Variational Inference in LDPC Codes

Parsa Rangriz*
Department of Physics, Sharif University of Technology, Tehran, Iran

Amir Hossein Boreiri†
Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran
(Dated: February 15, 2022)

Regarding the problem of LDPC codes, we will present a variational approach with the aids of graphical models and iterative algorithms in order to demonstrate a practical way of studying the problem. This can not be done without any knowledge of the background of belief propagation algorithm and the convex optimization. For these reasons, we will first introduce the main concepts of the graphical modes and variational optimization, and then use them in the LDPC codes. At last, we will discuss the quantum version of LDPC codes to focus on the these days' attempts.

Keywords: LDPC codes, belief propagation, variational inference

I. INTRODUCTION

Suppose Alice wants to send a message to Bob. In order to transfer a message, we need to send a code through a channel which may cause some errors. As we know, each message concludes with a sequence of 0 and 1, named bit and an error is some flips of 0 to 1 or 1 to 0. Many works have been done to make a strong code to prevent any misunderstanding between Bob and Alice. These codes, named error-correcting codes, are one of the main aims of information theory. Graphical models bring graph theory and probability theory together in a strong formalism for multivariate statistical modeling. These models have become a focus of research in many statistical, computational and mathematical fields, including information theory, statistical physics, combinatorial optimization, signal and image processing and statistical machine learning. Many problems that arise in specific instances - including the key problems of computing marginals and modes of probability distributions - are best studied in these general settings. One of the most important algorithms in the graphical model is named Sum-Product. This algorithm has been discovered independently in several different contexts: statistical physics (under the name 'Bethe-Peierls approximation'), coding theory (the 'Sum Product' algorithm), and artificial intelligence ('belief propagation'). Here we use 'belief propagation' since it is well-known in today's scientific communication. The central problem in information theory is transmitting information through a channel, represented as a sequence of bits, from Alice (sender) to Bob (receiver). If the channel is noisy, then some of the transmitted bits may be corrupted. To tackle this issue, a natural strategy is to add redundancy to the transmitted bits. Hereafter, by some methods such as maximum likelihood estimation, we can decode the codeword and evaluate the main message that Alice has sent. On the other hand, quantum mechanics was established in the 20th century and has become of the prior fields in physics. Also, quantum has reached information theory and made a new branch, namely quantum information theory. As we discussed above, error-correcting codes are a very wide side of information theory and quantum information theory is not an exception. In this regime, one way is to define stabilizer formalism and use it in error correction. First, we will review the main concepts of classical error correction and graphical model, which is an intuitively model for demonstrating the relation between probability and graph theory. After these preliminaries, we try to understand the Belief Propagation algorithm for evaluating marginal distributions and apply it in decoding problems. Second, we will focus on the mathematical properties of these concepts to tune our process and understand the deep levels of these algorithms and approximations. Then the final aim is to generalize this algorithm in the quantum regime. This allows us to define the quantum belief propagation algorithm for quantum decoding and calculate the partial density operators. In order to work with quantum error-correcting codes, we have to pay attention to non-commutative operators in Hilbert space and change the probability distributions with density operators. Also, we will show that we can define quantum factor graphs, which aid us to solve complex issues. Then, we are able to generalize the belief propagation algorithm for quantum operators.

II. GRAPHICAL MODELS

Graphical models are probability distributions that *factor* according to a graph structure. The specific class of graph structures used and the precise meaning

* Email:rangriz@pm.me
† Email:ah.boreiri@gmail.com

of factor depend on the type of graphical model under consideration. Typically, factorization according to a graph encodes a specific class of conditional independence properties.

1. The class of probability distributions that factors according to a suitably sparse graph is a low-dimensional subclass of the set of all probability distribution over a given domain. This enables concise representation, and efficient learnability.

2. Sparse graph structures often correspond to weak dependencies, or to highly structured dependencies in the corresponding distributions. This leads to efficient algorithms for statistical inference.

Specific families of graphical models include Bayesian network, factor graphs, Markov random fields. These allow to encode in various ways independency statements, and the choice of the most suitable formalism can be important in applications.

A. Bayesian Networks

A *Bayesian network* describes the joint distribution of variables associated to the vertices of a directed acyclic graph $G = (\mathcal{V}, \mathcal{E})$. (The graph is acyclic if it has no directed cycle) In such a graph, we say that a vertex $u \in V$ is a *parent* of v, and write $u \in \pi(v)$, if (u, v) is an edge of G. A random variable X_v is associated with each vertex v of the graph.

In a Bayesian network, the joint distribution of $\{X_v, v \in V\}$ is completely specified by the conditional probability kernels $\{p_v(x_v|\mathbf{x}_{\pi(v)}\}_{v\in V}$, where $\pi(v)$ denotes the set of parents of vertex v, and $\mathbf{x}_{\pi(v)} = \{x_u : u \in \pi(v)\}$. We also denote by $\pi(G)$ the set of vertices that have no parent in G.

Given a directed acyclic graph $G = (\mathcal{V}, \mathcal{E})$, and a probability distribution μ over $\mathcal{X}^V$, we say that μ factors according to the *Bayes network structure* G if it can be written as

$$\mu(\mathbf{x}) = \prod_{v\in\pi(G)} p_v(x_v) \prod_{v\in G\setminus\pi(G)} p_v(x_v|\mathbf{x}_{\pi(v)}) \qquad (1)$$

B. Pairwise Graphical Models

Pairwise graphical models are defined in terms of a simple graph $G = (\mathcal{V}, \mathcal{E})$ with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$. It is convenient to introduce a compatibility function $\psi_i : \mathcal{X} \to \mathbb{R}_+$ for each vertex $i \in \mathcal{V}$, and one $\psi_{ij} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ for each edge $(i, j) \in \mathcal{E}$. The joint distribution of $(X_1, ..., X_n)$, $P(\mathbf{X} = \mathbf{x}) = \mu(\mathbf{x})$ is then defined by

$$\mu(\mathbf{x}) = \frac{1}{Z} \prod_{(i,j)\in\mathcal{E}} \psi_{ij}(x_i, x_j) \prod_{i\in\mathcal{V}} \psi_i(x_i) \qquad (2)$$

A classical example of pairwise model is the *Ising model* from statistical physics. In this case $\mathcal{X} = \{-1, +1\}$, and it is customary to parameterize the potentials in the form

$$\psi_{ij}(x_i, x_j) = \exp\{J_{ij}x_ix_j\}, \quad \psi_i(x_i) = \exp\{h_ix_i\} \quad (3)$$

where J's are the correlation functions and h's are the magnetic field.

The same model is popular in machine learning under the name of *Boltzmann machine*. It includes as special case some toy models for neural networks, such as the *Hopfield model*.

C. Factor Graphs

A factor graph is a bipartite graph $G = (\mathcal{V}, \mathcal{F}, \mathcal{E})$, whereby $\mathcal{V}$ and $\mathcal{F}$ are two sets of vertices, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{F}$ a set of undirected edges. We will often identify $|\mathcal{V}| = n$, $|\mathcal{F}| = m$. We will call *variable nodes* the vertices in $\mathcal{V}$, and use for them letters $i, j, k, ...$, and *function* (or *factor*) *nodes* the vertices in $\mathcal{F}$ to be denoted by $a, b, c, ...$

Given $i \in \mathcal{V}$, the set of its neighbors is denoted by $\partial i = \{a \in F : (i, a) \in \mathcal{E}\}$. The neighborhood of $a \in F$, denoted by ∂a, is defined analogously.

A *factor graph model* specifies the joint distribution of random variables $\mathbf{X} = (X_1, ..., X_n) = \{X_i : i \in \mathcal{V}\}$ taking value in a domain $\mathcal{X}$. As above, we shall assume that $\mathcal{X}$ is a finite set. Indeed we shall mostly focus on the first case and write general equations in discrete notation. Finally, for any subset of variable nodes $\mathcal{A} \subseteq \mathcal{V}$, we write $\mathbf{x}_\mathcal{A} = \{x_i : i \in \mathcal{A}\}$.

The joint distribution μ over $\mathbf{x} \in \mathcal{X}^\mathcal{V}$ factors on the *factor graph* $G = (\mathcal{V}, \mathcal{F}, \mathcal{E})$ if there exists a vector of function $\boldsymbol{\psi} = (\psi_i, ..., \psi_m) = \{\psi_a : a \in \mathcal{F}\}$, $\psi_a : \mathcal{X}^{\partial a} \to \mathbb{R}_+$, and a constant Z such that

$$\mu(\mathbf{x}) = \frac{1}{Z} \prod_{a\in F} \psi_a(\mathbf{x}_{\partial a}) \qquad (4)$$

We then say that the pair (G, μ) is a factor graph model. Equivalently, we will call the triple $(G, \boldsymbol{\psi}, \mathcal{X})$ a factor graph model.

III. BELIEF PROPAGATION ALGORITHM

It is useful to keep in mind three inference problems that can be essentially reduced to reach other. We describe them for factor graphs but little needs to be changed for other types of graphical models.

- **Computing marginals.** Given a subset of vertices $\mathcal{A} \subseteq \mathcal{V}$, compute the probability $\mathbb{P}\{\mathbf{X}_\mathcal{A}\} = x_\mathcal{A} = \mu(\mathbf{x}_\mathcal{A})$.

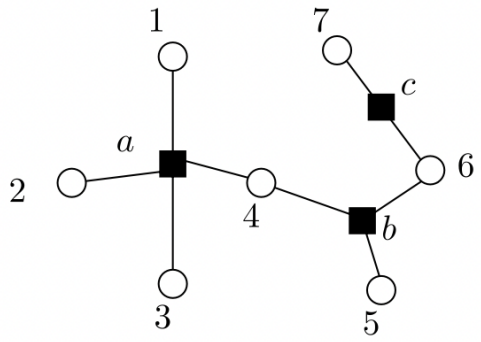

Figure 1. Illustration of factor graph

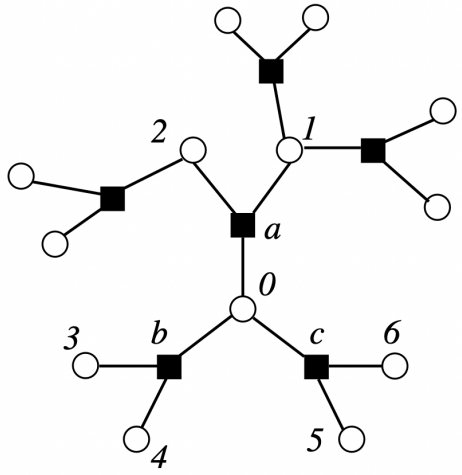

Figure 2. A simple tree factor graph

- **Computing conditional probabilities.** Given two subset $\mathcal{A}, \mathcal{B} \subseteq \mathcal{V}$, compute the conditional probability $\mathbb{P}\{\mathbf{X}_\mathcal{A} = \mathbf{x}_\mathcal{A} | \mathbf{X}_\mathcal{B} = \mathbf{x}_\mathcal{B}\} = \mu(\mathbf{x}_\mathcal{A}|\mathbf{x}_\mathcal{B})$

- **Partition function.** Compute the partition function Z.

In this section, we turn to a description of the basic exact inference algorithms for graphical models discussed above. In computing a marginal probability, we must sum or integrate the joint probability distribution over one or more variables. In order to compute such distribution for X_v, it suffices to choose a specific ordering of the remaining variables and to eliminate variables according to that order. Repeating this operation for each individual variable would yield the full set of marginals. However, this approach is wasteful because it neglects to share intermediate terms in the individual computations. The belief propagation algorithm is essentially dynamic programming algorithm based on a calculus for sharing intermediate terms. The algorithm involve "message-passing" operations on graphs, where the messages are exactly these shared intermediate terms. Upon convergence of the algorithms, we obtain marginal probabilities for all factor nodes of the original graph.

Throughout this chapter we will consider the factor graph model,

$$\mu(\mathbf{x}) = \frac{1}{Z} \prod_{a \in \mathcal{F}} \psi_a(\mathbf{x}_a) \tag{5}$$

defined on the factor graph $G = (\mathcal{V}, \mathcal{F}, \mathcal{E})$, and alphabet $\mathcal{X}$.

Consistently with our discussion of reduction between various inference tasks, we will focus on the problem of computing marginal, i.e. marginal distributions of a small subset of variables, e.g. $\mu(\mathbf{x}_\mathcal{A}) = \mathbb{P}_m u\{\mathbf{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}\}$. Explicitly,

$$\mu_\mathcal{A}(x_\mathcal{A}) = \sum_{\mathbf{x}_{\mathcal{V}\setminus\mathcal{A}}} \mu(\mathbf{x}) \tag{6}$$

To be definite, you can think of simple 'one-point' marginals $\mathcal{A} = \{i\}$. The most popular message-passing algorithm for computing marginals is known as the *sum-product* algorithm.

In the following we will often have to write identities in which the overall normalization of both sides is not really interesting. In order to get rid of all these normalization floating around, it is convenient to use a special notation. Throughout this paper, the symbol $\cong$ is used to denote equality between functions up to a multiplicative normalization.

A. Trees

Inference is easy if the underlying factor graph is a tree. By 'easy' we mean that it can be performed in time linear in the number of nodes, provided the factor nodes have bounded degree and the alphabet size is bounded as well. The algorithm that achieves this goal is a simple 'dynamic programming' procedure. Let us see how this works on the simple example reproduced in Fig. 2. We begin by assuming that we want to compute the marginal at node 0:

$$\mu(x_0) \cong \sum_{\mathbf{x}_{\mathcal{V}\setminus 0}} \prod_{l in \mathcal{F}} \psi_l(\mathbf{x}_l) \tag{7}$$

Node 0 is the root of three subtrees of G, that we can distinguish by the name of the factor node neighbor of 0 that they contain,

namely $G_{a\to 0} = (\mathcal{V}_{a\to 0}, \mathcal{F}_{a\to 0}, \mathcal{E}_{a\to 0}), G_{b\to 0} = (\mathcal{V}_{b\to 0}, \mathcal{F}_{b\to 0}, \mathcal{E}_{b\to 0}), G_{c\to 0} = (\mathcal{V}_{c\to 0}, \mathcal{F}_{c\to 0}, \mathcal{E}_{c\to 0})$. We can rewrite the sum as

$$\mu(x_0) \cong \mu_{a\to 0}(x_0)\mu_{b\to 0}(x_0)\mu_{c\to 0}(x_0) \tag{8}$$

This we reduced the problem of computing a marginal with respect to G to the ones of computing marginals with respect to subgraphs of G. We can repeat this recursively. Consider the subtree $G_{a\to 0}$. This can be decomposed into the factor node a, plus the subtrees $G_{1\to a}$ and $G_{2\to a}$. Using again the distributive property we have

$$\mu_{a\to 0}(x_0) \cong \sum_{x_1,x_2} \psi_a(\mathbf{x}_a)\mu_{1\to a}(x_1)\mu_{2\to a}(x_2) \tag{9}$$

It is quite clear that our arguments did not use the specific structure of the factor graph in Fig. 2. But they instead hold for any tree. Namely given a tree G and a directed edge $a \to i$ (factor-to-variable) or $i \to a$ (variable-to-factor) we can define the subgraphs $G_{a\to i}$ or $G_{i\to a}$ as above and the corresponding 'partial marginals' of the variable: $i : \mu_{a\to i}(x_i)$ and $\mu_{i\to a}(x_i)$. We will also call these *messages*.

For a *tree fator graph*, the 'partial marginals' are the unique solution of the equations

$$\mu_{j\to a}(x_j) \cong \prod_{b\in\partial j\setminus a} \mu_{b\to j}(x_j) \tag{10}$$

$$\mu_{a\to j}(x_j) \cong \sum_{\mathbf{x}_{\partial a\setminus j}} \psi_a(\mathbf{x}_{\partial a}) \prod_{k\in\partial a\setminus j} \mu_{k\to a}(x_k) \tag{11}$$

For all $(i, a) \in \mathcal{E}$.

B. The sum-product algorithm

The sum product algorithm is an iterative message passing algorithm. The basic variables are 'messages' which are probability distributions over $\mathcal{X}$. These are also called 'beliefs'. Two such distributions are used for each edge in the graph $\nu_{i\to a}(\cdot)$ (variables to factor node) $\nu_{a\to i}(\cdot)$ (factor to variable node). We shall denote the vector of *messages* by $\boldsymbol{\nu} = \{\nu_{i\to a}, \nu_{a\to i}\}$: it is a vector of probability distributions, indexed by directed edges in G.

We shall indicate the iteration number by superscripts, e.g. $\nu^{(t)}_{i\to a}$, $\nu^{(t)}_{a\to i}$ are the messages value after t iterations. Messages are initialized to some non-informative values, typically $\nu^{(0)}_{i\to a}$, $\nu^{(0)}_{a\to i}$ are equal to the uniform distribution over $\mathcal{X}$.

Various update scheduling are possible, but for the sake of simplicity we will consider parallel updates, which read:

$$\nu^{(t+1)}_{j\to a} \cong \prod_{b\in\partial j\setminus a} \nu^{(t)}_{b\to j}(x_j) \tag{12}$$

$$\nu^{(t)}_{a\to j}(x_j) \cong \sum_{\mathbf{x}_{\partial a\setminus j}} \psi_a(\mathbf{x}_{\partial a}) \prod_{k\in\partial a\setminus j} \nu^{(t)}_{k\to a}(x_k) \tag{13}$$

After t iterations, one can estimate the marginal distribution $\mu(x_i)$ of variable i using the set of all incoming messages. The BP estimate is

$$\nu^{(t)}_{i}(x_i) \cong \prod_{a\in\partial i} \nu^{(t-1}_{a\to i}(x_i) \tag{14}$$

The exact solution of fixed point is just reliable for tree factor graphs. On general graphs, the resulting fixed points will not be necessarily marginals of μ, but the hope is that they are nevertheless a good approximation of the actual marginals.

IV. LDPC CODES

Low-density parity-check (LDPC) error-correcting codes were introduced in 1963 by Robert Gallager in his PhD thesis. The basic motivation came from the observation that random linear codes had excellent theoretical performance but were unpractical. In particular, no efficient algorithm was known for decoding. In retrospect, this not surprising, since it was later shown that decoding for linear codes is an NP-hard problem.

The idea was then to restrict the random linear code ensemble, introducing some structure that could be exploited for more efficient decoding. Of course, the risk is that such a restriction of the ensemble might spoil its performance. Gallager's proposal was simple and successful: LDPC codes are among the most efficient codes around. For this purpose, first we define LDPC codes and LDPC code ensembles, then we discuss the belief propagation decoding for such codes.

A. Linear Codes

First we donte a code by its codebook $\mathcal{C}$, which is a subset of $\{0,1\}^N$. LDPC codes are *linear codes*, which means that the codebook is a linear subsapce of $\{0,1\}^N$. In practice, such a subspace can be specified through an $M\times N$ matrix $\mathbb{H}$, with binary entries $\mathbb{H}_{ij} \in \{0,1\}$, where $M < N$. The codebook is defined as the kernel of $\mathbb{H}$:

$$\mathcal{C} = \{\mathbf{x} \in \{0,1\}^N : \mathbb{H}\mathbf{x} = 0\} \tag{15}$$

Here and in all of this paper, the multiplications and sums involved in $\mathbb{H}\mathbf{x}$ are understood as being computed

modulo 2. The matrix $\mathbb{H}$ is called the *parity check matrix* of the code.

Given such a code, encoding can always be implemented as a linear operation. There exists an $N\times N-M$ binary matrix $\mathbb{G}$, called the *generator matrix*, such that the codebook is the image of $\mathbb{G}$: $\mathcal{C} = \{\mathbf{x} = \mathbb{G}\mathbf{z}, \mathbf{z} \in \{0,1\}^{N-M}$. Encoding is therefore realized as the mapping $\mathbf{z} \to \mathbf{x} = \mathbb{G}\mathbf{z}$.

B. Word and Symbol MAP

In this part we introduce two more popular methods used in decoding. Both of them are related to *MAP (maximum a posteriori probability)* and we call them *word MAP* and *symbol MAP*.

Suppose a memory-less channel with a transition probability $Q(\mathbf{y}|\mathbf{x})$, this probability has an explicit expression as a consequence of the Bayes rule:

$$\mathbb{P}(\mathbf{x}|\mathbf{y}) = \frac{1}{Z(\mathbf{y})}\prod_{i=1}^{N} Q(y_i|x_i)\mathbb{P}(\mathbf{x}) \tag{16}$$

where Z is the normalization factor.

The most obvious design criterion applicable to the design of a decoder is the minimum probability of error criterion. When the design criterion is to minimize the probability that the decoder fails to decode to the correct codeword, i.e., to minimize the probability of a codeword error, it can be shown that this is equivalent to maximizing the posterior probability $\mathbb{P}(\mathbf{x}|\mathbf{y})$. The optimal decision is then given by

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} \mathbb{P}(\mathbf{x}|\mathbf{y}) \tag{17}$$

This decision is called the *word MAP (maximum a posteriori)* role.

Finding the codeword $\mathbf{x}$ is very complex except for very simple codes such as the Hamming code. Thus, ML decoding algorithms have been developed that exploit code structure, vastly reducing complexity. Suboptimal but less complex algorithms, which performs slightly worse than the MAP decoded are based on the bit-wise MAP which minimizes the probability of bit error rather than the probability of codeword error. This *symbol MAP* criterion is

$$\hat{\mathbf{x}} = \left(\arg\max_{x_1} \mathbb{P}(x_1|\mathbf{y}), ..., \arg\max_{x_n} \mathbb{P}(x_n|\mathbf{y})\right) \tag{18}$$

C. Classical BP Decoding

In previous section, we described the factor graph associated with one particular linear code. The recipe to build the factor graph, knowing $\mathbb{H}$, is as follows. Let us denote by $i_1^a, ..., i_{k(a)}^a \in \{1, ..., N\}$ the column indices such that $\mathbb{H}$ has a matrix element equal to 1 at row a and column i_j^a. The a-th coordinate of the vector $\mathbb{H}\mathbf{x}$ is then equal to $x_{i_1^a} \oplus ... \oplus x_{i_{k(a)}^a}$. Let $\mu_{0,\mathbb{H}}(\mathbf{x})$ be the uniform distribution over all codewords of the code $\mathbb{H}$. It is given by

$$\mu_{0,\mathbb{H}}(\mathbf{x}) = \frac{1}{Z}\prod_{a=1}^{M} \mathbb{I}(x_{i_1^a} \oplus \cdots \oplus x_{i_{k(a)}^a} = 0) \tag{19}$$

Therefore, the factor graph associated with $\mu_{0,\mathbb{H}}(\mathbf{x})$ includes N variable nodes, one for each column of $\mathbb{H}$, and M function nodes (also called, in this context, *check nodes*), one for each row. A factor node and a variable node are joined by an edge if the corresponding entry in $\mathbb{H}$ is non-vanishing. Clearly, this procedure can be inverted: with any factor graph with N variable nodes and M function nodes, we can associate an $M \times N$ binary matrix $\mathbb{H}$, the *adjacency matrix* of the graph, whose non-zero entries correspond to the edges of the graph.

Now again suppose a *binary-input, output-symmetric, memoryless (BMS) channel.* THis is a channel in which the transmitted codeword is binary, $\mathbf{x} \in \{0,1\}^N$, and the output $\mathbf{y}$ is a sequence of N letters y_i from an alphabet $\mathcal{Y} \subset \mathbb{R}$. The probability of recieving letter y when bit x is sent, $Q(y|x)$, possesses the symmetry property $Q(y|0) = Q(-y|1)$.

Let us suppose that an LDPC error-correcting code is used in this communication. The conditional probability for the channel input is given by

$$\mu_y(\mathbf{x}) = \frac{1}{Z(\mathbf{y})}\prod_{i=1}^{N} Q(y_i|x_i)\prod_{a=1}^{M} \mathbb{I}(x_{i_1^a} \oplus \cdots \oplus x_{i_{k(a)}^a} = 0) \tag{20}$$

where, $\mu_y(\mathbf{x}) = \mathbb{P}(\mathbf{x}|\mathbf{y})$.

The factor graph associated with this distribution is the usual one: an edge joins a variable node i to a check node a whenever the variable x_i appears in the a-th parity check equation.

Messages $\nu_{i\to a}(x_i)$, $\nu_{a\to i}(x_i)$ are exchanged along the edges. We shall assume a parallel updating of BP messages,

$$\nu_{i\to a}^{(t+1)}(x_i) \cong Q(y_i|x_i)\prod_{b\in\partial i\setminus a} \nu_{b\to i}^{(t)}(x_i) \tag{21}$$

$$\nu_{a\to i}^{(t)}(x_i) \cong \sum_{\{x_j\}} \mathbb{I}(x_{i_1^a} \oplus \cdots \oplus x_{i_{k(a)}^a} = 0)\prod_{j\in\partial a\setminus i} \nu_{j\to a}^{(t)}(x_j) \tag{22}$$

The BP estimate for the marginal distribution at node i and time t, also called the 'belief' or *soft decision*, is

$$\nu_i^{(t)}(x_i) \cong Q(y_i|x_i)\prod_{b\in\partial i} \nu_{b\to i}^{(t-1)}(x_i) \tag{23}$$

Based on this estimate, the optimal BP decision for bit i at time t, sometimes called the *hard decision*, is

$$\hat{x_i}^{(t)} = \arg\max_{x_i} \nu_i^{(t)}(x_i) \tag{24}$$

V. FREE ENERGY APPROXIMATION

In this section, we saw that the BP algorithm can be defined in terms of the belief equations. We shall eventually show that these belief equations correspond to the stationarity conditions for a function of the beliefs called the Bethe free energy. minimizing the Bethe free energy is a sensible approximation procedure that has a long and successful history in physics. It also points to a variety of ways to improve upon or generalize BP, especially by improving upon the approximations used in the Bethe free energy. Suppose that one has a system of N particles, each of which can be in one of a discrete number of states. The overall state of the system will be denoted by the vector $x = (x1, .., xn)$. Each state of the system has a corresponding energy. A fundamental result of statistical mechanics is that, in thermal equilibrium, the probability of a state will be given by *Boltzmann's law*

$$p(\boldsymbol{x}) = \frac{1}{Z(T)} e^{-E(\boldsymbol{x})/T} \tag{25}$$

Here, $T$is the temperature, and $Z(T)$ is simply a normalization constant, known as the partition function

$$Z(T) = \sum_{\boldsymbol{x} \in S} e^{-E(\boldsymbol{x})/T} \tag{26}$$

Where S is the space of all possible states of the system. We shall set $T = 1$ throughout the rest of this section.

For the case of a factor graph probability distribution function

$$p(\boldsymbol{x}) = (1/Z) \prod_{a=1}^{M} \psi_a(\boldsymbol{x}_a) \tag{27}$$

we therefore define the *energy*

$$E(\boldsymbol{x}) = -\sum_{a=1}^{M} \log \psi_a(\boldsymbol{x}_a) \tag{28}$$

The *Helmholtz free energy* of a system is $F_H = -\log Z$.

One important technique is based on a variational approach. Suppose again that $p(x)$ is the true probability distribution of the system, and obeys Boltzmann's law $p(x) = e^{-E(x)}/Z$. It may be that even if we know $p(x)$ exactly, it is of a form that makes the computation of F_H difficult. We therefore introduce a "trial" probability distribution $b(x)$, and a corresponding *variational free energy* (this quantity is also sometimes called the *Gibbs free energy)* defined by

$$F(b) = U(b) - H(b) \tag{29}$$

$$H(b) = -\sum_{\mathbf{x} \in S} b(\mathbf{x}) \log b(\mathbf{x}) \tag{30}$$

$$U(b) = \sum_{\mathbf{x} \in S} b(\mathbf{x}) E(\mathbf{x}) \tag{31}$$

The distance between free energy and Gibbs free energy is

$$F(b) = F_H + D(b \| p) \tag{32}$$

Where D is the Kullback–Leibler divergence between b and p .So by calculating $F(b)$ we can find an upper bound for F_H. Minimizing the variational free energy $F(b)$ with respect to trial probability functions $b(x)$ is therefore an exact procedure for computing F_H and recovering $p(x)$. Of course, as N becomes large, this procedure is also totally intractable, as $b(x)$ will take exponentially large memory just to store. A more practical possibility is to upper-bound F_H by minimizing $F(b)$ over a restricted class of probability distributions. This is the basic idea underlying the mean field approach.

A. Naive Mean Field Approximation

One very popular mean-field form for $b(x)$ is the factorized form

$$b_{\mathrm{MF}}(\boldsymbol{x}) = \prod_{i=1}^{N} b_i(x_i) \tag{33}$$

we can easily compute the mean field free energy F_{MF} for an arbitrary factor graph

$$H_{\mathrm{MF}}(\{b_1, \ldots, b_N\}) = -\sum_{i=1}^{N} \sum_{x_i} b_i(x_i) \ln b_i(x_i) \tag{34}$$

$$U_{\mathrm{MF}}(\{b_1, \ldots, b_N\}) = -\sum_{a=1}^{M} \sum_{\boldsymbol{x_a}} \ln \psi_a(\boldsymbol{x_a}) \prod_{i \in \partial a} b_i(x_i) \tag{35}$$

B. Region-Based Approximation

We define a region R of a factor graph to be a set $\mathcal{V}_R$ of variable nodes and a set of $\mathcal{F}_R$ of factor nodes, such

that if a factor node a belongs to $\mathcal{F}_R$, all the variable nodes neighboring a are in $\mathcal{V}_R$.

For any region R , we define region energy $E_R(x_R)$, the region average energy $U_R(b_R)$, the region entropy $H_R(b_R)$, and the region free $F_R(b_R)$ energy , by

$$E_R\left(\boldsymbol{x}_R\right) = -\sum_{a \in A_R} \ln \psi_a\left(\boldsymbol{x}_a\right) \tag{36}$$

$$U_R\left(b_R\right) = \sum_{\boldsymbol{x}_R} b_R\left(\boldsymbol{x}_R\right) E_R\left(\boldsymbol{x}_R\right) \tag{37}$$

$$H_R\left(b_R\right) = -\sum_{\boldsymbol{x}_R} b_R\left(\boldsymbol{x}_R\right) \ln b_R\left(\boldsymbol{x}_R\right) \tag{38}$$

$$F_R\left(b_R\right) = U_R\left(b_R\right) - H_R\left(b_R\right) \tag{39}$$

The intuitive idea behind a region-based free energy approximation is that we will try to break up the factor graph into a set of large regions that include every factor and variable node, and say that the overall free energy is the sum of the free energies of all the regions. Of course, if some of the large regions overlap, then we will have erred by counting the free energy contributed by some nodes two or more times, so we then need to subtract out the free energies of these overlap regions in such a way that each factor and variable node is counted exactly once.

We define a *region-based approximate entropy* and the *region-based average energy* and the *region-based free energy* by

$$H_{\mathcal{R}}\left(\{b_R\}\right) = \sum_{R \in \mathcal{R}} c_R H_R\left(b_R\right) \tag{40}$$

$$F_{\mathcal{R}}\left(\{b_R\}\right) = U_{\mathcal{R}}\left(\{b_R\}\right) - H_{\mathcal{R}}\left(\{b_R\}\right) \tag{41}$$

$$U_{\mathcal{R}}\left(\{b_R\}\right) = \sum_{R \in \mathcal{R}} c_R U_R\left(b_R\right) \tag{42}$$

where c_R are normalization constants in order to count each vertex one time.

We say that a set of regions $\mathcal{R}$ and counting number c_R give a valid region-based approximation when, for every factor node a and every variable node i in the factor graph

$$\sum_{R \in \mathcal{R}} c_R \mathbb{I}_{\mathcal{F}_R}(a) = \sum_{R \in \mathcal{R}} c_R \mathbb{I}_{\mathcal{V}_R}(i) = 1 \tag{43}$$

VI. EXPONENTIAL FAMILIES AND MAXIMUM ENTROPY

A. Definition and Motivation

In this section, we describe how many graphical models are naturally viewed as exponential families, a broad class of distributions that have been extensively studied in the statistics literature. Taking the perspective of exponential families illuminates some fundamental connections between inference algorithms and the theory of convex analysis.

Suppose that given n independent and identically distributed (i.i.d.) observations $X_1, ..., X_n$, we compute the empirical expectations of certain functions — namely, the quantities

$$\hat{\mu}_\alpha := \frac{1}{n} \sum_{i=1}^{n} \phi_\alpha\left(X_i\right), \text{ for all } \alpha \in \mathcal{J} \tag{44}$$

where each α in some set $\mathcal{J}$ indexes a function $\phi_\alpha : X \to \mathbb{R}$. Our goal is to infer a full probability distribution over the random variable X. For a given distribution p, let us consider the expectations

$$\mathbb{E}_p\left[\phi_\alpha(X)\right] := \int_X \phi_\alpha(x) p(x) v(dx) \text{ for } \alpha \in I \tag{45}$$

We say that the distribution p is consistent with the data if

$$\mathbb{E}_p\left[\phi_\alpha(X)\right] = \hat{\mu}_\alpha \text{ for all } \alpha \in \mathcal{J} \tag{46}$$

We need a principle to choose among all the probability distributions that are consistent with the observations, the principle of maximum entropy is to choose, from among the distributions consistent with the data, the distribution p^* whose Shannon entropy is maximal. So p^* is given by the solution to the following constrained optimization problem:

$$p^* := \arg\max_{p \in \mathcal{P}} H(p) \text{ s.t. } \mathbb{E}_p\left[\phi_\alpha(X)\right] = \hat{\mu}_\alpha, \forall \alpha \in \mathcal{J} \tag{47}$$

it can be shown by calculus of variations in the general continuous case, and by ordinary calculus in the discrete case that the optimal solution p^* takes the form

$$p_\theta(x) \propto \exp\left\{\sum_{\alpha \in \mathcal{J}} \theta_\alpha \phi_\alpha(x)\right\} \tag{48}$$

where $\theta \in \mathbb{R}^d$ represents a parameterization of the distribution in exponential family form.

Given a random vector $(X_1, X_2, \ldots, X_m)$ taking values in somespace

$$\mathcal{X}^m = \bigotimes_{s=1}^{m} \mathcal{X}_s \tag{49}$$

Let $\phi = (\phi_\alpha, \alpha \in I)$ be a collection of functions $\phi_\alpha : \mathcal{X}^m \to \mathbb{R}$, known either as potential functions or sufficient statistics.

With this notation, the exponential family associated with ϕ consists of the following parameterized collection of density functions

$$p_\theta(x_1, x_2, \ldots, x_m) = \exp\{\langle \theta, \phi(x)\rangle - A(\theta)\} \tag{50}$$

The quantity A, known as the log partition function or cumulant function, is defined by the integral

$$A(\theta) = \log \int_{\mathcal{X}^m} \exp\langle \theta, \phi(x)\rangle v(dx) \tag{51}$$

B. Application

Now we use Exponential Families in the problem of error-control coding. To motivate the coding problem, suppose that Alice and Bob wish to communicate. they communicate using strings $(x_1, x_2, ..., x_m)$ of bits and moreover, they agree to use only a subset of the total number 2^m of length m binary strings. The communication channel is a "bit-flipping" channel, and this channel can be modeled by the conditional distribution

$$p(y \mid x) := \begin{cases} 1-\epsilon & \text{if } x = y \\ \epsilon & \text{if } x \neq y \end{cases} \tag{52}$$

where $x \in \{0, 1\}$ represents the bit transmitted by Alice, and $y \in \{0, 1\}$ represents the bit received by Bob.

Let us consider a collection F of such parity checks; each $a \in \mathcal{F}$ enforces a parity check constraint on some subset $\partial a \subset \{1, ..., m\}$ of bits. We define the indicator function

$$\psi_a\left(x_{N(a)}\right) := \begin{cases} 1 & \text{if } \bigoplus_{i\in\partial a} x_i = 0 \\ 0 & \text{otherwise} \end{cases} \tag{53}$$

Bobs goal is to use the received bits to infer which code word was transmitted by Alice. Depending on the error metric used, this decoding problem corresponds to either computing marginal or modes of the posterior distribution

$$p(x_1, \ldots, x_m \mid y_1, \ldots, y_m) \propto \prod_{i=1}^{m} p(y_i \mid x_i) \prod_{a\in\mathcal{F}} \psi_a(x_{\partial a}) \tag{54}$$

This distribution can be described by a factor graph, with the bits x_i represented as unshaded circular variable nodes, the observed values y_i as shaded circular variable nodes, and the parity check indicator functions represented at the square factor nodes. since the quantities y_i are observed, they may be viewed as fixed, so that the conditional distribution $p(y_i|x_i)$ can be viewed as some function $q(\cdot)$ of x_i only. A little calculation shows that we can write this function in exponential form as

$$q(x_i; \theta_i) = \exp\{\theta_i x_i - \log(1 + \exp(\theta_i))\} \tag{55}$$

where the exponential parameter θ_i is defined by the observation y_i and conditional distribution $p(y_i|x_i)$ via the relation

$$\theta_i = \log p(y_i \mid 1)/p(y_i \mid 0) \tag{56}$$

. In the particular case of the binary symmetric channel, these canonical parameters take the form

$$\theta_i = (2y_i - 1)\log\frac{1-\epsilon}{\epsilon} \tag{57}$$

With this set-up, the $p_\theta(x|y)$ distribution can be cast as an exponential family, where the density has the form

$$p_\theta(x \mid y) \propto \exp\left\{\sum_{i=1}^{m} \theta_i x_i\right\} \tag{58}$$

VII. QUANTUM THEORY

After the 20th century activities, quantum physics raised and involved most of the previous theories of classical physics. Information theory was not a specific case to avoid from this revolution. In late 1980's, quantum information theory was established and tried to fix most problems of what scientists observed in early 1920.

This new theory needs a mathematical space for calculation, and the *Hilbert space* is the best case that physicists and mathematicians found. In this regime, most real parameters changed to operations of the Hilbert space and the quantum theory categorized into three parts. State preparation, evolution, and measurement.

The state of any system represented by a vector $|\psi\rangle \in \mathcal{H}$, where $\mathcal{H}$ is the associated Hilbert space. The simplest quantum system is a two-state system: a physical qubit. Let $|0\rangle$ denote one possible state of the system and $|1\rangle$ denote another possible sate of the qubit. We can encode a classical bit into a qubit with the following mapping:

$$0 \to |0\rangle, \quad 1 \to |1\rangle \tag{59}$$

The quantum theory predicts that the above states are not the only possible states of a qubit. Arbitrary *superpositions* of the above states are possible as well because the quantum theory is a linear theory. Then this state is given by

$$|\psi\rangle \equiv \alpha|0\rangle + \beta|1\rangle \tag{60}$$

where the coefficients α and β are arbitrary complex numbers with unit norm: $|\alpha|^2 + |\beta|^2 = 1$. The coefficients α and β are *probability amplitudes* - they are not probabilities themselves, but they do allow us to calculate probabilities.

But in general, we may not know for certain whether we possess a particular quantum state. Instead, we may only have a probabilistic description of an ensemble of quantum states. The *density operator* formalism is a powerful mathematical tool for describing this scenario. We generally may not have perfect knowledge of a prepared quantum state. Suppose a third party, Bob, prepares a state for us and only gives us a probabilistic description of it. That is, we might only know that Bob selects the state $|\psi_x\rangle$ with a certain probability $\mathcal{P}_X(x)$. Our description of the state is then as an ensemble of quantum states $\{\mathcal{P}_X(x), |\psi_x\rangle\}$. Then the density operator is defined as follows,

$$\rho \equiv \sum_{x\in\mathcal{X}} \mathbb{P}_X(x) \, |\psi_x\rangle \langle\psi_x| \tag{61}$$

We can equivalently write the density operator,

$$\rho = \mathbb{E}_X\{|\psi_X\rangle \langle\psi_X|\} \tag{62}$$

where the expectation is with respect to the random variable X.

Suppose V is a set of subsystems, the the joint state is a density operator ρ_V. For $U \subset V$, then analog of a marginal distribution is the reduced state obtained by a *partial trace* over $V \setminus U$. i.e.

$$\rho_U = \mathrm{tr}_{V\setminus U} \, \rho_V \tag{63}$$

In classical probability, the conditional probability of an event B, given an event A is defined as

$$\mathbb{P}(B|A) = \frac{P(A \cap B)}{P(A)} \tag{64}$$

Further, a state ρ_{AB} can be written in terms of its diagonal components $(\rho_{AB})_{jk,jk}$ as

$$\rho_{AB} = \sum_{jk} (\rho_{AB})_{jk,jk} \, |j\rangle \langle j|_A \otimes |k\rangle \langle k|_B \tag{65}$$

and the reduced state on system A is given by $\rho_A = \mathrm{tr}_B(\rho_{AB}$. Now, the conditional probability that system B is in state $|k\rangle_B$, given that system A is in the state $|j\rangle_A$ is given by $(\rho_A)^{-1}_{j,j}(\rho_{AB})_{jk,jk}$. This can be written as a matrix of conditional probabilities, given by

$$(\rho_{B|A})_{jk,jk} = \frac{(\rho_{AB})_{jk,jk}}{(\rho_A)_{j,j}} \tag{66}$$

or in the operator notation

$$\rho_{B|A} = (\rho_A^{-1} \otimes I_B)\rho_{AB} \tag{67}$$

We can alternatively use the following equation

$$\rho_{B|A} = \lim_{n\to\infty} \left((\rho_A^{-1/2n} \otimes I_B)\rho_{AB}^{1/n}(\rho_A^{-1/2n} \otimes I_B)\right)^n \tag{68}$$

Thus it is convenient to define a family of products for pairs of operators A, B as follows,

$$A \odot B = \lim_{n\to\infty} \left(A^{-1/2n} B^{1/n} A^{-1/2n}\right)^n \tag{69}$$

Therefore, we have

$$\rho_{B|A} = (\rho_A \otimes I_B) \odot \rho_{AB} \tag{70}$$

VIII. QUANTUM FACTOR GRAPH AND BP ALGORITHM

A quantum factor graph consists of a pair (G, ρ_V), where $G = (\mathcal{V}, \mathcal{F}, \mathcal{E})$ is a bipartite graph and ρ_V is a quantum state. A bipartite graph is an undirected graph for which the set of vertices can be partitioned into two disjoint sets, $\mathcal{V}$ and $\mathcal{F}$, such that $(v, f) \in \mathcal{E}$ only if $v \in \mathcal{V}$ and $f \in \mathcal{F}$. Each variable node v is associated with a quantum system, also labeled v, with a Hilbert space $\mathcal{H}_v$, and ρ_V is a state on $\bigotimes_{v\in\partial f} \mathcal{H}_v$. The state associated with a factor graph is of the form

$$\rho_V \cong \prod_{f\in\mathcal{F}} X_f \odot \bigotimes_{v\in V} \mu_v \tag{71}$$

where μ_v is an operator on $\mathcal{H}_v$, X_f is an operator on $\mathcal{H}_f$ and $[X_f, X_g] = 0$.

Again, suppose a tree which we want to compute the partial density operator at node (qubit) 0. First, we have to write

$$\begin{aligned}
\rho_V &\cong \prod_{f\in\mathcal{F}} X_f \odot \bigotimes_{v\in V} \mu_v \\
&\cong \left(\prod_{f\in\mathcal{F}_{a\to 0}} X_f \odot \bigotimes_{v\in\mathcal{V}_{a\to 0}} \mu_v\right) \\
&\otimes \left(\prod_{f\in\mathcal{F}_{b\to 0}} X_f \odot \bigotimes_{v\in\mathcal{V}_{b\to 0}} \mu_v\right) \\
&\otimes \left(\prod_{f\in\mathcal{F}_{c\to 0}} X_f \odot \bigotimes_{v\in\mathcal{V}_{c\to 0}} \mu_v\right)
\end{aligned} \tag{72}$$

Now, from the above equation, we can conclude that

$$\rho_0 \cong \rho_{a\to 0}\rho_{b\to 0}\rho_{c\to 0} \tag{73}$$

Similarly, we can write

$$\rho_{a\to 0} \cong \mathrm{tr}_{\partial a\setminus 0} \, X_a \odot \rho_{1\to a}\rho_{2\to a} \tag{74}$$

For a tree factor graph, the partial density operators are the unique solution of the equations,

$$\rho_{i\to a} \cong \prod_{b\in\partial i\setminus a} \rho_{b\to i} \tag{75}$$

$$\rho_{a\to i} \cong \mathrm{tr}_{\partial a\setminus i}\left(X_a \odot \prod_{k\in\partial a\setminus i} \rho_{k\to a}\right) \tag{76}$$

for all $(i,a)\in\mathcal{E}$.

The BP algorithm can be generalized as follows,

$$\rho^{(t+1)}_{i\to a} \cong \prod_{b\in\partial i\setminus a} \rho^{(t)}_{b\to i} \tag{77}$$

$$\rho^{(t)}_{a\to i} \cong \mathrm{tr}_{\partial a\setminus i}\left(X_a \odot \prod_{k\in\partial a\setminus i} \rho^{(t)}_{k\to a}\right) \tag{78}$$

After $t < t_{\max}$ iterations, one can estimate

$$\rho^{(t)}_{i} \cong \prod_{a\in\partial i} \rho^{(t-1)}_{a\to i} \tag{79}$$

IX. CONCLUSION

In this note we presented a graphical approach based on the variational methods to study the LDPC codes which is a basic problem in communication theory. We showed for any memory-less channel there is an iterative algorithm to compute the decoded message of LDPC codes with the aids of a message-passing algorithm, called belief propagation. This algorithm is then exact for tree-like sparse graphs and in order to generalize this method for any non-tree graphs with some loops, we introduced two methods. First, the free energy approximation, which has a statistical physics notion of estimating the messages by studying the energy expansion of a given graph. Second, we represented a variational optimization for the posterior probabilities of codes with using the convex optimization and the exponential families interpretation.

Moreover, we discussed about the quantum version of LDPC codes which is a new field of study in these days. We defined a Hilbert space associated with the code and the matrix based literature for studying the quantum codes. There are some issues and problems which can make difficulties in order to tackle to the LDPC code problems, especially with variational methods. However, this paper tried to have a small look at this quantum problem for the sake of making attentions of the scientists of physics, electrical engineering, computer science, and communication science.

[1] M. J. Wainwright, M. I. Jordan, *Graphical Models, Exponential Families, and Variation Inference*, Foundation and Trends in Machine Learning, Vol. 1. Nos. 1-2 (2008)

[2] M. Mézard, A. Montanari, *Information, Physics, and Computation*, (Oxford University Press, 2009).

[3] W. E. Ryan, S. Lin, *Channel Codes: Classical and Moder*, (Cambridge University Press, 2009).

[4] J. S. Yedidia, W. T. Freeman, Y. Weiss, *Constructing Free-Energy Approximations and Generalized Belief Propagation Algorithms*, (IEEE Transaction on Information Theory, Vol. 51, No. 7, July 2005)

[5] M. M. Wilde, *Quantum Information Theory - 2nd Edition*, (Cambridge University Press, 2017).

[6] M. A. Nielsen, I. L. Chuang, *Quantum Computation and Quantum Information*, (Cambridge University Press, 2010).

[7] M. S. Leifer, *Conditional Density Operators and the Subjectivity of Quantum Operations*, (arxiv:quant-ph/0611233, 2006).

[8] M. S. Liefer, D. Poulin, *Quantum Graphical Models and Belief Propagation*, (arxiv:quant-ph/07081337, 2007).

[9] N. P. Breuckmann, J. N. Eberhardt, *Quantum LDPC Codes*, (PRX Quantum 2 (4), 040101, 2021)