Information Theory and Applications
Academic year 2019/2020

Message-passing algorithm

Prof. Roberto Garello

Notes taken by Raffaele Paolino

Contents

1 Introduction: message passing idea

The idea of **distributed computing** is that a big problem can be divided into small tasks easier to solve. This can be realized by using nodes with limited computation power, each performing some observation and some simple computations and sharing the result with its neighbors by sending messages.

Belief propagation (also known as sum-product algorithm) is a message passing algorithm introduced by Judea Pearl in 1982 to perform distributed computing, in particular inference on graphs.

A key role in belief propagation is played by the concept of extrinsic information. As shown in fig. 1, each node computes a value Y considering all the incoming messages (in our figure $Y = A + B + C$), adds its own observed value X and sends back to each neighbour a message containing the extrinsic information, which is equal to the total sum Y minus the incoming message from that particular node. It is important to note that, in this way, the information sent back to a node N is given by the information coming from all other sources but the node N itself. For a better understanding of the idea we show some examples.

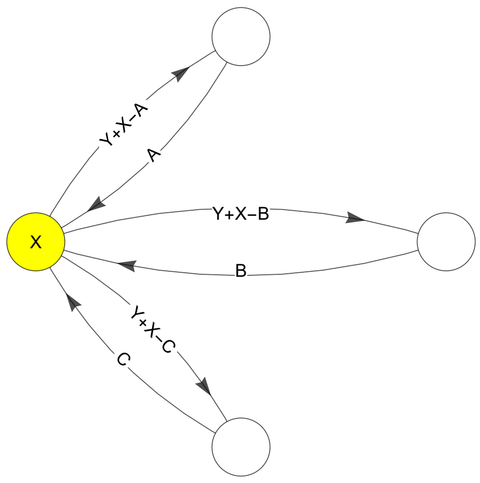
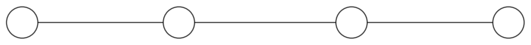


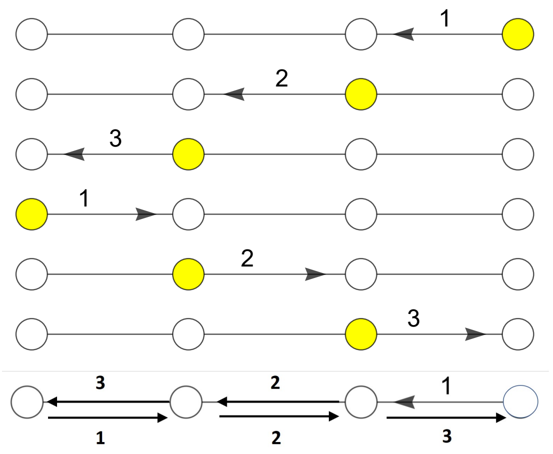
Figure 1: Visual representation of extrinsic information. Consider the highlighted node. It receives messages A, B, C from its neighbours. It sums the incoming messages, obtaining $Y = A + B + C$, adds its own value X and sends back to each neighbour a new message containing the extrinsic information, i.e. the information that that neighbour did not provide.

Example 1.1. Suppose we have a 4-nodes network as shown in fig. 2a. We want to compute the total number of nodes in a distributed way.

A simple implementation is the following: the first node activates and send to its neighbour its observed value (in this case its cardinality), i.e. the number 1. The second node receives 1 from the first node, adds its observed value 1 and send the result to the third node. The third node receives 2 from the second node, adds 1 and send the result to the last node. At the same time each node answers to the incoming message by sending to the previous node the value obtained by summing all the values received, adding its value and subtracting the value coming from its neighbour. What happens is displayed in fig. 2b.



(a) Line graph.

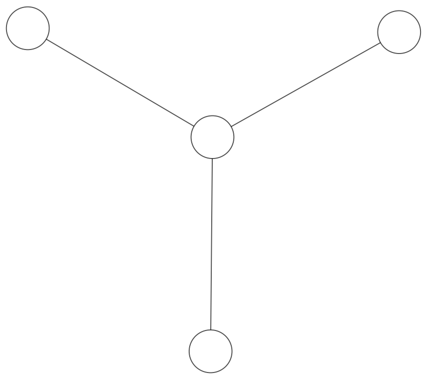


(b) Visual representation of the message passing algorithm.

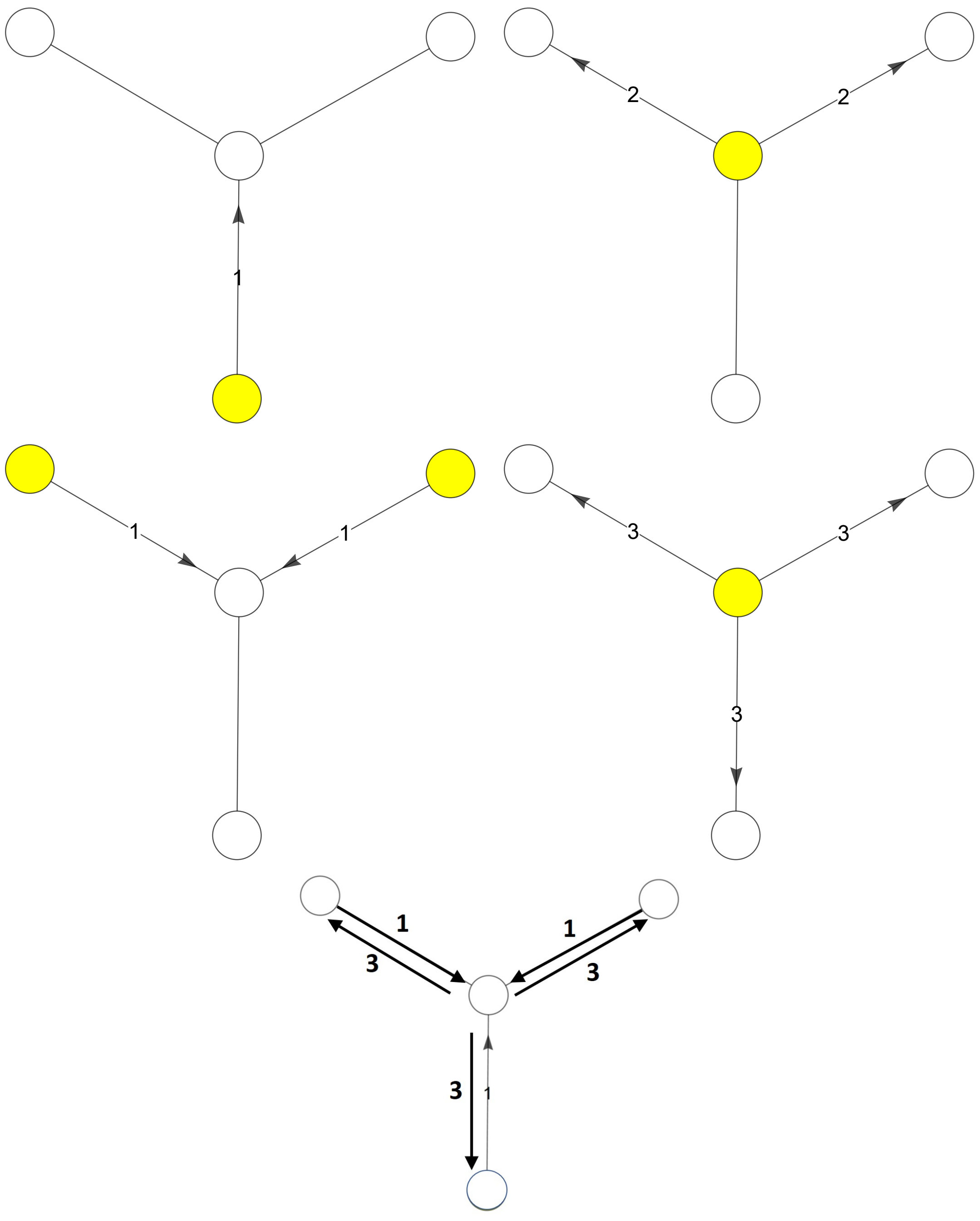
Figure 2: Message passing algorithm for a line graph. After few iterations the messages become stable.

Example 1.2. Suppose we have a 4-nodes network as shown in fig. 3a. We want to compute the total number of nodes in a distributed way.

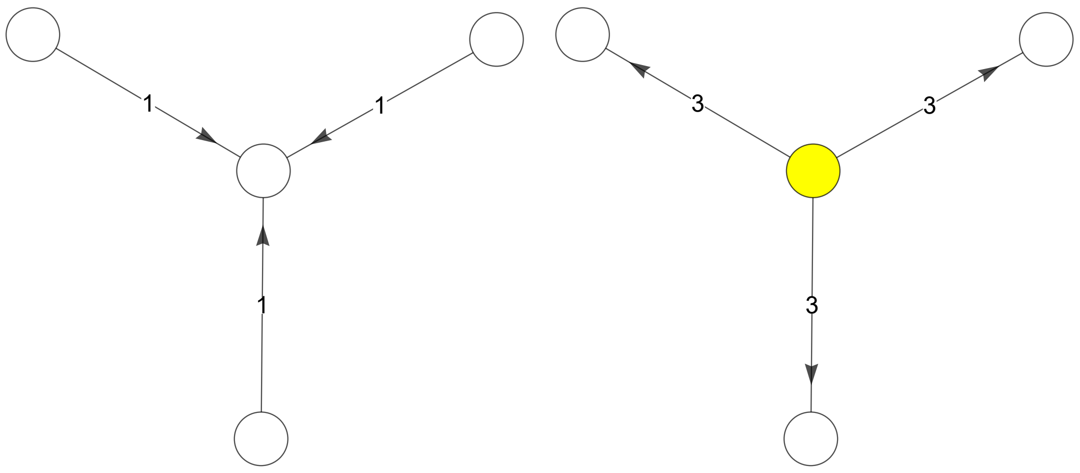
Reasoning as before, we obtain what is shown in fig. 3b



(a) Star graph.



(b) Visual representation of the message passing algorithm.



(c) Detail of the message passing algorithm. The central node receives 1 from all his neighbours, so it sends back $Y + X - 1 = 3 + 1 - 1 = 3$.

Figure 3: Message passing algorithm for a star graph. Also in this case after few iterations the messages become stable.

Example 1.3. The algorithm may fail when we consider a graph with loops such as the one shown in fig. 4.

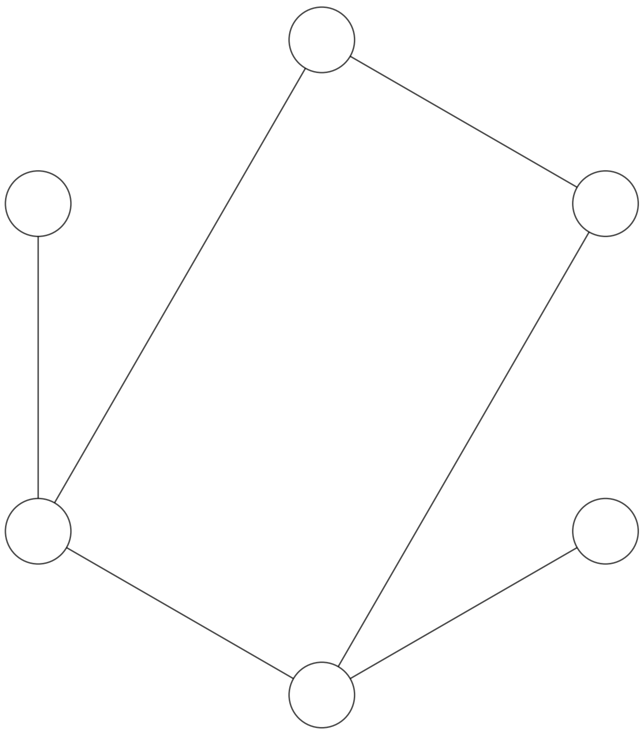


Figure 4: Example of a graph with a loop.

2 Factor graphs

These distributed algorithms based on message passing and extrinsic information idea allow to compute complex functions by working on a distributed network.
Suppose we have a scalar function

$$f : \mathbb{R}^n \to \mathbb{R} \tag{1}$$
$$\mathbf{x} \mapsto f(\mathbf{x}) \tag{2}$$

where $\mathbf{x} = (x_1, \ldots, x_j, \ldots, x_N)$, and suppose that f can be expressed as the product of some elementary function, called **factors**, each of them involving only a small subset of the entire $\mathbf{x}$, i.e.

$$f(\mathbf{x}) = \prod_{i=1}^{K} f_i(\mathbf{x}_i) \qquad \mathbf{x}_i \subseteq \mathbf{x} \tag{3}$$

Instead of computing f working on the entire vector $\mathbf{x}$ we can compute the elementary factors f_i, where each involves only a small subset of variables $\mathbf{x}_i$, and then combine their results.

In this context it is useful to introduce the concept of a **factor graph**. A factor graph $\mathcal{G} = (\mathcal{N}_v, \mathcal{N}_f, \mathcal{E})$ is a bipartite graph made by two kind of nodes (variables and factors) and edges, such that

- single variables and factors are the nodes of $\mathcal{G}$

  $$\begin{aligned} x_j &\in \mathcal{N}_v \qquad \forall j = 1, \ldots, N \\ f_i &\in \mathcal{N}_f \qquad \forall i = 1, \ldots, K \end{aligned}$$

  In the graphical representation of the factor graph, we denote the former with circles and the latter with square.

- a variable node is connected to a factor node by an edge if and only if it is one of its argument, i.e.

  $$(x_j, f_i) \in \mathcal{E} \iff x_j \in \mathbf{x}_i$$

An example of factor graph with $N = 5$ and $K = 3$ is shown in fig. 5 where we represent a function f that can be expressed as

$$f(x_1, x_2, x_3, x_4, x_5) = f_1(x_1, x_2)\, f_2(x_3, x_4)\, f_3(x_5)$$

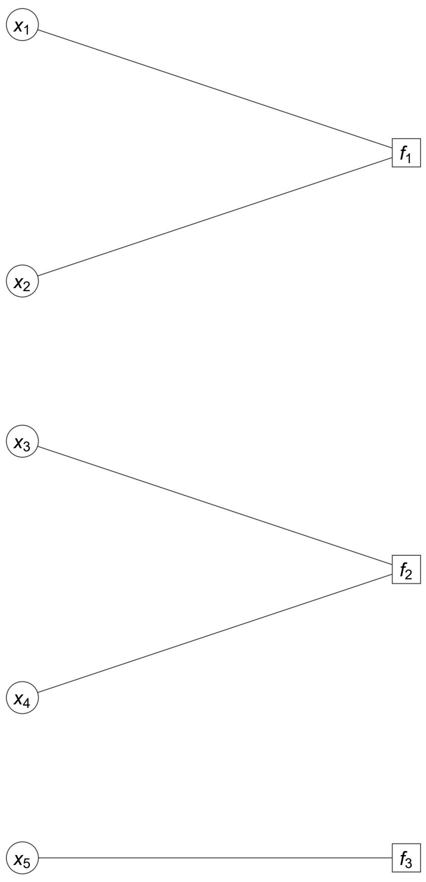


Figure 5: Example of a factor graph. Variables node are denoted with circles and factor nodes are denoted with squares. A link exists just between a factor and its arguments.

3 The Marginalization Problem

For us the variables x_j will be discrete random variables and the function f will represent their joint probability mass function. The problem we will try to face is the following: given the joint probability mass function $f(x_1, \dots, x_j, \dots, x_N)$ with x_j belonging to an alphabet $\mathcal{A}$, we are interested in computing the marginal distribution for some variable x_j, i.e., its probability mass function, that is given by:

$$\mathbb{P}[x_i = \alpha] = \sum_{\mathbf{x}:x_i=\alpha} f(\mathbf{x}) \qquad \forall \alpha \in \mathcal{A} \tag{4}$$

We present now an example of marginalization.

Example 3.1. Suppose $N = 3$ and $\mathcal{A} = \{0, 1\}$. It implies that $\mathbf{x} = (x_1, x_2, x_3)$ can have $2^3 = 8$ possible values, shown in table 1.

x_1	x_2	x_2	$f(x_1, x_2, x_3)$
0	0	0	p_1
0	0	1	p_2
0	1	0	p_3
0	1	1	p_4
1	0	0	p_5
1	0	1	p_6
1	1	0	p_7
1	1	1	p_8

Table 1: Possible values of $\mathbf{x}$ for $N = 3$, $\mathcal{A} = \{0, 1\}$.

We are interested in computing the marginal distribution for x_1. Applying eq. (4) we note that

$$\begin{aligned}\mathbb{P}[x_i = 0] &= \sum_{\mathbf{x}:x_1=0} f(\mathbf{x}) \\ &= f(0,0,0) + f(0,0,1) + f(0,1,0) + f(0,1,1) \\ &= p_1 + p_2 + p_3 + p_4\end{aligned}$$

$$\begin{aligned}\mathbb{P}[x_i = 1] &= \sum_{\mathbf{x}:x_1=1} f(\mathbf{x}) \\ &= f(1,0,0) + f(1,0,1) + f(1,1,0) + f(1,1,1) \\ &= p_5 + p_6 + p_7 + p_8\end{aligned}$$

Suppose we want to compute the marginal distribution for a given variable x_j:

$$\mathbb{P}[x_i = \alpha] = \sum_{\mathbf{x}: x_i = \alpha} f(\mathbf{x}) \qquad \forall \alpha \in \mathcal{A} \tag{5}$$

The problem is that the number of vectors to be considered in the sum can be huge. The key idea behind the application of message passing to the marginalization problem is that if $f(\mathbf{x})$ can be expressed as the product of factors, each of them involving a small subset of $\mathbf{x}$, the marginalization computation becomes much simpler because each factor involves only a small set of vectors. To better understand this idea we present a problem related to binary block codes.

4 Binary Block Codes

Let $\mathcal{A} = \{0, 1\}$ be the binary alphabet.

We define the binary sum as:

$+$	0	1
0	0	1
1	1	0

and the binary product as:

$\cdot$	0	1
0	0	0
1	0	1

We consider a k-bit information vector $\mathbf{v} \in \mathcal{A}^k$. We map it into an n-bit coded vector $\mathbf{c} \in \mathcal{A}^n$ (also called codeword), given by the linear function

$$f : \mathcal{A}^k \to \mathcal{A}^N \tag{6}$$

$$\mathbf{v} \mapsto \mathbf{c} \tag{7}$$

where

$$\mathbf{c} = \mathbf{v}\,\mathbf{G} \tag{8}$$

The binary matrix $\mathbf{G} \in \mathcal{A}^{k\times n}$ is the **generator matrix** defined as

$$\mathbf{G} = \left[\begin{array}{c|c} \mathbf{I}_k & \mathbf{R} \end{array}\right] \tag{9}$$

where $\mathbf{I}_k$ is the identity matrix of order k and $\mathbf{R}$ is the redundancy matrix whose dimension is $k \times (n-k)$.

We call **codebook** the set of all possible codewords and we denote it by $\mathcal{C}$. We note that (since the left part of the generator matrix is the identity matrix) the first part of the coded vector is always equals to the information vector, that is

$$c_i = v_i \qquad \forall i = 1, \ldots, k. \tag{10}$$

The cardinality of the set of all information vectors is 2^k and then the cardinality of the codebook $\mathcal{C}$ is 2^k too because f is a one to one map.

A key idea for factor graph representation is the **indicator function**

$$\mu : \mathcal{A}^N \to \{0, 1\} \tag{11}$$

that maps every codeword to 1 and every non codeword to 0

$$\mu(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \mathcal{C} \\ 0 & \text{if } \mathbf{x} \notin \mathcal{C} \end{cases} \tag{12}$$

We define the **parity check matrix** $\mathbf{H} \in \mathcal{A}^{n \times n-k}$ as

$$\mathbf{H} = \left[\begin{array}{c} \mathbf{R} \\ \hline \mathbf{I}_{n-k} \end{array}\right] \tag{13}$$

We note that

$$\begin{aligned} \mathbf{G}\,\mathbf{H} &= \left[\begin{array}{c|c} \mathbf{I}_k & \mathbf{R} \end{array}\right] \left[\begin{array}{c} \mathbf{R} \\ \hline \mathbf{I}_{n-k} \end{array}\right] && (14) \\ &= \mathbf{I}_k\,\mathbf{R} + \mathbf{R}\,\mathbf{I}_{n-k} && (15) \\ &= \mathbf{R} + \mathbf{R} && (16) \\ &= \mathbf{0} && (17) \end{aligned}$$

due to the property of binary sum.

Multiplying both members of (17) by an information vector $\mathbf{v}$ leads to

$$\mathbf{v}\,\mathbf{G}\,\mathbf{H} = \mathbf{c}\,\mathbf{H} = \mathbf{0} \tag{18}$$

The equation $\mathbf{c}\,\mathbf{H} = \mathbf{0}$ is the fundamental property of the parity check matrix $\mathbf{H}$. We note here that $\mathbf{c} \in \mathcal{A}^n$ and $\mathbf{H} \in \mathcal{A}^{n \times n-k}$, then $\mathbf{0} \in \mathcal{A}^{n-k}$. By using $\mathbf{H}$ definition, it is easy to show that for any vector $\mathbf{y} \in \mathcal{A}^n$ which is not a codeword we have $\mathbf{y}\,\mathbf{H} \neq \mathbf{0}$, then we can use $\mathbf{H}$ to distinguish between codewords and non-codewords.

If we consider the equation $\mathbf{c}\,\mathbf{H} = \mathbf{0}$ componentwise, we obtain $n - k$ conditions upon $\mathbf{c}$ being a codeword, i.e. a way to factorize the indicator function μ. We show this with an example.

Example 4.1. Consider a generator matrix defined as

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

The set of all possible information vectors is $\{00, 01, 10, 11\}$ so its cardinality is $2^2 = 4$. The codebook is:

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

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

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

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

so the mapping f is such that

$$\begin{aligned}00 &\mapsto 0000\\ 01 &\mapsto 0110\\ 10 &\mapsto 1011\\ 11 &\mapsto 1101\end{aligned}$$

The indicator function μ takes as input a 4-bit vector $\mathbf{x}\mathcal{A}^n$ and returns 1 if $\mathbf{x}$ is a codeword and 0 otherwise.

$\mathbf{x}$	$\mu(\mathbf{x})$
0 0 0 0	1
0 0 0 1	0
0 0 1 0	0
0 0 1 1	0
0 1 0 0	0
0 1 0 1	0
0 1 1 0	1
0 1 1 1	0
1 0 0 0	0
1 0 0 1	0
1 0 1 0	0
1 0 1 1	1
1 1 0 0	0
1 1 0 1	1
1 1 1 0	0
1 1 1 1	0

The matrix $\mathbf{H}$ is defined as

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

and we can verify that (17) holds

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

Applying (18) componentwise, it leads to

$$\begin{bmatrix} c_1 & c_2 & c_3 & c_4 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} c_1 + c_2 + c_3 \\ c_1 + c_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

The previous equation tells us that in order for $\mathbf{c}$ to be a codeword, i.e. $\mu(\mathbf{c}) = 1$, it must satisfy both these $n - k = 2$ conditions:

$$\begin{cases} c_1 + c_2 + c_3 = 0 \\ c_1 + c_4 = 0 \end{cases}$$

so the indicator function can be expressed as the product of elementary factors.

The Iverson bracket allows to convert a logical proposition into a bit, which is equal to 1 if the proposition is satisfied and 0 otherwise:

$$[P] = \begin{cases} 1 & \text{if P is true} \\ 0 & \text{if P is false} \end{cases}$$

Using Iverson bracket we can express the factorization as

$$\mu(\mathbf{c}) = [c_1 + c_2 + c_3 = 0] \cdot [c_1 + c_4 = 0]\,,$$

where $[c_1 + c_2 + c_3 = 0]$ and $[c_1 + c_4 = 0]$ are 1 only if the expressions between brackets are true. To show that this actually works we carry out all the computation for every 4-bit vector:

$\mathbf{c}$	$c_1 + c_2 + c_3$	$c_1 + c_4$	$[c_1 + c_2 + c_3 = 0] \cdot [c_1 + c_4 = 0]$	$\mu(\mathbf{x})$
0 0 0 0	0	0	1	1
0 0 0 1	0	1	0	0
0 0 1 0	1	0	0	0
0 0 1 1	1	1	0	0
0 1 0 0	1	0	0	0
0 1 0 1	1	1	0	0
0 1 1 0	0	0	1	1
0 1 1 1	0	1	0	0
1 0 0 0	1	1	0	0
1 0 0 1	1	0	0	0
1 0 1 0	0	1	0	0
1 0 1 1	0	0	1	1
1 1 0 0	0	1	0	0
1 1 0 1	0	0	1	1
1 1 1 0	1	0	0	0
1 1 1 1	1	0	0	0

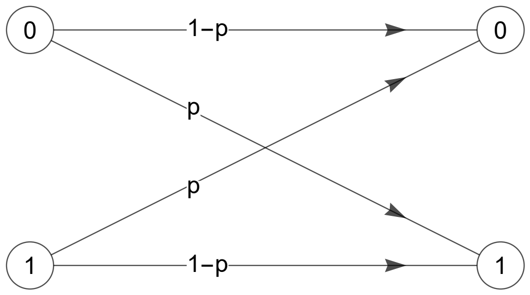


Figure 6: Binary symmetric channel: p is the probability of error, i.e. the probability that a symbol is coded into another one.

5 How to solve the marginalization problem for Binary Block Codes

In this section we show how to compute the marginal distribution for the codeword bits. We first solve it by a grute force approach, that is feasible only when K is very small. Then, we present the message passing approach based on factor graphs, that can be applied to any length.

5.1 First part: brute force approach

Consider a binary symmetric channel (BSC), as shown in fig. 6. Given $\mathbf{G}$ we know the entire codebook $\mathcal{C}$. We transmit one of the codeword $\mathbf{c} = (c_1, \dots, c_i, \dots, c_n)$ and we receive the binary vector $\mathbf{y} = (y_1, \dots, y_i, \dots, y_n)$ according to the BSC probabilities. Due to BSC errors, $\mathbf{y}$ could be any 4-bit vector, also a non-codeword.

At the receiver side we want to deduce which bits were transmitted. To do this we want to estimate the probabilities

$$\mathbb{P}\left[c_i = 0|\mathbf{y}\right] \tag{19}$$
$$\mathbb{P}\left[c_i = 1|\mathbf{y}\right] \tag{20}$$

In order to do that we compute first

$$\mathbb{P}\left[c_i = 0|y_i\right] \tag{21}$$
$$\mathbb{P}\left[c_i = 1|y_i\right] \tag{22}$$

We denote

$$\delta_i(0) \coloneqq \mathbb{P}\left[c_i = 0|y_i\right] \tag{23}$$
$$\delta_i(1) \coloneqq \mathbb{P}\left[c_i = 1|y_i\right] \tag{24}$$

We have

$$\mathbb{P}\left[\mathbf{c}|\mathbf{y}\right] = \prod_i \mathbb{P}\left[c_i|y_i\right] = \prod_i \delta_i(c_i) \tag{25}$$

and finally

$$\mathbb{P}\left[c_i = 0|\mathbf{y}\right] = \sum_{\mathbf{c}:c_i=0} \mathbb{P}\left[\mathbf{c}|\mathbf{y}\right] \tag{26}$$

$$\mathbb{P}\left[c_i = 1|\mathbf{y}\right] = \sum_{\mathbf{c}:c_i=1} \mathbb{P}\left[\mathbf{c}|\mathbf{y}\right] \tag{27}$$

In order to clarify this, we present an example.

Example 5.1. Consider $\mathbf{G}$ as in example 4.1. Suppose $\mathbf{y} = 1111$ and $p = 0.1$. We suppose that the transmitted bits are equiprobable:

$$\mathbb{P}\left[c_i = 0\right] = \mathbb{P}\left[c_i = 1\right] = \frac{1}{2}$$

then, due to the BSC properties:

$$\begin{aligned}\mathbb{P}\left[y_i = 1\right] &= \mathbb{P}\left[y_i = 1|c_i = 0\right]\mathbb{P}\left[c_i = 0\right] + \mathbb{P}\left[y_i = 1|c_i = 1\right]\mathbb{P}\left[c_i = 1\right]\\ &= p\,\frac{1}{2} + (1-p)\,\frac{1}{2}\\ &= \frac{1}{2}\end{aligned}$$

so by the Bayes theorem

$$\begin{aligned}&\mathbb{P}\left[c_i|y_i\right]\\ &= \frac{\mathbb{P}\left[y_i|c_i\right]}{\mathbb{P}\left[y_i\right]}\mathbb{P}\left[c_i\right]\\ &= \mathbb{P}\left[y_i|c_i\right]\end{aligned}$$

It follows that:

- if $y_i = 0$ we have $\delta_i(0) = 1 - p$ and $\delta_i(1) = p$
- if $y_i = 1$ we have $\delta_i(0) = p$ and $\delta_i(1) = 1 - p$

In this case, we have

$$\begin{aligned}\mathbb{P}'\left[\mathbf{c} = (0000)|\mathbf{y}\right] &= \delta_1(0)\,\delta_2(0)\,\delta_3(0)\,\delta_4(0)\\ &= p^4\\ &= 0.0001\\ \mathbb{P}'\left[\mathbf{c} = (0110)|\mathbf{y}\right] &= \delta_1(0)\,\delta_2(1)\,\delta_3(1)\,\delta_4(0)\\ &= p^2\,(1-p)^2\\ &= 0.0081\\ \mathbb{P}'\left[\mathbf{c} = (1011)|\mathbf{y}\right] &= \delta_1(1)\,\delta_2(0)\,\delta_3(1)\,\delta_4(1)\\ &= p\,(1-p)^3\end{aligned}$$

$$
\begin{aligned}
&= 0.0729 \\
\mathbb{P}'\left[\mathbf{c} = (1101)|\mathbf{y}\right] &= \delta_1(1)\,\delta_2(1)\,\delta_3(0)\,\delta_4(1) \\
&= p\,(1-p)^3 \\
&= 0.0729
\end{aligned}
$$

In order to get a set of probabilities, they have to sum up to 1, so we normalize them by dividing by the factor

$$
\begin{aligned}
g &= \sum_{\mathbf{c}\in\mathcal{C}} \mathbb{P}\left[\mathbf{c}|\mathbf{y}\right] \\
&= 0.1540
\end{aligned}
$$

and we obtain:

$$
\begin{aligned}
\mathbb{P}'\left[\mathbf{c} = (0000)|\mathbf{y}\right] &= 6.5e-4 \\
\mathbb{P}'\left[\mathbf{c} = (0110)|\mathbf{y}\right] &= 0.0526 \\
\mathbb{P}'\left[\mathbf{c} = (1011)|\mathbf{y}\right] &= 0.4734 \\
\mathbb{P}'\left[\mathbf{c} = (1101)|\mathbf{y}\right] &= 0.4734
\end{aligned}
$$

Finally we compute

$$
\begin{aligned}
\mathbb{P}\left[c_1 = 0|\mathbf{y}\right] &= \left(\mathbb{P}\left[\mathbf{c} = (0000)|\mathbf{y}\right] + \mathbb{P}\left[\mathbf{c} = (0110)|\mathbf{y}\right]\right) \\
&= 0.0532 \\
\mathbb{P}\left[c_2 = 0|\mathbf{y}\right] &= \left(\mathbb{P}\left[\mathbf{c} = (0000)|\mathbf{y}\right] + \mathbb{P}\left[\mathbf{c} = (1011)|\mathbf{y}\right]\right) \\
&= 0.4740 \\
\mathbb{P}\left[c_3 = 0|\mathbf{y}\right] &= \left(\mathbb{P}\left[\mathbf{c} = (0000)|\mathbf{y}\right] + \mathbb{P}\left[\mathbf{c} = (1101)|\mathbf{y}\right]\right) \\
&= 0.4740 \\
\mathbb{P}\left[c_4 = 0|\mathbf{y}\right] &= \left(\mathbb{P}\left[\mathbf{c} = (0000)|\mathbf{y}\right] + \mathbb{P}\left[\mathbf{c} = (0110)|\mathbf{y}\right]\right) \\
&= 0.0532
\end{aligned}
$$

Because of the normalization, we can get the other probabilities complementing to 1, i.e.

$$
\mathbb{P}\left[c_i = 1|\mathbf{y}\right] = 1 - \mathbb{P}\left[c_i = 0|\mathbf{y}\right]
$$

There is a problem in the previous approach to the problem. If we look more carefully (26)-(27) we note that the complexity of the task is due to the cardinality of the codebook: it is possible to list all the states if the cardinality of the codebook is not too big. Since the codebook cardinality is 2^k, the complexity increases exponentially with k and becomes rapidly unmanageable. To avoid this problem we show now how to apply the message passing algorithm.

5.2 Second part: message passing approach

Consider the vector $\mathbf{x} = (x_1, \dots, x_j, \dots, x_N)$ where each variable x_j belongs to an alphabet $\mathcal{A}_j$ and consider the function $f(\mathbf{x})$. Suppose f can be factorized as

$$f(\mathbf{x}) = \prod_i f_i(\mathbf{x}_i) \qquad \mathbf{x}_i \subseteq \mathbf{x} \tag{28}$$

We can build the corresponding factor graph. On this factor graph we exchange messages. All the messages represent the probability of the different values of a given variable node:
$P\left(x_j = \alpha_j\right)\ \forall \alpha_j \in A_j$. Now we describe the Variable and the Factor update rules.

5.2.1 Variable Node update

Given the variable node x_j, we denote with $E[x_j]$ the set of all factor nodes connected to x_j. The node x_j receives messages from all its neighbour factors. These messages represent the probability of the different values of x_j computed by that factor graph: $M_{f_i \to x_j}\left(x_j = \alpha_j\right)$ is the probability that $x_j = \alpha_j$ computed by the factor node f_i.

The variable node x_j computes the product of all incoming messages

$$\mathbb{P}\left[x_j = \alpha_j\right] = \prod_{f_i \in E[x_j]} M_{f_i \to x_j}\left(x_j = \alpha_j\right) \qquad \forall \alpha_j \in \mathcal{A}_j \tag{29}$$

and sends back to the node f_{i^*} the extrinsic information

$$M_{x_j \to f_{i^*}}\left(x_j = \alpha_j\right) = \prod_{\substack{f_i \in E[x_j] \\ f_i \neq f_{i^*}}} M_{f_i \to x_j}\left(x_j = \alpha_j\right) \qquad \forall \alpha_j \in \mathcal{A}_j \tag{30}$$

5.2.2 Factor Node update

Given the factor node f_i, we denote with $E[f_i]$ the set of all variable nodes connected to f_i. The factor node f_i node receives messages from all the nodes $x_j \in E[f_i]$. They have the form $M_{x_j \to f_i}\left(x_j = \alpha_j\right)$ and represent the probability that the variable x_j is equal to α_j according to the node x_j itself.

The factor node collects them and sends back to node x_{j^*} the messages

$$M_{f_i \to x_{j^*}}\left(x_{j^*} = \alpha_{j^*}\right) = \sum_{\mathbf{x}_i : x_{j^*} = \alpha_{j^*}} f_i(\mathbf{x}_i) \prod_{\substack{x_j \in E[f_i] \\ x_j \neq x_{j^*}}} M_{x_j \to f_i}\left(x_j = \alpha_j\right) \quad \forall \alpha_{j^*} \in \mathcal{A}_{j^*} \tag{31}$$

Example 5.2. Suppose we have the same $\mathbf{G}$ as in example 4.1 and suppose we have received $\mathbf{y} = 1101$, $p = 0.1$. In our example $f(\mathbf{x})$ corresponds to $\mathbb{P}[\mathbf{c}|\mathbf{y}]$. It can be factorized as

$$\begin{aligned}\mathbb{P}[\mathbf{c}|\mathbf{y}] &= \mu(\mathbf{c})\,\delta_1(c_1)\,\delta_2(c_2)\,\delta_3(c_3)\,\delta_4(c_4)\\ &= \underbrace{[c_1+c_2+c_3=0]}_{=f_1(c_1,c_2,c_3)}\cdot\underbrace{[c_1+c_4=0]}_{f_2(c_1,c_4)}\,\delta_1(c_1)\,\delta_2(c_2)\,\delta_3(c_3)\,\delta_4(c_4)\end{aligned}$$

The messages that variable nodes send to factor nodes can be stored in two matrices that have the same structure as $\mathbf{H}$

$$\begin{bmatrix} M_{c_1\to f_1}(c_1=0) & M_{c_1\to f_2}(c_1=0)\\ M_{c_2\to f_1}(c_2=0) & 0\\ M_{c_3\to f_1}(c_3=0) & 0\\ 0 & M_{c_4\to f_2}(c_4=0)\end{bmatrix}\quad\begin{bmatrix} M_{c_1\to f_1}(c_1=1) & M_{c_1\to f_2}(c_1=1)\\ M_{c_2\to f_1}(c_2=1) & 0\\ M_{c_3\to f_1}(c_3=1) & 0\\ 0 & M_{c_4\to f_2}(c_4=1)\end{bmatrix}$$

Even if one matrix of these is enough, because the elements are all probabilities that sum up to one, storing both of them makes the implementation easier.

We can store the messages sent back by factor nodes to variable nodes, for example, in two matrices of this form

$$\begin{bmatrix} M_{f_1\to c_1}(f_1=0) & M_{f_2\to c_1}(f_2=0)\\ M_{f_1\to c_2}(f_1=0) & 0\\ M_{f_1\to c_3}(f_1=0) & 0\\ 0 & M_{f_2\to c_4}(f_2=0)\end{bmatrix}\quad\begin{bmatrix} M_{f_1\to c_1}(f_1=1) & M_{f_2\to c_1}(f_2=1)\\ M_{f_1\to c_2}(f_1=1) & 0\\ M_{f_1\to c_3}(f_1=1) & 0\\ 0 & M_{f_2\to c_4}(f_2=1)\end{bmatrix}$$

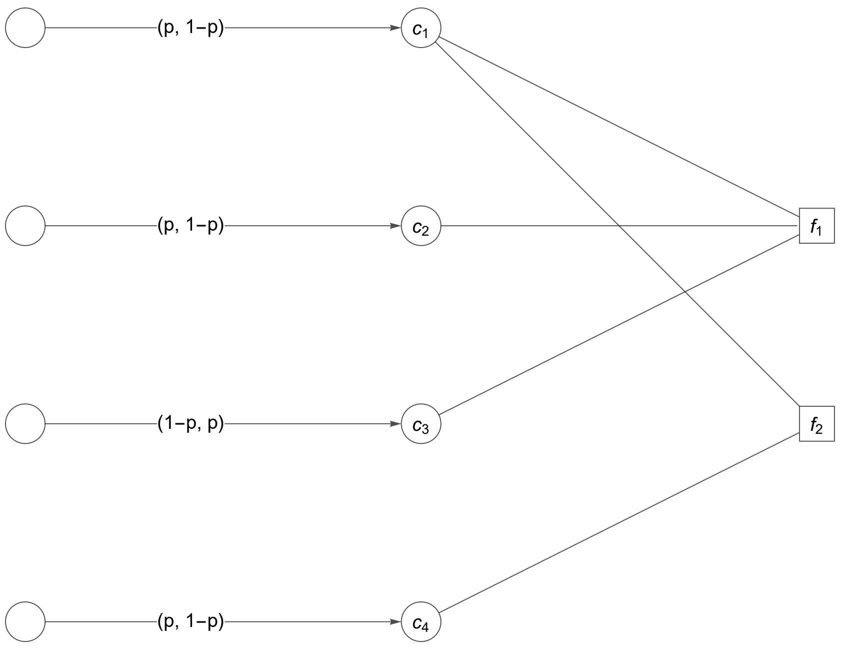
The starting point is considering fictitious nodes that send to variable nodes what is returned by the binary symmetric channel, as shown in fig. 7a.

Note that the probabilities on edges, which we denote by $(\delta_i(0), \delta_i(1))$, depend on the received vector $\mathbf{y}$ as shown in example example 5.1: if $y_i = 0$, then $\delta_i(0) = 1 - p$ and $\delta_i(1) = p$; if $y_i = 1$, then $\delta_i(0) = p$ and $\delta_i(1) = 1 - p$. The values $\delta_i(j)$ can be stored in a matrix such as

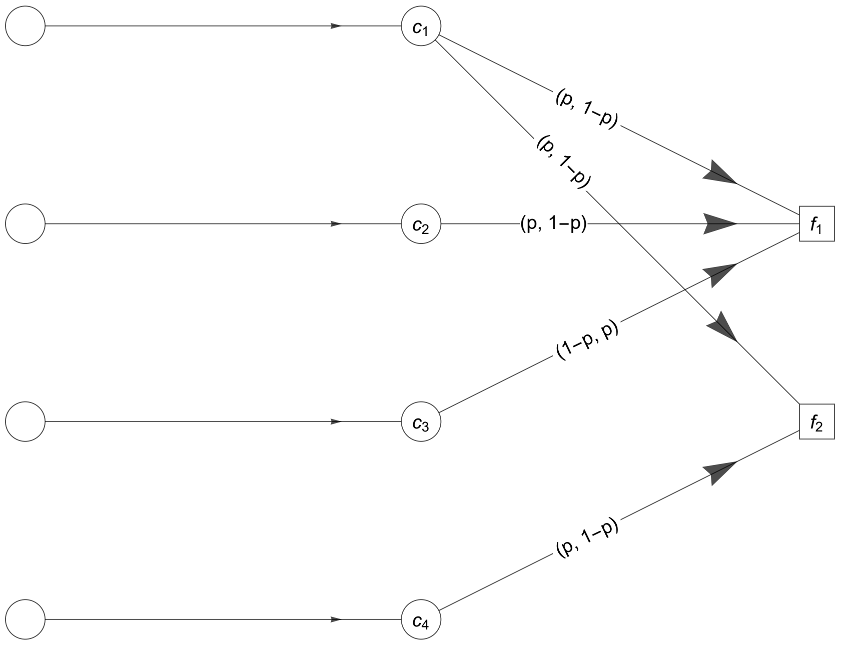
$$\begin{bmatrix} 0.1 & 0.9\\ 0.1 & 0.9\\ 0.9 & 0.1\\ 0.1 & 0.9\end{bmatrix} \tag{32}$$

The variable node propagates to the factor node the previous probabilities (as shown in fig. 7b), so the matrices assume the form

$$\begin{bmatrix} 0.1 & 0.1\\ 0.1 & 0\\ 0.9 & 0\\ 0 & 0.1\end{bmatrix}\quad\begin{bmatrix} 0.9 & 0.9\\ 0.9 & 0\\ 0.1 & 0\\ 0 & 0.9\end{bmatrix}$$



(a) Starting point of the algorithm. Fictitious nodes can just send information, they cannot receive anything: this is the reason why the edges are directed.



(b) The variable nodes propagate the message received by the fictitious nodes.

Figure 7

Now we want to compute the messages sent back by factor nodes. In doing so we can apply (31) and obtain

$$\begin{aligned}
M_{f_1\to c_1}(c_1=0) &= \sum_{(c_1,c_2,c_3):c_1=0} f_1(c_1,c_2,c_3) \prod_{c_j\neq c_1} M_{c_j\to f_1}(c_j) \\
&= f_1(0,0,0) M_{c_2\to f_1}(c_2=0)\, M_{c_3\to f_1}(c_3=0) \\
&+ \underbrace{f_1(0,0,1)}_{\substack{=0 \text{ because} \\ c_1+c_2+c_3\neq 0}} M_{c_2\to f_1}(c_2=0)\, M_{c_3\to f_1}(c_3=1) \\
&+ \underbrace{f_1(0,1,0)}_{\substack{=0 \text{ because} \\ c_1+c_2+c_3\neq 0}} M_{c_2\to f_1}(c_2=1)\, M_{c_3\to f_1}(c_3=0) \\
&+ f_1(0,1,1) M_{c_2\to f_1}(c_2=1)\, M_{c_3\to f_1}(c_3=1)
\end{aligned}$$

Just the vectors with $c_1+c_2+c_3=0$ survives after f_1. For this message $c_1=0$ so the condition becomes $c_2+c_3=0$. We have two possibilities in order to satisfy it: the first is taking both c_2 and c_3 equal to 0, the second is taking both c_2 and c_3 equal to 1.

Let us compute one more message

$$\begin{aligned}
M_{f_2\to c_1}(c_1=0) &= \sum_{(c_1,c_4):c_1=0} f_2(c_1,c_4) \prod_{c_j\neq c_1} M_{c_j\to f_2}(c_j) \\
&= f_2(0,0) M_{c_4\to f_2}(c_4=0) + \underbrace{f_1(0,1)}_{\substack{=0 \text{ because} \\ c_1+c_4\neq 0}} M_{c_4\to f_2}(c_4=1)
\end{aligned}$$

Computing for all the nodes we get

$$\begin{bmatrix} 0.18 & 0.1 \\ 0.18 & 0 \\ 0.82 & 0 \\ 0 & 0.1 \end{bmatrix}$$

$$\begin{bmatrix} 0.82 & 0.9 \\ 0.82 & 0 \\ 0.18 & 0 \\ 0 & 0.9 \end{bmatrix}$$

We now compute the estimated marginal probabilities applying (29).

$$\mathbb{P}[c_1=0] = \delta_1(0) M_{f_1\to c_1}(c_1=0)\; M_{f_2\to c_1}(c_1=0)$$

$$
\begin{aligned}
&= 0.1\,0.18\,0.1 = 0.0018 \\
\mathbb{P}\left[c_1 = 1\right] &= \delta_1(1) M_{f_1 \to c_1}\left(c_1 = 1\right)\, M_{f_2 \to c_1}\left(c_1 = 1\right) \\
&= 0.9\,0.82\,0.9 = 0.6642
\end{aligned}
$$

Due to the fact that this set of probabilities does not sum to 1, we normalize them

$$
\begin{aligned}
\frac{\mathbb{P}\left[c_1 = 0\right]}{\mathbb{P}\left[c_1 = 0\right] + \mathbb{P}\left[c_1 = 1\right]} &= \frac{0.0018}{0.0018 + 0.6642} = 0.0027 \\
\frac{\mathbb{P}\left[c_1 = 1\right]}{\mathbb{P}\left[c_1 = 0\right] + \mathbb{P}\left[c_1 = 1\right]} &= \frac{0.6642}{0.0018 + 0.6642} = 0.9973
\end{aligned}
$$

Computing for all the nodes, we get the following set of estimated marginal probabilities

$$
\begin{bmatrix}
0.0027 & 0.9973 \\
0.0238 & 0.9762 \\
0.9762 & 0.0238 \\
0.0122 & 0.9878
\end{bmatrix}
$$

Now we have to update the matrices of messages sent by the variable nodes and received by factor nodes. We use the concept of extrinsic information, so:

$$
\begin{aligned}
M_{c_1 \to f_1}\left(c_1 = 0\right) &= \delta_1(0)\, M_{f_2 \to c_1}\left(c_1 = 0\right) \\
&= 0.1\,0.1 = 0.01 \\
M_{c_1 \to f_1}\left(c_1 = 1\right) &= \delta_1(1)\, M_{f_2 \to c_1}\left(c_1 = 1\right) \\
&= 0.9\,0.9 = 0.81
\end{aligned}
$$

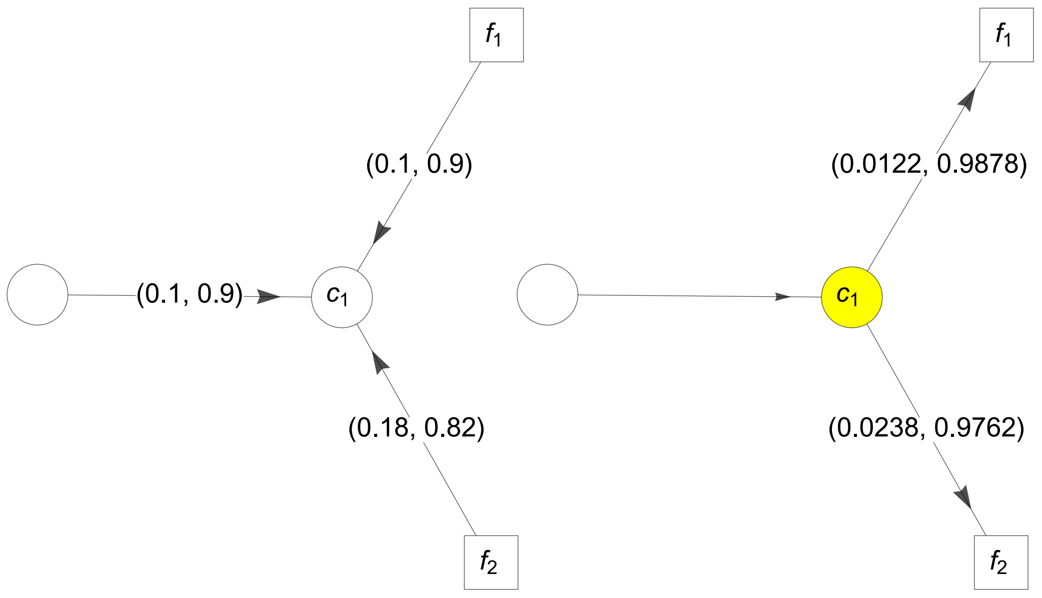
Again, because they are not normalize we normalize them. Computing all the messages and normalizing them we get

$$
\begin{bmatrix}
0.0122 & 0.0238 \\
0.1 & 0 \\
0.9 & 0 \\
0 & 0.1
\end{bmatrix}
\quad
\begin{bmatrix}
0.9878 & 0.9762 \\
0.9 & 0 \\
0.1 & 0 \\
0 & 0.9
\end{bmatrix}
$$

A picture of what is happening is shown in fig. 8.

We repeat the procedure twice, in order to converge to the marginal probabilities

$$
\begin{bmatrix}
0.0027 & 0.9973 \\
0.0135 & 0.9865 \\
0.9865 & 0.0135 \\
0.0027 & 0.9973
\end{bmatrix}
$$



(a) Incoming messages in c_1. (b) Normalized extrinsic information.

Figure 8: Detail of the first update of messages from variable to factors.

5.2.3 Gallager formula

Consider the simple expression

$$c_1 + c_2 = 0 \qquad c_1, c_2 \in \{0, 1\} \tag{33}$$

We denote with

$$p_i(0) = \mathbb{P}[c_i = 0] \qquad i = 1, 2 \tag{34}$$
$$p_i(1) = \mathbb{P}[c_i = 1] \qquad i = 1, 2 \tag{35}$$

We want to compute the probability that the equation is satisfied, i.e. the probability of having an even number of bits equal to 1. Consider the polynomial in t

$$\begin{aligned}&(p_1(0) + p_1(1)\,t)\,(p_2(0) + p_2(1)\,t)\\ &\qquad = p_1(0)\,p_2(0) + (p_1(0)\,p_2(1) + p_1(1)\,p_2(0))\,t + p_1(1)\,p_2(1)\,t^2\end{aligned} \tag{36}$$

We note that the coefficients of the even power of t contain all the necessary information. Specifically, summing the coefficients of the even power of t leads to the solution we were seeking for. In order to obtain just the even power of t we can consider

$$\frac{(p_1(0) + p_1(1)\,t)\,(p_2(0) + p_2(1)\,t) - (p_1(0) - p_1(1)\,t)\,(p_2(0) - p_2(1)\,t)}{2} \tag{37}$$

evaluating the expression in $t = 1$

$$\frac{(p_1(0) + p_1(1))\,(p_2(0) + p_2(1)) - (p_1(0) - p_1(1))\,(p_2(0) - p_2(1))}{2} \tag{38}$$

that can be further simplified

$$\frac{1-(1-2\,p_1(1))(1-2\,p_2(1))}{2} \tag{39}$$

obtaining the solution.

Generalizing, suppose we have the equation

$$\sum_{i=1}^{n} c_i = 0 \qquad c_i \in \{0,1\} \quad \forall i = 1,\dots,n \tag{40}$$

where

$$p_i(0) = \mathbb{P}[c_i = 0] \quad \forall i = 1,\dots,n \tag{41}$$

$$p_i(1) = \mathbb{P}[c_i = 1] \quad \forall i = 1,\dots,n \tag{42}$$

As before, we want to compute the probability that the equation is satisfied, i.e. the probability of having an even number of bits equal to 1. Consider the polynomial in t

$$\prod_{i=1}^{n}(p_i(0)+p_i(1)\,t) \tag{43}$$

we can extract the even powers of t considering

$$\frac{\prod_{i=1}^{n}(p_i(0)+p_i(1)\,t)-\prod_{i=1}^{n}(p_i(0)-p_i(1)\,t)}{2} \tag{44}$$

and we can obtain the sum of the coefficients evaluating the expression in $t=1$

$$\frac{\prod_{i=1}^{n}(p_i(0)+p_i(1))-\prod_{i=1}^{n}(p_i(0)-p_i(1))}{2} \tag{45}$$

It can be simplified in

$$\frac{1-\prod_{i=1}^{n}(1-2\,p_i(1))}{2} \tag{46}$$

This trick can be used in order to simplify (31) resulting in

$$M_{f_i\to c_j^*}\left(c_j^*=0\right) = \frac{1+\prod_{c_j\neq c_j^*}\left(1-2\,M_{c_j\to f_i}\left(c_j=1\right)\right)}{2} \tag{47}$$

because a factor graph leads always to equations of type (40).

Example 5.3. Consider example 5.2. The first factor node is

$$f_1 = [c_1 + c_2 + c_3 = 0]$$

We want to compute $M_{f_1\to c_1}\left(c_1=0\right)$. The approach that we used in example 5.2 is

$$M_{f_1\to c_1}\left(c_1=0\right) = \sum_{(c_1,c_2,c_3):c_1=0} f_1(c_1,c_2,c_3)\prod_{c_j\neq c_1} M_{c_j\to f_1}\left(c_j\right)$$

$$\begin{aligned}
&= f_1(0,0,0)M_{c_2\to f_1}\left(c_2=0\right)M_{c_3\to f_1}\left(c_3=0\right)\\
&+ f_1(0,0,1)\,M_{c_2\to f_1}\left(c_2=0\right)M_{c_3\to f_1}\left(c_3=1\right)\\
&+ f_1(0,1,0)\,M_{c_2\to f_1}\left(c_2=1\right)M_{c_3\to f_1}\left(c_3=0\right)\\
&+ f_1(0,1,1)M_{c_2\to f_1}\left(c_2=1\right)M_{c_3\to f_1}\left(c_3=1\right)\\
&= p\,(1-p)+(1-p)\,p\\
&= 2\,p\,(1-p)
\end{aligned}$$

We can use the Gallager formula after noting that $c_1 = 0$ is fixed so we have to solve $c_2 + c_3 = 0$, which is similar to (33). The solution is

$$\begin{aligned}
M_{f_1\to c_1}\left(c_1=0\right) &= \frac{1+\prod_{c_j\neq c_1}\left(1-2\,M_{c_j\to f_1}\left(c_j=1\right)\right)}{2}\\
&= \frac{1+\left(1-2\,M_{c_2\to f_1}\left(c_2=1\right)\right)\left(1-2\,M_{c_3\to f_1}\left(c_3=1\right)\right)}{2}\\
&= \frac{1+\left(1-2\,(1-p)\right)\left(1-2\,p\right)}{2}\\
&= \frac{1+\left(2\,p-1\right)\left(1-2\,p\right)}{2}\\
&= \frac{1+2\,p-1-4\,p^2+2\,p}{2}\\
&= 2\,p\,(1-p)
\end{aligned}$$

as we expected.

5.3 Matlab exercise

The values of $\mathbf{R}$, $\mathbf{y}$, p can change so the Matlab code must be able to accept different values for them.