QUANTUM INFORMATION and QUANTUM COMPUTATION THEORY

A Quick Guide

Huan Q. Bui

Colby College

PHYSICS & MATHEMATICS
Statistics

Class of 2021

March 17, 2019

Preface

Greetings,

Quantum Information Theory, A Quick Guide to is compiled based on my own exploration of *Quantum Computation and Quantum Information* by Nielsen and Chuang and John Preskill's Caltech Physics 219/Computer Science 219 lecture notes. There will also be additional elements and structures from *The Theory of Quantum Information* by John Watrous of University of Waterloo, *The Theory of Error-Correcting Codes* by F.J. MacWilliams, N.J.A. Sloane, and other resources that can be found online. *Quantum Information Theory, A Quick Guide to* requires some understanding of linear algebra, matrix analysis, probability theory, and of course quantum mechanics. There will be review chapters on these topics, but some familiarity is expected.

The development of this text comes in different layers. The first layer, that I'm building now in February 2019 and perhaps a better part of my junior year (Fall 2019 to Spring 2020), will on the key concepts and the grand scheme of things. This will be somewhat a survey of what quantum information and quantum computation are and introductory topics in these subjects. Ideas and content will be derived from Nielsen and Chuang's text and Preskill's notes. The later waves of development will focus more on the (numerous) theoretical aspects of quantum information as explored in Watrous' book.

Enjoy!

Contents

Preface 1

1 Mathematical Preliminaries **3**
1.1 Linear Algebra & Matrix Analysis 4
1.2 Analysis & Probability Theory 5

2 Classical Coding Theory **6**
2.1 History 6
2.1.1 Hamming 6
2.1.2 Shannon 6
2.2 Classical Linear Codes 6
2.3 Generator Matrix 6
2.4 Parity-Check Matrix 7
2.5 Errors in Classical Codes and The Syndrome of an Error 7
2.6 Distance of a code 8
2.7 Dual code 8
2.8 Finding Generator and Parity-Check Matrices 8
2.9 An example 8

3 Quantum Mechanical Preliminaries **9**

4 Introduction to Quantum Computation and Information **10**
4.1 Quantum bits 11
4.2 Quantum computation 12
4.3 Quantum algorithms 13
4.4 Experimental Quantum Information Processing 14
4.5 Quantum Information 15

1 Mathematical Preliminaries

1.1 Linear Algebra & Matrix Analysis

1.2 Analysis & Probability Theory

2 Classical Coding Theory

2.1 History

2.1.1 Hamming

2.1.2 Shannon

2.2 Classical Linear Codes

We shall restrict our consideration to only binary codes. Consider a binary code where k bits are encoded in a binary string of length n, i.e., among 2^n strings of length n, we have a 2^k-element subset of strings - or **codewords**. A k-bit message is one of these codewords.

For binary codes, this subset of 2^k codewords form a subspace C of the finite field $\mathbb{F}_2^n$, i.e., linear combinations of codewords are codewords, i.e., the set of codewords are closed under addition.

We note that

$$\dim(C) = k, \tag{1}$$

i.e., the subspace C is spanned by a basis of k vectors $v_1, v_2, \ldots, v_k$, each of length n. A general codeword in C can be expressed as

$$v(\alpha_1, \alpha_2, \alpha_2, \ldots, \alpha_k) = \sum_i \alpha_i v_i, \tag{2}$$

for $i \in \{1, 2, \ldots, k\}$ and $\alpha_i \in \{0, 1\}$, and addition is modulo 2. (Remember that we are working with the binary system here.) We should very clear about terminologies here. The **message** is $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k)$, while the codeword, or the n-length vector $v(\alpha_1, \alpha_2, \ldots, \alpha_k)$ encodes the k-bit message α. In the language of linear algebra, we can think of the message is a coordinatization of a codeword with respect to some basis. Or, the message is encoded as the codeword, by some encoding mechanism, or **generator**, which we discuss shortly.

2.3 Generator Matrix

As in linear algebra where we can generate an isomorphism by concatenating the basis vectors into an atrix, we can concatenate the k basis vectors $v_1, v_2, \ldots, v_k$ into a $k \times n$ matrix

$$G = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{bmatrix}, \tag{3}$$

called the **generator matrix** of the code. In matrix notation, a general codeword can now be expressed in terms of G:

$$v(\alpha) = \alpha G. \tag{4}$$

We say that the matrix G encodes the message α.

Notice how G acts right-to-left. This is due to the fact that G is a concatenation of row, rather than column, vectors. If we were to make G a matrix of n-length column vectors, our notation can get very messy. It is conventional to keep G this way.

2.4 Parity-Check Matrix

Another way to characterize the k-dimensional subspace of $\mathbb{F}_2^n$ is to specify the $n-k$ linear constraints. We define the **parity-check matrix** such that

$$Hv = 0 \tag{5}$$

for all those and only those vectors $v \in C$. H is called the parity-check matrix of code (subspace) C. The rows of H are $n-k$ linearly independent vectors that are *orthogonal* to all vectors in the code space, i.e., $\mathrm{Im}(G)$. Here, *orthogonality* is defined in the binary sense: two n-length binary strings are orthogonal if they "collide," i.e., take the value 1, at an evan number of locations.

Example 2.1. Show that $v_1 = 0110001$ and $v_2 = 0011101$ are orthogonal.

Solution 2.1. The inner product is assumed to be given by

$$\langle v_1, v_2 \rangle = \sum_{n=1}^{7} v_{1n} v_{2n} = 0 + 0 + 1 + 0 + 0 + 0 + 1 = 0, \tag{6}$$

since $1_2 + 1_2 = 0_2$ and $1_2 \cdot 1_2 = 1_2$ in $\mathbb{F}_2$. So, v_1 and v_2 are orthogonal codes.

Proposition 2.1. The rows of G are orthogonal to the rows of H, i.e.,

$$HG^\top = 0. \tag{7}$$

2.5 Errors in Classical Codes and The Syndrome of an Error

For a classical bit, the only kind of error is a **bit flip**. We can formally characterize an error occurring to an n-length code v by an n-length vector e where the 1's in e mark the locations where errors occur in v. When afflicted by the error e,

$$v \to v + e. \tag{8}$$

Example 2.2. Consider the code $v = 0011101$. There is an error on the 2^{nd} and 5^{th} bits in v. Express v in terms of the corrected versipn, v_c and the error e.

Solution 2.2. We simply perform 2 bit-flips at positions 2 and 5 on the code v and keep a record of the bit flips on e.

$$v = 0011101 = v_c + e = 0111001 + 0100100. \tag{9}$$

It is natural to ask the question of detecting an error, since we are almost never told whether a received code is error-free. Here's when the parity-check matrix H becomes handy. Recall that H sends a correct codeword in C to 0, so if we express $v = v_C + e$, then

$$Hv = H(v_C + e) = 0 + He = He. \tag{10}$$

We call He the **syndrome of the error** e. We also let $\mathcal{E}$ denote the set of errors $\{e_i\}$ that we wish to be able to correct.

Error correction is possible if all errors e_i have distinct syndromes, i.e., the map H on $\mathcal{E}$ is one-to-one. If this is the case, then given an He, we can find the corresponding error e such that given the error code $v = v_C + e$, we can perform an error-correction by performing a correcting bit flip:

$$(v_C + e) + e = v_C. \tag{11}$$

However, in the case of the map H not one-to-one, i.e., $He_1 = He_2$ but $e_1 \neq e_2$, it is not possible to accurately identify the error, i.e., an error e_1 can be misinterpreted as e_2. In which case, an attempt to recover the correct code will corrupt the code:

$$(v_C + e_1) + e_2 = v_C + (e_1 + e_2) \neq v_C, \tag{12}$$

even though $v_C + (e_1 + e_2) \in C$.

2.6 Distance of a code

The *distance* d of a code C is the minimum weight of any vector $v \in C$, where the *weight* is the number of 1's in the string v.

Theorem 2.1. A code with minimum distance $d = 2t + 1$ can correct t errors; the code assigns a distinct syndrome to each $e \in \mathcal{E}$, where each $\mathcal{E}$ contains all vectors of weight t or less.

Proof. Sketch: If $He_1 = He_2$, then

$$0 = He_1 + He_2 = H(e_1 + e_2), \tag{13}$$

and therefore $e_1 + e_2 \in C$. But if e_1 and e_2 are unequal and each has weight no larger than t, then the weight of $e_1 + e_2$ is greater than zero and no larger than $2t$. Since $d = 2t + 1$, there is no such vector in C. Therefore He_1 and He_2 cannot be equal. □

2.7 Dual code

We have seen that

$$HG^\top = 0. \tag{14}$$

Taking the transpose, we get

$$GH^\top = 0. \tag{15}$$

In this point of view, we can think of $H^\top$ as the *generator matrix* and G as the *parity-check matrix* of an $n-k$-dimensional code, denoted $C^\perp$ and called **the dual of** C. $C^\perp$ is the orthogonal complement of C in $\mathbb{F}_2$. We note that C and $C^\perp$ can intersect, since a vector can be self-orthogonal (if it has even weight).

A code contains its dual if all of its codewords have even weight and are mutually orthogonal. If $n = 2k$, it is possible tat $C = C^\perp$, in which case we say C is *self-dual*. We note this formally as

$$\sum_{v\in C}(-1)^{v\cdot v} = \begin{cases} 2^k, & u \in C^\perp \\ 0, & u \notin C^\perp. \end{cases} \tag{16}$$

The non-trivial content of the identity is the statement that the sum vanishes for $u \notin C^\perp$. This follows because

$$\sum_{v\in\{0,1\}^k}(-1)^{v\cdot w} = 0, \qquad w \neq 0, \tag{17}$$

where u, w are strings of length k. We can express $v \in G$ as $v = \alpha G$, where α is a k-vector. Then

$$\sum_{v\in C}(-1)^{v\cdot u} = \sum_{\alpha\in\{0,1\}^k}(-1)^{\alpha\cdot Gu} = 0, \tag{18}$$

for $Gu \neq 0$. Since G the generator matrix of C is the parity check matrix for $C^\top$, we conclude that the sum vanishes for $u \notin C^\perp$.

2.8 Finding Generator and Parity-Check Matrices

2.9 An example

3 Quantum Mechanical Preliminaries

4 Introduction to Quantum Computation and Information

4.1 Quantum bits

4.2 Quantum computation

4.3 Quantum algorithms

4.4 Experimental Quantum Information Processing

4.5 Quantum Information