Implementing and Experimenting with methods for finding the Weight Enumerators of MDS Codes

Joseph Koutsoutis

Spring 2024

Introduction

Given a linear $[n,k]$-code C over GF(q), its weight enumerator is defined to be:

$$W_C(t) = \sum_{i=0}^{n} A_i t^i,$$

where A_i is number of codewords in C with weight i. Knowing the weight enumerator of a code is often useful for error-detection and error-correction analysis [1]. The famous MacWilliams identity gives a relationship between the weight enumerator of a code C and its dual $C^\perp$ [2]. Using this identity and brute force, the weight enumerator of any linear code can be found in $O(nq^{\min(k,n-k)})$ time by computing the Hamming distance of all codewords in C or $C^\perp$, whichever is smaller.

While this is the fastest method to compute weight enumerators in general, there are certain classes of codes which have convenient properties that allow their weight enumerators to be computed more easily. Here, we will focus on the weight enumerators of some maximum distance separable (MDS) codes, which have parameters $[n,k,n-k+1]$. Assmus, Gleason, Mattson and Turyn [3], Forney and Kohlenberg [4], and Kasami, Lin and Peterson [5] all independently determined that the coefficients of such codes for $i \in \{n-k+1, \dots, n\}$ are given by:

$$A_i = \binom{n}{i}(q-1)\sum_{j=0}^{i-d}(-1)^j\binom{i-1}{j}q^{i-j-d}.$$

In this article, we will first show that the algorithm in our accompanying Maple package for computing these weight enumerators works without relying on this known result. Afterwards, we will compute the mean and variance of the weight distributions of many MDS codes, and present some experimentally-supported conjectures about other standardized moments.

Computing Weight Enumerators

For prime powers $q = p^k$, we will specifically investigate linear codes C of length $n \leq q-1$ with distance $d \leq n$ whose parity check matrices are of the form:

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{d-2} & x_2^{d-2} & \cdots & x_n^{d-2} \end{bmatrix},$$

where $x_1, \ldots, x_n \in \mathrm{GF}(q)^\times$ are distinct. Let's denote this class of linear codes with fixed q,d as $\mathcal{C}_{q,d}$. Our goal in this section is to find a faster way to compute the weight enumerators of such $C \in \mathcal{C}_{q,d}$ with runtime faster than $O(nq^{\min(d-1,n-d+1)})$. We will present out algorithm after first proving one lemma.

Lemma 0.1. *Fix q,d,n. Assume that we know that every code $C \in \mathcal{C}_{q,d}$ with codewords of length n has the same number of codewords of weight n. Let this number be denoted as a. Then for any $C' \in \mathcal{C}_{q,d}$ with length n', the number of codewords of weight n in C is equal to $a\binom{n'}{n}$.*

Proof. If $n' \leq n$, this statement trivially holds, so assume $n' > n$. Let's say that the parity check matrix of C' has columns $1, \ldots, n'$ containing powers of $x_1, \ldots, x_{n'}$ respectively. Suppose $c_1 \cdots c_{n'} \in \mathrm{GF}(q)^{n'}$ has weight n. Let $\sigma : [n] \to [n']$ map $i \in [n]$ to the i-th smallest index of $c_1 \cdots c_{n'}$ such that $c_{\sigma(i)} \neq 0$. If $c_1 \cdots c_{n'} \in C'$, then we should satisfy:

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_{\sigma(1)} & x_{\sigma(2)} & \cdots & x_{\sigma(n)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{\sigma(1)}^{d-2} & x_{\sigma(2)}^{d-2} & \cdots & x_{\sigma(n)}^{d-2} \end{bmatrix} \begin{bmatrix} c_{\sigma(1)} \\ c_{\sigma(2)} \\ \vdots \\ c_{\sigma(n)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

We observe that there exists $C \in \mathcal{C}_{q,d}$ with codewords of length n having this parity check matrix. Therefore $c_1 \cdots c_{n'} \in C'$ if and only if $c_{\sigma(1)} \cdots c_{\sigma(n)} \in C$. By assumption, all such C have a of codewords of weight n, and since there are $\binom{n'}{n}$ ways to assign zeros so that $c_1 \cdots c_{n'}$ has weight n, it follows that there are $a\binom{n'}{n}$ codewords of weight n in C'. □

Now we prove our algorithm for computing weight enumerators.

Theorem 0.2. *The following algorithm is correct:*

Algorithm 1:

Data: *Parameters n,q,d as described above*
Result: $f = W_C(t)$
$A \leftarrow [];$
for $i \leftarrow 1$ **to** $n - d + 1$ **do**
  $A[i] \leftarrow q^i - 1;$
end
$f \leftarrow 1;$
for $k \leftarrow d + 1$ **to** $n + 1$ **do**
  $a \leftarrow A[k - d];$
  for $i \leftarrow 0$ **to** $n - k + 1$ **do**
    $A[i] \leftarrow A[i] - a\binom{i+k-1}{k-1};$
  end
  $f \leftarrow f + a\binom{n}{k-1}t^{k-1};$
end

Proof. Fix q,d. We will prove this algorithm by induction on n.

Case $n = d$: Since we have a linear code with distance $n = d$, every codeword except for the zero codeword will have weight d. We know that there are $q^{n-(d-1)} = q$ codewords, and hence there are $q - 1$ codewords of weight d. This is what we set a to in the second for loop, so we compute $f = (q - 1)t^d + 1$ as desired. Note that a is set to the same value regardless of the specific parity check matrix being considered.

Case $n > d$: Assume by induction that our algorithm works for input length $n - 1$ and that for each a value set in the second for loop from $k = d + 1$ to $k = n$, a is equal to the number of codewords of weight $k - 1$ in any code from $\mathcal{C}_{q,d}$ with codewords of length $k - 1$. Now fix $k \in \{d + 1, \ldots, n\}$. By lemma 0.1, we have that $a\binom{n}{k-1}$ is the number of codewords of weight $k - 1$ in any code from $\mathcal{C}_{q,d}$ with codewords of length n. Therefore we set the coefficient of t^{k-1} correctly at the end of the for loop. Furthermore, when we subtract all of these values from $A[n]$, we will have accounted for all codewords of every weight except n. Hence when $k = n + 1$, we set a to be the number of codewords of weight n in any code from $\mathcal{C}_{q,d}$ with codewords of length n, and since $\binom{n}{n} = 1$, we set the coefficient of t^n in f correctly, meaning we now satisfy $f = W_C(t)$. Note again the value of a did not depend on a specific parity check matrix. □

Runtime Analysis

We have provided a Maple package if the reader would like to experimentally check that our algorithm is faster than the naive version. However, we will provide a rough runtime analysis here. Note we will use a slightly modified algorithm.

Lemma 0.3. *The following algorithm is correct:*

Algorithm 2:

Data: *Parameters n,q,d as described above*
Result: $f = W_C(t)$
$A \leftarrow []$;
for $i \leftarrow 1$ **to** $n - d + 1$ **do**
  $A[i] \leftarrow q^i - 1$;
end
$f \leftarrow 1$;
for $k \leftarrow d + 1$ **to** $n + 1$ **do**
  $a \leftarrow A[k - d]$;
  for $i \leftarrow 1$ **to** $n - k + 1$ **do**
    $a \leftarrow a \times (i + k - 1)$;
    $a \leftarrow a/i$;
    $A[i] \leftarrow A[i] - a$;
  end
  $f \leftarrow f + at^{k-1}$;
end

Proof. It is apparent that we are just computing the binomial coefficients in a different way from algorithm 1, so theorem 0.2 implies that this algorithm is correct. □

Theorem 0.4. *Computing all binomial coefficients in algorithm 2 takes* $O(n^3 \log^2(n) \log\log(n))$ *time.*

Proof. We first observe that we compute at most n binomial coefficients, and it is apparent that at any point in the algorithm, $a < n^n$. Such numbers can be stored using $O(\log(n^n)) = O(n \log(n))$ bits.

By Schönhage and Strassen [6], multiplication of two k-bit numbers may be computed in $O(k \log(k) \log\log(k))$ time. We also know that multiplication and division have the same runtime [7]. Therefore we can compute all binomial coefficients in $O(n^3 \log^2(n) \log\log(n))$ time. □

We have omitted the analysis for the other computations since computing the binomial coefficients will require the most time for inputs where n is close to q.

Weight Distribution Statistics

In this section, we'll compute the mean and variance of the weight distributions of many codes that we have been inspecting and will show some experimental results suggesting patterns for the moments. First we'll compute the means.

Theorem 0.5. *For all $C \in \mathcal{C}_{q,d}$ with codewords of length n, the mean of the weights is $\frac{n(q-1)}{q}$.*

Proof. Fix q,d. We will prove this by cases.

Case $n = d$: If $n = d$, then $|C| = q$. We also know that every nonzero codeword in C has weight at least d, and since $n = d$, there are $q-1$ codewords of weight d and one codeword of weight 0. Therefore the mean of the weights is $\frac{n(q-1)}{q}$.

Case $n > d$: Let C' be the code with parity check matrix constructed by taking the first $n-1$ columns of the parity check matrix for C. Let's say that $W_{C'} = \sum_{i=0}^{n-1} A_i t^i$. By theorem 0.2, we have:

$$W_C(t) = \sum_{i=0}^{n-1} \frac{n}{n-i} A_i t^i + (q^{n-d+1} - \sum_{i=0}^{n-1} A_i \frac{n}{n-i}) t^n.$$

Hence:

$$\begin{aligned} \frac{\sum_{i=0}^{n-1}(iA_i\frac{n}{n-i}) + n(q^{n-d+1} - \sum_{i=0}^{n-1}(A_i\frac{n}{n-i}))}{q^{n-d+1}} &= \frac{\sum_{i=0}^{n-1}((i-n)A_i\frac{n}{n-i}) + nq^{n-d+1}}{q^{n-d+1}} \\ &= \frac{-nq^{n-d} + nq^{n-d+1}}{q^{n-d+1}} \\ &= \frac{n(q-1)}{q}. \end{aligned}$$

□

We note that this is actually the largest possible mean of the weights [8]. Next we compute some variances in a similar way.

Theorem 0.6. *Fix q,n. For all $d < n$, the weight distribution of every code $C \in \mathcal{C}_{q,d}$ with codewords of length n has variance $\frac{n(q-1)}{q^2}$.*

Proof. Fix d,q. Let C' be the code with parity check matrix constructed by taking the first $n-1$ columns of the parity check matrix for C. Let $W_{C'} = \sum_{i=0}^{n-1} A_i t^i$. By theorem 0.5, we know the

mean of C. We compute:

$$\begin{aligned}
\text{Var} &= \sum_{i=0}^{n-1} \frac{n}{n-i} A_i \frac{1}{q^{n-d+1}} (i - \frac{n(q-1)}{q})^2 + (n - \frac{n(q-1)}{q})^2 \frac{1}{q^{n-d+1}} (q^{n-d+1} - \sum_{i=0}^{n-1} \frac{n}{n-i} A_i) \\
&= \sum_{i=0}^{n-1} \frac{n}{n-i} A_i \frac{1}{q^{n-d+1}} ((i - \frac{n(q-1)}{q})^2 - (n - \frac{n(q-1)}{q})^2) + (n - \frac{n(q-1)}{q})^2 \\
&= \sum_{i=0}^{n-1} \frac{n}{n-i} A_i \frac{1}{q^{n-d+1}} \frac{(i-n)((2-q)n + iq)}{q} + (n - \frac{n(q-1)}{q})^2 \\
&= -\frac{n}{q^{n-d+1}} \sum_{i=0}^{n-1} A_i (\frac{(2-q)n}{q} + i) + (n - \frac{n(q-1)}{q})^2 \\
&= -\frac{n}{q^{n-d+1}} (\frac{(2-q)n}{q} q^{n-d} + \frac{(n-1)(q-1)}{q} q^{n-d}) + (n - \frac{n(q-1)}{q})^2 \\
&= \frac{n(q-1)}{q^2}.
\end{aligned}$$

□

Finally we list two conjectures.

Conjecture 0.7. *Fix q,n. For all $d \leq n-2$, all codes $C \in \mathcal{C}_{q,d}$ with codewords of length n have weight distributions with third central moment given by $-\frac{n(q-1)(q-2)}{q^3}$. For all $d \leq n-3$, all codes $C \in \mathcal{C}_{q,d}$ with codewords of length n have weight distributions with fourth central moment given by $\frac{n(3qn+q^2-3n-6q+6)(q-1)}{q^4}$.*

Conjecture 0.8. *Suppose $k \in \mathbb{N}$. Fix q,n. For all $d \leq n+1-k$, all codes $C \in \mathcal{C}_{q,d}$ with codewords of length n have weight distributions with the same first k moments.*

We tested that conjecture 0.7 held for all primes less than 200 and we have provided a method in the accompanying Maple package showing some evidence for conjecture 0.8.

Conclusion

References

[1] Hill, Raymond (1986). A first course in coding theory. Oxford Applied Mathematics and Computing Science Series. Oxford University Press. ISBN 978-0-19-853803-5.

[2] MacWilliams, F. J. (1963). A theorem on the distribution of weights in a systematic code. Bell Syst. Tech. J. 42, 79-94.

[3] E. F. Assmus, Jr., H. F. Mattson, Jr., and R. J. Turyn, Cyclic Codes, Air Force Cambridge Research Labs., Report AFCRL-65-332, Bedford Mass., April 28, 1965.

[4] G. D. Forney, Jr., Concatenated Codes, (M.I.T. Press, Cambridge, MA., 1966).

[5] T. Kasami, S. Lin and W. W. Peterson, Some results on weight distributions of BCH codes, IEEE Trans. Info. Theory, 12 (1966) 274.

[6] Schönhage, Arnold; Strassen, Volker (1971). "Schnelle Multiplikation großer Zahlen" [Fast multiplication of large numbers]. Computing (in German). 7 (3–4): 281–292. doi:10.1007/BF02242355. S2CID 9738629.

[7] D. Knuth: *The art of computer programming* Vol. II, Addison Wesley, London, 1997.

[8] J. I. Hall, Notes On Coding Theory, September 9, 2010.