Outline

Recall: For **integers**
- Euclidean algorithm for finding gcd’s
- Extended Euclid for finding multiplicative inverses
- Extended Euclid for computing Sun-Ze
- Test for primitive roots

And for **polynomials** with
  - coefficients in $\mathbb{F}_2 = \mathbb{Z}/2$
  - Euclidean algorithm for gcd’s
  - Concept of equality mod $M(x)$
  - Extended Euclid for inverses mod $M(x)$

Looking for **good codes**
- High info rate vs. high min distance
- Hamming bound for arbitrary codes
- Idea of **linear** codes
- Gilbert-Varshamov bound for linear codes
Review: Hamming bound

Using the physical analogy that Hamming distance is really like *distance*:

the set of all length *n* codewords with an alphabet with *q* letters (maybe *q* = 2) is like a *container*

codewords with specified minimum distance *d* = 2*e* + 1 between are like *balls of radius e*

and the question of how many codewords of length *n* (alphabet size *q*) with minimum distance *d* = 2*e* + 1 can be chosen is analogous to asking

*How many balls of a fixed radius can be packed into a box with a specified volume?*

This is hard, but an easier version is clear:

the total volume of the balls packed cannot be greater than the volume of the container.
Here total volume is the number of length $n$ words on a $q$-character alphabet, namely $q^n$.

The volume of a ball of radius $e$ centered at a word $w$ is the number of length $n$ words that differ from $w$ at $\leq e$ positions:

\[
\begin{align*}
\text{no. differing at 0 positions} & = 1 \\
\text{no. differing at 1 positions} & = \binom{n}{1} \cdot (q-1) \\
\text{no. differing at 2 positions} & = \binom{n}{2} \cdot (q-1)^2 \\
\text{no. differing at 3 positions} & = \binom{n}{3} \cdot (q-1)^3 \\
& \quad \vdots \\
\text{no. differing at } e \text{ positions} & = \binom{n}{e} \cdot (q-1)^e
\end{align*}
\]

So the volume is

\[
\text{volume of ball radius } e \text{ of dimension } n = 1 + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \ldots + \binom{n}{e}(q-1)^e
\]
Thus, with \( \ell \) codewords of length \( n \), alphabet with \( q \) characters, with minimum distance \( d = 2e + 1 \), the constraint that the sum of the volumes of the balls cannot be greater than the volume of the whole container in which they’re packed is

\[
q^n \geq \ell \cdot \left[ 1 + \binom{n}{1}(q - 1) + \ldots + \binom{n}{e}(q - 1)^e \right]
\]

This is the **Hamming bound**.

If a code exists with \( \ell \) codewords, of length \( n \), and minimum distance \( d = 2e+1 \), this inequality must hold.

The **contrapositive** assertion is that, given \( \ell, n, q, \) and \( d = 2e + 1 \) if the inequality fails then there cannot exist any such code.

**Remark:** Even when the equality holds, there is no assurance that a code exists. Failure to meet the Hamming bound can prove non-existence, but meeting the bound cannot prove existence.
Linear codes

Again, it is hard to make good codes, in the sense of having a family approaching Shannon’s Noisy Coding Theorem bound. Codes should also be easy to encode and decode.

A class of codes easiest to study is linear codes. A revised form of Shannon’s theorem shows this restricted class still includes good codes.

If you already know a good version of linear algebra, you can understand linear codes in those terms: linear codes of length $n$ with alphabet $F_q$ are simply vector subspaces of the vector space $F_q^n$ of all column vectors of size $n$ with entries in the field $F_q$ with $q$ elements.

Yes, codes are just vector subspaces.
Example: Hamming binary $[7, 4]$ code

The first constructions of good codes (also easily decodable) was about 1952, due to Hamming. The notation $[7, 4]$ means the codewords are length 7 and the dimension (defined shortly) of the code is 4.

The source words are the $16 = 2^4$ binary words of length 4: 0001, 0010, 0011, 0100, 0101, 0110, etc. To each such word $abcd$ Hamming adds redundancy bits in a clever pattern:

$$abcd \text{ becomes } abcdefg$$

where

$$e = b + c + d$$

$$f = a + c + d$$

$$g = a + b + d$$
For example, encode

\[
\begin{align*}
1000 & \rightarrow 1000011 \\
0100 & \rightarrow 0100101 \\
0010 & \rightarrow 0010110 \\
0001 & \rightarrow 0001111
\end{align*}
\]

**Hamming decoding** is a further good feature. Write codewords as **vectors**

\[
\begin{align*}
1000011 &= (1, 0, 0, 0, 0, 1, 1) \\
0100101 &= (0, 1, 0, 0, 1, 0, 1)
\end{align*}
\]

(The **components** of these vectors are in \( \mathbb{F}_2 \).)

Define other vectors

\[
\begin{align*}
r &= (0, 0, 0, 1, 1, 1, 1) \\
s &= (0, 1, 1, 0, 0, 1, 1) \\
t &= (1, 0, 1, 0, 1, 0, 1)
\end{align*}
\]

Use the **dot product** defined by

\[
(x_1, \ldots, x_n) \cdot (y_1, \ldots, y_n) = x_1y_1 + x_2y_2 + \ldots + x_ny_n
\]

(inside \( \mathbb{F}_2 \)).
A source word such as 0100 is encoded as $x = (0, 1, 0, 0, 1, 0, 1)$. Suppose $y = (1, 1, 0, 0, 1, 0, 1)$ is received. **Hamming decoding** computes 3 inner products (in $\mathbf{F}_2$)

\[
\begin{align*}
y \cdot r &= (1, 1, 0, 0, 1, 0, 1) \cdot (0, 0, 0, 1, 1, 1, 1) = 0 \\
y \cdot s &= (1, 1, 0, 0, 1, 0, 1) \cdot (0, 1, 1, 0, 0, 1, 1) = 0 \\
y \cdot t &= (1, 1, 0, 0, 1, 0, 1) \cdot (1, 0, 1, 0, 1, 0, 1) = 1
\end{align*}
\]

Interpret the triple of inner products as a binary integer:

\[
001 = 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 1 \text{ (in decimal)}
\]

Hamming decoding says (correctly) that the received word had an error in the first position.
Another example of Hamming decoding:

If \( x = (0, 1, 0, 0, 1, 0, 1) \) was sent and \( y = (0, 1, 1, 0, 1, 0, 1) \) was received (error in the third position)

\[
\begin{align*}
y \cdot r &= (0, 1, 1, 0, 1, 0, 1) \cdot (0, 0, 0, 1, 1, 1, 1) = 0 \\
y \cdot s &= (0, 1, 1, 0, 1, 0, 1) \cdot (0, 1, 1, 0, 0, 1, 1) = 1 \\
y \cdot t &= (0, 1, 1, 0, 1, 0, 1) \cdot (1, 0, 1, 0, 1, 0, 1) = 1
\end{align*}
\]

In binary

\[
011 = 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 3 \text{ (in decimal)}
\]

which (correctly) tells that the third bit is wrong.
The **information rate** of the Hamming \([7, 4]\) code is

\[
\text{rate} = \frac{\log_2 2^4}{7} = \frac{4}{7}
\]

so there is a cost.

*Compare the Hamming \([7, 4]\) code to doing nothing, and also to adding a parity-check bit?*

On a channel with bit error probability \(1/8\), word error probability is

\[
\text{word error} = 1 - \text{probability no bit flipped}
\]

\[
= 1 - (7/8)^4 \approx 1 - 0.5862 \approx 0.4138
\]

With a parity-check bit, the probability of an **uncorrectable** error goes up, since the parity-check bit itself may get flipped, but we can **detect** a single bit error, though not correct it.

With Hamming, probability of a correctable error is

\[
\text{word error prob} = 1 - \left(\frac{7}{8}\right)^7 - \left(\frac{7}{8}\right)^6 \left(\frac{1}{8}\right)
\]

\[
\approx 1 - 0.3436 - 0.3436 \approx 0.2146 \ll 0.4138
\]
With bit error probability 1/12, raw word error probability is

\[ 1 - (\frac{11}{12})^4 \approx 0.2939 \]

but for Hamming [7, 4] it is

\[ 1 - \left( \left( \frac{11}{12} \right)^7 + \left( \frac{11}{12} \right)^6 \frac{1}{12} \right) \approx 0.1101 \]

With word error 1/20, word error for do-nothing encoding is

\[ 1 - (\frac{19}{20})^4 \approx 0.18549 \]

while for Hamming [7, 4] it is

\[ 1 - \left( \left( \frac{19}{20} \right)^7 + \left( \frac{19}{20} \right)^6 \frac{1}{20} \right) \approx 0.0444 \]

**Remark:** Hamming [7, 4] can correct single bit errors, by converting 4-bit words into 7-bit words cleverly. What about 2-bit errors, etc?
Review: linear algebra

Let $F$ be the scalars: the real numbers $\mathbb{R}$, the complex numbers $\mathbb{C}$, the rational numbers $\mathbb{Q}$, the finite field $\mathbb{F}_2$, the finite field $\mathbb{Z}/p = \mathbb{F}_p$ for $p$ prime, etc.

A vector of dimension $n$ is an ordered $n$-tuple of scalars, separated by commas and with parentheses on the ends.

Example: $(0, 1, 2, 3, 4, 5, 6)$ is a 7-dimensional vector, $(0, 0, 1, 2)$ is a 4-dimensional vector.

The set of all $n$-dimensional vectors with entries in $F$ is denoted

$$F^n = \{n\text{-dimensional vectors over } F\}$$

The scalars in a vector are called entries or components.

Remark: Sometimes the $i^{\text{th}}$ component of a vector $v$ is denoted $v_i$ but this is absolutely not reliable.
The **zero vector** (of whatever dimension) is the ordered tuple of 0s, denoted 0.

The **(vector) sum** of two vectors \( x = (x_1, \ldots, x_n) \) and \( y = (y_1, \ldots, y_n) \) is component-wise:

\[
x + y = (x_1, \ldots, x_n) + (y_1, \ldots, y_n)
= (x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n)
\]

The **scalar multiple** \( cx \) of a vector \( x = (x_1, \ldots, x_n) \) by a scalar \( c \) is obtained by multiplying each component by the scalar:

\[
cx = (cx_1, \ldots, cx_n)
\]
**Definition:** A linear combination of a collection of vectors \( v_1, \ldots, v_t \) is any other vector \( w \) expressible as

\[
w = c_1 v_1 + c_2 v_2 + \ldots + c_t v_t
\]

with scalars \( c_1, \ldots, c_t \).

**Definition:** A collection \( v_1, \ldots, v_t \) of vectors is **linearly dependent** if there is some linear combination (not with all coefficients \( c_i \)'s being 0) which is the zero vector:

\[
0 = c_1 v_1 + c_2 v_2 + \ldots + c_t v_t
\]

A collection \( v_1, \ldots, v_t \) of vectors is **linearly independent** if there is no linear combination (except with all coefficients 0) which is the zero vector.

**Definition:** A vector subspace of \( k^n \) is a set \( V \) of vectors of length \( n \) such that the **vector sum** of any two vectors in \( V \) is again in \( V \), and any **scalar multiple** of a vector in \( V \) is again in \( V \).
**Definition:** A set of vectors $v_1, \ldots, v_N$ spans a vector subspace $V$ of $k^n$ if every vector in $V$ is a linear combination of the $v_i$’s.

**Definition:** A set of vectors $v_1, \ldots, v_k$ in a vector subspace $V$ of $k^n$ is a **basis** for $V$ if the $v_i$ span $V$ and are **linearly independent**.

**Proposition:** For basis $v_1, \ldots, v_k$ of a vector subspace $V$, every vector $v \in V$ has a **unique** expression as a linear combination of the $v_1, \ldots, v_k$.

**Definition:** The **dimension** of a vector subspace $V$ of $k^n$ is the number of elements in any basis for $V$.

**Remark:** It is important to note that the *dimension* of a vector subspace is **not** the *length* of the vectors in it.

**Theorem:** (see appendix) Dimension of vector subspaces is **well-defined:** any two bases have the same number of elements.  

///
Remarks: These abstract definitions do not give any hints about how to do computations.

Row reduction will be our main device for
(*) Testing for linear independence
(*) Obtaining linear combination expressions
(*) Computing dimensions of subspaces

Much of the intuition we have from study of ‘vectors’ in a physics or advanced calculus or ‘concrete’ linear algebra context is useful even when the scalars are a finite field $\mathbb{F}_q$. 
Linear codes, GV bound

**Definition:** A linear code of dimension $k$, block length $n$ over alphabet $\mathbb{F}_q$ is a vector subspace $C$ of $\mathbb{F}_q^n$ of dimension $k$. These are denoted $[n, k]$-codes, for short.

**Definition:** A linear code of dimension $k$ and length $n$ with minimum distance $d$ is called an $[n, k, d]$-code.

**Theorem: Gilbert-Varshamov bound:** An $[n, k, d]$-code over alphabet $\mathbb{F}_q$ exists if

$$q^{n-k-1} > (q-1)\binom{n-1}{1} + \ldots + (q-1)^{d-2}\binom{n-1}{d-2}$$

In the simplest case of binary codes, an $[n, k, d]$-code over $\mathbb{F}_2$ exists if

$$2^{n-k-1} > \binom{n-1}{1} + \ldots + \binom{n-1}{d-2}$$

(Proof later.)
Remarks: The GV bound is a relation among parameters for linear codes which, if met, guarantee existence. The GV bound cannot prove non-existence.

This is opposite to the situation for the Hamming bound, which can prove non-existence, but never prove existence.

Remark: The Hamming bound applies to all codes, not just linear ones. An $[n, k, d]$ code over $\mathbb{F}_q$ has $q^k$ codewords in it, since (from the linear algebra definition of dimension) there is a basis $v_1, \ldots, v_k$ and everything is (uniquely!) expressible as a linear combination

$$c_1 v_1 + \ldots + c_k v_k$$

There are $q$ choices for $c_1$, $\ldots$, $q$ choices for $c_k$, so $q^k$ vectors in a $k$-dimensional vector space over $\mathbb{F}_q$. 
**Example:** Does there exist a binary \([5, 2, 3]\)-code?

This notation \([n, k, d] = [5, 2, 3]\) means that the block length is \(n = 5\), dimension is \(k = 2\), and minimum distance is \(d = 3\). The alphabet size is \(q = 2\). The binary GV bound is

\[
2^{n-k} - 1 > \binom{n-1}{1} + \cdots + \binom{n-1}{d-2}
\]

which here would be

\[
2^{5 - 2} - 1 > \binom{5-1}{1} + \cdots + \binom{5-1}{3-2}
\]

or

\[
2^3 - 1 > \binom{5-1}{1}
\]

which is

\[
7 > 4
\]

This is true, so such a code exists.
**Example:** Does there exist a binary $[5, 3, 3]$-code?  

This notation $[n, k, d] = [5, 3, 3]$ means that the block length is $n = 5$, dimension is $k = 3$, and minimum distance is $d = 3$. The alphabet size is $q = 2$. The binary GV bound is

$$2^{n-k} - 1 > \binom{n-1}{1} + \ldots + \binom{n-1}{d-2}$$

which here would be

$$2^{5-3} - 1 > \binom{5-1}{1} + \ldots + \binom{5-1}{3-2}$$

or

$$2^2 - 1 > \binom{5-1}{1}$$

which is

$$3 > 4$$

This is false, so we reach **no conclusion** from the GV bound.
Since the GV bound failed to prove \emph{existence} of a binary $[5,3,3]$ code, let's see if the Hamming bound can prove \emph{non-existence} of such a code:

The Hamming bound is

\[ q^n \geq \ell \cdot \left[ 1 + \binom{n}{1}(q-1) + \ldots + \binom{n}{e}(q-1)^e \right] \]

where $\ell$ is the number of codewords, $n$ is the length, $q$ is alphabet size. Here $n = 5$, $q = 2$, and minimum distance $d = 2e + 1 = 3$, so $e = 1$. As noted above, the number of codewords for a binary linear code of dimension $k$ is $2^k$. Thus, the Hamming bound would be

\[ 2^5 \geq 2^3 \cdot \left[ 1 + \binom{5}{1} \right] \]

or

\[ 32 \geq 2^3 \cdot 6 \]

or

\[ 32 \geq 48 \]

which is false, so there is \emph{no} binary $[5,3,3]$ code.