Low Density Parity Check Codes
Programming Exercises

Antoine O. Berthet

Dept. Signal – Information – Communications
CentraleSupélec – Paris-Saclay University

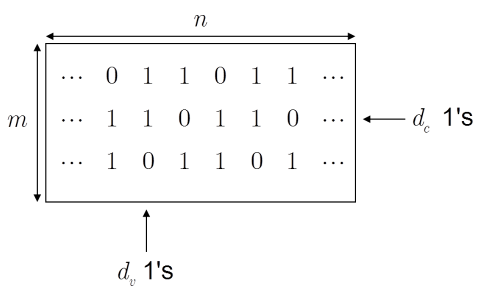
January 24, 2022

Presentation outline

LDPC codes

Definition

A regular-(d_v, d_c) binary low-density parity-check (LDPC) code of length n is a linear binary block code whose $m \times n$ parity-check matrix $\mathbf{H}$ is sparse with exactly d_v 1's on each column and d_c 1's on each row, under the constraints $d_v \ll n$ and $d_c \ll n$ and $md_c = nd_v$.



The code rate is given by

$$R_c \geq \frac{n-m}{n} = 1 - d_v/d_c$$

LDPC code structure

struct ldpc

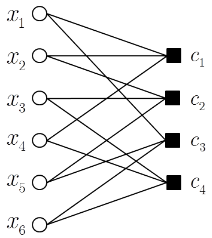
- number n of columns of LDPC matrix = code length
- number m of rows of LDPC matrix
- code dimension $k = \text{rank}(\mathbf{G}) = n - \text{rank}(\mathbf{H})$
- row degree profile $\mathbf{d}_c$
- column degree profile $\mathbf{d}_v$
- ldpc matrix, sparse format, non-zero entries
- indicates if the next two fields are empty or not
- submatrix $\mathbf{P}$ in $\mathbf{G}$ of dimensions $k \times m$
- column permutation array (see above)

Tanner graph

Definition

- A vector $\mathbf{x}$ is a valid configuration (codeword) of the behavior defined by the code iif all parity-check equations are simultaneously satisfied.
- The FG represents the factored characteristic function of this behavior.

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



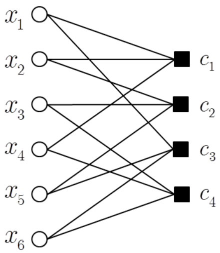
$\mathcal{N}_v(j) = \{i : \exists \text{ an edge } (x_j, c_i)\}$ $\quad \mathcal{N}_c(i) = \{j : \exists \text{ an edge } (x_j, c_i)\}$

Tanner graph structure

struct bipart

- number n_v of 1st type vertices (variable nodes)
- number n_c of 2nd type vertices (check nodes)
- number n_e of edges
- degree profile $\mathbf{d}_v$ of 1st type vertices (variable nodes)
- degree profile $\mathbf{d}_c$ of 2nd type vertices (check nodes)
- neighborhood $\mathcal{N}_v$ of 1st type vertices (variable nodes)
- neighborhood $\mathcal{N}_c$ of 2nd type vertices (check nodes)
- edge ensemble (first indexing convention)

Tanner graph structure



$$\begin{aligned} \mathcal{N}_v(1) &= \{1,3\} \\ \mathcal{N}_v(2) &= \{1,2\} \\ \mathcal{N}_v(3) &= \{2,4\} \\ \mathcal{N}_v(4) &= \{1,4\} \\ \mathcal{N}_v(5) &= \{2,3\} \\ \mathcal{N}_v(6) &= \{3,4\} \end{aligned}$$

$$\begin{aligned} \mathcal{N}_c(1) &= \{1,2,4\} \\ \mathcal{N}_c(2) &= \{2,3,5\} \\ \mathcal{N}_c(3) &= \{1,5,6\} \\ \mathcal{N}_c(4) &= \{3,4,6\} \end{aligned}$$

$$\begin{aligned} e(1) &= (v_1, c_1) \\ e(2) &= (v_2, c_1) \\ e(3) &= (v_4, c_1) \\ e(4) &= (v_2, c_2) \\ e(5) &= (v_3, c_2) \\ e(6) &= (v_5, c_2) \\ e(7) &= (v_1, c_3) \\ e(8) &= (v_5, c_3) \\ e(9) &= (v_6, c_3) \\ e(10) &= (v_3, c_4) \\ e(11) &= (v_4, c_4) \\ e(12) &= (v_6, c_4) \end{aligned}$$

Gallager's generation method I

Algorithm 1 Generation of a (d_v, d_c)-regular binary LDPC matrix

Require: m, n, d_v, d_c under condition $m.d_c = n.d_v$
Ensure: binary (d_v, d_c)-regular LDPC matrix of dimensions $m \times n$

generate the first submatrix $\mathbf{H}_1$ of dimensions $m/d_v \times n$ such that the i-th row, $i \leq m/d_v$, has non-zero entries in the $((i-1)d_c + 1)$-th to id_c-th columns
for $j = 2$ to d_v **do**
    construct $\mathbf{H}_j$ by randomly permutated the columns of $\mathbf{H}_1$
end for

MacKay and Neal's generation method

Definition

- Consider the partial matrix $\mathbf{H}^{(j)}$ of dimensions $m \times j$ made of $j \leq n$ already generated columns. The row degree profile $\mathbf{d}_c^{(j)}$ of $\mathbf{H}^{(j)}$ is the vector defined as

$$d_{c,i}^{(j)} = w_H\left(\mathbf{h}_i^{(j)}\right), \forall i \in \{1, \ldots, m\} \quad (1)$$

where $\mathbf{h}_i^{(j)}$ is the i^{th} row of $\mathbf{H}^{(j)}$

- The row degree profile $\mathbf{d}_c^{(j)}$ satisfies the degree constraints if

$$d_{c,i}^{(j)} \leq d_c, \forall i \in \{1, \ldots, m\} \quad (2)$$

- The row i is degree-saturated if $d_{c,i}^{(j)} = d_c$

MacKay and Neal's generation method I

Algorithm 2 Generation of a (d_v, d_c)-regular binary LDPC matrix

Require: m, n, d_v, d_c under condition $m.d_c = n.d_v$
Ensure: binary (d_v, d_c)-regular LDPC matrix of dimensions $m \times n$

```
set t ≤ n, set θ1 to a large value
j ← 1
backtrack:
s ← max{1, j − t}
optional: increase t (t ≤ n)
delete the last t + 1 columns of H^(j) and store it as H^(s−1)
for j = s to n do
    c1 ← 0
    repeat
        c1 ← c1 + 1
        if c1 > θ1 then
            goto backtrack
```


MacKay and Neal's generation method II

```
        else
            randomly generate weight-d_v column h̃^j
            evaluate d̃_c^(j) for H̃^(j) = [ H^(j−1)  h̃^j ]
        end if
    until d̃_c^(j) satisfies the degree constraints
    H^(j) ← H̃^(j)
    d_c^(j) ← d̃_c^(j)
end for
```


Storage of sparse LDPC matrices

Example: a length-12 $(3, 4)$-regular binary matrix

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

Storage of sparse LDPC matrices

Principle

- For each row (or each column), store only the non-zero positions

Example: sparse storage scheme by row.

$$\mathbf{H} = \left[\begin{array}{c|cccc} r1 & 1 & 6 & 8 & 10 \\ r2 & 1 & 4 & 5 & 11 \\ r3 & 2 & 5 & 7 & 9 \\ r4 & 3 & 6 & 11 & 12 \\ r5 & 3 & 7 & 8 & 12 \\ r6 & 2 & 5 & 9 & 11 \\ r7 & 1 & 4 & 7 & 10 \\ r8 & 2 & 6 & 8 & 10 \\ r9 & 3 & 4 & 9 & 12 \end{array}\right] \quad (4)$$

MacKay and Neal's generation method

Definition

- The row degree profile $\mathbf{d}_c^{(j)}$ satisfies the degree constraints if

$$d_{c,i}^{(j)} \leq d_{c,i}, \forall i \in \{1, \ldots, m\} \quad (5)$$

- The row i is degree-saturated if $d_{c,i}^{(j)} = d_{c,i}$

MacKay and Neal's generation method I

Algorithm 3 Generation of a $(\mathbf{d}_v, \mathbf{d}_c)$-irregular binary LDPC matrix

Require: $m, n, \mathbf{d}_v, \mathbf{d}_c$ under condition $\sum_{i=1}^{m} d_{c,i} = \sum_{j=1}^{n} d_{v,j}$
Ensure: binary irregular LDPC matrix of dimensions $m \times n$

set $t \leq n$, set θ_1 to a large value
$j \leftarrow 1$
backtrack:
$s \leftarrow \max\{1, j - t\}$
optional: increase t ($t \leq n$)
delete the last $t + 1$ columns of $\mathbf{H}^{(j)}$ and store it as $\mathbf{H}^{(s-1)}$
for $j = s$ to n **do**
  $c1 \leftarrow 0$
  repeat
    $c1 \leftarrow c1 + 1$
    if $c1 > \theta_1$ **then**
      goto backtrack

MacKay and Neal's generation method II

    else
      randomly generate weight-d_v column $\tilde{\mathbf{h}}^j$
      evaluate $\tilde{\mathbf{d}}_c^{(j)}$ for $\tilde{\mathbf{H}}^{(j)} = \left[\begin{array}{cc} \mathbf{H}^{(j-1)} & \tilde{\mathbf{h}}^j \end{array}\right]$
    end if
  until $\tilde{\mathbf{d}}_c^{(j)}$ satisfies the degree constraints
  $\mathbf{H}^{(j)} \leftarrow \tilde{\mathbf{H}}^{(j)}$
  $\mathbf{d}_c^{(j)} \leftarrow \tilde{\mathbf{d}}_c^{(j)}$
end for

Storage of sparse LDPC matrices

Example: A length-10 irregular binary matrix

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{6}$$

Storage of sparse LDPC matrices

Principle

For each row (or each column), store only the non-zero positions

Example: sparse storage scheme by row.

$$\mathbf{H} = \left[\begin{array}{c|ccccc} r1 & 1 & 2 & 4 & 5 & 8 \\ r2 & 2 & 3 & 5 & 6 & 7 \\ r3 & 4 & 8 & 9 & 10 & \\ r4 & 1 & 2 & 6 & 7 & 9 \\ r5 & 3 & 6 & 8 & 10 & \end{array}\right] \tag{7}$$

MacKay and Neal's generation method I

Algorithm 4 Generation of a full-rank $(\mathbf{d}_v, \mathbf{d}_c)$-irregular binary LDPC matrix

Require: $m, n, \mathbf{d}_v, \mathbf{d}_c$ under condition $\sum_{i=1}^{m} d_{c,i} = \sum_{j=1}^{n} d_{v,j}$
Ensure: binary irregular LDPC matrix of dimensions $m \times n$

set $t \leq n$, set θ_1, θ_2 to large values
$j \leftarrow 1$
$c1 \leftarrow 0$
repeat
  $c1 \leftarrow c1 + 1$
  if $c1 > \theta_1$ **then**
    exit with failure
  end if
  backtrack:
  $s \leftarrow \max\{1, j - t\}$
  optional: increase t $(t \leq n)$

MacKay and Neal's generation method II

  delete the last $t + 1$ columns of $\mathbf{H}^{(j)}$ and store it as $\mathbf{H}^{(s-1)}$
  for $j = s$ to n **do**
    $c2 \leftarrow 0$
    repeat
      $c2 \leftarrow c2 + 1$
      if $c2 > \theta_2$ **then**
        goto backtrack
      else
        randomly generate weight-d_v column $\tilde{\mathbf{h}}^j$
        evaluate $\tilde{\mathbf{d}}_c^{(j)}$ for $\tilde{\mathbf{H}}^{(j)} = \begin{bmatrix} \mathbf{H}^{(j-1)} & \tilde{\mathbf{h}}^j \end{bmatrix}$
      end if
    until $\tilde{\mathbf{d}}_c^{(j)}$ satisfies the degree constraints
    $\mathbf{H}^{(j)} \leftarrow \tilde{\mathbf{H}}^{(j)}$
    $\mathbf{d}_c^{(j)} \leftarrow \tilde{\mathbf{d}}_c^{(j)}$
  end for

MacKay and Neal's generation method III

compute rank($\mathbf{H}$)
until rank($\mathbf{H}$) $= m$

MacKay and Neal's generation method I

Algorithm 5 Generation of a full-rank $(\mathbf{d}_v, \mathbf{d}_c)$-irregular binary LDPC matrix without length-4 cycles

Require: $m, n, \mathbf{d}_v, \mathbf{d}_c$ under condition $\sum_{i=1}^{m} d_{c,i} = \sum_{j=1}^{n} d_{v,j}$
Ensure: binary irregular LDPC matrix of dimensions $m \times n$
set $t \leq n$, set $\theta_1, \theta_2, \theta_3$ to large values
$j \leftarrow 1$
$c1 \leftarrow 0$
repeat
$c1 \leftarrow c1 + 1$
if $c1 > \theta_1$ **then**
exit with failure
end if
backtrack:
$s \leftarrow \max\{1, j - t\}$
optional: increase t $(t \leq n)$

MacKay and Neal's generation method II

delete the last $t + 1$ columns of $\mathbf{H}^{(j)}$ and store it as $\mathbf{H}^{(s-1)}$
for $j = s$ to n **do**
$c2 \leftarrow 0$
repeat
$c2 \leftarrow c2 + 1$
if $c2 > \theta_2$ **then**
goto backtrack
end if
$c3 \leftarrow 0$
repeat
$c3 \leftarrow c3 + 1$
if $c3 > \theta_3$ **then**
goto backtrack
else
randomly generate weight-d_v column $\tilde{\mathbf{h}}^j$

MacKay and Neal's generation method III

evaluate $\tilde{\mathbf{d}}_c^{(j)}$ for $\tilde{\mathbf{H}}^{(j)} = \begin{bmatrix} \mathbf{H}^{(j-1)} & \tilde{\mathbf{h}}^j \end{bmatrix}$
end if
until $\tilde{\mathbf{d}}_c^{(j)}$ satisfies the degree constraints
check the presence of length-4 cycles in $\tilde{\mathbf{H}}^{(j)}$
until no length-4 cycle detected
$\mathbf{H}^{(j)} \leftarrow \tilde{\mathbf{H}}^{(j)}$
$\mathbf{d}_c^{(j)} \leftarrow \tilde{\mathbf{d}}_c^{(j)}$
end for
compute rank($\mathbf{H}$)
until rank($\mathbf{H}$) $= m$

Pseudo-code I

Algorithm 6 LBP for the BEC

Require: Tanner graph of the code, received word $\mathbf{y}$

preliminary test:
if $\forall j \in \{1, \dots, n\}$ $y_j \neq e$ **then**
    return $\mathbf{y}$
end if
define two arrays $\mathbf{\Phi}$ and $\mathbf{\Psi}$ of dimensions $n \times m$ and $m \times n$
$l \leftarrow 0$
init: *compute beliefs*
for $j = 1$ to n **do**
    if $y_j \neq e$ **then**
        $\beta_j \leftarrow y_j$
    end if
end for
init: *compute bit-to-check messages*

Pseudo-code II

for $j = 1$ to n **do**
    for all $i \in \mathcal{N}_v(j)$ **do**
        $\Phi_{j,i} \leftarrow y_j$
    end for
end for
repeat
    $l \leftarrow l + 1$
    step: *compute check-to-bit messages*
    for $i = 1$ to m **do**
        for all $j \in \mathcal{N}_c(i)$ **do**
            if $\forall k \in \mathcal{N}_c(i) \backslash \{j\}$, $\Phi_{k,i} \neq e$ **then**
                $\Psi_{i,j} \leftarrow \sum_{k \in \mathcal{N}_c(i) \backslash \{j\}} \Phi_{k,i} \pmod 2$
            else
                $\Psi_{i,j} \leftarrow e$
            end if

Pseudo-code III

        end for
    end for
    step: *compute beliefs*
    for $j = 1$ to n **do**
        if $y_j \neq e$ **then**
            $\beta_j \leftarrow y_j$
        else
            for all $i \in \mathcal{N}_v(j)$ **do**
                if $\forall k \in \mathcal{N}_v(j)$, $\Psi_{k,j} = e$ **then**
                    $\beta_j \leftarrow e$
                else if $\exists k \in \mathcal{N}_v(j)$ s.t. $\Psi_{k,j} \neq e$ **then**
                    $\beta_j \leftarrow \Psi_{k,j}$
                end if
            end for
        end if

Pseudo-code IV

    end for
    step: *compute bit-to-check messages*
    for $j = 1$ to n **do**
        if $y_j \neq e$ **then**
            for all $i \in \mathcal{N}_v(j)$ **do**
                $\Phi_{j,i} \leftarrow y_j$
            end for
        else
            for all $i \in \mathcal{N}_v(j)$ **do**
                if $\forall k \in \mathcal{N}_v(j) \backslash \{i\}$, $\Psi_{k,j} = e$ **then**
                    $\Phi_{j,i} \leftarrow e$
                else if $\exists k \in \mathcal{N}_v(j) \backslash \{i\}$ s.t. $\Psi_{k,j} \neq e$ **then**
                    $\Phi_{j,i} \leftarrow \Psi_{k,j}$
                end if
            end for

Pseudo-code V

    end if
  end for
  test: stopping criterion
  if all bit-to-check messages are either 0 or 1 $\vee\ l = l_{\max}$ **then**
    $STOP \leftarrow 1$
  end if
until $STOP = 1$
return β

Pseudo-code I

Algorithm 7 LBP for BMC

Require: Tanner graph, vector $\mathbf{\Gamma}$ of log likelihood ratios (LLRs) of coded bits, maximum number N_{it} of iterations
Ensure: vector $\mathbf{\Lambda}$ of log (approximated) APPs of coded bits and/or estimated codeword $\hat{\mathbf{x}}$
preliminary test: check syndrome
for $j = 1$ to n **do**
  if $\Gamma_j \geq 0$ **then**
    $\hat{x}_j \leftarrow 0$
  else
    $\hat{x}_j \leftarrow 1$
  end if
end for
if $\mathbf{z} = \hat{\mathbf{x}}\mathbf{H}^\top = \mathbf{0}$ **then**
  return $\hat{\mathbf{x}}$

Pseudo-code II

end if
define two arrays $\mathbf{\Phi}$ and $\mathbf{\Psi}$ of dimensions $n \times m$ and $m \times n$
init: get LLRs from channel (in parallel)
for $j = 1$ to n **do**
  for all $k \in \mathcal{N}_v(j)$ **do**
    $\Phi_{j,k} \leftarrow \Gamma_j$
  end for
end for
for $it = 1$ to N_{it} **do**
  step 1: *compute check-to-bit messages*
  for $i = 1$ to m **do**
    for all $k \in \mathcal{N}_c(i)$ **do**
      $\Psi_{i,k} \leftarrow 2\tanh^{-1}\left(\prod_{k' \in \mathcal{N}_c(i)\setminus\{k\}} \tanh\left(\frac{\Phi_{k',i}}{2}\right)\right)$
    end for
  end for

Pseudo-code III

  stop criterion: check syndrome
  step: *compute beliefs*
  for $j = 1$ to n **do**
    $\Lambda_j \leftarrow \Gamma_j + \sum_{k \in \mathcal{N}_v(j)} \Psi_{k,j}$
    if $\Lambda_j \geq 0$ **then**
      $\hat{x}_j \leftarrow 0$
    else
      $\hat{x}_j \leftarrow 1$
    end if
  end for
  if $\mathbf{z} = \hat{\mathbf{x}}\mathbf{H}^\top = \mathbf{0}$ **then**
    return $\mathbf{\Lambda}$ and/or $\hat{\mathbf{x}}$
  end if
  step 2: *compute bit-to-check messages*
  for $j = 1$ to n **do**

Pseudo-code IV

for all $k \in \mathcal{N}_v(j)$ **do**
  $\Phi_{j,k} \leftarrow \Gamma_j + \sum_{k' \in \mathcal{N}_v(j) \setminus \{k\}} \Psi_{k',j}$
  // $\Phi_{j,k} \leftarrow \Lambda_j - \Psi_{k,j}$ (to optimize the execution speed) //
end for
end for
end for

Monte Carlo simulation I

Algorithm 8 Performance of an LDPC codes over a BIAWGNC under message-passing decoding (SPA) (length $2^8 \leq n \leq 2^{20}$)

Require: generator matrix $\mathbf{G}_{rr}$, Tanner graph for LDPC matrix $\mathbf{H}$
Ensure: bit error probability P_b, block error probability P_e

init: open and read files, allocate workspace, store all structures, etc.
for all SNR in SNR range (dB) **do**
  compute the noise variance σ
  initialize bit and block error counters
  repeat
    randomly generate[1] an information vector $\mathbf{u} \in \mathbb{F}_2^k$
    encode $\mathbf{u}$ into codeword $\mathbf{x} = \mathbf{u}\mathbf{G}_{rr} \in \mathbb{F}_2^n$ (systematic encoding)
    BPSK modulate[2] codeword $\mathbf{x}$ into vector $\mathbf{s} \in \{\pm 1\}^n$
    transmit $\mathbf{s}$ on the BIAWGNC(σ) and generate the output[3] $\mathbf{y} \in \mathbb{R}^n$
    compute the vector of LLRs[4] $\mathbf{\Gamma} \in \mathbb{R}^n$
    permute the components of $\mathbf{\Gamma}$ if necessary[5]

Monte Carlo simulation II

    activate LBP[6] and find the vector of (approx) LAPPs $\mathbf{\Lambda} \in \mathbb{R}^n$
    deduce from $\mathbf{\Lambda}$ an estimate $\hat{\mathbf{u}} \in \mathbb{F}_2^k$ of $\mathbf{u}$
    update bit and block error counters
  until 10^6 codewords simulated or 500 codeword errors
  compute bit error probability and block error probability
end for
plot $P_b = f(SNR)$ and $P_e = g(SNR)$

[1] Due to symmetries (channel and message-passing decoder), the average bit (resp. block) error probability is equal to the bit (resp. block) error probability conditional on all-zero codeword transmitted.

[2] $\forall l \in \{1, \ldots, n\}$, $s_l = (-1)^{x_l}$.

[3] $\forall l \in \{1, \ldots, n\}$, $y_l = s_l + w_l$ where $w_l \sim \mathcal{N}(0, \sigma^2)$ with $\sigma^2 = \frac{N_0}{2R_c E_b}$.

[4] $\forall l \in \{1, \ldots, n\}$, $\Gamma_l = \log \frac{p(y_l|x_l=0)}{p(y_l|x_l=1)} = \frac{2}{\sigma^2} y_l$ (prove it).

[5] The codes used to encode and decode are equivalent but not necessarily identical.

[6] N_{it} is fixed and typicall large, e.g., $N_{it} = 100$, to ensure LBP convergence.