Exploring Bounds

Tyler Friedman

April 24, 2015

1 Introduction

An interesting problem in coding theory is determining the maximum number of codewords in a code with certain parameters. For an n-dimensional code, we are interested in the maximum number of codewords such that the code has minimum distance d. Formally, we use the notation $A_q(n, d)$ and $B_q(n, d)$ to define the maximum of number of codewords in a code over $\mathbb{F}_q$ of length n and minimum distance d for an arbitrary (linear on non-linear) code and linear code, respectively.

For arbitrary n and d, it is difficult to find $A_q(n, d)$ and $B_q(n, d)$ exactly. In lecture, we have considered upper bounds—Sphere Packing, Singleton, and Greismer—as well as lower bounds—Gilbert and Varshamov—on these values. The purpose of this paper is to take a survey of other well-known bounds in the literature that we have not discussed. In the first part of the paper, we will consider the Plotkin Upper bound, the Elias Upper bound, and the Linear Programming Upper Bound. In the second part of the paper, we will consider asymptotic versions of Singleton, Plotkin, Hamming, and Elias. (I am very certain that I won't get to discuss all of these bounds, but I am currently unsure how much space each discussion will take up so I've listed all that I am considering.) For each bound, we will consider its proof as well as related examples. Lastly, in the final part we will consider lexicodes, an interesting subset of linear codes that meet the Gilbert Bound.

Some implementation specific details: for the Linear Programming bound, I will introduce basic linear programming concepts and as well as the Krawtchouck polynomials. For the Asymptotic Hamming bound I will introduce the Hilbert entropy function.

2 Upper Bounds

2.1 Plotkin Upper Bound

The Plotkin Bound is an upper bound that often improves upon the Sphere Packing Bound on $A_q(n, d)$.

Theorem 2.1.1 (Plotkin)**.** *Let C be an (n, M, d) code over $\mathbb{F}_q$ such that $rn < d$ where $r = 1 - q^{-1}$. Then*

$$A_q(n, d) \leq \left\lfloor \frac{d}{d - rn} \right\rfloor \tag{1}$$

Before proving this bound, it is important to note that it is only valid when d is sufficiently close to n. For large q, this is can be a considerable limiting factor. Consider that our restriction is $n < 2d$ for $q = 2$, $n < \frac{3}{2}d$ for $q = 3$, $n < \frac{4}{3}d$ for for $q = 4$, etc.

Proof. Let

$$S = \sum_{x \in \mathcal{C}} \sum_{y \in \mathcal{C}} \mathrm{d}(x, y) \tag{2}$$

Note that for $x, y \in \mathcal{C}$, $\mathrm{d}(x, y) = 0$ for $x = y$, and $d \leq \mathrm{d}(x, y)$ for $x \neq y$. This implies that

$$M(M - 1)d \leq S \tag{3}$$

This comes directly from equation 2. We have M possible codewords to consider for x in the outer summation, $M - 1$ distinct codewords from x to consider in the inner summation for y, and the distance between x and y will always be greater than or equal to the minimum distance d.

Next, let $\mathcal{M}$ be the $M \times n$ matrix whose rows are the codewords of $\mathcal{C}$. For $1 \leq i \leq n$, let $n_{i,\alpha}$ be the number of times $\alpha \in \mathbb{F}_q$ occurs in column i of $\mathcal{M}$. Note that $\sum_{\alpha \in \mathbb{F}_q} n_{i,\alpha} = M$ for $1 \leq i \leq n$. This is simply because there are M total elements of $\mathbb{F}_q$ in each column, so the sum over all $n_{i,\alpha}$ must necessarily be M.

I claim that we can rewrite S as a double sum over these $n_{i,\alpha}$ values. Specifically,

$$S = \sum_{i=1}^{n} \sum_{\alpha \in \mathbb{F}_q} n_{i,\alpha}(M - n_{i,\alpha}). \tag{4}$$

Note that $n_{i,\alpha}(M - n_{i,\alpha})$ is multiplying the number of times α occurs in column i by the number of times it doesn't occur in column i. In other words, we're counting the number of times two elements of the same column are not equivalent, exactly the condition that constitutes the Hamming distance. By summing over all α and then all columns, we arrive at the same value generated by 2.

Next, simplify:

$$\begin{aligned} \sum_{i=1}^{n} \sum_{\alpha \in \mathbb{F}_q} n_{i,\alpha}(M - n_{i,\alpha}) &= M \sum_{i=1}^{n} \sum_{\alpha \in \mathbb{F}_q} n_{i,\alpha} - \sum_{i=1}^{n} \sum_{\alpha \in \mathbb{F}_q} n_{i,\alpha}^2 \\ &= nM^2 - \sum_{i=1}^{n} \sum_{\alpha \in \mathbb{F}_q} n_{i,\alpha}^2 \end{aligned} \tag{5}$$

Recall the Cauchy-Schwarz inequality, which states that $\left(\sum_{i=1}^{n} x_i y_i\right)^2 \leq \left(\sum_{i=1}^{n} x_i^2\right)\left(\sum_{i=1}^{n} y_i^2\right)$ For our purposes, this means that $\left(\sum_{\alpha\in\mathbb{F}_q} 1 \cdot n_{i,\alpha}\right)^2 \leq q\left(\sum_{\alpha\in\mathbb{F}_q} n_{i,\alpha}^2\right)$.

Using this inequality, we obtain

$$S \leq nM^2 - \sum_{i=1}^{n} q^{-1}\left(\sum_{\alpha\in\mathbb{F}_q} 1 \cdot n_{i,\alpha}\right)^2 = nM^2 - \frac{nM^2}{q} = nrM^2. \tag{6}$$

Combining 3 and 6, we get $M(M-1)d \leq nrM^2$, which simplifies to

$$M \leq \left\lfloor \frac{d}{d-rn} \right\rfloor. \tag{7}$$

□

It is interesting to see exactly how the Sphere Packing and Plotkin bounds compare for specific values of n and d. For the following examples we will assume $q = 2$. Also, recall that if d is even, $A_2(n,d) = A_2(n-1,d-1)$.

Example 2.1.1. $(n,d) = (7,4)$.

Applying the Sphere Packing bound, we have $A_2(7,4) = A_2(6,3) \leq \frac{2^6}{1+6}$, giving us $A_2(7,4) \leq \mathbf{9}$.

Applying the Plotkin bound, we have $A_2(7,4) \leq \frac{4}{4-\frac{1}{2}\cdot 7} = \mathbf{8}$.

Example 2.1.2. $(n,d) = (9,6)$.

Applying Sphere Packing, we have $A_2(9,6) = A_2(8,5) \leq \frac{2^8}{1+8+28}$, giving us $A_2(n,d) \leq \mathbf{6}$.

Applying the Plotkin bound, we have $A_2(9,6) \leq \frac{6}{6-\frac{1}{2}\cdot 9} = \mathbf{4}$.

Using MATLAB, I sought to visualize how the Plotkin and Sphere Packing bounds compare. To do so, I plotted their values over $8 \leq n \leq 25$, with $d > rn$.

In order to turn my discrete 3d data into a smooth mesh surface, I needed a mechanism to smooth out my data. My solution was to utilize Delaunay triangulation.

It is also interesting to consider a broader picture of how Plotkin compares to Sphere Packing, Greismer, and other bounds we have considered in lecture. Below is a graph comparing rates of codes versus the Sphere Packing and Plotkin bounds that we can calculate for those values for different values of q.

2.2 Elias Upper Bound

Another upper bound on $A_q(n,d)$ is due to Elias in 1960. The Elias bound is generally weaker than other bounds we have discussed, but is important because of the asymptotic form of this bound that we will discuss towards the end of this paper.

In order to prove the Elias bound, two lemmas are necessary.

Lemma 2.2.1. *Let $\mathcal{C}$ be an (n, K, d) code over $\mathbb{F}_q$ such that all codewaords have weights at most w, where $w \leq rn$ with $r = 1 - q^{-1}$. Then*

$$d \leq \frac{Kw}{K-1}\left(2 - \frac{w}{rn}\right). \tag{8}$$

Lemma 2.2.2. *Suppose C is an (n,M,d) code over $\mathbb{F}_q$. Then there is an (n, M, d) code $\mathcal{C}'$ over $\mathbb{F}_q$ with an (n,K,d) subcode $\mathcal{A}$ containing only codewords of weight at most w such that $K \geq MV_q(n, w)/q^n$.*

Theorem 2.2.1. *Let $r = 1 - q^{-1}$. Suppose that $w \leq rn$ and $w^2 - 2rnw + rnd > 0$. Then*

$$A_q(n, d) \leq \frac{rnd}{w^2 - 2rnw + rnd} \cdot \frac{q^n}{V_q(n, w)}. \tag{9}$$

We can work through a few examples to see how the Elias bound operates.

Example 2.2.1. $n = 14$, $d = 6$, $q = 2$. Recall from before that $A_2(n, d) = A_2(n-1, d-1)$ if d is even. We will again use this fact to find the best possible bound that this theorem can provide.

For $w = 0$, we get $A_2(13, 5) \leq \frac{.5 \cdot 13 \cdot 5}{.5 \cdot 13 \cdot 5} \cdot \frac{2^{13}}{V_q(13,0)} = 2^{13} = 8192$; similarly, $A_2(14, 6) = 2^{14} = 16384$.

For $w = 1$, we get $A_2(13, 5) \leq \frac{.5 \cdot 13 \cdot 5}{1 - 2 \cdot .5 \cdot 13 + .5 \cdot 13 \cdot 5} \cdot \frac{2^{13}}{V_q(13,1)} = \frac{32.5}{1 - 13 + 32.5} \cdot \frac{2^{13}}{1+13} = 927$; similarly $A_2(14, 6) \leq \frac{.5 \cdot 14 \cdot 6}{1 - 2 \cdot .5 \cdot 14 + .5 \cdot 14 \cdot 6} \cdot \frac{2^{14}}{V_q(14,1)} = \frac{42}{1 - 14 + 42} \cdot \frac{2^{14}}{1+14} = 1581$;

3 Asymptotic Bounds

We now wish to consider these $A_q(n, d)$ values as n goes to infinity. To do so, we must more formally define two terms.

In class, we have considered the *rate* of a linear code, k/n, as one measure of the goodness of a code. That is, the rate tells us how much information relative to redundancy that our codewords provide. The concept of rate can be generalized to non-linear codes as well. For a possibly nonlinear code over $\mathbb{F}_q$ with M codewords, the *rate* is defined to be $n^{-1} \log_q M$. Notice that for an $[n, k, d]$ linear code, $M = q^k$ and hence the rate is k/n as we expect.

A second notion of goodness that we have discussed, but I think not formally defined, is the *relative distance* of a code. For a linear or nonlinear code of length n has minimum distance d, this value is the ratio d/n.

Consequently, for our asymptotic bounds, we are interested in the largest possible rate for a family of codes over $\mathbb{F}_q$ of lengths going to infinity with a relative distance of some constant δ. In other words, we consider the equation:

$$\alpha_q(\delta) = \limsup_{n \to \infty} n^{-1} \log_q A_q(n, \delta n) \tag{10}$$

To better understand what is going on here, consider the case of an $[n, k, d]$ linear code $\mathcal{C}$. Before, we wanted to know how large we could get k given n and

d. Now, we want to know how large we can get $\frac{k}{n}$ given $\frac{d}{n} = \delta$ as $n \to \infty$. This idea generalizes to non-linear codes as in the non-asymptotic cases we discussed in class.

For the rest of this section, we will consider some bounds on $A_q(n, \delta n)$

3.1 Asymptotic Singleton Bound

Recall the Singleton Bound from lecture:

Theorem 3.1.1. *For $d \leq n$, $A_q(n, d) \leq q^{n-d+1}$.*

The asymptotic bound follows almost directly from the non-asymptotic variation.

Theorem 3.1.2. *If $0 \leq \delta \leq 1$, then $\alpha_q(\delta) \leq 1 - \delta$.*

Proof.

$$\begin{aligned} \alpha_q(\delta) &= \limsup_{n\to\infty} n^{-1} \log_q A_q(n, \delta n) \\ &\leq \limsup_{n\to\infty} n^{-1} \log_q q^{n-\delta n+1} \\ &\leq \limsup_{n\to\infty} \frac{n - \delta n + 1}{n} \\ &\leq 1 - \delta \end{aligned} \tag{11}$$

□

Example 3.1.1.

3.2 Asymptotic Plotkin Bound

4 Lexicodes

We end our discussion on bounds with an interesting class of binary linear codes called lexicodes, short for lexicographic codes. Lexicodes are linear and meet the Gilbert bound; we will discuss the former and prove the latter. Note that for this discussion we are solely working in $\mathbb{F}_2$.

We construct the lexicode $\mathcal{L}$ of length n and minimum distance d using a basic greedy algorithm as follows. First, order all n-tuples in an array A in lexicographic order: $A[0] = 0 \cdots 000$, $A[1] = 0 \cdots 001$, $A[2]0 \cdots 010$, $A[3] = 0 \cdots 011$, $\ldots$, $A[2^n - 1] = 1 \cdots 111$. Put the first vector from A (always the $\mathbf{0}$ vector) in $\mathcal{L}$. Next, find the first vector $\mathbf{x}$ of weight d in the lexicographic ordering, and put it in $\mathcal{L}$. Then, find the next vector in the lexicographic ordering in A whose distance from each vector in $\mathcal{L}$ is d or more and add it to $\mathcal{L}$. Repeat this process until you have scanned through all of A once.

The set $\mathcal{L}$ is a linear code of length n and minimum distance d. The set $\mathcal{L}$ also contains linear subcodes which are generated by the above algorithm, but stopping at the right spots.

Theorem 4.0.1. *After constructing $\mathcal{L}$ as above, label the vectors $\mathbf{c}_0$, $\mathbf{c}_1$, ... in the order they were selected such that $\mathbf{c}_0$ is the $\mathbf{0}$ vector.*

1. *$\mathcal{L}$ is a linear code and the vectors $\mathbf{c}_{2^i}$ are a basis of $\mathcal{L}$.*
2. *After $\mathbf{c}_{2^i}$ is chosen, the next $2^i - 1$ vectors can be rewritten as $\mathbf{c}_1 + \mathbf{c}_{2^i}$, $\mathbf{c}_2 + \mathbf{c}_{2^i}$, ..., $\mathbf{c}_{2^i-1} + \mathbf{c}_{2^i}$.*
3. *Let $\mathcal{L}_i = \{\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_{2^i-1}\}$. Then $\mathcal{L}_i$ is an $[n, i, d]$ linear code.*

Though I won't prove this, it is not difficult to see that 1. and 3. follow from 2. In 2., we are showing that all codewords in the lexicode are simply linear combinations of the c_{2^i} codewords, hence making the c_{2^i} codewords a basis for the code.

The codes $\mathcal{L}_i$ satisfy the inclusions $\mathcal{L}_1 \subset \mathcal{L}_2 \subset \cdots \subset \mathcal{L}_k = \mathcal{L}$, where k is the dimension of $\mathcal{L}$. In general, this dimension value k is not known until you have completed the construction as described above.

Example 4.0.1. Consider the codes $\mathcal{L}_i$ of length 5 and minimum distance 2. First, we must construct the entire lexicode $\mathcal{L}$. To do so, we enumerate all 32 vectors of $\mathbb{F}_2^5$ in lexicographic order: 00000,00001,00010,...,11111. We find that $\mathcal{L} = \{00000, 00011, 00101, 00110, 01001, 01010, 01100, 01111, 10001, 10010, 10100, 10111, 11000, 11011, 11101, 11110\}$. As a sanity check, note that $|\mathcal{L}| = 16$, a power of q. Consequently, we have $\mathcal{L}_1 = \{00000, 00011\}$, $\mathcal{L}_2 = \{00000, 00011, 00101, 00110\}$, $\mathcal{L}_3 = \{00000, 00011, 00101, 00110, 01001, 01010, 01100, 01111\}$, and $\mathcal{L}_4 = \mathcal{L}$. We can check to make sure that each $\mathbf{c}_j$ is a linear combination of the $\mathbf{c}_{2^i}$s. For this example, consider $\mathcal{L}_3$:

Notice that this is simply the even weight vectors. In fact, it is true that for any q. THIS IS THE EVEN WEIGHT CODE! prove it.

1. $\mathbf{c}_0$ is the trivial linear combination.
2. $\mathbf{c}_1 = \mathbf{c}_1$
3. $\mathbf{c}_2 = \mathbf{c}_2$
4. $\mathbf{c}_3 = \mathbf{c}_1 + \mathbf{c}_2$. $00110 = 00011 + 00101$.
5. $\mathbf{c}_4 = 1 \cdot \mathbf{c}_4$
6. $\mathbf{c}_5 = \mathbf{c}_1 + \mathbf{c}_4$. $01010 = 00011 + 01001$.
7. $\mathbf{c}_6 = \mathbf{c}_2 + \mathbf{c}_4$. $011000 = 00101 + 01001$.
8. $\mathbf{c}_7 = \mathbf{c}_3 + \mathbf{c}_4 = \mathbf{c}_1 + \mathbf{c}_2 + \mathbf{c}_4$. $01111 = 00011 + 00101 + 01001$.

The same analysis applies for all $\mathcal{L}_i$s.

Note that the greedy algorithm we descibed in the beginning of the section requires a strictly lexigraphical ordering, or else the code that is produced will not necessarily be linear. Again consider the case of $\mathbb{F}_2^5$, but instead of the lexigraphical ordering, shift every vector up one such that the ordering is identical to before, except now the zero vector is at the bottom of the list. If we run our algorithm on this ordering, we will clearly not choose the zero vector to be a part of $\mathcal{L}$ and hence $\mathcal{L}$ will not be linear.

Also, note that lexicodes meet one of the bound we discussed in lecture, the Gilbert bound. Recall the Gilbert bound:

$$B_q(n,d) \geq \frac{q^n}{\sum_{i=0}^{d-1} \binom{n}{i}(q-1)^i} \tag{12}$$

Claim 4.0.1. Lexicodes meet the Gilbert bound.

Proof. Consider an arbitrary lexicode $\mathcal{L}$. Recall our construction of $\mathcal{L}$ with vectors in $\mathbb{F}_q^n$, wherein we visit each vector in our ordering, and if it is distance d or more to every current element in $\mathcal{L}$ then we add it to our set. At the same time, notice that when a vector is visited in our algorithm, if it has distance $d-1$ or less to any vector in $\mathcal{L}$, then it is not placed in $\mathcal{L}$. Since all vectors in $\mathbb{F}_q^n$ are visited in our algorithm, any $v \in \mathbb{F}_q^n$ must either be in $\mathcal{L}$ or must have distance $d-1$ or less to some vector in $\mathcal{L}$. Hence the covering radius of $\mathcal{L}$ is $d-1$.

Consequently, it is enough to show that any code $\mathcal{C}$ with covering radius $d-1$ or less meets the Gilbert bound. Recall that the Gilbert bound can be thought of as a restatement of the following fact: (number of spheres) $\cdot$ (number of vectors in a sphere with radius $d-1$) $\geq$ (number of vectors in $\mathbb{F}_q^n$). Since our covering radius is $d-1$ or less, and we have that all codewords are distance d or more apart, consequently no sphere of radius $d-1$ in $\mathbb{F}_q^n$ can contain more than one codeword. Our spheres will still overlap, however, meaning (number of vectors in $\mathbb{F}_q^n$)/(number of vectors in a sphere with radius $d-1$) will certainly be greater than or equal to the number of spheres around codewords. Hence, $\mathcal{C}$ meets the Gilbert bound. □

Some notable codes we have discussed in class are in fact lexicodes. [REFERENCE]

We can prove a basic property of binary Lexicodes. First, though, we need an important definition.

Definition 4.0.1. The **nim − sum** NS(v) of a vector $v \in \mathbb{F}_2^n$ is the sum of it's coordinates in GF(2).

Definition 4.0.2. A vector $v \in \mathbb{F}_2^n$ is **zero − sum** if NS(v)=0.

We have used the idea of nim-sums and zero-sums throughout the course, but instead we have referred to this notion in terms of being even-like or odd-like. This is the exact same notion, only in different terms.

Claim 4.0.2. For $d = q = 2$, $\forall v \in \mathcal{L}$, $\mathrm{NS}(v) = 0$, and hence $\mathcal{L}$ is an even-like code.

Proof. We will prove this by inducting on n. In the base case of $n = 1$, $\mathcal{L} = \{0\}$, which is trivially zero-sum. Assume that for the lexicode of length n, which we will denote $\mathcal{L}(n)$, all codewords are zero-sum. Now, we must show that all the codewords in $\mathcal{L}(n+1)$ are zero-sum as well. First, note that all codewords in $\mathcal{L}(n)$ are also in $\mathcal{L}(n+1)$, except with an extra leading zero. By our assumption, these codewords are still zero-sum, as adding a zero in front will not change the nim-sum calculation. Now, we must show that any vector we can add to this set as prescribed by our algorithm must necessarily be a zero-sum vector as well.

Since $\mathcal{L}(n)$ is contained in $\mathcal{L}(n+1)$, we begin our construction of $\mathcal{L}(n+1)$ with the elements of $\mathcal{L}(n)$. Any new elements we will add must have a 1 in the leading coordinate as we have maxed out our possibilities for $\mathcal{L}(n)$, and so lexicographically this makes sense as the new vectors we now need to consider will be 'larger' than the ones we have already added in to $\mathcal{L}(n+1)$.

-If you approach an odd-weight vector that you can add, then there is even-weight vector that comes one before it that you should have added instead.

-SHIT! if it is odd weight, then by definition that means that has a 1 in the leading column plus an even weight vector from $\mathcal{L}(n)$!!!! That vector must necessarily be distance 1 away from a vector in $\mathcal{L}(n)$, and thus cannot be added to $\mathcal{L}(n+1)$. All even weight vectors will have a 1 in the leading coordinate and then an odd number of 1s in the next n coordinates. No odd numbers are in $\mathcal{L}(n)$, and so you will be at least distance one from all vectors already from $\mathcal{L}(n)$. So you get one component of your distance from the leading coordinate and at least one component of distance from you the last n coordinates. This completes the proof! □

For example, the binary lexicodes with d=3 and $n = 2^m - 1$ are the binary hamming codes. It can be quite tedious to compute and confirm something like this by hand, so I wrote a short C++ program that implements the Lexicode algorithm as described above.

Example 4.0.2. Let $n = 7$, $d = 3$. We find $\mathcal{L} = \{0000000, 000111, \}$

Need to talk about paper http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1057187

Asymptotic singleton, include 2 examples Asymptotic plotkin, some examples draw graph Asymptotic hamming define hilbert, prove 135,136, define vq(n,a), black box 2.10.3, prove 2.10.5

Table of commona$_q(n, d)$ values Recall basic theory: thm 2.1.2, 2.1.6

Sections

Elias talk about importance define it prove two lemmas that do the heavy lifting easy proof to finish it off work out some examples

Linear Programming Bound talk about importance define it prove it work out some examples

Note existence of two lower bounds, say we talked about them in class