Notes for Lecture 1

1.1 The Basic Coding Problem

A party called the *sender* has a message $x \in \Sigma^k$ that he wants to send to another party called the *receiver*.

The sender and the receiver communicate through a *noisy channel* that introduces errors. To eliminate errors (or, at least, to dramaticallt reduce the probability of errors) the sender first encodes his message using an encoding function $C : \Sigma^k \to \Sigma^n$ with $n > k$ that introduces some redundancy, and then sends $C(x)$ through the channel. The receiver receives a string y that is possibly different from $C(x)$ because of transmission errors. The receives then feeds y to a *decoding algorithm* D that, under some assumption about the error pattern introduced by the channel, is able to compute x.

We would like to design efficient procedures C and D such that the above holds under general assumptions about the channel and with n not much larger than k.

1.2 Error-Correcting Codes

The *Hamming Distance* $d_H(a, b)$ between two strings $a, b \in \Sigma^n$ is the number of entries i such that $a_i \neq b_i$.

An $[n, k, d]_q$ code is a function $C : \Sigma^n \to \Sigma^k$ such that

- $|\Sigma| = q$;
- For every $x, x' \in \Sigma^k$, $d_H(C(x), C(x')) \geq d$.

The paramerer k is called the *information length* of the code and n is called the *block length*. Abusing terminology a little, we call d the *minimum distance* of the code .[1]

If a $[n, k, d]_q$ code admits a decoding procedure that is always able to correct e errors, then it must be $d \geq 2e + 1$. Conversely, if $d \geq 2e + 1$ then there is a (possibly not efficiently computable) decoding procedure that is able to correct up to e errors.

Error-correcting codes solve the coding problem in models where there is a upper bound to the number of errors introduced by the channel. Error-correcting codes can also be used in settings where we have a probabilistic model for the channel, provided that we can show that with high probability the number of errors introduced by the channel is smaller than the number of errors that the decoding procedure can correct.

For a given k, we are interested in constructing $[n, k, d]_q$ codes where n is small (ideally, $n = O(k)$), d is large (ideally, $d = \Omega(n)$) and q is small (ideally, $\Sigma = \{0, 1\}$ and $q = 2$).

[1] More precisely, the minimum distance of a code C is $\min_{x \neq x'} \{d_H(C(x), C(x')\}$, so that if C is an $[n, k, d]_q$ code, then d is a *lower bound* to the minimum distance.

1.3 The Singleton Bound

We now prove a negative a result showing that it is not possible to have, at the same time, k and d very close to n.

Lemma 1 (Singleton Bound) *Let C be a $[n,k,d]_q$ code. Then*

$$k \leq n - d + 1$$

PROOF: Let $C' : \Sigma^k \to \Sigma^{n-d+1}$ be the projection of C to the first $n-d+1$ coordinates. That is, $C'(x)$ contains the first $n-d+1$ entries of $C(x)$. We see that C' must be an injective function, because if $C'(x) = C'(x')$ with $x \neq x'$, then $C(x)$ and $C'(x)$ can differ in at most $d-1$ coordinates, contradicting the fact that C has minimum distance at least d. But if C' is injective then its range must be at least as large as its domain, and so $n - d + 1 \geq k$. □

The following terminology gives perhaps a clearer perspective of the implication of the singleton bound. If C is a $[n,k,d]_q$ code, let us define its *rate* to be $R = k/n$ and its *relative minimum distance* to be $\delta = d/n$. Then the singleton bound says that $R \leq 1 - \delta + o(1)$.

1.4 The Gilbert Bound

First we recall some bounds about factorials and binomial coefficients. The binary entropy function is defined as $H_2(p) = p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}$. Stirling's approximation gives

$$n! = \Theta\left(\sqrt{n} \cdot \left(\frac{n}{e}\right)^n\right)$$

and so,

$$\binom{n}{k} = \Theta\left(\sqrt{\frac{n}{k(n-k)}} \cdot \left(\frac{n}{k}\right)^k \cdot \left(\frac{n}{n-k}\right)^{n-k}\right)$$

which implies

$$\log \binom{n}{k} = n \cdot H_2(k/n) + \Theta(\log n)$$

Lemma 2 (Gilber Bound) *For every n and $d < n/2$ there is a $[n,k,d]_2$ code such that*

$$k \geq n(1 - H_2(d/n)) - \Theta(\log n)$$

In terms of rate and relative minimum distance, the Gilbert bound shows the existence of codes for which $R \geq 1 - H_2(\delta) - o(1)$.

PROOF: We construct $C : \{0,1\}^k \to \{0,1\}^n$ using a greedy algorithm. Let $x_1, \ldots, x_{2^k}$ be an enumeration of the elements of $\{0,1\}^k$. We define $C(x_1)$ to be the all-zero string, and then, for $i = 2, \ldots, 2^k$, we define $C(x_i)$ to be the lexicographically first string of $\{0,1\}^n$ that has Hamming distance at least d with all the strings $C(x_1), \ldots, C(x_{i-1})$. We just need to show that the algorithm never gets stuck.

We use the probabilistic argument to show that if $c_1, \ldots, c_i$ are arbitrary strings of Σ^n (with $i < 2^k$), then there exists a string c that is at Hamming distance at least d from all of them. Let us pick c at random. For a particular c_j we see that

$$\mathbf{Pr}[d_H(c,c_j) < d] = 2^{-n} \cdot \sum_{i=0}^{d-1} \binom{n}{i} \leq 2^{-n} \cdot 2^{nH_2(d/n)+O(\log n)}$$

Applying a union bound, we have

$$\mathbf{Pr}[\exists j. d_H(c,c_j) < d] < 2^k \cdot 2^{-n} \cdot 2^{nH_2(d/n)+O(\log n)}$$

which is smaller than 1 under the assumption of the lemma.

Therefore the algorithm never gets stuck and the code C as in the conclusion of the Lemma exists. □

1.5 Linear Codes

An $[n,k,d]_q$ code $C : \mathcal{F}_q^k \to \mathcal{F}_q^n$ is *linear* if $\mathcal{F}_q$ is a field and C is a linear function.

For an element $c \in \mathcal{F}^n$, the *weight* of c, denoted by $w(c)$, is the number of non-zero coordinates of c. Equivalently, $w(c) = d_H(c, 0^n)$.

The following observation is useful in the study of linear codes.

Lemma 3 *If $C : \mathcal{F}^k \to \mathcal{F}^n$ is linear, then*

$$\min_{x,x'}\{d_H(C(x),C(x')\} = \min_{x \neq 0^k}\{w(C(x))\}$$

Proof: Notice that, in a linear code C, we always have $C(0^k) = 0^n$, so the right-hand side expression can be written as

$$\min_{x \neq 0^k}\{d_H(C(x), C(0^k)\}$$

To prove the lemma, we just have to show that $\min_{x,x'}\{d_H(C(x),C(x')\} \geq \min_{x \neq 0^k}\{d_H(C(x),C(0^k)\}$, since the other inequality is clear.

If $x', x'' \in \mathcal{F}^k$ are two different vectors, consider $C(x') - C(x'')$. By linearity, we have $C(x') - C(x'') = C(x' - x'')$, and from $x' \neq x''$ we have $x - x' \neq 0^k$. We also note that the Hamming distance between $C(x')$ and $C(x'')$ is equal to the number of non-zero entries of $C(x') - C(x'')$. So we have

$$d_H(C(x'),C(x'')) = w(C(x') - C(x'')) = w(C(x' - x'')) \leq \min_{x \neq 0^k}\{w(C(x))\}$$

□

1.6 The Varshamov Bound

In this section we show that the Gilbert bound can be achieved using a linear code.

Lemma 4 (Varshamov Bound) *For every n and $d < n/2$ there is a $[n,k,d]_2$ linear code such that*

$$k \geq n(1 - H_2(d/n)) - \Theta(\log n)$$

PROOF: We pick a linear function $C : \{0,1\}^k \to \{0,1\}^n$ at random by picking at random a $k \times n$ 0/1 matrix A and defining $C(x) = Ax$.

We use the probabilistic method to show that there is a positive probability that every non-zero message is encoded into a string with at least d ones.

For a particular $x \neq 0^k$, we note that $C(x)$ is uniformly distributed in $\{0,1\}^n$, and so

$$\mathbf{Pr}[w(C(x)) < d] = 2^{-n} \cdot \sum_{i=0}^{d-1} \binom{n}{i} \leq 2^{-n} \cdot 2^{nH_2(d/n)+O(\log n)}$$

A union bound shows that

$$\mathbf{Pr}[\exists x \neq 0^k : w(C(x)) < d] < 2^{k-1} \cdot 2^n \cdot 2^{nH_2(d/n)+O(\log n)}$$

which is smaller than 1 under the assumption of the lemma. □

1.7 Reed-Solomon Codes

Let q be a prime power and $\mathcal{F}_q$ be a finite field of size q. Let us fix n distinct elements of $\mathcal{F}_q$, $x_1, \ldots, x_n$, and let $k < n$. We define a $[n, k, n-k+1]_q$ linear code as follows.

Given a message $(c_0, \ldots, c_{k-1})$, we interpret it as a description of the polynomial $p(x) = c_0 + c_1 x + \ldots + c_{k-1} x^{k-1}$. The encoding of such a message will be the vector $(p(x_1), \ldots, p(x_n))$.

Such a procedure maps indeed a message of length k into an encoding of length n, and it is a linear mapping. To verify the claim about the minimum distance, if $(c_0, \ldots, c_{k-1})$ is not the all-zero vector, then the corresponding polynomial p is a non-zero polynomial of degree $k-1$. Such a polynomial can have at most $k-1$ roots, and so at lest $n-(k-1)$ of the values $p(x_1), \ldots, p(x_n)$ must be non-zero. Notice that such a code, called a Reed-Solomon code, meets the Singleton Bound.

1.8 The Berlekamp-Welch Algorithm

Decoding the Reed-Solomon code in a channel that introduces $e < (n-k+1)/2$ errors is equivalent to the following problem:

- Given: distinct elements $x_1, \ldots, x_n$ of $\mathcal{F}_q$, parameters e and k, with $e < (n-k+1)/2$, and elements $y_1, \ldots, y_n$ of $\mathcal{F}_q$;
- Find: a polynomial p of degree at most $k-1$ such that

$$\#i : p(x_i) \neq y_i \leq e$$

Note that, because of the constraint on e and k, the problem has always a unique solution p.

Consider the following algorithm:

1. If there is a polynomial p such that $p(x_i) = y_i$ for all $i = 1, \ldots, n$, output p. Otherwise:
2. Find polynomials $E(x)$ and $N(x)$ such that

(a) E is not identically zero;

(b) $E(x)$ has degree at most e and $N(x)$ has degree at most $e+k-1$;

(c) For every $i=1,\ldots,n$, $N(x_i)=E(x_i)\cdot y_i$.

3. Output $N(x)/E(x)$, or output `error` if $N(x)$ is not a multiple of $E(x)$.

We claim that the algorithm can be implemented to run in $O(n^3)$ time and that it correctly finds the unique solution.

Let p be the unique solution, and let $I=\{i : p(x_i)\neq y_i\}$. If I is empty, then the algorithm finds p in step (1). We want to show that, if I is not empty, then steps (2) and (3) can be implemented in $O(n^3)$ time and that the algorithm finds p in step (3).

Regarding efficiency, polynomial division can be realized in almost linear time, so we only need to worry about step (2). We can write $E(x)=\sum_{i=0}^{e} a_i x^i$ and $N(x)=\sum_{i=0}^{e+k-1} b_i x^i$, and see the problem of realizing step (2) of the algorithm as the problem of finding coefficients a_i and b_i such that the constraints $N(x_i)=E(x_i)y_i$ are satisfied. Such constraints are *linear* in a_i and b_i, and so, if the set of constraints has a non-zero solution, then a non-zero solution can be found in cubic time using Gaussian elimination.

To see that a non-zero solution exists, let us define the polynomials $E(x)=\prod_{i\in I}(x-x_i)$ and $N(x)=E(x)\cdot p(x)$. Then by definition the degree of E is at most e, and the degree of N is at most $k-1+e$. Furthermore, if $i\in I$ we have $E(x_i)=0$ and $N(x_i)=0$, so that $N(x_i)=E(x_i)y_i=0$; if $i\notin I$ we have $N(x_i)=E(x_i)p(x_i)=E(x_i)y_i$, and so all the constraints are satisfied. Finally, I is not empty (otherwise we would have found p at step (1) of the algorithm) and so E is not the all-zero polynomial.

Regarding correctness, let E,N be the polynomials defined above, and let E',N' be the solution found by the algorithm in step (2), we want to argue that $N(x)/E(x) = N'(x)/E'(x)$, which is the same as $N(x)E'(x)=N'(x)E(x)$. The polynomials $N(x)E'(x)$ and $N'(x)E(x)$ have degree at most $2e+k-1<n$, and so, to show that they are equal, it is enough to show that they agree in n inputs. This is easily verifier because, for every $i=1,\ldots,n$, we have

$$N(x_i)E'(x_i)=y_iE(x_i)E'(x_i)=N'(x_i)E(x_i)$$