ERROR CORRECTING CODES FROM COMBINATORIAL GAME THEORY

ROGER FAN

Abstract. Linear error correction codes are closely related to certain games in the field of combinatorial game theory. In this essay, we outline their relationship and a few key facts.

1. Introduction

Linear codes are motivated by the idea of sending only a select few code words that are sufficiently distinct from one another, so that even when an error occurs during transmission, the receiver can still make out what the original code word was.

Hamming Codes are a specific class of linear codes that correspond to the impartial game of Nim. However, we will show that linear codes in general correspond to the game of Turning Turtles, described later.

2. Preliminaries in Coding Theory

For the purposes of this essay, a code word will be a string of n digits. A *binary* code word will be a code word such that all digits are either 0 or 1.

We must first formalize the concept of code words being sufficiently distinct from one another. We do so by using Hamming distance:

Definition 2.1. The *hamming distance* between two equal-length strings $A = (\cdots a_3a_2a_1)$, $B = (\cdots b_3b_2b_1)$ is the number of i such that $a_i \neq b_i$. I sometimes express the hamming distance between A and B to be hamming (A, B).

Using this, we can require that code words must be a hamming distance of d away from each other to be sufficiently distinct. We will later see that this criteria will be useful in correcting errors.

Now, we can define a code:

Definition 2.2. A *code* with length n and hamming distance d is a set of *code words*, which are strings of length n that are at least a hamming distance of d away from each other pairwise.

We define an m-digit error to be a change in a code word where exactly m digits are changed.

Now, say the sender sent the code word S. If the receiver received the string S', then they could attempt to rectify any errors by assuming the sender sent the code word that is closest (in hamming distance) to S'. We call this strategy the "nearest neighbor rule", and in general, we call this process "decoding".

Date: August 15, 2021.

Proposition 2.3. *A code of hamming distance $d = 2m + 1$ can detect and correct up to m-digit errors. A code of hamming distance $d = 2m$ can detect up to m-digit errors and correct up to $(m - 1)$-digit errors.*

Proof. First, note that if given 3 strings A, B, C, $\text{hamming}(A, B) + \text{hamming}(B, C) \geq \text{hamming}(A, C)$. This is because if $a_i \neq c_i$, then either or both of $a_i \neq b_i$ or $b_i \neq c_i$ are true. The inequality then follows.

If $d = 2m+1$, we claim that any m-digit error can be corrected by the receiver. If a m-digit error occured, then $\text{hamming}(S, S') = d$. For any other code word T, note by definition, $\text{hamming}(S, T) = d = 2m + 1$. Then, by the above inequality,

$$\text{hamming}(S', T) + \text{hamming}(S, S') \geq \text{hamming}(S, T)$$

$$\text{hamming}(S', T) \geq m + 1$$

Specifically, this means that the code word S is closer in hamming distance to S' than any other code word. Therefore, the receiver can successfully rectify this m-digit error. (This can also be extended to any m-or-less-digit error.)

Also, note that for any p-digit error with $p > m$, the receiver will actually mistake it for a $2m + 1 - p$ digit error and will end up with the wrong code word. However, codes with an even hamming distance can somewhat mitigate this shortcoming:

The case for when $d = 2m$ works the same as for $d = 2m + 1$: all $(m - 1)$-or-less-digit errors can be successfully corrected.

However, when an m-digit error occurs, the receiver could still recognize that the distance to the nearest code word is m; there just might be two such code words. In this case, the receiver knows that an m-digit error occured rather than mistaking it for a different error, and the proposition is proven. ■

To construct such a code, one can use a greedy algorithm:

Definition 2.4. A *lexicode* of length n and distance d is the code that is generated by starting with an empty set of code words $S = \emptyset$, then greedily adding the lexicographically least length-n string that differs from all code words already in S by a hamming distance of at least d.

We will show that lexicodes hold surprising relations to impartial games.

3. The Game of Turning Turtles

Now, we must investigate an entirely different area of mathematics: combinatorial game theory, the study of games without chance.

In particular, we would like to study the impartial game of Turning Turtles. Each game of Turning Turtles is played with a line of turtles, but to avoid wanton animal cruelty, we play it with a line of coins instead. The rules are determined by some integer k: during your turn, you must flip k coins, given that the leftmost coin is flipped from heads to tails.

Notice we can represent any game of Turning Turtles as a binary string. We assign a digit for each coin in the line; More specifically, we construct a string of digits $(\cdots \zeta_3\zeta_2\zeta_1)$, where ζ_i is 0 (1) if the ith rightmost coin is tails (heads).

Remark 3.1. When written out, the 0s to the left of the leftmost 1 are often omitted. For example, the string $\cdots 0000000101011$ could be written simply as 101011.

Theorem 3.2. *The binary lexicode of length n and distance d is exactly the set of all $\mathcal{P}$ positions of* Turning Turtles *with $k = d - 1$. These $\mathcal{P}$ positions must also satisfy the condition that the largest i such that $\zeta_i = 1$ must have $i \leq n$.*

Proof. The proof is similar to the proof done in class, just more general.

First, note that there is a valid move from a game of Turning Turtles represented by string S to a game represented by string T if and only if they differ by k or fewer digits. Moreover, T must be lexicographically smaller than S. The reader can quickly convince themselves of these facts using the ruleset of Turning Turtles.

Now, let $\mathscr{P}$ be all code words in said lexicode, and let $\mathscr{N}$ be all strings of length n not in the lexicode. We treat these strings as games of Turning Turtles with $k = d - 1$.

Any move from a $\mathscr{P}$ game must go to a $\mathscr{N}$ game. This follows from the fact that any move can only change k digits of the game, and $\mathscr{P}$ games, by construction, differ pairwise by $d = k + 1$ positions.

Also, from any $\mathscr{N}$ game, one can move to a $\mathscr{P}$ game. This too is due to the way $\mathscr{P}$ is constructed: note that given an $\mathscr{N}$ game G, if there were no $\mathscr{P}$ position that it could move to, then all $\mathscr{P}$ positions would be either:

(1) lexicographically larger
(2) Differing to G by at least $k + 1 = d$ positions

In particular, this means that all lexicographically smaller $\mathscr{P}$ positions would differ to G by at least d positions, but this is a contradiction: if this were true, G would be an $\mathscr{P}$ position! Thus, there exists some $\mathscr{P}$ position that G can move to.

By a standard result in combinatorial game theory, this suffices to show that $\mathscr{P}$ are the $\mathcal{P}$ positions and $\mathscr{N}$ are the $\mathcal{N}$ positions. ∎

4. Perfect Binary Codes

We define a perfect code as follows:

Definition 4.1. Consider a lexicode of length n and hamming distance $d = 2m + 1$. It is *perfect* if for every string of length n, there is exactly one code word that is a hamming distance of m or less away.

Notice that perfect codes must have odd hamming distances. This is due to the fact that a code with an even hamming distance $d = 2m$ does not know how to correct a string with hamming distance m away from the nearest code words.

Proposition 4.2. *Let $|C|$ be the number of codewords in the code C. A binary lexicode C of length $n = 2m + 1$ is perfect if and only if*

$$|C| \cdot \sum_{i=0}^{m} \binom{n}{i} = 2^n$$

Proof. First, in a perfect code, every code word S corresponds to some number of strings that will be decoded to S. The number of such strings will be the number of strings that differ from S by m or fewer positions. Namely, this is $\sum_{i=0}^{m} \binom{n}{i}$.

(To see this, choose some $0 \leq i \leq m$. The number of strings that differ from a codeword by i digits can be constructed: you simply need to choose i digits from the code word's n digits to "flip", turning 1s to 0s and vice versa. This is just $\binom{n}{i}$ such strings. Then, sum it over all i's.)

It follows that the number of total strings 2^n is simply $\sum_{i=0}^{m} \binom{n}{i}$ times the number of codewords, so $2^n = \sum_{i=0}^{m} \binom{n}{i} \cdot |C|$.

For the other way around, in any binary lexicode, each string of length n must correspond to either 1 or 0 codewords. (A string cannot correspond to multiple.)

The same reasoning from before can show any code word corresponds to $\sum_{i=0}^{m} \binom{n}{i}$ strings. Then, there must be $|C| \cdot \sum_{i=0}^{m} \binom{n}{i}$ total strings that correspond to a code word. However, if it is 2^n, then *all* of the strings must correspond to a single code word, and we are done. ■

Remark 4.3. It is a necessary but *insufficient* condition that for any perfect code, $\sum_{i=0}^{m} \binom{n}{i} | 2^n$. The only way to find if a code truly exists is to find $|C|$ and show it satisifes $|C| \cdot \sum_{i=0}^{m} \binom{n}{i} = 2^n$.

There exist some simple perfect codes. The first kind is the lexicode with only 2 code words: all 0s and all 1s. This occurs when $d = n$ (and both are odd).

Then, there exist perfect codes where the code words are simply all the strings of length n. This occurs when $d = 1$.

Both of these families of perfect codes are not very useful. The first can only encode 2 code words, whereas the second cannot even correct a single error. We call these perfect codes the *trivial perfect codes*.

However, there exist non-trivial perfect binary codes as well.

Definition 4.4. Let the binary code with $n = 2^p - 1$ and $d = 3$ be the *Hamming Code* of length $2^p - 1$. It can correct a single bit error.

Of course, Hamming Codes correspond to the game of *Turning Turtles* with $k = 2$.

However, there is something special about the game of *Turning Turtles* with $k = 2$. It is actually the game of *Nim* in disguise!

For the sake of brevity, I will omit the description of NIM and will simply state its relation to TURNING TURTLES without proof, as we already have seen this during class.

Proposition 4.5. *The game of* TURNING TURTLES *with* $k = 2$ *is exactly the game of* NIM *in disguise. Each digit* ζ_i *in the game of* TURNING TURTLES *represents the number of piles of size* i *in the game of* NIM.

Example. The game of TURNING TURTLES with $k = 2$ and representation 1010010 is the game of NIM with 3 piles of sizes 2, 5, and 7, resp. Similarly, 1101010 corresponds to 4 piles of sizes 2, 4, 6 and 7.

Thus, it is relatively easy to verify if a string of length n is a code word in the Hamming Code; simply find if the underlying NIM game is a $\mathcal{P}$ position.

Theorem 4.6. *All Hamming Codes are perfect.*

Proof. Note that by definition, $n = 2^p - 1$ and $d = 3$. Since $d = 2 \cdot 1 + 1$, $m = 1$. We can first find $\sum_{i=0}^{m} \binom{n}{i} = 1 + n = 1 + 2^p - 1 = 2^p$. $2^p | 2^n$, which suggests that it might be a perfect code, but this is an insufficient condition. To show that Hamming Codes are indeed perfect, we must show $|C| = 2^{n-p}$.

We can find this by considering the "incomplete" binary string $(\zeta_n \cdots \zeta_{10}\zeta_9\zeta_7\zeta_6\zeta_5\zeta_3)$, where all ζ_t in which t is a power of 2 are omitted. There are $n - p$ digits in this string (since $n = 2^p - 1$, all ζ_{2^k} where $0 \leq k < p$ are omitted). Thus, because each digit can only be 0 or 1, there are 2^{n-p} such incomplete strings.

I claim there is a one-to-one correspondence between these incomplete strings and $\mathcal{P}$ positions of NIM. (That is, the $\mathcal{P}$ positions of NIM where the largest pile is at most n. This detail is assumed and not stated explicitly in the rest of the proof.)

Treat the incomplete string $(\zeta_n \cdots \zeta_{10}\zeta_9\zeta_7\zeta_6\zeta_5\zeta_3)$ as a NIM game where for each $\zeta_i = 1$, there is a pile of size i, and for each $\zeta_i = 0$, there is not a pile of size i. We do not know if there are piles of sizes $1, 2, 4, 8 \ldots$ yet.

Then, we can uniquely choose a subset of the piles of sizes $1, 2, 4, 8 \ldots$ to include such that the resulting NIM game becomes a $\mathcal{P}$ position. To do so, find the nim-sum of all the piles included so far by the incomplete string. More formally, find $S = \bigoplus_{\zeta_i=1} i$.

Represent S in binary so that $S = a_{p-1} \cdot 2^{p-1} + \cdots + a_1 \cdot 2 + a_0 \cdot 1$, where $a_i \in \{0, 1\}$. Notice that S has at most p digits, as all integers less than or equal to $n = 2^p - 1$ cannot have a nonzero 2^p digit.

For each a_i, if $a_i = 1$, then include the pile of size 2^i. Otherwise, exclude it. These piles of sizes 2^i have nim-sum exactly S. Thus, the entire NIM has nim-sum $S \oplus S = 0$, and it is a $\mathcal{P}$ position.

We have shown that any incomplete string corresponds to exactly one $\mathcal{P}$ position in NIM. Then, we must show each $\mathcal{P}$ position in NIM corresponds to exactly one incomplete string. But this is easy! Simply represent the NIM game as a string of length n and omit all the digits ζ_{2^k}.

Thus, since there are 2^{n-p} incomplete strings, there are exactly 2^{n-p} NIM $\mathcal{P}$ positions where all piles are less than n. Moreover, note that every NIM $\mathcal{P}$ position with all piles less than n corresponds to a single, unique TURNING TURTLES $\mathcal{P}$ position (with the largest i such that $\zeta_i = 1$ satisfying $i \leq n$). Then, each of these $\mathcal{P}$ positions corresponds to exactly one unique Hamming code word! As a result, there are exactly 2^{n-p} such Hamming code words, and $|C| = 2^{n-p}$, so we are done.

■

Remark 4.7. This actually makes the process of encoding data, or the process of mapping binary data to code words, remarkably easy for Hamming Codes. For $n - p$ binary bits of data ($n = 2^p - 1$), interpret it as an incomplete string $(\zeta_n \cdots \zeta_{10}\zeta_9\zeta_7\zeta_6\zeta_5\zeta_3)$. Then, one can follow the proof of Theorem 4.6 to construct the corresponding code word. After the receiver receives this code word, they can use the proof to extract the original $n - p$ bits from the code word.

Other constructions of the Hamming Code essentially follow this same method. The Hamming Code is remarkable in that it is rather easy to encode binary data into code words. This is not often true for other codes.

The Hamming Codes make up an infinite family of nontrivial perfect codes. However, there exists yet another nontrivial perfect code: the Golay code.

Definition 4.8. The binary *Golay Code* is the lexicode with $n = 23$ and $d = 7$.

The Golay Code can correct up to 3 errors. However, there is something special about sum $S = \sum_{i=0}^{m} \binom{n}{i}$. For the Golay Code, $S = 2^{11}$, which divides 2^{23}. This suggests, though does not prove, that the Golay Code is perfect.

Theorem 4.9. *The binary Golay Code is perfect.*

The proof for this requires finding $|C|$, which can either be done by computation or through techniques in coding theory, which is beyond the scope of this essay.

Moreover, it is true that the Golay Code and the Hamming Codes are the only nontrivial perfect binary codes. The proof is of this is also beyond the scope of this essay.

5. The Mock Turtle Theorem

Theorem 5.1 (The Mock Turtle Theorem)**.** *Consider the game of* Turning Turtles *with $k = 2m$, played with a line of coins. For every $\mathcal{P}$ position of this game with $k = 2m$, add a coin to the very right of all the coins, the "Mock Turtle". Flip it so that the number of heads-up coins is even. Call these new positions "good positions", and call all other positions "bad positions".*

The good positions are exactly the $\mathcal{P}$ positions of the game of Turning Turtles*, but with $k = 2m + 1$.*

Remark 5.2. Recall that in Turning Turtles, if a move turns any number of coins, it must turn the leftmost coin from face-up to face-down.

(In other texts, it is sometimes the rightmost coin, with the Mock Turtle being added to the very left, but we use the opposite for the purposes of this essay.)

Proof. Let $\mathscr{P}$ be the set of all good positions, and let $\mathscr{N}$ be the set of all bad positions. Note that all $\mathscr{P}$ positions have an even amount of heads-up coins, and that they all correspond to a $\mathcal{P}$ position in the game with $k = 2m$. We will show that $\mathscr{P} = \mathcal{P}$ and $\mathscr{N} = \mathcal{N}$ for Turning Turtles with $k = 2m + 1$.

First, we must show all $\mathscr{P}$ positions must move to $\mathscr{N}$ positions. Any move consists of turning $2m+1$ coins. However, note that $\mathscr{P}$ positions always have an even number of coins; this necessarily means that any move from one $\mathscr{P}$ position to another must flip an even number of coins.

This means that no move with $2m + 1$ flips can move from one $\mathscr{P}$ position to another. However, no move with $2m$ or fewer flips can do so either! Consider the $\mathcal{P}$ positions in the game with $k = 2m$ corresponding to the $\mathscr{P}$ positions. By definition, you cannot move from a such $\mathcal{P}$ to another by flipping $2m$ or fewer coins. Thus, it follows that you cannot move from any $\mathscr{P}$ position to another by flipping $2m$ or fewer coins as well!

Now, we must show all $\mathscr{N}$ positions can move to a $\mathscr{P}$ position. Let $G \in \mathscr{N}$, and Consider the position H that is obtained by removing the rightmost coin of G. There are only two reasons G could be a $\mathscr{N}$ position:

(1) either H is an $\mathcal{N}$ position in the game with $k = 2m$
(2) or the total number of heads in G is odd

If both were false, then it would be an $\mathscr{P}$ position by definition.

Now, if the first reason is true or if both reasons are true, then a move to a $\mathscr{P}$ position can be constructed as follows:

First, find $H' \in \mathcal{P}$ that H can move to in the game with $k = 2m$. By definition, H' must exist, as $H \in \mathcal{N}$. The move from H to H' takes at most $2m$ flips.

Then, define G' to be the position obtained by adding a coin to the very right of all coins in H', flipped so that the total number of coins is even.

Notice that G' is an $\mathscr{P}$ position, and that any move from G to G' takes at most $2m + 1$ flips. Thus, this case is done.

Then, if the first reason is false but the second reason is true, then H is a $\mathcal{P}$ position in the game with $k = 2m$. Flip over the leftmost coin in H to make it H', an $\mathcal{N}$ position. By definition, there must exist a move from H' to H'', where H'' is a $\mathcal{P}$ position. This move

flips over at most $2m$ coins. Now, we have shown one can flip $2m+1$ coins to get from H to H'', which is a distinct $\mathcal{P}$ position.

Now, obtain G' by taking H'' and adding an extra coin to the very right so that the total number of coins is even. Clearly, one can move from G to G' by flipping at most $2m+2$ coins.

However, notice that G has an odd number of heads, but G' has an even number of heads. Thus, it is impossible to get from G to G' in an even number of flips. It follows that you can get from G to G' in at most $2m+1$ coins, and we are done. ■

Remark 5.3. The Mock Turtle Theorem establishes a general bijection between the $\mathcal{P}$ positions with $k = 2m$ and the $\mathcal{P}$ positions with $k = 2m+1$. However, given n, the same proof can also be used to establish a bijection between

(1) the $\mathcal{P}$ positions with $k = 2m$ and with the largest i such that $\zeta_i = 1$ satisfying $i \leq n$
(2) and the $\mathcal{P}$ positions with $k = 2m+1$ and with the largest i such that $\zeta_i = 1$ satisfying $i \leq n+1$

Corollary 5.4. *Given the $\mathcal{P}$ positions of* Turning Turtles *with $k = 2m+1$, one can remove the rightmost coin to obtain all $\mathcal{P}$ positions of* Turning Turtles *with $k = 2m$.*

This is essentially the same thing as Theorem 5.1.

Notice that, surprisingly, the Mock Turtle Theorem has applications in constructing linear codes.

Corollary 5.5. *Consider a lexicode with length n and distance $d = 2m+1$. For each code word, add a digit to the end so that the number of* 1*s is even. We call this digit the "check digit". This new set of code words is simply the lexicode with length $n+1$ and distance $d = 2m+2$.*

This follows directly from previous theorems.

Definition 5.6. Let the *Extended Hamming Code* of length $n = 2^p$ be the code obtained by adding a check digit to the Hamming Code of length $n = 2^p - 1$.

The Extended Hamming Codes have $d = 4$, so, like before, they can correct 1-digit errors but can now also detect 2-digit errors.

Definition 5.7. Let the *Extended Golay Code* of length $n = 24$ be the code obtained by adding a check digit to the Golay Code.

The Extended Golay Code has $d = 8$ and can not only correct 3-digit errors, but can also detect 4-digit errors. It is often used in practical applications rather than the regular Golay Code.

6. Nonbinary Lexicodes

So far, we have only discussed binary codes. Here, we will quickly explore the basics of more general, nonbinary codes.

Consider the general game of Turning Objects. (This is sometimes simply named as Turning Turtles, but to avoid confusion, we give it a new name.)

Turning Objects is like Turning Turtles, except now, each object has B sides, numbered from 0 to $B-1$. On any turn, you can turn over k objects subject to the restriction that the leftmost object you turn must turn to a lesser-numbered side.

Now, we can extend lexicodes into an arbitrary base B. Let a base-B code word be code words with digits between (inclusive) 0 and $B-1$, and let a base-B code be a code with base-B code words.

Definition 6.1. A more general lexicode of length n, distance d, and base B is the base-B code that is generated by starting with an empty set of code words $S = \emptyset$, then greedily adding the lexicographically least length-n string that differs from all code words already in S by a hamming distance of at least d.

Many of the aforementioned theorems have analogues in Turning Objects. I state them without proof, as their proofs are almost identically the same as before.

Theorem 6.2. *The base-B lexicode of length n and distance d is exactly the set of all $\mathcal{P}$ positions of* Turning Turtles *with $k = d-1$ and B sides. These $\mathcal{P}$ positions must also satisfy the condition that the largest i such that $\zeta_i \neq 0$ must have $i \leq n$.*

Define perfect codes using the same definition as before.

Proposition 6.3. *A base-B lexicode C of length $n = 2m+1$ is perfect if and only if*

$$|C| \cdot \sum_{i=0}^{m} \binom{n}{i} (B-1)^i = B^n$$

References

[1] John H. Conway AND N. J. A. Sloane. *Lexicographic Codes: Error-Correcting Codes from Game Theory* http://neilsloane.com/doc/Me122.pdf

[2] Elwyn R. Berlekamp, John H. Conway, Richard K. Guy. *Winning Ways for Your Mathematical Plays, Volume 3, Volume 3*

[3] Aviezri S. Fraenkel. *Error-Correcting Codes Derived from Combinatorial Games* https://arxiv.org/pdf/math/9504211.pdf

[4] Terr, David. *Perfect Code.* From MathWorld–A Wolfram Web Resource, created by Eric W. Weisstein. https://mathworld.wolfram.com/PerfectCode.html