NOTES ON PROBABILISTIC DECODING OF PARITY CHECK CODES

PETER CARBONETTO*

1. Basics. The diagram given in p. 480 of [7] perhaps best summarizes our model of a noisy channel. We start with an array of bits $\boldsymbol{s} = (s_1, \ldots, s_K)$, where each $s_k \in \{0, 1\}$. This is our source message, and the length K is the "degrees of freedom." We encode the source message $\boldsymbol{s}$ into a vector of bits $\boldsymbol{x} \in \mathcal{C}$, where $\mathcal{C} \subseteq \{0, 1\}^N$ and $N \geq K$. A member of $\mathcal{C}$ is called a *codeword*, and N is the *code length*. The discrete codeword $\boldsymbol{x}$ is transformed to a continuous signal $\boldsymbol{t} \in \mathbb{R}^N$, next it is transmitted through a noisy channel and we, on the receiving end, see the signal $\boldsymbol{y} \in \mathbb{R}^N$. The signal $\boldsymbol{y}$ has been corrupted to some degree by noise. The relationship between the continuous transmitted signal $\boldsymbol{t}$ and the received signal $\boldsymbol{y}$ is usually modelled by

$$\boldsymbol{y} = \boldsymbol{t} + \boldsymbol{v}, \tag{1}$$

where $\boldsymbol{v}$ is the noise vector. We assume the individual entries of $\boldsymbol{v}$ are independently and identically drawn from a Normal distribution with zero mean and variance σ^2.

The object is to *decode* the received codeword $\boldsymbol{y}$ by inferring the (unseen) original codeword $\boldsymbol{x}$ that maximizes the posterior probability $p(\boldsymbol{x} \,|\, \boldsymbol{y})$, and then using $\boldsymbol{x}$ to find the original message $\boldsymbol{s}$.[1] In order to accomplish this, we need to come up with a prior probability distribution $p(\boldsymbol{x})$ over codewords $\boldsymbol{x}$, and a likelihood distribution $p(\boldsymbol{y} \,|\, \boldsymbol{x})$ over received signals $\boldsymbol{y}$ given codewords $\boldsymbol{x}$. Let's begin by investigating a distinguished class of codes called *linear codes.*

A linear code endows the set of codewords $\mathcal{C}$ with a special algebraic structure. A code is linear if, whenever vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ belong to $\mathcal{C}$, then their sum $\boldsymbol{a} + \boldsymbol{b}$ also belongs to $\mathcal{C}$.[2] One way to construct a linear code is to design a matrix A with M rows and N columns, where $M \leq N$, and say that all codewords are those vectors $\boldsymbol{x} \in \{0, 1\}^N$ satisfying $A\boldsymbol{x} = \boldsymbol{0}$. When $A\boldsymbol{a} = \boldsymbol{0}$ and $A\boldsymbol{b} = \boldsymbol{0}$, it follows that

$$A(\boldsymbol{a} + \boldsymbol{b}) = A\boldsymbol{a} + A\boldsymbol{b} = \boldsymbol{0}. \tag{2}$$

Therefore, the set $\mathcal{C}$ defined by $\mathcal{C} = \{\boldsymbol{x} \in \{0, 1\}^N \,|\, A\boldsymbol{x} = \boldsymbol{0}\}$ is a linear code. We call A the *parity check matrix.*

It is easy to see how one might make use of the parity check matrix to detect and correct errors in the received signal. Suppose we were to threshold the entries of $\boldsymbol{y}$ in some fashion to obtain a discrete signal.[3] Since the continuous signal has been subject to some level of interference, the thresholded $\boldsymbol{y}$ may not be the same as $\boldsymbol{x}$. Consider the case when one of the bits is incorrect. The received word is thus $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e}$, where $\boldsymbol{e}$ is zero everywhere except for a 1 in the ith bit. So we have

$$A\boldsymbol{y} = A(\boldsymbol{x} + \boldsymbol{e}) = A\boldsymbol{x} + A\boldsymbol{e}. \tag{3}$$

*Department of Computer Science, University of British Columbia (pcarbo@cs.ubc.ca).

[1] Once we know $\boldsymbol{x}$, we can solve for $\boldsymbol{s}$ via Gaussian elimination on the generator matrix. See Sec. 5 for more information.

[2] Arithmetic here is defined on $\mathbb{Z}_2$, the set of integers modulo 2. In this setting, addition is the same as subtraction.

[3] In order to distinguish the continuous representation $\boldsymbol{y}$ from its corresponding bit vector, we shall use the notation $\boldsymbol{y}$. This is consistent with the notation used in [6].

The term $A\boldsymbol{x}$ is $\mathbf{0}$, and the term $A\boldsymbol{e}$ is equal to the ith column of A. Therefore, all we have to do is compare the result $A\boldsymbol{y}$ with all the columns of A to locate and correct the error. In the next section, we will describe an elegant method for error correction with arbitrary numbers of errors.

2. The probabilistic decoding problem. As mentioned in the previous section, the objective is framed as follows: find a codeword $\boldsymbol{x} = (x_1, \ldots, x_N)$ that maximizes the posterior

$$p(\boldsymbol{x} \,|\, \boldsymbol{y}) \propto p(\boldsymbol{y} \,|\, \boldsymbol{x})\, p(\boldsymbol{x}). \tag{4}$$

What do the prior and likelihood look like? The prior can be formulated directly from the structure of the parity check matrix. Each row of A is a parity check: for every row $j = 1, \ldots, M$, the condition

$$\bigoplus_{i \,\in\, N(j)} x_i = 0 \tag{5}$$

must hold for $\boldsymbol{x}$ to be a valid codeword. The summation in (5) is over all columns i such that $A_{ji} = 1$. The 1s in the jth row of A tell us which codeword bits participate in the jth parity check. Now, we can interpret the parity check as a factor in a factor graph [16], where variable nodes correspond to codeword bits. The factor associated with the jth parity check is

$$f_j(\boldsymbol{x}_{N(j)}) = \begin{cases} 1 & \text{if } \bigoplus_{i \in N(j)} x_i = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

The prior distribution of $\boldsymbol{x}$ is thus given by

$$p(\boldsymbol{x}) \propto \prod_j f_j(\boldsymbol{x}_{N(j)}), \tag{7}$$

and resolves to 0 whenever $\boldsymbol{x}$ is not a valid codeword. The product in (7) is over all rows j of the parity check matrix A.

In order to evaluate the likelihood factors, we need to understand how a stream of bits $\boldsymbol{x}$ is represented by a continuous waveform $\boldsymbol{t}$ for transmission across a channel.

3. Transmission through a Gaussian channel. A proper understanding of this subject would necessitate a course in digital communication, but a cursory presentation should be sufficient for the purpose of this monograph. We shall examine binary phase-shift keying [12], in which each bit is represented by the sign of the waveform: $t_i = a$ represents the 1 bit, and $t_i = -a$ represents the 0 bit. The signal $\boldsymbol{t}$ passes through a channel with *i.i.d* noise, so the probability of observing $\boldsymbol{y}$ at the other end of the channel is given by the product of terms

$$p(y_i \,|\, x_i) \propto \begin{cases} \exp\{\frac{-1}{2\sigma^2}(y_i + a)^2\} & \text{if } x_i = 0 \\ \exp\{\frac{-1}{2\sigma^2}(y_i - a)^2\} & \text{if } x_i = 1 \end{cases} \tag{8}$$

for every codeword bit $i = 1, \ldots, N$. Computing the normalizing constant, we can express the likelihood as

$$p(y_i \,|\, x_i = 1) = \frac{1}{1 + \exp(-2ay_i/\sigma^2)}. \tag{9}$$

The likelihood $p(\boldsymbol{y} \,|\, \boldsymbol{x})$ thus introduces a factor $p(y_i \,|\, x_i)$ for every bit i.

4. An approximate solution to the probabilistic decoding problem. We now have a model that provides us with the *a posteriori* probability $p(\boldsymbol{x} \mid \boldsymbol{y}) \propto p(\boldsymbol{y} \mid \boldsymbol{x})\, p(\boldsymbol{x})$ of any codeword $\boldsymbol{x}$ given the observed signal $\boldsymbol{y}$. The prior and the likelihood terms are defined as products of factors. For our purposes, this can be reduced to computing the marginal densities of the posterior, $p(x_i \mid \boldsymbol{y})$, for every bit i.[4] And once we know the marginals, we will then select the highest probability bits. However, computing these probabilities is rather difficult for most desirable parity check matrices. Thus, we need a tractable method for estimating the posterior.

The best known tractable solution is to frame the inference problem (the problem of computing the marginals of the posterior) as an optimization problem using variational methodology, then to approximate the optimization problem using a *region-based approximation* [16] so we can arrive at the marginals fairly efficiently. Let's describe this in more detail.

Let $p(\boldsymbol{x})$ denote the target probability distribution. It is composed from a product of factors, so that the probability of configuration $\boldsymbol{x} = \{x_1, \ldots, x_N\}$ can be written

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_C f_C(\boldsymbol{x}_C), \tag{10}$$

where the product is over all defined factor neighbourhoods C. Each C refers to a subset of $\{1, \ldots, N\}$, so that $\boldsymbol{x}_C$ represents the restriction of configuration $\boldsymbol{x}$ to the subset C. We assume that no two factors are defined on the same subset C, so that a factor can be uniquely identified by its neighbourhood. The normalizing constant Z is designed to ensure that $p(\boldsymbol{x})$ represents a valid probability; that is, the probabilities of all configurations must sum to one. Based on the discussion in the previous sections, it is easy to see that the posterior distribution (4) for decoding can be represented as a product of factors, and that the normalizing constant is equal to the marginal density, $Z = \int p(\boldsymbol{y} \mid \boldsymbol{x})\, p(\boldsymbol{x})\, d\boldsymbol{x}$.

Variational methods frame the problem of inferring statistics with respect to a target distribution $p(\boldsymbol{x})$ as an optimization problem by introducing a *variational distribution* $q(\boldsymbol{x})$. The goal is to come up with a variational distribution which is as close as possible to the target. Mathematically speaking, this leads to the following optimzation problem: find a $q(\boldsymbol{x})$ which minimizes the *variational free energy*

$$F = -\sum_{\boldsymbol{x}} q(\boldsymbol{x}) \sum_C \log f_C(\boldsymbol{x}_C) + \sum_{\boldsymbol{x}} q(\boldsymbol{x}) \log q(\boldsymbol{x}). \tag{11}$$

The first set of terms is called the *average energy*, and is usually denoted with the symbol U. The second set of terms is the negation of the entropy. We use H as shorthand for the entropy. Clearly, the best variational distribution is one that matches the target exactly. And at this point, the variational free energy becomes $-\log Z$. While this variational formulation naturally lends itself to optimization, it still involves intractable sums over all possible configurations.

The strategy we will expound here, whose roots lie in the early work of Hans Bethe [2] and Ryoichi Kikuchi [9] in statistical physics, is to approximate the intractable sums in F by a linear combination of more manageable terms F_R. The R represents a "region" or "cluster" of the undirected graphical model, and is a subset of the indices $\{1, \ldots, N\}$. Bethe [2] proposed an approximation to the variational free energy F by forcing the entropy to decompose as a product of entropy terms on

[4] This is a technically incorrect leap to make, but it tends to work well anyhow.

the sets C and singleton sets $\{i\}$. This approximation is generally referred to as the *Bethe free energy*. The junction graph method is a natural generalization of the Bethe method in which the large regions (the sets C) and the small regions (the singletons $\{i\}$) can be chosen with greater freedom. One ordinarily uses a *junction graph* to formalize these notions [1].

A *region graph* is a graph with directed edges and labeled vertices. Each vertex is labeled by a *region* of the target factor graph. A region is defined to be a collection of variable nodes and factor nodes, with the single restriction that if a factor belongs to a region, then all its arguments also belong to the region. We denote a region by the capital letter R. Depending on the context, the symbol R may alternately refer to a collection of variable nodes, a collection of factor nodes, or a node of the region graph. In this manner, we may use $\boldsymbol{x}_R$ to denote the configuration $\boldsymbol{x}$ restricted to the set $R \subseteq \{1, \ldots, N\}$, we may use the notation $C \in R$ to refer to factors C that are members of region R, additionally $q_R(\boldsymbol{x}_R)$ denotes the marginal density function defined at region R. For the purposes of this exposition, we assume that a collection of variable nodes uniquely identifies a region; no two regions possess the exact same set of variables.

Given a region graph, its corresponding *region free energy* is defined to be

$$\tilde{F} = \sum_R c_R U_R - \sum_R c_R H_R, \tag{12}$$

where the average energy U_R and entropy H_R of region R are, respectively,

$$U_R = -\sum_{\boldsymbol{x}_R} \sum_{C \in R} q_R(\boldsymbol{x}_R) \log f_C(\boldsymbol{x}_C) \qquad \text{and} \qquad H_R = -\sum_{\boldsymbol{x}_R} q_R(\boldsymbol{x}_R) \log q_R(\boldsymbol{x}_R).$$

We define $q_R(\boldsymbol{x}_R)$ to be the marginal probability defined on region R, and c_R to be the "counting" number (or "overcounting" number) for region R. If the counting numbers are chosen well, the decomposition of the average energy is exact. On the other hand, the entropy decomposition can only ever be an approximation.

Yedidia et al. [16] present a recipe for coming up with reasonable counting numbers c_R for a given region graph. A good choice of numbers c_R ensures that we only count the contribution of each subset once in $\tilde{F}$, and this insight the basis for the *cluster variation method*. The important observation made by McEliece and Yildirim [11] is that this recipe has a profound connection to results in combinatorial mathematics and, in particular, the theory of partially ordered sets [4]. By introducing a *partial ordering*—a relation with specific properties—on the regions, we can treat the collection of regions as a partially ordered set (poset), where the partial ordering is the set inclusion relation, and we can then draw the regions as vertices in a Hasse diagram [13]. Since we have described the regions R as elements of a poset, we can frame the choice of counting numbers as a counting problem on the poset, and use the principle of inclusion and exclusion for partially ordered sets—widely known as the *Möbius inversion principle*—to come up with the answer [4].

In most decoding problems we will encounter, fortunately, it isn't quite this complicated. That's because most of the parity check matrices we will be working with have the property that each pair of columns (and each pair of rows) do not have more than one $A_{ji} = 1$ in common. This in turn means that each parity check factor does not have more than one bit in common with any other parity check factor.

Now suppose we define two sets of regions, and we denote their members by R and S, respectively. In the first set of regions, each R contains the jth parity check

factor, the variable nodes $i \in N(j)$ corresponding to the arguments of the jth factor, and the likelihood factors $p(y_i \,|\, x_i)$ for all $i \in N(j)$. In the second set of regions, each S contains the variable node i and the factor node corresponding to the ith likelihood term. We set the counting numbers c_R to 1 for all R. For each $S = \{i\}$, we set c_S to be equal to the number of parity checks with which the ith bit participates. A region graph defined in this way ensures that the average energy is exact, and that the contribution of every subset of is only counted once in $\tilde{F}$. This is so because: 1) every non-empty intersection of two regions is a member of the region graph, and 2) the counting numbers are equivalent to those obtained as a solution to the Möbius inversion principle. It is also worth noting the region-based approximation is equivalent to the junction graph approximation in this case, where the Rs correspond to large regions of the junction graph, the Ss correspond to the its small regions, or separators, and the counting numbers of the small regions are the degrees of the member variable nodes.[5] From now on, we shall restrict our attention to the junction graph approximation in which the large regions R have counting number $c_R = 1$, and the small regions $S = \{i\}$ have counting number $c_S = 1 - d_i$. The degree d_i of the ith variable node is defined to be the number of 1s in the ith column of A. The junction graph approximation to the variational free energy is given by

$$\tilde{F} = \sum_R U_R + \sum_S c_S U_S - \sum_R H_R - \sum_S c_S H_S. \tag{13}$$

Since the large regions R of the junction graph are in correspondence with the parity checks $j = 1, \ldots, M$, and the small regions S are in correspondence with the bits $i = 1, \ldots, N$, we can expand and simplify (13) to obtain the expression

$$\begin{aligned}\tilde{F} = &- \sum_j \sum_{\boldsymbol{x}_{N(j)}} q_j(\boldsymbol{x}_{N(j)}) \log f_j(\boldsymbol{x}_{N(j)}) - \sum_j \sum_{i \in N(j)} \sum_{x_i} q_j(x_i) \log g_i(x_i) \\ &- \sum_i (1 - d_i) \sum_{x_i} q_i(x_i) \log g_i(x_i) + \sum_j \sum_{\boldsymbol{x}_{N(j)}} q_j(\boldsymbol{x}_{N(j)}) \log q_j(\boldsymbol{x}_{N(j)}) \\ &+ \sum_i (1 - d_i) \sum_{x_i} q_i(x_i) \log q_i(x_i),\end{aligned} \tag{14}$$

where $g_i(x_i)$ is shorthand for the response of the likelihood term $p(y_i \,|\, x_i)$.

As we mentioned eariler, the object is to come up with a collection of marginals $q_R(\boldsymbol{x}_R)$ and $q_i(x_i)$ that minimizes the approximate variational free energy (14). This is a constrained optimization problem, since the marginals must be non-negative and sum to one. Moreover, they must be consistent with each other. The optimization problem is to minimize (14) subject to three types of constraints: 1) the marginal probabilities must be non-negative, 2) they must sum to one, and 3) the marginals on neighbouring regions should agree. The constrained, nonconvex program is

$$\begin{array}{ll} \text{minimize} & \tilde{F} \\ \text{subject to} & \forall\, j,\, \boldsymbol{x}_{N(j)},\, q_j(\boldsymbol{x}_{N(j)}) \geq 0 \qquad \forall\, i,\, x_i,\, q_i(x_i) \geq 0 \\ & \forall\, j,\, \textstyle\sum_{\boldsymbol{x}_{N(j)}} q_R(\boldsymbol{x}_{N(j)}) = 1 \qquad \forall\, i,\, \textstyle\sum_{x_i} q_i(x_i) = 1 \\ & \forall\, j,\, i \in N(j),\, x_i,\, c_{ji}(x_i) = 0, \end{array} \tag{15}$$

[5] In general, an approximation to the variational free energy based on a proper junction graph can only guarantee that the contribution of terms involving variable node i is not counted more than once in F. This is a weaker guarantee than what we mentioned earlier, because it could still happen that terms defined on an arbitrary subset of $\{1, \ldots, N\}$ is not counted exactly once. In this particular case, however, the junction graph approximation harbours this stronger guarantee.

where the consistency constraint function is defined to be

$$c_{ji}(x_i) = \sum_{\boldsymbol{x}_{N(j)-\{i\}}} q_j(\boldsymbol{x}_{N(j)}) - q_i(x_i). \tag{16}$$

The sensible course of action at this point would be to solve the program (15) by deriving the gradient $\nabla\tilde{F}$ and Hessian $\nabla^2\tilde{F}$, then executing a software package for constrained, nonlinear programming such as IPOPT [15]. The Hessian is diagonal, so this solution would not seem costly. There is a major problem with this seemingly harmless approach, however. Evaluation of the objective function $\tilde{F}$ requires that we evaluate the response of the parity check factor $f_j(\boldsymbol{x}_{N(j)})$ for every configuration of the bits $i \in N(j)$ involved in the parity check. When $N(j)$ is large, the summations in (14) can become extraordinarily expensive to evaluate (not to mention the large number of entries to the gradient and Hessian we would have to keep track of!). We will see that we can dispense of this problem by following the standard course of action and exploit results in duality, leading to familiar sum-product updates [16].

The Lagrangian function corresponding to (15) is

$$\begin{aligned}\tilde{L} = \tilde{F} &+ \sum_j \gamma_j \big\{ \textstyle\sum_{\boldsymbol{x}_{N(j)}} q_j(\boldsymbol{x}_{N(j)}) - 1 \big\} + \sum_i \gamma_i \big\{ \textstyle\sum_{x_i} q_i(x_i) - 1 \big\} \\ &+ \sum_j \sum_{i \in N(j)} \sum_{x_i} \nu_{ji}(x_i) \big\{ q_i(x_i) - \textstyle\sum_{\boldsymbol{x}_{N(j)-\{i\}}} q_j(\boldsymbol{x}_{N(j)}) \big\},\end{aligned} \tag{17}$$

For the purposes of this exposition, we assume that marginal probabilities are strictly positive so that their associated Lagrange multipliers vanish. Starting from the Lagrange dual function and its properties, it is possible to derive necessary conditions for optimality, otherwise known as the conditions of Karush, Kuhn and Tucker [5]. For a candidate point to be optimal, it must minimize the Lagrangian for some point $\{\boldsymbol{\gamma}, \boldsymbol{\nu}\}$, which in turns means that the gradient of the Lagrangian with respect to the primal variables must vanish. The partial derivatives of the Lagrangian (17) with respect to the primal variables are given by

$$\frac{\partial \tilde{L}}{\partial q_j(\boldsymbol{x}_{N(j)})} = \log q_j(\boldsymbol{x}_{N(j)}) + 1 - \log f_j(\boldsymbol{x}_{N(j)}) - \sum_{i \in N(j)} \log g_i(x_i) + \gamma_j - \sum_{i \in N(j)} \nu_{ji}(x_i) \tag{18}$$

$$\frac{\partial \tilde{L}}{\partial q_i(x_i)} = (1 - d_i)\big[\log q_i(x_i) + 1 - \log g_i(x_i) \big] + \gamma_i + \sum_{j \in N(i)} \nu_{ji}(x_i), \tag{19}$$

where $N(i)$ is the collection of parity checks with which the ith bit participates, which is equivalent to the collection of bits participating in the corresponding parity check. We recover the coordinate ascent equations finding by equating the partial derivatives to zero and solving for the variables $q_R(\boldsymbol{x}_R)$ and $q_i(x_i)$, arriving at

$$\begin{aligned} q_j(\boldsymbol{x}_{N(j)}) &\propto \exp\big\{ \log f_j(\boldsymbol{x}_{N(j)}) + \textstyle\prod_{i \in N(j)} \log g_i(x_i) + \nu_{ji}(x_i) \big\} \\ q_i(x_i) &\propto \exp\big\{ \tfrac{1}{d_i - 1} \big(\log g_i(x_i) + \textstyle\sum_{j \in N(i)} \nu_{ji}(x_i) \big) \big\}. \end{aligned}$$

Next, by making the substitutions

$$\nu_{ji}(x_i) = \log m_{i \to j}(x_i) \tag{20}$$

$$m_{i \to j}(x_i) = \textstyle\prod_{j' \in N(i) - \{j\}} m_{j' \to i}(x_i) \tag{21}$$

the expressions for the marginals become

$$q_j(\boldsymbol{x}_{N(j)}) \propto f_j(\boldsymbol{x}_{N(j)}) \times \textstyle\prod_{i \in N(j)} g_i(x_i)\, m_{i \to j}(x_i) \tag{22}$$

$$q_i(x_i) \propto g_i(x_i) \times \textstyle\prod_{j \in N(i)} m_{j \to i}(x_i), \tag{23}$$

which give us the familiar expressions for the marginal beliefs on the small and large regions of the junction graph. The message update from the ith small region to the jth large region is given in (21), so the remaining piece of the puzzle is the update equation for a message passed from j to i. Starting from the identity

$$q_i(x_i) = \textstyle\sum_{\boldsymbol{x}_{N(j)-i}} q_j(\boldsymbol{x}_{N(j)}),$$

then plugging equations (22) and (23) into this identity, we obtain the standard sum-product rule

$$m_{j \to i}(x_i) \propto \sum_{\boldsymbol{x}_{N(j)-\{i\}}} f_j(\boldsymbol{x}_{N(j)}) \prod_{i' \in N(j)-\{i\}} m_{i' \to j}(x_{i'}). \tag{24}$$

If we dispense with the marginal probabilities (22) altogether, we can almost, but not quite, breath a little easier. We still have to deal with a potentially monstrous summation over the configurations $\boldsymbol{x}_{N(j)-\{i\}}$ in the sum-product message update (24).

With some measure of cleverness, it is possible to compute the message $m_{j \to i}$ in linear time with respect to the size of the parity check. To make things as simple as possible, let's assume that the set of bits implicated in the jth factor is $N(j) = \{1, \ldots, n\}$. Let's introduce a random variable Z_k that is defined to be the sum of the bits from 1 to k, and we denote $\boldsymbol{x}_{1:k}$ to be the configuration restricted to the set of bits from 1 to k. The probability of event $Z_k = z_k$ is simply given by

$$p(z_k) = \sum_{\boldsymbol{x}_{1:k}} \delta_{z_k} \left(\textstyle\bigoplus_{l=1}^{k} x_l \right). \tag{25}$$

That is, the probability of event $Z_k = z_k$ is equal to sum of the probabilities of events in which the bits from 1 to k add up to z_k. For any k, we can compute $p(z_k)$ by induction. Clearly, $p(z_1) = p(x_1 = z_1)$. The probability of z_2 can then be computed according to

$$\begin{aligned} p(z_2) &= p(z_2 \,|\, z_1 = z_2)\, p(z_1 = z_2) + p(z_2 \,|\, z_1 \neq z_2)\, p(z_1 \neq z_2) \\ &= p(x_2 = 0)\, p(z_1 = z_2) + p(x_2 = 1)\, p(z_1 \neq z_2). \end{aligned}$$

In general, z_k is computed inductively according to the equation

$$p(z_k) = p(x_k = 0)\, p(z_{k-1} = z_k) + p(x_k = 1)\, p(z_{k-1} \neq z_k). \tag{26}$$

Now, how can we use this to compute the sum-product message? By inspection, we see that the following identity holds:

$$m_{j \to i}(x_i) = p(z_n = 0 \,|\, x_i), \tag{27}$$

where the marginal probabilities $p(x_k)$ are given by the messages $m_{k \to j}(x_k)$. If it so happens that $i = n$, then we have

$$m_{j \to n}(x_n) = p(z_{n-1} = x_n).$$

We now have all the ingredients to implement a practical algorithm for decoding a received signal $\boldsymbol{y}$. Usually, the message passing terminates when a *maximum a posteriori* estimate $\boldsymbol{x}$ satisfies $A\boldsymbol{x} = \boldsymbol{0}$, or when a specified maximum number of iterations is reached (in which case, we report a decoding failure).

5. Encoding a message. What we have described so far is a principled method for estimating the original message given a noisy received signal. We have not yet described how to encode the message $\boldsymbol{s}$ is the first place.

Suppose we are provided with an $M \times N$ parity check matrix A, where $M \leq N$. How can we use A to encode the original message? We need to find the corresponding $N \times K$ systematic generator matrix G that takes a message s and transforms it into a codeword $\boldsymbol{x}$. The transformation in matrix notation is

$$\boldsymbol{x} = G\boldsymbol{s}. \tag{28}$$

Since $\boldsymbol{x}$ is a codeword, we should have

$$A\boldsymbol{x} = AG\boldsymbol{s} = \mathbf{0} \tag{29}$$

for any message $\boldsymbol{s}$. Note that the generator corresponding to A is not unique (see p. 84 of [12]). The result (29) implies that

$$AG = 0, \tag{30}$$

because a row of A must be orthogonal to every basis vector f the space of codewords (*i.e.* the columns of G). Therefore, if we've going to find a generator matrix G corresponding to A, it needs to satisfy (30).

Assuming the $M \times N$ matrix A has rank M, we can use the Gauss-Jordan algorithm [14] to find an $M \times M$ matrix B such that

$$B^{-1}A = H = [\, I \;\; -U \,]. \tag{31}$$

Here, I is the $M \times M$ identity matrix and U is a matrix of height M and width $N - M$. The matrix $H = [\, I \;\; -U \,]$ serves the same role as A because $H\boldsymbol{x} = 0$ if and only if $A\boldsymbol{x} = 0$. Amazingly, the matrix

$$G = \begin{bmatrix} U \\ I \end{bmatrix} \tag{32}$$

satisfies the condition (30), since $A = BH$ and

$$AG = BHG = B\,[\, I \;\; -U \,] \begin{bmatrix} U \\ I \end{bmatrix} = 0.$$

Therefore, G is a generator matrix. Under the assumption that the parity check matrix has rank M, the width of the generator matrix is $N - M$, which means that the number of degrees of freedom is given by $K = N - M$. The specific form (32) is called *systematic form*, because the message symbols may be found explicitly and unchanged in the codeword (see p. 85 of [12]). If we have the generator matrix is systematic form, we can obtain an estimate of the source message $\boldsymbol{s}$ by inspecting the bits of $\boldsymbol{x}$.

6. The design of a parity check matrix. Biggs captures the fundamental problem of coding theory on p. 380 of [3]:

> We should like K to be reasonably large comparsed with N, in order that a good number of different messages can be sent without too much effort. But if there are lots of codewords, then the distance between them will be small, and few errors can be corrected.

The measure of a code's efficiency is its *rate*, which defined to be the number of message bits communicated per codeword bit. It is equal to K divided by N. The standard measure of a code's effectiveness is the probability of bit error, also known as the bit error rate. This is the probability that a message bit obtained by decoding the signal $\boldsymbol{y}$ is not the same as the source message bit.

It has been shown that parity check matrices with very few 1s—that is, they are sparse—can be extremely effective, while permitting very efficient, approximate decoding using variational methodology. These codes are generically called *low-density parity check codes*, or alternately Gallager codes after their inventor. For more information on constructing low-density parity check matrices, consult Gallager's doctoral dissertation [8] or more recent work [10].

7. Miscellany. Very often in coding theory, results are expressed by plotting bit error rate versus the signal-to-noise ratio $SNR = E_b/N_0$, where E_b is the energy expended per message bit and $\sigma^2 = N_0/2$ is the variance and spectral density of the noise. We can derive the variance of the noise from the signal-to-noise ratio as follows:

$$\sigma^2 = \frac{N_0}{2} = \frac{E_c}{2R \times SNR}, \tag{33}$$

where $R = K/N$ is the coding rate, and $E_c = RE_b$ is the energy expended per codeword bit. Assuming the energy expended per coded bit is 1, we have

$$\sigma^2 = (2R \times SNR)^{-1}. \tag{34}$$

REFERENCES

[1] S. M. Aji and R. J. McEliece, *The generalized distributive law and free energy minimization*, in Proceedings of the 39th Allerton Conference, 2001, pp. 672–681.

[2] H. A. Bethe, *Statistical theory of superlattices*, Proceedings of the Royal Society of London, 150 (1935), pp. 552–575.

[3] N. L. Biggs, *Discrete Mathematics*, Oxford University Press, revised ed., 1989.

[4] K. P. Bogart, *Introductory Combinatorics*, Academic Press, 2nd ed., 1990.

[5] Stephen Boyd and Lieven Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[6] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley, 1991.

[7] B. J. Frey and D. J. C. MacKay, *A revolution: belief propagation in graphs with cycles*, in Advances in Neural Information Processing Systems, vol. 10, 1997.

[8] R. G. Gallager, *Low-Density Parity-Check Codes*, MIT Press, 1963.

[9] R. Kikuchi, *A theory of cooperative phenomena*, Physical Review, 81 (1951), pp. 988–1003.

[10] D. J. C. Mackay, *Good error-correcting codes based on very sparse matrices*, IEEE Transactions on Information Theory, 45 (1999), pp. 399–431.

[11] R. J. Mceliece and M. Yildirim, *Belief propagation on partially ordered sets*, in Mathematical Systems Theory in Biology, Communications, Computation and Finance, D. Gilliam and J. Rosenthal, eds., 2002, pp. 275–299.

[12] T. K. Moon, *Error Correction Coding: mathematical methods and algorithms*, Wiley-Interscience, 2005.

[13] K. H. Rosen, *Discrete Mathematics and its Applications*, McGraw-Hill, 2007.

[14] G. Strang, *Linear Algebra and its Applications*, Academic Press, second ed., 1980.

[15] Andreas Wächter, *An interior point algorithm for large-scale nonlinear optimization with applications in process engineering*, PhD thesis, 2002.

[16] J. S. Yedidia, W. T. Freeman, and Y. Weiss, *Constructing free-energy approximations and generalized belief propagation algorithms*, IEEE Transactions on Information Theory, 51 (2005), pp. 2282–2312.