Today we will talk about:

- Hamming-Elias-Bassalygo/Johnson Bound
- MacWilliams Identities
- Linear Programming Bound

1 Review Hamming-Elias-Bassalygo/Johnson Bound

In the past several lectures, we were bounding the rate of a code, given its relative distance. Previous to last lecture, the picture of the upper bounds involved two incomparable bounds: the Hamming bound was stronger for smaller δ and the Plotkin bound was stronger for larger δ. At the end of last class, we were introduced to the Hamming-Elias-Bassalygo bound that unifies the two techniques and thus is always stronger, though it is very close to the Hamming bound for small δ.

At the end of last lecture, we saw that the HEB bound required that the rate $R \leq 1 - H(\tau)$, where τ comes from the Johnson bound.

Theorem 1 (Johnson bound [1]) *Every $(n, k, \delta n)_2$ code is also a $(\tau n - 1, n)$-error-correcting code for $\tau = \frac{1}{2} \cdot (1 - \sqrt{1 - 2\delta})$.*

Proof As usual we turn the problem into a geometric one by embedding in Euclidean space. Let $\mathbf{c}_1, \ldots, \mathbf{c}_m$ be codewords of a code of minimum distance $d = \delta n$ that are within a Hamming ball of radius $t = \tau n - 1$ from a received vector $\mathbf{y}$. We wish to show $m \leq O(n)$.

We will embed the vectors into Euclidean space, scaling them by a factor of $1/\sqrt{n}$ to get vectors of unit norm. Let $0 \to \frac{1}{\sqrt{n}}$, $1 \to \frac{-1}{\sqrt{n}}$, and $c_i \to \mathbf{x}_i$. Then all of our vectors $\mathbf{x}_i \in \{1/\sqrt{n}, -1/\sqrt{n}\}^n \in \mathbb{R}^n$. Observe that our inner products evaluate to: (1) $\langle \mathbf{x}, \mathbf{x} \rangle = 1$. (2) $\langle \mathbf{x}_i, \mathbf{x}_j \rangle \leq 1 - 2\delta$, if $i \neq j$. (3) $\langle \mathbf{y}, \mathbf{x}_i \rangle > 1 - 2\tau$. In other words, we have a collection of unit vectors $\mathbf{x}_i$ whose pairwise inner product is small, but which have a common, large inner product with $\mathbf{y}$. We now wish to restrain m by reasoning that not more than n vectors can subtend an angle of $\pi/2$ or more in n-dimensions. To do so, we will just shift our origin to a new vector $\mathbf{v}$ so that from this vector, the vectors $\mathbf{x}_i$ mutually subtend an angle of at least $\pi/2$. And how do we find such a vector $\mathbf{v}$? Well the most natural idea is to shift the origin closer to the scene of action — namely, towards $\mathbf{y}$. Specifically, we will move to some point $\alpha\mathbf{y}$ and inspect our world from there. The following claim asserts that we will see what we hope to see.

Claim 2 *There exists an α such that for every $i \neq j \in [m]$, $\langle \mathbf{x}_i - \alpha\mathbf{y}, \mathbf{x}_j - \alpha\mathbf{y} \rangle \geq 0$, while for every i, $\langle \mathbf{x}_i - \alpha\mathbf{y}, \mathbf{y} - \alpha\mathbf{y} \rangle > 0$.*

Proof We will not specify α yet only that it will lie in the interval $0 \leq \alpha < 1$. For such α, note that

$$\langle \mathbf{x}_i - \alpha\mathbf{y}, \mathbf{x}_j - \alpha\mathbf{y} \rangle \leq 1 - 2\delta - 2\alpha(1 - 2\tau) + \alpha^2 = (1 - \alpha)^2 + 4\alpha\tau - 2\delta.$$

The right-hand side is minimized at $\alpha = 1 - 2\tau$. For this setting, the RHS above equals $4\tau - 4\tau^2 - 2\delta$. Recall we set $\tau = \frac{1}{2}(1 - \sqrt{1 - 2\delta})$, and so we have $(1 - 2\tau)^2 = 1 - 2\delta$, which in turn implies $4\tau - 4\tau^2 - 2\delta = 0$. We conclude that for this setting $\langle \mathbf{x}_i - \alpha\mathbf{y}, \mathbf{x}_j - \alpha\mathbf{y} \rangle \geq 0$, as desired.

To conclude, we note that for the same setting $\alpha = 1 - 2\tau$, we have

$$\langle \mathbf{x}_i - \alpha\mathbf{y}, (1-\alpha)\mathbf{y}\rangle > (1-\alpha)(1-2\tau) - (\alpha)(1-\alpha) = 0,$$

which yields the other part of the claim. ■

We are now in a position to apply a lemma shown in previous lectures:

Lemma 3 *If $\mathbf{y}, \mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{R}^n$ satisfy $\langle \mathbf{x}_i, \mathbf{x}_j\rangle \le 0$ for distinict i, j, while $\langle \mathbf{y}, \mathbf{x}_i\rangle \ge 0$, then $m \le n$.*

to conclude that $m \le n$. This concludes the proof of the theorem. ■

On more fact is needed to show that the Johnson bound is tight.

Proposition 4 *Suppose a $(n,k,d)_2$ code C is a (t,ℓ)-error-correcting code. Then $2^k \mathrm{Vol}(t,n) \le \ell \cdot 2^n$.*

Proof The proposition follows easily. If we consider the balls of radius t around the codewords, then any word in $\{0,1\}^n$ is considered at most ℓ times. Thus the sum of the volumes of these balls is at most $\ell \cdot 2^n$. ■

Combining Proposition 4 with Theorem 1 gives us the Elias-Bassalygo upper bound on the rate of a family of codes with relative distance δ.

Theorem 5 (Elias-Bassalygo bound [2],[3]) *If $\mathcal{C}$ is an infinite family of binary codes with rate R and relative distance δ, then $R \le 1 - H(\frac{1}{2} \cdot (1 - \sqrt{1-2\delta}))$.*

Proof The theorem follows essentially from Proposition 4 and Theorem 1. The missing ingredients are dry exercises showing that one can pick n large enough so as to overcome all problems posed by identities which hold only asymptotically. (The volume of Hamming balls is not exactly related to the binary entropy function; the Johnson bound only lower bounds the (t,ℓ)-error-correcting radius of codes when ℓ is a growing function of n. And the bound only allows a list-decoding radius of $\tau n - 1$ and not τn.) We'll spare the reader the details. ■

Before concluding the section, let us digress briefly to understand when and why the Elias-Bassalygo bound is better. To do so, let us recall two of the previous bounds that we have worked with. The Hamming upper bound says $R \le 1 - H(\delta/2)$, while the GV lower bound says $R \ge 1 - H(\delta)$. The Elias-Bassalygo bound shows $R \le 1 - H(\tau(\delta))$, for $\tau(\delta) = \frac{1}{2} \cdot (1 - \sqrt{1-2\delta})$. First note that $\delta/2 \le \tau(\delta) \le \delta$ and H_2 is monotone decreasing, and so the Elias-Bassalygo bound is always between the Hamming bound and the GV bound. Further if $\delta > 0$, then $\tau(\delta) > \delta/2$ and so the Elias-Bassalygo bound is strictly better than the Hamming bound. However if δ is close to zero then $\delta/2$ is a very good approximation to $\tau(\delta)$ and so for small values of δ the Elias-Bassalygo bound is not much better than the Hamming bound. However for large values of δ, $\tau(\delta)$ starts to get closer to δ. In particular, if $\delta = \frac{1}{2} - \epsilon$, then $\tau(\delta) = \frac{1}{2} - \sqrt{\epsilon/2}$ and so τ approaches $\frac{1}{2}$ as δ approaches $\frac{1}{2}$. So as $\delta \to \frac{1}{2}$, the Elias-Bassalygo bound really starts to get better and approaches the GV bound!

What else could we hope for? While the Elias-Bassalygo bound gives us the right bound for $\delta = \frac{1}{2}$, it does not quite have the right growth around this point. In particular, the GV bound shows that one can find codes of rate $O(\epsilon^2)$ and relative distance $\frac{1}{2} - \epsilon$, as $\epsilon \to 0$. The Elias-Bassalygo bound only rules out codes of rate $\Omega(\epsilon)$ at this distance. Which bound is closer to the truth? Actually, it's the GV bound. To understand why, we must first discuss a magical result called MacWilliams Identities. The Linear Programming (LP) bound is the most important result of the MacWilliams Indentites (for our purposes), because it ends up showing the tightness of the GV bound.

2 MacWilliams Identities

Florence Jessie MacWilliams, one of the first published women in coding theory, proved in her PhD Thesis at Harvard that one can determine the weight distribution of a code from the weight distribution of its dual. This surprising result has many nice consequences, including the strongest known upper bound on codes, which we will see later. First, let us understand MacWilliams Identities.

2.1 Linear Code and Their Duals

Recall that a linear code C is specified by a $k \times n$ generator matrix $\mathbf{G}$. Alternatively, the code can be specified by a $n \times (n-k)$ parity check matrix $\mathbf{H}$. The matrix $\mathbf{G}$ consists of k linearly independent rows while $\mathbf{H}$ consists of $n-k$ linearly independent columns.

The dual of the linear code C, denoted $C^{\perp}$, is the the code *generated* by $\mathbf{H}^T$, the transpose of the parity check matrix of C. It is obvious that $C^{\perp}$ is also a linear code, with block length n and message length $n-k$. Furthermore every pair of codewords $\mathbf{b} \in C$ and $\mathbf{c} \in C^{\perp}$ satisfy $\langle \mathbf{b}, \mathbf{c} \rangle = 0$. A slightly stronger fact is proven in the proposition below.

Proposition 6 *For linear code $C \subseteq \mathbb{F}_q^n$ and vector $\mathbf{x} \in \mathbb{F}_q^n$, the following hold:*

- *If $\mathbf{x} \in C^{\perp}$, then for every $\mathbf{c} \in C$, $\langle \mathbf{c}, \mathbf{x} \rangle = 0$.*
- *If $\mathbf{x} \notin C^{\perp}$, then for every $\alpha, \beta \in \mathbb{F}_q$, the sets $\{\mathbf{c} \in C \mid \langle \mathbf{c}, \mathbf{x} \rangle = \alpha\}$ and $\{\mathbf{c} \in C \mid \langle \mathbf{c}, \mathbf{x} \rangle = \beta\}$ have the same cardinality.*

Proof The first part of the proposition follows from the definition of the dual of a code. For the second part note first that without loss of generality, we may assume $\beta = 0$ and $\alpha \neq 0$. Let $S_\alpha = \{\mathbf{c} \in C \mid \langle \mathbf{c}, \mathbf{x} \rangle = \alpha\}$, and $S_0 = \{\mathbf{c} \in C \mid \langle \mathbf{c}, \mathbf{x} \rangle = 0\}$. Note that $\mathbf{0} \in S_0$ and hence the latter is non-empty. We note next that the former is also non-empty. Since $\mathbf{x} \notin C^{\perp}$ we have that there exists $\mathbf{c} \in C$ such that $\langle \mathbf{c}, \mathbf{x} \rangle = \alpha' \neq 0$. Then we have that $\mathbf{b} = (\alpha')^{-1}\alpha\mathbf{c}$ is an element of C with $\langle \mathbf{b}, \mathbf{x} \rangle = \alpha$, and thus $\mathbf{b} \in S_\alpha$.

For a set $A \subseteq \mathbb{F}_q^n$ and vector $\mathbf{y} \in \mathbb{F}_q^n$, let $\mathbf{y} + A$ denote the set $\{\mathbf{y} + \mathbf{x} \mid \mathbf{x} \in A\}$. Fix $\mathbf{b} \in S_\alpha$. We get $|S_\alpha| = |S_0|$ from the following series of facts:

1. $\mathbf{b} + S_0 \subseteq S_\alpha$: If $\mathbf{a} \in S_0$ then $\langle \mathbf{b} + \mathbf{a}, \mathbf{x} \rangle = \langle \mathbf{b}, \mathbf{x} \rangle + \langle \mathbf{a}, \mathbf{x} \rangle = \alpha + 0 = \alpha$.
2. $|\mathbf{b} + S_0| = |S_0|$: Follows since $\mathbf{b} + \mathbf{a} = \mathbf{b} + \mathbf{a}'$ iff $\mathbf{a} = \mathbf{a}'$.
3. $|S_0| \leq |S_\alpha|$: Follows from Parts (1) and (2) above.
4. $|S_\alpha| \leq |S_0|$: Similar to Parts (1)-(3) above, except that here we work with the set $(-\mathbf{b}) + S_\alpha$.

■

2.2 Weight Distributions and the Weight Generating Function

Definition 7 (Weight distribution & generating function) *Given an $[n, k, d]_q$ code C, and index $i \in \{0, \ldots, n\}$, the ith weight enumerator of C is the number of codewords in C of Hamming weight exactly i. The weight distribution of C is the sequence $\langle A_0, \ldots, A_n \rangle$, where A_i is the ith weight enumerator of C. The weight generating function of C is the formal polynomial $A_C(x) = \sum_{i=0}^{n} A_i x^i$, where $\langle A_0, \ldots, A_n \rangle$ is the weight distribution of C.*

The MacWilliams Identities relate the weight distribution of a code C with that of the code $C^\perp$. We will do so by studying the *weight generating function*, $A_C(y) = \sum_{i=0} A_i y^i$, of the code C, and relating A_C to $A_{C^\perp}$. As is by now usual, we will prove the identity first for the binary case, and then move to the q-ary case later.

2.3 The Extended Generating Function

We will start by defining an elaborate generating function of a code C, called its *extended generating function*. It will be quite evident that this generating function preserves all information about the code C (unlike the weight generating function which does not tell us exactly what the code is). We will then relate this elaborate generating function of the code to that of its dual.

We will introduce the extended generating function gently. We will start by defining the extended generating function of a single bit, and then define it for a word in $\{0,1\}^n$ and finally define the extended generating function of a code.

Definition 8 (Extended generating function of a bit) *The weight generating function of C is the formal polynomial $A(x) = \sum_{i=0}^n A_i x^i$. For a bit $b \in \{0,1\}$, the extended generating function $W_b(x,y)$, is a polynomial in two variables x and y defined as $W_b(x,y) = (1-b)x + by$, or in other words:*

$$\begin{aligned} W_b(x,y) &= x \quad \text{if } b = 0 \\ &= y \quad \text{if } b = 1 \end{aligned}$$

Definition 9 (Extended generating function of a word) *For a vector $\mathbf{b} = \langle b_1, \ldots, b_n \rangle \in \{0,1\}^n$, the extended generating function $W_{\mathbf{b}}(\mathbf{x},\mathbf{y})$, is a polynomial in $2n$ variables, $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$ and $\mathbf{y} = \langle y_1, \ldots, y_n \rangle$, defined as: $W_{\mathbf{b}}(\mathbf{x},\mathbf{x}) = \prod_{i=1}^n W_{b_i}(x_i, y_i)$.*

Finally we define the extended generating function of a code C.

Definition 10 (Extended generating function of a code) *For a code $C \subseteq \{0,1\}^n$, the extended generating function $W_C(\mathbf{x},\mathbf{y})$, is a polynomial in $2n$ variables, $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$ and $\mathbf{y} = \langle y_1, \ldots, y_n \rangle$, defined as: $W_C(\mathbf{x},\mathbf{y}) = \sum_{\mathbf{b} \in C} W_{\mathbf{b}}(\mathbf{x},\mathbf{y})$.*

To make sense of the definitions above, it would help to see an example. Consider the code $C = \{000, 101, 110, 011\}$. The extended generating function for this code is:

$$W_C(\mathbf{x},\mathbf{y}) = x_1x_2x_3 + y_1x_2y_3 + y_1y_2x_3 + y_1y_2x_3 + x_1y_2y_3.$$

It should be immediately clear that the extended generating function carries all the information about a code and does not do so in any especially clever way. The extended generating function is not intended to be a way of compressing information about the code, but rather an elaborate way of representing the code, which will come in useful in studying the combinatorics of codes. At the outset we are not hoping to use them to glean any information about codes in a computationally efficient manner.

Given this elaborate representation of a code, it should not come as a surprise that given the extended generating function of a code C we can derive the extended generating function of its dual. After all. all the information of C is embedded into $W_C(\mathbf{x},\mathbf{y})$, so C can be derived from W_C, $C^\perp$ can be derived from C, and finally $W_{C^\perp}$ can be derived from $C^\perp$. The generalized MacWilliams Identity just gives an explicit form of this rather obvious statement. The strength of the result lies in the fact that the relationship takes an especially simple closed form - one that will turn out to be especially amenable to manipulations later on.

The crux of the identity is some sort of a "Discrete Fourier Transform" (or Hadamard transform or Walsh transform, depending on your loyalties). Instead of focusing on the function $W_C(\mathbf{x}, \mathbf{y})$ we will focus on the function $W_C(\mathbf{x}+\mathbf{y}, \mathbf{x}-\mathbf{y})$. Before saying what this transform does to the generating function of a code, we will describe what the transform does to the generating function of a single vector $\mathbf{b}$.

Lemma 11

$$W_{\mathbf{b}}(\mathbf{x}+\mathbf{y}, \mathbf{x}-\mathbf{y}) = \sum_{\mathbf{c} \in \{0,1\}^n} (-1)^{\langle \mathbf{b}, \mathbf{c} \rangle} W_{\mathbf{c}}(\mathbf{x}, \mathbf{y}).$$

Proof Note by definition of $W_{\mathbf{b}}$ that

$$W_{\mathbf{b}}(\mathbf{x}+\mathbf{y}, \mathbf{x}-\mathbf{y}) = \prod_{i=1}^{n} W_{b_i}(x_i + y_i, x_i - y_i) = \prod_{i=1}^{n} \left(x_i + (-1)^{b_i} y_i\right).$$

Expanding the right hand side above we get a sum of 2^n terms of the form $\pm z_1 \cdots z_n$, where $z_i \in \{x_i, y_i\}$ and the sign is negative iff there is an odd number of indices i where $z_i = y_i$ and $b_i = 1$. Letting $\mathbf{c} \in \{0,1\}^n$ denote the index of this sum of 2^n terms, and setting $z_i = y_i$ if $c_i = 1$, we see that $z_i = W_{c_i}(x_i, y_i)$ and the sign in front is negative iff $\langle \mathbf{b}, \mathbf{c} \rangle = 1$. Thus we have

$$\prod_{i=1}^{n} \left(x_i + (-1)^{b_i} y_i\right) = \sum_{\mathbf{c} \in \{0,1\}^n} (-1)^{\langle \mathbf{b}, \mathbf{c} \rangle} \prod_{i=1}^{n} W_{c_i}(x_i, y_i) = \sum_{\mathbf{c} \in \{0,1\}^n} (-1)^{\langle \mathbf{b}, \mathbf{c} \rangle} W_{\mathbf{c}}(\mathbf{x}, \mathbf{y}).$$

This yields the lemma. ■

Theorem 12 (Generalized MacWilliams Identity [4]) *The extended generating function of a code C and its dual $C^{\perp}$ satify the identity:*

$$W_{C^{\perp}}(\mathbf{x}, \mathbf{y}) = \frac{1}{|C|} W_C(\mathbf{x}+\mathbf{y}, \mathbf{x}-\mathbf{y}).$$

Proof The proof follows easily from Proposition 6 and Lemma 11. Note that

$$\begin{aligned}
W_C(\mathbf{x}+\mathbf{y}, \mathbf{x}-\mathbf{y}) &= \sum_{\mathbf{b} \in C} W_{\mathbf{b}}(\mathbf{x}+\mathbf{y}, \mathbf{x}-\mathbf{y}) \\
&= \sum_{\mathbf{b} \in C} \sum_{\mathbf{c} \in \{0,1\}^n} (-1)^{\langle \mathbf{b}, \mathbf{c} \rangle} W_{\mathbf{c}}(\mathbf{x}, \mathbf{y}) \quad \text{(By Lemma 11)} \\
&= \sum_{\mathbf{b} \in C} \sum_{\mathbf{c} \in C^{\perp}} W_{\mathbf{c}}(\mathbf{x}, \mathbf{y}) + \sum_{\mathbf{b} \in C} \sum_{\mathbf{c} \notin C^{\perp}} (-1)^{\langle \mathbf{b}, \mathbf{c} \rangle} W_{\mathbf{c}}(\mathbf{x}, \mathbf{y}) \\
&= |C| W_C(\mathbf{x}, \mathbf{y}) + 0,
\end{aligned}$$

where the first part of the last equation is by definition of the extended generating function and the second part applies Proposition 6 to every $\mathbf{c} \notin C^{\perp}$. The theorem follows. ■

2.4 The MacWilliams Identities

We now return to the goal of studying the weight distribution of a code C. The following, simple proposition shows that we have made some progress already!

Proposition 13 *For every linear code C, we have $x^n A_C(y/x) = W_C(x,\ldots,x,y,\ldots,y)$.*

Proof The proposal follows easily by inspecting the right hand side. Under the substitution $x_i = x$ and $y_i = y$, we have

$$
\begin{aligned}
W_C(x,\ldots,x,y,\ldots,y) &= \sum_{\mathbf{b}\in C} x^{n-\mathrm{wt}(\mathbf{b})} y^{\mathrm{wt}(\mathbf{b})} \\
&= x^n \sum_{\mathbf{b}\in C} (y/x)^{\mathrm{wt}(\mathbf{b})} \\
&= x^n \sum_{i=0}^{n} \sum_{\mathbf{b}\in C \ | \ \mathrm{wt}(\mathbf{b})=i} (y/x)^i \\
&= x^n \sum_{i=0}^{n} A_i (y/x)^i \\
&= x^n A_C(y/x).
\end{aligned}
$$

■

The MacWilliams Identity now follows easily:

Theorem 14 (MacWilliams Identity [4])

$$A_{C^\perp}(y) = \frac{1}{|C|}(1+y)^n A_C\left(\frac{1-y}{1+y}\right).$$

Proof The proposition we just proved, Proposition 13, tells us that $A_{C^\perp}(y) = 1^n W_{C^\perp}(1,\ldots,1,y,\ldots,y)$. Now, applying the generalized MacWilliams identity (Theorem 12), we get

$$W_{C^\perp}(1,\ldots,1,y,\ldots,y) = \frac{1}{|C|} W_C(1+y,\ldots,1+y,1-y,\ldots,1-y).$$

Finally applying Proposition 13 again, we have

$$W_C(1+y,\ldots,1+y,1-y,\ldots,1-y) = (1+y)^n A_C((1-y)/(1+y)).$$

Putting the above together, we get the theorem. ■

We briefly state the q-ary version (not covered in class) so that we can discuss the more general result and its implications.

Theorem 15 (q-ary MacWilliams Identity) *For a q-ary linear code C of block length n, we have*

$$A_{C^\perp}(y) = \frac{(1+(q-1)y)^n}{|C|} A_C\left(\frac{1-y}{1+(q-1)y}\right).$$

The MacWilliams identity shows that the weight distribution of a linear code can be computed from the weight distribution of its dual! Note, this is totally non-trivial - we can see no obvious reason why specifying the weight distribution of C, should fix the weight distribution of $C^\perp$. In a sense the identity suggests that any two linear codes of a given weight distribution are essentially the same, motivating the following question (the answer to which is not obvious to Madhu).

Question: **True or False:** For any pair of q-ary codes C_1 and C_2 of block length n that have the same weight distribution, there exists a permutation $\pi : [n] \to [n]$ such that $\mathbf{c} = \langle c_1, \ldots, c_n \rangle \in C_1$ iff such that $\mathbf{c}' = \langle c_{\pi(i)}, \ldots, c_{\pi(n)} \rangle \in C_2$.

We now examine the exact form of the relationship obtained between the weight distribution of a code and its dual. As in the case of the relationship between the extended (elaborate?) generating functions of a code and its dual, the exact form of the relationship turns out to be simple and quite useful. In particular, the weight distribution of the dual code is just a linear function of the weight distribution of the primal code, as we note below.

Corollary 16 *For every q and n there exists an $(n+1) \times (n+1)$ rational matrix $\mathbf{M}$ such that the following holds: Let $\mathbf{a} = \langle A_0, \ldots, A_n \rangle$ be the weight distribution of a q-ary linear code C of block length n, and let $\mathbf{b} = \langle B_0, \ldots, B_n \rangle$ be the weight distribution of $C^\perp$. Then*

$$\mathbf{b} = \frac{1}{A_0 + \cdots + A_n} \mathbf{aM}.$$

Furthermore, the matrix $\mathbf{M} = \{m_{ij}\}$ is given by $m_{ij} = \sum_{l=0}^{i} \binom{n-j}{i-\ell} \binom{j}{l} (-1)^\ell (q-1)^{i-\ell}$.

Proof Since B_i is the coefficient of y^i in $A_{C^\perp}(y)$, we need to examine the coefficient of y^i in $\frac{(1+(q-1)y)^n}{|C|} A_C \left(\frac{1-y}{1+(q-1)y} \right)$. Write this quantity as $\frac{1}{|C|} \sum j = 0^n A_j (1 + (q-1)y)^{n-j} (1-y)^j$. The coefficient of y^i in this sum is obtained as the coefficient of y^ℓ in the expansion of $(1-y)^j$ times the coefficient of $y^{i-\ell}$ in the expansion of $(1+(q-1)y)^{n-j}$, summed up over $\ell \in [i]$. We thus get:

$$\begin{aligned} B_i &= \frac{1}{|C|} \sum_{j=0}^{n} A_j \sum_{\ell=0}^{i} \binom{n-j}{i-\ell} \binom{j}{\ell} (q-1)^{i-\ell} (-1)^\ell \\ &= \frac{1}{|C|} \sum_{j=0}^{n} A_j m_{ij} \end{aligned}$$

The corollary follows from the fact that $|C| = A_0 + \cdots + A_n$. ■

3 The Linear Programming Bound

We now describe the most powerful application of the MacWilliams Identities, namely the Linear Programming (LP) bound on the rate of a code.

Let C be a $[n, ?, d]_q$ code with weight distribution $A_0, A_1, \ldots, A_n$. The number of codewords for the code is $\sum_i A_i$ and hence the rate $R = \log_q(\sum A_i)/n$. As mentioned earlier our goal is to get an asymptotic upper bound on the rate R. This reduces to deriving an upper bound on $\sum A_i$ subject to the restrictions that

$$A_0 = 1, A_1 = 0, A_2 = 0, \ldots, A_{d-1} = 0$$

$$B_0 = 1, B_1, B_2, \ldots, B_n \geq 0$$

As noted earlier, B_i's are essentially linear functions of A_i's. Precisely, $B_i \cdot \left(\sum_{j=0}^{n} A_j \right)$ is given by a linear function of the A_i's.

This motivates the following linear program:

$$\begin{array}{rl}\text{Maximize} & K_n = \sum_{i=0}^{n} A_i \\ \text{subject to} & \\ & \mathbf{aM} \geq \mathbf{0} \\ & A_0 = 1 \\ & A_1 = \cdots = A_{d-1} = 0 \\ & A_d, \ldots, A_n \geq 0 \end{array}$$

where $\mathbf{a} = \langle A_0, \ldots, A_n \rangle$, $\mathbf{M}$ is the matrix described in Corollary 16, and the notation $\mathbf{x} \geq \mathbf{0}$, implies every coordinate of $\mathbf{x}$ is non-negative.

The linear program above bounds the number of codewords that any linear code of distance d can have over a q-ary alphabet, and one can hope that linear programming and duality could help anaylze the quantity K_n above. The quantity $R_{\text{LP}} \stackrel{\text{def}}{=} \log_q(K_n)/n$ is called the LP (upper) bound on the rate of a linear code. Somewhat surprisingly, even though the linear program was motivated only for linear codes, the LP bound holds also for non-linear codes, as implied by the following theorem.

Theorem 17 *If C is an $(n, k, d)_q$ code with $A_i \stackrel{\text{def}}{=} \frac{1}{|C|} \sum_{\mathbf{c} \in C} |\{\mathbf{b} \in C \mid \Delta(\mathbf{b}, \mathbf{c}) = i\}|$. Let $\mathbf{a} = \langle A_0, \ldots, A_n \rangle$ and let $\mathbf{M}$ be as in Corollary 16. Then $\mathbf{aM} \geq \mathbf{0}$.*

Thus the quantity R_{LP} is the LP bound on the rate of *all* codes. The LP upper bound was discovered by Delsarte [5]who showed how to derive several known upper bounds using this framework. Performing a tight asymptotic analysis of the LP bound is non-trivial and took a while after the work of Delsarte. The best asymptotic analysis of the LP bound is due to McEliece, Rodemich, Rumsey and Welch [6]who give two upper bounds on the LP bound. We state their bounds for the binary case:

Theorem 18 (MRRW bound [6]) *If $\mathcal{C}$ is a family of binary codes (not necessarily linear) of rate R and relative distance δ then*

$$R \leq H(\frac{1}{2} - \sqrt{\delta(1-\delta)}),$$

$$\text{and} \quad R \leq \min_{u \in [0, 1-2\delta]} \{1 + g(u^2) - g(u^2 + 2\delta u + 2\delta)\} \quad \text{where} \quad g(x) = H_2\left(\frac{1 - \sqrt{1-x}}{2}\right).$$

Proving these bounds is out of scope of this lecture! The interested reader is pointed to thesis of Samorodnitsky [7], or the article by Levenshtein [8] for further details.

4 References

1. Selmer M. Johnson. A new upper bound for error-correcting codes. *IEEE Transactions on Information Theory.* 9:198-205. 1963.

2. L.A. Bassalygo. New upper bounds for error-correcting codes. *Problems of Information Transmission.* 1(1):32-35. 1965.

3. Claude E. Shannon, Robert G. Gallager, and Elwyn R. Berlekamp. Lower bounds to error probability for coding on discrete memoryless channels. *Information and Control.* 10:65-103 (Part 1), 522-552 (Part 2). 1967.

4. Florence Jessie MacWilliams. A theorem on the distribution of weights in a systematic code. *Bell Systems Technical Journal.* 42:79-94. 1963.

5. Phillipe Delsarte. An algebraic approach to the association schemes of coding theory. *Philips Research Reports.* Supplement 10. 1973.

6. Robert J McEliece, Eugene R. Rodemich, Howard Rumsey Jr., and Lloyd R. Welch. New upper bounds of the rate of a code via the Delsarte-MacWilliams inequalities. *IEEE Transactions on Information Theory.* 23:157-166. 1977.

7. Alex Samorodnitsky. *PhD Thesis.* Hebrew University, 1998.

8. V.I. Levenshtein. *Universal Bounds for codes and designs.* Pages 499-648. 1998.