The Mathematics of the Compact Disc

von Jack H. van Lint

In this lecture we consider a branch of discrete mathematics, namely the theory of error-correcting codes and its application to the design of the Compact Disc audio system developed by Philips Electronics and Sony.

1. Words and Codes.

We shall need the following easy mathematical concept. If we consider two $n$-tuples $a = (a_1, a_2, \ldots, a_n)$, $b = (b_1, b_2, \ldots, b_n)$, where the $a_i$ and $b_i$ are from some set $Q$, which we call the alphabet, then the Hamming distance of $a$ and $b$ is defined by

$$d(a, b) := |\{i : a_i \neq b_i, 1 \leq i \leq n\}|.$$

We shall call $a$ and $b$ words. The analogy with language is intentional. Suppose we read a word (say in a book in English) and we see that the printed word contains two misprints. Then in our terminology, the printed word has Hamming distance 2 to the correct word. The reason that we are able to see that there are two misprints is the fact that the English language contains only one word with such a small distance to the printed word. That is also the main principle of error-correcting codes: construct a language (which we call a code) of words of some fixed length over a given alphabet $Q$ such that any two codewords have distance at least $2e + 1$. This is called the minimum distance of the code. Clearly a word that contains at most $e$ errors is closer to the original word than to any other codeword. This allows us to correct those errors.

2. A simple example.

We actually see examples of encoded information every day in the form of the bar-codes with which most products are identified nowadays. First, each product is identified by a sequence of twelve decimal symbols. Subsequently, each decimal symbol is replaced by a codeword consisting of seven 0’s and 1’s. On the product, these are represented by a thin vertical white bar for 0 and a black bar for 1. For example, 5 is encoded as 0110001 which is seen on the product as a white bar, one unit wide, followed by a black bar with a width of two units, a white one of three units and a final black bar. When the product is scanned in the supermarket, it sometimes happens that a bit (i. e., a 0 or a 1) is interpreted incorrectly. The codes for the digits and those for the products were designed in such a way that the scanning device establishes the occurrence of an error and gives a signal that the scanning should be repeated. We see this happen quite regularly.

One of the early successes of error-correcting codes was the quality of pictures taken by satellites, e. g. of
Mars. If these pictures had been transmitted to earth without using error-correcting codes they would have shown no image at all; just random noise!

The first efficient error-correcting code was designed by R. W. Hamming at Bell Laboratories in 1948. A binary information sequence was split into fourtuples. Hamming’s idea was to adjoin three redundant bits to each fourtuple according to the following rule. Consider the Venn diagram below.

![Venn Diagram](image)

The four bits (say 1101) are written in the parts numbered 1 to 4. The redundant bits come from the parts 5 to 7. The rule is that each circle must contain an even number of 1’s. A configuration of seven bits that contains one error will result in some circle(s) with odd parity. The bit that is contained in each of these but not in the circle(s) with even parity is the one in error. The reader can easily convince himself that if we see a sequence of seven bits, two of which are not wrong but illegible (so-called erasures), then the erased bits can be retrieved. (It is not possible that all three the circles contain two erased bits.)

We shall indicate the fact that in this coding scheme a codeword consists of four bits that carry information and three redundant bits by saying that the information rate is $4/7$. The code is called a binary $[7,4]$ code. Note that repeating a sequence of four bits produces an $[8,4]$ code, i.e., a code with information rate $4/8=1/2$ (smaller than in our example) that can only detect one error but not correct it.

3. The audiobits

Before looking at the error-correcting codes used for CD we consider the conversion of music into a sequence of so-called audiobits. In the Compact Disc system the analog signal is sampled at a rate of 44100 times per second. By Nyquist’s sampling theorem this rate of 44.1 kHz suffices for reproduction of frequencies up to 20000 Hz. The samples are uniformly quantized to 16 bits. Since we are considering stereo music, each sample leads to 32 bits (the audiobits). This sequence is seen as four consecutive bytes (a byte is a sequence of eight bits). For the encoding algorithm the bytes are considered as elements of the field $\mathbb{F}_2$. Just as in the example of the Hamming code, the sequence of bytes is divided into groups of a certain fixed length and then redundant symbols are added. In the CD system an information sequence of 24 bytes is encoded in two steps to a codeword of length 32 (bytes). The codes that are used are so-called $[28,24]$ and $[32,28]$ Reed-Solomon codes. An extra byte is added containing the Control and Display bits. This allows the player and the listener to see where we are on the track. So six samples lead to 33 data symbols, each consisting of eight bits.

As will be explained later, the sequence of data bits cannot be recorded on the disc. The conversion into the sequence of channel bits (that produces the track) transforms eight data bits to seventeen channel bits. After each sequence of 33 times 17 channel bits a sequence of 27 synchronization bits is inserted. This identifies uniquely the beginning of the next coded sequence. The synchronization sequences are recognizable because they are chosen in such a way that they cannot possibly occur within a coded sequence. All together, this implies that one second of music leads to a sequence of 4321800 bits on the track. If the probability that the player interprets a bit incorrectly were a random event with the very small probability of $10^{-4}$, then there would still be hundreds of errors occurring every second!

There are several causes of errors on a CD. Some are: small unwanted particles in or on the disc, air...
bubbles in the plastic material, inaccuracies due to stamping, fingerprints, scratches, surface roughness. We remark that the errors occur mainly in bursts. Since we use codes designed to cope with random errors, the recording requires a scrambling of the bytes (cross interleaving) which has the effect that bytes that are adjacent in codewords are not adjacent on the disc.

4. Reed-Solomon codes

The coding scheme used on the compact disc is called CIRC (Cross-Interleaved Reed-Solomon Code). The aim of this lecture is to explain the mathematical principles that play a role in the error-correction for the CD. To do this we shall use a completely different example. In this example, the codes that we describe are Reed-Solomon codes and the encoding and decoding use the same ideas as in the CD system. As explained in the previous section, the alphabet for the CD is \( \mathbb{F}_2^6 \). All calculations are done in that field. To avoid the complicated calculations in this field and to make it possible to check the arithmetic operations, we choose an example where the alphabet has only 31 symbols, namely \( \mathbb{F}_{31} \). All arithmetic operations are as usual but the results are reduced mod 31. (So \( 15+16=0 \) and \( 5 \times 9 = 14 \).)

As our example, we consider a book of 200 pages that is to be printed with a font that allows 3000 symbols per page. We identify the letters a to z with the integers 1 to 26, the space with 0, and use 27 to 30 for the period, comma, semicolon and one other punctuation mark, respectively.

We are informed by the printer that the procedure that he uses is not too good. In fact, it is so poor that each symbol on a printed page has a probability of 0.1% of being incorrect. This implies that on average each page will have about three misprints which is rather sloppy. Remember how annoying even an occasional tick on a gramophone record is. We shall use coding to improve the quality of the book. As a first example, we divide the sequence of symbols to be encoded into groups of four symbols. To each of these groups two redundant symbols will be added. We will denote the resulting codeword by \( a = (a_0, a_1, \ldots, a_5) \), where \( a_0 \) and \( a_1 \) are the two redundant symbols and \( a_2 \) to \( a_5 \) are the elements of \( \mathbb{F}_{31} \) corresponding to the four symbols of the group.

Let us consider the word CODE. This corresponds to \( (a_2, \ldots, a_5) = (3, 15, 4, 5) \). The encoding rules are:

\[
\begin{align*}
  a_0 + a_1 + \cdots + a_5 & \equiv 0 \pmod{31}, \\
  a_1 + 2a_2 + \cdots + 5a_5 & \equiv 0 \pmod{31}.
\end{align*}
\]

From (ii) we find \( a_1 = 1 \) and then (i) yields \( a_0 = 3 \). So, CODE will be encoded as CACODE.

Suppose we see EPGOOD. Leaving out the redundant symbols would yield GOOF. However, we shall not goof! In fact, we immediately see that this word is wrong because EPGOOD corresponds to \( (a_0, \ldots, a_5) = (5, 16, 7, 15, 15, 6) \) and the sum of the symbols \( a_i \) is 2 (mod 31). In fact, we conclude that probably one of the symbols \( a_i \) has been replaced by \( a_i + 2 \). The value of the error in \( a_1 \) is 2 and therefore we will find 2\( i \) if we substitute \( a_0 \) to \( a_5 \) in (ii). We find

\[
a_1 + 2a_2 + \cdots + 5a_5 \equiv 10 \pmod{31}.
\]

Hence \( i = 10/2 = 5 \). So the 6 in position 5 should be a 4 and the correct word is GOOD (encoded as EPGOOD). Note that encoding and decoding are based on arithmetic operations including division. This is more complicated work in \( \mathbb{F}_{2^8} \) but for all Reed-Solomon codes the defining rules are like (i) and (ii).

When we show what this code achieves for us, we have to be honest and introduce a problem in this example that also occurs on the CD. Since we wish to use the same number of pages of the same size, we have to print 1½ times as many symbols on a page. (The code described above has information rate \( 4/6 \).) To our dismay the printer tells us that for the smaller font that we now have to use, the probability of error is twice as large! This implies that before error correction a printed page will contain nine errors on average (using the smaller font). What about after error correction? A word of six symbols is decoded correctly if it contains one error (or no errors at all). The probability that it contains more than one error is

\[
\sum_{i=2}^{6} \binom{6}{i} (0.002)^i (0.998)^{6-i} \approx 0.00006.
\]

In the book, there will now be some words that contain two or three misprints but we do not expect more than 10 such words in our book, a considerable improvement.

Now things become really impressive. We are going to use a slightly more complicated Reed-Solomon code. We divide the symbols into groups of eight, which we call \( (a_4, a_5, \ldots, a_{11}) \). To these we adjoin four redundant symbols in front with the following rules:

\[
a_0 + a_1 + \cdots + a_{11} \equiv 0 \pmod{31},
\]

\[
a_1 + 2^k \cdot a_2 + \cdots + 11^k \cdot a_{11} \equiv 0 \pmod{31},
\]

\[
(k = 1, 2, 3).
\]

Encoding is a question of solving four linear equations with four unknowns. The reader can easily see how one error is corrected. If there are two errors, say of
size $e_1$ and $e_2$ in positions $i$ and $j$, then substitution of the word in the equations (i) and (ii) tells us the value of $e_1 + e_2$ and $i^ke_1 + j^ke_2$ for $k = 1, 2, 3$. Eliminating $e_1$ and $e_2$ yields a quadratic equation with $i$ and $j$ as solutions. After that, we can solve for $e_1$ and $e_2$. We now have a 2 error-correcting code.

Luckily the printer does not create new problems. This Reed-Solomon code also has rate $2/3$, like the previous one. So our error probability is again $0.002$ per symbol. Decoding will only fail if a printed word of twelve symbols contains more than two errors. The probability that this happens is

$$\sum_{i=3}^{12} \binom{12}{i} (0.002)^i (0.998)^{12-i} < 0.000002.$$ 

So, after error correction it is unlikely that there is a misprint at all in the book. This is indeed an impressive improvement.

On the CD the rate of the error-correcting code is $3/4$. This also increases the bit error probability compared to the situation in which no coding is used. As mentioned earlier, a CD can easily contain 500000 errors. As mentioned before, these are not random errors as in the example of the book but they tend to occur in bursts. It can happen that a few thousand adjacent symbols are wrong. This is combated by the interleaving which sees to it that adjacent information symbols end up in different codewords. One of the things that had to be kept in mind in the design of the CD-system was that the memory needed to store the symbols to be processed could not be too large. This implied restrictions on the length of the code and on the interleaving. All the computation necessary to correct errors is done in a fraction of a second and so we start hearing music practically immediately after the player is switched on. In the example of the book we saw that a longer code has a better performance but implies more calculations.

As mentioned earlier, the encoding for the CD is done in two steps using a [28,24] code $C_1$ and a [32,28] code $C_2$. These are short and not very strong codes. Each has minimum distance 5 and can therefore correct only two errors in a received word. One of the main reasons for the effectiveness of the coding scheme is the fact that the two codes collaborate. Instead of describing the actual scheme used on the CD, we describe a very similar idea in which $C_1$ and $C_2$ also collaborate to achieve remarkable results. This idea is a so-called product code. Suppose that $24 \times 28$ information symbols are taken as entries of a 24 by 28 matrix. This matrix is taken as the upper left hand corner of a 28 by 32 matrix $A$. The remaining (redundant) symbols of $A$ are defined by requiring that each column of $A$ is in the code $C_1$ and each row of $A$ is in $C_2$. Since the two codes $C_1$ and $C_2$ are vector spaces over the alphabet, the two requirements can indeed be satisfied simultaneously.

It is not difficult to show that this product code has minimum distance 25 and therefore can correct up to 12 errors in a word (i.e., a matrix). Consider a particularly bad situation where 21 errors occur, namely 12 columns with one error, one column with two errors, one column with three errors and one column with four errors. We do not expect to be able to handle this situation and indeed, decoding the columns is not feasible. We decide to use $C_1$ to correct only one error. This has the effect that we recognize the columns with two or three errors because they have distance at least 2 to the closest codeword. In the case of the column with four errors it is not unlikely that the error correction algorithm thinks it has corrected one error but in fact it has introduced a fifth error! Now comes the trick. We correct the columns we (think we) can and the two columns that were recognized as suspicious are declared erased. This means that after handling all the columns of this matrix, two are erased, 29 are correct and one ended up with five errors. Now look at the rows, which are supposed to be codewords of the code $C_2$. In each row there are now two erasures and in five of the rows there is an error. Since the code $C_2$ has minimum distance 5, all the rows are decoded to the correct codeword.

On the earlier CD system the idea was similar. One of the two codes did not correct as many errors as it could and this made it possible to recognize words with two or three errors. These were considered as erased and again the construction of the code was such that erased words of one code had their symbols in different words of the other code. On later systems the codes are used to full capacity but if more than one error is corrected, the word is marked as suspicious. Subsequently, suspicious words can be erased in case decoding difficulties arise on the second step. For the CD some tricks were added for situations in which the code could not handle the error pattern. For example, an unreliable word with reliable neighbors could be replaced by the average value of these neighbors; (remember that the information words are values of samples of the analog audio signal).

5. The Compact Disc

The music is recorded on a Compact Disc in digital form as a 5 km long spiral track consisting of a succession of pits. The parts of the track between successive pits are called lands. The pitch of the track is 1.6 $\mu$m, the width is 0.6 $\mu$m and the depth of the pits is 0.12$\mu$m. The figure below shows an enlargement of several parallel tracks (from [1]).
The Mathematics of the Compact Disc

The track is optically scanned by a laser beam. Each land/pit or pit/land transition is interpreted as a 1, all other bits are 0. Here a (channel)bit on the track has length 0.3 μm. So, a pit of length 1.8 μm followed by a land of length 0.9 μm is interpreted as the sequence 100000100. The diameter of the light spot is about 1 μm. When it falls on a land, it is almost totally reflected. Since the depth of a pit is about 1/4 of the wavelength of the light in the material of the disc, interference causes less light to be reflected by the pits into the aperture of the objective.

There are several requirements on the sequence of bits stored on the CD. Each pit or land must be at least three bits long. This is to avoid that the light spot would see two successive transitions between pits and lands simultaneously, causing so-called inter-symbol interference. Every time there is a transition, the bit clock in the CD player is synchronized. Since this must happen in time to avoid loss of synchronization, no pit or land may be longer than 3.3 μm, i.e., two 1’s in the sequence are separated by at most ten 0’s. The restriction on the maximal landlength is also necessary for the servomechanism that keeps the laser on the track. A final requirement is that the low-frequency content of the signal read from the disc is minimal. This implies that the total length of the pits and lands, accumulated from the beginning of the disc, remains nearly the same. This requirement is necessary for the servomechanism that regulates the decision level between what is considered as “light”, respectively “dark”, in the reflected light. (This decreases the negative effect of a fingermark on a disc.)

We now look at the conversion from the bytes (that are the letters in our codewords) to the sequences that are actually recorded on the disc. This is known as EFM (=Eight-to-Fourteen Modulation). The “fourteen” has to be explained. Clearly a number was needed that made it possible to satisfy the constraints and was as small as possible.

A well known problem, occurring in many elementary courses on combinatorics is that of determining the number of sequences of 0’s and 1’s of length n in which no two 1’s are adjacent. Let us call this number $F_n$. A sequence ending in a 1 must end in 01. In front of these two symbols can be any sequence of length $n-2$ with no adjacent 1’s. Similarly, a sequence ending in a 0 starts with any acceptable sequence of length $n-1$. Therefore we have $F_n = F_{n-1} + F_{n-2}$. Since $F_0 = 1$ and $F_1 = 2$, we find $F_2 = 3$, $F_3 = 5$, $F_4 = 8$, etc.; the well known Fibonacci sequence.

If we denote by $a(n)$ the number of sequences of 0’s and 1’s of length $n$ with at least two 0’s between two consecutive 1’s and never more than ten adjacent 0’s, then a similar recurrence relation for $a(n)$ can be set up. We need to translate the 256 possible bytes into sequences satisfying these constraints. It turns out that $n = 14$ is the smallest value of $n$ for which $a(n) > 256$, namely $a(14) = 267$.

We are faced with one more problem. If we concatenate two sequences that both satisfy the constraints, then the resulting 28-tuple may violate the constraints. After removing the eleven most difficult sequences from the 267, it turns out that it is possible to merge two fourteen-tuples using three merging bits and still have some freedom in the choice of these merging bits. These are then chosen in such a way that along the track the running sum length of pits and lands is nearly the same. This sees to it that the low frequency content of the read signal is as small as possible.

We have shown that each byte in a codeword results in seventeen channel bits on the disc. When the disc is read, these are translated back into bytes (by a table lookup). Then follow de-interleaving, error correction, digital to analog conversion and finally, we hear perfect music with thanks to Mozart and others and also to mathematics.

The reader interested in more details (about the earlier systems) should consult the references.

Bibliography


Adresse des Autors:
Jack H. van Lint
Stan Ackermans Institute
Eindhoven University of Technology