An Introduction to Low Density Parity Check (LDPC) Codes

Jian Sun

jian@csee.wvu.edu

Wireless Communication Research Laboratory

Lane Dept. of Comp. Sci. and Elec. Engr.

West Virginia University

Outline

1. History of LDPC codes
2. Properties of LDPC codes
3. Basics of LDPC codes
   - Encoding of LDPC codes
   - Iterative decoding of LDPC codes
   - Simplified approximations of LDPC decoders
4. Applications of LDPC codes

Features of LDPC Codes

- Approaching Shannon capacity
  - For example, 0.3 dB from Shannon limit
  - Irregular LDPC code with code length 1 million. (Richardson:1999)
  - An closer design from (Chung:2001), 0.0045 dB away from capapcity
- Good block error correcting performance
- Low error floor
  - The minimum distance is proportional to code length
- Linear decoding complexity in time
- Suitable for parallel implementation

History of LDPC Codes

- Invented by Robert Gallager in his 1960 MIT Ph. D. dissertation. Long time being ignored due to
  1. Requirement of high complexity computation
  2. Introduction of Reed-Solomon codes
  3. The concatenated RS and convolutional codes were considered perfectly suitable for error control coding.
- Rediscovered by MacKay(1999) and Richardson/Urbanke(1998).

Foundamentals of Linear Block Codes

- The structure of a code is completely described by the generator matrix $\mathbf{G}$ or the parity check matrix $\mathbf{H}$.
- The capacity of correcting symbol errors in a codeword is determined by the minimum distance (d_{min}).
  - d_{min} is the least weight of the rows in $\mathbf{G}$.
  - d_{min} is the least number of columns in $\mathbf{H}$ that sum up to $\mathbf{0}$.
  - Example: (7, 4) Hamming code

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

Properties of LDPC Codes

- **H** is sparse.
  - Very few 1's in each row and column.
  - Expected large minimum distance.
- Regular LDPC codes
  - **H** contains exactly W_c 1's per column and exactly $W_r = W_c(n/m)$ 1's per row, where $W_c << m$.
  - The above definition implies that $W_r << n$.
  - $W_c \geq 3$ is necessary for good codes.
- If the number of 1's per column or row is not constant, the code is an *irregular* LDPC code.
  - Usually irregular LDPC codes outperforms regular LDPC codes.

A Sample LDPC Code

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

- $W_c = 3$;
- Any two columns have an overlap of at most one 1;
- The sparse property allows us to avoid overlapping;
- In the part of $\mathbf{H}$ shown, there does not exist a set of columns that add up to $\mathbf{0}$.
- The above facts make the d_{min} large;
- $\mathbf{G}$ is found by Gaussian elimination.
  - $\mathbf{H}$ can be put in the form $\mathbf{H} = \left[\mathbf{P}^T \vdots \mathbf{I} \right]$.
  - The generator matrix $\mathbf{G} = \left[\mathbf{I} \vdots \mathbf{P} \right]$.

Encoding of LDPC Codes

- General encoding of systematic linear block codes

$$\mathbf{c} = \mathbf{x}\mathbf{G} = \left[\mathbf{x} \vdots \mathbf{x}\mathbf{P} \right] \tag{1}$$

- Issues with LDPC codes
  - The size of $\mathbf{G}$ is very large.
  - $\mathbf{G}$ is not generally sparse.
  - Example: A (10000, 5000) LDPC code.
    * $\mathbf{P}$ is 5000×5000.
    * We may assume that the density of 1's in $\mathbf{P}$ is 0.5
    * There are 12.5×10^6 1's in $\mathbf{P}$
    * 12.5×10^6 addition (XOR) operations are required to encode one codeword.
- An alternative approach to simplified encoding is to design the LDPC code via algebraic or geometric methods.
  - Such "structured" codes can be encoded with shift register circuits.

Iterative Decoding of LDPC Codes

- General decoding of linear block codes
  - Only if $\mathbf{c}$ is a valid codeword, we have

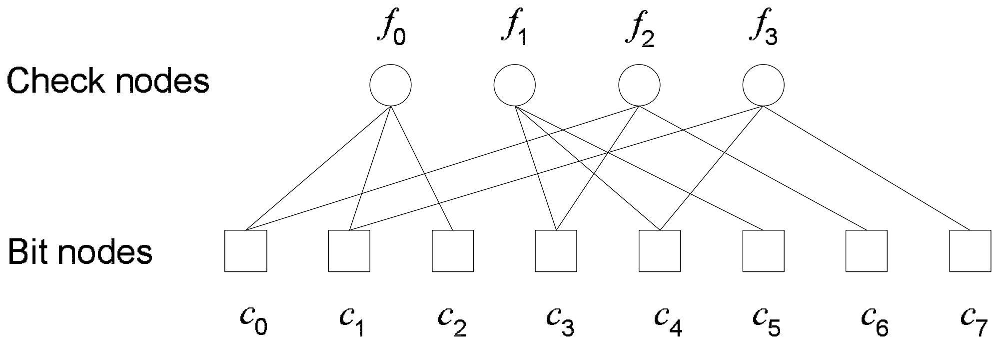
$$\mathbf{c}\mathbf{H}^T = \mathbf{0} \tag{2}$$

  - For binary symmetric channel (BSC), the received codeword is $\mathbf{c}$ added with an error vector $\mathbf{e}$.
  - The decoder needs to find out $\mathbf{e}$ and flip the corresponding bits.
  - The decoding algorithm is based on linear algebra.
- Graph-based algorithms
  - Sum-product algorithm for general graph-based codes;
  - MAP (BCJR) algorithm for trellis graph-based codes;
  - Message passing algorithm for bipartite graph-based codes.

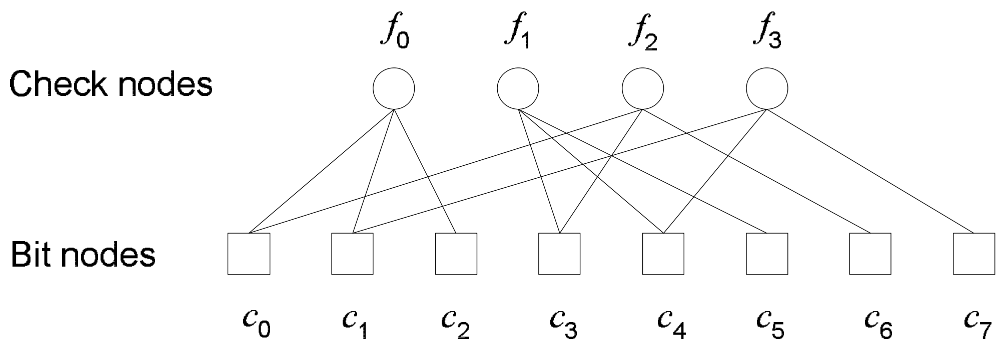
Tanner Graph

- Bipartite graph
  A bipartite graph is an undirected graph whose nodes may be separeted into two classes, where edges only connect two nodes not residing in the same class.
- Tanner graph
  The two classes of nodes in a Tanner graph are the *bit nodes* and the *check nodes*.
- Example: An (8, 4) product code

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



Message Passing Algorithm (1/4)



- Maximum likelihood (ML) decoding

$$L = \frac{\Pr[c_i = 1\,|\mathbf{y}]}{\Pr[c_i = 0\,|\mathbf{y}]}$$

$$\hat{c}_i = \begin{cases} 1 & \text{if } L \geq 1 \\ 0 & \text{if } L < 1 \end{cases}$$

- Constraint

$$\mathbf{c}\mathbf{H}^T = \mathbf{0} \tag{3}$$

Message Passing Algorithm (2/4)

- Messages are probabilities (or likelihood) of “1” or “0” been transmitted
- Similar to *extrinsic information* in BCJR algorithm.
- Two stages of message passing.
  - Probabilities of bit nodes;
  - Probabilities of check nodes.
- Assumptions
  - Independence of *a posterio* probabilities

Message Passing Algorith (3/4)

- Denotations
  - q_{ij} — messages to be passed from bit node c_i to check nodes f_j.
  - r_{ji} — messages to be pased from check node f_j to bit node c_i.
  - $R_j = \{i : h_{ji} = 1\}$ — the set of column locations of the 1's in the jth row
  - $R_{j\backslash i} = \{i' : h_{ji'} = 1\} \setminus \{i\}$ — the set of column locations of the 1's in the jth row, excluding location i.
  - $C_j = \{i : h_{ji} = 1\}$ — the set of row locations of the 1's in the ith column
  - $C_{i\backslash j} = \{i' : h_{j'i} = 1\} \setminus \{j\}$ — the set of row locations of the 1's in the ith column, excluding location j.
  - $p_i = \Pr(c_i = 1|y_i)$

Message Passing Algorith (4/4)

Compute for $\forall i, j$ that satisfies $h_{ij} = 1$.

1. Initialize

$$\begin{aligned} q_{ij}(0) &= 1 - p_i = \Pr(c_i = 0 \,|y_i) = \frac{1}{1 + e^{-2y_i/\sigma^2}} \\ q_{ij}(1) &= p_i = \Pr(c_i = 1 \,|y_i) = \frac{1}{1 + e^{2y_i/\sigma^2}} \end{aligned}$$

2. First half round iteration

$$\begin{aligned} r_{ji}(0) &= \frac{1}{2} + \frac{1}{2} \prod_{i' \in R_{j \setminus i}} \left(1 - 2q_{i'j}(1)\right) \\ r_{ji}(1) &= 1 - r_{ji}(0) \end{aligned}$$

3. Second half round iteration

$$\begin{aligned} q_{ij}(0) &= K_{ij}(1 - p_i) \prod_{j' \in C_{i \setminus j}} r_{j'i}(0) \\ q_{ij}(1) &= K_{ij} p_i \prod_{j' \in C_{i \setminus j}} r_{j'i}(1) \end{aligned}$$

where constants k_{ij} are selected to ensure

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

4. Soft decision

$$\begin{aligned} Q_i(0) &= K_i (1 - p_i) \prod_{j \in C_i} r_{ij}(0) \\ Q_i(1) &= K_i p_i \prod_{j \in C_i} r_{ij}(1) \end{aligned}$$

where constants k_i are selected to ensure

$$Q_i(0) + Q_i(1) = 1$$

5. Hard decision

$$\hat{c}_i = \begin{cases} 1 & \text{if } Q_i(1) > 0 \\ 0 & \text{elsewhere} \end{cases}$$

If $\hat{\mathbf{c}}\mathbf{H}^T = \mathbf{0}$ or number of iterations exceeds limitation then stop, else go to Step 2.

Log-Domain Algorithm (1/2)

- A log-domain algorithm is desirable because there are many multiplications
  - In log-domain, multiplications will become additions which have less computational complexity;
  - Multiplications may cause overflow or saturation with large numbers of iterations.
- The log-likelihood ratio is defined as

$$L(c_i) \triangleq \log \frac{1-p_i}{p_i}$$

$$L(q_{ij}) \triangleq \log \frac{q_{ij}(0)}{q_{ij}(1)}$$

- The most frequently involved computation can be defined as

$$\phi(x) \triangleq -\log \tanh\left(\frac{1}{2}x\right) = \log \frac{e^x+1}{e^x-1}$$

  - We have the property $\phi^{-1}(x) = \phi(x)$ for $x > 0$

Log-Domain Algorithm (2/2)

- Seperate $L\left(q_{ij}\right)$

$$\begin{aligned} L\left(q_{ij}\right) &= \alpha_{ij}\beta_{ij} \\ \alpha_{ij} &= \operatorname{sign}\left(L\left(q_{ij}\right)\right) \\ \beta_{ij} &= \operatorname{abs}\left(L\left(q_{ij}\right)\right) \end{aligned}$$

- The log-domain algorithm
  1. Initialize

$$L\left(q_{ij}\right) = 2y_i/\sigma^2$$

  2. First half round iteration

$$L\left(r_{ji}\right) = \prod_{i' \in R_{j\setminus i}} \alpha_{i'j} \cdot \phi\left[\sum_{i' \in R_{j\setminus i}} \phi\left(\beta_{i'j}\right)\right]$$

3. Second half round iteration

$$L\left(q_{ij}\right) = L\left(c_i\right) + \sum_{j' \in C_{i \backslash j}} L\left(r_{j'i}\right)$$

4. Soft decision

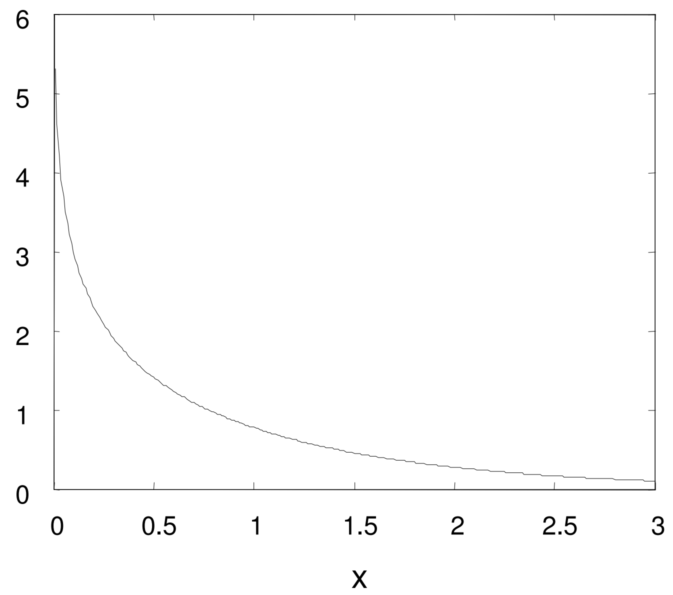
$$L\left(Q_i\right) = L\left(C_i\right) + \sum_{j \in C_i} L\left(r_{ji}\right)$$

5. Hard decision

$$\hat{c}_i = \begin{cases} 1 & \text{if } L\left(Q_i\right) < 0 \\ 0 & \text{elsewhere} \end{cases}$$

If $\hat{\mathbf{c}}\mathbf{H}^T = \mathbf{0}$ or number of iterations exceeds limitation then stop, else go to Step 2.

Min-Sum Algorithm



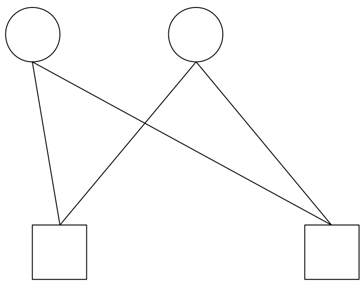
- The shape of $\phi(x)$.
- The smallest β_{ij} dominates.

$$\begin{aligned}\phi\left[\sum_{i'}\phi\left(\beta_{i'j}\right)\right] &\approx \phi\left[\phi\left(\min_{i'}\beta_{i'j}\right)\right]\\ &= \min_{i'}\beta_{i'j}\end{aligned}$$

- The min-sum algorithm is the log-domain algorithm with step 2 modified by

$$L\left(r_{ji}\right) = \prod_{i'\in R_{j\backslash i}}\alpha_{i'j}\cdot\min_{i'\in R_{j\backslash i}}\beta_{i'j}$$

Design of LDPC Codes



- Large d_{min}
- No short cycles
  - Cycles exist in Tanner graphs.
  - Cycles hurt the performance of the message passing algorithm because they invalidate the assumption of independence.
  - The shortest possible cycle has the length 4.
  - Although we can eliminate all cycles with length 4, we may still have cycles with length 6.
- No eliminating sets
  - Applications in binary erasure channels.

Open Problems in LDPC Codes

- LDPC codes with near-capacity performance
  - Very long codewords, many iterations, low signal-to-noise ratio.
- LDPC codes with relatively short codeword
  - High coding rate $r \approx 1$
  - Short codewords enable easy encoding
- Combination with other technologies
  - LDPC codes with OFDM systems
  - LDPC codes with MIMO systems

References

[1] R. Gallager, "Low-density parity-check codes," *IRE Trans. Information Theory*, pp. 21–28, January 1962.

[2] D. MacKay, "Good error correcting codes based on very sparse matrices," *IEEE Trans. Information Theory*, pp. 399–431, March 1999.

[3] J. L. Fan, *Constrained coding and soft iterative decoding for storage*. PhD thesis, Stanford University, 1999.

[4] T. richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 638–656, Feb. 2001.

[5] Y. Kou, S. Lin, and M. Fossorier, "Low-density parity-check codes based on finite geometries: a rediscovery and new results," *IEEE Trans. Inform. Theory*, vol. 47, pp. 2711–2736, Nov. 2001.