CDM

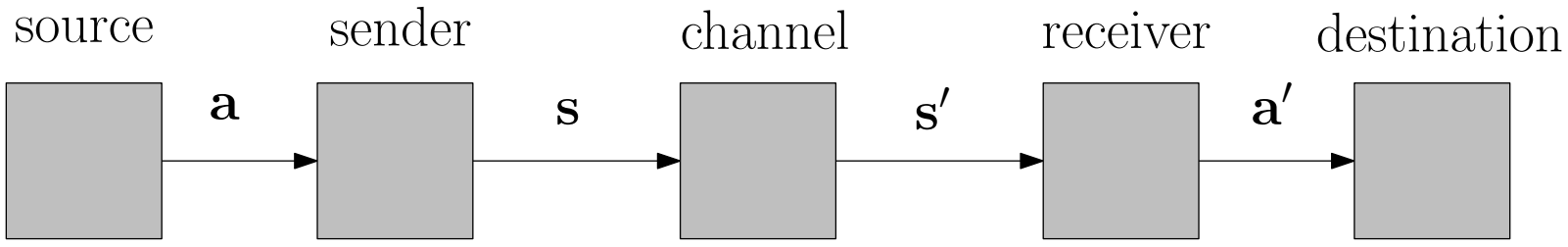
Finite Fields and Codes

Klaus Sutner

Carnegie Mellon University

codes 2018/3/2 14:07

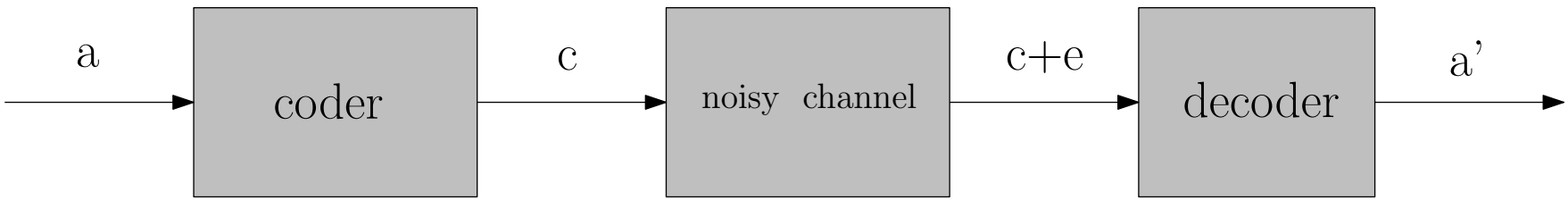
A message a is generated by some source, converted into a signal by the sender (transmitter) which is sent over the channel. The receiver turns the signal into a received message a' which reaches the destination.



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 $a \neq b$.

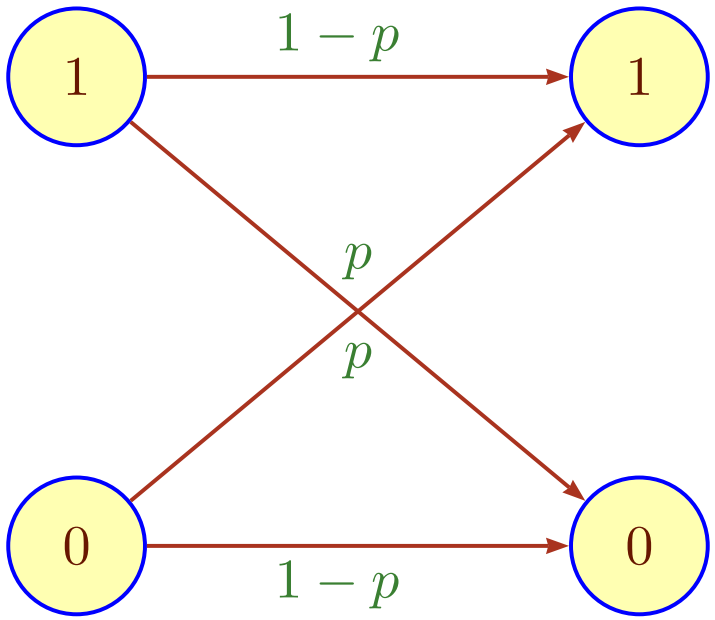
The probability that a is wrongly transmitted is thus $(m-1)p$.

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

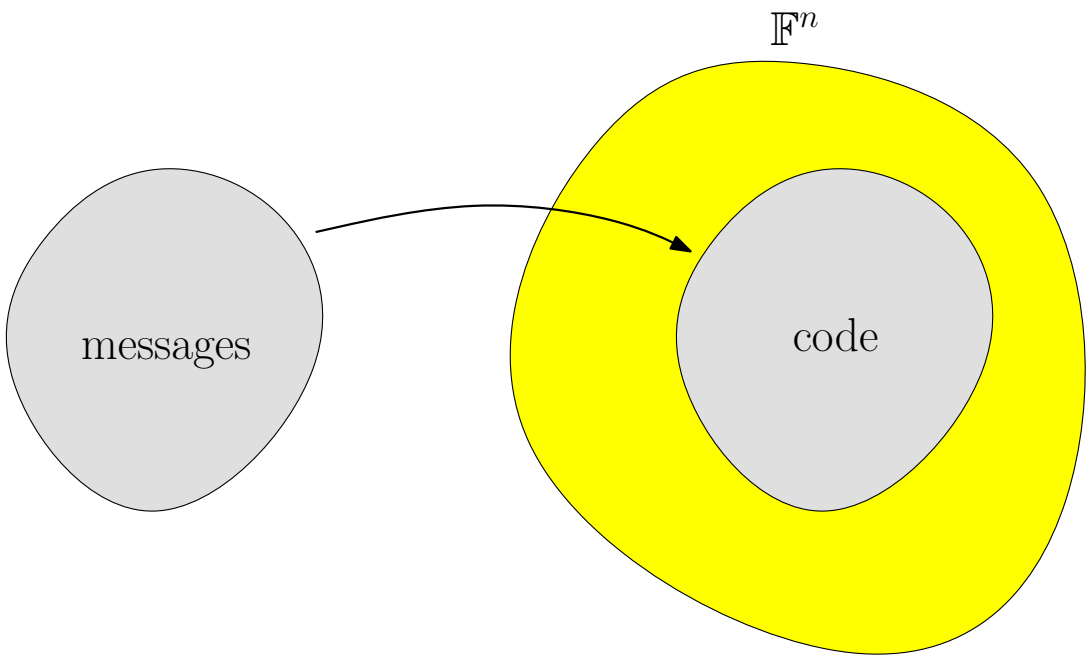
- Error Detection
  Detect when $\boldsymbol{e} \neq 0$ (and ask for a re-transmit).
- Error Correction
  Detect an error and correct it (by adding $-\boldsymbol{e}$ to the transmitted message).

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.

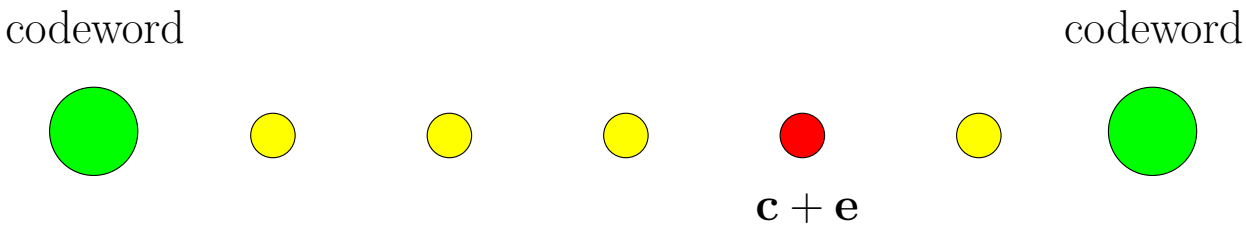
In coding theory texts, when one says a code detects e errors, what is actually meant is this:

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 $(m-1)p$.

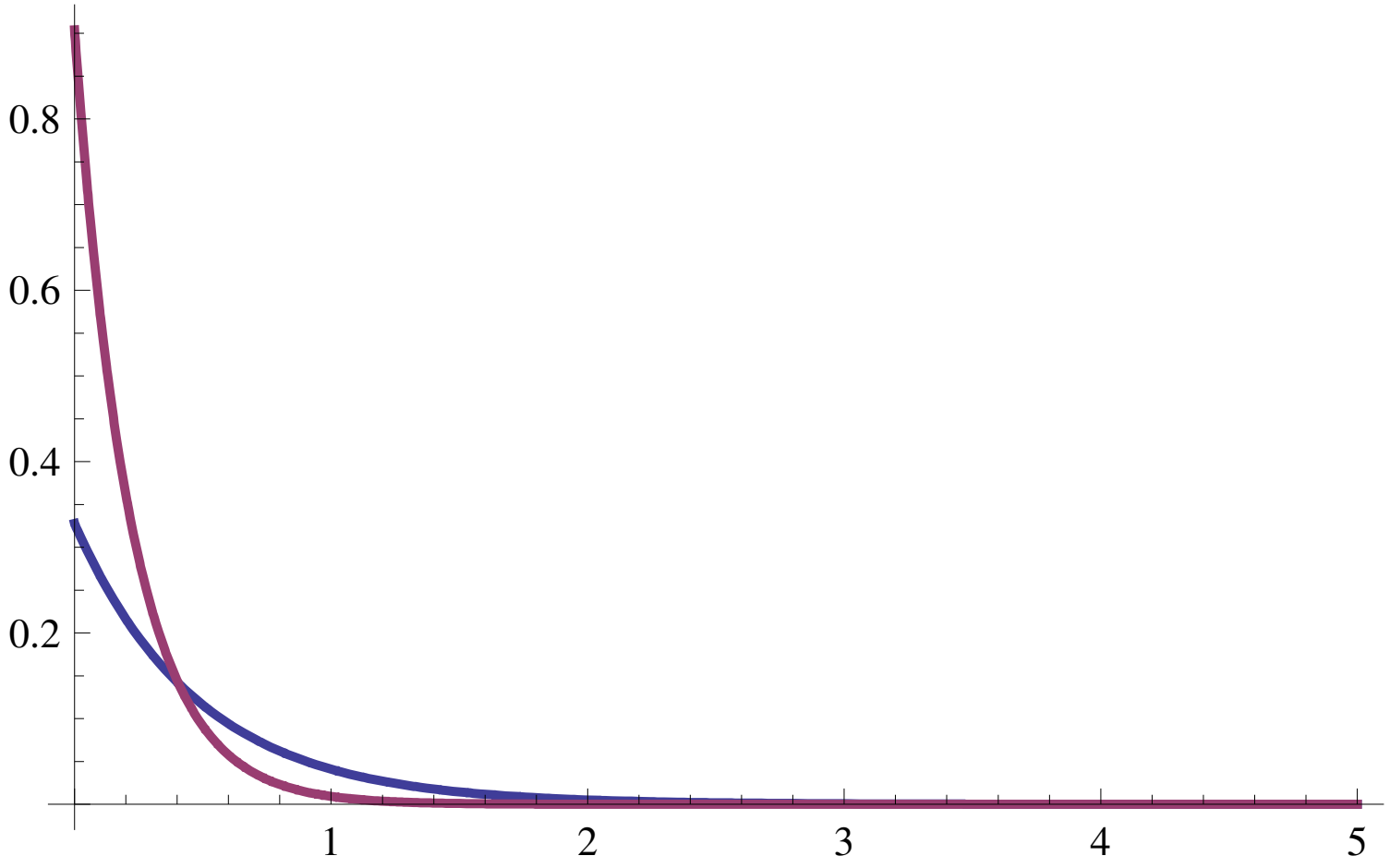
The probability that $\boldsymbol{y}$ is received when $\boldsymbol{x}$ as been transmitted is

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

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 $\boldsymbol{x}$.

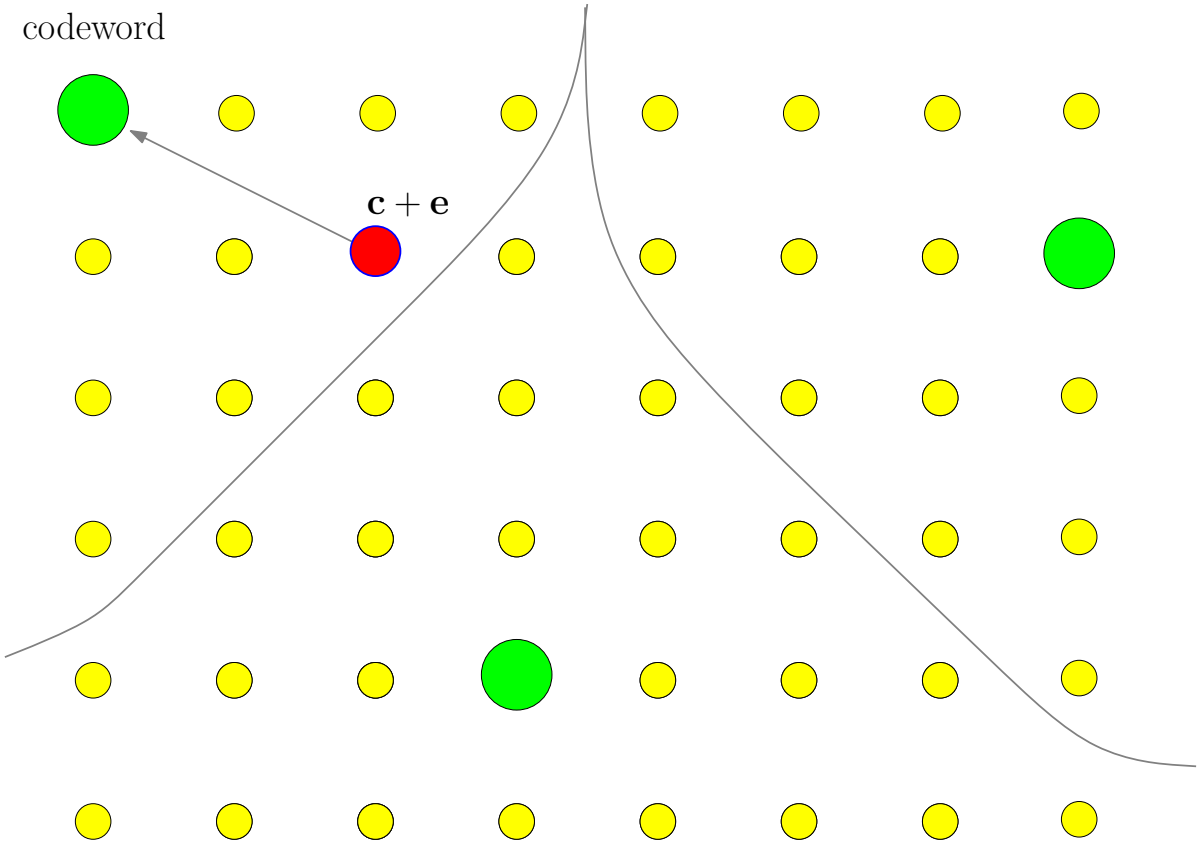
As already mentioned, if p is too large one has to redesign the channel.



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).

- Codes

2. Linear 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:

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

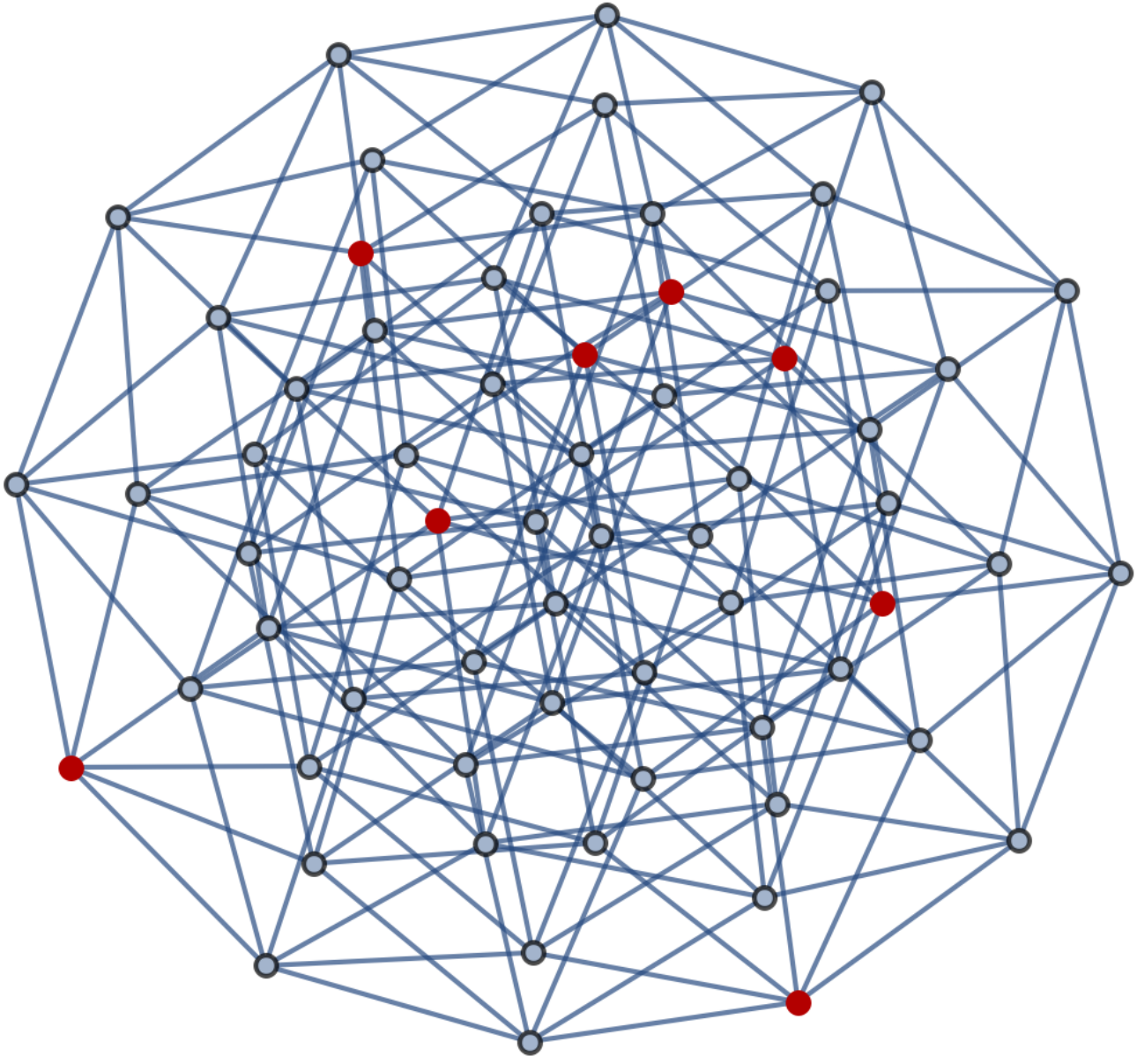
Let $Q = \mathbf{2}$, $k = 3$, $n = 6$ and

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

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?



We want the generator matrix to have full rank, which is easily accomplished by insisting on 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. Alternatively, we can spread the columns of I_k through G; this is called systematic form.

In standard form, 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

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

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 $\mathrm{syn}(\boldsymbol{y}) = (s_1, s_2, s_3)$ where

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

So suppose $\mathrm{syn}(\boldsymbol{y}) = (1, 0, 1)$. 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 $\boldsymbol{x}' = \boldsymbol{x} - \boldsymbol{e}$.

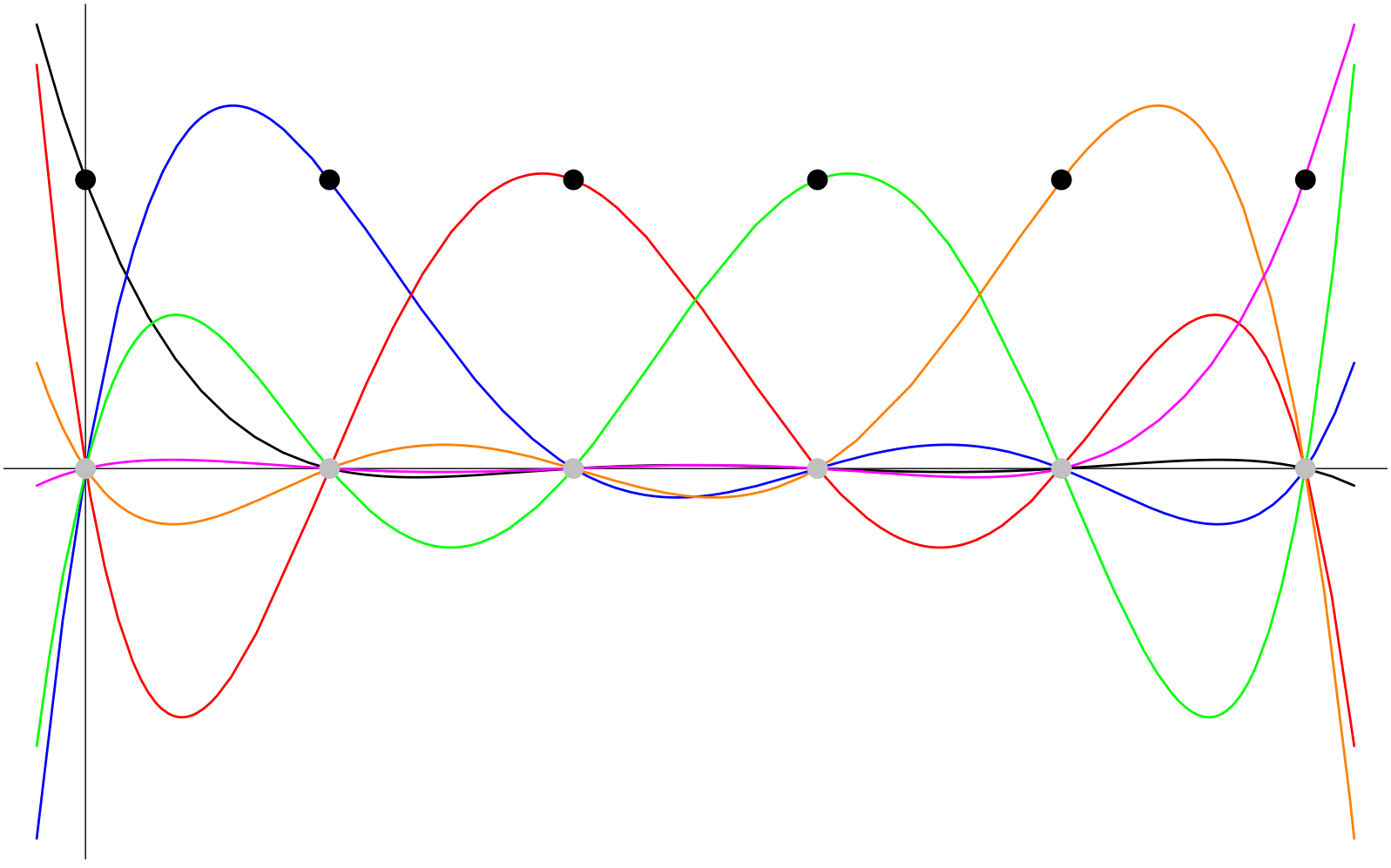
On occasion, a linear code may be described in a way that obscures linearity a bit.

Suppose we have a field $\mathbb{F}_p$. Given k field elements $a_0, \ldots, a_{k-1}$ we can define a polynomial $P_{\boldsymbol{a}}(t) \in \mathbb{F}_p[t]$ by

$$P_{\boldsymbol{a}}(t) = a_{m-1}t^{m-1} + a_{m-2}t^{m-2} + \ldots a_1 t + a_0$$

We can evaluate this polynomial in $n \geq m$ places to obtain a code vector $\boldsymbol{c} \in \mathbb{F}_p^n$.

Note that $P_{\boldsymbol{a}}$ and thus $\boldsymbol{a}$ can be reconstructed from $\boldsymbol{c}$: a polynomial of degree d is determined by its values in (at least) places: we can use Lagrange interpolation.



To see that this is actually a linear code, recall Vandermonde matrices:

$$V(\gamma_1, \ldots, \gamma_m) = \begin{pmatrix} 1 & \gamma_1 & \gamma_1^2 & \cdots & \gamma_1^{n-1} \\ 1 & \gamma_2 & \gamma_2^2 & \cdots & \gamma_2^{n-1} \\ 1 & \gamma_3 & \gamma_3^2 & \cdots & \gamma_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \gamma_m & \gamma_m^2 & \cdots & \gamma_m^{n-1} \end{pmatrix}$$

So this is an $m \times n$ matrix where each row is a geometric progression, and for $n = m$ we have determinant $\prod_{i<j}(\gamma_j - \gamma_i)$.

Thus, if all the γ_i are distinct, the matrix has full rank.

Evaluation of $P_{\boldsymbol{a}}(t)$ in positions $0, 1, \ldots, n-1$ is given by the linear map $\boldsymbol{a} \cdot W$ where W is the transpose of the Vandermonde matrix $V(0, 1, \ldots, n-1)$ (all to be construed as field elements).

$$W = \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 0 & 1 & 2 & \ldots & n-1 \\ 0^2 & 1^2 & 2^2 & \ldots & (n-1)^2 \\ & & \vdots & & \\ 0^{k-1} & 1^{k-1} & 2^{k-1} & \ldots & (n-1)^{k-1} \end{pmatrix}$$

Lemma

The Reed-Solomon code over $\mathbb{F}_p$ with parameters $k \leq n \leq p$ has weight $n - k + 1$.

Hence, choosing $n = k + 2e$ the code corrects e errors.

Proof.

It suffices to find a polynomial that produces the weight $n - k + 1$:

$$t(t-1)(t-2)\dots(t-(k-2))$$

□

$p = 13,\ k = 6,\ n = 9$

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 0 & 1 & 4 & 9 & 3 & 12 & 10 & 10 & 12 \\ 0 & 1 & 8 & 1 & 12 & 8 & 8 & 5 & 5 \\ 0 & 1 & 3 & 3 & 9 & 1 & 9 & 9 & 1 \\ 0 & 1 & 6 & 9 & 10 & 5 & 2 & 11 & 8 \end{pmatrix}$$

For example,

$$P_{\boldsymbol{a}}(5) = a_0 + 5a_1 + 12a_2 + 8a_3 + a_4 + 5a_5$$

The real reason Reed-Solomon codes are interesting and heavily used is that there is a fast decoding algorithm due to Berlekamp and Welch from 1983.

Alas, they decided to patent their algorithm. This is total insanity, patents should not apply to

> . . . laws of nature, natural phenomena, and abstract ideas.

These type of patents have one main function: to obstruct progress in the interest of enriching the few. They are an abomination.