Simplification of the Belief propagation algorithm

Tomáš Filler

November 17, 2009

Let $C = \{x \in GF(2)^n | \mathbb{H}x^T = 0\}$ be a linear $[n, k]$ code represented by a parity-check matrix $\mathbb{H}$. Given the received word y, we observed that the Belief Propagation (BP) algorithm (as described in Figure 1) is an efficient way of calculating bitwise Maximum Likelihood (ML) estimate of individual bits x_i, $i \in \{1, \ldots, n\}$,

$$\hat{x}_i = \arg \max_{x_i \in \{0,1\}} P(x_i|y) = \arg \max_{x_i \in \{0,1\}} \sum_{\substack{x_j \\ j \neq i}} \prod_{l=1}^{n} P(y_l|x_l) \mathbb{1}_{\{x \in C\}}. \tag{1}$$

This algorithm calculates $P(x_i|y)$ up to a multiplicative constant exactly if and only if the Tanner graph of the code C is a tree (graph does not contain cycles). In practice, we use this algorithm to approximate (1) even if the graph contains cycles. When the number of cycles is small, or the graph does not contain short cycles, the results obtained from this approximation are very good. This is exactly the case of LDPC codes, because due to the sparseness of the matrix $\mathbb{H}$, the Tanner graph usually do not contain short cycles. This justifies the use of this algorithm for the decoding of LDPC codes.

By using the BP algorithm, we drastically simplified the evaluation of the bitwise ML decoding rule (1). Unfortunately, the complexity of the check-node update rule,

$$M(x) = \sum_{x_1, \ldots, x_d} \mathbb{1}_{\{x_1 + \cdots + x_d = x\}} \prod_{i=1}^{d} m_i(x_i), \tag{2}$$

is still exponential in the degree of the check node. If the node is of degree $d + 1$, then the sum (2) contains 2^d elements. In this handout, we simplify the update rules so that only one number needs to be exchanged between the nodes and the check-node update rule will be of a linear complexity (we will need roughly d instead of 2^d operations).

The main idea is to calculate the log-likelihood ratio $l = \log(P(x_i = 0|y)/P(x_i = 1|y))$ instead of $P(x_i = 0|y)$ and $P(x_i = 1|y)$ separately. We start with (1) and derive the new final bitwise ML estimate rule (see Figure 2). By calculating the log-likelihood ratio, we obtain the rule that $x_i = 0$ if

$$\log \frac{P(y_i|x_i = 0) \prod_{j=1}^{d} M_j(0)}{P(y_i|x_i = 1) \prod_{j=1}^{d} M_j(1)} = \log \frac{P(y_i|x_i = 0)}{P(y_i|x_i = 1)} + \sum_{j=1}^{d} \underbrace{\log \frac{M_j(0)}{M_j(1)}}_{=L_j} = \log \frac{P(y_i|x_i = 0)}{P(y_i|x_i = 1)} + \sum_{j=1}^{d} L_j > 0$$

and $x_i = 1$ otherwise. Here, the original check-to-variable message $M = (M(0), M(1))$ is compressed to a single number $L = \log \frac{M(0)}{M(1)}$. Let $m_i = (m_i(0), m_i(1))$ be the incoming messages as shown in Figure 1 and define $r_i = m_i(0)/m_i(1)$ and $l_i = \log r_i$. The number l_i will represent a new variable-to-check message. Now, we need to derive new update rules, which will be used to calculate L in terms of l_is.

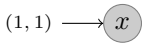
First, observe that for $d = 2$ equation (2) has the following form

$$\begin{aligned} M(0) &= m_1(0)m_2(0) + m_1(1)m_2(1) \\ M(1) &= m_1(0)m_2(1) + m_1(1)m_2(0), \end{aligned}$$

whereas for $d = 3$ we have

$$\begin{aligned} M(0) &= m_1(0)m_2(0)m_3(0) + m_1(1)m_2(1)m_3(0) + m_1(1)m_2(0)m_3(1) + m_1(0)m_2(1)m_3(1) \\ M(1) &= m_1(1)m_2(0)m_3(0) + m_1(0)m_2(1)m_3(0) + m_1(0)m_2(0)m_3(1) + m_1(1)m_2(1)m_3(1). \end{aligned}$$

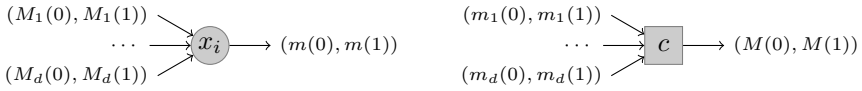
Initialization:



set all check-to-variable messages to $(1, 1)$

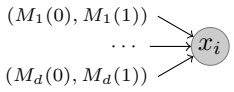
Node processing update rules:

Update all variable-to-check and then all check-to-variable messages.



$m(x) = P(y_i|x) \prod_{j=1}^{d} M_j(x)$ $\qquad$ $M(x) = \sum_{x_1,\ldots,x_d} \mathbb{1}_{\{x_1+\cdots+x_d=x\}} \prod_{i=1}^{d} m_i(x_i)$

Bitwise ML estimate of bit x_i:



$\hat{x}_i = \arg\max_{x\in\{0,1\}} P(y_i|x) \prod_{j=1}^{d} M_j(x)$

Figure 1: Update equations for the original formulation of the BP algorithm. The node-processing step (both update rules) is repeated as many times as needed. In practice, the number of iterations is fixed in advance.

In general, $M(0)$ always contains $\prod_{i=1}^{d} m_i(0)$ and all other products that contain an even number of ones in the argument. In the rest, we show how to derive the result for $d = 3$ and comment all the steps that are needed for the extension to an arbitrary $d \in \mathbb{N}$. Next, we take the ratio of $M(0)/M(1)$ and multiply it with 1 written in a special form,

$$R = \frac{\frac{1}{m_1(1)m_2(1)m_3(1)}}{\frac{1}{m_1(1)m_2(1)m_3(1)}} \frac{M(0)}{M(1)} = \frac{r_1r_2r_3 + r_1r_2 + r_1r_3 + r_2r_3}{r_1 + r_2 + r_3 + 1},$$

where $r_i = \frac{m_i(0)}{m_i(1)}$. For general d, we divide both terms with $\prod_{i=1}^{d} m_i(1)$ and thus the first term in the numerator is always of the form $\prod_{i=1}^{d} r_i$ since $M(0)$ always contains $\prod_{i=1}^{d} m_i(0)$. It is left to the reader to verify that the numerator and the denominator can be written as

$$r_1r_2r_3 + r_1r_2 + r_1r_3 + r_2r_3 = \frac{1}{2}\Big(\prod_{i=1}^{3}(r_i + 1) + \prod_{i=1}^{3}(r_i - 1)\Big)$$

$$r_1 + r_2 + r_3 + 1 = \frac{1}{2}\Big(\prod_{i=1}^{3}(r_i + 1) - \prod_{i=1}^{3}(r_i - 1)\Big).$$

This approach is a general one because $\prod_{i=1}^{d}(r_i + 1)$ evaluates to a sum of all possible products of r_is, whereas $\prod_{i=1}^{d}(r_i - 1)$ contains the same terms with a alternatig sign exactly as we need. We represent the result in the form

$$R = \frac{1 + \prod_{i=1}^{3} \frac{r_i-1}{r_i+1}}{1 - \prod_{i=1}^{3} \frac{r_i-1}{r_i+1}}.$$

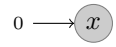
Let $f(x) = (1 + x)/(1 - x) = y$, then by calculating $f(x) - 1$ and $f(x) + 1$, we observe that the inverse mapping is $f^{-1}(y) = (y - 1)/(y + 1)$. This allows us to write

$$f^{-1}(R) = \frac{R - 1}{R + 1} = \prod_{i=1}^{3} \frac{r_i - 1}{r_i + 1}.$$

We finish the derivation by taking the logarithm, $L = \log R$ and $l = \log r$, thus $r = e^l$ and $R = e^L$. By substituting all the values and by recognizing $\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$, we conclude that

$$\frac{R - 1}{R + 1} = \frac{e^L - 1}{e^L + 1} = \tanh(L/2) = \prod_{i=1}^{3} \frac{r_i - 1}{r_i + 1} = \prod_{i=1}^{3} \frac{e^{l_i} - 1}{e^{l_i} + 1} = \prod_{i=1}^{3} \tanh(l_i/2)$$

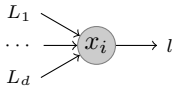
Initialization:

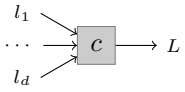


set all check-to-variable messages to 0

Node processing update rules:

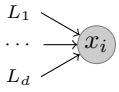
Update all variable-to-check and then all check-to-variable messages.





$$l = \ln \tfrac{P(y_i|0)}{P(y_i|1)} + \textstyle\sum_{j=1}^{d} L_j \qquad\qquad L = 2\tanh^{-1}\big(\textstyle\prod_{i=1}^{d} \tanh(l_i/2)\big)$$

Bitwise ML estimate of bit x_i:



$$\hat{x}_i = \begin{cases} 0 & \text{if } \ln \tfrac{P(y_i|0)}{P(y_i|1)} + \sum_{j=1}^{d} L_j > 0 \\ 1 & \text{otherwise} \end{cases}$$

Figure 2: Update equations of the simplified BP algorithm. The mode-processing step (both update rules) is repeated as many times as needed. In practice, the number of iterations is fixed in advance.

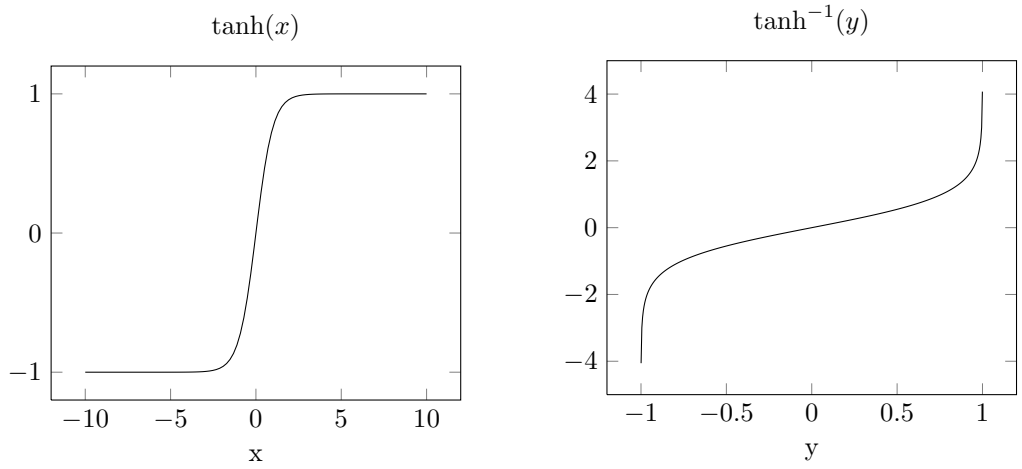


Figure 3: Plots of the hyperbolic tangent and its inverse.

and thus

$$L = 2\tanh^{-1}\Big(\prod_{i=1}^{3} \tanh(l_i/2)\Big).$$

Since we are working with the log-likelihoods, the variable-node update rule simplifies to

$$l = \log \frac{M(0)}{M(1)} = \log \frac{P(y_i|x_i=0)\prod_{j=1}^{d} M_j(0)}{P(y_i|x_i=1)\prod_{j=1}^{d} M_j(1)} = \log \frac{P(y_i|x_i=0)}{P(y_i|x_i=1)} + \sum_{j=1}^{d} \log \frac{M_j(0)}{M_j(1)} = \log \frac{P(y_i|x_i=0)}{P(y_i|x_i=1)} + \sum_{j=1}^{d} L_j,$$

and the initialization message changes to $\log \frac{1}{1} = 0$. Figure 2 summarizes the derived update rules.