ and $C = 0$.

5.2.2 Capacity of the qSC

The logic is identical. We want to maximize $I(X;Y) = H(Y) - H(Y|X)$.

1. **Calculate $H(Y|X)$:** If we know the input $X = x$, the output is x with probability $1 - p$, and each of the other $q - 1$ symbols with probability $p/(q - 1)$. The entropy of this distribution is:

$$H(Y|X = x) = -(1 - p)\log_2(1 - p) - (q - 1)\frac{p}{q - 1}\log_2\left(\frac{p}{q - 1}\right)$$

$$= -(1 - p)\log_2(1 - p) - p\log_2(p) + p\log_2(q - 1) = H_2(p) + p\log_2(q - 1)$$

This is the entropy of the noise for the q-ary channel.

2. **Maximize $H(Y)$:** We need to maximize $H(Y)$. The output alphabet has size q, so the maximum possible entropy is $\log_2(q)$, which is achieved when the output distribution is uniform. This, in turn, is achieved by a uniform input distribution.

3. **Result:** The capacity of the q-ary symmetric channel is:

$$C_{qSC} = \log_2(q) - (H_2(p) + p\log_2(q - 1))$$

Again, the capacity is the maximum possible output entropy ($\log_2(q)$) minus a penalty term for the noise.

6 Reed-Solomon Codes

Reed-Solomon (RS) codes are a remarkable family of non-binary, linear codes that achieve the Singleton bound, making them Maximum Distance Separable (MDS). Their algebraic structure, based on polynomials over finite fields, allows for powerful and efficient decoding algorithms. They are among the most widely used codes in practice, found in everything from QR codes and data storage (CDs, DVDs) to deep-space communication.

6.1 Construction and Properties

The core idea of Reed-Solomon codes is to encode a message by interpreting it as the coefficients of a polynomial, and then evaluating that polynomial at several points.

Construction Let $\mathbb{F}_q$ be a finite field of size q.

1. **Parameters:** Choose a block length $n \leq q$ and a dimension $k < n$.
2. **Message:** A message is a vector $\vec{m} = (m_0, m_1, \ldots, m_{k-1}) \in \mathbb{F}_q^k$.
3. **Encoding Polynomial:** Associate the message $\vec{m}$ with a polynomial $P_m(x)$ of degree less than k:
$$P_m(x) = m_0 + m_1 x + m_2 x^2 + \cdots + m_{k-1} x^{k-1}$$
4. **Evaluation Points:** Choose n distinct evaluation points $\alpha_1, \alpha_2, \ldots, \alpha_n$ from $\mathbb{F}_q$.
5. **Codeword:** The corresponding codeword $\vec{c}$ is the evaluation of $P_m(x)$ at these points:
$$\vec{c} = (P_m(\alpha_1), P_m(\alpha_2), \ldots, P_m(\alpha_n)) \in \mathbb{F}_q^n$$

The set of all such codewords forms the Reed-Solomon code, denoted $RS[n, k]_q$.

Theorem 8 (Properties of RS Codes)**.** *The Reed-Solomon code $RS[n, k]_q$ as constructed above is a linear $[n, k, d]_q$ code with minimum distance $d = n-k+1$.*

Proof. **Linearity:** The mapping from message $\vec{m}$ to codeword $\vec{c}$ is a linear transformation, so the resulting code is a linear subspace of $\mathbb{F}_q^n$. The dimension is clearly k, as there are k message symbols.

Distance: We need to find the minimum distance, which for a linear code is the minimum weight of any non-zero codeword. A non-zero codeword corresponds to a non-zero message polynomial $P_m(x)$ of degree at most $k-1$. The components of the codeword are the evaluations $P_m(\alpha_i)$. The weight of the codeword is the number of non-zero components.

A fundamental property of polynomials states that a non-zero polynomial of degree at most $k-1$ can have at most $k-1$ roots. Therefore, $P_m(x)$ can be zero for at most $k-1$ of the evaluation points α_i. This means the codeword must have at most $k-1$ zero-components.

The weight of the codeword is the number of non-zero components, so:

$$wt(\vec{c}) = n - (\text{number of zero components}) \geq n - (k-1) = n - k + 1$$

Thus, the minimum distance is $d = n - k + 1$. This meets the Singleton bound ($d \leq n - k + 1$) with equality, proving that RS codes are MDS. □

6.2 Unique Decoding: The Berlekamp-Welch Algorithm

The Berlekamp-Welch algorithm provides an efficient way to correct errors in RS codes, as long as the number of errors is not too large. It can uniquely correct up to $t = \lfloor (d-1)/2 \rfloor = \lfloor (n-k)/2 \rfloor$ errors.

The Setup: We sent a codeword $\vec{c} = (P(\alpha_1), \ldots, P(\alpha_n))$ but received a word $\vec{y} = (y_1, \ldots, y_n)$ which may have up to t errors. Let $E \subset \{1, \ldots, n\}$ be the set of error locations, where $|E| \leq t$. We know:

- $y_i = P(\alpha_i)$ if $i \notin E$ (correct position)
- $y_i \neq P(\alpha_i)$ if $i \in E$ (error position)

The Key Idea: Instead of trying to find $P(x)$ directly, we search for two polynomials:

1. An **error-locator polynomial** $E(x)$ of degree at most t. We define it to have roots at the error locations: $E(x) = \prod_{j \in E}(x - \alpha_j)$.
2. An auxiliary polynomial $Q(x) = P(x)E(x)$. Its degree is at most $(k-1)+t$.

These two polynomials are linked by a key equation that holds for *all* positions $i = 1, \ldots, n$:

$$y_i E(\alpha_i) = Q(\alpha_i)$$

Proof. Case 1 ($i \notin E$): This is a correct position. $y_i = P(\alpha_i)$. So, $y_i E(\alpha_i) = P(\alpha_i)E(\alpha_i) = Q(\alpha_i)$. Case 2 ($i \in E$): This is an error position. By definition, $E(\alpha_i) = 0$. So, $y_i E(\alpha_i) = 0$. And $Q(\alpha_i) = P(\alpha_i)E(\alpha_i) = P(\alpha_i) \cdot 0 = 0$. The equation holds in both cases. □

The Algorithm:

1. **Set up linear equations:** The equation $y_i E(\alpha_i) = Q(\alpha_i)$ is a linear equation in the unknown coefficients of $Q(x)$ and $E(x)$. We have n such equations, one for each received symbol. Let $Q(x) = q_0 + q_1 x + \cdots + q_{k-1+t}x^{k-1+t}$ and $E(x) = e_0 + e_1 x + \cdots + e_t x^t$. To make the system homogeneous, we can set the leading coefficient of $E(x)$ to 1 (monic). The number of unknowns is $(k+t)$ for $Q(x)$ and t for $E(x)$ (since $e_t = 1$), for a total of $k+2t$ unknowns. The number of equations is n. Since $d = n-k+1$ and $t = \lfloor (d-1)/2 \rfloor$, we have $n \geq k + 2t$. This means we have enough equations to find a unique solution.
2. **Solve the system:** Find a non-zero solution for the coefficients of $Q(x)$ and $E(x)$.
3. **Recover P(x):** Once $Q(x)$ and $E(x)$ are found, the original message polynomial is recovered by polynomial division:

   $$P(x) = \frac{Q(x)}{E(x)}$$

   If the division has a remainder or if $E(x)$ is the zero polynomial, then more than t errors occurred and the algorithm fails. Otherwise, the resulting $P(x)$ is the decoded message.

6.3 List Decoding: The Sudan Algorithm

What if more than $\lfloor (n-k)/2 \rfloor$ errors occur? Unique decoding is no longer possible. List decoding offers a powerful alternative: instead of returning one codeword, it returns a small list of all codewords that are close to the received word.

The Sudan algorithm finds all polynomials $P(x)$ of degree $< k$ that agree with the received word $\vec{y}$ on at least T positions, where T can be much smaller than the $n - t$ required for unique decoding.

Algorithm 1: Interpolation Step The goal is to find a non-zero bivariate polynomial $Q(x, y)$ that has a root at every received point.

1. **Goal:** Find a non-zero polynomial $Q(x, y)$ such that for all $i = 1, \ldots, n$:

$$Q(\alpha_i, y_i) = 0$$

2. **Method:** We define $Q(x, y) = \sum_{j=0}^{d_x} \sum_{l=0}^{d_y} c_{j,l} x^j y^l$. The conditions $Q(\alpha_i, y_i) = 0$ give n linear equations in the unknown coefficients $c_{j,l}$.

3. **Ensuring a Solution:** The number of unknown coefficients is $(d_x + 1)(d_y + 1)$. If we choose the degrees d_x, d_y such that the number of unknowns is greater than the number of equations (n), a non-zero solution is guaranteed to exist. A standard choice is $d_y = \ell$ and $d_x = \lceil n/\ell \rceil$ for some parameter ℓ. For RS list decoding, one can choose degrees such that $(d_x + 1)(d_y + 1) > n$. For example, $d_y \approx \sqrt{n}$ and $d_x \approx \sqrt{n}$.

Algorithm 2: Root-Finding Step The key insight is that if a message polynomial $P(x)$ is a good candidate (i.e., it agrees with $\vec{y}$ on many points), then $(y - P(x))$ will be a factor of the bivariate polynomial $Q(x, y)$ we just found.

1. **Setup:** Consider the polynomial $Q(x, y)$ from the previous step as a polynomial in y with coefficients that are polynomials in x.

2. **Key Property:** If $P(x)$ is a polynomial of degree $< k$ such that $P(\alpha_i) = y_i$ for at least T values of i, and if T is large enough (specifically $T > d_x$), then $(y - P(x))$ must be a factor of $Q(x, y)$.

   Sketch. Define a new polynomial $R(x) = Q(x, P(x))$. The degree of $R(x)$ is at most $d_x + d_y(k - 1)$. For every point i where $y_i = P(\alpha_i)$, we have $R(\alpha_i) = Q(\alpha_i, P(\alpha_i)) = Q(\alpha_i, y_i) = 0$. So $R(x)$ has at least T roots. If T is greater than the degree of $R(x)$, then $R(x)$ must be the zero polynomial. This implies that $Q(x, P(x)) \equiv 0$, which means that $(y - P(x))$ is a factor of $Q(x, y)$. □

3. **Factorization:** Use a standard algorithm for polynomial factorization to find all factors of $Q(x, y)$ that have the form $(y - P(x))$ where $\deg(P) < k$.

4. **Output:** The list of all such polynomials $P(x)$ is the result of the list decoding.

7 Expander Codes

Expander codes represent a significant breakthrough in coding theory, providing the first family of codes with both a constant rate, constant relative distance (asymptotically good), and a linear-time decoding algorithm. They are constructed from a special class of highly connected, sparse graphs called expander graphs. The structure of the graph itself is the key to their remarkable properties.

7.1 Expander Graphs

The "backbone" of an expander code is a bipartite graph with strong connectivity properties.

Bipartite Graph A graph $G = (L \cup R, E)$ is bipartite if its vertices can be divided into two disjoint sets, L (left) and R (right), such that every edge in E connects a vertex in L to one in R. We will consider graphs where every vertex on the left has degree D, called D-regular.

Neighborhood For a set of vertices $S \subseteq L$, its neighborhood $\Gamma(S) \subseteq R$ is the set of all vertices in R that are connected by an edge to at least one vertex in S.

Expander Graph An $(N, M, D, \gamma, \alpha)$ expander is a D-regular bipartite graph with N vertices on the left and M vertices on the right. Its expansion property is defined as follows:

Any subset $S \subseteq L$ of size $|S| \leq \gamma N$ has a large neighborhood: $|\Gamma(S)| > \alpha D|S|$

Here, α is a constant typically close to 1 (e.g., 3/4) and γ is a constant fraction. In simple terms, any "small" set of left vertices connects to a "very large" set of right vertices. This property guarantees that there are no small, isolated parts of the graph, ensuring high connectivity.

7.2 Code Construction and Properties

Expander codes are defined by using a bipartite graph as the blueprint for a parity-check matrix.

Construction Given a D-regular bipartite graph G with N left vertices and M right vertices, we construct a binary linear code as follows:

1. The bits of the codeword correspond to the left vertices (the variable nodes). The block length is $n = N$.
2. The parity checks correspond to the right vertices (the check nodes).
3. The parity-check matrix H is the $M \times N$ bi-adjacency matrix of the graph G. An entry H_{ij} is 1 if the i-th check node (in R) is connected to the j-th variable node (in L), and 0 otherwise.

A binary vector $\vec{c} \in \{0,1\}^N$ is a codeword if and only if $H\vec{c}^T = \vec{0} \pmod 2$. This means for every check node, the sum of its neighboring variable nodes (bits) must be zero.

Theorem 9 (Properties of Expander Codes)**.** *An expander code constructed from an $(N, M, D, \gamma, \alpha)$ expander with $\alpha > 1/2$ has the following properties:*

- ***Block Length:*** *$n = N$.*
- ***Rate:*** *The number of checks is M. The rate $R = (N-M)/N = 1-M/N$. For a constant-degree expander, M is proportional to N, so the rate is a constant greater than 0.*
- ***Distance:*** *The minimum distance d is at least γN. This means the code has a constant relative distance $\delta = d/n \geq \gamma > 0$.*

Distance Sketch. The distance of the code is the size of the smallest non-zero codeword. Let $\vec{c}$ be a non-zero codeword and let S be the set of its non-zero positions (its support). Since $\vec{c}$ is a codeword, all checks are satisfied. This implies that every check node in $\Gamma(S)$ must be connected to at least two nodes in S. If a check node were connected to only one node in S, that check would fail. This "two-connections" property limits how small S can be, and using the expansion property, one can show that $|S|$ must be large (at least γN). □

7.3 The Sipser-Spielman Decoding Algorithm

The true power of expander codes lies in their incredibly simple and fast decoding algorithm. It is an iterative, message-passing algorithm that flips bits one by one until all parity checks are satisfied.

The Setup: We receive a word $\vec{y} \in \{0,1\}^n$ that may contain errors. We want to find the closest codeword $\vec{c}$.

The Algorithm:

1. **Initialization:** Let the current word be $\vec{z} = \vec{y}$.
2. **Iteration:** While there are any unsatisfied parity checks:
   (a) Identify the set of unsatisfied check nodes, $U = \{i \in R \mid (H\vec{z}^T)_i = 1\}$.
   (b) Find a variable node $j \in L$ that is "badly-behaved". A simple and effective rule is to find any variable node j that is connected to at least one unsatisfied check node in U. A stronger rule used in the analysis is to find a node j that is connected to more unsatisfied checks than satisfied ones.
   (c) Flip the corresponding bit in our working vector: $z_j \leftarrow z_j + 1 \pmod 2$.
3. **Termination:** When all checks are satisfied (the set U is empty), the algorithm terminates. The final vector $\vec{z}$ is the decoded codeword.

Theorem 10 (Correctness and Runtime)**.** *For an expander code built from a sufficiently strong expander, if the number of initial errors in $\vec{y}$ is less than $\gamma N/2$, the Sipser-Spielman algorithm is guaranteed to converge to the correct codeword in linear time (i.e., in $O(N)$ steps).*

Sketch. The core of the proof lies in showing that every bit flip makes progress. Let E be the set of error positions in the current vector $\vec{z}$ compared to the

original codeword $\vec{c}$. The unsatisfied checks are a subset of the neighborhood of E, $\Gamma(E)$.

The expansion property guarantees that the number of edges between the error set E and its unsatisfied neighbors is large. When we flip a bit z_j (where $j \in E$), we satisfy some of its unsatisfied neighbors. While this flip might dissatisfy some previously satisfied checks, the expansion property ensures that, on balance, the total number of unsatisfied checks decreases with each step. Since the number of unsatisfied checks is a positive integer that decreases with each flip, the algorithm must terminate. The linear time complexity follows from the fact that the number of initial unsatisfied checks is small and each step reduces this number. □

8 Locally Decodable Codes

In many modern applications, such as cloud storage and distributed systems, datasets are enormous. If a massive file is encoded and stored, and we only need to access one small piece of the original data (e.g., one bit of a file), it would be incredibly inefficient to download and decode the entire multi-gigabyte codeword. Locally Decodable Codes (LDCs) are designed to solve this exact problem, allowing for the recovery of individual message bits by reading only a tiny, sub-linear portion of the (potentially corrupted) codeword.

8.1 Definitions

Locally Decodable Code (LDC) An (r, δ, q)-LDC is a code $C : \Sigma^k \to \Sigma^n$ with the following property: For any message $\vec{m} \in \Sigma^k$, any desired index $i \in [k]$, and any received word $\vec{y} \in \Sigma^n$ that is δ-close to the codeword $C(\vec{m})$ (i.e., $\triangle(\vec{y}, C(\vec{m})) \leq \delta n$), there exists a randomized decoding algorithm D. The algorithm D, given index i and access to $\vec{y}$, makes at most q queries to the coordinates of $\vec{y}$ and outputs the correct message symbol m_i with high probability (e.g., $> 2/3$). The number of queries, q, is called the locality of the code.

Locally Correctable Code (LCC) The definition is nearly identical, but the goal is to recover a symbol of the *codeword*, not the message. An (r, δ, q)-LCC allows for the recovery of any codeword symbol c_i by making at most q queries to the received word $\vec{y}$. Every LCC can be turned into an LDC (by first correcting a codeword symbol and then figuring out which message symbol it corresponds to), but the reverse is not always true.

The key feature of these codes is that the locality q is a small constant (like 2 or 3), independent of the block length n.

8.2 The Hadamard Code

The Hadamard code is the canonical example of an LDC. It has a terrible rate but excellent distance and local decodability.

Construction Let the message space be $\{0,1\}^k$. We identify the message vectors $\vec{m} \in \{0,1\}^k$ with linear functions $L_{\vec{m}} : \{0,1\}^k \to \{0,1\}$ defined by the inner product: $L_{\vec{m}}(\vec{x}) = \vec{m} \cdot \vec{x} = \sum_{i=1}^{k} m_i x_i \pmod 2$. The Hadamard code encodes $\vec{m}$ into a codeword of length $n = 2^k$ by evaluating this linear function on all possible inputs $\vec{x} \in \{0,1\}^k$. The codeword is a truth table of the linear function.

$$C(\vec{m}) = (L_{\vec{m}}(\vec{x}))_{\vec{x} \in \{0,1\}^k}$$

Properties The Hadamard code is a linear $[2^k, k, 2^{k-1}]$ code. The rate is $k/2^k$, which is extremely low, but the relative distance is $1/2$, which is excellent.

Theorem 11. *The Hadamard code is a 2-query LDC.*

Proof. Suppose we want to recover the i-th bit of the message, m_i. Let $\vec{e}_i$ be the standard basis vector with a 1 in the i-th position and 0s elsewhere. The key insight comes from the linearity of the inner product:

$$L_{\vec{m}}(\vec{x}) + L_{\vec{m}}(\vec{x} + \vec{e}_i) = (\vec{m} \cdot \vec{x}) + (\vec{m} \cdot (\vec{x} + \vec{e}_i)) = \vec{m} \cdot (2\vec{x} + \vec{e}_i) = \vec{m} \cdot \vec{e}_i = m_i$$

The equation $m_i = L_{\vec{m}}(\vec{x}) + L_{\vec{m}}(\vec{x} + \vec{e}_i)$ holds for any $\vec{x} \in \{0,1\}^k$. This gives us a simple randomized decoding algorithm:

Decoding Algorithm for m_i:

1. Choose a vector $\vec{x} \in \{0,1\}^k$ uniformly at random.
2. Query the received word $\vec{y}$ at the two positions corresponding to $\vec{x}$ and $\vec{x} + \vec{e}_i$. Let the queried values be $y_{\vec{x}}$ and $y_{\vec{x}+\vec{e}_i}$.
3. Output their sum: $y_{\vec{x}} + y_{\vec{x}+\vec{e}_i} \pmod 2$.

If the received word $\vec{y}$ has no errors, this output is always correct. If there are errors, this procedure gives the correct answer as long as neither of the two queried positions are corrupted. Since the queries are chosen uniformly at random, the probability of hitting a corrupted coordinate is low, and the algorithm succeeds with high probability. □

8.3 Reed-Muller Codes

Reed-Muller codes generalize Hadamard codes by using multivariate polynomials of higher total degree. They provide a trade-off between rate and locality.

Construction The Reed-Muller code $RM(m, r)$ encodes a message by interpreting it as the coefficients of an m-variable polynomial $P(x_1, \ldots, x_m)$ over a field $\mathbb{F}_q$ with total degree at most r. The codeword is the evaluation of this polynomial on all possible q^m points in $\mathbb{F}_q^m$.

$$C(P) = (P(\vec{a}))_{\vec{a} \in \mathbb{F}_q^m}$$

(Note: The Hadamard code is the special case $RM(m, 1)$ over $\mathbb{F}_2$).

Theorem 12. *The Reed-Muller code $RM(m, r)$ is a (q, δ, q)-LCC. (A slightly different construction gives an $(r + 1)$-query LCC).*

Sketch of Local Correction. Suppose we want to correct the value of the codeword at a specific point $\vec{a} \in \mathbb{F}_q^m$. The true value should be $P(\vec{a})$.

The key idea is that the restriction of a total-degree-r multivariate polynomial P to any line is a univariate polynomial of degree at most r.

Correction Algorithm for position $\vec{a}$:

1. Choose a random line passing through the point $\vec{a}$. A line through $\vec{a}$ can be parameterized as $L(t) = \vec{a} + t\vec{v}$ for a random non-zero direction vector $\vec{v}$ and a scalar $t \in \mathbb{F}_q$.
2. Query the received word $\vec{y}$ at all q points on this line: $y_{L(0)}, y_{L(1)}, \ldots, y_{L(q-1)}$.

3. These q values are evaluations of a univariate polynomial of degree at most r. Since $q > r$, we can use a standard decoding algorithm for single-variable polynomials (like Berlekamp-Welch for Reed-Solomon codes) to find the unique polynomial $p(t)$ that agrees with the queried values on the most points.

4. The corrected value for position $\vec{a}$ is $p(0)$, since $\vec{a} = L(0)$.

If the number of errors is small enough, the random line will have few corrupted points, allowing the univariate decoding to succeed and recover the correct value at $p(0)$. The number of queries is the number of points on the line, which is q. □

8.4 Private Information Retrieval (PIR)

LDCs have a surprising and deep connection to a cryptographic problem called Private Information Retrieval.

Definition of PIR A PIR scheme allows a user to retrieve the i-th bit of a database $DB \in \{0,1\}^k$ that is replicated across N non-communicating servers, without any single server learning anything about the index i. The goal is to minimize the total communication between the user and the servers.

Connection to LDCs A q-query LDC gives a simple construction for a q-server PIR scheme.

1. **Setup:** The user treats the database DB as a message $\vec{m}$. They compute the corresponding LDC codeword $C(\vec{m})$ of length n. Each of the q servers stores this entire codeword. (This is a major simplification; more advanced schemes have servers store parts of the codeword).
2. **Query:** To retrieve the i-th bit m_i, the user runs the LDC's randomized decoding algorithm. This algorithm generates q query positions, say $j_1, j_2, \ldots, j_q$.
3. **Distribution:** The user sends the first query position j_1 to Server 1, j_2 to Server 2, ..., and j_q to Server q.
4. **Response:** Each server responds with the bit stored at its requested position.
5. **Reconstruction:** The user combines the q received bits to reconstruct m_i.

Privacy: The privacy comes from the fact that each server only sees one query position. Since the LDC decoder's queries are randomized, a single query position j_k reveals statistically nothing about the desired index i. For the scheme to be perfectly private, we need the distribution of the k-th query to be identical regardless of which index i is being decoded. LDCs with this property are sometimes called "smooth".