General framework: (n,k) linear block codes (Same as Hamming.)

k = # source bits, 2^k codewords.
n = codeword length
$R = k/n$ = rate.

- parity-check matrix $H = [F \ \ I_m]$ $(m \times n)$, $m = n-k$ = # parity checks
- generator matrix $G = [I_k \ \ F^T]$ $(k \times n)$
- codebook $\mathcal{C} = \{t \in \mathbb{B}^n : Ht \equiv 0\} \Rightarrow \mathcal{C} = \{G^T s : s \in \mathbb{B}^k\}$. (proof same as before in Hamming case)
- encoding: $t \equiv G^T s$
- transmission: $r \equiv t + u$ where $u = \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix}$, $u_1, \ldots, u_n \sim \text{Bernoulli}(\alpha)$ iid (Binary Symmetric Channel.)
- decoding: (Optimal:) decode to nearest codeword ~~decode to most probable codeword~~
  i.e. decode to most probable source msg $\hat{s}$, i.e. $\hat{s} = \underset{s' \in \mathbb{B}^k}{\arg\max}\ p(s'|r)$.

More precisely, decode to $\hat{s} \in \mathbb{B}^k$ s.t. ~~$G^T\hat{s} = r - \hat{u}$~~ $r \equiv G^T \hat{s} + \hat{u}$ where $\boxed{\hat{u} = \underset{\substack{u' \in \mathbb{B}^n: \\ Hu' \equiv Hr}}{\arg\max}\ p(u')}$.

(This is the essence of the decoding problem)

Note: $\hat{u}$ may not be unique

(Note that $Hr \equiv Ht + Hu \equiv Hu$.)

(The decoding method we derived for Hamming was optimal, and comp. efficient.
For Gallager, we won't be able to do optimal decoding, but will just approximate it.)

Decoding problem simplifies to the following: $\boxed{\text{Given } z \in \mathbb{B}^m, \text{ find } \hat{u} = \underset{\substack{u \in \mathbb{B}^n: \\ Hu \equiv z}}{\arg\max}\ p(u).}$
(Can ignore s, t, r.)

Gallager Codes

(aka LDPC codes)

(Discovered by Robert Gallager, published in his MIT PhD thesis and in journals around 1960. Gallager proved initially promising results, and demonstrated good empirical performance, but his codes didn't catch on, ~~Gallager~~ even though Gallager went on to become a leader in the field of info. theory. Perhaps due to the approximate, heuristic nature of the decoding alg., Gallager's codes ~~were~~ were quickly forgotten about as ~~the~~ people explored its competitors: algebraic ~~BCH~~ codes, (BCH, Reed-Solomon), and convolutional codes. In 1993, Turbocodes were invented by a group of French engineers, and ~~perhaps due to this to~~ they achieved much better perf. (close to Sh. limit) than other codes. In 1995-1996, David MacKay and Radford Neal (independently) rediscovered Gallager's codes, ~~which are extra~~ and with (more) modern computing power, were able to demonstrate their extraordinary performance. (Perf. is similar to Turbo codes but Turbocodes are ~~very~~ rather complicated, while Gallager codes are exceedingly simple.)

(# parity bits)

Parameters: Let $c \geq 3$ be an odd integer. Let $d > c$ be an integer. Let $m \in \mathbb{Z}$ s.t. $\frac{md}{c} \in \mathbb{Z}$.

Constants: Define $n = md/c$ (codeword length), $k = n - m$ (# source bits), $R = \frac{k}{n}$ (rate).

(Note: $R = \frac{k}{n} = \frac{n-m}{n} = 1 - \frac{m}{n} = 1 - c/d$. So c, d determine the rate.)

Parity check matrix Define $\mathcal{B}_{m,n}(c,d) = \{ A \in \mathbb{B}^{m \times n} : \sum_{i=1}^{m} a_{ij} = c \;\forall j \text{ and } \sum_{j=1}^{n} a_{ij} = d \;\forall i \}$.

m by n binary matrices (i.e. all col. sums are c, all row sums are d.)

(Note: $\exists$ efficient approx. sampling algs)

Draw a matrix H uniformly from $\mathcal{B}_{m,n}(c,d)$. (H is $m \times n$, with $n > m$)

Modify H by permuting its columns to obtain $H = [H_{(1)} \;\; H_{(2)}]$ s.t. $H_{(2)}$ is invertible mod 2.

$H_{(1)}$: $m \times k$, $H_{(2)}$: $m \times m$

(i.e. $\exists\, A \in \mathbb{B}^{m \times m}$ s.t. $AH_{(2)} \equiv I_m$ and $H_{(2)}A \equiv I_m$) (Note: with high probability, this is possible. If not, resample H.)

$$\Rightarrow \quad AH = [AH_{(1)} \;\; AH_{(2)}] \equiv [F \;\; I_m] \quad \text{where } \cancel{F \equiv AH_{(1)}}\; F \in \mathbb{B}^{m \times k} \text{ s.t. } F \equiv AH_{(1)}.$$

(Later maybe we'll see how to do these steps in comp. efficient ways.)

~~Generator~~ ~~Note: there are~~. (Note: In practice, some minor variations on this scheme.)

Generator matrix

$$G = [I_k \;\; F^T].$$

Encoding $t \equiv G^T s$

Trans: $r \equiv t + u$, $u_i \sim \text{Bernoulli}(\alpha)$ iid

Decoding Problem

Let $U_1, \dots, U_n \sim \text{Bernoulli}(\alpha)$ iid, $\alpha \in (0, 1/2)$. Let $Z \equiv HU$, $(Z \in \mathbb{B}^m)$.

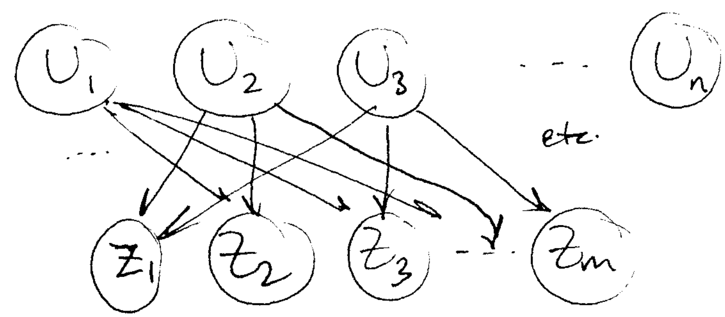
$\Rightarrow$ $p(u,z) = p(u)\,p(z|u) = \alpha^{w(u)}(1-\alpha)^{n-w(u)}\, I(Hu \equiv z)$ where $w(u) = \#\{i : u_i = 1\}$ = "weight of u"

Also, $p(u,z) \underset{u}{\propto} p(u|z)$.

Optimal decoding: ~~Goal~~: Find $u^* = \underset{u:\,Hu\equiv z}{\operatorname{argmax}}\, p(u) = \underset{u \in \mathbb{B}^m}{\operatorname{argmax}}\, p(u)\, I(Hu \equiv z) = \underset{u \in \mathbb{B}^m}{\operatorname{argmax}}\, p(u|z)$.

$= \underset{u:\,Hu\equiv z}{\operatorname{argmin}}\, w(u)$.

$$H: \begin{array}{c|cccc|} & u_1 & u_2 & u_3 & \cdots\ u_n \\ \hline z_1 & & & 1 & \\ z_2 & & & 0 & \\ z_3 & & & 1 & \\ \vdots & & & \vdots & \\ z_m & & & 1 & \\ \hline \end{array}$$



each u_j has c nbrs

(H corresponds to a bipartite graph)

each z_i has d nbrs

- Optimal is too hard. Use approximation.

In general, NP-complete. (Not sure if still NP-complete for sparse H.)

(Approx #1): Assume $p(U_i = 1 | Z = z) \approx I(u_i^* = 1)$ $\forall i$.

optimal

Justification: If α is small and $w(u^*) \ll w(u)$ $\forall u \neq u^*$ s.t. $Hu \equiv z$, then $p(u|z) \approx I(u = u^*)$,

so $p(u_i|z) \approx I(u_i = u_i^*)$ $\forall i$.

(New goal: Try to approximate $\cancel{p(U_i=1|Z=z)}$ $p(u_i|z)$ $\forall i$.)

Justification of algorithm

(Iterative fixed-point algorithms. We derive a set of eqns that the soln should satisfy (approximately), and iterate to try to find a fixed point (i.e. a soln of the eqns.)

Fixed point algs are extremely powerful (and ancient) — can be very fast & accurate even on high-dim problems. However, deriving them is something of an art, and proving perf. guarantees can be difficult.)

Example (Babylonian method for square roots:) To find $x=\sqrt{a}$, ~~iterate~~ set $x_0=1$ and iterate:

$$x_{i+1} = \left(\frac{a}{x_i} + x_i\right)/2 \quad \text{until convergence.}$$

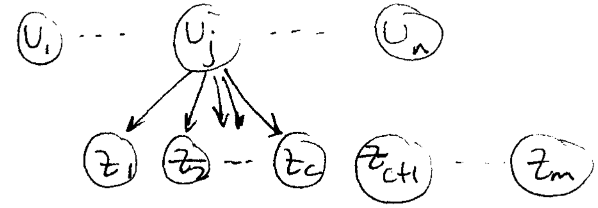
Notation:

- $N_i = \{j : H_{ij}=1\}$ = neighbors of z_i
- $M_j = \{i : H_{ij}=1\}$ = " " " " u_j
- Given $S \subset \{1,\ldots,n\}$, denote $u_S = (u_{s_1}, u_{s_2}, \ldots, u_{s_\ell}) = (u_s : s\in S)$, where $S=\{s_1,\ldots,s_\ell\}$.
- $N_i \setminus \ell = \{j : H_{ij}=1,\ j\neq \ell\}$.

and $S^c = \{\ell \in \{1,\ldots,n\} : \ell \notin S\}$.

~~$S^c = \{\ell \in \{1,\ldots,n$~~

Suppose $M_j = \{1, 2, \dots, c\}$.



$$p(u_j \mid z_2, \dots, z_m) \underset{u_j}{\propto} \underbrace{p(z_2, \dots, z_m \mid u_j)}_{=} \overbrace{p(u_j)}^{\alpha^{u_j}(1-\alpha)^{1-u_j} \;=}$$

$$= \underbrace{p(z_2, \dots, z_c \mid z_{c+1}, \dots, z_m, u_j)}_{=} \overbrace{p(z_{c+1}, \dots, z_m \mid u_j)}^{p(z_{c+1}, \dots, z_m) \,=\, \text{const wrt } u_j \;=}$$

$(z_{c+1}, \dots, z_m) \perp\!\!\!\perp U_j$ by d-separation since any path goes through a head-head junction at z_i for some $i \le c$.

$=$ ← chain rule

$\cancel{p(z_2 \mid z_{c+1:m}, u_j)\, p(z_3 \mid z}$

$$p(z_c \mid z_{c+1:m}, u_j)\, p(z_{c-1} \mid z_{c:m}, u_j)\, p(z_{c-2} \mid z_{c-1:m}, u_j) \cdots p(z_2 \mid z_{3:m}, u_j) \approx \prod_{\ell=2}^{c} p(z_\ell \mid u_j, z_i \ \forall i \ne \ell)$$

Approx #2 Intuitively reasonable since the influence of the nbrs of u_j on z_ℓ is mediated predominantly through u_j.

$$\Rightarrow \quad p(u_j \mid z_2, \dots, z_m) \underset{\text{approx}}{\propto} \alpha^{u_j}(1-\alpha)^{1-u_j} \prod_{\ell=2}^{c} p(z_\ell \mid u_j, z_i \ \forall i \ne \ell).$$

More generally, if $i \in M_j$:

$\cancel{p(u_j \mid z_1, \dots, z_{i-1}, z_{i+1}, \dots, z_m) \underset{\text{approx}}{\propto} \alpha^{u_j}(1-\alpha)^{1-u_j} \prod_{\ell \in M_j \setminus i} p(z_\ell \mid}$

$$p(u_j \mid z_a \ \forall a \ne i) \underset{\text{approx}}{\propto} \alpha^{u_j}(1-\alpha)^{1-u_j} \prod_{\ell \in M_j \setminus i} p(z_\ell \mid u_j, z_a \ \forall a \ne \ell).$$

~~For $i \in M_j$:~~

This motivates the formula: For $i \in M_j$:

$$\boxed{q_{ij}(u_j) \propto \alpha^{u_j}(1-\alpha)^{1-u_j} \prod_{\ell \in M_j \setminus i} r_{\ell j}(u_j)}$$

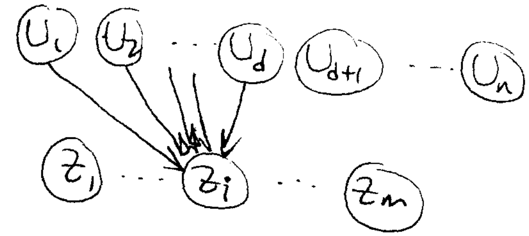
(Vertical Step)

~~(vertical step)~~

~~(Horizontal Step)~~

with $q_{ij}(0) + q_{ij}(1) = 1$,

Now, suppose $i \in M_1$ and $N_i = \{1, \dots, d\}$.



$$p(z_i \mid u_1, z_a \; \forall a \neq i) =$$

$$\sum_{u_2 \in \{0,1\}} \sum_{u_3 \in \{0,1\}} \cdots \sum_{u_d \in \{0,1\}} \underbrace{p(z_i \mid u_1, \dots, u_d, z_a \; \forall a \neq i)}_{= \; p(z_i \mid u_1, \dots, u_d) = \mathbb{I}(z_i \equiv \sum_{l=1}^{d} u_l)} \underbrace{p(u_2, \dots, u_d \mid u_1, z_a \; \forall a \neq i)}_{\approx \; \prod_{l=2}^{d} p(u_l \mid z_a \; \forall a \neq i)}$$

Approx #3 Reasonable since any dependence among $u_1, \dots, u_d$ (given $z_a \; \forall a \neq i$) must be mediated through some path not through z_i.

$$\Rightarrow p(z_i \mid u_1, z_a \; \forall a \neq i) \approx \sum_{u_{2:d}} \mathbb{I}\Big(z_i \equiv \sum_{l=1}^{d} u_l\Big) \prod_{l=2}^{d} p(u_l \mid z_a \; \forall a \neq i)$$

More generally, if $i \in M_j$,

$$p(z_i \mid u_j, z_a \; \forall a \neq i) \approx \sum_{\substack{u_l \in \{0,1\} \\ \text{for } l \in N_i \setminus j}} \mathbb{I}\Big(z_i \equiv \sum_{l \in N_i} u_l\Big) \prod_{l \in N_i \setminus j} p(u_l \mid z_a \; \forall a \neq i)$$

(This motivates the formula:) For $j \in N_i$ ~~$i \in M_j$~~:

$$\boxed{r_{ij}(u_j) = \sum_{\substack{u_l \in \{0,1\} \\ \text{for } l \in N_i \setminus j}} \mathbb{I}\Big(z_i \equiv \sum_{l \in N_i} u_l\Big) \prod_{l \in N_i \setminus j} q_{il}(u_l)}$$

(Horizontal Step)
(~~Vertical step~~)
~~Horiz~~

More efficient computation of $r_{ij}(u_j)$

Lemma: For $i = 1, \dots, n$, let $X_i = \begin{cases} 0 & \text{wp } p_i(0) \\ 1 & \text{wp } p_i(1) \end{cases}$ (indep). Let $S = \sum_{i=1}^{n} X_i$.

ⓐ $\mathbb{P}(S \text{ even}) + \mathbb{P}(S \text{ odd}) = \prod_{i=1}^{n} (p_i(0) + p_i(1)) = 1$

and ⓑ $\mathbb{P}(S \text{ even}) - \mathbb{P}(S \text{ odd}) = \prod_{i=1}^{n} (p_i(0) - p_i(1))$.

Pf: ⓐ is clear, since $p_i(0) + p_i(1) = 1$.

ⓑ Multiplying out the product, terms w/ odd # of ones have sign -1,

— — even — — — — — 1. $\square$

Corollary: $r_{ij}(0) + r_{ij}(1) = 1$ and $r_{ij}(0) - r_{ij}(1) = (-1)^{z_i} \prod_{\ell \in N_i \setminus j} (q_{i\ell}(0) - q_{i\ell}(1))$.

Pf: (Exercise).

Rk: To compute $r_{ij}(0)$ and $r_{ij}(1)$,

① Compute $\delta_{ij} := (-1)^{z_i} \prod_{\ell \in N_i \setminus j} (q_{i\ell}(0) - q_{i\ell}(1))$

② Set
$$r_{ij}(0) := \tfrac{1}{2}(1 + \delta_{ij}),$$
$$r_{ij}(1) := \tfrac{1}{2}(1 - \delta_{ij}).$$