A Short Introduction to Channel Coding

Supplemental Material for Graphical Models and Inference

Henry D. Pfister

September 15th, 2014

1 Introduction

1.1 What is channel coding and why do we use it?

Channel coding is the art of adding redundancy to a message in order to make it more robust against noise. It is used because noise and errors are essentially unavoidable in many systems (e.g., wireless communications and magnetic storage). Coding allows one to trade-off rate for reliability and usually provides large gains in overall system efficiency. In contrast, source coding (or compression) is used to remove the redundancy from sources (e.g., zip files and JPEG pictures). Channel coding carefully adds redundancy to a message so that it can be transmitted reliably over noisy channels.

Example 1.1 (Repeat Code)**.** *Consider the 1 bit message* $u \in \{0,1\}$ *mapped to a codeword of length* $2t+1$ *by repeating the message bit. This gives a binary code with two codewords:*

$$\mathcal{C} = \{\underbrace{00\ldots00}_{2t+1}, \underbrace{11\ldots11}_{2t+1}\}.$$

If fewer than t errors, then received sequence is closer to correct codeword. Therefore, a decoder which chooses the codeword closest to the received sequence will decode successfully. For a binary code, the code rate is defined to be

$$R = \frac{\#\ \text{information bits}}{\#\ \text{code bits}},$$

and this gives $\frac{1}{2t+1}$ *for the repeat code.*

Example 1.2 (Credit Card Numbers)**.** *Credit card numbers use a "checksum" to detect all single errors and most adjacent transpositions. Let* $\underline{x}$ *be a credit card*

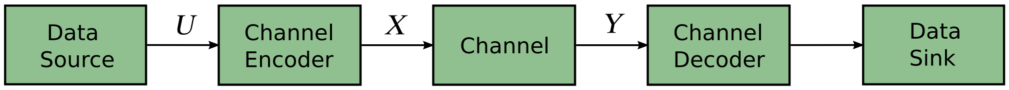


Figure 1: Block diagram of digital communication from a coding perspective.

number whose digits are given by $\underline{x} = (x_1, x_2, \ldots, x_{16})$*, then*

$$\left[\sum_{i=1}^{8} x_{2i} + \sum_{i=1}^{8} (2x_{2i-1})\,\#9\right] \bmod 10 = 0,$$

where $x\#9$ *is the sum of the digits in* x *(e.g.,* $11\#9 = 2$ *and* $18\#9 = 9$*). Consider the number 5420 1213 7904 9260. In this case, the first sum gives* $4+0+2+3+9+4+2+0 = 24$ *and the second sum gives:* $1+4+2+2+5+0+9+3 = 26$*. So, we have the overall checksum* $[24+26] \bmod 10 = 0$*. The code detects single errors because, for* $i = 1, \ldots, 8$*, changing* x_{2i} *to* x'_{2i} *changes the checksum by* $x'_{2i} - x_{2i}$ *and changing* x_{2i-1} *to* x'_{2i-1} *changes the checksum by* $2(x'_{2i} - x_{2i})\#9$*. Likewise, adjacent transpositions make the checksum non-zero unless the transposed values are 0 and 9. For example, swapping* x_1, x_2 *changes the check by*

$$[(x_1 - x_2) + (2x_2\#9 - 2x_1\#9)] \bmod 10.$$

Coding is used in many systems and devices including:

- CD / DVD players : Modulation code + Reed-Solomon (RS) code
- Digital Video Broadcasting (DVB): Convolutional Code + RS code
- Deep Space Communications: Advanced Turbo and LDPC codes
- Cell Phones: Convolutional codes for voice and Turbo/LDPC codes for data

1.2 Channels and Error Models

When designing and analyzing channel codes, one often uses a simple model of a communications channel known as a **discrete memoryless channel** (DMC). The channel input X is an element of input alphabet $\mathcal{X}$ and the channel output Y is an element of the output alphabet $\mathcal{Y}$. The main assumption is that each channel use is independent and governed by the probability law

$$W(y|x) \triangleq \Pr\left(Y = y | X = x\right).$$

[Add figure with transition diagrams]

The **binary symmetric channel** (BSC) with error probability p is defined by $\mathcal{X} = \mathcal{Y} = \{0, 1\}$ and

$$W(y|x) = \begin{cases} p & \text{if } x \neq y \\ 1-p & \text{if } x = y \end{cases}$$

The **binary erasure channel** (BEC) with erasure probability ϵ is defined by $\mathcal{X} = \{0, 1\}$, $\mathcal{Y} = \{0, 1, ?\}$ and

$$W(y|x) = \begin{cases} \epsilon & \text{if } y = ? \\ 1-\epsilon & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}$$

The **binary-input AWGN channel** (BIAWGN) with $\sigma^2 = \frac{N_0}{2}$ is defined by $\mathcal{X} = \{-1, 1\}$, $\mathcal{Y} = \mathbb{R}$ and

$$W(y|x) = \frac{1}{\sqrt{\pi N_0}} e^{-|y-x|^2/N_0},$$

where N_0 is the noise spectral density at the receiver.

The SNR of communication system is defined in terms of the **energy per information bit**, E_b, and the average **energy per transmitted symbol**, E_s. The conversion between these two quantities is based on keeping track of the units

$$E_s = \frac{\#\text{ information bits}}{\#\text{ transmitted symbols}} \frac{\text{Energy}}{\text{information bit}} = R' E_b.$$

The information rate R' (bits/channel use) is equal to the code rate R for binary-input channels. To make a fair comparisons, one must use the rate-normalized quantity E_b/N_0 (pronounced ebb-no). The normalization adjusts for the extra energy used to send parity symbols. The **coding gain** is the reduction in required E_b/N_0 to achieve a particular error rate. In other cases, it more convenient to use the quantity E_s/N_0 (pronounced ess-no).

For example, a bit-error rate (BER) of 10^{-5} is achieved on a BIAWGN channel by uncoded transmission with $E_b/N_0 = 9.6$ dB. Whereas, a rate-$\frac{1}{2}$ code with moderate decoding complexity (Viterbi decoding of a convolutional code) has a BER of 10^{-5} at $E_b/N_0 = 4.2$ dB. The coding gain in this case is $9.6 - 4.2 = 5.4$ dB

2 The Basics of Coding

2.1 Codes

Definition 2.1. *A length-n* **code** *over the alphabet $\mathcal{X}$ is simply a subset $\mathcal{C} \subseteq \mathcal{X}^n$ of all input sequences.*

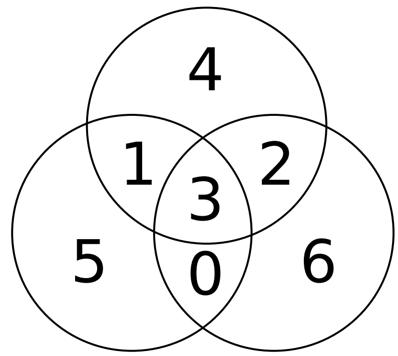


Figure 2: Venn diagram representation of the (7,4) binary Hamming code.

If a binary code has $M = |\mathcal{C}|$ codewords, then the code rate is $R = \frac{\log_2 M}{n}$. This means we can send k information bits when $M = 2^k$.

For example, the binary repeat code of length 5 is defined by $\mathcal{X} = \{0,1\}$ and

$$\mathcal{C} = \{00000, 11111\} \subset \{0,1\}^5 .$$

Likewise, the binary even parity code of length 3 is

$$\mathcal{C} = \{000, 110, 101, 011\} \subset \{0,1\}^3 .$$

Definition 2.2. *The **Hamming distance** $d(\underline{x}, \underline{y})$ is equal to the number of places where the vectors differ. It can be defined mathematically by*

$$d(\underline{x}, \underline{y}) = \sum_{i=1}^{n} (1 - \delta_{x_i, y_i}),$$

where $\delta_{a,b}$ is Kronecker delta function

$$\delta_{a,b} = \begin{cases} 0 & \text{if } a \neq b \\ 1 & \text{if } a = b \end{cases}.$$

The distance between codewords is typically measured with the Hamming distance. Using this metric, the set $\mathcal{X}^n$ forms a discrete metric space Another important code parameter is the minimum distance d_{min} between any two codewords is

$$d_{min} \triangleq \min_{\underline{x}, \underline{y} \in \mathcal{C}, \underline{x} \neq \underline{y}} d(\underline{x}, \underline{y}).$$

Example 2.3 (Hamming Code)**.** *The (7,4) binary Hamming Code has $n = 7$, $M = 16$, and $d_{min} = 3$. The code can be defined in terms of a Venn diagram showing three partially overlapping sets. Each of the seven subregions represent a* *code bit and the three circles represent even parity constraints. Encoding can be done by choosing $x_0, \ldots, x_3$ arbitrarily and then computing the last three parity bits. Any single error can be corrected by observing each bit error gives a unique pattern of parity violations. The codewords can be listed as follows:*

0000000	0100110	1000011	1100101
0001111	0101001	1001100	1101010
0010101	0110011	1010110	1110000
0011010	0111100	1011001	1111111

2.2 Decoding

Consider the decoding problem for binary codes with $\mathcal{X} = \mathcal{Y} = \{0, 1\}$ and $\mathcal{C} \subseteq \mathcal{X}^n$. The channel input is $\underline{x} \in \mathcal{C}$, the received sequence is $\underline{y}$, and the number of errors is $t = d(\underline{x}, \underline{y})$ It is not hard to verify that minimum distance decoding, which returns the codeword closest to the channel output, is optimal. Breaking ties arbitrarily, one can write

$$\hat{\underline{x}} = \arg\min_{\underline{w} \in \mathcal{C}} d(\underline{w}, \underline{y})$$

The following implies that the minimum distance decoder can always correct t errors if $d_{min} \geq 2t + 1$.

Proposition 2.4. *For any received sequence $\underline{y}$, there is at most one codeword $\underline{w}$ such that $d(\underline{y}, \underline{w}) \leq \frac{d_{min}-1}{2}$.*

Proof. Suppose there are codewords $\underline{w}, \underline{z}$ where $d(\underline{y}, \underline{w})$ and $d(\underline{y}, \underline{z})$ are $\leq \frac{d_{min}-1}{2}$. Then, the triangle inequality implies $d(\underline{w}, \underline{z}) \leq d(\underline{y}, \underline{w}) + d(\underline{y}, \underline{z}) \leq d_{min} - 1$ and contradicts the definition of d_{min}. Therefore, if $t \leq \frac{d_{min}-1}{2}$, then $\underline{x}$ is the unique codeword such that $d(\underline{x}, \underline{y}) \leq \frac{d_{min}-1}{2}$. □

The following allows simultaneous error correction of t errors and detection of d errors.

Proposition 2.5. *If $d_{min} \geq t + d + 1$ and $d \geq t$, then a single decoder can both correct t and detect d errors.*

Proof. Assume that each codeword is surrounded a inner ball of radius t and an outer ball of radius d. If the received vector is in an inner ball, decode to the codeword at the center. Otherwise, declare decoding failure.

From the previous result, we see that no two inner balls overlap and that the inner ball of one codeword does not overlap the outer ball of any codeword. If

+	0	1
0	0	1
1	1	0

*	0	1
0	0	0
1	0	1

Table 1: The addition and multiplication operations for the binary field.

the number of errors is at most t, then received vector will be in the inner ball of the transmitted codeword and will be decoded correctly. If the number of errors is between $t+1$ and d, then received vector will not be in the inner ball of any codeword and failure will be declared. $\square$

Proposition 2.6. *If $d_{min} \geq e+1$, then there is a decoder which corrects all patterns of e erasures.*

Proof. Make a list of all codewords and then erase any e positions. Each erasure reduces the minimum distance between any two codewords by at most one. After e steps, the new $d_{min} \geq e+1-e=1$. This implies that the codewords, defined by the remaining symbols, are all unique. $\square$

3 Binary Linear Codes

3.1 Basic Properties

This chapter focuses almost exclusively on binary linear codes, which are the simplest and most important class of error-correcting codes. The restriction to linear codes can be motivated by two things: simplicity and performance. We will see later that linear codes are much simpler to describe, encode, and analyze. Moreover, there are very few cases where non-linear codes are better than linear codes. So, there is essentially no performance penalty for this simplicity.

Linear codes, like linear algebra, make use of matrices and vectors of elements that can be added, subtracted, multiplied, and divided. A set of numbers which obey all the standard rules of arithmetic is an algebraic object known as a **field**. For example, the real and complex numbers are both fields.

There are also fields which have a finite number of elements. Let a, d be positive integers so that the division of a by d gives the equation $a = dq + r$, where q is quotient and $0 \leq r \leq d-1$ is the remainder. The modulo operation is defined to return the remainder from division and is denoted $r = a \bmod d$. It turns out that the binary alphabet, $\{0,1\}$, with standard arithmetic $(+,-,*,/)$ performed modulo 2 is also a field. The operations are shown explicitly in Table 1.

For linear algebra over a field, the scalar (i.e., field) operations are used to define vector and matrix operations. Vector addition is defined element-wise, so that $\left[\underline{x}+\underline{y}\right]_i = x_i + y_i$. An (n,k) **binary linear code** is $\mathcal{C} \subseteq \{0,1\}^n$ with $|\mathcal{C}| = 2^k$ where $\underline{x}, \underline{y} \in \mathcal{C}$ implies $\underline{x} + \underline{y} \in \mathcal{C}$ Since $\underline{x} + \underline{x} = \underline{0}$, this implies all zero vector $\underline{0} \in \mathcal{C}$. For example, the $n = 3$ "even parity" code is a (3,2) binary linear code with codewords $\mathcal{C} = \{000, 110, 101, 011\}$.

Definition 3.1. *The* ***Hamming weight*** $w(\underline{x})$ *is the number of non-zero elements in* $\underline{x}$ *or*

$$w(\underline{x}) = \sum_{i=1}^{n} (1 - \delta_{x_i,0}).$$

For binary vectors, this also implies that the Hamming distance is given by

$$d(\underline{x}, \underline{y}) = w(\underline{x} - \underline{y}).$$

Linear codes also have a simplified distance structure. Instead of considering the minimum distance between all codewords, it suffices to focus only on the all-zero codeword.

Proposition 3.2. *The minimum distance of a linear code is given by*

$$d_{min} = \min_{\underline{x} \in \mathcal{C}, \underline{x} \neq \underline{0}} w(\underline{x} - \underline{y}).$$

Proof. The linear property of the code allows one to translate computations involving the distance between two codewords to expressions involving the Hamming weight of one codeword. This gives

$$\begin{aligned} d_{min} &\triangleq \min_{\underline{x}, \underline{y} \in \mathcal{C}, \underline{x} \neq \underline{y}} d(\underline{x}, \underline{y}) \\ &= \min_{\underline{x}, \underline{y} \in \mathcal{C}, \underline{x} \neq \underline{y}} w(\underline{x} - \underline{y}) \\ &= \min_{\underline{x} \in \mathcal{C}, \underline{x} \neq \underline{0}} w(\underline{x} - \underline{y}), \end{aligned}$$

where the last step follows from the fact that

$$\left\{\underline{x} - \underline{y} \,|\, \underline{x}, \underline{y} \in \mathcal{C}, \underline{x} \neq \underline{y}\right\} = \{\underline{x} \in \mathcal{C} \,|\, \underline{x} \neq \underline{0}\}.$$

□

3.2 Generator and Parity-Check Matrices

Linear codes can be represented compactly using matrices. The generator defines the code by allowing one to list all the codewords.

Definition 3.3. *The* ***generator matrix*** $\underline{G}$ *of an* (n,k) *binary linear code is a* $k \times n$ *binary matrix such that all codewords,* $\underline{x} \in \mathcal{C}$*, can be written as* $\underline{u} \cdot \underline{G} = \underline{x}$ *for some message vector* $\underline{u} \in \{0,1\}^k$*. Therefore, the code is the row space of* $\underline{G}$*).*

Example 3.4. *The generator matrix*

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

defines the $(5,2)$ *code*

$$\mathcal{C} = \{00000, 10110, 01011, 11101\}.$$

Encoding $\underline{u} = [11]$ *gives* $\underline{u} \cdot \underline{G} = [0\ 1\ 0\ 1\ 1]$

If $\underline{G}$ has full rank k (over the binary field), then the code has 2^k distinct codewords. Otherwise, some non-zero messages encode to the all-zero codeword and there are at most 2^{k-1} codewords.

Definition 3.5. *The generator matrix is in* ***systematic form*** *if* $\underline{G} = [\underline{P}\ \ \underline{I}_k]$*, where* $\underline{I}_k$ *is the* $k \times k$ *identity matrix. The matrix* $\underline{P}$ *is called the* ***parity generator*** *of the code because* $\underline{u} \cdot \underline{P}$ *computes the parity bits for* $\underline{u}$*. For a generator matrix in systematic form, the message vector appears in codeword*

$$\underline{u} \cdot \underline{G} = \underline{u} \cdot [\underline{P}\ \ \underline{I}_k] = [\underline{u} \cdot \underline{P}\ \ \underline{u}].$$

The parity-check (PC) matrix defines the code by listing the parity-check equations that each codeword must satisfy.

Definition 3.6. *The* ***parity-check matrix*** $\underline{H}$ *of an* (n,k) *binary linear code is an* $(n-k) \times n$ *binary matrix such that* $\underline{x} \cdot \underline{H}^T = \underline{0}$ *for all* $\underline{x} \in \mathcal{C}$*. Therefore, the code is the null space of* $\underline{H}$*.*

While the generator matrix defines the code and an encoder, the parity-check matrix defines only the code; there is no implied encoder. There is also a relationship between and generator and parity-check matrix for the same code. Recall that, for all codewords $\underline{x}$, there is a message $\underline{u}$ such that $\underline{x} = \underline{u} \cdot \underline{G}$. This means that $\underline{G} \cdot \underline{H}^T = \underline{0}$.

Example 3.7. *For the* $(5, 2)$ *code we saw previously, one possible parity-check matrix is*

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

Notice that it satisfies $\underline{G} \cdot \underline{H}^T = 0$ *for the previous* $\underline{G}$*.*

In general, we assume that $\underline{H}$ has full rank. Otherwise, there are redundant constraints and some row can eliminated without changing the code.

A parity-check matrix is in **systematic form** if $H = \begin{bmatrix} \underline{I}_{n-k} & -\underline{P}^T \end{bmatrix}$. When both the generator and parity-check matrices are in systematic form, we can write

$$\begin{aligned} \underline{G} \cdot \underline{H}^T &= [\underline{P} \ \ \underline{I}_k] \cdot \begin{bmatrix} \underline{I}_{n-k} & -\underline{P}^T \end{bmatrix}^T \\ &= [\underline{P} \ \ \underline{I}_k] \cdot \begin{bmatrix} \underline{I}_{n-k} \\ -\underline{P} \end{bmatrix} \\ &= \underline{P} - \underline{P} \\ &= \underline{0}. \end{aligned}$$

Example 3.8. *A* ***single parity-check code*** *is an* $(n, n-1)$ *binary linear code with parity-check matrix*

$$H = [\underbrace{1, 1, \ldots, 1}_{n \text{ times}}].$$

For all codewords $\underline{x}$*, the parity-check constraint* $\underline{x} \cdot \underline{H}^T = \underline{0}$ *implies that* $\sum_{i=1}^{n} x_i \bmod 2 = 0$ *(i.e., the codeword has an even number of ones). The minimum distance is* $d_{min} = 2$ *and the generator matrix is given by*

$$G = \begin{bmatrix} 1 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 1 & \cdots & 0 \\ \vdots & 0 & \ldots & \ddots & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Next, we consider the minimum distance of a code in terms of its generator and parity check matrices. In general, it is very difficult to compute the minimum distance without enumerating all codewords. But, one can get upper and lower bounds on the minimum distance much more easily. In this way, the minimum distance can be found approximately.

Since the minimum distance is equal to the minimum of the Hamming weight overall codewords, it is clearly upper bounded by the Hamming weight of any particular non-zero codeword. This gives, for any non-zero codewords $\underline{x}$,

$$d_{min} \leq w(\underline{x}).$$

Likewise, the parity-check matrix can be used to lower bound the minimum distance.

Proposition 3.9. *The minimum distance of a code with parity-check matrix*

$$\underline{H} = [\underline{h}_1, \underline{h}_2, \ldots, \underline{h}_n]$$

is equal to the minimum number of columns that sum to zero or

$$d_{min} = \min \left\{ w(\underline{x}) | \underline{x} \cdot \underline{H}^T = \underline{0}, \underline{x} \in \{0,1\}^n \setminus \underline{0} \right\}.$$

Proof. Notice that

$$\underline{x} \cdot \underline{H}^T = \sum_{i=1}^{n} x_i \underline{h}_i = \sum_{i:x_i=1} \underline{h}_i,$$

where $\underline{h}_i$is the ith column of $\underline{H}$. Therefore, the statement that $\underline{x} \cdot \underline{H}^T = \underline{0}$ (i.e., $\underline{x}$ is a codeword) is equivalent to the statement that the sum of $w(\underline{x})$ columns is $\underline{0}$. Taking the minimum over all non-zero codewords gives the minimum distance. □

One can also bound the minimum distance in terms of error-correction ability. Recall that a code corrects all error patterns of weight t if and only if $d_{min} \geq 2t+1$. This gives a simple lower bound on the minimum distance.

Example 3.10. *Correcting all single errors requires $\frac{d_{min}-1}{2} = 1$ or $d_{min} = 3$. Let us try to find the longest code, for a fixed number of parity bits m, that corrects all single errors. In this case, $\underline{H}$ is matrix with m rows and we can add columns, one at a time, until it is not possible to add a column without losing the ability to correct a single error. How many columns can we choose? The maximum value is $n = 2^m - 1$ and the resulting optimal code is called the binary* ***Hamming code*** *of length n.*

3.3 Decoding

Assume a codeword $\underline{x} \in \mathcal{C}$ is transmitted over a channel and $\underline{r} = \underline{x} + \underline{e}$ is received, where $\underline{e}$ is the **error pattern** with $w(\underline{e})$ errors. For any received sequence $\underline{r}$, a **decoder** either returns a codeword $\hat{\underline{x}} = D(\underline{r})$ or declares failure. The decoded message $\hat{\underline{u}}$ associated with $\hat{\underline{x}}$ is the unique message satisfying $\hat{\underline{x}} = \underline{\hat{u}G}$. A decoder makes

- a **block error** (or word error) if $\underline{\hat{x}} \neq \underline{x}$ and P_B is used to denote the probability of block error,
- b **code bit errors** if $w(\underline{\hat{x}} - \underline{x}) = b$ and P_b is used to denote the probability that a randomly chosen bit in $\underline{\hat{x}}$ is in error,
- and b **message bit errors** if $w(\underline{\hat{u}} - \underline{u}) = b$ and P_b is used to denote the probability that a randomly chosen bit in $\underline{\hat{u}}$ is in error.

From this, we see that the probability of bit error P_b can have multiple meanings. The correct meaning can usually be inferred from the context. Decoding can also be simplified for linear codes.

Definition 3.11. *Let $\underline{s} = \underline{r} \cdot \underline{H}^T$ be the **syndrome** of the received vector $\underline{r}$.*

It turns out that $\underline{s}$ depends only on the error pattern. Since $\underline{x} \in \mathcal{C}$, we have

$$\underline{s} = \underline{r} \cdot \underline{H}^T = (\underline{x} + \underline{e}) \cdot \underline{H}^T = \underline{x} \cdot \underline{H}^T + \underline{e} \cdot \underline{H}^T = \underline{e} \cdot \underline{H}^T.$$

A **syndrome decoder** $\underline{\hat{e}} = D_s(\underline{s})$ maps the syndrome $\underline{s}$ to an estimated error pattern $\underline{\hat{e}}$. Let us define the equivalence relation $\sim$ by $\underline{x} \sim \underline{y}$ iff $\underline{x} \cdot \underline{H}^T = \underline{y} \cdot \underline{H}^T$. This means that two binary vectors are equivalent if they have the same syndrome. A syndrome decoder can be designed correct exactly one error pattern in each equivalence class. The best choice for correction is the most-likely error pattern in that equivalence class. For most channels, these vectors are chosen to be the minimum weight vector in the equivalence class.

The **standard array** is way of listing all vectors of length-n that exposes the connection between syndromes, codewords, and error correction. In general, it is a $2^{n-k} \times 2^k$ table that contains each length-n binary vector exactly once. The main idea behind this table is that, when one chooses to correct a particular error pattern, the decoder is automatically defined for all received sequences equal to that error pattern plus a codeword. Of course, this limits one's ability choose correctable error patterns.

Each row is indexed by a syndrome $\underline{s}$ and contains all binary vectors $\underline{x}$ that satisfy the equation $\underline{s} = \underline{x} \cdot \underline{H}^T$. The first row is reserved for $\underline{s} = \underline{0}$ and contains the all-zero codeword $\underline{c}_1$ in the first column followed the remaining codewords $\underline{c}_2, \ldots, \underline{c}_{2^k}$ in any order. The first column contains the correctable error patterns $\underline{e}_1, \ldots, \underline{e}_{2^{n-k}}$ where $\underline{e}_1$ is the all-zero sequence. The row-i column-j entry always contains $\underline{e}_i + \underline{c}_j$, and is therefore defined by the first row and first column of the table. The column associated with a particular codeword can be seen as all the received vectors that will be decoded to that codeword.

Using the parity-check matrix for our $(5, 2)$ our example code, the process is as follows

1. Start with 8 by 5 table and list the zero syndrome and all codewords on first row
2. Pick a minimum weight vector of length n that is not already in table and compute its syndrome
3. Add a new row by writing syndrome followed by the minimum weight vector plus each codeword
4. Repeat from step 2 until table is complete.

The resulting standard array is

syn\cw	00000	10110	01011	11101
100	10000	00110		01101
010	01000		00011	
001	00100	10010		11001
101	00010		01001	
111	00001	10111		11100
110	11000	01110	10011	
011	01100	11010		10001

To see if you understand this example, try filling in the missing entries. One can prove that this process always enumerates all 2^n binary vectors, so you can test your answers by checking if all binary vectors appear in the table exactly once.

Example 3.12. *To see syndrome decoding in action, let* $\underline{x} = 10110$ *and* $\underline{e} = 11000$*. Then,* $\underline{r} = 01110$ *and* $\underline{s} = 110$*. Looking in the syndrome table, we find that* $\underline{\hat{e}} = 11000$ *and* $\underline{\hat{x}} = 10110$*.*

3.4 Manipulating Linear Codes

It follows from linear algebra that any $\underline{G}, \underline{H}$ can be put in systematic form using elementary row operations and (possibly) a column permutation. For the parity-check matrix H, the basic idea is to use elementary row operations to form an identity matrix in the first few columns (i.e., put it in reduced row-echelon form).

For the generator matrix G, elementary row operations are used to form an identity matrix in the last few columns. Sometimes an identity cannot be formed in the desired columns and a column permutation is required to complete the process. For this reason, two codes are called **equivalent** if and only if they differ only in the order of the code symbols.

Definition 3.13. *For a matrix, an* ***elementary row operation*** *is any one of the following operations:*

1. *interchanging any two rows,*
2. *scaling any row by a non-zero constant,*
3. *and adding a multiple of one row to another row.*

Example 3.14. *In this example, we consider a parity-check matrix*

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

that requires a column permutation for systematic form. Let us put $\underline{H}$ *is reduced row-echelon form and then find a column permutation to achieve an identity in the first few rows. The first step gives*

$$\underline{H} \to \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \to \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \to \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix},$$

and a column permutation gives

$$\underline{\tilde{H}} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

Now, we can compute parity generator and generator associated with $\underline{\tilde{H}}$ *to get*

$$\underline{\tilde{P}} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \quad \underline{\tilde{G}} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Finally, reversing the column permutation gives, for the original code, a generator

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

that contains an identity in a subset of its columns.

4 Performance Analysis

Consider an (n, k) binary code of rate $R = k/n$ with $d_{min} \geq 2t^* + 1$ and a decoder that corrects all error patterns of weight at most t^*. It is relatively easy to upper bound the probability of block error when this code and decoder are used on a BSC, because a block error cannot occur unless there are more than t^* errors. This gives

$$P_B \leq 1 - \sum_{i=0}^{t^*} \binom{n}{i} p^i (1-p)^{n-i} = \sum_{i=t^*+1}^{n} \binom{n}{i} p^i (1-p)^{n-i}.$$

If p is small enough, then we can approximate this by the first term in the sum

$$P_B \approx \binom{n}{t^*+1} p^{t^*+1}.$$

If the code bits are transmitted over an AWGN channel using BPSK followed by a hard-decision detector, then we also have

$$p = Q\left(\sqrt{\frac{2E_s}{N_0}}\right) = Q\left(\sqrt{\frac{2RE_b}{N_0}}\right).$$

The latter expression allows us to make relatively fair comparisons between coding systems with different rates.

Definition 4.1. *The* ***coding gain*** *achieved by a channel code is the reduction in E_b/N_0 required achieve a particular error rate. For example, if the uncoded system achieves an error rate of 10^{-5} at $E_b/N_0 = 9.6$ dB and the coded system achieves an error rate of 10^{-5} at $E_b/N_0 = 6.6 dB$, then the coding gain is $9.6 - 6.6 = 3$ dB at the error rate 10^{-5}. The coding gain often approaches a limit, for low error rates, known as the* ***asymptotic coding gain****.*

Now, we compute the asymptotic coding gain, in terms of t^* and R, of hard-decision decoding of a block code. For the uncoded system, let γ_u be the E_b/N_0 so that the probability of block error is given by

$$P_B^{(u)}(\gamma_u) = 1 - \left(1 - Q\left(\sqrt{2\gamma_u}\right)\right)^k.$$

For the coded system, let γ_c be the E_b/N_0 so that the probability of block error is

$$P_B^{(c)}(\gamma_c) \approx \binom{n}{t^*+1} Q\left(\sqrt{2\gamma_c}\right)^{t^*+1}.$$

The exponential decay rate of these error probabilities with E_b/N_0 is given by

$$\lim_{\gamma\to\infty} \frac{1}{\gamma} \ln\left(P_B(\gamma)\right).$$

For large x, the Q-function is very well approximated by

$$Q(x) \approx \frac{1}{\sqrt{2\pi x^2}} e^{-x^2/2}.$$

This implies that the exponential decay rate of $P_B^{(u)}$ is γ_u and the exponential decay rate of $P_B^{(c)}$ is $(t^*+1)R\gamma_c$. Matching these exponential decay rates gives the equation

$$\frac{\gamma_u}{\gamma_c} = (t^*+1)R,$$

and converting to dB shows that the asymptotic coding gain is

$$10\log_{10}\left((t^*+1)R\right).$$

Example 4.2. *Consider the* $(15,11)$ *binary Hamming code with* $d_{min} = 3$ *and* $t^* = 1$*. This code achieves an asymptotic coding gain of*

$$10\log_{10}\left(2\frac{11}{15}\right) \approx 1.66 \text{ dB}.$$

For syndrome decoding, one can compute exactly the probability of block error (including decoding failure) by observing that the decoder only returns the correct codeword if the error vector is a coset leader. In this case, we can let A_h be the number of coset leaders with Hamming weight h and write

$$P_B = 1 - \sum_{h=0}^{n} A_h p^h (1-p)^{n-h}.$$

5 General Memoryless Channels

5.1 Optimal Decoding Rules

Let $\mathcal{X}$ be a finite alphabet and $\mathcal{C} \subset \mathcal{X}^n$ be a length-n code. Consider a discrete memoryless channel (DMC) defined by

$$W(y|x) \triangleq \Pr(Y = y \mid X = x)$$

and assume the codeword $x_1^n \in \mathcal{C}$ is transmitted with probability $p\left(x_1^n\right)$. The memoryless property implies that

$$\Pr\left(Y_1^n = y_1^n | X_1^n = x_1^n\right) \triangleq W\left(y_1^n | x_1^n\right) = \prod_{i=1}^{n} W\left(y_i|x_i\right)$$

and the **maximum likelihood** (ML) decoding rule is defined by

$$\hat{x}^{ML}(y_1^n) = \arg\max_{x_1^n \in \mathcal{C}} W\left(y_1^n | x_1^n\right),$$

with ties broken arbitrarily. Also, a binary channel is called **symmetric** if there is a permutation $\pi : \mathcal{Y} \to \mathcal{Y}$ satisfying $\pi(\pi(x)) = x$ such that $W(y|0) = W(\pi(y)|1)$.

The decoding error rate is minimized by MAP (or maximum a posteriori) decoding. For example, **block-MAP decoding** returns the codeword $x_1^n \in \mathcal{C}$ that maximizes

$$\Pr\left(X_1^n = x_1^n | Y_1^n = y_1^n\right) = \frac{p\left(x_1^n\right) W\left(y_1^n | x_1^n\right)}{\sum_{\tilde{x}_1^n \in \mathcal{C}} p\left(\tilde{x}_1^n\right) W\left(y_1^n | \tilde{x}_1^n\right)}.$$

Since the denominator only depends on y_1^n, we find that

$$\hat{x}^{MAP}\left(y_1^n\right) \triangleq \arg\max_{x_1^n \in \mathcal{C}} p\left(x_1^n\right) W\left(y_1^n | x_1^n\right),$$

with ties broken arbitrarily. Similarly, **bit-MAP decoding** of bit i maximizes $\Pr(X_i = x | Y_1^n = y_1^n)$ and

$$\begin{aligned}\hat{x}_i^{MAP}\left(y_1^n\right) &\triangleq \arg\max_{x \in \mathcal{X}} \Pr(X_i = x | Y_1^n = y_1^n) \\ &= \arg\max_{x \in \mathcal{X}} \sum_{x_1^n \in \mathcal{C}, x_i = x} \Pr(X_1^n = x_1^n | Y_1^n = y_1^n),\end{aligned}$$

where the sum marginalizes out all variables except x_i.

5.2 The Bhattacharyya Union Bound

The block error rate of block-MAP decoding can be written as

$$P_B = \sum_{x_1^n \in \mathcal{C}} p(x_1^n) \Pr\left(\left\{\cup_{z_1^n \in \mathcal{C} \setminus x_1^n} \left\{\hat{x}^{MAP}(Y_1^n) = z_1^n\right\}\right\} \,\middle|\, x_1^n \text{ sent}\right).$$

Upper bounding the probability of the union by the sum of the probabilities gives

$$P_B \leq \sum_{x_1^n \in \mathcal{C}} p(x_1^n) \sum_{z_1^n \in \mathcal{C} \setminus x_1^n} \Pr\left(\hat{x}^{MAP}(Y_1^n) = z_1^n \mid x_1^n \text{ sent}\right). \tag{1}$$

Let the **pairwise error probability** (PEP) of block-ML decoding be

$$P\left(x_1^n \to z_1^n\right) \triangleq \sum_{y_1^n \in \mathcal{Y}^n} W\left(y_1^n | x_1^n\right) I\left(W\left(y_1^n | x_1^n\right) \leq W\left(y_1^n | z_1^n\right)\right),$$

where the I function is 1 if its argument is true and 0 otherwise. If all codewords are transmitted with equal probability (i.e., $p(x_1^n) = 1/\left|\mathcal{C}\right|$), then MAP decoding is the same as ML decoding and

$$\Pr\left(\hat{x}^{MAP}(Y_1^n) = z_1^n \mid x_1^n \text{ sent}\right) \leq P\left(x_1^n \to z_1^n\right).$$

Since the PEP counts all ties as errors, there is a possibility of strict inequality in the previous expression. Regardless, (1) implies the standard **union bound**

$$P_B \leq \frac{1}{|\mathcal{C}|} \sum_{x_1^n \in \mathcal{C}} \sum_{z_1^n \in \mathcal{C} \setminus x_1^n} P(x_1^n \to z_1^n).$$

For any $s \in [0,1]$, one can write the bound

$$I\big(W\left(y_1^n | x_1^n\right) \leq W\left(y_1^n | z_1^n\right)\big) \leq \left(\frac{W\left(y_1^n | z_1^n\right)}{W\left(y_1^n | x_1^n\right)}\right)^s,$$

because the LHS $=0$ if the RHS <1 and the LHS $=1$ if the RHS >1. For symmetric binary channels, $s = 1/2$ gives the best bound and

$$\begin{aligned} P\left(x_1^n \to z_1^n\right) &\leq \sum_{y_1^n \in \mathcal{Y}^n} W\left(y_1^n | x_1^n\right) \left(\frac{W\left(y_1^n | z_1^n\right)}{W\left(y_1^n | x_1^n\right)}\right)^{1/2} \\ &= \prod_{i=1}^{n} \sum_{y_i \in \mathcal{Y}} \sqrt{W\left(y_i | x_i\right) W\left(y_i | z_i\right)}. \end{aligned}$$

Now, observe that

$$\sum_{y_i \in \mathcal{Y}} \sqrt{W\left(y_i | x_i\right) W\left(y_i | z_i\right)} = \begin{cases} \sum_{y \in \mathcal{Y}} \sqrt{W(y|x_i)\, W(y|x_i)} = 1 & \text{if } x_i = z_i \\ \sum_{y \in \mathcal{Y}} \sqrt{W(y|0)\, W(y|1)} \triangleq \gamma & \text{if } x_i \neq z_i. \end{cases}$$

Since $x_i \neq z_i$ gives a factor of γ and $x_i = z_i$ gives a factor of 1, we find that

$$P\left(x_1^n \to z_1^n\right) \leq \prod_{i: x_i \neq z_i} \overbrace{\sum_{y \in \mathcal{Y}} \sqrt{W(y|0)\, W(y|1)}}^{\gamma} = \gamma^{d_H\left(x_1^n, z_1^n\right)},$$

where γ is called the **Bhattacharyya constant** of the channel.

For the BSC(p) channel, a simple computation shows that $\gamma_{BSC} = 2\sqrt{p(1-p)}$. For the BEC($\epsilon$) channel, a simple computation shows that $\gamma_{BEC} = \epsilon$. For BPSK in AWGN, a simple computation shows that $\gamma_{AWGN} = e^{-E_s/N_0}$ for SNR E_s/N_0. We also note that γ is the best possible constant for a bound of the form γ^{d_H} and thus, we say that this bound is exponentially tight. Inserting the Bhattacharyya bound on pairwise error probability into the union bound gives the following upper bound on block error probability

$$P_B \leq \frac{1}{|\mathcal{C}|} \sum_{x_1^n \in \mathcal{C}} \sum_{z_1^n \in \mathcal{C} \setminus x_1^n} \gamma^{d_H\left(x_1^n, z_1^n\right)}.$$

5.3 Simplifications Due to Symmetry

Assume the pairwise error probability satisfies, for some f,

$$P(x_1^n \to z_1^n) \leq f\left(d_H\left(x_1^n, z_1^n\right)\right)$$

If the code is linear, then P_B is upper bounded by

$$\begin{aligned} P_B &\leq \sum_{x_1^n \in \mathcal{C}} \frac{1}{|\mathcal{C}|} \sum_{z_1^n \in \mathcal{C},\, z_1^n \neq x_1^n} f\left(d_H\left(x_1^n, z_1^n\right)\right) \\ &= \sum_{x_1^n \in \mathcal{C}} \frac{1}{|\mathcal{C}|} \sum_{z_1^n \in \mathcal{C},\, z_1^n \neq \mathbf{0}} f\left(d_H\left(\mathbf{0}, z_1^n\right)\right) \\ &= \sum_{z_1^n \in \mathcal{C},\, z_1^n \neq \mathbf{0}} f\left(w_H\left(z_1^n\right)\right) = -f(0) + \sum_{h=0}^{n} A_h f(h), \end{aligned}$$

where $A_h = |\{x_1^n \in \mathcal{C} \mid w_H(x_1^n) = h\}|$ is the weight enumerator of $\mathcal{C}$.

The weight enumerator function is given by

$$A(z) \triangleq \sum_{h=0}^{n} A_h z^h.$$

Combining this with the Bhattacharyya bound, we find

$$P_B \leq -1 + \sum_{h=0}^{n} A_h \gamma^h = -1 + A(\gamma).$$

Since the (7,4) binary Hamming code has $A(z) = 1 + 7z^3 + 7z^4 + z^7$, the implied bound for BPSK in AWGN is

$$P_B \leq 7e^{-3E_s/N_0} + 7e^{-4E_s/N_0} + e^{-7E_s/N_0}.$$

6 Random Codes

6.1 The Random Generator Matrix Ensemble

Let G be a random $k \times n$ matrix with i.i.d. equiprobable binary entries and $\mathcal{C}$ is the set of vectors formed by encoding all 2^k message vectors. Then, the number of codewords of weight h, A_h, is a random variable and its average is denoted $\overline{A}_h = \mathbb{E}\left[A_h\right]$. To compute $\overline{A}_h$, we let $u(i)$ be the k-bit binary expansion of i and write

$$\begin{aligned}\overline{A}_h &= \mathbb{E}\left[\sum_{i=0}^{2^k-1} I\left(w_H\left(u(i)G\right)=h\right)\right] \\ &= \sum_{i=0}^{2^k-1} \mathbb{E}\left[I\left(w_H\left(u(i)G\right)=h\right)\right] \\ &= \sum_{i=0}^{2^k-1} \Pr\left(w_H\left(u(i)G\right)=h\right) \\ &= \begin{cases} 1+\frac{2^k-1}{2^n} & \text{if } h=0 \\ \frac{(2^k-1)\binom{n}{h}}{2^n} & \text{otherwise.}\end{cases}\end{aligned}$$

It is important to note that, due to the random selection, the matrix G may not be full rank. This deficiency can be seen in the above formula as an increase (beyond 1) in the number of weight 0 codewords. In particular, if $\text{rank}(G) = k - i$, then there are 2^i vectors $u \in \{0,1\}^n$ such that $uG = 0$.

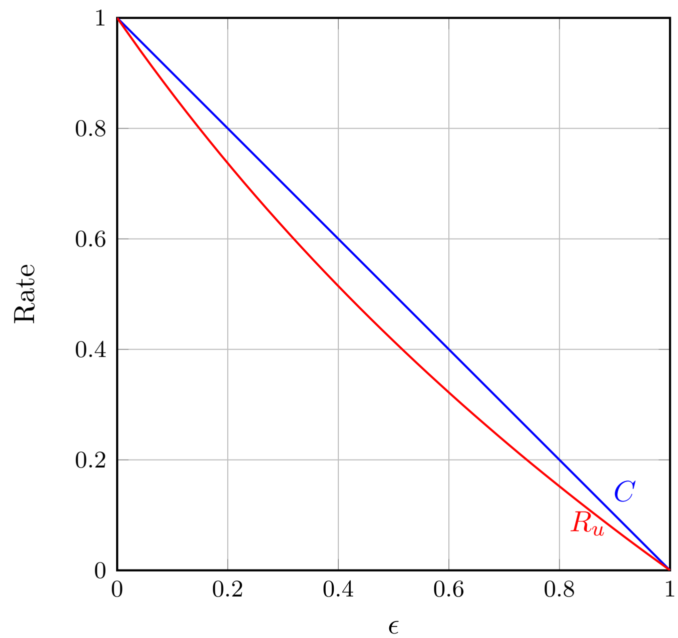
Using this to compute the Bhattacharyya union bound gives

$$\begin{aligned}P_B &\le -1 + \sum_{h=0}^{n} \overline{A}_h \gamma^h \le \frac{2^k-1}{2^n}\sum_{h=0}^{n}\binom{n}{h}\gamma^h \\ &= \frac{2^k-1}{2^n}(1+\gamma)^n \le \frac{2^{Rn}}{2^n}(1+\gamma)^n \\ &= \left(2^{R-1}(1+\gamma)\right)^n.\end{aligned}$$

Thus, the RHS vanishes as $n \to \infty$ iff $R < R_u = 1 - \log_2(1+\gamma)$. For the BEC($\epsilon$), we find that

$$\begin{aligned}R_u &= 1 - \log_2(1+\gamma_{BEC}) \\ &= 1 - \tfrac{1}{\ln 2}\ln(1+\epsilon).\end{aligned}$$

It is known that this ensemble achieves the capacity, $C = 1 - \epsilon$. But, as shown below, the Bhattacharyya union bound is too weak to prove this.



6.2 Joint Typicality Decoding

The Joint Typicality (JT) Decoder is defined as follows:

1. If there is exactly one $x_1^n \in \mathcal{C}$ jointly typical with y_1^n, then return it
2. Otherwise, declare failure.

Clearly this depends on how we define jointly typical. In general, it means that the output has roughly the right likelihood given the input. For the BEC, the received vector must have roughly ϵn erasures and the two sequences must match on all non-erased positions

Consider $x_1^n, z_1^n \in \mathcal{C}$ with $d_H(x_1^n, z_1^n) = h$ and assume Y_1^n is x_1^n altered by random pattern of exactly ϵn erasures. How many erasure patterns are there? How many will make z_1^n jointly typical with Y_1^n?

- the total number of ways to choose the erasures is $\binom{n}{\epsilon n}$
- the number of ways to choose the erasures so that z_1^n jointly typical is $\binom{n-h}{\epsilon n-h}$

Hence, the pairwise error probability for the JT decoder is

$$\begin{aligned}
\Pr\left(Y_1^N \text{ JT } z_1^n\right) &= \tbinom{n-h}{\epsilon n-h} / \tbinom{n}{\epsilon n} \\
&= \tfrac{(n-h)!}{(\epsilon n-h)!(n-\epsilon n)!} \cdot \tfrac{(\epsilon n)!(n-\epsilon n)!}{n!} \\
&= \tfrac{(n-h)!}{(\epsilon n-h)!} \cdot \tfrac{(\epsilon n)!}{n!} \cdot \tfrac{h!}{h!} \\
&= \tbinom{\epsilon n}{h} / \tbinom{n}{h}.
\end{aligned}$$

Plugging this pairwise error probability into the union bound gives

$$\begin{aligned}
P_B &\le -1 + \textstyle\sum_{h=0}^{n} \overline{A}_h \left[\tbinom{\epsilon n}{h} / \tbinom{n}{h}\right] \\
&= \textstyle\sum_{h=0}^{n} \left[2^{-n}(2^k - 1)\tbinom{n}{h}\right] \left[\tbinom{\epsilon n}{h} / \tbinom{n}{h}\right] \\
&\le 2^{n(R-1)} \textstyle\sum_{h=0}^{n} \tbinom{\epsilon n}{h} \\
&= 2^{n(R-1+\epsilon)}.
\end{aligned}$$

Since the RHS tends to 0 if $R < 1 - \epsilon$, we see that this bound is strong enough to prove the ensemble achieves capacity. It is instrictive to think about why this bound is better than the Bhattacharyya union bound.

Also, if the number of erasures E is random and the pattern is conditionally uniform given E, then the upper bound requires an additional additive term of the form $\Pr(E > \epsilon n)$ because the original upper bound holds when $E \le \epsilon n$ and P_B is upper bounded by 1 otherwise.