Contents

1 Lecture 1

1.1 What are Error-Correcting Codes?

Error-correcting codes provide a judiciously designed, noise-resilient redundant form of data.

The idea of "coding data", is robustness against noise through redundancy and the possibility of error recovery (pinpointing mistakes). Our hopes are:

- It is information-theoretically possible to recover the data
- Minimize redundancy (i.e. overhead)
- The algorithm to encode/recover the original data is efficient

Error-correcting codes have plenty of applications:

- Communication
- Storage (can be thought of communication in time)
- Central tool in discrete math, theoretical CS, information theory
- Quantum computing (heat and noise is present in every computation!)

We have many noise models in thinking about what we want our codes to protect against. We can mix and match to make a problem.

- Erasures, errors, deletions
- Bit level vs symbol/packet level
- Worst-case noise vs probabilistic/stochastic noise (or some hybrid model)
- Many desiderata/decoding models: local decoding/list decoding

The flow of a coding model is as follows.

1. Encoding: info $\to$ codeword
2. Noise: codeword $\to$ noised encoding
3. Decoding (Error Correction): noised encoding $\to$ info

The encoding map is generally injective (you can go from "valid" codeword to info easily).

1.2 One-Bit Erasures

Definition 1.1
A (binary) code is some subset $C \subseteq \{0,1\}^n$. The elements of C are called “codewords”.

Suppose our noise is such that one bit is missing (and the location of the noise is known). So the transformation was:

$$c_1c_2\dots c_n \mapsto c_1c_2\dots c_{i-1}?c_{i+1}\dots c_n$$

we want to find what was in the question mark.

A simple code is repeating the same bit twice:

$$C_{rep} = \{(c_1, \ldots, c_n) \mid c_1 = c_2, c_3 = c_4, \ldots, c_{n-1} = c_n\}$$

Then the question mark is clearly recoverable from the neighbor bit.

A better code that works with this is one with only even parity (called the parity-check code).

$$C_{pc} = \{(c_1, \ldots, c_n) \in \{0,1\}^n \mid c_1 + c_2 + \cdots + c_n \equiv 0 \pmod 2\}$$

Then, the question mark is just the parity of the rest, i.e.

$$? = c_1 + c_2 + \cdots + c_{i-1} + c_{i+1} + \cdots + c_n \bmod 2$$

The second one is better since we can represent more messages, i.e.

$$|C_{pc}| = 2^{n-1} > |C_{rep}| = 2^{\frac{n}{2}}$$

The number of redundant bits in C_{pc} is 1, while there are $n/2$ redundant bits in C_{rep}. Here are the encoding maps:

$$(x_1, \ldots, x_{n-1}) \mapsto (x_1, \ldots, x_{n-1}, \bigoplus_i x_i)$$

$$(x_1, \ldots, x_{n/2}) \mapsto (x_1, x_1, x_2, x_2 \ldots, x_{n/2}, x_{n/2})$$

Fact: C_{pc} is the BEST code to correct 1-bit erasure (it has the least overhead). Proof shell: suppose you had a code with more codewords. This means that there are some codewords that differ by at most 1 bit, but then if you erase that bit the noisy data is ambiguous.

To generalize this setup there are two models. Consider $\alpha < 1$:

1. Hamming model: worst-case, erase any αn out of n bits (zero-error info theory)
2. Shannon model: stochastic, erase each bit independently with probability α

1.3 One-Bit Errors

In this setup, one bit is flipped but we don't know which one.

$$c_1 c_2 \ldots c_n \mapsto c_1 c_2 \ldots c_{i-1} \overline{c_i} c_{i+1} \ldots c_n$$

Now, we can replicate the message 3 times as a code; we can use majority vote to correct one bit. But $\frac{2}{3}$ of the bits are redundant! This means $|C| = 2^{n/3}$. Let's see if we can be smarter.

$$C_{VT} = \{(c_1, c_2, \ldots, c_n) \mid c_1 + 2c_2 + \cdots + nc_n \equiv a \pmod B\}$$

This is a "weighted" parity check. The claim is that if $B > 2n$, then you can correct 1-bit errors. Proof shell: suppose you flip a bit, it will go up or down by at most n and this uniquely identifies which bit got flipped. If B is larger, note that there will be less codewords, so we'd prefer B to be as low as possible.

Since there are some that are bigger than average, there exists an $a \in \{0, \ldots, B-1\}$ such that $|C_{VT}| \geq \frac{2^n}{B} = \frac{2^n}{2n+1}$.

Definition 1.2 (Rate)
An **alphabet** Σ are the characters in the codeword.
The number of redundant bits is $n - \log_{|\Sigma|} |C|$. The **rate** of C is $R(C) = \frac{\log_{|\Sigma|} |C|}{n}$, i.e. the fraction of

non-redundant bits in your code. Note $|C| = |\Sigma|^{Rn}$.

So our code has $\leq \log_2 n + O(1)$ redundant bits, so a rate of nearly 1 asymbtotically.

Theorem 1.1 (Hamming Bound)
If $C \subseteq \{0,1\}^n$ corrects 1-bit flips, then $|C| \leq \frac{2^n}{n+1}$.

Here is a simple proof: consider the hamming ball of distance 1 (i.e. everything produced by 0 or 1 bit flips). These balls cannot overlap (this would imply ambiguous decoding). Thus, by pigeonhole there can only be at most $\frac{2^n}{n+1}$ such balls.

1.4 More than 1-bit flips

To correct two bit errors we can build off our old code:

$$\alpha_1 c_1 + \cdots + \alpha_n c_n \equiv m \pmod{M}$$

where $\pm\alpha_p \pm \alpha_q \neq \pm\alpha_r \pm \alpha_s$ for any 4 tuple. This is sufficient to avoid ambiguity.

We can also add a second check:

$$1^2 c_1 + 2^2 c_2 + \cdots + n^2 c_n \equiv a' \pmod{A}$$

where $A = O(n^2)$. This is a sufficient α because $f(n) = n^2$ is strictly convex. Thus, with the old check together, this makes a code strong enough. Similar to before:

$$|C| \geq \frac{2^n}{A \cdot B} = \Omega\left(\frac{2^n}{n^3}\right)$$

Theorem 1.2 (Generalized Hamming Bound)
If $C \subseteq \{0,1\}^n$ corrects (up to) k-bit flips, then $|C| \leq \frac{2^n}{\sum_{i=0}^{k}\binom{n}{k}} = O\left(\frac{2^n}{n^k}\right)$.

This follows by a similar ball argument, except you consider up to k bit errors.

1.5 Hamming Code

Let's revisit the single-bit flip case. Our "VT Check" was:

$$1c_1 + 2c_2 + \cdots + nc_n \equiv 0 \pmod{B}$$

Let's use linear algebra instead and map ic_i to $\mathbf{v_i}c_i$, thus writing

$$\mathbf{v_1}c_1 + \mathbf{v_2}c_2 + \cdots + \mathbf{v_n}c_n = \mathbf{0}$$

where we take operations mod 2.

To take a (potentially corrupted) $\mathbf{c}$ we can check the following using the parity check matrix on the left.

$$\begin{pmatrix} \mathbf{v_1} & \mathbf{v_2} & \dots & \mathbf{v_n} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \mathbf{0}$$

where m, the height of the matrix, is the total amount of bits to represent all the codewords. $m = \lceil \log_2(n+1) \rceil$.

Definition 1.3 (Code notation)
A linear $[n, k, d]_q$ code has block size n, dimension k, minimum distance d and alphabet size q. The dimension k is the dimension of C as a subspace.

If we choose $m = 3$, what we constructed above is the $[7, 4, 3]_2$ Hamming code.

We argue this achieves the Hamming bound. Suppose $n = 2^m - 1$ exactly. Note that $\dim(C_{Ham}) = 2^m - 1 - m$ (because the rank of the parity-check matrix is m) so

$$|C_{Ham}| = 2^{2^m - 1 - m} = \frac{2^n}{2^m} = \frac{2^n}{n+1}$$

Let's revisit our two-bit error correction with our Hamming knowledge. Let's write the equations in matrix vector form.

$$\begin{pmatrix} 1 & 2 & \dots & n \\ 1^2 & 2^2 & \dots & n^2 \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix}$$

To match the setup we made in Hamming, we would want some check like

$$\begin{pmatrix} \mathbf{v_1} & \mathbf{v_2} & \dots & \mathbf{v_n} \\ \mathbf{v_1}^2 & \mathbf{v_2}^2 & \dots & \mathbf{v_n}^2 \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

We want a nice canonical way to square vectors. We can identify the m-bit vectors with a field, meaning we can add and multiply them. Formally:

$$\mathbb{F}_2 m \simeq \mathbb{F}_2^m$$

In particular, this identification are polynomials $\mathbb{F}_2$ of degree up to $m - 1$.

Note 1.1
If you have $f, g \in \mathbb{F}_2[x]$, then $(f(x) + g(x))^2 = f(x)^2 + g(x)^2$. (The cross term has coefficient 2, which is 0 mod 2).

This means squaring the vectors doesn't actually get us anything new that we couldn't do with the linear terms. So, let's instead define C_{BCH_2}, with parity check matrix:

$$H = \begin{pmatrix} \mathbf{v_1} & \mathbf{v_2} & \dots & \mathbf{v_n} \\ \mathbf{v_1}^3 & \mathbf{v_2}^3 & \dots & \mathbf{v_n}^3 \end{pmatrix}$$

Exercise: Show that C_{BCH_2} can correct 2-bit flips, i.e. show that if:

$$\alpha \neq \beta, \gamma \neq \delta$$
$$\alpha + \beta = \gamma + \delta$$
$$\alpha^3 + \beta^3 = \gamma^3 + \delta^3$$

then $\{\alpha, \beta\} = \{\gamma, \delta\}$