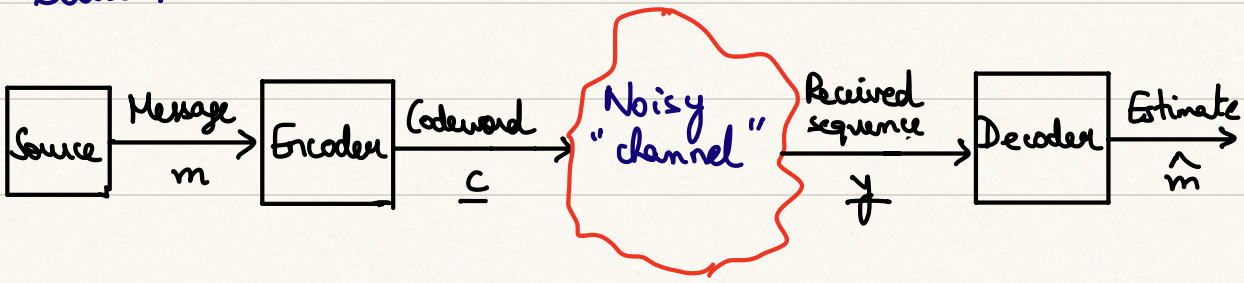
Lecture : Towards modern coding theory : LDPC codes

In the lectures that follow, we shall work towards an exposition of codes that are "good" over the point-to-point communication model we had seen earlier:



We had earlier seen the ML/MAP decoder and used its equivalence to minimum distance decoding (over the BSC) to inform design criteria (good codes = small minimum distance and high rate) for the construction of algebraic codes.

Although some algebraic codes admit efficient implementations of ML decoding, the problem of ML decoding is in general computationally very expensive (naively, search over all 2^{nR} codewords!)

Coding theorists hence turned to other codes that are not necessarily rooted in the algebraic machinery we've seen, but which exhibit provably good performance via <u>efficient</u> decoders (that could be suboptimal compared to ML).

A quick historical perspective [Lecture 1, "Advanced Channel Coding", Henry Pfister, Duke U.]

1945: Hamming works with ECC for computers

1948: Shannon's work "A Mathematical Theory of Communication"

1955: Elias publishes first paper on convolutional codes

1960: Reed-Solomon codes

1963: Gallager introduces LDPC codes and iterative decoding in his PhD thesis

1993: Berrou et al. propose turbo codes that revolutionize coding

1995: MacKay and Neal rediscover LDPC codes

2001: TJ Richardson and R. Urbanke introduce density evolution to optimize LDPC codes

2009: Arıkan introduces polar codes that deterministically achieve the capacity of memoryless, symmetric channels

2017: Kudekar et al. prove that Reed-Muller (RM) codes achieve the capacity of erasure channels

2023: E. Abbe and C. Sandon prove that RM codes achieve capacity on all symmetric channels (see also 2024: G. Reeves and H. Pfister prove that RM codes achieve vanishing bit-error probability on symmetric channels for all rates below capacity)

In what follows, we shall first describe a class of low-density parity-check (LDPC) codes, and describe an iterative decoding algorithm for such codes. We shall then discuss the rationale behind the decoding algorithm, which is rooted in the design of more general message-passing algorithms on probabilistic graphical models.

Fix an $(m = (n-k)) \times n$ binary parity-check matrix H.

Def 1: The Tanner graph of an $m \times n$ binary parity-check matrix H is a bipartite graph with one vertex for each code symbol (variable node) and one vertex for each parity-check (check node / factor node). Formally, the Tanner graph $\mathcal{G} = (\mathcal{V} \cup \mathcal{C}, \mathcal{E})$, with

$$\mathcal{V} = \{1, \dots, n\}, \quad \mathcal{C} = \{1, \dots, m\}, \text{ and}$$

$$\mathcal{E} = \{(i,j) \in \mathcal{C} \times \mathcal{V} : H_{ij} \neq 0\}.$$

We first take a look at a specific LDPC code ensemble.

Def 2 (Regular Code Ensemble): We define $R(n, \ell, r)$ to be the regular LDPC code ensemble, for $n \in r\mathbb{N}$, defined as follows. First, n variable nodes are initialized, each with ℓ "half-edges"; the half-edges are labelled from 1 to $n\ell$. Next, $\frac{n\ell}{r}$ check nodes are initialized, each with r half-edges. Finally, the half-edges are attached via a uniformly random permutation on

nl elements.

If there are multiple edges, collapse them as follows: if an odd number of edges connect a variable node to a check node, replace them with a single edge; else, all (even no. of) edges are removed.

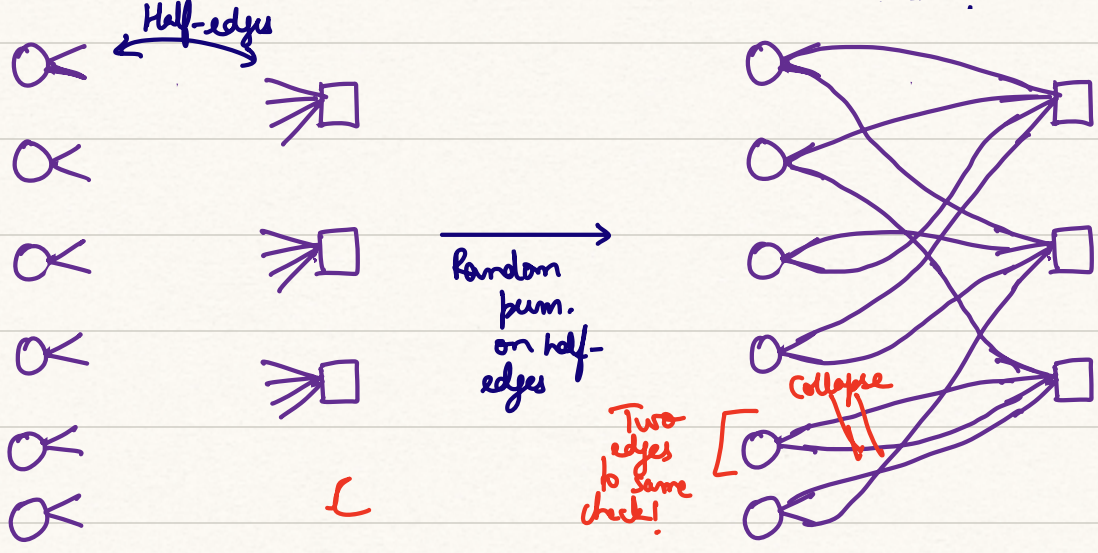
Remarks:

(i) There exists a bijection b/w Tanner graphs and 0-1 matrices, which can be viewed as parity-check matrices H.

(ii) LDPC codes (from a regular ensemble or otherwise) are "sparse", i.e., their parity-check matrix has very few 1s. Note that in the construction above, each row of H induced is designed to have exactly r 1s.

(iii) The parameters (n, l, r) are only design parameters, since the actual node degrees will vary based on edge collapses.

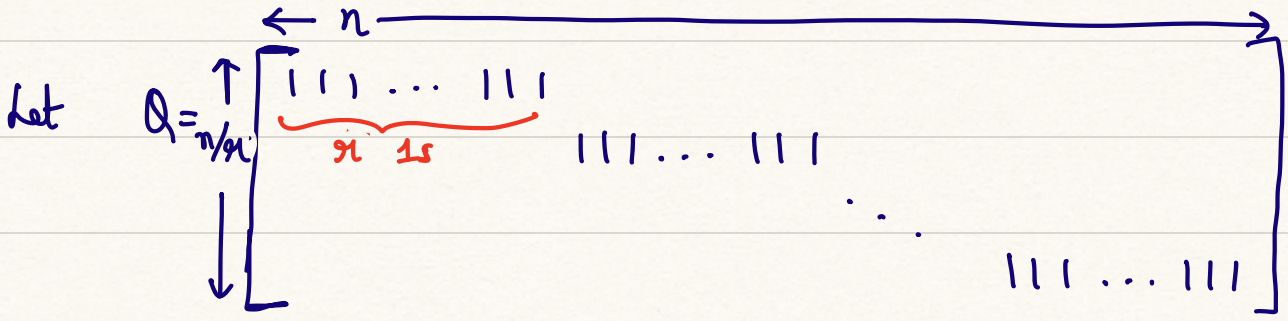
Example: Consider the graph with $n=6$ variable nodes, $m=3$ check nodes and $l=2$, with r s.t. $mr=nl$, i.e., $r=4$.



We now take a look at a specific regular LDPC code ensemble.

Gallager's ensemble

We now describe a specific class of LDPC codes, with a fixed parity-check matrix structure.



We define H as

$$H = \begin{bmatrix} QP_1 \\ QP_2 \\ \vdots \\ QP_l \end{bmatrix},$$

for some fixed integer l, where $P_1, \ldots, P_l$ are uniformly random permutation matrices, chosen independently.

Programming exercise: (in parts)

① Construct a matrix $[H]_{1024 \times 2048}$, as above, for the case when $r = 32$.

[Think about how you would construct the permutation matrices $P_1, \ldots, P_l$]

② Initialize a graph data structure (can be represented via a suitable array / adjacency matrix too) with 2048 variable nodes and 1024 check nodes, corresponding to H.

HW: Consider the parity-check matrix

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

Draw the Tanner graph $\mathcal{G}$ corresponding to H.

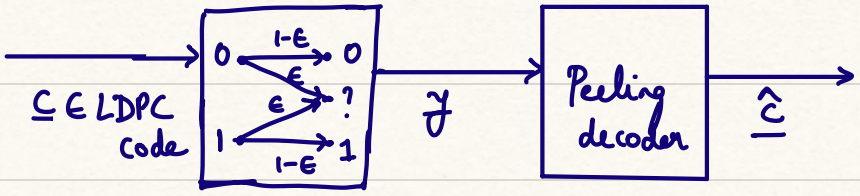
(i) What is the (designed) code rate?

(ii) What is the length of the smallest cycle in $\mathcal{G}$? $\triangleq$ girth

(iii) What is the length of the longest path in $\mathcal{G}$? (Std. be 10) $\triangleq$ diameter

We now briefly take a look at a simple decoding algorithm, the peeling decoder, for LDPC codes over the binary erasure channel (BEC).

Recall that the channel model is:



The decoder was first introduced in [Luby, Mitzenmacher, Shokrollahi, Spielman (2001), "Efficient Erasure Correcting Codes"].

Peeling decoder ($\underline{y}$)

(0) Initialize variables $\hat{c}_1, \dots, \hat{c}_n$ to ? and $y_1, \dots, y_m$ to 0.

(1) For each non-erased code symbol c_j, set $\hat{c}_j = c_j$.

(2) If $\exists$ a degree-1 check node i, set the unique $\hat{c}_j \in N(i)$ to y_i. ($N(i)$: neighbour set)

(3) If $\underline{y}$ contains a variable node $\hat{c}_j \neq ?$:

  (a) For all $i \in N(j)$, update $y_i \leftarrow y_i \oplus \hat{c}_j$

  (b) Set $V \leftarrow V \setminus \{j\}$, $E \leftarrow E \setminus \{(i,j) : i \in N(j)\}$.

  (c) Go to Step (2).

(4) Return $\underline{\hat{c}}$.

We say that decoding is successful if $\underline{\hat{c}} = \underline{c}$, and unsuccessful, otherwise.

Programming exercise: Simulate the performance of the peeling decoder for a uniformly random codeword drawn from the (7,4) Hamming code transmitted over the BEC(ϵ), for varying $\epsilon \in [0.1, 0.9]$.

Use $H = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$ as the parity-check matrix of the code and perform 100 simulations (uniformly random codeword, random erasures) for each value of ϵ.

Plot the probability of error $P_\epsilon \triangleq \frac{\#\text{ unsuccessful decoding trials at } \epsilon}{100}$, vs. ϵ.

Lecture : Peeling decoder (contd.) and an introduction to message-passing decoding

Recall the definition of the peeling decoder, from last time. We now present a bound on its performance, via the idea of using stopping sets.

Def 1 (Stopping set): A stopping set (ss) $\mathcal{S} \subseteq \mathcal{V}$ is a collection of variable nodes whose induced subgraph has no check nodes with degree 1.

By definition, we let the empty set be a ss.

Remark: Let $\text{supp}(\underline{c})$ denote the support of a codeword $\underline{c}$ in the code of interest. In order to satisfy the parity-check equations, all check nodes in $N(\text{supp}(\underline{c}))$ must be connected at least twice to variable nodes in $\text{supp}(\underline{c})$. Hence, $\text{supp}(\underline{c})$, for any codeword $\underline{c}$, is a stopping set.

Stopping sets have the following properties:

Proposition 1: (1) If $\mathcal{S}_1$ and $\mathcal{S}_2$ are ss, then so is $\mathcal{S}_1 \cup \mathcal{S}_2$.

(2) Each subset $\mathcal{W} \subseteq \mathcal{V}$ has a unique maximal ss.

(3) If $\mathcal{W} \subseteq \mathcal{V}$ is the collection of erased variable nodes, then the peeling decoder recovers the value of all variable nodes except those in the unique maximal ss. in $\mathcal{W}$.

Proof:

(1) Note that if $i \in N(\mathcal{A}_1 \cup \mathcal{A}_2)$, then $i \in N(\mathcal{A}_1)$ or $i \in N(\mathcal{A}_2)$, implying that i is connected at least twice to nodes in $\mathcal{A}_1 \cup \mathcal{A}_2$.

(2) Define $\mathcal{Z}$ to be the union of all ss contained in $\mathcal{W}$; by (1), it follows that $\mathcal{Z}$ is the unique maximal ss in $\mathcal{W}$ as any ss in $\mathcal{W}$ is contained in $\mathcal{Z}$.

(3) Note that since $\mathcal{Z} = \mathcal{Z}(\mathcal{W})$ is a ss, the variable nodes in $\mathcal{W}$ will not be recovered by the peeling decoder. Further, any $\mathcal{T} \subseteq \mathcal{W} \setminus \mathcal{Z}$ is such that $\mathcal{T} \cup \mathcal{Z}$ is not a s.s. and will be recovered in part by one step of the peeling decoder. ▨

Def 2: A ss is minimal if the only ss it contains besides itself is the empty set.

For a given code $\mathcal{C}$ with a fixed parity-check matrix H (that defines a Tanner graph), let $A_{ss}(h)$ denote the number of minimal stopping sets of cardinality h. Now, over a BEC, note that the peeling decoder is unsuccessful at decoding from the erasures introduced (at some set $\eta \subseteq \mathcal{V}$), only if η contains some minimal ss [Why?].

Hence, it follows that

$$\Pr[\hat{\mathcal{C}} \neq \mathcal{C}] = P_{\text{ers,peeling}}(\epsilon) \leq \sum_{h=1}^{n} A_{ss}(h)\, \epsilon^h.$$