Linear codes & LDPC codes

Last week: RS codes
* good distance properties (# flips between codewords, see HW!)
* decode by closest codeword in Hamming distance
* algebraic structure → efficient decoding

However: NOT optimal rate ← in the sense of Shannon's theorem.
See MacKay ch. 13, bounded distance decoders never achieve capacity!

Random codes achieve capacity, but
① inefficient encoding (2^K codewords in memory)
② inefficient decoding (typical set decoder means you loop over all 2^K codewords as you saw in HW)

Today: a practical approach to solve these problems!

Linear codes

$$s \longrightarrow x \xrightarrow{\text{BSC}} y \longrightarrow \hat{s}$$

$s \in \{0,1\}^K$, $x \in \{0,1\}^N$, $y \in \{0,1\}^N$, $\hat{s} \in \{0,1\}^K$

encoding $x^N = G^T s^K$ Today: all arithmetic modulo 2

"generator matrix" K×N e.g. $G = \begin{pmatrix} 1000101 \\ 0100110 \\ 0010111 \\ 0001011 \end{pmatrix}$ Hamming code

Find H ← "parity check matrix" such that
* rows of H are lin. independent (so rows of H span the kernel of G)
* $HG^T = 0$
* H is $M \times N$ matrix, $M = N-K$

then x^N is a codeword $\iff Hx^N = 0$

receive $y^N = G^T s^K + n^N$ ← "noise vector" independent of the input; if we know n^N, can reconstruct x^N as $y^N - n^N$

decode $z^K = Hy^N$, $Hn^N = z^K$ ← "error syndrome" ⇝

> **Optimal decoding**
> given z^K, find most likely n^N such that $z^K = Hn^N$

(RS) codes are linear!

Are linear codes good enough to approach capacity? YES:

Thm (informal): Random parity check ~~codes~~ matrices give codes at rates arbitrarily close to the capacity $R < 1 - H(\{f, 1-f\})$ for the BSC with error probability going to 0 with increasing block size.

Proof (sketch): Consider uniformly random $M \times N$ parity check matrices H (each entry equally like 0 or 1).

Typical noise $T_{N,\varepsilon} := \{ n^N \text{ s.t. } \left| \frac{1}{N} \log \frac{1}{P(n^N)} - H(\{f, 1-f\}) \right| < \varepsilon \}$

typical set decoder for $\varepsilon > 0$ estimates the noise to be

$$\begin{cases} n^\perp \text{ if } \exists \text{ unique } n^N \in T_{N,\varepsilon} \text{ s.t. } H n^N = z^k \\ \perp \text{ otherwise} \end{cases}$$

$P_{error} = P^{(1)} + P_H^{(2)}$

$P^{(1)}$ = prob. real noise **not** typical $\xrightarrow[N\to\infty]{} 0$ (as seen before)

$P_H^{(2)}$ = prob. real noise is typical, but $\exists$ multiple typical $n^N, \tilde{n}^N$ s.t. $H n^N = H \tilde{n}^N = z^k$.

$$P_H^{(2)} \le \sum_{n^N \in T_{N,\varepsilon}} P(n^N) \left[\#\text{ of typical sequences } \tilde{n}^N \neq n^N \text{ with } H n^N = H \tilde{n}^N \right]$$

$$\le \sum_{n^N \in T_{N,\varepsilon}} P(n^N) \sum_{\substack{\tilde{n}^N \in T_{N,\varepsilon} \\ \tilde{n}^N \neq n^N}} \mathbb{1}\left[H(n^N - \tilde{n}^N) = 0 \right]$$

$\mathbb{1}$ = indicator function, 1 if $H(n^N - \tilde{n}^N) = 0$, 0 otherwise

Average over all H

$$\overline{P^{(2)}} = \mathbb{E}_H P_H^{(2)} = \sum_H P(H) P_H^{(2)}$$

$$\le \sum_{n^N \in T_{N,\varepsilon}} P(n^N) \sum_{\substack{\tilde{n}^N \in T_{N,\varepsilon} \\ \tilde{n}^N \neq n^N}} \underbrace{\sum_H P(H) \mathbb{1}\left[H(n^N - \tilde{n}^N) = 0 \right]}_{= 2^{-M}}$$

$= 2^{-M}$ ← for each row, exactly half of all possible rows are such that $n^N - \tilde{n}^N$ satisfies this check. So, $P(\text{all } M \text{ checks satisfied}) = \left(\frac{1}{2}\right)^M = 2^{-M}$

$$\le \underbrace{\sum_{n^N \in T_{N,\varepsilon}} P(n^N)}_{\le 1} \underbrace{|T_{N,\varepsilon}|}_{\le 2^{N(H(\{f,1-f\})+\varepsilon)}} 2^{-M}$$

$$\le 2^{N(H(\{f,1-f\})+\varepsilon) - M}$$

$$\xrightarrow[N\to\infty]{} 0 \quad \text{if } \frac{M}{N} > H(\{f, 1-f\}) + \varepsilon$$

So with high probability for random H, $P_{error} \to 0$

Encoding: efficient for linear codes! → only need to store $M \times N$ matrix
Decoding: still (NP-) hard

Solution: Low Density Parity Check (LDPC) codes (or "Gallager codes")

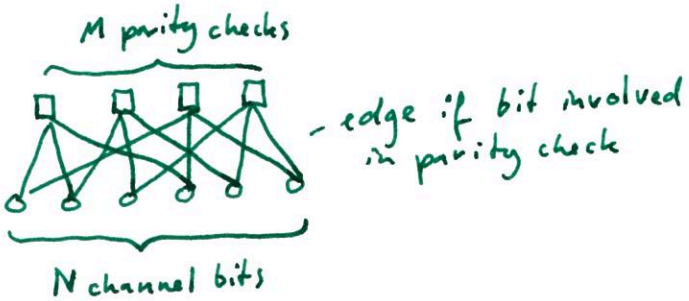
uniformly random H ⇝ sparse random H
↳ e.g. each column has exactly t ones
" row " " w ones

(so each bit is involved in t parity checks, and each parity check involves w bits)

Proof of random coding theorem still works (but more annoying)
and we can approximately solve decoding problem efficiently by message passing!

These codes are state of the art: will be used in 5G!
(and are already used in many communication protocols)

Tanner graph associated to H:



Recall: receive y^N, $z^K = Hy^N$, find most likely n^N s.t. $z^K = Hn^N$

$$P(n^N, z^K) = P(n^N)\,\mathbb{1}[z^K = Hn^N]$$
$$= \prod_{i=1}^{N} P(n_i) \prod_{j=1}^{M} \mathbb{1}\left[z_j = \sum_{k \in N(j)} n_k\right]$$

noise is IID; expand over rows of H; ↳ $N(j)$ denotes neighbours of j in Tanner graph

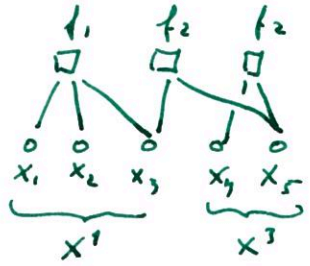
Try to compute $p_i = P(n_i = 1 \mid z^K, H)$ (bitwise decoding problem!)

and guess $n_i = \begin{cases} 1 & p_i > \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$

General marginal problem: (also relevant for other inference tasks)

given $P^*(x^N) = \prod_{j=1}^{M} f_j(x^j)$ $\longrightarrow$ factor graph

notation: x^j denotes subset of variables on which f_j depends.
x^N denotes all variables



Goal: compute marginals

$$P_i^*(x_i) = \sum_{\{x_j\}, i \neq j} P^*(x^N)$$

and normalized marginals

$$P_i(x_i) = \frac{P_i^*(x_i)}{\sum_{x_i} P_i^*(x_i)}$$

Idea: do sums as local as possible

$$P_1^*(x_1) = \sum_{x_2, x_3, x_4, x_5} f_1(x_1, x_2, x_3) f_2(x_3, x_5) f_3(x_4, x_5)$$

$$= \sum_{x_2, x_3} f_1(x_1, x_2, x_3) \sum_{x_5} f_2(x_3, x_5) \sum_{x_4} f_3(x_4, x_5)$$

and 'pass on' local computations to neighbours in factor graph

Message passing algorithm to compute P_i^* : messages $r_{j \to i}$ factors → nodes
$q_{i \to j}$ nodes → factors

Sum-product — input: $\{f_j\}$

* $q_{i \to j}(x_i) \leftarrow 1$
* While not converged: (converged meaning that update does not change any $r_{j \to i}$ or $q_{i \to j}$)

  For edges (i,j) in factor graph:

  → downward pass

  $$r_{j \to i}(x_i) \leftarrow \sum_{x^j \setminus i} f_j(x^j) \prod_{k \in N(j) \setminus i} q_{k \to j}(x_k)$$

  ($x^j \setminus i$: all x_k involved in f_j except x_i)

  → $N(j)$ denotes neighbours of j in factor graph

  $$q_{i \to j}(x_i) \leftarrow \prod_{k \in N(i) \setminus j} r_{k \to i}(x_i)$$

  → upward pass

* Return $P_i^*(x_i) \leftarrow \prod_{j \in N(i)} r_{j \to i}(x_i)$

Converges to correct answer if the factor graph is a tree!

Note: can keep track of normalization along the way
→ just normalize all $q_{i \to j}$

For the LDPC code decoder we pretend that the Tanner graph has no cycles. Notice:

* Sparse H gives graph with relatively few (and long) cycles
* We don't care about the actual $p_i = P_i(n_i | z^k, H)$, just about n^N!

Works very well in practice!

To get the algorithm:

- $f_j = \mathbb{1}\left[z_j = \sum_{k \in N(j)} n_k\right]$
- additional set of factors (leaves, not in Tanner graph)
  $f_i(n_i) = P(n_i) = \begin{cases} f & n_i = 1 \\ 1-f & n_i = 0 \end{cases}$
- normalize $q_{i \to j}$ along the way
- do a finite number of iterations, then stop
  (otherwise we have no halting guarantee)

LDPC decoder — input: H, z^k, max iterations

* $q_{i \to j}(n_i) \leftarrow P(n_i)$
* For max iterations:
  For edges (i,j) in Tanner graph:

$$r_{j \to i}(n_i) \leftarrow \sum_{n_j : n_i} \mathbb{1}\left[z_j = \sum_{k \in N(j)} n_k\right] \prod_{k \in N(j) \setminus i} q_{k \to j}(n_k)$$

$$q_{i \to j}(n_i) \leftarrow \alpha_{ij} P(n_i) \prod_{k \in N(i) \setminus j} r_{k \to i}(n_i)$$

α_{ij}: normalizes $q_{i \to j}$ s.t. $q_{i \to j}(0) + q_{i \to j}(1) = 1$

$P(n_i)$: this is a factor from a leaf, could also have made additional update for this, think for yourself why this makes sense!

$$p_i(n_i) \leftarrow \alpha_i P(n_i) \prod_{j \in N(i)} r_{j \to i}(n_i)$$

α_i: normalizes p_i s.t. $p_i(0) + p_i(1) = 1$

n^N s.t. $n_i = \begin{cases} 1 & p_i(1) > \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$

If $Hn^N = z^k$:
Return n^N

* Return $\perp$