CDM

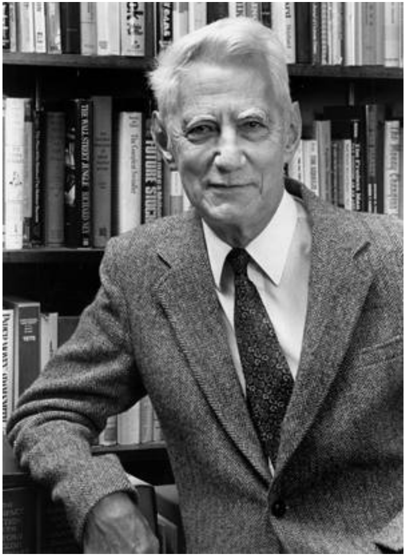
Finite Fields and Codes

Klaus Sutner

Carnegie Mellon University

Fall 2024

Claude Elwood Shannon (1916–2001)



Shannon's Information Theory

Two crucial papers:

A Mathematical Theory of Communication

Bell System Technical J., 27(1948)3/4, pp. 379–423, 623–656.

Communication Theory of Secrecy Systems

Bell System Technical J., 28(1949)4, pp. 656–715.

The first paper introduces the notion of a bit (binary digit, suggested by John Tukey). Since Shannon was working for the phone company, he was interested in how many phone calls could be transmitted in parallel on the same copper wire. His work relies heavily on probability theory.

Measuring Information

Hard Question:
How can we assign a numerical measure to information content?

This may seem like a hopeless task, but it becomes manageable if one stays away from semantics: figuring out whether a text by Immanuel Kant has more information than a text by Albert Einstein is indeed hopeless.

Key Idea: Information has to do with surprise: the more surprising an event is, the more information it carries.

Technically, this approach allows us to study probabilities rather than embark on the hopeless enterprise of defining a semantics for natural languages.

Axioms for Information

Consider some event E that occurs with probability $\mathrm{Pr}[\,E\,]$. Here are some simple axioms for the information content $I(E)$.

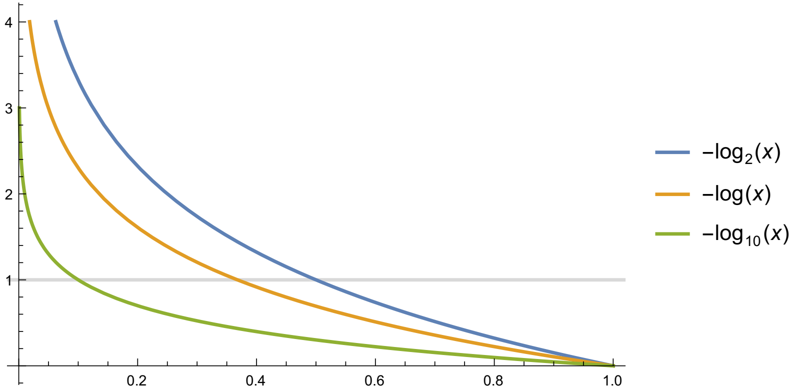
$\mathrm{Pr}[\,E\,] = 1$	implies	$I(E) = 0$
$\mathrm{Pr}[\,E\,] = 0$	implies	$I(E) = \infty$
$\mathrm{Pr}[\,E_0\,] \leq \mathrm{Pr}[\,E_1\,]$	implies	$I(E_0) \geq I(E_1)$
$E_0,\ E_1$ indep.	implies	$I(E_0 \wedge E_1) = I(E_0) + I(E_1)$

From these basic assumptions it follows easily that

$$\boxed{I(E) = -\log_b \mathrm{Pr}[\,E\,]}$$

Base $b = 2$ is the standard choice, corresponding to Yes/No answers.

Picture



Information and Entropy

In most applications, Alice sends a stream of symbols to Bob that constitute a message. The symbols are chosen from a fixed collection, an alphabet $\{a_1, a_2, \ldots, a_q\}$. Let's call this scenario a channel. Many symbols are transmitted, so it makes sense to speak of a_i being sent with probability p_i, where $\sum_i p_i = 1$.

Key Question: How much information is contained in a message?

Well, it's the average information content of the symbols that make up the message.

This is called the entropy of the channel:

$$H = -\sum_i p_i \log p_i$$

Uniform Probabilities

Here is a simple special case: uniform probabilities $p_i = 1/q$.

$$H = -\sum_i 1/q \log 1/q = \log q$$

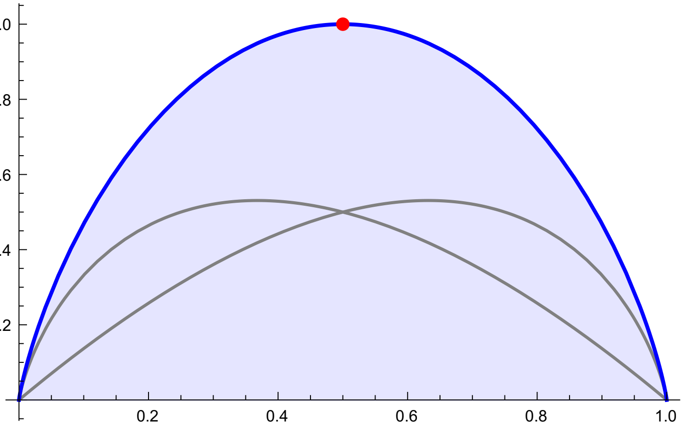
In particular for $q = 2$ we have $H = 1$. Makes some intuitive sense.

How about $q = 2$ with biased probabilities?

$$H = -p \log p - (1-p) \log(1-p)$$

This is the binary entropy function (and very useful in some combinatorial arguments).

Binary Entropy



Redundancy and Compression

Redundancy is very closely related to entropy and measures the amount of superfluous information being transmitted. Superfluous in the sense that, in principle, we could transmit the same amount of information using fewer bits. In other words, we try to measure possible compression.

Why would anyone use redundant communication? Because it provides some degree of protection against noise.

Flipping a few bits in a page of English text is usually no problem, but in a compressed file it will be a disaster.

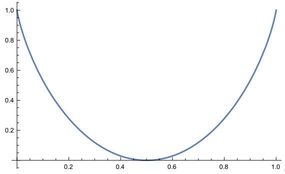
Unsurprisingly, human speech relies heavily on redundancy to protect against noisy channels.

Technically

The redundancy for a q-symbol channel is given by

$$R = \frac{H_{\text{max}} - H}{H_{\text{max}}}$$

Here $H_{\text{max}} = \log q$ represents the maximum entropy of any q-symbol channel. So $0 \leq R \leq 1$, and in the uniform case we have redundancy 0: all possible messages are equally likely.

In the binary case this is just upside-down entropy: .

How About English?

So what is the redundancy of English? Shannon was the first to try to come up with a reasonable answer to this question.

The problem here is that English is not a formal language, so one can try to approximate it by fixing an alphabet (say, 26 letters or 27 if one includes a space symbol) and then looking at probabilities of blocks of N consecutive letters.

onecanalmostalwaysfillinthespacesfromasequenceofwordswithnospaces

Shannon came up with an estimate of $R \approx 0.54$.

But How?

The big problem when dealing with natural languages is that one has to make lots of simplifying assumptions.

- English text consists of sequences of blocks of letters (aka words),
- the letters within a block are not equidistributed,
- the arrangement of blocks is controlled by grammar and far from random.

So all one can hope for are some rough approximations.

N-Grams

One approach is to determine probabilities of the next letter assuming knowledge of the current N letters (think of assigning probabilities to the edges of a de Bruijn graph):

$$a_1 a_2 \dots a_N \rightsquigarrow b$$

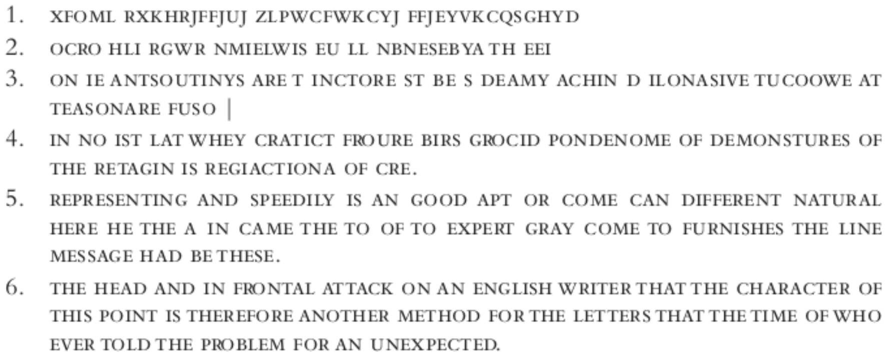
This is easy for small N and one can hope that the numbers converge rapidly to a limit of sorts. Shannon did concrete calculations and came up with the following estimates:

N	0	1	2	3	word
H_N	4.7	4.14	3.56	3.3	2.62

If all 3-letter blocks were equally likely we would get $H = 14.1$.

The last value for words comes from Shannon's analysis of the entries in a dictionary.

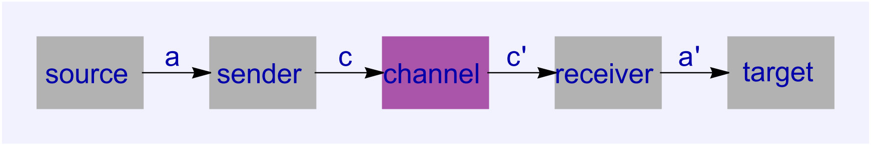
English Approximations



1. XFOML RXKHRJFFJUJ ZLPWCFWKCYJ FFJEYVKCQSGHYD
2. OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA TH EEI
3. ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIN D ILONASIVE TUCOOWE AT TEASONARE FUSO |
4. IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID PONDENOME OF DEMONSTURES OF THE RETAGIN IS REGIACTIONA OF CRE.
5. REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF TO EXPERT GRAY COME TO FURNISHES THE LINE MESSAGE HAD BE THESE.
6. THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPECTED.

Shannon's Model

A message a is generated by some source, converted into a signal by the sender (transmitter) which is sent over the channel. The receiver turns the signal into a received message a' which reaches the destination.



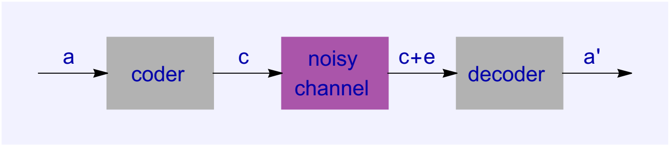
Two major scenarios:

- The channel is noisy: $c \approx c'$.
  Interesting question: how to recover the original message, coding theory.

- The channel is noiseless: $c = c'$.
  Interesting question: how to protect against eaves-dropping, cryptography.

Noisy Channels

We will only deal with the noisy scenario here. The sender here needs to turn a message $\boldsymbol{a}$ into a code $\boldsymbol{c}$, which is structured in such a way that the decoder has a chance to recover $\boldsymbol{a}$ even though $\boldsymbol{c} \neq \boldsymbol{c}'$.



The decoder receives $\boldsymbol{c}' = \boldsymbol{c} + \boldsymbol{e}$ where $\boldsymbol{e}$ is the error introduced by the channel (for the moment, don't worry what exactly addition means here).

It outputs a decoded message $\boldsymbol{a}'$ which is hopefully identical to $\boldsymbol{a}$. If only . . .

Keeping It Real

Obviously, if the noisy channel introduces too many errors, there is no way to recover $\boldsymbol{c}$.

Just think about a channel that returns a stream of 0s, or a sequence of random bits.

We need to make probabilistic assumptions about the likelihood of errors. As long as these probabilities are low, we want to be able to decode. If the the error probabilities are large, we are really faced with an issue of channel design, not of coding theory.

General Case

Suppose we want to transmit a word over some alphabet Q of size q. This is called an q-ary channel.

In general, all we have is a probability matrix

$$P = (p_{ab}) \in [0,1]^{q \times q}$$

where p_{ab} indicates the likelihood that symbol a sent into the channel is received at the other end as symbol b. So we would like p_{aa} to be large.

In a civilized channel, we may assume that the matrix is stochastic: all rows and columns sum to 1.

In other words, we will not deal with errors based on insertions and deletions.

Symmetric Channels

Suppose again we have a q-ary channel.

Major Simplification: We assume that there is a single parameter $p \geq 0$ that describes all error probabilities: p is the probability that the channel will transmit any symbol as a different symbol.

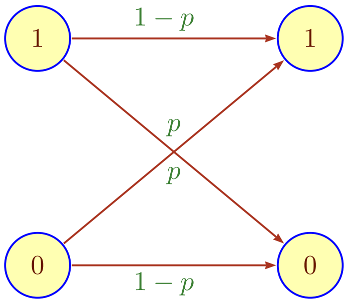
So we are sending a symbol a into the channel, and receive b at the other end. The error probability p is the same for all a and b.

$$\begin{aligned} p_{aa} &= \mathsf{Pr}[\,a = b\,] = 1 - (q-1)p \\ p_{ab} &= \mathsf{Pr}[\,a \neq b\,] = (q-1)p \end{aligned}$$

Sanity Assumptions:

- We will always assume that p is small relative to q (see below).
- We will not deal with insertions and deletions of symbols.

Binary Symmetric Channel



$q = 2$: This is the most elementary scenario, but already very interesting

$$P = \begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix}$$

Block Codes

We will think of messages and codes as words over an alphabet Q:

$$\boldsymbol{a} = a_1 a_2 \ldots a_k$$

$$\boldsymbol{c} = c_1 c_2 \ldots c_n$$

We are mostly interested in the binary case $Q = \mathbf{2}$, but larger alphabets can be handled similarly.

Useful Trick:
Think of Q as being some finite field $\mathbb{F}_q$.

Messages $\boldsymbol{a}$ and codes $\boldsymbol{c}$ are then simply vectors over this finite field, of length k and n, respectively.

For example, if we want to use bytes as letters, we pick $\mathbb{F}_{2^8}$.

In characteristic 2 we are just dealing with various bit operations, so efficiency should be good.

But Why?

Why not simply think of the symbols as purely combinatorial objects, say, letters a, b, c, $\ldots$

Because algebra is your friend.

The whole purpose of algebra is to "trivialize" the solution of certain types of problems. If some particular problem can naturally be expressed in terms of algebra, it is probably easier to handle than otherwise.

Errors

One advantage of the vector model is that we can model the error as just another vector:

$$\boldsymbol{c} \mapsto \boldsymbol{c} + \boldsymbol{e}$$

The key challenges are

- Error Detection
  Detect when $\boldsymbol{e} \neq 0$ (and ask for a re-transmit).
- Error Correction
  Detect an error and correct it (by adding $-\boldsymbol{e}$ to the transmitted message).

Again, error detection and correction only work if the channel is reasonably well behaved. In practice, the major bottleneck is usually the efficiency of the decoder.

Terminology

Recall that we think of messages as k-vectors over some finite field $\mathbb{F}$ of size $q = p^\ell$.

Hence we have $q^k = p^{\ell\, k}$ possible messages and need to assign a unique codeword to each.

Definition

A (q-ary) code is a subset $C \subseteq \mathbb{F}^n$ of size q^k.

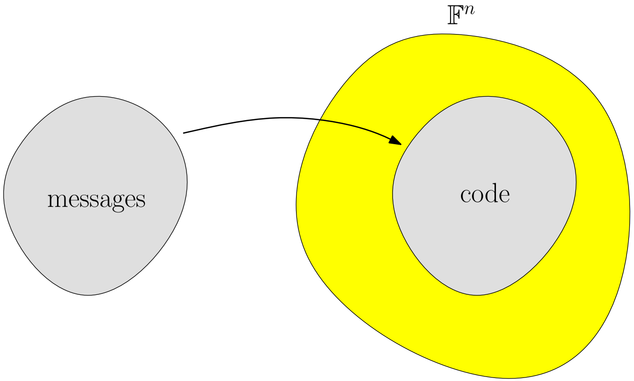
The elements of C are codewords and the ambient space $\mathbb{F}^n$ is the codespace.

So the key parameters of a code are the arity q, the dimension k and the (block) length n[†].

Obviously the length is at least the dimension, but for error detection and correction it needs to be larger, we need to introduce some redundancy.

[†]Bad terminology, as a vector space, $\mathbb{F}^n$ has dimension n.

Codespace



This picture is slightly misleading: the code words ought to be spaced out nicely in the ambient space.

Encoding and Decoding

Once we have fixed arity, dimension and block length, a coding function is an injective map

$$E : \mathbb{F}^k \longrightarrow \mathbb{F}^n$$

and a decoding function is a surjection

$$D : \mathbb{F}^n \longrightarrow \mathbb{F}^k$$

The whole point is that not just $D(E(\boldsymbol{x})) = \boldsymbol{x}$, but

$$\boldsymbol{c} = E(\boldsymbol{x}) \qquad \text{implies} \qquad D(\boldsymbol{c} + \boldsymbol{e}) = \boldsymbol{x}$$

for as many $\boldsymbol{e}$ as manageable.

Getting Serious

To construct a code, at the very least, we need to do the following:

- choose a proper value for the length $n \geq k$,
- pick a subset C of $\mathbb{F}^n$ of cardinality q^k, and
- determine the maps E and D.

Note, though, that unlike with cryptography (and in particular public key cryptography) we only need to do this once, we can use the same code over and over again.

Coding and Decoding

How large should the dimension be? We need to space out codewords so we can recover from some errors, so n should be large—but efficiency requires to keep n small.

Also, we need to specify $C \subseteq \mathbb{F}^n$ explicitly. Sine q^n is large this requires some systematic tool, lookup tables are not an option.

Once the code C is fixed, we need to find efficient algorithms to compute the maps E and D.

The good news: There is quite a bit of leeway in how to organize this. Before we talk about specific codes, let's look at some fundamental properties of this setup.

Hamming Distance

There is a natural way to introduce geometry in the codespace $\mathbb{F}^n$: we can measure distances between points.

Definition

The Hamming distance between two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}^n$ is defined by

$$\operatorname{dist}(\boldsymbol{x}, \boldsymbol{y}) = |\{\, i \mid x_i \neq y_i \,\}|$$

The weight (or support) of a vector $\boldsymbol{x} \in \mathbb{F}^n$ is defined by

$$w(\boldsymbol{x}) = |\{\, i \mid x_i \neq 0 \,\}|$$

Thus $\operatorname{dist}(\boldsymbol{x}, \boldsymbol{y}) = w(\boldsymbol{x} - \boldsymbol{y})$.

Exercise

Check that d really is a metric.

Geometric Intuition

We are making an attempt to bring geometry to bear on our problem. Careful, though, we are not in the standard, continuous world of Euclidean space $\mathbb{R}^3$, but in some high-dimensional discrete space. Geometric intuition in high-dimensional spaces can be quite limited.

In many ways, this is just fine. E.g., it still makes perfect sense to talk about a ball of radius r centered at $\boldsymbol{x}$:

$$B_r(\boldsymbol{x}) = \{\, \boldsymbol{z} \in \mathbb{F}^n \mid \operatorname{dist}(\boldsymbol{x}, \boldsymbol{z}) \leq r \,\}$$

But the radius here is integral, and the difference between open and closed balls evaporates: open radius r is closed radius $r-1$.

On the other hand, everything is discrete and we can, say, count the number of points in a ball.

Counting Errors

A basic error in the channel is thus represented by an error vector of weight 1:

$$\boldsymbol{e} = (0, \dots 0, x, 0, \dots 0)$$

where $x \neq 0$. In the characteristic 2 case $x = 1$, the error is a unit vector.

We can think of the channel as introducing a number of basic errors sequentially. An error vector of weight e is thus referred to as "e errors." This makes perfect sense in symmetric channels where each flip is independent and equally likely.

Goal: We want codes that can correct reasonably many errors.

Strategy: We will use redundancy to protect against errors.

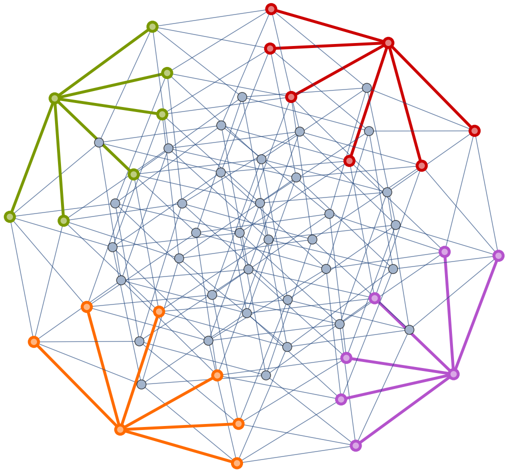
An Idiosyncrasy

In coding theory texts, when one says a code detects e errors, what is actually meant is this: For any codeword $\boldsymbol{c}$ and any error $\boldsymbol{e} \neq \mathbf{0}$ of weight at most e:

- one can detect that $\boldsymbol{z} = \boldsymbol{c} + \boldsymbol{e} \notin C$, and
- one can determine the weight of $\boldsymbol{e}$ from $\boldsymbol{z}$.

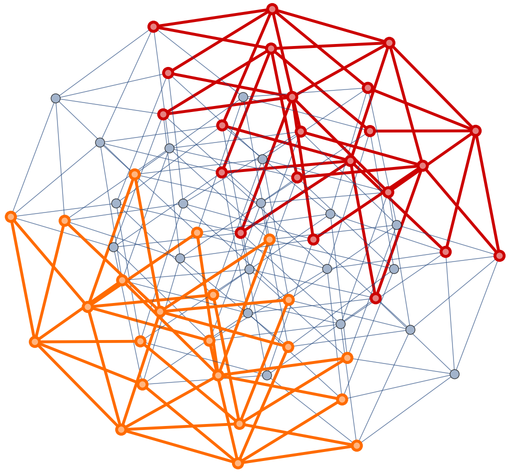
So this better than just realizing that there was an error, but slightly weaker than being able to correct it: we know the weight e of the error, but not necessarily the actual error $\boldsymbol{e}$.

6-Dim Hypercube



4 balls of radius 1 in the 6-dimensional hypercube $\mathbf{2}^6$.

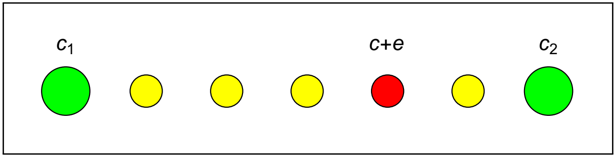
6-Dim Hypercube



2 balls of radius 2 in the 6-dim hypercube $\mathbf{2}^6$, each containing 22 points.

Abstract Space

Suppose the received noisy transmission $\boldsymbol{c} + \boldsymbol{e}$ winds up "between" two codewords at distance 6. For simplicity, let's ignore all other codewords for the time being and organize points by distance.



We have $d(\boldsymbol{c}_1, \boldsymbol{c}_2) = 6$ and the transmission could come from a weight-2 or a weight-4 error. We can only detect errors of weight 3 and we can correct errors of weight 2.

Conditional Probabilities

Suppose we have an q-ary symmetric channel with (single) error probability $(q-1)p$.

The probability that $\boldsymbol{y}$ is received when $\boldsymbol{x}$ as been transmitted is

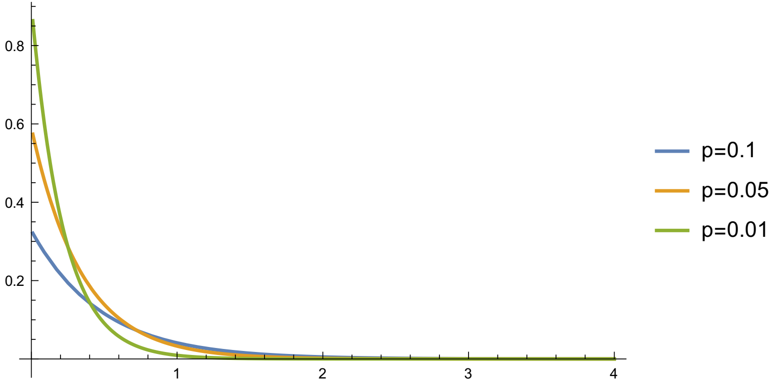
$$\Pr[\,\boldsymbol{y} \mid \boldsymbol{x}\,] = p^d (1-(q-1)p)^{n-d}$$

where $d = \operatorname{dist}(\boldsymbol{x}, \boldsymbol{y})$.

As long as $p < 1/q$ this function is sharply decreasing in d, so it is unlikely that we will hit a $\boldsymbol{y}$ far away from $\boldsymbol{x}$.

As already mentioned, if p is too large one has to redesign the channel.

Example: $q = 3$, $n = 5$, $p = 0.1, 0.05, 0.01$

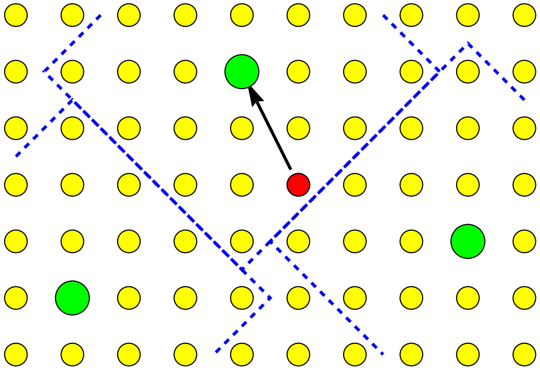


So $d = 2$ is already quite unlikely, even for an error probability of 10%.

Maximum Likelihood Decoding

If $\boldsymbol{x}$ is transmitted and $\boldsymbol{y} \notin C$ is received, the natural decoding strategy based on $\Pr[\,\boldsymbol{y} \mid \boldsymbol{x}\,]$ is to find the codeword $\boldsymbol{c} \in C$ closest to $\boldsymbol{y}$.

This makes sense in particular when this $\boldsymbol{c}$ is uniquely determined.



Balls and Errors

Suppose we have chosen a code C. We define the minimal distance of C to be

$$\mathrm{md}(C) = \min\big(\,\mathrm{dist}(\boldsymbol{x},\boldsymbol{y}) \mid \boldsymbol{x} \neq \boldsymbol{y} \in C\,\big)$$

Thus a ball of radius $\mathrm{md}(C) - 1$ around a codeword in C contains no other points in C.

Proposition

- *If* $\mathrm{md}(C) \geq 2e$ *then* C *detects* e *errors*[†].
- *If* $\mathrm{md}(C) \geq 2e + 1$ *then* C *corrects* e *errors.*

Note the active voice; we really should have said: it is possible to detect/correct–we don't yet know how to do this efficiently.

[†]Remember the meaning of "detects."

Hamming Bound

Lemma (Hamming)

Suppose the q-ary code C has length n and corrects e errors.
Let $V = \sum_{i=0}^{e} \binom{n}{i}(q-1)^i$. Then

$$V\,|C| \leq q^n$$

Proof. Since the code corrects e errors, all balls of radius e centered at $\boldsymbol{c} \in C$ are necessarily disjoint. The number of points at distance i to $\boldsymbol{c}$ is $\binom{n}{i}(q-1)^i$, so V is just the volume of a ball of radius e. □

A code is perfect iff it realizes this bound. Needless to say, Hamming found a way to construct perfect codes.

Examples

Time for some concrete examples.

In all cases, we introduce redundancy in a more or less clever way to help with error detection and correction.

One could object that these codes are really just hacks, for a more systematic approach look at the next section.

Example 1: Repetition Code

This is a bit embarrassing, but bear with me. In $\mathbb{F}_2^n$ define E by

$$x \mapsto (\underbrace{x, x, \ldots, x}_{n})$$

So the only codewords are $00\ldots0$ and $11\ldots1$.

The minimum (and only) distance is n and balls of radius $\lfloor (n-1)/2 \rfloor$ around the codewords are still disjoint.

More importantly, it easily corrects $\lfloor (n-1)/2 \rfloor$ errors: we use a majority count to determine the original codeword.

For odd n, these codes are perfect.

The obvious fatal problem with this is the low rate of transmission: the codeword is n-times longer than the original message.

Example 2: Parity Check Code

To encode, add a parity “bit” to the message:

$$\boldsymbol{x} \mapsto \left(\boldsymbol{x}, \sum x_i\right)$$

so that $n = k + 1$. Here we are exploiting the fact that the x_i are field elements, so the sum naturally makes sense.

A parity check code detects 1 error.

Alas, it corrects none.

This may sound next to useless, but it is not: variants of this scheme are used in ISBN, good enough for a low key sanity check.

Example 3: Correcting One Error

Consider $q = 2$, dimension $k = 4$ and length $n = 7$.

To encode a 4-bit vector $\boldsymbol{x} = (x_1, x_2, x_3, x_4)$, we transmit 7 bits $\boldsymbol{c} = (c_1, c_2, c_3, c_4, c_5, c_6, c_7)$ where

$$c_3 = x_1 \qquad c_5 = x_2 \qquad c_6 = x_3 \qquad c_7 = x_4$$

are the original message bits. The weird numbering will become clear in a moment.

The missing bits c_1, c_2, c_4 are determined by

$$\begin{aligned} c_1 &= c_3 + c_5 + c_7 = x_1 + x_2 + x_4 \\ c_2 &= c_3 + c_6 + c_7 = x_1 + x_3 + x_4 \\ c_4 &= c_5 + c_6 + c_7 = x_2 + x_3 + x_4 \end{aligned}$$

Matrix View

We can think of the encoding process as a simple vector-matrix multiplication $\boldsymbol{x} \mapsto \boldsymbol{x}\, G$ where $G \in \mathbf{2}^{4\times 7}$:

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

G has full rank 4, so it generates a 4-dimensional subspace.

This is easy to see if we move the columns around to get an identity matrix plus the parity part.

$$\left(\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array} \right)$$

OK, but why mess things up as in G?

A Hack

For decoding, calculate 3 bits α, β and γ as follows:

$$\gamma = c_1 + c_3 + c_5 + c_7$$
$$\beta = c_2 + c_3 + c_6 + c_7$$
$$\alpha = c_4 + c_5 + c_6 + c_7$$

If there is no error, $\alpha = \beta = \gamma$ and we can read off $\boldsymbol{x}$ directly.

If there is a single error, then $1 \leq \alpha\beta\gamma \leq 7$ is the binary expansion of the place where the error occurred.

Example:

$\boldsymbol{x} = (1, 1, 0, 0)$ yields $\boldsymbol{c} = (0, 1, 1, 1, 1, 0, 0)$.
Let $\boldsymbol{c}' = (0, 1, 1, 1, 0, 0, 0)$.
Then $\alpha\beta\gamma = 101$, the error is in position 5.

Example 4: . . . and Detecting Two

The last code does not handle 2 errors in a civilized way: the following codewords have identical 1-error and 2-error versions:

$$\begin{array}{ccc} 0\,0\,0\,0\,0\,0\,0 & & 0\,1\,0\,1\,0\,1\,0 \\ \downarrow & & \downarrow \\ 0\,0\,0\,0\,0\,1\,0 & = & 0\,0\,0\,0\,0\,1\,0 \end{array}$$

We could fix this problem by adding another parity bit

$$c_0 = c_1 + c_2 + c_3 + c_4 + c_5 + c_6 + c_7$$

This extended code detects two errors (recall the definition) and still corrects one as before.

Using Algebra

So far, everything is a bit ad-hoc. Here comes the first real idea:

Use linear algebra to define codes and handle coding/decoding.

So instead of some clever hack, we would like C to be determined by some algebraic definition.

Hopefully this approach will automatically take care of spacing out the the codewords, and provide some computational support for coding/decoding, including error correction.

Linear Codes

Definition

A linear code is a linear subspace $C \subseteq \mathbb{F}^n$.

A linear code of length n and dimension k is an $[n, k]$ code.

A generator matrix G for C is a k by n matrix over $\mathbb{F}_q$ whose rows form a basis for the code space C.

In other words, the subspace C is the row space of G:

$$C = \{\, \boldsymbol{x}\, G \mid \boldsymbol{x} \in \mathbb{F}^k \,\}$$

Example: $[6,3]$ Code

Let $q = 2$, $k = 3$, $n = 6$ and

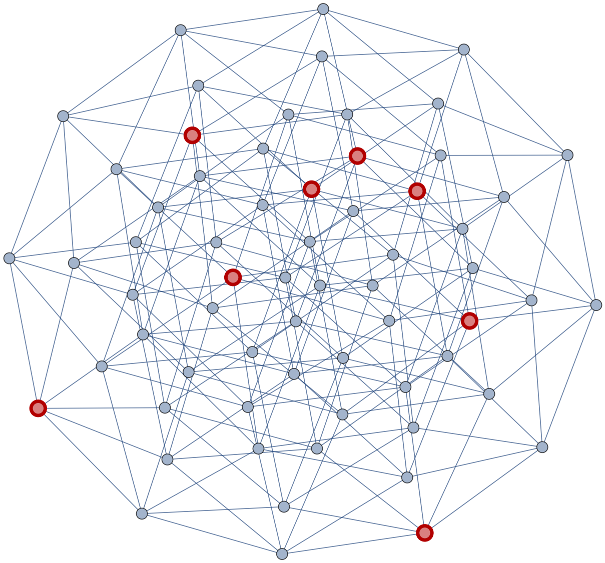
$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

The encoding map is

$$\boldsymbol{x} \mapsto \boldsymbol{y} = (x_1, x_2, x_3, x_2 + x_3, x_1 + x_3, x_1 + x_2)$$

Decoding in the absence of errors is entirely trivial since $\boldsymbol{x}$ is just the first 3 components of $\boldsymbol{y}$, but what if the decoder receives $\boldsymbol{y} + \boldsymbol{e}$ instead?

Code Space



Standard and Systematic Codes

Note that our generator matrix has the form

$$G = \big(I_k \; P\big) \in \mathbb{F}^{k \times n}$$

where I_k is the identity matrix of order k, so G is in reduced echelon form and has full rank $n{-}k$. Such codes are said to be in standard form. Alternative the columns of I_k can be spread out in G; this is called systematic form. This is essentially the same, but recall the hack from above.

Correspondingly, in a standard form code, one calls

- the first k bits information symbols
- the last $n - k$ bits parity check symbols

In this sense all linear codes are parity check codes.

Parity Check Matrix

Suppose we are in characteristic 2. Let $G = \big(I_k \; P\big)$ be the $k \times n$ generator matrix of a standard form $[n, k]$ linear code. Define the $(n-k) \times n$ matrix H by

$$H = (P^T \; I_{n-k})$$

Definition

H is the parity check matrix for code C.
The vector $\mathrm{syn}(\boldsymbol{z}) = \boldsymbol{z} \cdot H^T \in \mathbb{F}^{n-k}$ is the syndrome of $\boldsymbol{z} \in \mathbb{F}^n$.

Since we are working in characteristic 2 we have $G \cdot H^T = \mathbf{0}$ so that

$$\boldsymbol{z} \in C \iff \mathrm{syn}(\boldsymbol{z}) = \mathbf{0}$$

and we can test membership in C via a simple vector-matrix multiplication.

Example contd.

For the $[6, 3]$ code from above we have

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

and

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

In this case, the P-part of G is symmetric and thus invariant under transpose.

Dealing with Errors

For our example, the syndrome looks like $\mathrm{syn}(\boldsymbol{y}) = (s_1, s_2, s_3)$ where

$$s_1 = e_2 + e_3 + e_4$$
$$s_2 = e_1 + e_3 + e_5$$
$$s_3 = e_1 + e_2 + e_6$$

and here are the syndromes associated with 1-bit errors (really H^T):

e_1	0 1 1
e_2	1 0 1
e_3	1 1 0
e_4	1 0 0
e_5	0 1 0
e_6	0 0 1

So we can fix all 1-bit errors.

Decoding

Decoding here hinges on the following observation (which is not hard to prove):

Claim

If $(s_1, s_2, s_3) \neq (1, 1, 1)$ *there is a unique choice for such an error vector* e *(of weight at most 1).*

Alas, if $(s_1, s_2, s_3) = (1, 1, 1)$ all the following 3 weight 2 vectors work: $(1, 0, 0, 1, 0, 0)$, $(0, 1, 0, 0, 1, 0)$, $(0, 0, 1, 0, 0, 1)$.

In this case we can either ask for a retransmit, or we can simply pick one of the 3 possibilities–which will be right $1/3$ of the time.

So?

How much have we gained?

If we transmit 3 bits in a symmetric binary channel with error probability $p = 0.001$, the likelihood of correct transmission is 0.997003.

But if we use our $[6, 3]$ code, the likelihood of a correct transmission is

$$(1-p)^6 + 6(1-p)^5 p + (1-p)^4 p^2 =$$
$$0.994015 + 0.00597006 + 9.9600610^{-7} = 0.999986$$

corresponding to $\boldsymbol{s = 0}$, $\boldsymbol{0 < s < 1}$ and $\boldsymbol{s = 1}$.

The new error probability is smaller by a factor of about 214, at a cost of having to send twice as many bits. A nice improvement.

Determining Minimum Distance

According to our definition the minimum distance is the least $\mathrm{dist}(\boldsymbol{x}, \boldsymbol{y})$ where $\boldsymbol{x} \neq \boldsymbol{y} \in C$. In a linear code we can do better.

Lemma

For a linear code we have

$$\mathrm{md}(C) = \min\Big(w(\boldsymbol{x}) \mid 0 \neq \boldsymbol{x} \in C \Big)$$

This may not look too impressive, but at least it yields a linear time method as opposed to the obvious quadratic one that works for an any code.

Parity Check and Distance

Let H be the check matrix of linear code C with columns $\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n$.

Then for $\boldsymbol{z} = \boldsymbol{c} + \boldsymbol{e}$ we have $\mathrm{syn}(\boldsymbol{z})^T = H\,\boldsymbol{e}^T = \sum \boldsymbol{h}_i e_i$.

This linear combination is 0 iff $\boldsymbol{z} \in C$.

Hence we have the following alternative (and often superior) way to compute the minimum distance.

Lemma

The minimum distance of C is the least d such that there exists a set of d linearly dependent column vectors in H.

Orthogonality

Definition

For $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}^n$ define the inner product

$$\boldsymbol{x} \cdot \boldsymbol{y} = \sum x_i y_i$$

Given a subset $X \subseteq \mathbb{F}^n$ define its orthogonal complement by

$$X^\perp = \{\, \boldsymbol{y} \in \mathbb{F}^n \mid \forall\, \boldsymbol{x} \in X\, (\boldsymbol{x} \cdot \boldsymbol{y} = 0) \,\}$$

This is similar to the notion of orthogonal complements in real spaces but the geometry here is more complicated. For example, it may happen that $X^\perp = X$: consider $X = \{\, (x_1, x_1, x_2, x_2, \dots, x_k, x_k) \mid \boldsymbol{x} \in \mathbb{F}_2^k \,\}$.

Orthogonality and Parity

Our codes are linear subspaces

$$C = \{\, \boldsymbol{x}G \mid \boldsymbol{x} \in \mathbb{F}^k \,\}$$

and G is an $k \times n$ matrix of full rank (in this case rank k).

Alternatively we can use the $(n-k) \times n$ parity check matrix to describe C like so:

$$C = \{\, \boldsymbol{z} \in \mathbb{F}^n \mid \boldsymbol{z}H^T = \mathbf{0} \,\}$$

So C is the space orthogonal to another which is described by the parity check matrix.

Dual Codes

We can push this idea one step further:

Parity check matrices are actually generators for a closely related code.

Definition

Let C be a linear $[n, k]$ code. The dual code is $C^{\perp}$.

- The dual code has dimension $n - k$.
- The dual of the dual is the code: $C^{\perp\perp} = C$.
- If C has generator matrix G then $C^{\perp}$ has parity check matrix G.
- If C has parity check matrix H then $C^{\perp}$ has generator matrix H.

Decoding

How do we decode a linear code systematically?

Suppose we receive $\boldsymbol{z} = \boldsymbol{c} + \boldsymbol{e}$. Note that

$$\boldsymbol{z} + C = \boldsymbol{e} + C \subseteq \mathbb{F}^n$$

since $\boldsymbol{c} \in C$ and C is a linear subspace. Hence, the $\boldsymbol{e}$ lies in the same coset as $\boldsymbol{z}$. We can decompose $\mathbb{F}^n$ into $n - k$ disjoint cosets (affine subspaces)

$$C, \boldsymbol{a}_2 + C, \ldots, \boldsymbol{a}_{n-k} + C$$

Two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are in the same coset iff they have the same syndrome:

$$\mathrm{syn}(\boldsymbol{x}) = \mathrm{syn}(\boldsymbol{y}) \iff \boldsymbol{x} - \boldsymbol{y} \in C$$

Coset Leaders

Definition

A coset leader is an element of the coset of minimal weight.

So if we want to do maximum likelihood decoding we should pick a minimum weight vector in the coset, the coset leader, as the error vector.

Unfortunately, “minimum weight” is not a property expressible in terms of linear algebra, it adds a component of combinatorics. So we have to precompute a minimum weight vector for each possible syndrome.

Example Contd.

For our standard $[6, 3]$ example we have the following coset leaders, parametrized by syndrome (the s-vectors from above):

syndrome	leader
000	000000
001	000001
010	000010
100	000100
011	100000
101	010000
110	001000
111	$001001, 010010, 100100$

Syndrome Decoding

As we have seen, the syndrome function is constant on each coset and differs on each coset (if you like, the coset partition is just the kernel relation induced by the syndrome function).

But then is suffices to store a syndrome table: for each syndrome store the corresponding coset leader.

To decode $\boldsymbol{z} = \boldsymbol{c} + \boldsymbol{e}$:

- compute $\boldsymbol{s} = \mathrm{syn}(\boldsymbol{z})$,
- look up the corresponding error vector $\boldsymbol{e}$,
- return $\boldsymbol{z} - \boldsymbol{e}$, presumably just the message $\boldsymbol{c}$.

Shifting Codewords

Definition

A linear code C is cyclic if for all $\boldsymbol{c} \in C$ the cyclic shift of $\boldsymbol{c}$ is also in C.

As we will see, cyclic codes have a very simple description (a single generator). They are used mostly for error detection: cyclic redundancy check (CRC). A retransmit is requested if an error is found.

Example

$C = \{000, 011, 101, 110\}$ is cyclic code of length 3.

This generalizes to all words in $\mathbb{F}^n$ with even weight.

Codewords and Code Polynomials

Instead of thinking about vectors $\boldsymbol{a} \in \mathbb{F}^n$ we will interpret codewords as polynomials, so-called code polynomials:

$$a(x) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}$$

Write $\boldsymbol{a}'$ for the cyclic right-shift of $\boldsymbol{a}$ and note that the corresponding code polynomial has the form

$$a_{n-1} + a_0 x + a_1 x^2 + \ldots + a_{n-1} x^n$$

But then $a'(x) = x \cdot a(x) \bmod (x^n - 1)$.

So if we think of code polynomials as elements of the quotient ring $\mathbb{F}_q[x]/(x^n - 1)$, then the combinatorial shift operation corresponds simply to the algebraic opertion "multiplication by x."

Ideals

Is there a good description for cyclic codes, viewed as subsets of $\mathbb{F}_q[x]/(x^n - 1)$?

Lemma

A linear code C is cyclic iff C is an ideal in $\mathbb{F}_q[x]/(x^n - 1)$.

Proof. If C is cyclic then the set of corresponding code polynomials is closed under multiplication by x, and hence under multiplication by any polynomial. Closure under addition follows from C being a code.

In the opposite direction, C must be additively closed since the ideal is so closed. And C must be invariant under shifts since the ideal is closed under multiplication by x. □

Generator Polynomials

It is known from algebra that $\mathbb{F}_q[x]/(x^n - 1)$ is a principal ideal domain: all ideals in this ring are principal ideals. Thus, $C = (g(x)) = \mathbb{F}_q[x]\, g(x)$ for some polynomial $g(x)$.

More precisely, the generator polynomial $g(x)$ is a monic polynomial in $C - \{0\}$ of minimal degree (this polynomial is uniquely determined by C).

Note that C has dimension $n - \deg(g)$.

Once we have a generator polynomial $g(x)$, coding is very easy:

$$\boldsymbol{a} \mapsto \left(\sum a_i x^i\right) g(x)$$

This can be implemented with the help of dedicated hardware (feedback shift register), so coding can be very fast.

Check Polynomial

The generator polynomial $g(x)$ must divide $x^n - 1$: otherwise $x^n - 1 = g(x)h(x) + r(x)$ where $\deg(r) < \deg(g)$. But $r(x) \in C$, contradiction.

So let $g(x)h(x) = x^n - 1$, so g and h are zero-divisors in $\mathbb{F}_q[x]/(x^n - 1)$.

But then

$$\boldsymbol{c} \in C \iff c(x)\, h(x) = 0 \pmod{x^n - 1}$$

More generally, we can compute the syndrome of a received message $\boldsymbol{y}$ by multiplying $y(x)$ and $h(x)$, and reducing modulo $x^n \mapsto 1$. Note that this reduction is easier than the ones we saw in connection with finite fields.

More on Generators

From now on assume that the length of the code n and p, the characteristic of the underlying field, are coprime. In this case $x^n - 1$ factors into distinct irreducible polynomials:

$$x^n - 1 = f_1(x)\, f_2(x) \dots f_{t-1}(x)\, f_t(x)$$

We can produce 2^t generators by choosing $I \subseteq [t]$:

$$g(x) = \prod_{i \in I} f_i(x)$$

A maximal cyclic code has $I = \{i\}$: we pick exactly one of these irreducible factors.

Dually, a minimal cyclic code has $I = [n] - \{i\}$, we choose all but one.

Example: Length 9

Let $q = 2$ and $n = 9$. Then

$$x^9 + 1 = (x+1)(x^2+x+1)(x^6+x^3+1)$$

The maximum code based on $g = x^6 + x^3 + 1$ has codewords

$$(c_0, c_1, c_2, c_0, c_1, c_2, c_0, c_1, c_2)$$

and minimum distance 3.

The minimum code based on $h = (x+1)(x^2+x+1) = x^3+1$ is a $[9,6]$ code with minimum distance 2; it has codewords

$$(c_0, c_1, c_2, c_4, c_5, c_6, c_0 + c_4, c_1 + c_5, c_2 + c_6)$$

Note that these codewords are in fact cyclic (which is not entirely obvious from the definition).

Standard Encoding

We can compute a cyclic code by polynomial multiplication $c(x) = a(x)g(x) \bmod (x^n - 1)$.

Slightly better performance can be obtained by coding

$$\boldsymbol{a} \mapsto (\boldsymbol{a}, -\boldsymbol{s})$$

where $\boldsymbol{s}$ is determined by

$$x^{n-k}a(x) = f(x)g(x) + s(x)$$

The reason this is slightly better is that one can send the first k information symbols while computing the following $n - k$ parity symbols.

Richard Hamming

The purpose of computation is insight, not numbers.

Errors and Check Columns

Suppose H is the check matrix of a binary $[n, k]$ code.

Note that the syndrome of a received word $\boldsymbol{x} = \boldsymbol{c} + \boldsymbol{e}$ is the sum of the columns of H in positions where an error occurred:

$$\mathrm{syn}(\boldsymbol{x}) = \mathrm{syn}(\boldsymbol{c}) + \mathrm{syn}(\boldsymbol{e}) = H\,\boldsymbol{e}^T$$

In the special case where there is only **one** error we just get a single column in H. If the columns are all distinct this tells us the position of the error and we can correct it.

Example: A $[7,4]$ Code

Let $Q = \mathbf{2}$, $k = 4$, $n = 7$ and define a code C by

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

H is organized so as to make G simple (standard form, the first 4 bits are information bits). Note, though, that every 3-bit word other than $\mathbf{0}$ appears as a column in H.

It is not hard to see that $\mathrm{md}(C) = 3$, so we can correct one error by computing the syndrome.

General Case

A little care is needed to lift the binary example to the q-ary case.

Definition

A linear code over $\mathbb{F}_q$ is a Hamming code of redundancy r if it has a check matrix H that contains a unique column $a\boldsymbol{x}$ for all $\mathbf{0} \neq \boldsymbol{x} \in \mathbb{F}^r$ and some $0 \neq a \in \mathbb{F}$.

Thus every non-zero vector in $\mathbb{F}^r$ appears as exactly one column in H, up to multiplication by a non-zero scalar.

One can determine the multiplicative factor for example by insisting that the first non-zero entry in each column be 1.

Hamming Length and Dimension

Theorem

Let C be a Hamming code of redundancy $r \geq 2$. Then C is a

$$\left[\frac{q^r-1}{q-1}, \frac{q^r-1}{q-1} - r\right]$$

code. The minimum distance is 3.

Proof. There are $q^r - 1$ non-zero vectors of length r over $\mathbb{F}$. Since each column in H is a non-zero multiple there are $n = (q^r-1)/(q-1)$ columns, the length of the code. The dimension then is $n - r$.

The claim about distance follows from the lemma about linear dependence of columns in H above. □

An Improvement

We can add a parity bit to the Hamming $[7,4]$ to get a $[8,4]$ code. This code

- has minimum distance 4,
- corrects 1 error,
- detects 2 errors (in the strong sense above, 3 in the weak sense).

This type of code is sometimes called a SECDED code (single error correction, double error detection) and used in memory systems.

Perfect Codes

So far we have focused on codewords being sufficiently far from each other. Another reasonable condition is that no point in codespace should be far from a codeword.

Definition

A code with minimum distance $2e+1$ is perfect if for all $\boldsymbol{z} \in \mathbb{F}^n$ the ball of radius e contains exactly one codeword:

$$B_e(\boldsymbol{z}) \cap C = \{\boldsymbol{c}\}.$$

In other words, a code is perfect if we have equality in Hamming's bound.

Here is a bad example for a perfect code: the repetition code of length $n = 2e+1$ has $\rho(C) = e$ and $\mathrm{md}(C) = 2e+1$.

Hamming is Perfect

Recall that in the q-ary case we have length $n = (q^k - 1)/(q - 1)$ and dimension $n - r$.

Claim

Every Hamming code is perfect.

We have minimum distance 3, so $e = 1$. We need to check that for any sphere S of radius 1 we have $|C||S| = q^n$. But the latter is equivalent to

$$q^{n-r}(1 + (q - 1)n) = q^n$$

which is easily verified.