Lecture Notes 6: Empirical Behavior of LDPC ensembles on BEC

Lecturer: Rüdiger Urbanke *Scribe: Vahid Aref*

1 Introduction

In the last lecture we learned how to marginalize a multivariate function by employing message passing rules. We saw that on trees, message passing starts at the leaf nodes and a node that has received messages from all its children processes the messages and forwards the result to its parent node.

For the case of binary input memoryless channels, we studied the message passing rules for bit-wise MAP decoding of a parity check code and saw that if the factor graph of a code is a tree, the sum-product solution is equal to the MAP decoding solution.

Let $\mu = (\mu(1), \mu(-1))$ denote the message emanated from nodes of a factor graph. For the leaf node x_i which receive the channel observations, y_i, we let

$$\mu_i(+1) = \mathbb{P}_{Y_i|X_i}(y_i \mid x_i = 1), \quad \mu_i(-1) = \mathbb{P}_{Y_i|X_i}(y_i \mid x_i = -1). \tag{1}$$

Since $\mu(+1) + \mu(-1) = \mathbb{P}_{Y_i|X_i}(y_i \mid x_i = 1) + \mathbb{P}_{Y_i|X_i}(y_i \mid x_i = -1) = \mathbb{P}(y_i)$ is constant for the fixed observation y_i, we can consider the likelihood ratio $r_i = \frac{\mu(x_i=1)}{\mu(x_i=-1)}$ as the message of x_i instead of the pair of $(\mu_i(1), \mu_i(-1))$. In chapter 4 of [1], it is proven that for the binary memoryless channel, likelihood ratios constitute a *sufficient statistic* with respect to decoding. This means that an optimal decoder can be based on the likelihood ratio of channel observation instead of the channel observation itself.

To make the updating equations simpler, we defined $l_i = \log(r_i)$ which is called log-likelihood ratio (LLR) and showed that the message passing rule at variable node v is

$$l_i = \sum_{k \in \mathcal{N}(v) \setminus i} l_k, \tag{2}$$

where $\mathcal{N}(v)$ is the set of (check) nodes connected to v and l_i is the outgoing message to node i (see Fig. 1). The message passing rule at check node c is

$$l_i = 2 \tanh^{-1}\Big(\prod_{j \in \mathcal{N}(c) \setminus i} \tanh(\frac{l_j}{2}) \Big), \tag{3}$$

where $\mathcal{N}(c)$ is the set of (variable) nodes connected to c and l_i is the outgoing message to node i.

The message passing algorithm can efficiently perform MAP decoding for the codes whose corresponding factor graph is a tree. Unfortunately, the class of codes that admit a tree-like factor graph is not powerful enough to perform well. The reason is that they contain many low-weight codewords and, hence, have a large probability of error. Thus, we focus our efforts on the performance of message passing algorithms for codes with cycles.

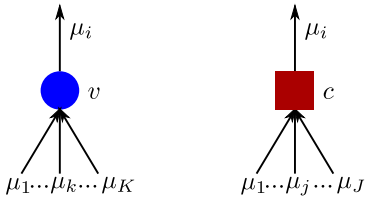

Figure 1: A variable node with $K+1$ neighbors and a check node with $J+1$ neighbors.

In this lecture, we study the error performance of regular LDPC ensemble and show that as the length of codewords increases, the size of the cycles in the corresponding factor graph grows and, therefore, the factor graph looks like a tree.

For this purpose, we will study the performance of regular LDPC codes in binary erasure channel (BEC) and finally obtain a recursion on updating the probability density function of messages which is called *density evolution*.

2 Regular LDPC ensemble on BEC

Binary erasure channel is a special class of binary memoryless symmetric channels. As depicted in Fig. 2, the transmitted bit is either correctly received at the channel output with probability $1-\epsilon$ or erased by the channel with probability ϵ and thus, nothing received at the channel output. The erased bits are denoted by ?. For example, if $x=1$ is transmitted in the BEC, then the set of possible channel observation is $\{1, ?\}$.

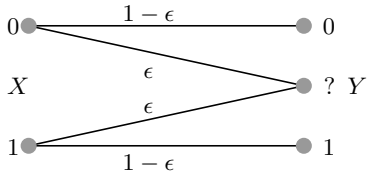

Figure 2: Binary erasure channel with parameter ϵ.

The LLR of channel observations is

$$l = \log\left(\frac{\mathbb{P}_{Y|X}(y \mid x=1)}{\mathbb{P}_{Y|X}(y \mid x=-1)}\right) = \begin{cases} \log(\frac{1-\epsilon}{0}) = \infty & y = 1, \\ \log(\frac{\epsilon}{\epsilon}) = 0 & y = ?, \\ \log(\frac{0}{1-\epsilon}) = -\infty & y = -1. \end{cases}$$

Therefore the possible values for LLR are $\{\pm\infty, 0\}$. According to (2), the outgoing message from a variable node is $+\infty$ (or $-\infty$) if at least the incoming message from one of its neighbors is $+\infty$ (or $-\infty$), otherwise it is equal to 0. Note that it is not possible that a variable node receives both $+\infty$ and $-\infty$ simultaneously.

Since $\tanh(l/2) \in \{\pm 1, 0\}$, the updating rule of check nodes, according to (3), simplifies (use $\tan(l/2) = \text{sign}(l)$) to the following equation,

$$\text{sign}(l_i) = \prod_{j \in \mathcal{N}(c) \setminus i} \text{sign}(l_j). \tag{4}$$

On BEC, knowing the sign of messages is sufficient to compute outgoing messages, thus we can assume that the set of messages is $\{\pm 1, 0\}$ instead of $\{\pm\infty, 0\}$.

2.1 Scheduling

If the Tanner graph is a tree, then message-passing clearly starts from the leaf nodes and messages propagate through the graph until they reaches to the root node; However, as we mentioned before, the cycle free parity codes do not perform well, hence we have to design linear codes whose Tanner graph has cycles.

In Factor graph with cycles, it is not clear in which node we should start the algorithm. Thus, we need a scheduling in order to update messages. A naive scheduling which is convenient for analysis of belief propagation is flooding algorithm. It means that at each step every node updates its belief (or message) according to the beliefs of its neighbors. This scheduling is usually called *parallel scheduling*.

2.2 (l, r) Regular LDPC Ensemble

In the first lecture, the Gallager's (l, r)-regular LDPC ensemble was introduced. To review shortly, consider the bipartite graph with two separate sets of nodes, variable nodes and check nodes. Each variable node is connected randomly and uniformly to l check nodes such that the degree of each check node is eventually equal to r. If there exists n variable nodes in the graph, it is possible to construct $(nl)!$ graphs by this way.

To analyze the (l, r)-regular LDPC code over BEC(ϵ), we pick a code, $\mathcal{C}$, uniformly from the ensemble of graphs and run the message passing algorithm. For a given code $\mathcal{C}$ and channel parameter ϵ, let $\mathbb{P}_b^{BP}(\mathcal{C}, X, \epsilon, L)$ denote the average bit error probability of message passing decoder for codeword X at iteration L. We will study the behavior of $\mathbb{P}_b^{BP}(\mathcal{C}, X, \epsilon, L)$ in terms of ϵ and L as a measure of performance of the code $\mathcal{C}$.

3 Two Basic Simplifications

The two following statements simplify the analysis of LDPC ensembles on BMS channels:

3.1 Restriction To The All-One Codeword

Since the distribution of LLRs in BEC is symmetric, the performance is *independent* of the transmitted codeword and is only a function of the erasure pattern: at any iteration the set of known variable nodes is only a function of the set of known messages but independent of their values. The equivalent statement is true for the set of known check nodes. It means that

$$\mathbb{P}_b^{BP}(\mathcal{C}, X, \epsilon, L) = \frac{1}{|\mathcal{C}|} \sum_{X' \in \mathcal{C}} \mathbb{P}_b^{BP}\left(\mathcal{C}, X', \epsilon, L\right) = \mathbb{P}_b^{BP}(\mathcal{C}, \epsilon, L). \tag{5}$$

3.2 Concentration

The second major simplification stems from the fact that, rather than analyzing individual codes, it suffices to assess the ensemble average performance. This is true, since, as the

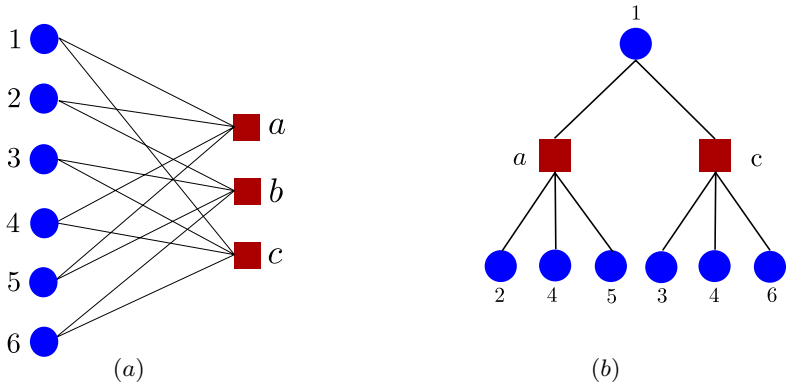

Figure 3: (a) The Tanner graph of a $(2,4)$-regular LDPC code with 6 variable nodes; (b) The corresponding computation graph of node 1 for the first iteration.

theorem 3.30 of [1] asserts, the individual behavior of elements of an ensemble is with high probability close to the ensemble average, i.e.

$$\lim_{n\to\infty} \mathbb{E}_{\mathcal{C}(n)}\{\mathbb{P}_b^{BP}\left(\mathcal{C}(n),\epsilon,L\right)\} \tag{6}$$

We therefore focus our effort on the design and analysis of the behavior of average performance of ensembles.

4 Computation Graph

Message passing takes place on the local neighborhood of a node. At each iteration, variable nodes send their beliefs along their edges toward check nodes and, then, the check nodes compute the outgoing message for each of their edges according to the beliefs of incoming edges and send it back to the variable nodes. Afterwards, each variable node updates the outgoing messages along its edges according to beliefs returned back on its edges.

After L iterations, the belief of a variable node depends on its initial belief and the beliefs of all the nodes placed in distance less than $2L$. The graph consisting of these nodes is called the computation graph of that variable node with height L. For example, the factor graph of a $(2,4,6)$-regular LDPC code is shown in Fig. 3(a) and the computation graph of node 1 with height 1 is also depicted in Fig. 3(b).

If a computation graph is tree, then no nodes is used more than once in the graph. Therefore the incoming messages of each node are independent; However, by increasing the number of iterations, the number of nodes in a computation graph grows exponentially and in some step, a node has to be used again.

The following theorem states that computation graphs in large enough LDPC codes are tree-like.

Theorem 1. *Let T denote the computation graph of a variable node with height L in (l,r,n)-regular LDPC ensemble. If L is kept fixed,*

$$\lim_{n\to\infty} \mathbb{P}\left(T \textit{ is a tree }\right) = 1. \tag{7}$$

Proof. Sketch of the proof: We pick an arbitrary variable node as root. To construct a tree with height L, at each step a new check or variable node must be added. By picking each check node, we can not pick $r-1$ remaining sockets of the chosen check node in the next steps. Similarly, by picking each variable node, we have to remove $l-1$ remaining sockets from all remaining sockets of variable nodes.

Assume that e_v variable nodes and e_c check nodes are already picked and the code totally has nl sockets (at each side). By adding a new check node, the probability that the graph remains tree is

$$\mathbb{P}\left(T \text{ is a tree} \mid e_v, e_c\right) = \frac{nl - e_c(r-1)}{nl}. \tag{8}$$

And similarly, if we add a variable node,

$$\mathbb{P}\left(T \text{ is a tree} \mid e_v, e_c\right) = \frac{nl - e_v(l-1)}{nl}. \tag{9}$$

If $l \leq r$, we can bound the above equations as follows:

$$\mathbb{P}\left(T \text{ is a tree} \mid e_v, e_c\right) \geq \frac{nl - e(r-1)}{nl}, \tag{10}$$

where $e = e_c + e_v$ is the total number of edges in the new tree. Hence, if the complete computation graph has E edges,

$$\mathbb{P}\left(T \text{ is a tree}\right) \geq \prod_{e=0}^{E} 1 - \frac{e(r-1)}{nl}. \tag{11}$$

If n tends to infinity and $E \ll n$, then $1 - \frac{e(r-1)}{nl} \approx (1 - \frac{(r-1)}{nl})^e \approx \exp(-\frac{(r-1)}{nl}e)$. Thus,

$$\mathbb{P}\left(T \text{ is a tree}\right) \geq \prod_{e=0}^{E} \exp(-\frac{(r-1)}{nl}e) = \exp(-\frac{(r-1)}{nl}\sum_{e=0}^{E} e) \approx \exp(-\frac{(r-1)}{2nl}E^2). \tag{12}$$

The number of edges in computation graph is roughly about $[(l-1)(r-1)]^L$. Therefore, in order to $\lim_{n\to\infty} \mathbb{P}\left(T \text{ is a tree}\right) = 1$,

$$L = o(\frac{\log(n)}{2\log((l-1)(r-1))}). \tag{13}$$

For the fixed L, the equation (13) is fulfilled as n tends to ∞ and, thus, the desired result is concluded. □

Hence, the above result implies that for the fixed L and as n grows large, the error probability is equal to that observed on a tree.

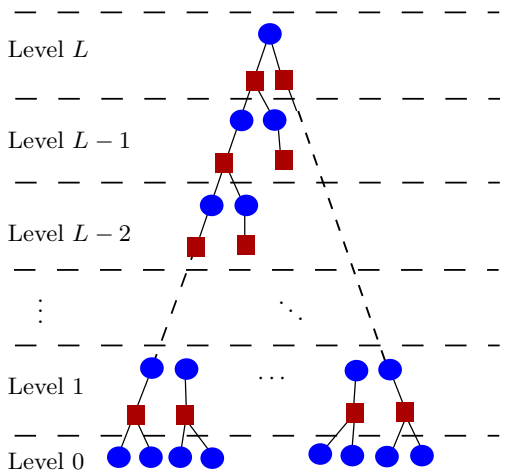

Figure 4: A computation graph of $(2,3)$-regular LDPC code with height L. The graph is split to $L+1$ levels.

5 Density Evolution

Consider a binary erasure channel with probability of erasure ϵ. We knew that computation graphs look like trees as n tends to infinity. Therefore, the incoming messages to each node of these graphs are independent.

Consider a computation graph T with height L. We divide this computation graph to $L+1$ levels, from 0 to L. Level 0 contains the leaf nodes and the 1st level contains the parent check nodes and the grandparent variable nodes of the leaf nodes (Fig. 4).

Every variable node at the i-th level is the root of a computation tree with height i. However, its root has degree $l-1$. Let x_i denote the outgoing message emitted by variable nodes in the i-th level. It is equal to either `?` with probability x_i or a known value (± 1) with probability $1-x_i$.

At level $i+1$, each variable node is connected to $l-1$ check nodes and each check node is connected to $r-1$ variable nodes of i-th level. The outgoing message of each check node is erasure message, if at least one of its incoming messages is `?`. Since x_i are independent, then the probability that a check node at level $i+1$ sends erasure message is equal to $1-(1-x_i)^{r-1}$. The outgoing message from a variable node of $i+1$-th level, i.e. x_{i+1}, is erasure message if its initial message is erasure message and all of its children (check nodes) at level $i+1$ also send erasure messages. Hence,

$$x_{i+1} = \epsilon(1-(1-x_i)^{r-1})^{l-1} = f(\epsilon, x_i). \tag{14}$$

By definition, the outgoing message at level 0 is $x_0 = \epsilon$. Therefore, the erasure probability of the root of T which is connected to l check nodes of level L is

$$\lim_{n\to\infty} \mathbb{P}_b^{BP}(l,r,n,\epsilon,L) = \epsilon(1-(1-x_{L-1})^{r-1})^l = F(\epsilon, x_{L-1}). \tag{15}$$

References

[1] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge University Press, 2008.