CSE 533: Error-Correcting Codes **(Autumn 2006)**

Lecture X: The Elias-Bassalygo Bound

October 18, 2006

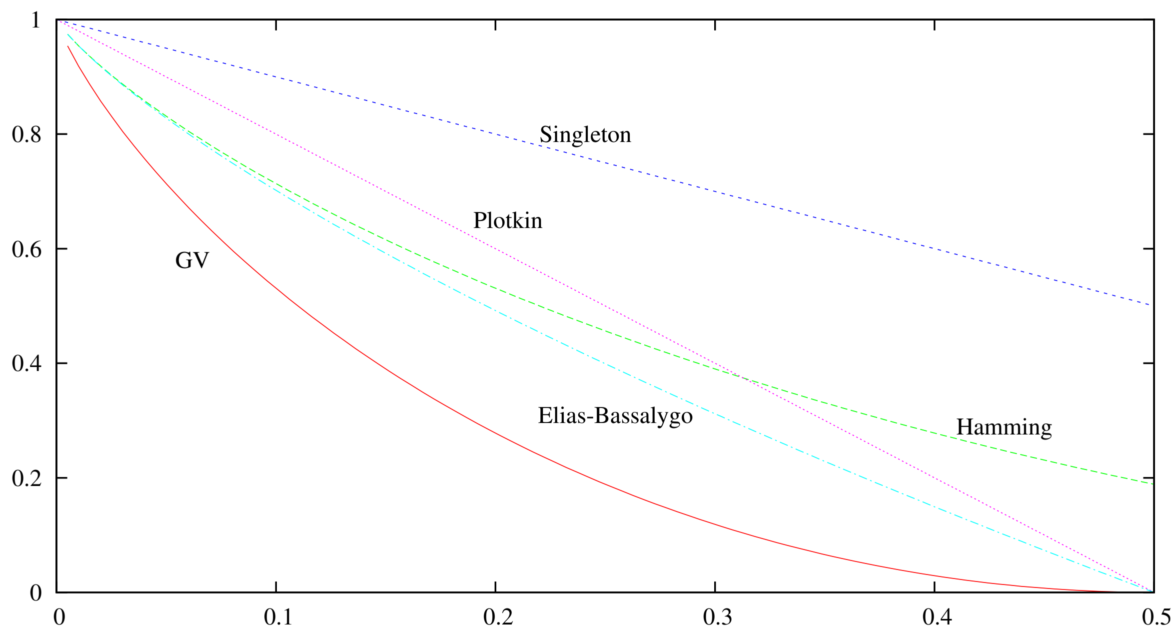
Lecturer: Venkatesan Guruswami *Scribe: Matt Cary*

1 Recapitulation

We currently know the following four bounds, three of them upper bounds (which tell us what rate-distance combinations are impossible), and one lower bound (which tells us what rate-distance combinations we can achieve). In the following, R is the rate and δ the relative distance of a code. For example, a (n, k, d) code has $R = k/n$ and $\delta = d/n$. $H_q(x)$ is the q-ary entropy function, $H_q(x) = x\log_q\big((q-1)/x\big) + (1-x)\log_q\big(1/(1-x)\big)$.

Gilbert-Varshamov:	$R \geq 1 - H_q(\delta)$	(1)
Volume (Hamming):	$R \leq 1 - H_q(\delta/2)$	(2)
Singleton:	$R \leq 1 - \delta$	(3)
(Binary) Plotkin:	$R \leq 1 - 2\delta$	(4)

Below we plot these bounds, along with the Elias-Bassalygo bound we will prove today, for the binary case ($q = 2$).



Note that the Shannon bound is achieved at the Hamming bound, so before today's bound, any hope of achieving the Shannon bound using combinatorial codes, according to the above graph, is only possible with δ less than about 0.3. Today we will extinguish even this faint hope, by showing the Elias-Bassalygo upper bound which is a strict improvement on all our current upper bounds, as the graph shows.

This Elias-Bassalygo bound is the best known bound that can be shown by elementary methods. The current world-record upper bounds extend this proof technique with bounds derived from linear programming duality.

2 The Johnson Bound

We begin by defining a parametrized bound known as the *Johnson bound*. Let n be a block length, d be a distance, and e be a radius. Then $J(n, d, e)$ is the maximum number of codewords contained in any ball of radius e, of any code with blocklength n and distance d. Hence $J(n, d, n)$ is our usual upper bound, usually denoted $A(n, d)$: the most number of codewords in any code of distance d and blocklength n. Looking $J(\cdot)$ from the other end, we have that $J(n, d, \lfloor (d-1)/2 \rfloor) = 1$, as in a distance d code, the balls of radius $\lfloor (d-1)/2 \rfloor$ do not intersect. As a side note, Johnson actually studied the maximum number of codewords of the specified distance all lying on the *surface* of the specified sphere, that is, $\mathrm{wt}(c) = e$ rather than $\mathrm{wt}\, c \leq e$ as we are using.

While there is a geometric proof of the Johnson bound, we will prove it using a useful combinatorial technique of counting two ways. Let $c_1, \ldots, c_M$ be M codewords, with $\Delta(c_i, c_j) \geq d$ for all $i \neq j$, and $\mathrm{wt}(c_i) \leq e$ for all i. Note that by translation we can assume without loss of generality that our sphere is centered at zero. We will now bound the sum all distances $S = \sum_{1 \leq i < j \leq M} \Delta(c_i, c_j)$ in two ways.

First, as $\Delta(c_i, c_j) \geq d$, we have that

$$S \geq \binom{M}{2} d. \tag{5}$$

Now, consider the codewords arranged in an $n \times M$ matrix, and look at the i-th column. Suppose this contains m_i 1's, and $M - m_i$ zeros. Then each pair of different bits contributes one to S, for a total of $m_i(M - m_i)$ per column. Define $\sum m_i / M = e'$ as the average weight per codeword, and note that $e' \leq e$ as $\mathrm{wt}(c_i) \leq e$ for all codewords. Hence

$$S = \sum m_i (M - m_i) = M^2 e' - \sum m_i^2.$$

As Cauchy-Schwartz tells us $\sum m_i^2 \geq \left(\sum m_i\right)^2 / n$, we have that

$$\leq M^2 e' - M^2 e'^2 / n.$$

Thus by combining this with (5), we have that

$$M(M-1)d \leq 2S \leq 2M^2 (e' - e'^2/n).$$

Rearranging,

$$M(d - 2e' + 2e'^2/n) \leq d.$$

Thus if the left-hand side is positive, we can divide to bound M by

$$\begin{aligned} M &\leq \frac{nd}{nd - 2e'n + 2e'^2} \\ &= \frac{2nd}{(n-2e')^2 - n(n-2d)} \\ &\leq \frac{2nd}{(n-2e^2) - n(n-2d)}, \end{aligned}$$

as $e' \leq e$. Finally, if the denominator is positive, it must be at least one as it is an integer. Rearranging the denominator shows that it is positive when $e < (1 - \sqrt{n(n-2d)})/2$. Hence we have that

$$J(n, d, e) \leq 2nd \quad \text{if } \frac{e}{n} < \frac{1}{2}\left(1 - \sqrt{1 - 2d/n}\right).$$

In other words, any ball with radius smaller than e as above, contains only polynomially many codewords. In fact, the quantity we are really interested in is *when* the Johnson bound holds, and not what the Johnson bound in fact is, just as long as it is polynomial. For example, it is not very intuitive that the bound should *increase* with d: as d gets bigger, we know we can fit fewer, not more, codewords in a ball of a certain size! But that won't actually concern us much; we can use the very weak bound $d \leq n$ and still have $O(n^2) = \mathrm{poly}(n)$ codewords in our ball.

Hence we define $J(\delta) = (1 - \sqrt{1 - 2\delta})/2$ as a bound on what the maximum size ball containing a polynomial number of codewords of relative distance δ can be. Though we will not prove it here, the Johnson bound can be extended to q-ary alphabets by

$$J_q(\delta) = \frac{q-1}{q}\left(1 - \sqrt{1 - \frac{q\delta}{q-1}}\right)$$

and to an alphabet-independent version that holds for all q,

$$J(n, d, e) \leq nd \quad \text{if } e < n - \sqrt{n(n-d)}.$$

3 Using the Johnson Bound

We know show how the Johnson bound can be used to give an upper bound for the coding problem. The technique we will use has been called the *fishnet* method, and is useful to know about. The outline is that we have a bound on how many codewords can fit into a ball of a certain size. We'll then show that there exists a ball containing relatively large fraction of the code, which when combined with our bound constrains the size of the code. Intuitively, we'll set up a net that only catches big fish, and then prove that we catch one.

Lemma 3.1. *Given a code C of blocklength n, for any e, n there exists a Hamming ball of radius e containing at least $|C| \cdot \mathrm{Vol}(n,e)/2^n$ codewords.*

Proof. Consider the event of picking a Hamming ball B of radius e around a random center. For each $c \in C$, let X_c be an indicator variable equal to 1 if $c \in B$ and 0 otherwise. Then for all c, $\mathbf{E}(X_c) = \Pr(X_c) = \mathrm{Vol}(n,e)/2^n$. The total number of codewords in B is equal to $\sum X_c$, so by linearity of expectation, $\mathbf{E}(\text{\# codewords in } B) = \sum \mathbf{E}(X_c) = |C| \, \mathrm{Vol}(n,e)/2^n$. As there must exist at least one ball achieving the expectation, the lemma is proved. □

Theorem 3.2 (The Elias-Bassalygo Bound)**.**

$$R \leq H\left((1 - \sqrt{1-2\delta})/2\right)$$

Proof. Set $e = J(\delta)$. Then by the above lemma there is a ball of radius e containing $|C| \, \mathrm{Vol}(n,e)/2^n$ codewords. By the Johnson bound, no such ball can contain more than $2nd$ codewords, hence $|C| \, \mathrm{Vol}(n,e)/2^n < 2nd$. Rearranging and using the entropy approximation for the volume of a Hamming ball, we have that

$$|C| \leq 2^{1-H\left(J(\delta)\right)n} \cdot 2^{o(n)}.$$

By taking $n \to \infty$, we show the bound. □

As shown in the graph at the beginning of this lecture, this bound is a strict improvement to both the Hamming bound and the binary Plotkin bound.

4 Revisiting the Gilbert-Varshamov Bound: Linear Codes

The Elias-Bassalygo bound has been slightly improved, but there has been no asymptotic improvement for binary codes to the GV bound. Hence it is still open whether the GV bound of $R = 1 - H(\delta)$ the best asymptotic rate that can be achieved. One way to address this is to study what sort of codes meet the GV bound. The version of the bound that we saw constructed a general code, so it is natural to ask if there is a *linear* code that meets the GV bound. We show below that there is. The theorem below is due to Varshamov, who proved it independently of the theorem of Gilbert that achieved a similar bound for general, nonlinear codes. The two results are usually combined, giving us the Gilbert-Varshamov bound (or Varshamov-Gilbert bound, depending on if you learned it in Cyrillic or not).

Theorem 4.1. *For any $\varepsilon > 0$ and $\delta \leq 1/2$, there is a sufficiently large n such that there exists a linear $[n, k, \delta n]$ code with $kn \geq 1 - H(\delta) - \varepsilon$.*

Proof. Pick a random linear code by forming $(n-k) \times n$ Boolean parity-check matrix H with each entry chosen independently at random. The probability that the resulting code does not have distance at least $d = \delta n$ is equal to the probability that there exists a point x with weight $\leq d-1$ and $Hx = 0$. Let us call this event $\mathcal{E}$. Then by the union bound, we have that

$$\Pr(\mathcal{E}) \leq \sum_{\substack{\mathrm{wt}(x)\leq d-1 \\ x\neq 0}} \Pr(Hx = 0).$$

Consider computing the product Hx one row at a time. For any random vector, $\Pr(\langle x, r\rangle = 0) = 1/2$, as fixing all random choices except one whose coefficient in x is one, we have that the dot product is 1 with probability $1/2$. Note this use critically the constraint that $x \neq 0$. Now as the rows of H are chosen independently, for any fixed x, $\Pr(Hx = 0) = (1/2)^{n-k}$. Hence

$$\begin{aligned} \sum_{\substack{\mathrm{wt}(x)\leq d-1 \\ x\neq 0}} \Pr(Hx = 0) &\leq \sum_{\mathrm{wt}(x)\leq d-1} \left(\frac{1}{2}\right)^{n-k} \\ &\leq 2^{H(\delta)n-n+k}, \end{aligned}$$

which goes to 0 with n if $k < \big(1 - H(\delta)\big)n$. Hence by choosing n sufficiently large, we ca reduce the probability to strictly less than one, implying there must exist a parity-check matrix that is nonzero on all words of weight less than d. □

Note that this proof says that, assuming that choosing its parity-check matrix at random induces a reasonable distribution over linear codes, almost all linear codes meet the GV bound. Thus we can construct one with high probability simply by choosing a random parity matrix after setting n and k appropriately. The problem is that there is no known efficient decoding procedure—in fact, the problem is NP-complete—so this fact does not have much practical impact.

An interesting exercise is to deterministically construct a linear code meeting the GV bound in $2^{O(n)}$ time. The trivial brute-force algorithm is $2^{O(n^2)}$ time, so a faster algorithm, while still exponential, is interesting.