Complexity Theory II Course Instructor: V. Arvind

Lecture 6 : Aug 23, 2006

Lecturer: V. Arvind *Scribe: Vipul Naik*

1 Overview

In the last lecture we ...

2 Today's topic: Error correcting codes

2.1 Goal for the day

In this lecture, we shall look at:

- The notion of communication across a *binary symmetric channel* in the presence of *noise*
- The use of *error-correcting codes* with a view to optimizing communication with a good *rate* and good *capacity for correcting errors*
- Upper bounds on how good an error-correcting code can be: the *Shannon capacity* and the *Singleton bound.*
- Lower bounds that guarantee the existence of "good" codes, in fact, that guarantee that the *average* code is good: the *Gilbert bound* and the *Varshmov bound.*
- Codes that can be encoded and decoded efficiently: the *Reed Solomon code* and the *Berlekamp Welch decoder*

2.2 Binary symmetric channel

A binary channel is a communications channel from a *sender* to a *receiver* where the sender sends *binary data* (a stream of 0s and 1s) to the receiver, and there is a likelihood that some of the bits may get flipped. A binary *symmetric* channel is a binary channel where the bit flip probabilities are:

- Same for flip from 0 to 1 and 1 to 0 (hence the term *symmetric*)

- Same for all bits (probability $p < 1/2$)
- *Independent* for all bits. That is, the flipping of one bit in no way affects the flipping of another bit.

In the BSC setup, the *expected number of bit flips* in the received message is p times the length of the message.

2.3 Use of error-correcting codes

Although the BSC specifically talks of a binary channel, we shlal develop the theory overa general finite alphabet.

The idea of error-correcting codes is to use the fact that if the total number of bit flips (in general, changes in letters) is less than a certain threshold, the original message should be recovered perfectly. This is achieved by introducing *redundancy* into the original message. Here's how it's done.

- The original message, called the *message*, is what the sender is trying to convey. The collection of all possible messages is the *message space* and the length of the message is termed the *message length.*
- The sender uses an *encoding algorithm* to convert the message into a (somewhat larger) message called the *code* or *encoded message.* This is what is sent across the channel. The collection of all possible codes is termed the *code space.*
- The channel flips some bits of the encoded message.
- The receiver receives a *corrupted encoded message.* The receiver applies a *decoding algorithm* to this corrupted message. If the message is completely uncorrupted, the decoding algorithm must retrieve the original message. Hopefully, if only a few of the letters of the encoded message have been changed, the decoding algorithm still recovers the message accurately.

Three parameters used to measure the effectiveness of the encoding-decoding algorithms:

- Comparison between the size of the message space and the code space. Ideally, the code space should not be much larger than the message space. A reasonable thing to hope for is that the *code length* is linear in the *message length.*

- The maximum number of bit flips (letter changes) that can be tolerated still giving the correct decoding.
- The efficiency of the encoding and decoding algorithms, as a function of the length of the original message.

We shall use the following symbols throughout the discussion of codes:

Σ	the alphabet
k	message length
Σ^k	message space
n	code length
Σ^n	code space
$C : \Sigma^k \to \Sigma^n$	encoding algorithm
$D : \Sigma^n \to \Sigma^k$	decoding algorithm

2.4 Hamming distance

The Hamming distance between two words in Σ^l is the number of coordinates where they differ. The Hamming distance between x and y in Σ^l is denoted as $d_H(x, y)$. Some properties of the Hamming distance:

- The Hamming distance is a metric. That is, d_H is symmetric, nonnegative, and satisfies the triangle inequality.
- When $\Sigma = F_q$ (a finite field) the Hamming distance is translation-invariant on the vector space F_q^n. That is, $d_H(x, y) = d_H(x + z, y + z)$ for all $z \in F_q^n$.

For binary codes, the Hamming distance between two words x and y measures the *minimum* number of bit flips to go from x to y. Thus, Hamming distance is a realistic measure of distance via bit flips.

For a code, the minimum distance d is defined as the minimum of distances between pairs of codewords:

$$d = \min_{x \neq y \in \Sigma^k} d_H(C(x), C(y)) \tag{1}$$

2.5 Error correcting/decoding radius

The idea in coding theory as we shall study it is to replace the probabilistic BSC model with a deterministic model:

- The sender takes the message $m \in \Sigma^k$ and uses C to convert m to a codeword $c \in \Sigma^n$.
- c is transmitted over the channel. The channel has the property that it will alter at most e coordinates of c'.
- The receiver obtains c', with the guarantee that $d_H(c, c') \leq e$, and is expected to recover m using the decoding algorithm D.

Two easy observations:

Observation 1 (error-correcting radius and minimum distance)**.** Let $c = C(m)$. The encoding-decoding algorithm is said to be capable of correcting upto e errors if $D(c') = m$ whenever $d_H(c', c) \leq e$. The maximum e for which this holds is termed the *error-correcting radius* or *decoding radius* of the algorithm. Then, if e is the error-correcting radius and d is the minimum distance:

$$e \leq \frac{d-1}{2}$$

Proof. The intuitive idea is that if e is the error-correcting radius, then the closed Hamming spheres of radius e about the codewords must be pairwise disjoint. Let x and y be codewords such that $d_H(x, y) = d$. Then, x and y differ in exactly d coordinates.

Suppose $2e \geq d$. Then, consider z obtaiend by changing e of the differing coordinates from x to y. Clearly, $d(x, z) = e$ and $d(y, z) = d - e \leq e$. Hence z lies in the Hamming balls of radius e centered at x and y, causing a contradiction. Hence, the assumption $2e \geq d$ is flawed, giving the result. $\square$

The converse is also true, albeit only at a theoretical level:

Observation 2. *If $e \leq \frac{d-1}{2}$, then there exists a decoding algorithm that corrects upto e errors. However, we cannot say* a priori *whether this algorithm will be polynomial time, or even efficient.*

Proof. If $e \leq (d-1)/2$, triangle inequality forces the Hamming balls about codewords to be pairwise disjoint. Thus, whatever word the receiver gets lies in *at most one* Hamming ball. The receiver can now apply the brute-force algorithm: compute distance from every codeword, and find the codeword from which the distance is not greater than the error-correcting radius. $\square$

2.6 Formal measures of goodness of codes

We can now formally define two measures of the efficiency of the code:

- *Rate* R of the code: This is the ratio of message length to code length. Using smybols as above, $R = k/n$.
- *Minimum distance* d of the code: This is $\min_{x \neq y} d_H(x, y)$ where x and y range over codewords.
- *Relative minimum distance* δ of the code: This is d/n where d is the minimum distance and n is the code length.

The four challenges in front of us:

- Make R as large as possible
- Make δ as large as possible
- Make C as efficient as possible (in terms of time)
- Make D as efficient as possible (in terms of time), and try to make sure that D decodes upto the full error-correcting radius.

A code over an alphabet of size q with message length k, code length n and minimum distance $\geq d$ is termed a $(n, k, d)_q$ code.

2.7 Families of codes

A *family of codes* is a collection of codes parametrized by $i \in \mathbb{N}$, wherein the i^{th} code is a (n_i, k_i, d_i) code. We can have two kinds of families of codes: those with constant alphabet size, and those where the alphabet size also depends on i. Our ultimate aim is to construct families of codes over a constant sized alphabet (in fact, a binary alphabet) that are *good.* Further, we assume, for simplicity, that the k_i are increasing.

A first consideration for families of codes is the relative spacing of valid message lengths. Ideally, we would like to be able to encode and decode words of any length. Thus, we would like to be able to *pad* any message to a message length k_i. Because complexities of encoding and decoding algorithms are given as functions of k_i, we want that the expansion in the message to reach a valid message length is by a constant factor. Thus, we would like that the ratios k_{i+1}/k_i are bounded by a constant.

The four measures of goodness of a code can be extended to give measures of goodness for a family of codes:

- The rate R: We look at the limit (inferior) of the rates of all codes in the family.
- The relative minimum distance δ: We look at the limit (inferior) of the relative minimum distance of all codes in the family.
- The encoding algorithm C: We measure the bound on the running time of C_i (the encoding algorithm for the i^{th} code) as a function of k_i (the i^{th} message length). Note that because the ratios k_{i+1}/k_i are constant, the running time of the algorithm as a function of actual message length differs only by a constant factor (if the running time is polynomial).
- The decoding algorithm D: We measure the bound of the running time of D_i (the decoding algorithm for the i^{th} code) as a function of n_i (the i^{th} code length). Note that if the rate R is bounded below by a constant, this growth rate is roughly the same as the growth rate with respect to k_i (The uncoded message).

2.8 Probabilistic model

Fact 3. A code with error-decoding radius e for code length n is "good" for binary symmetric channels with the bit flip probability $\leq ce/n$ where $c \leq 1$ is a suitable constant(?).

The heuristic justification for the above: the *expected number of errors* in the received code word is pn where p is the bit flip probability. Hence, if $pn \leq ce$, the expected number of errors is ce, which is quite small with respect to e. By using Chernoff or Chebyshev bounds, and the nature of the probability distribution, we can actually obtained that the probability that the number of errors is bounded above by e is quite high, dependign on the choice of c. Moreover, the suitable choice of c depends only on p and not on n.

The *Shannon capacity* of a channel is the maximum posible value a probabilistic analogue of what we called the *rate*.

Fact 4 (Shannon)**.** The Shannon capacity of a binary symmetric channel with bit flip probability p is $1 - H_2(p)$ where $H_2(p)$ is the Shannon entropy of the probability distribution of the distribution $(p, 1-p)$.

3 Bounds both ways

3.1 Singleton bound: an upper bound

The Singleton bound (named after a person called Singleton) establishes an *additive tradeoff* between the rate R and relative minimum distance of a code. It works for codes over all finite alphabets.

Theorem 5 (Singleton bound)**.** For a code with mesage length k, code length n and relative minimum distance d:

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

In particular, for a family of codes, as $n \to \infty$, we have:

$$\lim_{n\to\infty} R + \delta = 1$$

Proof. Consider the map $\Sigma^n \to \Sigma^{k-1}$ obtained by projecting onto the first $k-1$ coordinates. Composing this with the encoding map, we obtain a map $\Sigma^k \to \Sigma^{k-1}$. The map cannot be injective, hence there are two distinct messages whose corresponding codewords are equal in the first $k-1$ positions. The Hamming distance between these two codewords is bounded by $n-k+1$, giving that $d \leq n-k+1$. □

Prima facie, the Singleton bound seems a very weak upper bound. Surprisingly, however, it is possible to obtain a *lower bound* such that the gap between the two is not much. Even more surprisingly, a code satisfying the lwoer bound can be found *randomly.*

3.2 Volume of the Hamming ball

We are interested in estimating the number of points in the Hamming ball of radius $d-1$ in Σ^n where $|\Sigma| = q$.

For any r, the number of points of weight exactly equal to r is:

$$(q-1)^r \binom{n}{r}$$

To find the number of points in the Hamming ball of radius $d-1$, we need to sum up for $0 \leq r \leq d-1$. If we denote this by $V_q(n,d)$, we have

$$V_q(n,d) = \sum_{r=0}^{d-1} (q-1)^r \binom{n}{r}$$

We want to find a good upper bound on $V_2(n,d)$. Here goes:

Proposition 6 (Volume bound)**.**

$$V_2(n,d) \leq n2^{nH_2(\delta)}$$

Proof.

$$\begin{aligned}
V_2(n,d) &\leq sum_{r=0}^{d-1}\binom{n}{r} \\
\implies V_2(n,d) &\leq n\binom{n}{d} \\
\implies V_2(n,d) &\leq n\binom{n}{\delta n} \\
\implies V_2(n,d) &\leq n \exp n \log n - (\delta n)\log(\delta n) - ((1-\delta)n)\log((1-\delta)n) \\
\implies V_2(n,d) &\leq n \exp -n(\delta \log \delta + (1-\delta)\log(1-\delta)) \\
\implies V_2(n,d) &\leq n2^{nH_2\delta}
\end{aligned}$$

□

3.3 Gilbert bound

Theorem 7 (Gilbert bound)**.** For $d < n/2$, there is a $[n,k,d]_2$ code such that $k \leq n(1 - H_2(\delta)) - \log n - c$ where $c > 0$ si a suitable positive constant.

Proof. A greedy algorithm produces the code successfully with high probability. Let $C : \{0,1\}^k \to \{0,1\}^n$ be defined as follows. Enumerate the elements of the message space as $x_1, x_2 \ldots$.

Traverse over the x_i and to each x_i, assign a value $y_i = C(x_i)$ that satisfies the constraints imposed by y_j for $j < i$. In other words, at each step, pick y_i such that for all $j < i$:

$$d_H(y_i, C(x_j)) \geq d$$

To show that the greedy algorithm works, we need to show that *at each stage*, there exists at least one candidate for y_i. Equivalently, it suffices to show that the union of Hamming balls of radius $(d-1)$ about the previous y_is does not cover the whole region. The volume bound (proposition **??**) tells us that:

$$\begin{aligned}
\text{Volume of first } i-1 \text{ Hamming balls} &\leq \sum_{j=1}^{i-1} \text{Volume of } j^{th} \text{ Hamming ball} \\
\implies \text{Volume of first } i-1 \text{ Hamming balls} &\leq (i-1)V_2(n,d) \\
\implies \text{Volume of first } i-1 \text{ Hamming balls} &\leq 2^k V_2(n,d) \\
\implies \text{Volume of first } i-1 \text{ Hamming balls} &\leq 2^k n 2^{nH_2\delta}
\end{aligned}$$

Putting $k \leq n(1 - H_2(\delta)) - \log_2(n)$ gives:

$$\begin{aligned}
\text{Volume of first } i-1 \text{ Hamming balls} &\leq n2^{n-\log_2(n)-c} \\
\implies \text{Volume of first } i-1 \text{ Hamming balls} &\leq \frac{2^n}{c}
\end{aligned}$$

The first $i-1$ balls cover at most $1/c$ fraction of the points. Thus, there is always a candidate for y_i outside the Hamming balls. In fact, the probability that a randomly picked point lies outside the Hamming balls is nonnegligible. □

Observation 8 (Randomized algorithm also works)**.** *Note that the above gives an algorithm in two senses:*

- *It gives a randomized polynomial-time algorithm, because at each stage, the number of candidates for y_i is a nonnegligible fraction of the total nubmer of words, and the* checking *for eahc y_i based on the past y_js is a polynomial-time subroutine.*
- *It gives a deterministic polynomial-time algorithm, if at each stage, we do a brute-force (lexicographically) on the y_js, and always pick the* least *y_j. The brute-force algorithm, however, may not (in fact, will not) run in polynomial time.*

3.4 Linear codes

Let Σ be identified with F_q, the finite field of q elements. Then the alphabet Σ^n can be treated as a vector space F_q^n with addition coordinatewise. The code is termed a *linear code* if the encoding map $C : \Sigma^k \to \Sigma^n$ is a linear map. Note that for linear codes, the code space is a k-dimensional vector subspace of Σ^n. Since the Hamming distance is translation-invariant, we have $d_H(x, y) = d_H(x - y, 0)$ and hence the minimum distance is equal to the distance between 0 and the closest codeword.

The *weight* of a codeword is defined as the number of nonzero coordinates in that codeword. The weight of a codeword equals its distance from zero. The above observations tell us that for a linear code, the minimum distance equals the minimum of weights over all possible codewords.

3.5 Varshamov bound

The Varshamov bound establishes the existence of, and in fact, gives a method to probabilistically obtain, a *good* linear code.

Theorem 9 (Gilbert bound)**.** For $d < n/2$, there is a $[n, k, d]_2$ linear code such that $k \leq n(1 - H_2(\delta)) - 3\log n$.

Proof. Pick a random linear transformation $C : F_2^k \to F_2^n$. We argue that C has a significant probability of defining a linear code of minimum distance $\geq d$. In other words, the probability that the image of C has a vector of weight strictly smaller than d is less than 1.

For each vector of weight less than d, the probability that it occurs in the code space is 2^{k-n}. by the *union bound*, the probability that the Hamming ball of radius $d - 1$ intersects the code space is bounded above by $2^{k-n}V_2(n, d)$. But from the same computation used for the Gilbert bound, we have that $2^{k-n}V_2(n, d) < 1$ when □

4 Reed-Solomon code

4.1 Setting of the code

The Reed-Solomon code is constructed over a suitable finite field of q elements.

4.2 Description of the code

Idea of the Reed-Solomon code:

- The message word goes into constructing the *coefficients* of the degree k (over a suitable finite field)
- The code word stores the *value* taken by the polynomial at a fixed collection of n points. This collection of points of evaluation is *fixed beforehand* and is independent of the message (hence, in particular, it is known to the receiver with certainty).

The n points of evaluation are denoted $x_1, x_2 \dots x_n$. The message (whose letters form the coefficients) is written $c_0 c_1 \dots c_{k-1}$.

4.3 Decoding: Berlekamp-Welch

Proposition 10 (Berlekamp-Welch decoding)**.** The Reed-Solomon code can be decoded upto the maximum decoding radius, in polynomial time.

Proof. We first outline an algorithm, and then prove its correctness:

1. Find polynomials N and E such that:
   (a) $deg(E) \leq e \leq (n-k-1)/2$
   (b) $deg(N) \leq (n+k-1)/2$
   (c) For each i, $N(x_i) = E(x_i)y_i$
2. If $E|N$ output the polynomial $N(x)/E(x)$, otherwise output "error" (indicating that the code word went outside a Hamming ball).

Let us examine step (1) carefully. We begin by proving that there do exist candidates for E and N. let I be the set of coordinates at which the polynomial value got corrupted. Then the following are possible candidates for N and E:

$$\begin{aligned} E(x) &\equiv \prod_{i \in I}(x - x_i) \\ N(x) &\equiv E(x)P(x) \end{aligned}$$

Let's see why this works.

1. since $|I| \leq e$, $deg(E) \leq e$.
2. Since $deg(P) = k$, we have $deg(N) \leq k + e$. But $e = (d-1)/2$ and $d = n - k$, so simplifying, we get $deg(N) \leq (n+k-1)/2$.
3. the condition $N(x_i) = E(x_i)y_i$ is clearly satisfied for $i \in I$, because $E(x_i) = 0$ and $N(x_i) = 0$ as well. For $i \notin I$, $N(x_i) = E(x_i)p(x_i)$. But $p(x_i) = y_i$ (Because this is an uncorrupted letter) and hence $N(x_i) = E(x_i)y_i$.

Suppose (N, E) and (N', E') are two pairs saitsfying the definition. Then we claim that $NE' = N'E$. To show this, we observe that NE' and $N'E$

agree on n values: namely all the x_is. Because the total degree on both sides is at most $(n-1)$, the two polynomials are identically equal.

Thus we are guaranteed two things: there exists a pair (N, E) satisfying the three conditions with the property that $p(x) = N(x)/E(x)$, and that any other pair (N', E') satisfying the three conditions also gives the same value of $p(x)$. $\square$