Lecture 13
Message Passing Algorithms

13.1. Introduction

In this lecture, we will focus on the belief propagation algorithm, which belongs to the class of message passing algorithms. Message passing algorithms are iterative decoding algorithms based on the code's Tanner graph. In each iteration, messages are exchanged between the variable nodes and the check nodes along the edges. Each iteration starts with check nodes processing the incoming messages according to some processing rule and transmitting the output , then proceeds with variable nodes processing the incoming messages according to some other processing rule and transmitting the output. Note that a message sent along an edge does not depend on the message received on that same edge in the previous half iteration. That is, only extrinsic information is passed. This property is common to good message passing algorithms.

13.2. Belief Propagation

In belief propagation, messages sent along edges are conditional densities on the bit value corresponding to the variable node that the edge is incident to. Since these densities can assume non-zero values only at points $x = 0$ and $x = 1$ (call these values p_0 and p_1 respectively), they can be represented by a log-likelihood ratio, $log\frac{p_0}{p_1}$. It is assumed that each message is conditionally independent of all the others, i.e. the random variables on which different messages depend are independent. This assumption would hold true in the l^{th} iteration of the algorithm if all nodes appearing in any computation graph of depth $2l$ are distinct , i.e. if the code's Tanner graph has girth at least $2l$. In practice, dependence does not affect performance drastically: Suppose that the computation graph closes upon itself after m iterations. In this case the output of the $(m-1)^{st}$ iteration can be thought of as a new sequence of observations which incorporates less uncertainty than the original sequence.

Consider the message passed from variable node v to check node c in the l^{th} iteration. This is a probability distribution on the bit that v is associated with, given all the observations of its neighbors except c. Under the cycle-free assumption for l iterations, the computation graph for node v will be a tree of depth $2l$ (Fig.13.1).

Lecture by Amin Shokrollahi
Scribes by Eren Sasoglu

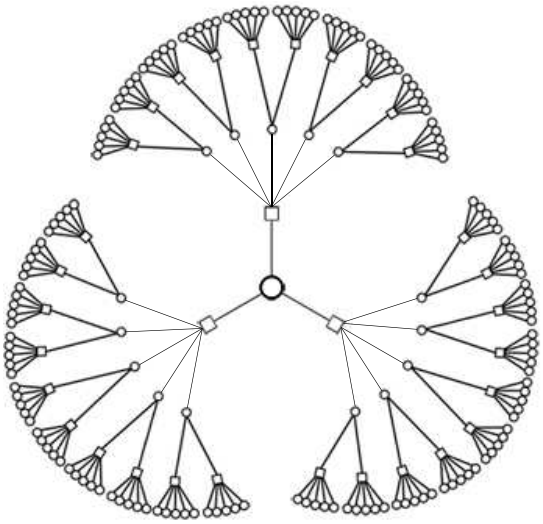

Figure 13.1: Computation graph of depth 4. All nodes appearing in the graph must be distinct in order for the independence assumption to hold.

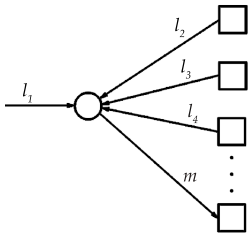

Figure 13.2: Processing at the variable nodes

13.2.1. Processing at the Variable Nodes

Consider Fig.13.2 where l_1 corresponds to the channel observation (in log-likelihood form). Check node $i = 2 \dots d$ sends the variable node a log-likelihood ratio on the value x of the variable node, conditioned on its observations.

$$l_i = \log \frac{Pr[x = 0|y_i]}{Pr[x = 1|y_i]} \qquad i = 1 \dots d$$

l_i's can be thought of as metrics on noisy observations of x. Moreover, these observations are assumed to be independent when conditioned on x. The variable node computes the log-likelihood ratio m representing the density of its value x, given all the conditionally independent noisy observations y_i. i.e.

$$m = \log \frac{Pr[x = 0|y_1 \dots y_d]}{Pr[x = 1|y_1 \dots y_d]}$$

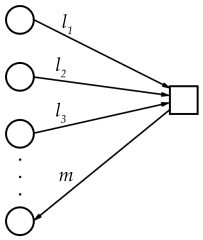

Figure 13.3: Processing at the check nodes

We have

$$\begin{aligned} Pr[x=0|y_1 \dots y_d] &= \frac{Pr[x=0, y_1 \dots y_d]}{Pr[y_1 \dots y_d]} \\ &= \frac{Pr[x=0] \cdot Pr[y_1 \dots y_d|x=0]}{Pr[y_1 \dots y_d]} \\ &= \frac{Pr[x=0] \cdot \prod_{i=1}^{d} Pr[y_i|x=0]}{Pr[y_1 \dots y_d]} \end{aligned}$$

and similarly,

$$Pr[x=1|y_1 \dots y_d] = \frac{Pr[x=1] \cdot \prod_{i=1}^{d} Pr[y_i|x=1]}{Pr[y_1 \dots y_d]}$$

Assuming $Pr[x=0] = Pr[x=1] = 1/2$ we have $\frac{Pr[y_i|x=0]}{Pr[y_i|x=1]} = \frac{Pr[x=0|y_i]}{Pr[x=1|y_i]}$ and therefore

$$\begin{aligned} \frac{Pr[x=0|y_1 \dots y_d]}{Pr[x=1|y_1 \dots y_d]} &= \prod_{i=1}^{d} \frac{Pr[x=0|y_i]}{Pr[x=1|y_i]} \\ m = \log \frac{Pr[x=0|y_1 \dots y_d]}{Pr[x=1|y_1 \dots y_d]} &= \sum_{i=1}^{d} l_i \end{aligned} \tag{13.1}$$

Therefore, when the messages are in log-likelihood form, the optimal probabilistic processing rule at the variable nodes is the summation of all incoming messages.

13.2.2. Processing at the Check Nodes

Consider Fig.13.3. Let y_i denote the noisy observation on the i^{th} summand in the parity check equation and define

$$p_i(0) := Pr[x_i = 0|y_i], \qquad p_i(1) := Pr[x_i = 1|y_i] \qquad i = 1 \dots d$$

Let S denote the event that the parity-check equation is satisfied. Then the outgoing message will be $m = log\frac{p(0)}{p(1)}$ where

$$p(k) := Pr[x_{d+1} = k|S, y_1 \dots y_d] = Pr[x_1 + \dots + x_d = k|y_1 \dots y_d] \qquad k = 0, 1$$

Since p is a density on the $mod2$ sum of independent random variables, it is the cyclic convolution of p_i's. Therefore letting $\mathscr{F}$ denote the Fourier transform, we have

$$\mathscr{F}\{p\} = \prod_{i=1}^{d} \mathscr{F}\{p_i\}$$

Fourier transform of a function f over $\mathbb{Z}_2$ is also defined over $\mathbb{Z}_2$ and is given by

$$\mathscr{F}\{f\}(0) = f(0) + f(1)$$
$$\mathscr{F}\{f\}(1) = f(0) - f(1)$$

Since we have probability distributions as our functions,

$$\begin{aligned} \mathscr{F}\{p\}(0) &= 1 \\ \mathscr{F}\{p\}(1) &= p(0) - p(1) = \prod_{i=1}^{d} (p_i(0) - p_i(1)) \end{aligned} \tag{13.2}$$

Recall that the received messages are in the log-likelihood form $l_i = \log \frac{p_i(0)}{p_i(1)}$. By direct computation we get

$$p_i(0) - p_i(1) = \frac{e^{l_i} - 1}{e^{l_i} + 1} = \tanh \frac{l_i}{2} \tag{13.3}$$

and

$$p(0) - p(1) = \frac{e^{m} - 1}{e^{m} + 1} = \tanh \frac{m}{2} \tag{13.4}$$

Substituting 13.3 and 13.4 in 13.2 we have

$$\begin{aligned} \tanh \frac{m}{2} &= \prod_{i=1}^{d} \tanh \frac{l_i}{2} \\ m &= \log \frac{1 + \prod_{i=1}^{d} \tanh \frac{l_i}{2}}{1 - \prod_{i=1}^{d} \tanh \frac{l_i}{2}}. \end{aligned} \tag{13.5}$$

13.3. Binary Erasure Channel

Probabilistic decoding problem reduces to a combinatorial problem in the BEC case. This is due to the fact that there is no uncertainty in a bit's value unless the received value is an erasure.

13.3.1. Variable node processing rule

Recall that the variable node processing rule is simply summation of the incoming log-likelihood ratios. Without loss of generality, it can be assumed that the channel outputs log-likelihood ratios instead of bit values. Then channel output alphabet $\mathscr{M}$ and message alphabet $\mathscr{O}$ are equal: $\mathscr{M}$=$\mathscr{O}$=$\{+\infty, 0, -\infty\}$, where the values correspond to $0, e$ and 1 respectively in the usual BEC case. At time $t = 0$, the only input that a variable node will receive will be the output $l_1 \in \mathscr{O}$ of the channel, which will be passed to the neighboring check nodes. Notice that variable node processing rule

$$m = \sum l_i$$

reduces to

$$m = \begin{cases} 0 & if \quad l_i = 0 \qquad \forall i \\ \beta & if \quad \exists \beta \in \{l_1 \dots l_d\} \qquad \beta \neq 0 \end{cases}$$

One might ask what happens to the sum when we have both $+\infty$ and $-\infty$ in the incoming message set. This in fact is an impossible situation since the messages passed under belief propagation are consistent (a $+\infty$ implies $Pr[x = 1] = 1$ and a $-\infty$ implies $Pr[x = 0] = 1$).

Observe that the above processing rule would be valid for any message alphabet $\mathscr{M}$=$\{-\beta, 0, \beta\}$. Therefore taking the channel input alphabet to be $\{1, -1\}$, the channel output alphabet to be $\mathscr{O}$=$\{1, 0\ \text{-}1\}$ with the correspondence $1 \leftrightarrow 0$, $-1 \leftrightarrow 1$, $0 \leftrightarrow e$, and the message alphabet to be $\mathscr{M}$=$\mathscr{O}$, the variable node processing rule is equivalent to sending the bit value if any of the received messages is non-zero, and sending a zero otherwise.

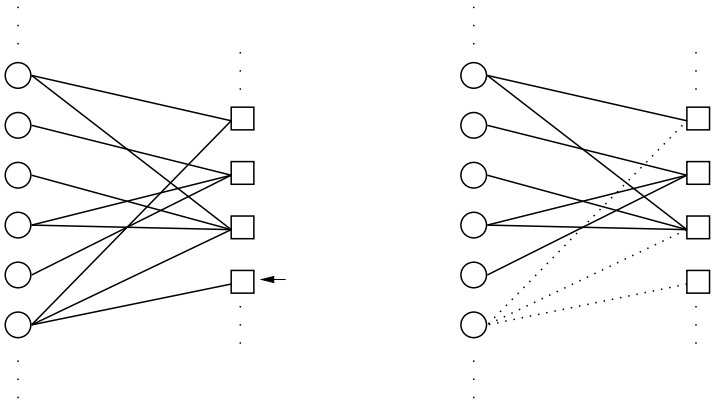

Figure 13.4: Iterative decoding for BEC. In each iteration, first find a check node of degree 1 (left), then remove all the edges emanating from the neighboring variable node (right)

13.3.2. Check node processing rule

Recall that at the check nodes the output message was determined according to $m = \log \frac{1+\prod \tanh \frac{l_i}{2}}{1-\prod tanh \frac{l_i}{2}}$. Note that

$$\begin{aligned} tanh(+\infty) &= 1 \\ tanh(0) &= 0 \\ tanh(-\infty) &= -1 \end{aligned}$$

Therefore when $\mathscr{M} = \{+\infty, 0, -\infty\}$, the above processing rule is equivalent to

$$m = \log \frac{1 + \prod_{i=1}^{d} \text{sgn}(l_i)}{1 - \prod_{i=1}^{d} \text{sgn}(l_i)}$$

which reduces to

$$m = \begin{cases} 0 & \text{if } \exists i \text{ s.t. } l_i = 0 \\ +\infty & l_i \neq 0 \ \forall i \text{ and there is an even number of } +\infty\text{'s} \\ -\infty & l_i \neq 0 \ \forall i \text{ and there is an odd number of } +\infty\text{'s} \end{cases}$$

Note that if we again take the channel input alphabet $\mathscr{I}$=$\{1, -1\}$, the parity-check constraint becomes, $\prod_i m_i = 1$. Therefore taking $\mathscr{M}$=$\mathscr{O}$=$\{1, 0, -1\}$, the above processing rule is equivalent to

$$m = \prod_i m_i$$

With the above two reductions, switching back to the representation $\mathscr{I} = \{0, 1\}$, belief propagation algorithm on the BEC is equivalent to the following decoding algorithm:
Step 1. Pass all the received (non-erasure) bit values from the variable nodes to the check nodes along the edges.
Step 2. At the check node side, store the XOR of the incoming bits and delete the edges along which bit values came.
Step 3. Find a check node of residual degree 1, and send its (stored) value to the variable node v it is connected to.
Step 4. Delete all edges emanating from v.
Step 5. Repeat Steps 3 and 4.

Fig.13.4 shows one iteration of the algorithm defined above. Clearly, we need a non-zero fraction of the check nodes to have degree 1 in each iteration of decoding in order to recover all variable nodes.