ERROR-CORRECTING CODES WHEN ERROR PROBABILITY VARIES

SQUID TAMAR-MATTIS

Abstract. In this paper we discuss the basics of error-correcting codes and why they are useful. We also generalize some of the frequently-used theorems relating to error-correcting codes so they work in cases where the probability of error may not be the same for every bit. We then briefly explore the idea of error-correcting codes in infinite channels with varying error probability.

Contents

1. Introduction

There are many situations where people want to send information over an unreliable channel. This can be a problem when the message one wants to send is too important for the risk of error created by the channel. In these cases it is useful to encode the message in an error-detecting or error-correcting code. These are not ways of obscuring information, as in the popular conception of a code, but instead these codes add redundant information to a message so that the receiver will probably be able to determine whether parts of the transmission have been modified, and in the case of an error-correcting code, will also be able to restore the original message. Error-detecting codes are generally simpler to construct and require fewer bits of redundant information, but they are less versatile as they are mostly only useful in contexts where the receiver can ask the sender to resend a corrupted message. In both code types, there is a three-directional trade-off between the amount of information (i.e. number of distinct messages) one can send, the amount of redundant information that must be added, and the number of errors that the receiver of the message will be able to detect or correct. Much of coding theory is dedicated to finding codes that maximize the first and third property while minimizing the second property.

2. Definitions and basic examples

Definition 2.1. An **alphabet** is a set of symbols. The codes discussed in this paper will all use the binary alphabet, $\{0, 1\}$.

Definition 2.2. If $n \in \mathbb{N} \cup \{0\}$ and Σ is an alphabet, a **string** of length n over Σ is a function $s : [n] \to \Sigma$, where $[n] = \{1, 2, \ldots, n\}$ (which means [0] is the empty set). Strings are usually denoted by writing the symbols from the alphabet next to each other in order, but they can also be written as vectors. Σ^n denotes the set of strings of length n over Σ.

Definition 2.3. A **symbol replacement error** is an error in transmission of a string where one symbol in the string is replaced with another symbol from the same alphabet.

A string may be subject to multiple symbol replacement errors when it is transmitted. Note that while other types of errors, like symbol insertion and deletion errors, do exist, we will not deal with them in this paper, so references to "errors" can be assumed to refer to replacement errors.

Definition 2.4. A **code** of length n over Σ is a set $C \subset \Sigma^n$.

Most of the time, the messages we want to send are strings of length k over Σ, where $k \leq n$. In order to turn these into elements of a code, we need an encoding function $f : \Sigma^k \to C$, which must be injective. Usually the encoding function is defined so that for every $m \in \Sigma^k$, the first k symbols in $f(m)$ are just a copy of m. This makes it easy to decode the message on sight.

Example 2.5. The doubling code for messages of length k is $D_k = \{xx : x \in \Sigma^k\}$, where xx denotes the concatenation of x with itself. Note that this means $D_k \subset \Sigma^{2k}$.

Doubling codes are a simple, intuitive example of a class of error-detecting codes. They are also quite powerful, as they can detect any number of errors as long as a symbol in the first half of the encoded string and its duplicate in the second half do not get changed to the same symbol. However, doubling the amount of data you send is a large increase, and this code cannot correct errors because if there is a mismatch between the two halves, the receiver has no way of knowing which half is right and which one contains an error. A tripling (or larger) code can correct errors, by looking at which symbol has the majority, but that uses even more data.

In order to find better codes, it helps to have a way to measure how different two strings are.

Definition 2.6. If x and y are strings in Σ^n, where $x = x_1x_2 \ldots x_n$ and $y = y_1y_2 \ldots y_n$, then the **Hamming distance** between x and y is defined as

$$\Delta(x, y) = |\{i : x_i \neq y_i\}|.$$

Trivially, this is a metric on Σ^n. Additionally, it has a useful property:

Proposition 2.7. *Let $a, b, c \in \{0, 1\}^n$ and suppose there exists some $p \in [0, \frac{1}{2}]$ such that when we transmit an element of $\{0, 1\}^n$, each bit has probability p of being flipped, where we determine whether each bit flips independently. For $x, y \in \{0, 1\}^n$, let $P(x \to y)$ denote the probability that the receiver of a transmission receives y, given that they were sent x. If $\Delta(a, b) < \Delta(a, c)$, then $P(a \to b) \geq P(a \to c)$. This inequality is strict if $p \in (0, \frac{1}{2})$.*

Proof. We observe that $P(x \to y)$ is the probability that all $\Delta(x, y)$ bits which are different in the two strings are flipped multiplied by the probability that all the other bits don't flip, which is $p^{\Delta(x,y)}(1-p)^{n-\Delta(x,y)}$. Thus the probability expression is always the product of n terms which are either p or $1-p$. Since $1-p \geq p$, this product only grows when more of those terms are $1-p$ which happens when $\Delta(x, y)$ is smaller. Thus $P(a \to b) \geq P(a \to c)$.

If $p \in (0, \frac{1}{2})$, then there is no 0 in the product and $p > 1-p$, so the inequality is strict. □

We often refer to a code C over an alphabet Σ as an (n, k, d) code, where $C \subset \Sigma^n$, k is the number of symbols in any message encoded by C, and d is the minimum Hamming distance between any two distinct elements of C. We also sometimes do not include the distance and just refer to it as an (n, k) code.

The elements of Σ do not need to be arbitrary. Since Σ can be any set, it can have its own algebraic structure. In particular, it can be a field, in which case Σ^n is a vector space. This gives rise to a useful type of code:

Definition 2.8. If Σ is a field, then $C \subset \Sigma^n$ is a **linear code** if C is a subspace of Σ^n.

The motivation for using linear codes may not be immediately obvious, but they turn out to have useful properties that make them easy to work with, especially one particular class of linear codes.

3. Hamming Codes

Definition 3.1. Let $r \in \mathbb{N}$, and let H_r be the $r \times (2^r - 1)$ matrix over $\mathbb{F}_2$ where the ith column of H_r is the binary representation of i (where the top entry is the 1s place, the second entry is the 2s place, the third entry is the 4s place, and so on). The r**th Hamming code**, C_r, is the null space of H_r.

Since the columns H_r include the standard basis vectors for $\mathbb{F}_2^r$, it is clear that $\text{rank}(H_r) = r$. Then by the rank-nullity theorem, $\text{nul}(H_r) = 2^r - r - 1$. Thus our choice of an element of C_r contains $2^r - r - 1$ bits of information. In fact, for any choice of the first $2^r - r - 1$ entries of a vector $\mathbf{x} \in \mathbb{F}_2^{2^r-1}$, there is a unique choice of the last r entries so that $\mathbf{x} \in C_r$. We therefore generally encode a $2^r - r - 1$-bit message in C_r by converting the message to the unique element of C_r where the message is the first $2^r - r - 1$ bits. This allows us to decode a transmission easily by simply deleting the last r entries.

Proposition 3.2. *Every Hamming code has minimum distance 3.*

Proof. Let $r \in \mathbb{N}$ and let $\mathbf{x}, \mathbf{y} \in C_r$ be such that $\mathbf{y} \neq \mathbf{x}$. Then $\mathbf{x} + \mathbf{y} \in C_r$ because C_r is linear. We know that $\mathbf{x} + \mathbf{y}$ has ones exactly in the entries where $\mathbf{x}$ and $\mathbf{y}$ differ, so if $\Delta(\mathbf{x}, \mathbf{y}) = 1$, then $\mathbf{x}+\mathbf{y}$ has a 1 in only one entry and 0 everywhere else, which means that $H_r(\mathbf{x} + \mathbf{y})$ is one of the columns of H_r, but H_r doesn't have a column of just 0, so then $\mathbf{x} + \mathbf{y}$ can't be in $\ker(H_r)$.

Similarly, if $\Delta(\mathbf{x}, \mathbf{y}) = 2$, then $\mathbf{x}+\mathbf{y}$ has ones in exactly two entries, thus requiring the sum of the corresponding columns of H_r to be 0, but H_r doesn't have any two identical columns, so this is impossible. Thus $\Delta(\mathbf{x}, \mathbf{y}) \geq 3$. This bound is tight for all Hamming codes because every Hamming code contains the zero vector and a vector where the first three entries are 1 and the rest are 0. □

This is a useful property because it means that an element of $\mathbb{F}_2^{2^r-1}$ cannot be within Hamming distance 1 of more than one element of C_r, because Hamming distance obeys the triangle inequality. Thus it is possible to correct any single bit error that occurs when an encoded message is transmitted.

Proposition 3.3. *Every element of $\mathbb{F}_2^{2^r-1}$ is either an element of C_r or is Hamming distance 1 from an element of C_r.*

Proof. As previously shown, $\dim(C_r) = 2^r - r - 1$, so C_r has 2^{2^r-r-1} elements. For each $\mathbf{x} \in C_r$, we can find $2^r - 1$ vectors $\mathbf{y} \in \mathbb{F}_2^{2^r-1}$ such that $\Delta(\mathbf{x}, \mathbf{y}) = 1$ by adding each of the standard basis vectors to $\mathbf{x}$. Because the minimum distance of C_r is 3, no vector in $\mathbb{F}_2^{2^r-1}$ that is Hamming distance 1 from an element of C_r can be an element of C_r or Hamming distance 1 from a different element of C_r, so all of these elements are distinct. Therefore the total number of vectors that are Hamming distance 1 from an element of C_r is $2^{2^r-r-1}(2^r - 1) = 2^{2^r-1} - 2^{2^r-r-1}$. If we add to this the number of elements of C_r, we get $2^{2^r-1} - 2^{2^r-r-1} + 2^{2^r-r-1} = 2^{2^r-1}$, which is the number of elements of $\mathbb{F}_2^{2^r-1}$. □

This fact allows us to prove a really interesting property of Hamming codes.

Proposition 3.4. *Let $\mathbf{y} \in \mathbb{F}_2^{2^r-1}$. If $\mathbf{y} \in C_r$, then $H_r\mathbf{y} = \mathbf{0}$. Otherwise, $H_r\mathbf{y}$, when read as a binary number, gives the index of the entry in $\mathbf{y}$ that must be changed to yield an element of C_r.*

Proof. The first part is trivial. Since $C_r = \ker(H_r)$, if $\mathbf{y} \in C_r$, then $H_r\mathbf{y} = \mathbf{0}$.

Suppose $\mathbf{y} \notin C_r$. Then by proposition 3.3, $\mathbf{y} = \mathbf{x} + \mathbf{e}_i$, where $\mathbf{x}$ is some element of C_r, $i \in [2^r - 1]$, and $\mathbf{e}_i$ is the ith standard basis vector. Then $H_r\mathbf{y} = H_r\mathbf{x} + H_r\mathbf{e}_i = \mathbf{0} + H_r\mathbf{e}_i$, which is the ith column of H_r. By the definition of H_r, this column forms the binary representation of i. □

This gives us a simple, efficient algorithm for correcting single-bit errors in Hamming codes.

Theorem 3.5 (Hamming bound)**.** *Let C be a binary (n, k, d) code. Then*

$$|C| \leq \frac{2^n}{\sum_{i=0}^{\lfloor \frac{d-1}{2} \rfloor} \binom{n}{i}}.$$

Proof. Let $\mathbf{x}, \mathbf{y} \in C$ be distinct. If $\mathbf{v} \in \mathbb{F}_2^n$, then $\Delta(\mathbf{v}, \mathbf{x})$ and $\Delta(\mathbf{v}, \mathbf{y})$ cannot both be less than $\frac{d}{2}$, because then $\Delta(\mathbf{x}, \mathbf{y}) < d$. Thus the sets $B_{\frac{d}{2}}(\mathbf{x}) = \{\mathbf{v} \in \mathbb{F}_2^n : \Delta(\mathbf{v}, \mathbf{x}) < \frac{d}{2}\}$ are disjoint for $\mathbf{x} \in C$. Since $\mathbb{F}_2^n$ has 2^n elements, this means

$$\sum_{\mathbf{x} \in C} |B_{\frac{d}{2}}(\mathbf{x})| \leq 2^n.$$

To find $|B_{\frac{d}{2}}(\mathbf{x})|$, we just count the number of points that are Hamming distance exactly i from $\mathbf{x}$ for $0 \leq i < \frac{d}{2}$. Each such point can be uniquely constructed by choosing i entries in $\mathbf{x}$ and changing them, so there are $\binom{n}{i}$ of them. Thus $|B_{\frac{d}{2}}(\mathbf{x})| = \sum_{i=0}^{\lfloor \frac{d-1}{2} \rfloor} \binom{n}{i}$. Since this value does not depend on $\mathbf{x}$, adding these together for each $\mathbf{x} \in C$ is equivalent to multiplication by $|C|$, and thus

$$|C| \sum_{i=0}^{\lfloor \frac{d-1}{2} \rfloor} \binom{n}{i} \leq 2^n.$$

□

Definition 3.6. Codes for which the Hamming bound is tight are called **perfect codes**.

Perfect codes are the "best" codes in the sense that their codewords are optimally distributed in $\mathbb{F}_2^n$, but they are not necessarily the most useful codes in every situation. Furthermore, a theorem proven by Tietavainen and Van Lindt shows that the only perfect binary codes are trivial codes (i.e. 1-element codes, 2-element codes, and the code containing all of $\mathbb{F}_2^n$), Hamming codes, and the Golay $(23, 12, 7)$ code.

We can use a similar technique we used to prove the Hamming bound to show the following:

Theorem 3.7. *There exists a binary (n, k, d) code C such that $|C| \geq \frac{2^n}{\sum_{i=0}^{d-1} \binom{n}{i}}$.*

Proof. Let C be a maximal (n, k, d) code. Then every $\mathbf{v} \in \mathbb{F}_2^n$ must be in $B_d(\mathbf{x})$ for some $\mathbf{x} \in C$ because if there was some $\mathbf{v}$ that wasn't, we would be able to add $\mathbf{v}$ to C and it would still be an (n, k, d) code. As above, we know that for each $\mathbf{x}$, $|B_d(\mathbf{x})| = \sum_{i=0}^{d-1} \binom{n}{i}$. Therefore $|C| \sum_{i=0}^{d-1} \binom{n}{i} \geq 2^n$. □

Theorem 3.8. *If C is a binary (n, k, d) code, then $|C| \leq 2^{n-d+1}$.*

Proof. If $\mathbf{x}, \mathbf{y}$ are distinct elements of C, then their last $n - d + 1$ entries cannot all be the same, or else $\Delta(\mathbf{x}, \mathbf{y})$ would be at most $d - 1$. Therefore each element of C has a distinct combination of those $n - d + 1$ entries, and there are 2^{n-d+1} such combinations. □

4. Codes in Variable Error Probability Spaces

So far, we've only dealt with situations where there is a probability p of change in transit that is the same for every bit. In this case, if we fix n and k, the best (n, k) code, and its corresponding correction function, will be the same regardless of what p happens to be. But we can also consider cases where for each $i \in [n]$ there is some probability p_i that the ith bit flips during transmission. In these situations, the best code and correction function do depend on the values of the p_i.

Definition 4.1. A **variable probability channel** is a triple $(\Sigma, n, (p_1, \ldots, p_n))$, where Σ is an alphabet, n is the number of elements of Σ we can send over the channel, and $(p_1, \ldots, p_n)$ is an n-tuple of error probabilities where the ith character we send has probability p_i of being modified in transmission.

We're generally going to be discussing cases where $\Sigma = \mathbb{F}_2$ and $0 < p_i < \frac{1}{2}$ for all $i \in [n]$.

Example 4.2. If we want to create a $(1, 3)$ code in a constant probability space, the best option is to encode by sending $1 \mapsto 111$ and $0 \mapsto 000$, and decode by sending a string to the bit that is in the majority. If the bit error probability is p, then the probability of receiving a correct 1-bit message is $1 - p$, but the probability of being able to correctly decode a message that is encoded this way is $(1 - p)^3 + 3p(1 - p)^2$, which is greater than $1 - p$ for $p \in (0, \frac{1}{2})$.

Consider the channel $(\mathbb{F}_2, 3, (.01, .2, .2))$. If we use the same code to send the message 0, then we send 000, which becomes either 011 or 111 with probability

$0.2^2 = .04$. In these cases, our error-correction function gives us 111, which decodes to 1, which is wrong. Therefore our probability of the message being understood correctly is at most .96, whereas if we had just used the first bit in the channel and ignored the other two, it would have been .99.

In this space, the string 000 is in some sense closer to 011 than it is to 100, even though it shares more digits with the latter. It is therefore helpful for us to have a metric that allows us to quantify this closeness.

Definition 4.3. If $\mathbf{x}$ and $\mathbf{y}$ are vectors in a variable error probability space, the **probabilistic distance** between them is defined as

$$d(\mathbf{x}, \mathbf{y}) = \sum_{\{i : x_i \neq y_i\}} \log_2 \frac{1 - p_i}{p_i}.$$

It can be easily shown that probabilistic distance is a metric as long as $0 < p_i < \frac{1}{2}$ for all i. If we allow $p_i = \frac{1}{2}$, it is a pseudometric, and if we allow $p_i = 0$, it is an extended metric.

Probabilistic distance does not generalize Hamming distance, but it does do so up to multiplication by a scalar. If $\mathbf{x}, \mathbf{y}$ are in a constant error probability channel with error probability p, then

$$d(\mathbf{x}, \mathbf{y}) = \Delta(\mathbf{x}, \mathbf{y}) \log_2 \frac{1 - p}{p}.$$

Proposition 4.4. *Let $\mathscr{P}$ be a variable error probability space and define $c(P)$ to be the probability that a string in $\mathscr{P}$ is unchanged in transmission. Then if $\mathbf{x}, \mathbf{y} \in \mathscr{P}$,*

$$P(\mathbf{x} \to \mathbf{y}) = c(\mathscr{P}) 2^{-d(\mathbf{x}, \mathbf{y})}.$$

Proof. Obviously

$$c(\mathscr{P}) = \prod_{i=1}^{n} (1 - p_i)$$

and

$$P(\mathbf{x} \to \mathbf{y}) = \prod_{\{i : x_i = y_i\}} (1 - p_i) \prod_{\{i : x_i \neq y_i\}} p_i,$$

so

$$\begin{aligned} P(\mathbf{x} \to \mathbf{y}) &= c(\mathscr{P}) \prod_{\{i : x_i \neq y_i\}} \frac{p_i}{1 - p_i} \\ &= c(\mathscr{P}) 2^{\log_2 \prod_{\{i : x_i \neq y_i\}} \frac{p_i}{1 - p_i}} \\ &= c(\mathscr{P}) 2^{-\log_2 \prod_{\{i : x_i \neq y_i\}} \frac{1 - p_i}{p_i}} \\ &= c(\mathscr{P}) 2^{-\sum_{\{i : x_i \neq y_i\}} \log_2 \frac{1 - p_i}{p_i}} \\ &= c(\mathscr{P}) 2^{-d(\mathbf{x}, \mathbf{y})}. \end{aligned}$$

□

Thus a greater distance between two strings corresponds to a lower probability that one will become the other in transmission.

Theorem 4.5 (Generalized Hamming Bound)**.** *Let C be a code in $(\mathbb{F}_2, n, (p_1, \dots, p_n))$ with minimum probabilistic distance d. Let $\mathscr{A}$ be the family consisting of all $I \subset [n]$ such that $\sum_{i\in I} \log_2 \frac{1-p_i}{p_i} < \frac{d}{2}$. Then*

$$|C| \leq \frac{2^n}{|\mathscr{A}|}.$$

Proof. In order to have minimum distance d, the open balls of radius $\frac{d}{2}$ around the points in C must be disjoint. If we fix some $\mathbf{x} \in C$, the elements of $B_{\frac{d}{2}}(\mathbf{x})$ are uniquely defined by taking some $I \in \mathscr{A}$ and flipping the entries of $\mathbf{x}$ that correspond to the elements of I. Thus $|B_{\frac{d}{2}}(\mathbf{x})| = |\mathscr{A}|$. Then since $\mathbb{F}_2^n$ only has 2^n elements, $|C||\mathscr{A}| \leq 2^n$. □

Theorem 4.6. *Let $\mathscr{B}$ be the family consisting of all $I \subset [n]$ such that $\sum_{i\in I} \log_2 \frac{1-p_i}{p_i} < d$. Then there exists some code C with minimum distance d such that*

$$|C| \geq \frac{2^n}{|\mathscr{B}|}.$$

Proof. As in the previous proof, we know $|B_d(\mathbf{x})| = |\mathscr{B}|$. Let C be a code and suppose that balls of radius d around the elements of C don't cover $\mathbb{F}_2^n$. Then we can take one of the points that isn't in any of the balls and add it to C without making the minimum distance less than d. We can repeat this until the balls cover $\mathbb{F}_2^n$, at which point $|C||\mathscr{B}| \geq 2^n$. □

Theorem 4.7. *If C has minimum distance d, then $|C| \leq 2^t$ where $t = \min\{|I| : I \subset [n], \sum_{i\in[n]\setminus I} \log_2 \frac{1-p_i}{p_i} < d\}$.*

Proof. Let $\mathbf{x}, \mathbf{y} \in C$ and let $I \subset [n]$ be one of the subsets that give us t. Then $\mathbf{x}$ and $\mathbf{y}$ cannot have the same character in every entry indexed by I, or else we would have $d(\mathbf{x}, \mathbf{y}) \leq \sum_{i\in[n]\setminus I} \log_2 \frac{1-p_i}{p_i} < d$. Thus the elements of C all have different characters in the entries indexed by I, which there are t of, so $|C| \leq 2^t$. □

5. Infinite Channels

In some contrived situations, we can send as many bits as we want, but the probability of error changes over time. In particular, when the probability of error increases in such a way that it asymptotically approaches $\frac{1}{2}$, it is useful to know whether we can send arbitrarily long messages with a high probability of them being understandable.

Definition 5.1. An **infinite variable error probability channel** is a pair $(\Sigma, (p_1, p_2, \dots))$ where Σ is an alphabet and the p_i are an infinite sequence of error probabilities.

Definition 5.2. For an alphabet Σ, let $\mathscr{C}_\Sigma(n, k)$ be the set of (n, k) codes over Σ. For an infinite channel H and a code $C \in \mathscr{C}_\Sigma(n, k)$, let $q_H(C)$ be the probability that an element of C sent over H (using the first n slots) can be successfully corrected (using the best error-correction function for C and H). Let

$$Q_H(n, k) = \max_{C\in\mathscr{C}_\Sigma(n,k)} q_H(C).$$

In other words, $Q_H(n, k)$ is our probability of success when using the best (n, k) code we can on the first n slots in H.

Definition 5.3. We say H is **infinitely useful** if for all $k \in \mathbb{N}$,

$$\lim_{n\to\infty} Q_H(n,k) = 1.$$

Which is to say that a message of any length can be encoded in H with a probability success that can get as close to 1 as we want it to (without actually being 1).

Theorem 5.4. *If $H = (\mathbb{F}_2, (p_1, \dots))$ such that $\lim_{i\to\infty} \log_2 \frac{1-p_i}{p_i} \neq 0$ or the limit does not exist, then H is infinitely useful.*

Proof. Let H be such a channel. Then there exists some $L > 0$ such that there exists an infinite subsequence p_{i_j} such that for all j, $\log_2 \frac{1-p_{i_j}}{p_{i_j}} > L$. Then $L = \log_2 \frac{1-x}{x}$ for some $x < \frac{1}{2}$ and for all j, $p_{i_j} < x$. We can then encode our message by repeating it many times in the i_j entries, and by the law of large numbers, each bit's probability of being right in the majority of repeats approaches 1. □

Lemma 5.5. *Let H be a channel such that $\lim_{n\to\infty} Q_H(n,1) = 1$ and there is no i such that $p_i = 0$. Let $m \in \mathbb{N}$ and let H' be H, but with the error probability sequence replaced by $(p_m, p_{m+1}, p_{m+2}, \dots)$. Then $\lim_{n\to\infty} Q_{H'}(n,1) = 1$.*

Proof. It suffices to prove this for $m = 2$, as then all other values are obvious by induction.

First of all, note that when the message is 1 bit, the best code is always the one that consists of the string $\mathbf{0}$ of all 0 entries and the string $\mathbf{h}$ of all 1 entries (or we can use any other string and its complement, but this way is the least confusing), with a correction function that sends a received string to whichever of these it is closer to in probabilistic distance (or to $\mathbf{h}$ if it is the same distance from both).

Let $\epsilon > 0$. Let $\epsilon' = p_1\epsilon$. Then there exists some N such that $Q_H(N,1) > 1 - \epsilon'$. Now say we're trying to transmit a bit over H. Without loss of generality, let's say we're trying to transmit 0. Then we send the string $\mathbf{0}$ and one of four things happens to it in transit. Let $\mathbf{x}$ denote the received string, let $\mathbf{x}'$ denote $\mathbf{x}$ without the first entry, and let $\mathbf{0}'$ and $\mathbf{h}'$ denote the strings of $N-1$ zeroes and $N-1$ ones respectively:

(1) The first bit is received correctly and $d(\mathbf{x}', \mathbf{0}') < d(\mathbf{x}', \mathbf{h}')$. In this case the message is received correctly.
(2) The first bit is received correctly but $d(\mathbf{x}', \mathbf{0}') \geq d(\mathbf{x}', \mathbf{h}')$. In this case, the message may or may not be received correctly, depending on p_1 and $d(\mathbf{x}', \mathbf{0}') - d(\mathbf{x}', \mathbf{h}')$.
(3) The first bit is flipped but $d(\mathbf{x}', \mathbf{0}') < d(\mathbf{x}', \mathbf{h}')$. In this case, the message may or may not be received correctly, depending on p_1 and $d(\mathbf{x}', \mathbf{0}') - d(\mathbf{x}', \mathbf{h}')$.
(4) The first bit is flipped and $d(\mathbf{x}', \mathbf{0}') \geq d(\mathbf{x}', \mathbf{h}')$. In this case, the message is received incorrectly.

Let $P(i)$ be the probability of case (i). We observe that $Q_{H'}(N-1,1) = P(1)+P(3)$ because (1) and (3) are exactly the cases where a bit encoded in bits $2, \dots, N$ would be received correctly, and $Q_H(N,1) \leq P(1) + P(2) + P(3)$. Then $1 - Q_H(N,1) \geq$

$P(4)$ and $1 - Q_{H'}(N-1,1) = P(2) + P(4)$. Therefore

$$\frac{1 - Q_{H'}(N-1,1)}{1 - Q_H(N,1)} \leq \frac{P(2)+P(4)}{P(4)} = \frac{1 - Q_{H'}(N-1,1)}{p_1(1 - Q_{H'}(N-1,1))} = \frac{1}{p_1}.$$

Since $Q_H(N,1) > 1-\epsilon'$, this means $Q_{H'}(N-1,1) > 1-\epsilon$. Thus $\lim_{n\to\infty} Q_{H'}(n,1) = 1$. □

Theorem 5.6. *If $\lim_{n\to\infty} Q_H(n,1) = 1$ and there is no i such that $p_i = 0$, then H is infinitely useful.*

Proof. Let H have the properties described, let $k \in \mathbb{N}$, and let $\epsilon > 0$ such that ϵ is very small. Then there exists some N_1 such that for all $n \geq N_1$, $Q_H(n,1) > \sqrt[k]{1-\epsilon}$. We can therefore encode the first bit of our message in the first N_1 bits of H. By lemma 5.5, when H starts from position $N_1 + 1$, so it still has the properties described, so we can repeat this process with some N_2, and so on for all k bits, at which point we have encoded each bit with success probability greater than $\sqrt[k]{1-\epsilon}$, so $Q_H(\sum_{i=1}^{k} N_i, k) > \sqrt[k]{1-\epsilon}^k = 1 - \epsilon$. Thus H is infinitely useful. □

Theorem 5.7. *If $\sum_{i=1}^{\infty} \log_2 \frac{1-p_i}{p_i}$ converges, then H is not infinitely useful.*

Proof. Let $P(n,d)$ denote the probability that a string on the first n entries of H and the string it becomes in transmission are at most distance d from each other. Then

$$Q_H(n,1) = P\left(n, \frac{1}{2}\sum_{i=1}^{n} \log_2 \frac{1-p_i}{p_i}\right).$$

.

Let $L = \sum_{i=1}^{\infty} \log_2 \frac{1-p_i}{p_i}$. Then there exists some N such that $\sum_{i=1}^{N} \log_2 \frac{1-p_i}{p_i} > \frac{L}{2}$. Let $n \geq N$. Then

$$\begin{aligned} Q_H(n,1) &= P\left(n, \frac{1}{2}\sum_{i=1}^{n} \log_2 \frac{1-p_i}{p_i}\right) \\ &\leq P\left(n, \frac{L}{2}\right) \\ &\leq P\left(N, \frac{L}{2}\right) \\ &< 1. \end{aligned}$$

Thus $\lim_{n\to\infty} Q_H(n,1) < 1$, so H is not infinitely useful. □

Thus we have statements about the usefulness of infinite channels where the distance sequences do not converge to zero and those where the distance series converges. It is left as an exercise to the reader to find what happens when the sequence converges to zero, but the series diverges.

Acknowledgments. I would like to thank my mentor Randy van Why, for providing valuable guidance and putting up with my unhelpful meandering explanations of concepts I understood poorly. I would also like to thank Peter May for running the Chicago REU, thereby giving me the opportunity to write this paper, and Robert Greene for telling me about coding theory in the first place.

References

[1] Robert Green. Alphabet Soup. http://math.uchicago.edu/ may/REU2016/REUPapers/Green.pdf.
[2] Vera Pless. Introduction to the Theory of Error-Correcting Codes (Second Edition). John Wiley & Sons, Inc.