LDPC Codes, Construction and Applications

Manu T S

October 19, 2013

Abstract

In this report we study construction of LDPC codes and implementation of encoder and decoder for LDPC codes. We implement the progressive edge growth algorithm, algorith for construction of LDPC codes based on Reed-Solomon Codes. We also implement an encoder for LDPC codes. For decoder belief propogation decoder is implemented.

1 Motivation

Coding Theory is one of the foundations of Communication Engineering. Search for better and better codes has always been an area of intense research. For a long time there was a significant gap between the theoretical capacity and practically achievable capacity. It is proved that LDPC codes introduced by Gallager, form a class of capacity approaching codes. The research in that area was dormant for a long time till they were rediscovered in the late 1990s[6]. It is shown that these codes decoded with iterative decoding algorithms like sum-product algorithm[6] or message passing algorithms[1] achieve good error performance.

GNU Radio provides a free and open-source platform for implementing practical communication systems using various compatible hardware like USRP, and also provides a learning and simulation tool. The use of GNU Radio in research related to Software Defined Radio, Digital Signal Processing and Communication Engineering is becoming increasingly widespread. In this project we implement a generic encoder and decoder for LDPC codes in GNU Radio and also implement algorithms for construction of LDPC codes.

2 LDPC Codes

The LDPC codes, as the name suggests is characterized by a sparse parity check matrix $\mathbf{H}$. From parity check matrix we can construct the *Tanner graph* of the code. For example fig. 1 represents the Tanner graph of [7, 4, 3] Hamming code with

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

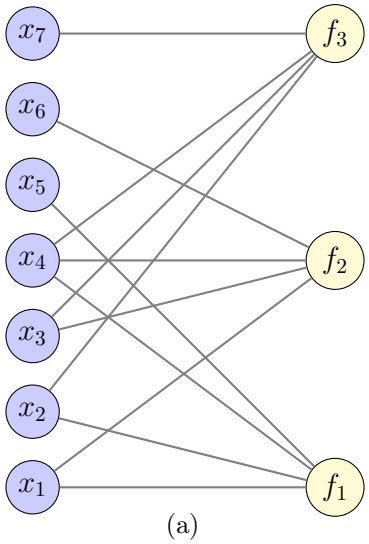
A codeword $\mathbf{x}$ from the codebook described by parity check matrix $\mathbf{H}$ should satisfy

$$\mathbf{Hx} = \mathbf{0} \tag{2}$$

In the Tanner graph of a (t_c, t_r) regular binary LDPC code, each variable node will have degree t_c and each check node will have degree t_r. Irregular LDPC codes can have different degrees for nodes. Let λ and ρ be vectors such that their i^{th} component λ_i and ρ_i represent the fraction of edges connecting to a variable node of degree i and check node of degree i respectively. Thus

$$\lambda(x) = \sum_i \lambda_i x^{i-1} \quad \text{and} \quad \rho(x) = \sum_i \rho_i x^{i-1}$$

are called variable and check degree distribution polynomials respectively. Standard ensemble LDPC(n, λ, ρ) of bipartite graphs is defined as the set of bipartite graphs with n variable nodes, variable degree distribution $\lambda(x)$ and check degree distribution $\rho(x)$.



(a)

Figure 1: Tanner graph of [7, 4, 3] Hamming code with **H** given in eq. (1). The blue nodes on the left represents the components of a codeword. They are called variable nodes. The yellow nodes on the right represents each of the constraints in the system eq. (2). They are called check nodes. When $h_{i,j}$ is 1 there exists a link between j^{th} variable node and i^{th} check node.

3 Marginalization Via Message Passing

Consider a function f with factorization:

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = f_1(x_1, x_2, x_3) f_2(x_1, x_4, x_6) f_3(x_4) f_4(x_4, x_5) \tag{3}$$

Now, the marginal of function f w.r.t x_1 is:

$$f(x_1) = \sum_{x_2, x_3, x_4, x_5, x_6} f(x_1, x_2, x_3, x_4, x_5, x_6) \tag{4}$$

$$= \left[\sum_{x_2, x_3} f_1(x_1, x_2, x_3)\right] \left[\sum_{x_4} f_3(x_4) \left(\sum_{x_6} f_2(x_1, x_4, x_6)\right) \left(\sum_{x_5} f_4(x_4, x_5)\right)\right] \tag{5}$$

Factor graph associated with above factorization is given in Figure 2. Here the red circles represent the

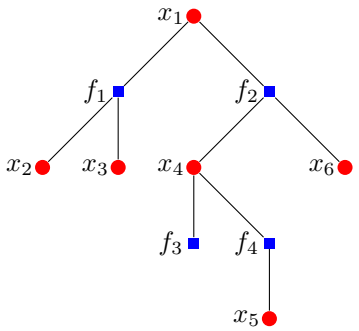


Figure 2: Factor Graph

variables $(x_1, x_2, \cdots, x_6)$ and are called the *variable-nodes*. Blue circles represent the factor (f_1,f_2,f_3 and f_4) and referred as the *check-nodes*. In the factor graph, a variable-node is connected to a check-node iff the corresponding variable takes part in that factor. Notice that this factor graph is a tree (i.e. there is no cycle in the graph). In the case where the factor graph is a tree the marginalization problem can be broken into smaller tasks according to the structure of the tree. The algorithm is explained in the next paragraph

The alogorithm proceeds by sending messages along the edges of the tree. Messages are functions on $\mathcal{X}$, or equivalently, vectors of length $|\mathcal{X}|$. The messages signify marginals of parts of the function and these parts are combined to form the marginal of the whole function. Message passing originates at the leaf nodes. Messages are passed up the tree and as soon as a node has received messages from all its children, the incoming messages are processed and the result is passed up to the parent node. Figure 3 explains the computation to be performed at the nodes during marginalization.

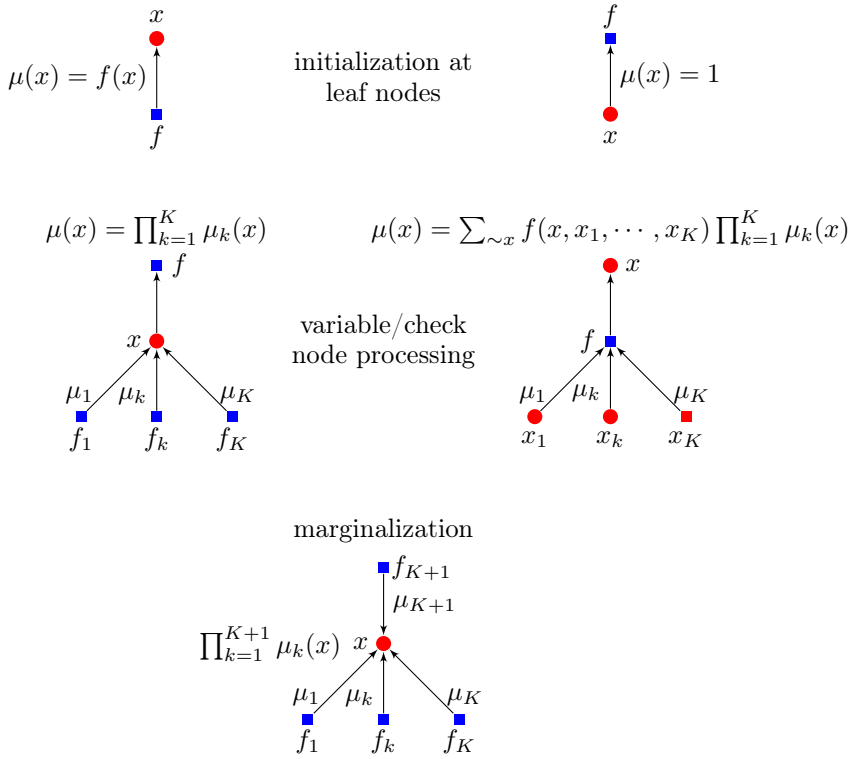


Figure 3: Message Passing Rules

To compute tha marginal with respect to all the variables, one can consider for each variable, a tree rooted in this variable and execute single marginal message passing algorithm on each rooted tree. Here all the computations can be performed simultaneously on a single tree. Start at all leaf nodes and for every edge compute the outgoing message along this edge as soon as you have the incoming messages along all other edges that connect to a given node.

4 Decoding Via Message Passing

For memoryless channel without feedback, bitwise maximum a posteriori (MAP) decoding of a codeword is given by [1]:

$$\hat{x}_i(y) = \arg\max_{x_i} p_{X_i|Y}(x_i|y) \tag{6}$$

$$= \arg\max_{x_i} \sum_{\sim x_i} p_{X|Y}(x|y) \tag{7}$$

$$= \arg\max_{x_i} \sum_{\sim x_i} p_{Y|X}(y|x) p_X(x) \tag{8}$$

$$= \arg\max_{x_i} \sum_{\sim x_i} \prod_j p_{Y_j|X_j}(y_j|x_j) p_X(x) \tag{9}$$

The notation $\sum_{\sim x_i}$ indicates that sum is over all components of x, except x_i. We obtain (7) from (6) by total probability rule. On application of Bayes's rule, we obtain (8) from (7). Equation (7) has very high computational complexity. For a codeword of length n and rate R, approximate computational

complexity is $O(2^{nR})$. The computational complexity can be reduced by factorizing the sequence probability. To this end, in equation(9), we use the fact that the channel is memoryless, without feedback. Furthermore, since all the input codewords are equiprobable,

$$\hat{x}_i(y) = \arg\max_{x_i} \sum_{\sim x_i} \prod_j p_{Y_j|X_j}(y_j|x_j)\mathbf{1}_{\{x\in C\}} \tag{10}$$

Where $\mathbf{1}_{\{x\in C\}}$ is a code membership function for the codebook C. Code membership function $\mathbf{1}_{\{x\in C\}}$ has a factorized form, and this factorization is used in efficient decoding.

4.1 Message Passing decoder for Binary Erasure Channel

By the standard message-passing rules the initial messages are $(\mu_j(0), \mu_j(1)) = (p_{Y_j|X_j}(y_j|0), (p_{Y_j|X_j}(y_j|1))$. In the case of binary erasure channel, the messages are either $(1-\epsilon, 0), (\epsilon, \epsilon)$, or $(0, 1-\epsilon)$ corresponding to 0, ? (erasure), or 1 respectively received from channel output. In this case we can normalize the message to (1, 0), (1, 1) and (0, 1) correspondingly. Now the message processing rules are discussed.

Claim : For BEC general message passing rules shown in Figure 3 simplify to the following. At a variable node the outgoing message is an erasure if all incoming messages are erasures. Otherwise, since the channel never introduces errors, all non-erasure messages must agree and either be 0 or 1. In this case the outgoing message is equal to this common value. At a check node the outgoing message is an erasure if anu of the incoming message is an erasure. Otherwise, if all the incoming messages are either 0 or 1, then the outgoing message is the mod-2 sum of the incoming messages

Proof : If all messages entering a variable node are from the set $\{(1,0),(1,1)\}$, then the outgoing message, which is equal to the component-wise product of the incoming messages according to the general message-passing rules, is also from this set. Further, it is equal to (1, 1) only if all incoming messages are of the form (1, 1) i.e, erasures. The equivalent statement is true if all incoming messages are from the set $\{(1,0),(1,1)\}$. Since channel never introduces errors, we only need to consider these two cases.

Next we consider the claim concerning the message-passing rule at a check node: We consider only a check node of degree 3 with two incoming messages, since check node of higher degree can be modeled as the cascade of several check nodes, each of which has 2 inputs. Let $(\mu_1(0), \mu_1(1))$, and $(\mu_2(0), \mu_2(1))$ denote the incoming messages. By the message processing rules the outgoing message is

$$\begin{aligned}(\mu(0), \mu(1)) &= \left(\sum_{x_1,x_2} \mathbf{1}_{\{x_1+x_2=0\}}\mu_1(x_1)\mu_2(x_2), \sum_{x_1,x_2} \mathbf{1}_{\{x_1+x_2=1\}}\mu_1(x_1)\mu_2(x_2)\right) &(11)\\ &= (\mu_1(0)\mu_2(0) + \mu_1(1)\mu_2(1), \mu_1(0)\mu_2(1) + \mu_1(1)\mu_2(0)) &(12)\end{aligned}$$

If $(\mu_j(0), \mu_j(1)) = (1,1), j \in \{1,2\}$, then after normalization, $(\mu(0), \mu(1)) = (1,1)$. Thus if any of the input is an erasure then output is an erasure. Also if both messages are known, then by explicite check, we can see that the output corresponds to mod-2 sum. Figure 4 shows an example of this decoder on (7, 4, 3) Hamming code with parity check matrix given in Equation(1)

4.2 Message Passing Decoder for Binary Input AWGN Channel

Given in Figure 5 is a mathematical model of AWGN channel. We think that the channel input $x_i \in \{-1, +1\}$. Here again a message can be thought of as a real-valued vector of length 2, $(\mu(-1), \mu(1))$. We initialize the message passing iterations by sending from i^{th} variable node the vector $(p_{Y_i|X_i}(y_i| -1), p_{Y_i|X_i}(y_i|1))$. In case of binary input channels, it suffices to send the log-likelihood ratios.

$$l = \log\left(\frac{\mu(-1)}{\mu(1)}\right)$$

On using log-likelihood ratios, the message passing rules simplipy to the following. At variable node, simply add the incoming messages and at a check node of degree $J+1$, use the following computation.

$$l = 2\,\mathrm{arc\,tanh}\left(\prod_{j=1}^{J} \tanh\left(l_j/2\right)\right) \tag{13}$$

In the case of Binary Input AWGN Channel, we can initialize the message passing iterations by sending

$$
\begin{aligned}
l_i &= \log\left(\frac{\mu(-1)}{\mu(1)}\right) && (14)\\
&= \log\left(\frac{p_{Y_i|X_i}(y_i|-1)}{p_{Y_i|X_i}(y_i|1)}\right) && (15)\\
&= \frac{-2y_i}{\sigma^2} && (16)
\end{aligned}
$$

from variable nodes to the check nodes. Further computations are carried out as expained above. One iteration consists of message passing from variable nodes to check nodes, computations at the check nodes and message passing from check nodes to variable nodes. After one such iteration, we find the probability ratios by the marginalization step shown in the Figure 3. Based on the probability ratios, we estimate the codeword, by the following rule. If the probability ratio is greater than 0, we decode that bit in favor of -1 and else in favor of 1. We check if the estimated codeword is an actual codeword by parity checks. If the estimated codeword is an actual codeword, we stop, else we continue with the next round of message passing iterations.

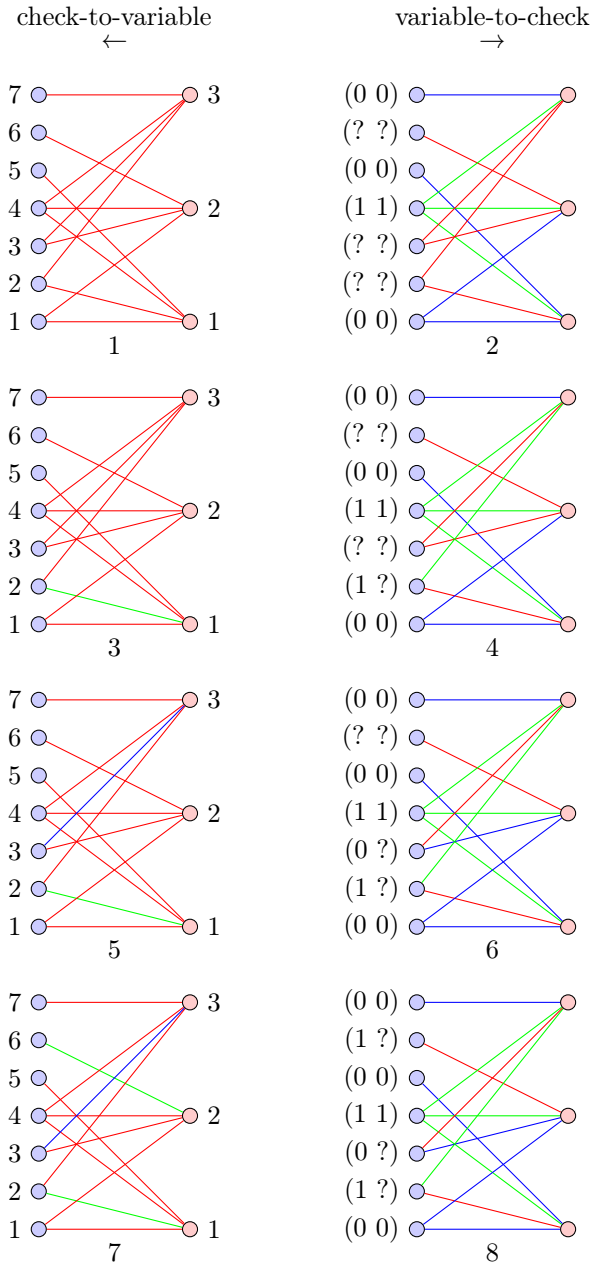


Figure 4: Message-passing decoding of the [7, 4, 3] Hamming code with the received code word y = (0, ?, ?, 1, 0, ?, 0). The left column is represents message passing from check to variable nodes. Right column represents message passing from variable to check nodes. Red links represents erasure messages, blue links represents messages 1 (corresponds to codebit 0) and green links represents message -1 (corresponds to codebit 1). The labels on the variable nodes in the right column represents the pair (estimated bit, received bit)

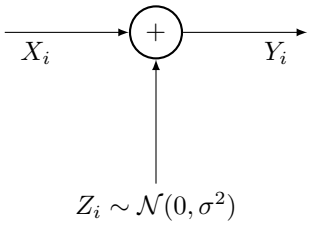


Figure 5: AWGN Channel

References

[1] T. Richardson and R. Urbanke, "Modern Coding Theory", Cambridge 2003.

[2] I. Djurdjevic, Jun Xu, K. Abdel-Ghaffar and Shu Lin, "A class of low-density parity-check codes constructed based on Reed-Solomon codes with two information symbols", IEEE Communication Letters, July 2003.

[3] T. Richardson, M Amin Shokrollahi and R. Urbanke,"Design of Capacity Approaching Irregular Low-Density Parity-Check Codes", Transactions On Information Theory, February 2001.

[4] Xiao-Yu Hu, Evangelos Eleftheriou and Dieter M. Arnold, "Regular and Irregular Progressive Edge-Growth Tanner Graphs", Transactions on Information Theory, January 2005.

[5] Jun Xu, Lei Chen, I. Djurdjevic, Shu Lin and K. Abdel-Ghaffar, "Construction of Regular and Irregular LDPC codes: Geometry Decomposition and Masking", Transactions on Information Theory, January 2007.

[6] David J. C. MacKay, "Good Error-Correcting Codes Based on Very Sparse Matrices", Transactions on Information Theory, March 1999.

[7] Richard E. Blahut, "Algebraic Codes for Data Transmission", Cambridge University Press 2003.

[8] F.J. MacWilliams and N.J.A. Sloane. "Theory of Error-Correcting Codes", North-Holland Publishing Company 1977.