Reed-Muller Codes

Prateek Sharma

April 11, 2010

This Seminar Report gives an introduction to the Reed-Muller family of Error Correcting Codes. Reed-Muller codes are one of the oldest codes used for error correction. The construction and various interpretations of the codewords is discussed. Reed Muller codes are primarily used because of their large error correcting ability and easy decoding - hence a survey of the decoding techniques and algorithms is presented. Some applications of the code are also presented. Some applications of the code to error correction and other areas are listed.

1 Introduction

Reed-Muller ($\mathcal{R}$) codes are amongst the oldest and most well-known codes. They were discovered and proposed by D. E. Muller and I. S. Reed in 1954. [**?**] [**?**] Reed-Muller codes have many interesting properties - they can be defined recursively, and are an infinite family of codes. Although they become weaker as their length increases, they are often used as building blocks in other codes. One of the major advantages of Reed-Muller codes is their relative simplicity to encode and decode messages. Reed-Muller codes, like many other codes, have tight links to design theory; we briefly investigate the link between Reed-Muller codes and affine geometries.

1.1 Notation

F_q denotes a finite field of order q. Throughout, unless otherwise explicity stated, $q = 2$ and therefore $F = \{0, 1\}$, with the addition operation being equivalent to the binary exclusive-or (xor), and the multiplication operation being the binary *and*.

A vector field over F,(with the length of the vector being n) will be denoted by F^n. For more on vector fields and finite fields, refer to [**?**],chapter 4 of [**?**] , but a very basic understanding is sufficient for understanding Reed-Mullercodes.

We use a string of length n with elements in F_2 to write a vector in the vector space F_2^n . For example, if we have the vector $\mathbf{v} = (1, 0, 1, 1, 0, 1, 0, 1) \in F_2^8$, we simply write $\mathbf{v}$ as 10110101.

We shall also deal with boolean functions, and shall denote the boolean *xor* by $+$, instead of the customary $\oplus$ to maintain consistency with the addition operation in F_2.

Three parameters of linear error-correcting codes are it's length, dimension, and minimum distance, denoted by the triplet $[n, k, d]$. The elements of a code C are called the *codewords*. Codewords are represented as vectors of length n.

The distance between two vectors $\mathbf{u}$ and $\mathbf{v}$ which we are concerned about is the *Hamming Distance* which is the number of positions in which 2 vectors differ. Analogously, the hamming weight of a vector is defined as:

$$wt(x) = d(x, 0) = \begin{cases} 1 & \text{if } x \neq 0 \\ 0 & \text{if } x = 0 \end{cases} \tag{1}$$

We will use $d(x, y) = wt(x-y)$ and $wt(x+y) = wt(x)+wt(y)-2wt(x*y)$, where $x*y = (x_1y_1, x_2y_2, \ldots, x_ny_n)$.

2 Construction of the Code

The simplest construction of Reed-Mullercodes is by using boolean functions. Let $(x_1, x_2, \ldots, x_m) \in F^m$ be the set of binary m-tuples, and let $\mathbf{x} \equiv (x_1, x_2, \ldots, x_m)$. Consider a boolean function f, $f : F^{2^m} \rightarrow \{0, 1\}$. Thus $f(\mathbf{x}) = f(x_1, x_2, \ldots, x_m)$ takes an m-tuple and returns either 0 or 1 .

$\mathbf{x}$ can take 2^m values. For each value of $\mathbf{x}$ we compute $f(\mathbf{x})$. This is the familiar and ubiquitious *truth-table* wherein a boolean function is evaluated for all possible inputs. In the example below, x_1, x_2, x_3 are rows and $f(x_1, x_2, x_3)$ is computed and represented as another row.

x_1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1
x_3	0	1	0	1	0	1	0	1
f	0	0	0	1	1	0	0	0

Since $(x_1, x_2, \ldots, x_m)$ can take 2^m values, $\mathbf{f}$ is a $n = 2^m$ length vector over F_2. Since there are 2^{2m} such boolean functions possible, this gives us a collection of 2^{2m} vectors, each of length 2^m.

The Reed-Muller codes are particular subsets of this collection as described below.

f can be written in the disjunctive normal form. Using logical operations, f can be represented as a function of the $x_1, x_2, \ldots, x_m$. In the above example, for instance, $f = x_1 \vee x_2 \vee x_3$.... We can represent any boolean function in the Disjunctive Normal form (with or replaced by xor) [**?**], for instance in the above example $f =$

Define a Boolean Monomial as . The monomials are

$$1, x_1, x_2,$$

There are

$$1 + \binom{m}{1} + \binom{m}{2} + \ldots + \binom{m}{m} = 2^m$$

distinct monomials. The degree of the boolean monomial is ...

A Boolean polynomial is a linear combination (with coefficients in F_2) of boolean monomials. The degree of the boolean polynomial is the degree of the its highest term (monomial).

Therefore, the polynomials (we will drop the term boolean from here on) of degree 1 are of the form:

$$1 + a_1 x_1 + a_{2x} 2 + \ldots + a_n x_n \tag{2}$$

The above simply represents a linear combination of m vectors.

With this, we can now define Reed-Mullercodes of length m and order r as :

$$\mathcal{R}(r, m) = \{\mathbf{f} : degree(f(x_1, \ldots, x_m)) = r\} \tag{3}$$

¡this is not clear¿

Since there are $\binom{m}{r}$ monomials of degree r, the number of The boolean monomials are

$$1, v_1, v_2, \ldots, v_m, v_1 v_2, v_1 v_3, \ldots, v_1 v_m, ldots, v_1 v_2 .. v_m$$

Thus there are

$$1 + \binom{m}{1} + ldots + \binom{m}{m} = 2^m$$

boolean monomials.

We can write the generator matrix such that the columns are the binary representations of the numbers from 0 to $2^m - 1$.

If there f is an $m - ary$ function, it can take the 2^m input values. Hence vector $\mathbf{f}$ has length 2^m. Since $\mathbf{f}$ is also a linear combination, it follows that the length of $x_1, x_2, ldots, x_m$ is 2^m.

2.1 clarification

In what follows, we frequently switch back and forth between B(r, v1 , . . . , vm) and R(r, m). Doing so in the most explicit manner would make the reasonings a lot harder to read, and for this reason we decided to treat codewords and Boolean functions as interchangeable. In fact, boolean logic using xor is also called reed-muller logic! [**?**]

Properties: Length, $n = 2^m$ Minimum Distance, $d = 2^{m-r}$ Dimension, $k = 1 + ..$

The rate of an $[n, k]$ code is dened as k/n. (Thinking of the code as having nk check digits and k message digits.

2.2 Other interpretations

Set theory definition can also be used to define the codes. See [0]

The natural polynomial ring definition is standard but the boolean logic one is easier to understand.

2.3 Code properties

We have seen that the first order Reed-Muller code $\mathcal{R}(1, m)$ consists of all vectors $u_0 1 + \sum_{i=1}^{m} u_i \mathbf{v_i}$ with $u_i = 0$ or 1. Define the *orthogonal code* $\prime_m$ to be the $[2m, m, 2m1]$ code consisting of the vectors $\sum_{i=1}^{m} u_i \mathbf{v_i}$ Then

$$\mathcal{R}(1, m) = \mathcal{O}_m(\mathbf{1} + \mathcal{O}_m)$$

Uniqueness: Any linear code with parameters $[2^m, m + 1, 2^m]$ is equivalent to the first order Reed-Mullercode.

from [?]. □

Run-length properties at [**?**]

2.4 Recursive Formulation

Theorem 1. $\mathcal{R}(r + 1, m + 1) = \{\mathbf{u}|\mathbf{u} + \mathbf{v} : \mathbf{u} \in \mathcal{R}(r + 1, m), \mathbf{v} \in \mathcal{R}(r, m)\}$

This is known as the concatenation construction of codes, with | denoting the concatenation.

Proof. We use the boolean logic definition of the codewords. Let $\mathbf{f} \in \mathcal{R}(r + 1, m + 1)$. $\mathbf{f}$ can be written as

$$f(v_1, v_2, \ldots, v_{m+1}) = g(v_1, v_2, \ldots, v_m) + v_{m+1} h((v_1, v_2, \ldots, v_m)$$

Where $\mathbf{g} \in \mathcal{R}(r + 1, m)$ and $\mathbf{h} \in \mathcal{R}(r, m)$. Consider the associated vectors $\mathbf{f}, \mathbf{g}', \mathbf{h}'$ as polynomials over $v_1, \ldots, v_{m+1}$. Then, $\mathbf{g}' = (\mathbf{g}|\mathbf{g})$ and $\mathbf{h}' = (\mathbf{0}|\mathbf{h})$. [Problem7, [0].] Thus

$$\mathbf{f} = \mathbf{g}|\mathbf{g} + \mathbf{0}|\mathbf{h}$$

□

The generator matrix version of the above theorem is

$$G(r + 1, m + 1) = \begin{pmatrix} G(r + 1, m) & G(r + 1, m) \\ 0 & G(r, m) \end{pmatrix} \tag{4}$$

Because of the systematic way the basis vectors are chosen,

$$G(1, m + 1) = \begin{pmatrix} G(1, m) & G(1, m) \\ 0 & 1 \end{pmatrix} \tag{5}$$

Recursive definition naturally is very helpful for proving certain properties, particularly since it enables the use of induction.

Theorem 2. *Minimum distance,* $d = 2^{m-r}$

Proof. Proof:... □

$\mathcal{R}(0, m)$ is the repetition code. At the other extreme $\mathcal{R}(m, m)$ is a code consisting of all possible binary sequences of length m.

Another recursive construction Let

$$R_1 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Then, $R_{n+1} = \begin{pmatrix} R_n & R_n \\ R_n & negR_n \end{pmatrix}$

2.5 Properties

The Reed-Mullercodes are a nested family of codes - the codes of higher order contain those of the lower order.

Theorem 3.

$$\mathcal{R}(r, m) \subset \mathcal{R}(t, m) \text{ if } 0 \leq r \leq t \leq m$$

Proof. By Induction. Trivially true for $m = 1$. Let $\mathcal{R}(k, m-1) \subset \mathcal{R}(l, m-1)$ for all $0 \leq k \leq l < m$. Let $0 < i \leq j < m$. By the recursive definition, we get:

$$\mathcal{R}(i, m) = \{(\mathbf{u}, \mathbf{u} + \mathbf{v} | \mathbf{u} \in \mathcal{R}(i, m-1), \mathbf{v} \in \mathcal{R}(i-1, m-1)\}$$

Induction hypothesis gives :

$$\subset \{(\mathbf{u}, \mathbf{u} + \mathbf{v} | \mathbf{u} \in \mathcal{R}(j, m-1), \mathbf{v} \in \mathcal{R}(j-1, m-1)\} = \mathcal{R}(j, m)$$

□

$$dim(\mathcal{R}(r, m)) = dim(\mathcal{R}(r, m-1)) + dim(\mathcal{R}(r-1, m-1)) \tag{6}$$

Theorem 4.

(7)

Theorem 5. $\mathcal{R}(m-r-1, m)$ *is the dual code of* $\mathcal{R}(r, m)$

Proof. We induct on r. Let $0 \leq i \leq r$. Inductively, assume that $\mathcal{R}(i, m-1)^{\perp} = \mathcal{R}(m-i-2, m-1)$

□

Theorem 6.

$$\mathcal{R}(m-2, m) \text{ is the extended binary hamming code}$$

Proof. □

We prove a general theorem

Theorem 7. *Let C_i be an $[n, k_i, d_i]$ code. Then the concatenated code defined by*

$$C = \{(\mathbf{u}, \mathbf{u}+\mathbf{v}) | \mathbf{u} \in C_1, \mathbf{v} \in C_2\}$$

has the parameters $[2n, k_1 + k_2, min2d_1, d_2]$ linear code.

Proof. Length: Clearly, since length of C_1 and C_2 is n each, concatenating produces vectors of length $2n$.

Dimension: The number of words in C is product of number of words in C_1 and C_2. because $(c_1, c_2) \to (c_1, c_1 + c_2)$ is a bijection. Thus, the dimension is

$$k = k_1 + k_2$$

Distance: We can split this into cases, depending on whether $c_1 = \mathbf{0}$ or $c_2 = \mathbf{0}$.

Case: $\mathbf{c_2} = \mathbf{0}$

$$wt((c_1, c_1 + c_2)) = wt(c_1, c_1) = 2wt(c_1) = 2d_1$$

Case: $\mathbf{c_2} \neq \mathbf{0} wt((c_1, c_1 + c_2)) = wt(c_1) + wt(c_1 + c_2) \geq wt(c_1) + wt(c_2) - wt(c_1) = wt(c_2) = d_2$

Thus, the minimum distance of the code, which is equal to the weight of the minimum weight vector is $min2d_1, d_2$

□

Lemma 1. *Every codeword in $\mathcal{R}(1, m)$ (except $\mathbf{1}, \mathbf{0}$) has weight 2^{m-1}*

Proof. This can also be proved by using the randomization lemma for boolean functions as done in [0]. However, if we notice that

$$\mathcal{R}(1, m) = \{(u, u) : u \in \mathcal{R}(1, m-1)\}\{(u, \vec{1} + u) : u \in \mathcal{R}(1, m-1)\}$$

We use induction on m and are done. □

Theorem 8.

$$\mathcal{R}(m - r - 1, m) = \mathcal{R}(r, m)^{\perp}$$

Proof. Let $\mathbf{a} \in \mathcal{R}(m - r - 1, m)$, and $\mathbf{b} \in \mathcal{R}(r, m)$. Then a and b are polynomials of degree $m - r - 1$ and r respectively. $\mathbf{ab} \in \mathcal{R}(m-2, m)$, and since $\mathcal{R}(m-2, m)$ contains all the even weight vectors, $a \cdot b \equiv 0 (mod 2)$. Thus, $\mathcal{R}(m - r - 1, m) \subset \mathcal{R}(r, m)^{\perp}$. But by considering the dimensions we get

$$dim(\mathcal{R}(m - r - 1, m)) + dim(\mathcal{R}(r, m) = 2^m = n$$

This implies the relation. □

Theorem 9. *The dual code $\mathcal{R}(1, m)^{\perp}$ is the extended binary Hamming code $H(m)$*

Proof. TODO. this is important. From singa book □

3 Bounds

$A_q(n, d)$ is the largest integer M such that there exists a q-ary (n, M, d) code. A q-ary (n, M, d) code C is called optimal if $M = A_q(n, d)$.

The plotkin bound shows that the codes are all optimal.

4 Weight distribution

Stated without proof [**?**]

$$\text{There are at least} 2^{mrr(r1)}.\text{minimum weight codewords in } \mathcal{R}(r,m).$$

(8)

5 cyclic decoding

In addition, it is next shown that a cyclic RM (r, m) code is simpler to decode. The argument is as follows. It is shown in Chapter 3 that in a cyclic code C , if (v1 , v2 , . . . , vn) is a code word of C , then its cyclic shift (vn , v1 , . . . , vn 1) is also a code word of C . Therefore, once a particular position can be corrected with ML decoding, all the other positions can also be corrected with the same algorithm (or hardware circuit) by cyclically shifting received code words, until all n positions have been tried.

$\mathcal{R}(,)$

6 Decoding

As mentioned earlier, the RM family of codes came into prominence because of the ease of decoding. In this section we present a variety of decoding algorithms. Several very efficient algorithms have been known specially for the Reed-Mullercodes of the first order.

6.1 Majority Logic Decoding

Majority Logic Decoding is the oldest and simplest method to decode the Reed-Mullercodes. [**?**] Some terminology is needed for a clearer understanding: Let C be an arbitrary code. Then the *parity-check* is simply a code vector in the orthogonal code $C^{\perp}$. Define the *support vector* of a vector $\vec{v}$ as the coordinates where it is non-zero.For example, the support-vector of 01010100 is $(2, 4, 6)$. Suppose we are given J parity checks and a coordinate position, i, such that the intersection of all the J supports is precisely i. The J parity checks essentially 'vote' for the position i , and the result of the majority for the parity of i is the value. A collection of such parity checks described above is said to be *orthogonal* to i. If each of the coordinates of the code has a collection of J parity checks orthogonal to it, then the code is capable of correcting upto $J/2$ errors. What are the conditions under which a code satisfies this? Another definition of orthogonality is: A set of parity check equations is called orthogonal on the i coordinate if x_i appears in each equation and no other x_j appears more than once in the set. Clearly, the minimum weight must be atleast $J+1$, because the

If there are J parity checks on every co-ordinate, the code can correct $J/2$ errors. Proof: Let m_i be the message bit at coordinate i. x_i is the received bit. Let $m_i = 0$ Since there are J orthogonal parity equations voting for x_i, if there are fewer than $J/2$ errors, then the result is unchanged. Therefore atleast $J/2$ equations are 0 and thus $x_i = 0$. If $m_i = 1$ Still less than $J/2$ equations are affected, and we get the right result.

Note: Ties can be broken in any way consistently. Either 0, etc.

The number of errors that can be corrected by one-step majority logic decoding, E_1 is at most $n - 1/2(d' - 1)$ where d' is the minimum distance of the dual code.

Proof: The dual code gives the parity check equations. Consider the 1st coordinate, then the parity checks orthogonal will have the form

111100000100011111

Since there are J orthogonal equations there will be J such vectors. Therefore by the definition of orthogonality the coordinate positions in the dual code vectors cannot overlap, hence $J \leq n - 1/(d' - 1)$. We get the desired result by using the previous theorem.

6.2 L-step decoding

We can extend the idea of step-decoding using orthogonal parity checks by generalizing for parity check equations to be orthogonal on a set of coordinates.

Definition: A set of parity checks $S_1, S_2, \ldots$ is orthogonal on coordinates $i, j, ..$ if the sum $x_i + x_j + \ldots$ appears in each S but no other x_p appears more than once in the set.

The number of errors which can be corrected by an L-step majority decoding scheme, $E_l \leq n/d' - 1/2$

Proof: Let the $i - th$ parity check involve a_i coordinates (besides the l). Since these checks correspond to the codewords in the dual code, we have

$$l + a_i \geq d'$$

$a_i + a_j \geq d'(9)$

We also have

$$S = \sum_{i=1}^{J} a_i \leq n - l$$

Thus,

$$Jl + S \geq Jd'$$

(J-1)S $\geq \binom{J}{2} d'$Eliminating l and S, we get

$$J \leq 2n/d' - 1$$

6.3 Reed Decoding Algorithm

The Reed decoding algorithm is a majority logic decoding scheme for decoding Reed-Mullercodes of any order.

Each codeword from $\mathcal{R}(r, m)$ is an incidence vector that denes a subspace in , in which the nonzero coordinates of the codeword are points lying in the subspace. Based upon this geometry we can nd equations for orthogonal checksums as follows. To nd a set of orthogonal checksums that provide an estimate for the kth order message bit $m_{i_1,i_2,\ldots,i_k} correspondingtothebasisvectors v_{i_1} v_{i_2} \ldots v_{ik}$.

Let the corresponding subspace be S. Let T be the complementary subspace of S where $j_1 j_{2m-k} = 1, 2, ldots, m - i_1, i_2, \ldots, i_k$. The incidence vector of T is v. ¡todo¿

Define a *Translate* of a subspace with respect to a point to be the subspace obtained by adding the binary representation of the point to each element of the subspace.

If there are no errors, the message bit m_b is given by

$$m_b = \sum_{P \in U_i} x_P \quad i = 1, 2, \ldots, 2^{m-r}$$

geometry. □

For a detailed algorithm refer to [**?**] and [**?**]

6.4 Geometrical

We can use the finite geometry construction of the codes to help in the decoding.

6.5 Hadamard Transforms

$of a binary vector \mathbf{V} is a vector with 0 replaced by 1 and 1 by$ -1.

Lemma 2. *The Orthogonal code $\mathcal{O}_m$ with real vectors is equivalent to the Hadamard matrix H_m.*

Proof. Let the elements of this real-vectorized orthogonal code be v.

Claim 1.

$$v_i, v_j \in \mathcal{R}(1,m) \implies v_i \cdot v_j = 0$$

Proof. The systematic choice of the basis vectors helps. We use the recursive definition of the Generator matrix.

□

This is the definition of the hadamard matrix too. □

$$\mathcal{R}(1,m) = \begin{pmatrix} H_m \\ -H_m \end{pmatrix} \tag{10}$$

Keeping note that the code has been converted to the real-vector form, maximizing FH_m is equivalent to finding the least-distance neighbour to $\mathbf{v}$. This would entail doing a simple vector product with each row of the hadamard matrix. There are n such rows, and we need $O(n^2)$ operations. However, by using the *Fast Hadamard Transform* , we can speed it up to $O(nlogn)$.

This is the fastest classical decoding technique and was employed in the *Green Machine* decoder for the Mariner-9 space mission.

6.6 List Decoding

A list decoding procedure for a code C is a function which for given a $\mathbf{u} \in F_q^n$ and $e > 0$ outputs all vC such that $(u, v) \leq e$. Alternately, given a decoding radius T, it should produce a list (set) of codewords which are distance less than T. List decoding is an old technique given by P.Elias in 1950 [**?**], but has recently found significant attention.

The existance of a good, fast list decoding algorithm for the first order Reed-Mullercodes was given by goldreich and levin [**?**], [**?**]. However they do not give a usable algorithm. Recently, Kabatiansky, Dumer and Tavernier [**?**], [**?**] have presented a simple list decoding algorithm for $\mathcal{R}(1,m)$ capable of correcting $n(\frac{1}{2} - \epsilon)$ errors in $O(n\epsilon^3)$. This is significantly better than the $\frac{n}{4}$ correcting capability of the Hadamard transform **??**, which uses $O(nlogn)$ time.

Let $\mathbf{y}$ be a received vector and $L_\epsilon(y) = \{f \in \mathcal{R}(1,m) : d(\mathbf{y}, \mathbf{f}) \leq n(\frac{1}{2} - \epsilon)\}$ be the desired list. The algorithm works recursively by nding on the i-th step a list $L_\epsilon^i(y)$ of 'candidates' which should (but may not) coincide with i-prex of some $f(x1, ..., xm) = f0 + f1x1 + ... + fmxmL(y)$

The main idea is to approximate the Hamming distance between the received vector y and an arbitrary "propagation" of a candidate $c(i)(x1, ..., xm) = c1x1 + ... + cixi$ by the sum of Hamming distances over all i-dimensional "facets" of the m-dimensional Boolean cube.

The algorithm

Johnson Bound [0] If C is an [n,k,d] code with $d \geq (1/2 - \epsilon)$, then

$$\forall u \in F_q^n | \{v \in C : \Delta(u,v) \leq (1/2 - \sqrt{\epsilon})\} | leq 1/\epsilon$$

Proof: (from lec10.ps)

6.7 RM(1,m) decoding algorithm

kabatiansky

7 Generalized codes

8 Applications

8.1 Communication

The oldest and original use. Reed-Mullercodes were used in the 1969 Mariner-9 deep space probe to the IEEE 802.11b standard for Wireless Local Area Networks (WLANs) [**?**].

8.2 Tesing Low-degree polynomials

[**?**] Using $\mathcal{R}(1, m)$ codes to test whether a binary function is a low-degree polynomial. The functions are mapped to the Reed-Mullercodes, and the properties are used to prove the bounds on the number of queries needed.

8.3 Sidenkov cryptanalysis

[**?**] A variant of the McEliece cryptosystem using Reed-Mullercodes instead of Goppa codes , called the Sidenkov system, because of the efficient decoding. A cryptanalysis attack also uses the properties of Reed-Mullercodes to break the cryptographic code. The uniqueness result proved earlier is a central feature in the cryptographic attack.

8.4 Side Channel attacks

[**?**]

[**?**]

9 References

References

[1] M.K. Behbahani, GB Khosrovshahi, and B. Tayfeh-Rezaie. Reed-Muller Codes and Incidence Matrices.

[2] N. Sloane and F. MacWilliams. The Theory of Correcting Codes, 1979.

[3] David Zuckerman. Lecture notes cs295t, lecture 10.