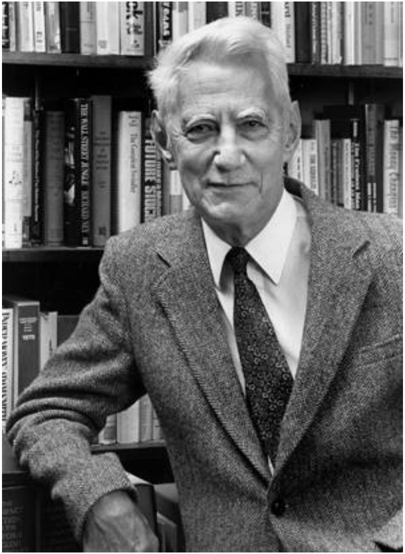
CDM

Finite Fields and Codes

Klaus Sutner

Carnegie Mellon University

44-ffields-codes 2017/12/15 23:16



Two crucial papers:

A Mathematical Theory of Communication

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

Communication Theory of Secrecy Systems

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

First paper introduces the notion of a bit. 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.

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.

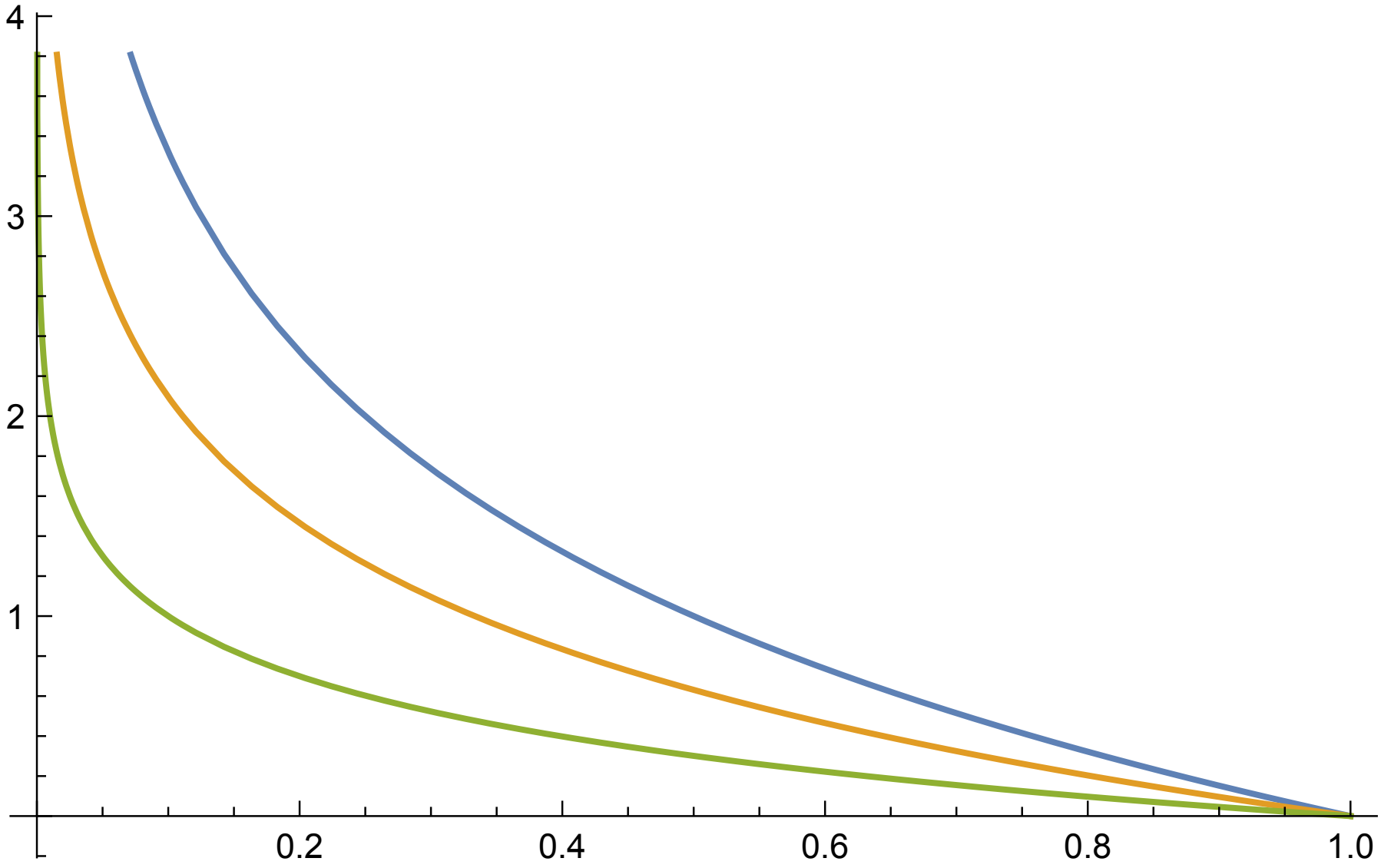
Consider some event E that occurs with probability $\mathsf{Pr}[E]$. Here are some simple axioms for the information content $I(E)$.

$\mathsf{Pr}[E] = 1$	implies	$I(E) = 0$
$\mathsf{Pr}[E] = 0$	implies	$I(E) = \infty$
$\mathsf{Pr}[E_0] \leq \mathsf{Pr}[E_1]$	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

$$I(E) = -\log_b \mathsf{Pr}[E]$$

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


4
3
2
1
0.2
0.4
0.6
0.8
1.0

In most applications, Alice sends a stream of symbols to Bob that constitute a message. The symbols are chosen from a fixed collection of symbols, 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:

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

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

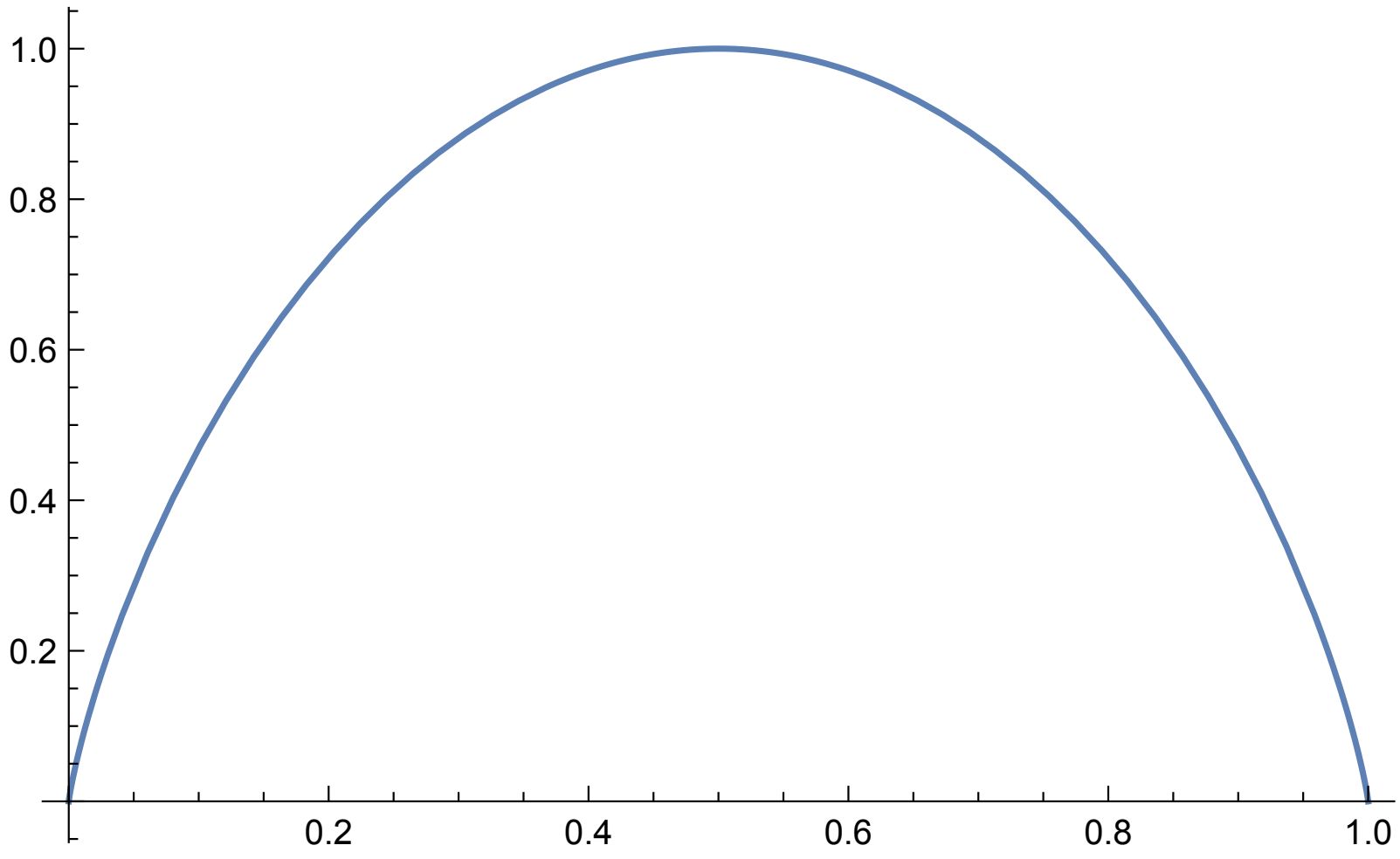
$$H = -\sum_i 1/q \log 1/q = \log q$$

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).


1.0
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1.0

Redundancy is very closely related to entropy and measures the amount of superfluous information being transmitted.

High redundancy can be a blessing (conversation in a noisy room) or a curse (expensive transmission via semaphores).

The redundancy for a q-symbol channel is given by

$$R = 1 - \frac{H}{\log q}$$

Here $\log q$ represents the maximum entropy of any q-symbol channel. So in the uniform case we have redundancy 0.

Here $q = 4$ for the bases adenine, thymine, cytosine, and guanine.

Suppose counting in a DNA sample yields probabilities as follows (these numbers are made up, not measured):

$$p_A = 0.5 \qquad p_T = 0.25 \qquad p_C = 0.125 \qquad p_G = 0.125$$

This produces

$$H = 1.75 \qquad R = 0.125$$

Redundancy is rather low, so copying must be nearly error-free to avoid disaster.

We can think of English text as being composed of 26 letters (or 27 if we include space, or more if we allow punctuation, capitalization, . . .). If all letters were equally likely and we use a 26 letter alphabet we get $H = 4.7$.

Alas, this is a lousy approximation since

- 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 there is a big problem in calculating entropy of English, or any other natural language.

One approach is to determine probabilities of the $N+1$ letter assuming knowledge of the current N letters (think of assigning probabilities to the edges of a de Bruijn graph). This is easy for small N and the hope is 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 $H = 14.1$. Redundancy is high at 0.77.

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

Rolf Landauer in 1996:

> *Information is not a disembodied abstract entity; it is always tied to a physical representation. It is represented by an engraving on a stone tablet, a spin, a charge, a hole in a punched card, a mark on paper, or some other equivalent. This ties the handling of information to all the possibilities and restrictions of our real physical world, its laws of physics and its storehouse of available parts.*

Landauer realized that only erasing a bit carries a thermodynamical cost (dissipates heat), all other operations are free – in principle, not using current technology where a computer consists mostly of a cooling unit (as opposed to reversible computation).

How much information is needed to specify an apple completely?

In classical physics this question makes little sense (how do you describe the position of an electron?).

But according to quantum physics there is a certain granularity in all physical systems – the real world is actually discrete.

For each particle making up the apple we need a few bits (order of a few hundred at most) to describe its state. So the total number of bits is approximately the same as the number of particles making up the apple, something like

$$10^{25} = 10000000000000000000000000$$

Not bad at all. For a computer scientist this is a tiny number, think $A(10, 10)$.

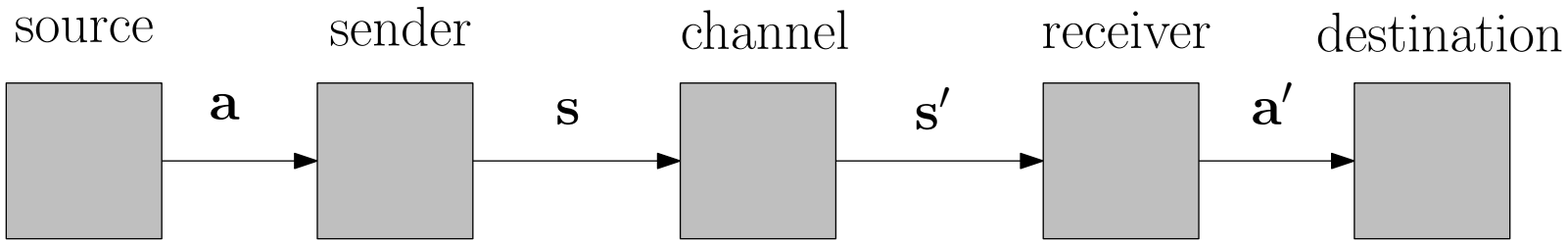
OK, so it's more information than stored in all books and computers on planet Earth, but not hugely much more.

- Information Theory

2. Codes

- Linear Codes
- Hamming Codes
- Cyclic Codes *

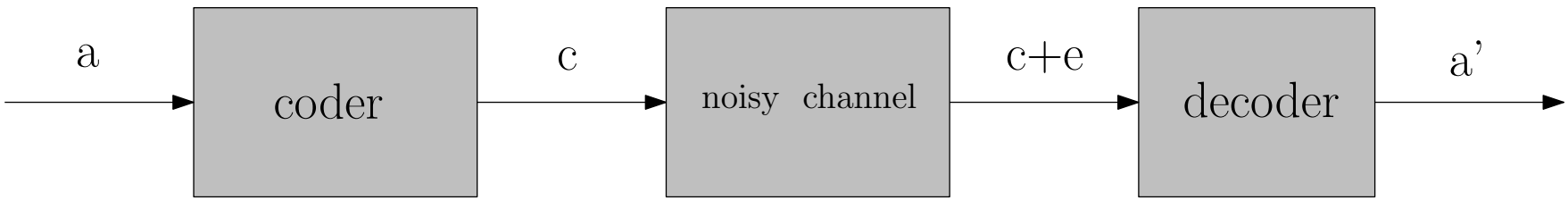
A message a is generated by some source, converted into a signal by the sender (transmitter) which is sent over the channel. The receiver turns the signal into a received message a' which reaches the destination.



Two major scenarios:

- The channel is noiseless: $s = s'$.
  Interesting question: how to protect against eaves-dropping, cryptography.
- The channel is noisy: $s \approx s'$.
  Interesting question: how to recover the original message, coding theory.

The coder turns a message $\boldsymbol{a}$ is turned into a code $\boldsymbol{c}$, which is sent over a noisy channel.



At the other end, 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 . . .

Suppose we have an m-ary channel.

We assume that there is a single parameter p that describes error probabilities: p is the probability that the channel will transmit any symbol a as a different symbol b.

Thus the probability depends neither on a nor on b, it is uniform for all $a \neq b$.

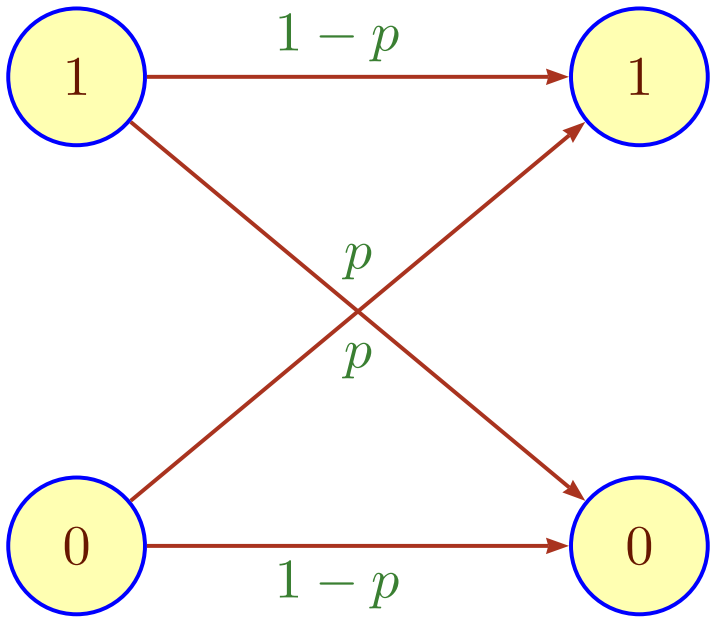
The probability that a is wrongly transmitted is thus $(m-1)p$.

The probability that a is correctly transmitted is $1-(m-1)p$.

We will always assume that p is small relative to m (see below). If not, the problem is really one of channel design, not of coding theory.

We will not deal with insertions and deletions of symbols.

In particular in a binary channel we have



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: We can always think of Q as being some finite field.

So a message and its code are just vectors of field elements. In particular in the binary case we are just dealing with bit-vectors.

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

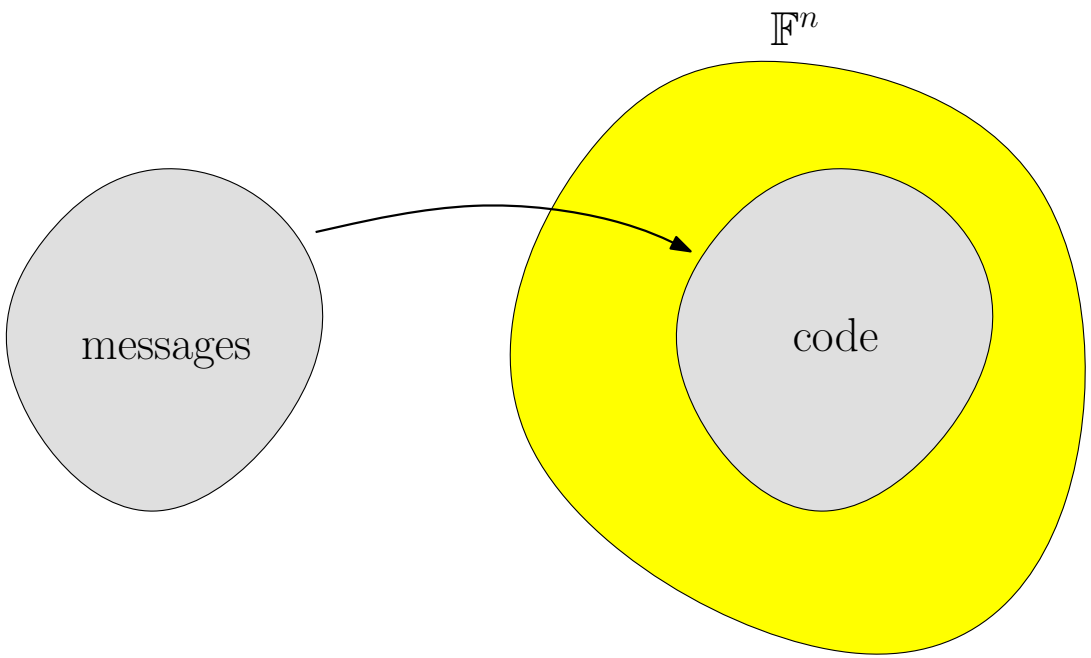
- Error Detection
  Detect when $\boldsymbol{e} \neq 0$ (and ask for a re-transmit).
- Error Correction
  Detect an error and correct it (by adding $-\boldsymbol{e}$ to the transmitted message).

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.

We have q^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. Here k is the dimension of the code and n is its length. $\mathbb{F}^n$ is the codespace. The elements of C are codewords.



Of course, equal cardinality of message space and code is not enough: we need to worry about coding algorithms that translate a message $\boldsymbol{x} \in \mathbb{F}^k$ into the corresponding codeword $f(\boldsymbol{x}) \in \mathbb{F}^n$ and decoding algorithms that go in the opposite direction, and deal with errors.

As we will see, there is usually quite a bit of flexibility in assigning codewords to messages; there is no magic connection between the two.

Before we talk about specific codes let's look at some fundamental properties of this setup.

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

Definition

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

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

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

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

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

Exercise

Check that d really is a metric.

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

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

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 would like simple codes that can correct reasonably many errors, the more the merrier.

Strategy: The main strategy is also clear: we will used redundancy to protect against errors. Alas, the details are complicated.

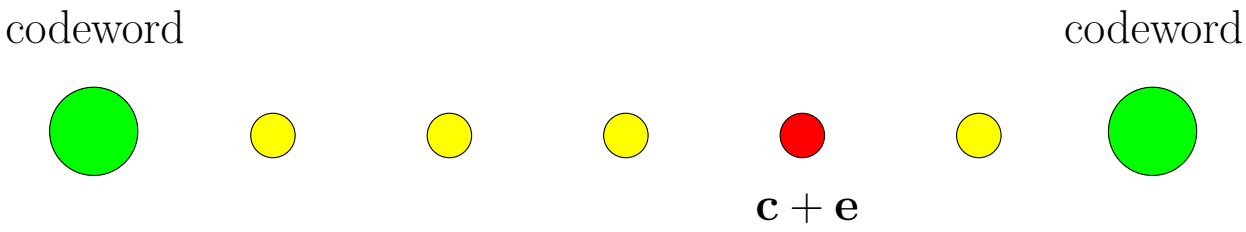
In coding theory texts, when one says a code detects e errors, what is actually meant is this:

For any codeword $\boldsymbol{x}$ and any error $\boldsymbol{e}$ of weight at most e:

- one can tell that $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e} \notin C$, and
- one can determine the weight of $\boldsymbol{e}$ from $\boldsymbol{y}$.

Being able to determine whether some symbol sequence is a codeword or not is still useful, but this terminology actually makes more sense.

Suppose the received message $\boldsymbol{c} + \boldsymbol{e}$ winds up "between" two codewords at distance 6.



This transmission could be caused by an error of weight 2 or an error of weight 4. We can only detect errors of weight at most 3 in this case; we can correct errors of weight at most 2.

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

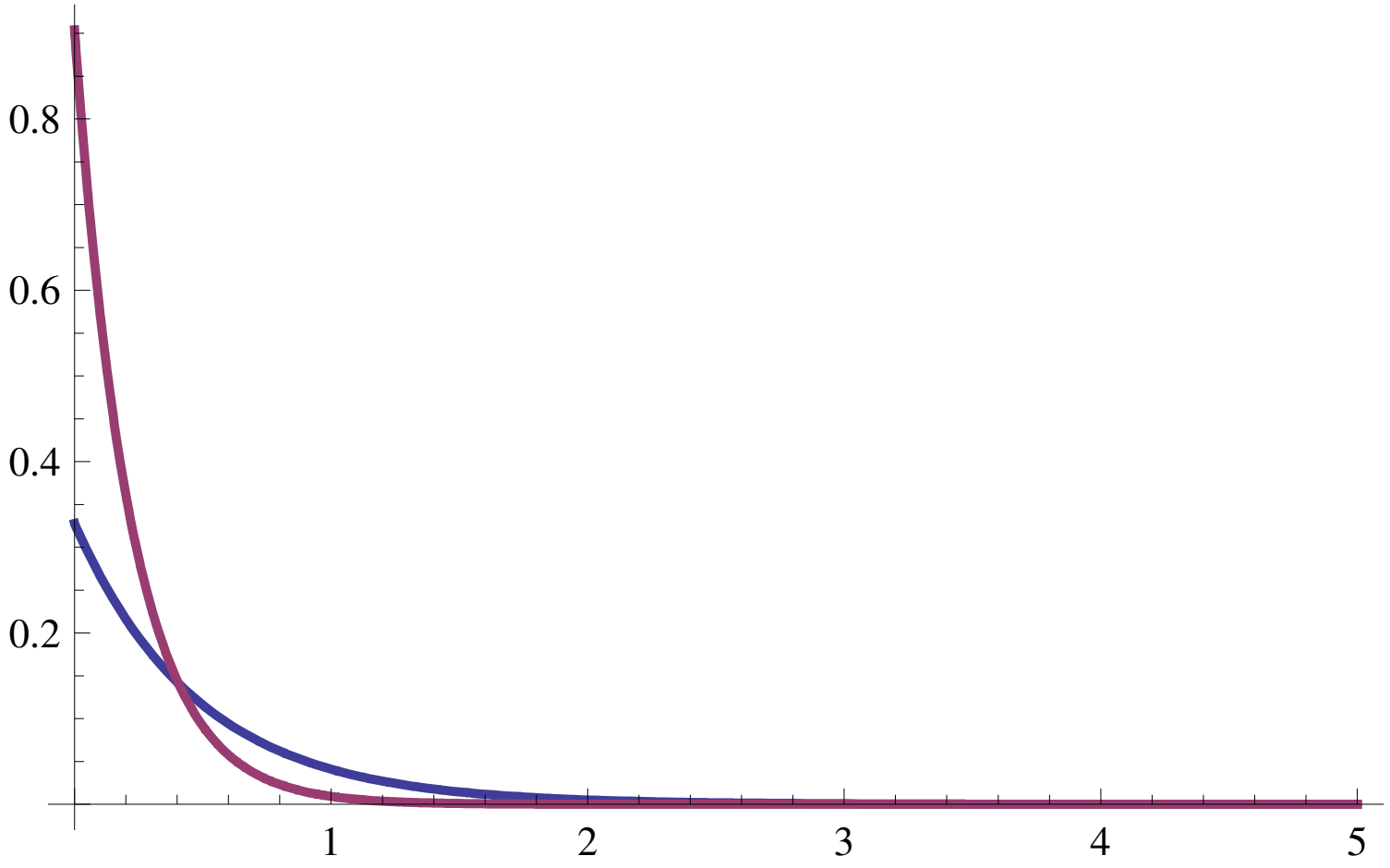
The probability that $\boldsymbol{y}$ is received when $\boldsymbol{x}$ as been transmitted is

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

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

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

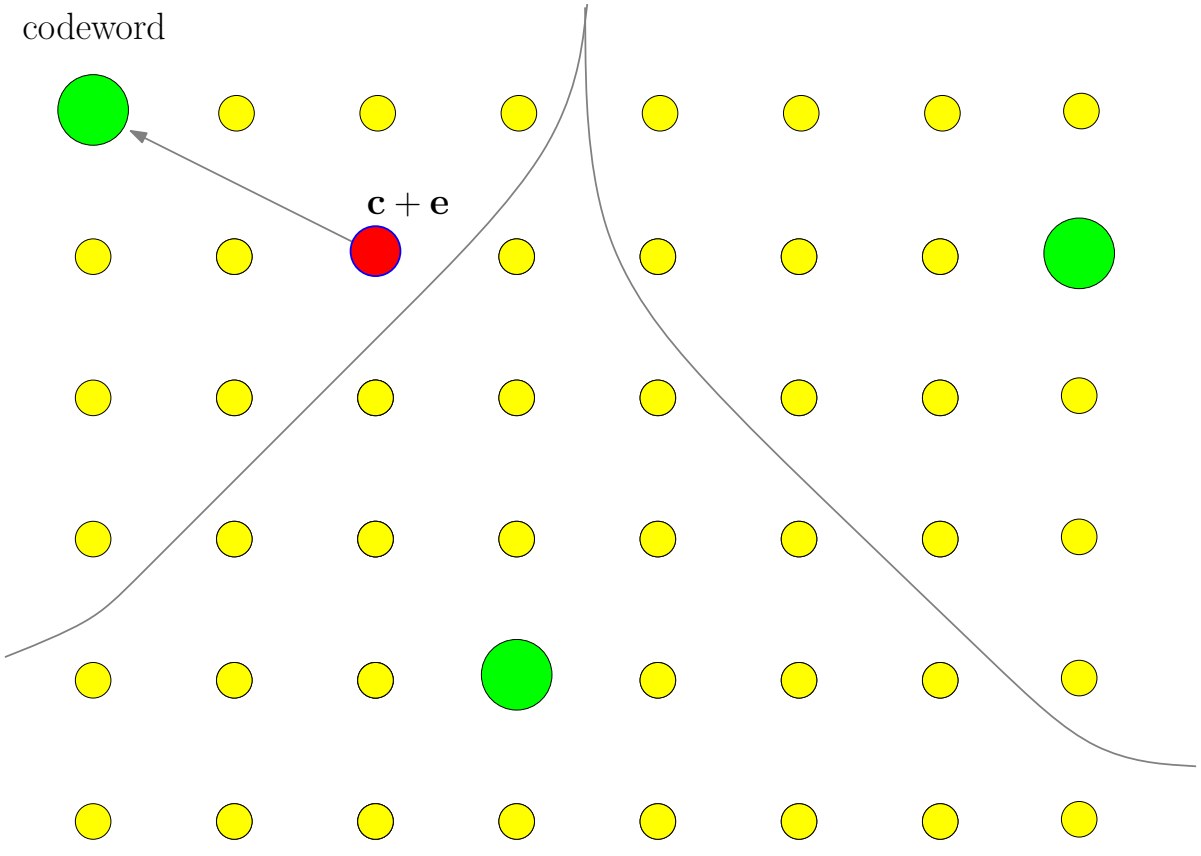
As already mentioned, if p is too large one has to redesign the channel.



This behavior is intuitively clear, but it never hurts to check the math.

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.



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.

Lemma (Hamming)

Suppose q-ary code C has size M, length n and corrects e errors. Then

$$M \sum_{i=0}^{e} \binom{n}{i} (q-1)^i \leq q^n$$

Proof.

To see this, consider (necessarily disjoint) balls of radius e centered at $\boldsymbol{x} \in C$. The number of elements in a ball at distance i to the center is $\binom{n}{i}(q-1)^i$. □

Note that the code size M is forced to be smaller when e becomes larger.

This is a bit embarrassing, but bear with me. Let $Q = \mathbf{2}$ and code

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

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

A repetition code detects $n-1$ errors.

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

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

For any Q, code

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

so that $n = k + 1$.

A parity check code detects 1 error.

Alas, it corrects none.

This may sound useless, but it is not: variants of this scheme are used in ISBN.

Let $q = 2$ and $k = 4$.

We transmit 7 bits

$$\boldsymbol{c} = (c_1, c_2, c_3, c_4, c_5, c_6, c_7)$$

Here $c_3 = x_1$, $c_5 = x_2$, $c_6 = x_3$, $c_7 = x_4$ are the original message bits. Then pick c_1, c_2, c_4 such that

$$\gamma = c_1 + c_3 + c_5 + c_7 = 0$$
$$\beta = c_2 + c_3 + c_6 + c_7 = 0$$
$$\alpha = c_4 + c_5 + c_6 + c_7 = 0$$

For decoding, calculate α, β and γ. If there is a single error then $\alpha\beta\gamma$ is the binary expansion of the place where the error occurred. Nice hack.

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

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

To fix this problem we can add one more bit

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

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

- Information Theory
- Codes

3. Linear Codes

- Hamming Codes
- Cyclic Codes *

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

Definition

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

If C has dimension k we refer to it as an $[n, k]$ code.

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

It follows that C is the row space of G:

$$C = \{ \boldsymbol{x}\, G \mid \boldsymbol{x} \in \mathbb{K} \}$$

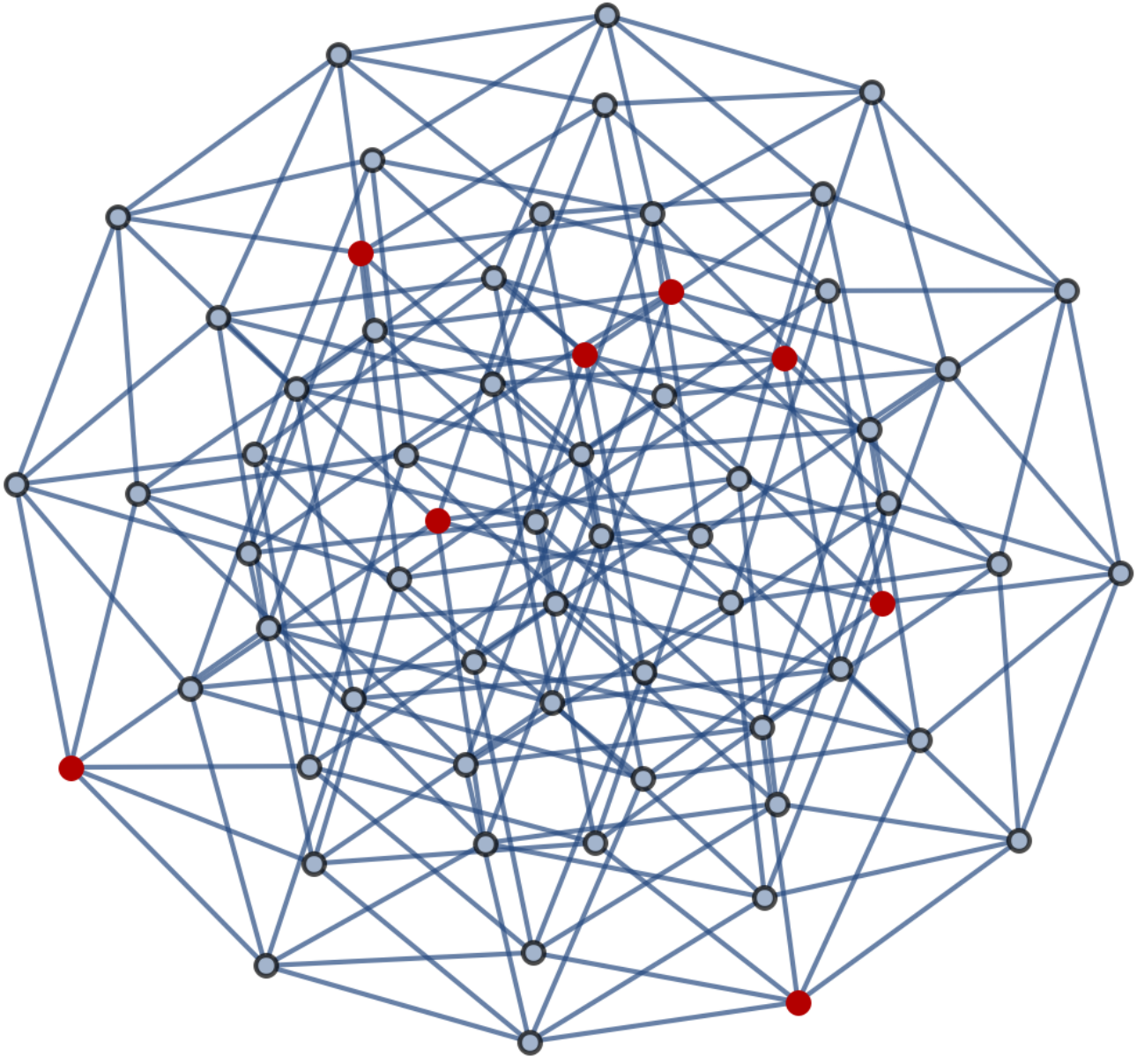
Let $Q = \mathbf{2}$, $k = 3$, $n = 6$ and

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

Then

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

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



Note that our generator matrix has the form

$$G = (I_k \, P)$$

where I_k is the identity matrix of order k (so G is in reduced echelon form). Such codes are said to be in standard form. A slightly weaker condition is that the columns of I_k can be spread out through G; this is called systematic form. This is no real constraint, one can always permute symbols in the codewords to get standard form.

Correspondingly 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.

Let $G = (I_k\ P)$ 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 $\mathrm{syn}(\boldsymbol{x}) = \boldsymbol{x} \cdot H^T$ is the syndrome of $\boldsymbol{x}$.

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

$$\boldsymbol{x} \in C \iff (\boldsymbol{x}) = \mathbf{0}$$

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

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.

Note that for our example code $(\boldsymbol{y}) = (s_1, s_2, s_3)$ where

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

So suppose $(\boldsymbol{y}) = (1, 0, 1)$. It follows that $\boldsymbol{e} = \boldsymbol{e}_2$ and we can correct the error by flipping the second bit.

Similarly we can handle all syndromes other than $(1, 1, 1)$: we can correct all 1-bit errors.

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* $\boldsymbol{e}$ *(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.

How much have we gained?

If we transmit 3 bits in a symmetric binary channel with error probability $p = 0.001$ the likelihood of no error is $(1-p)^3 = 0.997003$, roughly 3 times larger than a single error.

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

$$(1-p)^6 + 6(1-p)^5 p + (1-p)^4 p^2 = 0.999986$$

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

In improvement of roughly a factor of 200, but we have to send twice as many bits.

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.

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

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

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

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.

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. E.g., it may happen that $X^\perp = X$. For example, consider $X = \{\, (x_1, x_1, x_2, x_2, \ldots, x_k, x_k) \mid \boldsymbol{x} \in \mathbb{F}_2^k \,\}$.

Recall that a code with a $k \times n$ generator matrix has a $(n-k) \times n$ parity check matrix.

Here is a good way to think about check matrices: there 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.

How do we decode a linear code systematically?

Suppose $\boldsymbol{c} \in C$ is sent but $\boldsymbol{x} = \boldsymbol{c} + \boldsymbol{e}$ is received.

Obviously it suffices to compute the error vector $\boldsymbol{e}$.

Note that

$$\boldsymbol{x} + C = \boldsymbol{e} + C$$

so the error vector lies in the same coset as the received message. Since C is a subgroup of the additive group of $\mathbb{F}^n$, we can decompose $\mathbb{F}^n$ into $n - k$ disjoint cosets

$$\boldsymbol{a}_1 + C, \boldsymbol{a}_2 + C, \ldots, \boldsymbol{a}_{n-k} + C$$

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.

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

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

So we have to precompute a minimum weight vector for each possible syndrome.

For our standard $[6, 3]$ example we have the following coset leaders, parametrized by syndrome (which is none other than 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

Ideally we would like to enumerate the code and the coset leaders

$$C = \boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_q$$
$$E = \boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_r$$

and then store a $r \times q$ table, the so-called standard array, with $\boldsymbol{e}_i + \boldsymbol{c}_j$ in position i, j.

To decode we look up $\boldsymbol{x} = \boldsymbol{e}_i + \boldsymbol{c}_j$ and return $\boldsymbol{c}_j$ as decoded message.

The problem is that $q = p^k$ and $r = p^{n-k}$, so the table has size q^n, the same as the codespace (duh). Usually this is more information than we can store.

Also note that no one would actually store the table: one would then have to search for $\boldsymbol{x}$. Instead, one would hash the j index on the codespace.

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 $\boldsymbol{x}$:

- Compute $\boldsymbol{s} = \mathrm{syn}(\boldsymbol{x})$.
- Then look up the corresponding $\boldsymbol{e}$.
- Return $\boldsymbol{x}' = \boldsymbol{x} - \boldsymbol{e}$.

- Information Theory
- Codes
- Linear Codes

4. Hamming Codes

- Cyclic Codes *

The purpose of computation is insight, not numbers.

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.

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.

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.

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. □

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 SECDEC code (single error correction, double error detection) and used in memory systems.

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.

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 $\mathrm{md}(C) = 2e+1$.

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.

- Information Theory
- Codes
- Linear Codes
- Hamming Codes

5. Cyclic Codes *

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.

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

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

This is also called a code polynomial.

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 shift corresponds simply to multiplication by x.

Very nice.

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. □

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.

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

$$\boldsymbol{c} \in C \iff c(x)\, h(x) = 0 \text{ in } \mathbb{F}_q[x]/(x^n - 1)$$

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.

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:

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

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.

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).

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.