Reed Muller Codes

Prateek Sharma

April 6, 2010

Abstract

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 various decoding techniques and algorithms is presented.

1 Introduction and Terminology

Reed-Muller ($\mathcal{R}$) codes are are amongst the oldest and most well-known of codes. They were discovered and proposed by D. E. Muller and I. S. Reed in 1954. Reed-Muller codes have many interesting properties that are worth examination; they form an innite family of codes, and larger Reed-Muller codes can be constructed from smaller ones. This particular observation leads us to show that Reed-Muller codes can be dened recursively. Unfortunately, Reed-Muller codes become weaker as their length increases. However, they are often used as building blocks in other codes. One of the ma jor advantages of Reed-Muller codes is their relative simplicity to encode messages and decode received transmissions. We examine encoding using generator matrices and decoding using one form of a process known as majority logic. Reed-Muller codes, like many other codes, have tight links to design theory; we briey investigate this link between Reed-Muller codes and the designs resulting from 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 boolean exclusive-or (xor), and the multiplication operation being the boolean *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 [**?**], but this very basic understanding is sufficient for the purpose of understanding this report.

We use a string of length n with elements in F_2 to write a vector in the vector space F_n^2 . 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, minimum distance, and dimension, 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 $HammingDistance$, defined as:

$$d_H = \sum_{u_i \neq v_i} 1 \tag{1}$$

In other words, the hamming distance is the number of positions in which 2 vectors differ. Analogously, the hamming weight of a vector is defined as:

$$wt(x) = d(x, 0) = \{ \begin{array}{ll} 1 & \text{if } x \neq 0 \\ 0 & \text{if } x = 0 \end{array} \tag{2}$$

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

The simplest construction of 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 $v \equiv (x_1, x_2, \ldots, x_m)$. Consider a Boolean Function f, $f : F_2^m \to 0, 1$. Thus $f(\mathbf{x}) = f(x_1, x_2, \ldots, x_m)$ takes an m-tuple and returns either 0 or 1 .

v can take 2^m values. For each value of v we compute $f(v)$. 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, 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.

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 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:

$$1 + a_1x_1 + a_{2x}2 + \ldots + a_nx_n \tag{3}$$

The above simply represents a linear combination of m vectors. We now define the codes.

$$\mathcal{R}(r, m) = \text{Boolean polynomials over of degree r} \tag{4}$$

The boolean monomials are

$$1, v_1, v_2, \ldots, v_m, v_1v_2, v_1v_3, \ldots, v_1v_m, ldots, v_1v_2..v_m$$

Thus there are $1+ \text{m}\binom{}{1+ldots+\binom{m}{m}=2^m}$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.

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

2.1 Recursive Formulation

$\mathcal{R}(r + 1, m + 1) = \{\mathbf{u}+\mathbf{v}— \mathbf{u} \in \mathcal{R}(r + 1, m), \mathbf{v} \in \mathcal{R}(r, m)\}$This is known as the concatenation construction of codes, with + denoting the concatenation. Proof: 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 $g \in \mathcal{R}(r + 1, m)$ and $h \in \mathcal{R}(r, m)$

Theorem is also equivalent to:

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

Because of the systematic way the basis vectors are chosen,

$$G(1, m + 1) = \begin{pmatrix} G(1, m) & G(1, m) \\ 0 & 1 \end{pmatrix} \tag{6}$$

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

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

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

$$\mathcal{R}(r, m) \subset \mathcal{R}(t, m)$$

if $0 \leq r \leq t \leq m(7)$

2.2 Weight distribution

2.3 Distance, error correction

3 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.

4 Decoding

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

4.1 Majority Logic Decoding

Majority Logic Decoding is the oldest and simplest method to decode the codes. 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^T. 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.

4.2 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

$$l + a_i \geq d'$$

$\mathrm{a}_i + a_j \geq d'(8)$

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

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

4.3 Geometrical

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

4.4 Hadamard Transforms

Green machine

4.5 List Decoding

A list decoding procedure for a code C is a function which for given a u_q^n and $e > 0$ outputs all vC such that $\Delta(u, v) \leq e$.

Johnson Bound [**?**] 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)

4.6 RM(1,m) decoding algorithm

5 Generalized codes

6 Uses of Reed Muller codes

[**?**]

7 References