6.897 Algorithmic Introduction to Coding Theory September 4, 2001

Lecture 1

Lecturer: Madhu Sudan *Scribe: Hanson Zhou*

In 1950, Richard Hamming wrote a paper dealing with the inevitable errors that occur when storing digital information on magnetic disk. To start with a simple case, we may consider what it would take to handle one bit errors when storing a sequence of bits. A straightforward approach would be to store each bit three times, so that any one bit that is erroneously flipped can be detected and corrected by majority decoding on its block of three. This is known as the repetition code, and is rather inefficient as the ratio of actual information bits to total bits stored is only $\frac{1}{3}$.

The Hamming encoding tries to do better with the following matrix:

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Given a sequence of bits, we chop it into 4 bit chunks. Let b be the vector representing one such chunk, then we encode b as the 7 bit bG, where all arithmetic is performed mod 2. Clearly the efficiency is a much better $\frac{4}{7}$, though we still need to show that this code can correct one bit errors.

Claim 1 *$\forall\ b_1 \neq b_2, b_1G$ and b_2G differ in ≥ 3 coordinates.*

First we present some definitions. We denote by Σ the alphabet, and the ambient space Σ^n represents the set of n letter words over the alphabet Σ.

Definition 2 *The Hamming distance $\Delta(x, y)$ between $x, y \in \Sigma^n$ is the number of coordinates i where $x_i \neq y_i$.*

The Hamming distance is a metric since it is easy to verify that:

$$\Delta(x, y) = \Delta(y, x)$$

$$\Delta(x, z) \leq \Delta(x, y) + \Delta(y, z)$$

$$\Delta(x, y) = 0 \Leftrightarrow x = y$$

In our case, consider the space of all possible encoded words $\{0, 1\}^7$. If we can prove our claim, then this means that in this space and under the Hamming metric, each code word bG will have no other code word within a radius of 2 around it. In fact, any point at Hamming distance 1 from a code word is guaranteed to be closer to that code word than any other, and thus we can correct one bit errors.

Definition 3 *An Error Correcting Code is a set of code words $C \subseteq \Sigma^n$. The minimum distance of C, written $\Delta(C)$, is the minimum Hamming distance between pairs of different code words in C.*

Definition 4 *An Error Correcting Code is said to be e error detecting if it can tell that an error occurred when there were $\leq e$ errors, and at least one error occurred. It is said to be t error correcting if it can tell where the errors are when there were $\leq e$ errors, and at least one error occurred.*

Definition 5 *The Hamming weight $wt(v)$ of a vector v is the number of non-zero coordinates of v.*

Proposition 6 *$\Delta(C) = 2t + 1 \Leftrightarrow$ the code C is $2t$ error detecting and t error correcting.*

Proof : $\Delta(C) = 2t + 1 \Rightarrow$ 2t error detecting since the word would simply not be a code word. $\Delta(C) = 2t + 1 \Rightarrow$ t error correcting since the word would be closer to the original code word than any other code word. We omit the reverse implications for now, though we note that the case for t error correcting is easy. ■

We now present some key code parameters:

q = $|\Sigma|$
n = block length of code(encoded length)
k = message length(pre-encoded length) = $\log_q |C|$
d = $\Delta(C)$

Usually, we fix three of the above parameters and try to optimize the fourth. Clearly, larger k and d values, and smaller n values are desirable. It also turns out that smaller q values are desirable. We may also try to maximize the rate of the code(efficiency ratio) $\frac{k}{n}$ and the relative distance $\frac{d}{n}$. We denote an error correcting code with these parameters as a $(n, k, d)_q$ code.

Thus, proving our claim boils down to showing that $\{bG|b \in \{0,1\}^4\}$ is a $(7, 4, 3)_2$ code.

Proof : Assume that $\Delta(b_1G, b_2G) < 3$ for $b_1 \neq b_2 \Rightarrow \Delta((b_1 - b_2)G, 0) < 3 \Rightarrow \exists$ non-zero $c \in \{0,1\}^4$ s.t. $wt(cG) < 3$.

Consider the matrix

$$H = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

We show in problem set 1 that $\{bG|b\} = \{y|yH = 0\}$. Hence, it suffices to prove that: if a non-zero $y \in \{0,1\}^7$ s.t. $yH = 0 \Rightarrow wt(y) \geq 3$, since this would contradict that $wt(cG) < 3$ for some non-zero c.

Assume $wt(y) = 2 \Rightarrow 0 = yH = h_i + h_j$, where $h_1, h_2, ..., h_7$ are the rows of the matrix H. But by the construction of H, this is impossible. Assume $wt(y) = 1 \Rightarrow$ some row h_i has all zeros. Again, impossible by construction.

Thus $wt(y)$ is at least 3 and we are done. ■

From the properties of the matrix used above, we see that we may generalize the H matrix for codes of block length $n = 2^l - 1$ and minimum distance ≥ 3 simply by forming the $2^l - 1$ by l matrix where the rows are the binary representations of all integers between 1 and $2^l - 1$. The message length k that this generalized H corresponds to is left as an exercise.

Error correcting codes find application in a variety of different fields in mathematics and computer science. In Algorithms, they can be viewed as interesting data structures. In Complexity, they are used for pseudo-randomness, hardness amplification, and probabilistically checkable proofs. In Cryptography, they are applied to implement secret sharing schemes and proposal cryptosystems. Finally, they also arise in combinatorics and recreational mathematics. We mention two examples in which error correcting codes are useful. The first requires the concept of pairwise independence.

Definition 7 *$S \subseteq \{0,1\}^n$ is pairwise independent if $\forall\ i, j \in \{1, ..., n\}$, and $b, c \in \{0,1\}$, we have*

$$Pr_{y \in S}[y_i = b \text{ and } y_j = c] = \frac{1}{4}$$

Given a randomized algorithm A that uses n random, independent bits, it may be the case that we require only limited independence from these bits. In particular, if we only require pairwise independence, then we can derandomize A as follows. Instead of picking a random y, we fix y and run the algorithm for all $y \in S$, since for our purposes, the set S is good enough. Since A had a positive probability of success before, it must succeed for some $y \in S$, and so we have a deterministic algorithm. However, the running time is now $O(|S|) \cdot runningtime(A)$, and we would like to make sure that S is not too large. It turns out that $\exists$ S s.t. $|S| = O(n)$.

The second example is the Hat Problem. There are n people wearing black or white hats. Each person can see all hats except his/her own, and no communication is allowed. Each person tries to guess his/her own hat color and writes down "Black", "White", or "Abstain". If everyone abstains, the game is lost. If someone guesses incorrectly, the game is lost. Otherwise, the game is won. What's the optimal strategy and the corresponding probability of winning?

Later in the course, we will cover more history, more error correcting codes, lower bounds and impossibility results, algorithms in coding theory, and various applications.