

LDPC Notes

520.460

November 29, 2007

1 Introduction

LDPC codes are block codes. Some refs:

- Gallager, *Trans. on Inform Thy*, pp 21-28, 1962, “Low-Density Parity Check Codes.”
- PhD dissertation published under the same title as an MIT Press monograph, 1963, now found here: <http://web.mit.edu/gallager/www/pages/ldpc.pdf>.

Gallager’s definition (now called “regular LDPC codes”):

Definition 1 *The block code $C_{LDPC}(n, s, v)$ (text notation) is a regular LDPC code with block length $= n$ and*

- v ONEs per row of \mathbf{H} -matrix;
- s ONEs per column of \mathbf{H} -matrix;
- no more than 1 ONE in common between any two rows;
- both s and v are “small” in comparison to n .

□

Lemma 1 *If the rows of \mathbf{H} are linearly independent, then the code rate is $(v - s)/v$.*

Proof: The number of ONEs in \mathbf{H} is $sn = v(n - k)$ where k = the block code dimension. Then,

$$\frac{n - k}{n} = \frac{s}{v} = 1 - R,$$

from which

$$R = 1 - \frac{s}{v} = \frac{v - s}{v}$$

□

Clearly, $v > s$ for a useful code.

Gallager's Construction of Regular LDPC Codes

(with embedded example). Codes are constructed by building the \mathbf{H} matrix.

1. Select integer $\kappa > 1$ and integers v and s as previously defined; e.g., $\kappa = 3$, $v = 3$, $s = 2$.
2. Construct a $\kappa s \times \kappa v = 6 \times 9$ matrix \mathbf{H} from $s(= 2)$ submatrices with dims. $\kappa \times \kappa v = 3 \times 9$, as follows:
 - (a) each row of the first submatrix \mathbf{H}_1 has $r = e$ ONES;
 - (b) each column has 1 ONE.
3. Construct submatrices $\mathbf{H}_2, \mathbf{H}_3, \dots, \mathbf{H}_s$ similarly. Note that these are merely column permutations of H_1 . (The boundary condx. are met.)

Example:

$$\mathbf{H}_1 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Select the column permutation to be $\Pi = [9 \ 2 \ 7 \ 1 \ 6 \ 3 \ 5 \ 4 \ 8]$. Then

$$\mathbf{H}_2 = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

So, far, the first two properties of LDPC codes are satisfied merely by construction, and

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

But what about the requirement, "The number of ones in common between any two columns ≤ 1 ?"

It depends upon the **permutation**.

In this case, we were **LUCKY**. #3 is satisfied.

Otherwise, **computer search** is required for regular LDPC code.

See also, Gallager's construction for the $(n, k) = (15, 5)$ LDPC code.

Finally, it is required that s and v be small compared with n . In the example,

$$\begin{aligned} \frac{s}{n} &= \frac{2}{9} = 0.22 \\ \frac{v}{n} &= \frac{3}{9} = 0.33 \end{aligned}$$

Is this “small” enough?

Sparse

The larger κ (in Gallager’s construction), the more sparse is \mathbf{H} because the fraction of ONEs in \mathbf{H} as constructed is:

$$\frac{\kappa S V}{\kappa \cdot \kappa S V} = \frac{1}{\kappa}$$

(because $n = \kappa V$ in the construction).

Generating Code Words

1. Find matrix \mathbf{H} as above or by other means, e.g., computer search.
2. Using elementary row operations, convert \mathbf{H} to

$$\mathbf{H}' = [I_{n-k} P^T]$$

3. Write

$$\mathbf{G} = [P I_k]$$

So, \mathbf{G} is used for encoding: $\mathbf{c} = \mathbf{m} \cdot \mathbf{G}$.

2 The Sum-Product Algorithm

2.1 Tanner Graphs

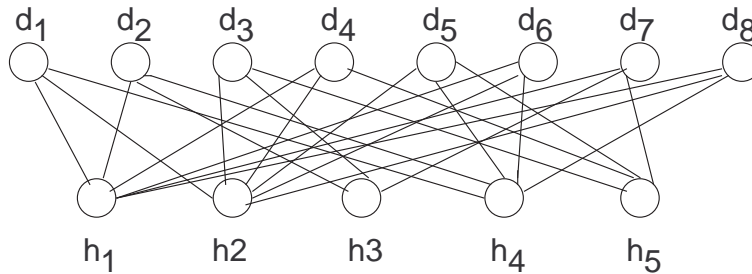
- bipartite
- defined by **parity check equations** described by \mathbf{H} .
- **Symbol nodes:** $\{d_j\}$, $j = 1, 2, \dots, n$.
- **Parity check nodes:** $\{h_i\}$, $i = 1, 2, \dots, n - k$.

Write $\mathbf{H} = [H_{ij}]$. If $H_{ij} = 1$, then symbol d_j is connected to parity node h_i , $j = 1, 2, \dots, n$, $i = 1, 2, \dots, n - k$.

Example: An irregular LDPC code is given by:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

The corresponding Tanner graph is:



Note:

1. The weight of each column of \mathbf{H} is 3.
2. Each d_j terminates 3 edges from parity check nodes.
3. Or, every symbol is checked by three parity positions.
 - h_1 checks 6 symbols ($d_1 + d_2 + d_4 + d_6 + d_7 + d_8 = 0$).
 - h_2 checks 6 symbols.
 - h_3 checks 3 symbols.
 - h_4 checks 5 symbols.
 - h_5 checks 4 symbols.

We define by example:

- h_1, h_2, h_4 are **children** of d_1 .
- $d_1, d_2, d_4, d_6, d_7, d_8$ are **parents** of h_1 .

2.2 Sum-Product Algorithm

2.2.1 General Introduction

1. In the **Sum-Product Algorithm**,

- each symbol node d_j sends Q_{ij}^x (**prior** information) to its child h_i , where Q_{ij}^x is an estimate of the probability that node h_i is in state x , based on information from all **other** children of d_j ;
- Each parity check node h_i sends R_{ij}^x (**extrinsic** information) to its parent d_j , where R_{ij}^x is an estimate of the probability that the parity check equation related to h_i is satisfied if the parent d_j is in state x .

2. So, in the example:

- d_1 sends Q_{12}^x to h_1 , based upon information provided by h_3 and h_4 .
- h_1 sends R_{12}^x to d_2 , based upon information from d_1, d_4, d_6, d_7, d_8 .

(Please note to reverse subscripts $43 \rightarrow 34$ in Figure 8.1 of Castinera and Farrell.)

3. Information exchange proceeds iteratively between the two types of nodes and **halts** when the decoded vector $\mathbf{d} = (d_1, \dots, d_n)$ has **zero syndrome**.
4. After a **predetermined** number of iterations, if the syndrome does not become zero, the algorithm is **stopped**, and \mathbf{d} will not be a code word.
5. **Nevertheless**, the individual symbols are optimally decoded in the MAP sense.
6. Iterative LDPC decoding converges to true message information when the bipartite graph has a **tree structure** (no cycles).

2.2.2 Sum-Product Algorithm Details

Let

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \vdots \\ \mathbf{h}_r \end{bmatrix}$$

where $r = n - k$, the code redundancy, and where

$$\mathbf{h}_j \triangleq (h_{j0}, h_{j1}, \dots, h_{j,n-1})$$

Let $B(\mathbf{h}_j)$ be the “index set” for \mathbf{h}_j (or the “support” of \mathbf{h}_j , the indices of the columns where \mathbf{h}_j is 1:

$$B(\mathbf{h}_j) \triangleq \{\ell : h_{j\ell} = 1, 0 \leq \ell < n\}$$

Computing Priors

- Let $Q_{j\ell}^{x(i)}$ = the probability that, at the i^{th} iteration, code symbol $c_\ell = x$, given the values of the check sums $\{h_i\}$ as computed from the parity vectors in $A_\ell \setminus \mathbf{h}_j$, where
 - A_ℓ = the set of parity vectors that are orthogonal on the ℓ^{th} symbol, and
 - $A_\ell \setminus \mathbf{h}_j$ is A_ℓ excluding \mathbf{h}_j .
- In other words,

$$Q_{j\ell}^{x(i)} = P(c_\ell = x | \{h_i : \mathbf{h}_i \in A_\ell \setminus \mathbf{h}_j\})$$

These **priors** are sent from node $d_{\ell i}$ to parity node h_j ; they are conditioned on all the **other** check nodes that check c_ℓ but are not conditioned on h_j because, on a prior iteration, h_j provided (extrinsic) information to d_ℓ .

Extrinsic Information

- Let $R_{j\ell}^{x(i)}$ = the probability that the j^{th} parity check is satisfied (i.e., $h_j = 0$) given that $c_\ell = x$. This probability is obtained by **jointly** conditioning
 1. the event $c_\ell = x$ and on
 2. the values of c_t for all *other* $t \in B(\mathbf{h}_j)$,

then averaging over the values of c_t . That is,

$$R_{j\ell}^{x(i)} = \sum_{c_t} P(h_j = 0 | (c_\ell = x), c_t) \cdot P(c_t).$$

- The c_t are those for which t enumerates the nonzero columns of \mathbf{h}_j , i.e.,

$$t \in B(\mathbf{h}_j) = \{\ell : h_{j\ell} = 1, 0 \leq \ell < n\}.$$

and

$$P(c_t) = \prod_{t \in B(\mathbf{h}_j) \setminus \ell} Q_{j\ell}^{c_t(i)},$$

the joint prior of all nodes d_ℓ checked by the same h_j .

- So finally,

$$R_{j\ell}^{x(i)} = \sum_{\{c_t : t \in B(\mathbf{h}_j) \setminus \ell\}} P(h_j = 0 | (c_\ell = x), \{c_t : t \in B(\mathbf{h}_j) \setminus \ell\}) \cdot \prod_{t \in B(\mathbf{h}_j) \setminus \ell} Q_{j\ell}^{c_t(i)}$$

Updating the Priors

- Node d_ℓ updates its estimate of the prior probability from the extrinsic information provided by parity check nodes connected to it in the Tanner graph. In simple terms,

$$Q_{j\ell}^{x(i+1)} \propto P(c_\ell = x | \text{checksums at } i^{\text{th}} \text{ iteration})$$

it is proportional to $P(c_\ell = x)$ multiplied by the joint conditional probability that all of the **other** parity checks are satisfied at the i^{th} iteration. In full notation,

$$Q_{j\ell}^{x(i+1)} \propto P(c_\ell = x) \cdot \prod_{\mathbf{h}_t \in A_\ell \setminus h_j} R_{t\ell}^{x(i)}$$

- The constant of proportionality $\alpha_{j\ell}^{(i+1)}$ is chosen so that

$$Q_{j\ell}^{0(i+1)} + Q_{j\ell}^{1(i+1)} = 1$$

- So, first the prior $Q_{j\ell}^{x(i)}$ is computed and sent to parity check node h_j where extrinsic information $R_{j\ell}^{x(i)}$ is computed and sent to data node d_ℓ , where the updated prior, $Q_{j\ell}^{x(i+1)}$ is computed and sent to h_j , etc...

Finally,

- the joint probability that parity checks in A_ℓ are satisfied at iteration $i - 1$ is given by

$$\prod_{\mathbf{h}_j \in A_\ell} R_{j\ell}^{x(i-1)};$$

- values of the parity checks are determined solely by the received word $\mathbf{y} = \mathbf{c} + \mathbf{n}$ because (1) the checks are satisfied for \mathbf{c} , so any checks that fail are determined solely by \mathbf{n} . Therefore, we can write:

$$P^{(i)}(c_\ell = x | \mathbf{y}) = \alpha_\ell^{(i)} P(c_\ell = x) \prod_{\mathbf{h}_j \in A_\ell} R_{j\ell}^{x(i-1)}$$

where the value of $\alpha_\ell^{(i)}$ is chosen to make

$$P^{(i)}(c_\ell = 0 | \mathbf{y}) + P^{(i)}(c_\ell = 1 | \mathbf{y}) = 1.$$

- Now let the decoded estimate at the i^{th} iteration be

$$\mathbf{z}^{(i)} = \left(z_0^{(i)}, z_1^{(i)}, \dots, z_{n-1}^{(i)} \right)$$

from which we get the decision rule:

$$z_\ell^{(i)} = \begin{cases} 1, & P^{(i)}(c_\ell = 1 | \mathbf{y}) > 0.5 \\ 0, & \text{otherwise} \end{cases}$$

If $\mathbf{z}^{(i)} \cdot \mathbf{H}^T = 0$, STOP.

Gathering the foregoing analysis into algorithmic form gives:

STEP 0: Initialize
 $i = 0$
 I_{max} = maximum number of iterations
 For every pair $(j, \ell) : h_{j,\ell} = 1, 1 \leq j \leq r, 0 \leq \ell < n$, set

$$Q_{j,\ell}^{0(0)} = P(c_\ell = 0)$$

$$Q_{j,\ell}^{1(0)} = P(c_\ell = 1)$$

STEP 1: For

$$0 \leq \ell < n$$

$$1 \leq j \leq r$$

each $\mathbf{h}_j \in A_\ell$

compute $R_{j\ell}^{0(i)}$ and $R_{j\ell}^{1(i)}$

GO TO STEP 2

STEP 2: For

$$0 \leq \ell < n$$

$$1 \leq j \leq r$$

each $\mathbf{h}_j \in A_\ell$ compute

$$Q_{j,\ell}^{0(i+1)} \text{ and } Q_{j,\ell}^{1(i+1)}$$

$$P^{(i+1)}(c_\ell = 0|\mathbf{y}) \text{ and } P^{(i+1)}(c_\ell = 1|\mathbf{y})$$

If $\mathbf{z}^{(\ell)} \cdot \mathbf{H}^T = 0$ or $i = I_{max}$, GOTO 3.
 otherwise, GOTO 1.

STEP 3: Output $\mathbf{z}^{(\ell)}$ and STOP