

Basic Theory

Distance

Suppose that you knew that an English word was transmitted and you had received the word SHIP. If you suspected that some errors had occurred in transmission, it would be impossible to determine what word was really transmitted - it could have been SKIP, SHOP, STOP, THIS, actually any four letter word. The problem here is that English words are in a sense "too close" to each other. What gives a code its error correcting ability is the fact that the code words are "far apart". We shall make this distance idea more precise.

Assumptions

First of all we shall restrict our horizons and only consider block codes, so all codewords will have the same length. There are other types of codes, with variable length codewords, which are used in practice, but their underlying theory is quite different from that of the block codes.

Our second assumption is that the symbols used in our codewords will come from a *finite* alphabet Σ . Typically, Σ will just consist of the integers $\{0, 1, \dots, k-1\}$ when we want our alphabet to have size k , but there will be other alphabets used in our work. Note that unless otherwise specified, these numbers are only being used as symbols – they have no arithmetic properties.

Settings

A *code* with codewords of length n , made up from an alphabet Σ of size k , is then just a subset of $\Sigma^n = \Sigma \times \Sigma \times \dots \times \Sigma$, that is the set of n -tuples with entries from Σ . Since the actual alphabet is important (only its size) we will denote this “space” by

$$V(\mathbf{n},k) := \Sigma^n$$

The elements of $V(\mathbf{n},k)$ are called *words*.

In those situations where we wish to use algebraic properties of the alphabet, we modify the notation by replacing the parameter k by the name of the algebraic structure we are using. Thus,

$$V(\mathbf{n}, \mathbb{Z}_4)$$

indicates that the n -tuples are made up from the elements of \mathbb{Z}_4 and that we can add n -tuples componentwise using the operations of \mathbb{Z}_4 (namely, adding mod 4). [Technically, this space is known as a \mathbb{Z}_4 -*module* since the alphabet is a ring.]

Settings

The most important setting occurs when the alphabet is a finite field. To indicate this setting we will use the notation

$$V[n,q]$$

implying that the alphabet is the finite field with q elements ([as we shall see later, \$q\$ must be a prime or power of a prime](#)). In this case, $V[n,q]$ is a *vector space* (with scalars from the finite field).

Many of the codes we will encounter, especially those that have been useful in computer science, have the vector space setting $V[n,2]$. These are often called *binary codes* since the alphabet is the binary field consisting of only two elements. Codes in $V[n,3]$ are called *ternary codes*, and, in general, codes in $V[n,q]$ are called *q -ary codes*.

Hamming Distance

The *Hamming distance* between two words in $V(n,k)$ is the number of places in which they differ.

So, for example, the words $(0,0,1,1,1,0)$ and $(1,0,1,1,0,0)$ would have a Hamming distance of 2, since they differ only in the 1st and 5th positions. In $V(4,4)$, the words $(0,1,2,3)$ and $(1,1,2,2)$ also have distance 2.

This Hamming distance is a *metric* on $V(n,k)$, i.e., if $d(x,y)$ denotes the Hamming distance between words x and y , then d satisfies:

- 1) $d(x,x) = 0$
- 2) $d(x,y) = d(y,x)$, and
- 3) $d(x,y) + d(y,z) \geq d(x,z)$. (triangle inequality)

Hamming Distance

The first two of these properties are obvious, but the triangle inequality requires a little argument ([this is a homework problem](#)).

Since we will only deal with the Hamming distance (there are other metrics used in Coding Theory), we will generally omit the Hamming modifier and talk about the *distance* between words.

Minimum Distance

The *minimum distance* of a code C is the smallest distance between any pair of distinct codewords in C . It is the minimum distance of a code that measures a code's error correcting capabilities. If the minimum distance of a code C is $2e + 1$, then C is a $2e$ -error detecting code since $2e$ or fewer errors in a codeword will not get to another codeword and is an e -error correcting code, since if e or fewer errors are made in a codeword, the resulting word is closer to the original codeword than it is to any other codeword and so can be correctly decoded (*maximum-likelihood decoding*).

In the 5-repeat code of $V(5,4)$ (codewords: 00000, 11111, 22222, and 33333) the minimum distance is 5. The code detects 4 or fewer errors and corrects 2 or fewer errors as we have seen.

Weight of a Word

We always assume that 0 is one of the symbols in our alphabet.

The *weight* of a word is the number of non-zero components in the word. Alternatively, the weight is the distance of the word from the zero word.

In $V(6,6)$ the word $(0,1,3,0,1,5)$ has weight 4.

When we are working with an alphabet in which one can add and subtract then there is a relationship between distance and weight,

$$d(\mathbf{x}, \mathbf{y}) = \text{wt}(\mathbf{x} - \mathbf{y}),$$

since whenever a component of \mathbf{x} and \mathbf{y} differ, the corresponding component of $\mathbf{x} - \mathbf{y}$ will not be 0.

(n, M, d) – Codes

Let C be a code in $V(n,k)$. If C has M codewords and minimum distance d , we sometimes refer to it as an (n, M, d) -code.

For fixed n , the parameters M and d work against one another - the bigger M , the smaller d and vice versa. This is unfortunate since for practical reasons we desire a large number of codewords with high error correcting capability (large M and large d). The search for “good” codes always involves some compromise between these parameters.

Covering Radius

Since $V(n,k)$ has a metric defined on it, it makes sense to talk about spheres centered at a word with a given radius. Thus,

$$S_r(\mathbf{x}) = \{y \in V(n,k) \mid d(\mathbf{x},y) \leq r \}$$

is the *sphere* of radius r centered at \mathbf{x} .

The *covering radius* of a code C is the smallest radius s so that

$$V(n, k) \subseteq \bigcup_{x \in C} S_s(x)$$

i.e., every word of the space is contained in some (at least one) sphere of radius s centered at a codeword.

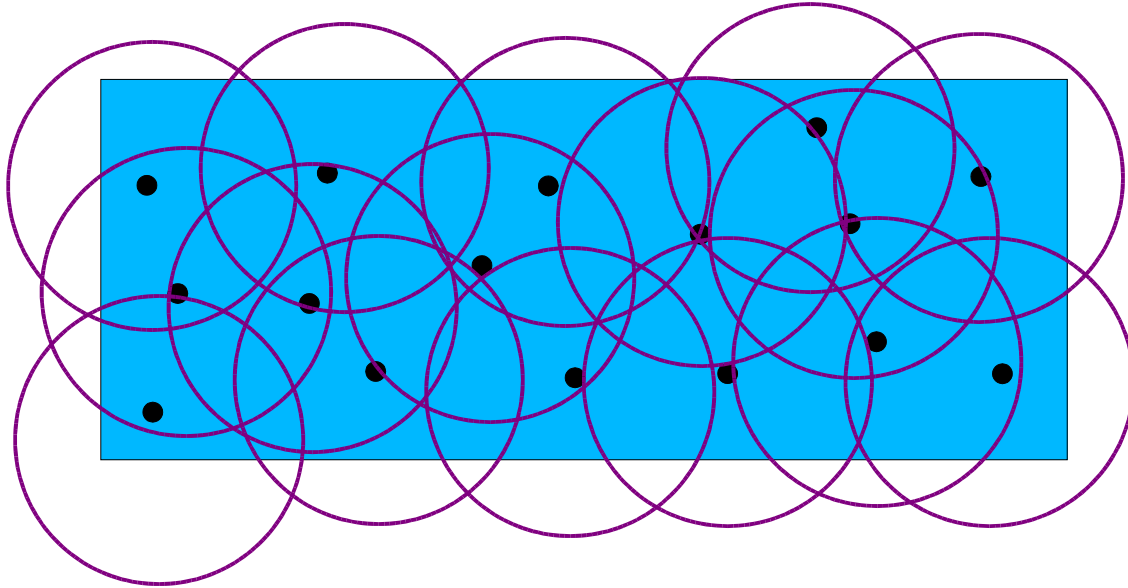
Packing Radius

The *packing radius* of a code C is the largest radius t so that the spheres of radius t centered at the code words are disjoint.

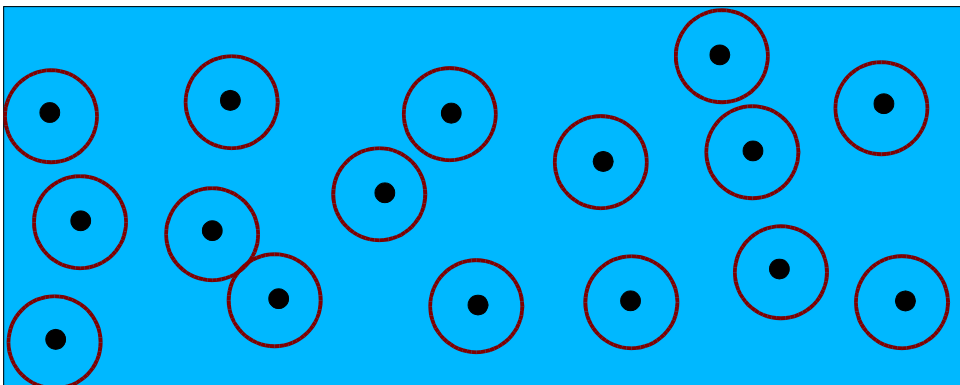
$$S_t(x) \cap S_t(y) = \emptyset \quad \forall x \neq y \in C$$

Clearly, $t \leq s$. When $t = s$, we say that C is a *perfect code*. While perfect codes are very efficient codes, they are very rare – most codes are not perfect.

Covering and Packing



Covering Radius



Packing Radius

Example

Consider the 5-repeat code in $V(5,3)$. There are just 3 codewords, 00000, 11111, and 22222. A word such as 01221 is at distance 4 from 00000, and distance 3 from both 11111 and 22222. **The distance of a word x from a code word is just $5 - (\# \text{ symbols in common with the codeword})$.** Since there are just 3 symbols and 5 positions, every word must have at least one repeated symbol, and so distance at most 3 from some codeword. Spheres of radius 3 around the codewords will therefore contain all of $V(5,3)$. This means that $s \leq 3$. The example of 01221 shows that if $s = 2$ this word would not be contained in any sphere, thus the **covering radius $s = 3$** . This same example shows that spheres of radius 3 are not disjoint, so $t < 3$. Two spheres of radius 2 must be disjoint, since a word in both would have 3 symbols in common with both codewords $\rightarrow \leftarrow$ So, the **packing radius $t = 2$** .

Sphere Packing Bound

We can count the number of words in a sphere of radius e in $V(n,q)$ and obtain:

$$|\mathcal{S}_e(\mathbf{x})| = \sum_{i=0}^e \binom{n}{i} (q-1)^i.$$

To count the number of words at distance i from the word \mathbf{x} , we first select which i positions will be different and then in each of these positions we select an element of the alphabet different from the one in that position in \mathbf{x} (there are $q-1$ choices per position).

If C is an (n,M,d) -code in $V(n,q)$ and the spheres of radius e centered at the codewords are disjoint, we obtain the sphere packing bound – since $V(n,q)$ contains q^n words:

$$M \sum_{i=0}^e \binom{n}{i} (q-1)^i \leq q^n$$

Sphere Packing Bound

The minimum distance d of a perfect code must be odd. If it were even, there would be words at an equal distance from two code words and spheres around those codewords could not be disjoint if they had to contain these words. So, $d = 2e + 1$ and it is easy to see that for a perfect code $t = s = e$. Furthermore,

Proposition 3 : *An (n, M, d) -code in $V(n, q)$ is perfect if and only if $d = 2e + 1$ and*

$$M \sum_{i=0}^e \binom{n}{i} (q-1)^i = q^n .$$

Pf: If C is perfect the spheres of radius e centered at codewords are disjoint ($e = t$) and all words are contained in them ($e = s$). On the other hand, if $d = 2e + 1$, we have $e \leq t$ and the counting equality shows that $s \leq e$. Thus, $s \leq e \leq t$, which implies $s = e = t$. \square

Equivalent Codes

The capabilities of a code are determined by the number of and the distances between the codewords. Thus, two codes should be considered equivalent if these statistics are the same.

There are two operations on the words of $V(n,k)$ which, while changing the words, do preserve the distances between any set of words. The first is **an arbitrary permutation of the coordinate positions applied to all words** of $V(n,k)$. This doesn't change distances since the distance is determined only by the number of positions in which the entries differ and not on where these occur. The second operation applies independently to any coordinate position. Since it is only whether or not the symbols in this position differ, and not what the symbols are that matter, **any arbitrary permutation of the alphabet can be applied in this coordinate position.**

Equivalent Codes

Formally, we say that two codes in $V(n,k)$ are *equivalent* if one can be obtained from the other by a series of the two operations:

- 1) arbitrary permutations of the coordinate positions of all words,
- 2) arbitrary permutations of the symbols independently applied to the symbols in any set of coordinate positions.

The following codes (the 3 words in a column) are equivalent in $V(4,4)$:

| | | |
|------|------|------|
| 0123 | 0213 | 0000 |
| 1120 | 1210 | 1002 |
| 0333 | 0333 | 0120 |

Note that in all cases the distance between the first two codewords is 2, as is the distance between the first and third, and 4 between the second and third.

Equivalent Codes

Note that as a consequence of the notion of equivalence, given a code C in $V(n,k)$, there will always exist an equivalent code to C which contains the zero word (in fact, an equivalent code to C which contains any word you may like).