An Application of Vector Space Theory in Data Transmission

LieJune Shiau
University of Houston-Clear Lake
2700 Bay Area Boulevard
Houston, Texas 77058 USA
shiau@cl.uh.edu

Abstract
This work illustrates an application of vector spaces to data transmission theory. We show how Hamming code error detection and error correction are done through the tool of various theories in vector space. It is hoped that this article will explain the importance of abstract mathematics, such as vector space and basis, in the application of data transmission, which enlightens mathematics and computer science majors.

Keywords: vector space, data transmission, code words, code space

1. Introduction

The purpose of this article is to introduce the reader to an interesting real world application of vector spaces. We wish to exhibit one of the many strong links between abstract and applied mathematics. The ubiquity of the Internet makes data transmission a particularly relevant "real world" problem. Computer data is transmitted in a binary mode and hence is represented in strings of 0's and 1's called bits. When transmitting from a home computer via a modem, data is susceptible to "noise" corruption by an electromotive force, or to loss of signal due to attenuation. Corruption of the binary signal simply means some 0's are changed into 1's and vice versa. To counteract such corruption Engineers utilize data encoders and decoders. The diagram in Figure 1 shows a typical digital data communication system.

The most basic method of error detection utilizes a single-parity-check-bit scheme. An encoder simply attaches an extra binary bit, called a parity check bit, to the end of the message. The parity check bit can be computed the sum of all message bits (*even parity check*). When the decoder receives a string, it computes the binary sum of all bits, including the parity check bit. If there are an odd number of errors, the sum will be 1; if there are no errors, or an even number of errors, the sum will be 0. The weakness of this method is that only an odd number of errors can be detected, and the location of the error remains unknown. This means we cannot fix the errors, and instead the decoder must request a re-send of the data.

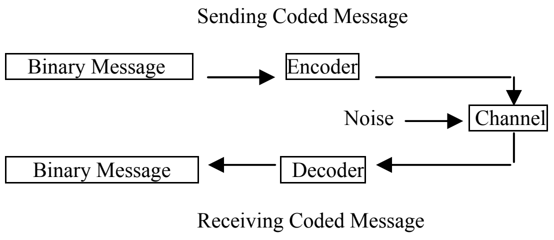

Figure 1: Data Communication Model

In spite of the obvious limitations, the single parity check bit scheme is often use in low speed, non-critical applications. When a higher degree of accuracy is needed, engineers turn to Hamming Code Theory to find and correct errors. The Hamming Code was devised by Richard Hamming in late 1940's and is currently the predominant error detection and correction method in use. What we propose to explain the process by which Hamming Code error correction finds and corrects a single transmission error. It will be seen that vector spaces provide the natural language in which to explain the process.

Hamming Code utilizes the idea of multiple parity check bits. The sole purpose of the parity check bits is to detect the location of transmission errors in the message. Since we are encoding data in a binary fashion, any corrupted bit is simply corrected by replacing the value with its complement (i.e. 0's are replaced with 1's and vice versa.) A major concern in the design of the encoding scheme is that the parity check bits will provide the decoder with enough information to accurately correct transmission errors [3]. Parity check bits are simply appended to the data being transmitted. For example, the transmission of a k bit message will actually require that n bits be transmitted, with n-k parity check bits added to the message, as shown in Figure 2.

It can be seen that the string of message bits, with parity check bits appended, form the elements of a *vector space*. Let $\wedge_2^n$ be the set of all binary n-tuples of numbers from $\wedge_2 = \{0,1\}$. $\wedge_2^n$ is a vector space over $\wedge_2$ by the usual modulo-2 addition and scalar multiplication operations. In $\wedge_2^n$, let $\mathbf{e_1} = (1,0,..., 0)$, $\mathbf{e_2} = (0, 1, 0,..., 0)$,..., $\mathbf{e_n} = (0,..., 0, 1)$. We know $\{\mathbf{e_1}, \mathbf{e_2},..., \mathbf{e_n}\}$ as the standard basis for $\wedge_2^n$ and hence $\wedge_2^n$ is an n-dimensional vector space [2]. Our goal is to explain Hamming Code theory in terms of this vector space. We will explain in detail how Hamming Code theory allows us to encode a message so that a single transmission error can be detected and corrected.

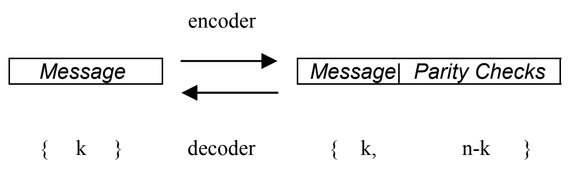

Figure 2: Bit Assignment

Consider the task of transmitting a 4-bit message. We will see that Hamming Code requires us to add 3 parity check bits in order to be able to correct a single transmission error. Therefore, we are actually transmitting a binary string of 7 bits which looks like $\mathbf{c} = (c_1, c_2, c_3, c_4, c_5, c_6, c_7)$ in the $\wedge_2^7$ vector space. Each parity check bit is set equal to the binary sum of a particular subset of the message bits. The parity check bits, c_5, c_6 and c_7, can be defined according to the following rules [1]

$c_5 = c_1 + c_2 + c_4$ (1)
$c_6 = c_1 + c_3 + c_4$ (2)
$c_7 = c_2 + c_3 + c_4$ (3)

These equations may be restated as
$c_1 + c_2 + c_4 + c_5 = 0$
$c_1 + c_3 + c_4 + c_6 = 0$
$c_2 + c_3 + c_4 + c_7 = 0$

Every *code word* (source message plus parity check bits) must satisfy the above parity check equations. In matrix notation the equations may be expressed as

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Define **H** to be the left matrix in the equation, i.e.

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

We call **H** the *parity check matrix*, because the rows of **H** are derived from the parity check equations. In this design, any code word **c** in the code word set, denoted as $C_{7,4}$ which is a subset of $\wedge_2^7$, satisfies $\mathbf{Hc} = \mathbf{0}$. This means any string **c** is in the code word set $C_{7,4}$ if and only if **c** is in the *Null Space* of **H**. Thus, $C_{7,4} = N.S.(\mathbf{H})$. The code word set $C_{7,4}$ is therefore called the code space, which is the vector space of all correctly sent code words. It is impossible for a single transmission error to change one code word into another code word. We explain why with the use of a code space metric function, called the Hamming Distance. In our design, there are only 16 possible code words in the code space $C_{7,4}$. We list them all as follows:

(0, 0, 0, 0, 0, 0, 0) (1, 0, 0, 0, 1, 1, 0) (0, 1, 0, 0, 1, 0, 1)
(0, 0, 1, 0, 0, 1, 1) (0, 0, 0, 1, 1, 1, 1)
(1, 1, 0, 0, 0, 1, 1) (1, 0, 1, 0, 1, 0, 1) (1, 0, 0, 1, 0, 0, 1)
(0, 1, 1, 0, 1, 1, 0) (0, 1, 0, 1, 0, 1, 0)
(0, 0, 1, 1, 1, 0, 0) (1, 1, 1, 0, 0, 0, 0) (1, 1, 0, 1, 1, 0, 0)
(1, 0, 1, 1, 0, 1, 0) (0, 1, 1, 1, 0, 0, 1)
(1, 1, 1, 1, 1, 1, 1)

Since a code space is derived from its parity check equations, the design of the parity check equations determines the code words in the code space. A good choice of parity check equations is one that yields a sufficiently "spaced out" code space, meaning that each code word is in some sense distant from every other code word. To correct a mis-transmitted string means to re-assign the string to its correct code word in the code space, which is identified by being the closest code word to the corrupted string. If two code words are "too close" to each other so that they are equally close to a corrupted string, then we have a 50-50 chance of choosing the correct code word. Naturally this leads to two questions: how do we measure distance between two code words, and how do we define a well "spaced out" code space such that there is a unique nearest code word. *Hamming distance* is the metric used to define how far apart two strings are in a code space.

For any element $\mathbf{x}, \mathbf{y} \in \wedge_2^n$, the *Hamming distance* between $\mathbf{x}$ and $\mathbf{y}$, denoted $d(\mathbf{x}, \mathbf{y})$, is the number of components where $x_i \neq y_i$, for $i = 1,\dots,n$.

In our data transmission example, notice that $d(\mathbf{x}, \mathbf{y}) \geq 3$ for any $(\mathbf{x}, \mathbf{y})$ pair in the code space $C_{7,4}$. This means that any two code words differ by at least 3 bits. In other words, our 16 code words are all at least a *Hamming distance* of 3 units apart. This also means that when exactly one bit has been mis-transmitted, the corrupted string $\mathbf{x_p}$ can be corrected to the closest code word with no ambiguity, since there is only one correct code word $\mathbf{x_c}$ with $d(\mathbf{x_p}, \mathbf{x_c}) = 1$. When two or more errors occur in transmission, we may possibly detect the existence of errors but cannot correct them, because when $d(\mathbf{x_p}, \mathbf{x_c}) \geq 2$, there is a strong possibility that the corrupted string is actually closer to an incorrect code word. In this case, an incorrect code word will mistakenly be assigned to correct $\mathbf{x_p}$. Note that the vector space $C_{7,4}$ along with the Hamming distance metric defined on it, give us a *metric space* in which to work.

Since $C_{7,4}$ is a vector space, without too much difficulty, we can compute a **basis** for $C_{7,4}$ (i.e. N.S.($\mathbf{H}$)) and place it in row vector form in a matrix, $\mathbf{G}$, as follows:

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

This means *R.S.*($\mathbf{G}$) = *N.S.*($\mathbf{H}$) (i.e. the *Row Space* of $\mathbf{G}$ equals to the *Null Space* of $\mathbf{H}$). Since *rank*($\mathbf{H}$) = 3, the dimension of *N.S.*($\mathbf{H}$) = 7 – 3 = 4 = *rank*($\mathbf{G}$). The code space $C_{7,4}$ generated by $\mathbf{G}$ is thus a 4-dimensional subspace in the 7-dimensional vector space $\wedge_2^7$. We call $\mathbf{G}$ the *generator matrix* of the code space $C_{7,4}$. Since the set of row vectors, $\{\mathbf{g_1}, \mathbf{g_2}, \mathbf{g_3}, \mathbf{g_4}\}$, of $\mathbf{G}$ form a basis for the code space $C_{7,4}$, any code word in $C_{7,4}$ can be expressed as a linear combination of $\mathbf{g_i}$'s. We also see that the rows of the generator matrix $\mathbf{G}$ and the rows of parity check matrix $\mathbf{H}$ are *orthogonal* to each other by observing that is $\mathbf{G} \cdot \mathbf{H}^\mathbf{T} = 0$. Hence *N.S.*($\mathbf{G}$) = *R.S.*($\mathbf{H}$) = *C.S.*($\mathbf{H}^\mathbf{T}$) (i.e. *Column Space* of $\mathbf{H}^\mathbf{T}$). The column space of $\mathbf{H}$, *C.S.*($\mathbf{H}$), provides useful information regarding the position of the error. For any received string $\mathbf{x_r}$, the product of the parity check matrix $\mathbf{H}$ and $\mathbf{x_r}$ is called the *syndrome vector*. We see that any syndrome vector $\mathbf{Hx_r} \in \wedge_2^3$ = *C.S.*($\mathbf{H}$) and every non-zero vector of $\wedge_2^3$ is a column of $\mathbf{H}$. Thus if $\mathbf{x_r} \in \wedge_2^7$ but $\mathbf{x_r} \notin C_{7,4}$ (i.e. $\mathbf{x_r}$ is a string, but not a code word), then $\mathbf{Hx_r} \neq 0$. Therefore, in order for $\mathbf{x_r}$ to be correct, $\mathbf{Hx_r}$ must be in the column space of $\mathbf{H}$. We call *C.S.*($\mathbf{H}$) the *syndrome space* generated by $\mathbf{H}$ and $\mathbf{x_r}$. We will discuss the details of *syndrome decoding* as an alternative decoding method.

2. Code Words in Code Space

2.1 Encoding Messages

To encode a message means to convert the message from $\wedge_2^4$ to a vector in code space $C_{7,4}$. We define the converting function $F : \wedge_2^4 \rightarrow C_{7,4}$, as

$$\begin{aligned} F(x_1, x_2, x_3, x_4) &= F(x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + x_3\mathbf{e}_3 + x_4\mathbf{e}_4) \\ &= x_1\mathbf{g}_1 + x_2\mathbf{g}_2 + x_3\mathbf{g}_3 + x_4\mathbf{g}_4, \end{aligned}$$

where $\{\mathbf{e_i}\}_{\{i=1,\dots,4\}}$ with the i^{th} component being 1 and all others being 0 is the standard basis for $\wedge_2^4$, and $\{\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3, \mathbf{g}_4\}$ is a basis for $C_{7,4}$. Since $x_1\mathbf{g}_1 + x_2\mathbf{g}_2 + x_3\mathbf{g}_3 + x_4\mathbf{g}_4 = (x_1, x_2, x_3, x_4) \cdot \mathbf{G}$, the generator matrix $\mathbf{G}$ is also a matrix induced by the converting function. For example, to encode (0, 1, 1, 1), function F converts (0, 1, 1, 1) = $\mathbf{e}_2 + \mathbf{e}_3 + \mathbf{e}_4$ to $\mathbf{g}_2 + \mathbf{g}_3 + \mathbf{g}_4$ = (0, 1, 1, 1, 0, 0, 1) which is the same as converting (0, 1, 1, 1) through G by using (0,1,1,1) $\cdot$ $\mathbf{G}$. Now the coded message has 3 attached parity check bits.

2.2 Decoding Messages

To decode a message means to check the message to determine if there has been an error, and if so, correct the error and extract the original message. The parity check matrix $\mathbf{H}$ will determine if the message is received correctly. Again here we assume there is at most one bit error. For example when a message is received as

$$\mathbf{x} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

the fact that

$$\mathbf{Hx} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

indicates $\mathbf{x} \in$ *N.S.*($\mathbf{H}$), which means $\mathbf{x}$ was received correctly. Therefore, the source message is (0, 1, 1, 1) which is $\mathbf{x}$ without the attached parity check bits.

2.3 Error Detecting and Correcting

If a message is received as

$$\mathbf{y} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

the fact that

$$\mathbf{Hy} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

indicates **y** was received incorrectly, because $\mathbf{y} \notin N.S.(\mathbf{H})$. Since the string (0, 0, 1, 1, 1, 0, 0) is the nearest code word to **y** in $C_{7,4}$ (the only code word with distance one from **y**), the encoder will correct the received string **y** to the code word $\mathbf{y_c}$ = (0, 0, 1, 1, 1, 0, 0). We can see, even by this simple example, that performing a full-scale nearest distance search through the entire code space is rather tedious, and it should be avoided. A slightly more sophisticated method called "syndrome decoding" can be used instead.

2.4 Syndrome Decoding

The method of syndrome decoding is used to avoid performing the calculation of Hamming distance between $\mathbf{x_r}$ and every single vector in the code space in order to identify the nearest code word. Rather, the syndrome vector reveals, through the pattern of parity check failures, the location of the error in $\mathbf{x_r}$. A transmitted string **y** = (0, 0, 1, 0, 1, 0, 0) with syndrome vector **Hy** = (1, 1, 1), indicates **y** failed to satisfy all three of equations 1, 2 and 3. From Table 1, we see that the syndrome vector indicates an error in bit 4. Correcting the 4th bit in **y** will correct the received string. Hence, the corrected string is the code word $\mathbf{y_c}$ = (0, 0, 1, 1, 1, 0, 0), and the source message is (0, 0, 1, 1).

Table 1: Parity Check

	possible bit error 1	2	3	4
failed eqn. 1	x	x		x
failed eqn. 2	x		x	x
failed eqn. 3		x	x	x

If a string is received as **z** = (1, 1, 1, 1, 0, 1, 0) and the syndrome vector is **Hz** = (1, 0, 1), then **z** failed to satisfy equations 1 and 3. From parity check Table 1, bit 2 was not received correctly, and therefore the correct string is the code word $\mathbf{z_c}$ = (1, 0, 1, 1, 0, 1, 0), and the correct source message is (1, 0, 1, 1). A syndrome vector of the form (1, 0, 0), shows failure of equation 1 and therefore bit 5 of the received string was in error. Table 2 shows how the syndrome vector can be used to find the error bit for an arbitrary syndrome vector **Hv**. It is interesting to note that if syndrome vector **Hv** equals the *i*th column vector in **H**, then the *i*th bit of the received string **v** is in error.

Table 2: Error Location Table

	0: eqn. satisfied; 1: eqn. failed						
eqn. 1	1	1	0	1	1	0	0
eqn. 2	1	0	1	1	0	1	0
eqn. 3	0	1	1	1	0	0	1
bit error location	1	2	3	4	5	6	7

3. Conclusion

What we have developed in this example is a motivation for the understanding of vector space, basis, null space, row space, column space and metric space as they apply to data transmission theory. Hamming code, as it applies to the correction of a single transmission error, is explained via vector space theory. Various methods of data transmission and error correction are chosen based on the particular needs of each project by weighing the cost of error correction (reduced speed of transmission) against the need to correct corrupted data. In order to detect and correct multiple bit errors, we would have to create a larger and more "spaced out" code space. It is hoped that this example will show students of applied mathematics and computer sciences that the study of vector spaces has real world applications, as well as provide students a solid example with which they can understand some abstract vector space concepts.

References

[1] Berlekamp , E., *Algebraic Coding Theory*, McGraw-Hill Book Company, 1968.
[2] Hill, R., *Elementary Linear Algebra with Applications,* Saunders College publishing, 1996.
[3] Lin, S., *An Introduction to Error-Correcting Codes,* Prentice-Hall, 1970.