Ising model / repetition code: codewords

0 000 ... vs. 1111 ...

+ + ++ ... − − −− ...

↳ ground states of energy $E = -\sum_{i\sim j} z_i z_j$ where $z_i \in \{\pm 1\}$.

Write $x_i = \frac{1-z_i}{2} \in \{0,1\}$. $z_i z_j = +1 \Rightarrow x_i + x_j = 0 \pmod 2$

$= -1 \Rightarrow x_i + x_j = 1 \pmod 2$

parity-check code

It will be helpful to convert this to a more mathematical language.

addition + multiplication (mod 2) ⇒ work in field $\mathbb{F}_2$.

If bits $(x_1, \dots, x_n) \in \mathbb{F}_2^n$ (n-component vector)

1d repetition code: $x_1 + x_2 = 0$, $x_2 + x_3 = 0$, ...

$$\begin{pmatrix} 1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 1 & 0 & \cdots & 0 \\ & & \vdots & & & \\ 0 & 0 & 0 & \cdots & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

understood in $\mathbb{F}_2$.

parity-check matrix $H \in \mathbb{F}_2^{(n-1)\times n}$

codeword ⟺ right null vector $\underbrace{H\vec{x} = \vec{0}}$.

$\vec{x} \in \ker(H)$ [kernel]

for repetition code: $\ker(H) = \text{span}\begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} = \left\{ \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \right\}$

Generator matrix $G = \begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix}$ for codewords

so $HG^T = 0$. $\quad x \in \ker(H) \iff x \in \text{im}(G^T)$

So far this might seem like an overly complicated way to do the repetition code. BUT we have an obvious generalization...

other parity-check codes $\Rightarrow$ pick different matrix H (and G)

Wishlist: ① store more codewords $\Rightarrow$ $\ker(H)$ bigger?

Thm: $\ker(H)$ has dimension $k = n - \text{rank}(H)$

↳ vector space over $\mathbb{F}_2$. $\quad k = \#$ of logical bits

Goal: $k > 1$

② robustness to errors?

Define Hamming weight $|\vec{x}| :=$ # of 1s in $\vec{x}$.

Code distance $d = \min_{\substack{\vec{x} \in \ker(H) \\ \vec{x} \neq \vec{0}}} |\vec{x}|$ [# of errors needed to change codeword]

How many bit flip errors can we protect?

$\vec{x} \in \ker(H) \longrightarrow \vec{x} + \vec{e}$ ← error

Decoder looks for $\vec{y} \in \underbrace{\ker(H)}_{\text{valid codeword}}$ so $|\underbrace{(\vec{x}+\vec{e})+\vec{y}}_{=(\vec{x}+\vec{e})-\vec{y} \text{ in } \mathbb{F}_2^n}|$ is smallest?

Prop: decoder works if $|\vec{e}| \leq \left\lfloor \frac{d-1}{2} \right\rfloor$ [detect $\leq d-1$ codewords]

In practice, decoder works by measuring syndrome

$\vec{s} = H\vec{e} = H(\vec{e}+\vec{y})$ $\quad \vec{s}$ is invariant if we shift by a logical: $\vec{e} \to \vec{e} + \vec{y}$ (undetectable error!)

Look for $[n, k, d]$ classical codes w/ k & d "large"?

Repetition code is $[n, 1, n]$.

Example: 7-bit Hamming code.

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

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

$d=3$: smallest weight logical

$k=4$ linearly independent rows

Check: $HG^T = 0$

So this is a $[7, 4, 3]$ code

Like w/ Ising model we can also think of the Hamming codewords as the ground state of a classical Hamiltonian:

$$H_0 = -Z_1Z_3Z_5Z_7 - Z_2Z_3Z_6Z_7 - Z_4Z_5Z_6Z_7$$

This is invariant if I apply bit flips along any logical, e.g.

flip Z_1, Z_2, Z_3 $\Rightarrow$ $H_0 \to -(-Z_1)(-Z_3)Z_5Z_7 - (-Z_2)(-Z_3)Z_6Z_7 - Z_4Z_5Z_6Z_7$

$= H_0$

So the logical operations are again symmetries of H_0.

$d=3 \Rightarrow$ correct against any single bit error

higher-weight (4) checks of Hamming code are <u>useful</u>

If we used 4 repetition codes (weight 2 checks):

$$H = \left(\begin{array}{ccc|c} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{array}\right) \quad \text{etc.}$$

we'd have $[4 \cdot 3, 4 \cdot 1, 3] = [12, 4, 3]$

higher-weight checks $\Rightarrow$ better $\frac{k}{n}$ & $\frac{d}{n}$ vs. (multiple) repetition codes.

If we keep increasing check weight, what's possible?

Prop: Hamming bound: $[n,k,d]$ impossible if k, d both too large

Proof: Linear code $\Rightarrow$ $\vec{s} = H[\vec{e} + \vec{y}]$ has 2^k solutions for fixed $\vec{s}$.
($\vec{y}$: logical)

code distance $d \Rightarrow$ each $\vec{e}$ w/ $|\vec{e}| \le t$ has unique $\vec{s}$

if not: $H\vec{e}_1 + H\vec{e}_2 = \vec{s} + \vec{s} = 0 \Rightarrow \vec{e}_1 + \vec{e}_2 \in \ker(H)$
$\Rightarrow$ logical of weight $2t < d$.

# of possible $\vec{s} \le 2^{n-k}$

" " $\ge \sum_{j=0}^{t} \binom{n}{j}$ $\Rightarrow$ $k \le n - \log_2\left(\sum_{j=0}^{t}\binom{n}{j}\right)$

($\sum_{j=0}^{t}\binom{n}{j}$: # of $\vec{e}$ that map to unique $\vec{s}$)

Check: $[7,4,3]$ Hamming code saturates this bound!
$t=1$: $2^{7-4} = 8 = \binom{7}{0} + \binom{7}{1} = 1 + 7$

Asymptotics: if n, j are large: $\binom{n}{j} \to 2^{n h(j/n)}$ (h: binary entropy)

$h(x) = -x\log_2 x - (1-x)\log_2(1-x)$

So as $n \to \infty$: $\frac{k}{n} \le 1 - h\left(\frac{d}{2n}\right) \Rightarrow \frac{k}{n}, \frac{d}{n}$ can be finite as $n \to \infty$

But these codes may have checks involving $O(n)$ weights...
Most codes of interest are (for reasons we'll discuss in later lectures):

Low-density parity-check (LDPC) code:

- each check contains $\le \Delta_c$ bits ($O(1)$ number (e.g. 4))

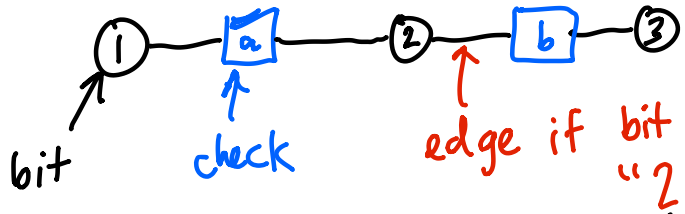
- each bit involved in $\leq \underbrace{\Delta_B}_{O(1)}$ checks

Thm: LDPC codes exist w/ $\boxed{\lim_{n\to\infty} \frac{k}{n}, \frac{d}{n} > 0}$ ← Prove it much later...

"asymptotically good"

A useful way to depict a classical linear code is:

Tanner graph:

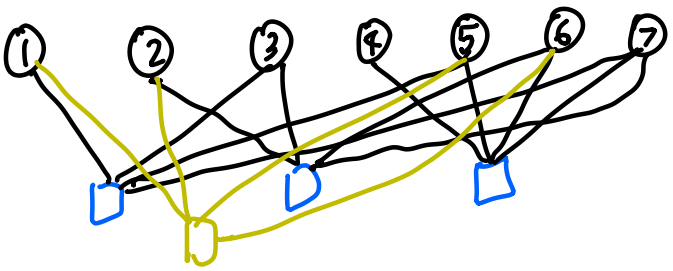


What code is this? Repetition code!

$$H = \begin{array}{c} \\ a \\ b \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \end{array} \quad \begin{array}{l} \text{bit} \\ \\ \text{check} \end{array}$$

[7,4,3] Hamming code:

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



This choice of H is not unique.

Add redundant check?
(Linearly-dependent row of H):

$$H \to \left(\begin{array}{c} H \\ \hline 1\ 1\ 0\ 0\ 1\ 1\ 0 \end{array} \right)$$

add first 2 rows

↳ k unchanged since $\text{rank}(H)$ unchanged

$\Rightarrow$ codespace $\ker(H)$ same!

but energetics (H_0) changed. can be useful

↳ as in the 1d vs 2d Ising model

Canonical form of H: use
- row-reduction (adding rows)
- remove redundant rows
- permute columns (relabel bits)

$$H_{can} = \left(\underbrace{I_{(n-k)\times(n-k)}}_{\text{identity}} \middle| \underbrace{A_{(n-k)\times k}}_{\text{arbitrary}} \right)$$

← This generally destroys LDPC property.

Prop: canonical logicals! $G_{can} = \left((A^T)_{k\times(n-k)} \middle| I_{k\times k} \right)$

Indeed: $H_{can} G_{can}^T = \left(I_{n-k} \middle| A \right) \begin{pmatrix} A \\ \hline I_k \end{pmatrix} = \underbrace{A + A = 0}_{\text{since } \mathbb{F}_2}$

And: $s^T H_{can} = 0 \Rightarrow s = 0$
$G_{can}^T v = 0 \Rightarrow v = 0$
} due to I blocks!

Both $\ker(H)$ and $\text{im}(G^T)$ have dimension k so:

$\Rightarrow \ker(H) = \text{im}(G^T)$.