Код Рида-Маллера

Илья Коннов

Факультет компьютерных наук

Высшая Школа Экономики

15 февраля 2022 г.

1. Существует три различных варианта этого доклада:
   1.1 Краткая презентация, которую несложно рассказать, но может быть сложно понять (ReedMuller-trans.pdf).
   1.2 Более длинная презентация с ценными комментариями, дополнительными доказательствами и интересными фактами (ReedMuller-slides.pdf). Слайды с особенным фоном — не вошедшие в маленькую презентацию.
   1.3 Текстовая статья со всем содержимым длинной презентации, комментариями на своих местах, а также бонусным приложением с более подробным описанием алгоритма (ReedMuller-article.pdf).

   Их все можно посмотреть здесь: https://sldr.xyz/ReedMuller/

По любым вопросам: r-m@sldr.xyz или t.me/iliago или vk.com/iliago.

Введение

Код описан Дэвидом Маллером (автор идеи) и Ирвингом Ридом (автор метода декодирования) в сентябре 1954 года. Обозначается как $\mathrm{RM}(r,m)$, где r — ранг, а 2^m — длина кода. Кодирует сообщения длиной $k = \sum_{i=0}^{r} C_m^i$ при помощи 2^m бит.
Традиционно, считается что коды бинарные и работают над битами, т.е. $\mathbb{F}_2$.
Соглашение: сложение векторов $u, v \in \mathbb{F}_2^n$ будем обозначать как $u \oplus v = (u_1 + v_1, u_2 + v_2, ..., u_n + v_n)$.

Булевы функции и многочлен Жегалкина

Всякую булеву функцию можно записать при помощи таблицы истинности:

x	y	$f(x,y)$
0	0	1
0	1	0
1	0	0
1	1	0

Или при помощи многочлена Жегалкина:

$$f(x,y) = xy + x + y + 1$$

Многочлены Жегалкина

В общем случае, многочлены будут иметь следующий вид:

$$f(x_1, x_2, ..., x_m) = \sum_{S \subseteq \{1,...,m\}} c_S \prod_{i \in S} x_i$$

Например, для $m = 2$:
$f(x_1, x_2) = c_3 \cdot x_1 x_2 + c_2 \cdot x_2 + c_1 \cdot x_1 + c_0 \cdot 1$
Всего $n = 2^m$ коэффициентов для описания каждой функции.

Функции небольшой степени

Рассмотрим функции, степень многочленов которых не больше r:

$$\{f(x_1, x_2, ..., x_m) \mid \deg f \leq r\}$$

Каждую можно записать следующим образом:

$$f(x_1, x_2, ..., x_m) = \sum_{\substack{S \subseteq \{1,...,m\} \\ |S| \leq r}} c_S \prod_{i \in S} x_i$$

В каждом произведении используется не больше r переменных.
Сколько тогда всего коэффициентов используется?

$$k = C_m^0 + C_m^1 + C_m^2 + ... + C_m^r = \sum_{i=0}^{r} C_m^i$$

1. Замечу, что при $S = \varnothing$, мы считаем, что $\prod_{i \in S} x_i = 1$, таким образом всегда появляется свободный член.
2. Если говорить несколько проще, то для составления многочленов мы сложим сначала одночлены $(x + y + z + \ldots)$, затем произведения одночленов $(xy + yz + xz + \ldots)$ и т.д. вплоть до r множителей (поскольку мы работаем в поле $\mathbb{F}_2$, здесь нету x^2, y^2, z^2, т.к. $a^2 = a$). Тогда легко видеть, почему k именно такое: мы складываем все возможные перестановки сначала для 0 переменных, потом для одной, двух, и так вплоть до r (не не больше, ведь $\deg f \leq r$).

Идея кодирования

Пусть каждое сообщение (длины k) — коэффициенты многочлена от m переменных степени не больше r.
Тогда мы можем его представить при помощи 2^m бит, подставив все возможные комбинации значений переменных.
Таким образом получим таблицу истинности, из которой позднее сможем восстановить исходный многочлен, а вместе с ним и сообщение.
Зафиксировав в таблице порядок строк, можно выделить **вектор значений**, который и будет кодом.

x	y	$f(x,y)$
0	0	1
0	1	0
1	0	0
1	1	0

$$\implies \mathrm{Eval}(f) = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}$$

1. Их 2^m, поскольку рассматриваем многочлены только над $\mathbb{F}_2$ от m переменных.
2. Вектор значений — обозначается $\mathrm{Eval}(f)$ — столбец таблицы истинности, содержащий значения функции. Имеет смысл только при зафиксированном порядке строк в таблице. У меня он везде самый обычный, как в примере выше.

Пример

- $r = 1$ (степень многочлена), $m = 2$ (переменных). Это $\mathrm{RM}(1, 2)$.
- Тогда наш многочлен: $f(x_1, x_2) = c_2x_2 + c_1x_1 + c_0$.
- Сообщение: `101`, тогда $f(x_1, x_2) = x_2 + 0 + 1$.
- Подставим всевозможные комбинации:

x_1	x_2	$f(x_1, x_2)$
0	0	1
0	1	1
1	0	0
1	1	0

- Получили код: $\mathrm{Eval}(f) =$ `1100`.

1. Здесь и далее я для краткости и удобства записываю битовые векторы не как $\begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix}$, а как `1001` при помощи нескучного шрифта.

Декодирование когда потерь нет

- Мы получили код: `1100`
- Представим таблицу истинности.

x	y	$f(x_1, y_2)$
0	0	1
0	1	1
1	0	0
1	1	0

- Подстановками в $f(x_1, x_2) = c_2x_2 + c_1x_1 + c_0$ получим СЛАУ.

$$\begin{cases} c_0 = 1 \\ c_1 + c_0 = 1 \\ c_2 + c_0 = 0 \\ c_2 + c_1 + c_0 = 0 \end{cases}$$

- $c_2 = 1, c_1 = 0, c_0 = 1$, исходное сообщение: `101`.

1. Теперь покажем, как можно декодировать когда потерь нет. Этот пример — продолжение предыдущего.

Коды 0-го порядка

Для случая $\mathrm{RM}(0, m)$ нужна функция от m аргументов, степени не выше 0.

- $f(x_1, x_2, ..., x_m) = 0$
- $g(x_1, x_2, ..., x_m) = 1$

Таблица истинности:

$$
\begin{array}{c}
\\
2^m\left\{\begin{array}{c} \\ \\ \\ \\ \end{array}\right.
\end{array}
\begin{array}{cccc|cc}
x_1 & x_2 & \dots & x_m & f(x_1, ..., x_m) & g(x_1, ..., x_m) \\
\hline
0 & 0 & \cdots & 0 & 0 & 1 \\
0 & 0 & \cdots & 1 & 0 & 1 \\
 & & \ddots & & \vdots & \vdots \\
1 & 1 & \cdots & 1 & 0 & 1
\end{array}
$$

Вывод: это 2^m-кратное повторение символа

- Сообщение 0 даст код $\underbrace{\texttt{00...0}}_{2^m}$
- Сообщение 1 даст код $\underbrace{\texttt{11...1}}_{2^m}$

1. Отдельно стоит рассмотреть вариант кода при $r = 0$, он нам в будущем пригодится для доказательств.
2. Таких функций существует всего лишь две, поскольку мы можем влиять лишь на свободный член. Все остальные коэффициенты обнуляются из-за требования $\deg f \leq 0$.
3. Здесь число строк, как и в любой другой таблице истинности, равно 2^m, а колонки со значениями никак не зависят от аргументов функций. Получается две колонки – одна с нулями, другая с единицами.

Коды m-го порядка

Есть m переменных, и мы рассматриваем многочлены $f \in \mathbb{F}_2[x_1, ..., x_m] : \deg f \leq m$, т.е. все возможные.
Для $\mathrm{RM}(m, m)$ мы используем все доступные коэффициенты многочлена для кодирования сообщения.
Тогда нет избыточности: $k = \sum_{i=0}^{m} C_m^i = 2^m = n$ – длина сообщения равна длине кода.

Чем меньше порядок кода r, тем больше избыточность.

1. Есть ещё один тривиальный случай, когда $m = r$.

Доказательство линейности

Пусть $C(x)$ кодирует сообщение $x \in \mathbb{F}_2^k$ в код $C(x) \in \mathbb{F}_2^m$.

$$C(x) = (p_x(a_i) \mid a_i \in \mathbb{F}_2^m)$$

где $p_x(a_i)$ — соответствующий сообщению x многочлен.
Причём p_x берёт в качестве своих коэффициентов биты из x. Поскольку многочлены степени не выше r образуют линейное пространство, то $p_{(x \oplus y)} = p_x + p_y$.
Тогда:

$$C(x \oplus y)_i = p_{(x \oplus y)}(a_i) = p_x(a_i) + p_y(a_i) = C(x)_i + C(y)_i$$

т.е. $\forall x, y \quad C(x \oplus y) = C(x) + C(y)$, ч.т.д.

1. Хотим показать, что этот код является линейным, т.е. что его кодовые слова образуют линейное пространство, и у нас есть изоморфизм из пространства сообщений ($\mathbb{F}_2^k$) в пространство слов ($\mathbb{F}_2^m$).
   Для этого необходимо немного формализовать всё описанное раньше.
2. Пояснение: перебираем все векторы a_i (2^m штук), подставляем каждый в p_x в качестве переменных и таким образом получаем вектор значений (длины 2^m). Именно он и называется кодом.
3. Напомню, что базис пространства многочленов выглядит примерно так: $1, x, y, z, xy, yz, xz$ (для трёх переменных, степени не выше 2). Чтобы преобразовать сообщение в многочлен, мы берём каждый бит сообщения и умножаем его на соответствующий базисный вектор. Очевидно, такое преобразование будет изоморфизмом. Именно поэтому $p_{(x \oplus y)} = p_x + p_y$. Обратите внимание, что сообщение x это не просто число ($\mathbb{Z}_{2^k}$) и мы рассматриваем его биты, а реально вектор битов ($\mathbb{Z}_2^k$). У него операция сложения побитовая.
4. Здесь я использую запись $C(x)_i$ для i-го элемента вектора $C(x)$. Поскольку i произвольное, то и весь вектор получился равен. Таким образом, этот код действительно линейный и к нему применимы уже известные теоремы!

Последствия линейности

1. Существует порождающая матрица G.

$$C(x) = x_{1\times k} G_{k\times n} = c_{1\times n}$$

2. Минимальное расстояние будет равно минимальному весу Хемминга среди всех кодов.

$$d = \min_{\substack{c \in C \\ c \neq 0}} w(c)$$

3. Корректирующая способность:

$$t = \left\lfloor \frac{d-1}{2} \right\rfloor$$

1. Так можно кодировать сообщения x в коды c. Но искать её мы не будем, обойдёмся одними многочленами, это интереснее.
2. Вес Хэммингa вектора — количество в нём ненулевых элементов.
3. Доказательство очень просто: минимальное расстояние — вес разности каких-то двух различных кодов, но разность двух кодов тоже будет кодом, т.к. мы в линейном пространстве. Значит достаточно найти минимальный вес, но не учитывая нулевой вектор, т.к. разность равна нулю тогда и только тогда, когда коды равны.
4. Однако мы ещё не знаем как выглядят наши коды (как выглядят таблицы истинности функций степени не больше r?). А значит не можем ничего сказать про минимальное расстояние.

Конструкция Плоткина: многочлены

Хотим понять как выглядят кодовые слова.

- Код — вектор значений функции $f(x_1, ..., x_m) \in \mathrm{RM}(r, m)$, причём $\deg f \le r$.
- Разделим функцию по x_1: $f(x_1, ..., x_m) = g(x_2, ..., x_m) + x_1 h(x_2, ..., x_m)$.
- Заметим, что $\deg f \le r$, а значит $\deg g \le r$ и $\deg h \le r - 1$.

1. Порядок очевидно не больше r, потому что это условие для включения в пространство кодов $\mathrm{RM}(r, m)$.
2. Теперь у нас есть две функции от меньшего числа аргументов. Очевидно, так можно сделать всегда, когда m > 1.

Конструкция Плоткина: таблица истинности

Ранее: $f(x_1, ..., x_m) = g(x_2, ..., x_m) + x_1 h(x_2, ..., x_m)$.

- Заметим, что таблица истинности f состоит из двух частей: при $x_1 = 0$ и при $x_1 = 1$.

$$\mathrm{Eval}(f) = \begin{pmatrix} \mathrm{Eval}^{[x_1=0]}(f) \\ \mathrm{Eval}^{[x_1=1]}(f) \end{pmatrix}$$

- Причём $\mathrm{Eval}^{[x_1=0]}(f) = \mathrm{Eval}(g)$, а $\mathrm{Eval}^{[x_1=0]}(f) \oplus \mathrm{Eval}^{[x_1=1]}(f) = \mathrm{Eval}(h)$.
- Таким образом, $\mathrm{Eval}(f) = (\mathrm{Eval}(g) \mid \mathrm{Eval}(g) \oplus \mathrm{Eval}(h))$.

1. Теперь рассмотрим те же функции, но со стороны их таблиц истинности. Нам же интересны именно коды, а они как раз очень тесно связаны с этими таблицами.
2. Про обозначения: $\mathrm{Eval}(f)$ — таблица для всей функции (вектор значений, если точнее), $\mathrm{Eval}^{[x_1=0]}(f)$ — кусок таблицы при $x_1 = 0$, $\mathrm{Eval}^{[x_1=1]}(f)$ — кусок таблицы при $x_1 = 1$. Они нам после этого доказательства больше не понадобятся.
3. Это всё следует из ранее полученного утверждения. Если мы подставим $x_1 = 0$, то останется только g — первое равенство очевидно. Если же мы рассмотрим $\mathrm{Eval}^{[x_1=1]}(f)$, то получим $\mathrm{Eval}(g+h)$, но если туда прибавить ещё раз $\mathrm{Eval}(g)$, то останется только $\mathrm{Eval}(h)$ (поскольку $1+1=0$ в $\mathbb{F}_2$) — получили второе равенство.
4. Палочка по центру — конкатенация векторов.

Конструкция Плоткина: вывод

Если дана $f(x_1, ..., x_m)$, причём $\deg f \le r$, то можно её разделить:

$$f(x_1, ..., x_m) = g(x_2, ..., x_m) + x_1 h(x_2, ..., x_m)$$

Также известно, что
$\mathrm{Eval}(f) = (\mathrm{Eval}(g) \mid \mathrm{Eval}(g) \oplus \mathrm{Eval}(h))$.

Заметим, что $\mathrm{Eval}(f)$ – кодовое слово (как и для g и h).
Тогда:

$$\begin{array}{ll} c = \mathrm{Eval}(f) \in \mathrm{RM}(r, m) & (\text{т.к. } \deg f \le r) \\ u = \mathrm{Eval}(g) \in \mathrm{RM}(r, m-1) & (\text{т.к. } \deg g \le r) \\ v = \mathrm{Eval}(h) \in \mathrm{RM}(r-1, m-1) & (\text{т.к. } \deg h \le r-1) \end{array}$$

1. Теперь собираем всё это в одно важное утверждение.
2. Причём мы уже знаем, что $\deg g \le r$ и $\deg h \le r-1$, если $\deg f \le r$
3. Напомню, что $\mathrm{RM}(r, m)$ включает в себя **все** функции (их таблицы истинности, если точнее) от m аргументов и степени не выше r. Очевидно, наши годятся.

Конструкция Плоткина

Код Рида-Маллера
Введение
Кодирование
Свойства кода
Конструкция Плоткина
Минимальное расстояние
Параметры
Декодирование
Пара слов о синдромах
Алгоритм Рида
Пример
Домашнее задание
Источники

Теорема

Для всякого кодового слова $c \in \mathrm{RM}(r, m)$ можно найти $u \in \mathrm{RM}(r, m-1)$ и $v \in \mathrm{RM}(r-1, m-1)$, такие что $c = (u \mid u+v)$.

1. Что здесь важно отметить — оба наших новых кодовых слова u, v получились «меньше», чем исходное c.
Это позволяет, во-первых, устраивать индукцию, чем мы скоро и займёмся. Во-вторых, это позволяет легко строить большие порождающие матрицы, но мы этим не будем заниматься.

Минимальное расстояние

Код Рида-Маллера
Введение
Кодирование
Свойства кода
Конструкция Плоткина
Минимальное расстояние
Параметры
Декодирование
Пара слов о синдромах
Алгоритм Рида
Пример
Домашнее задание
Источники

Хотим найти минимальное расстояние для кода $\mathrm{RM}(r, m)$

$$d = \min_{c \in C, c \neq 0} w(c)$$

Предположим, что $d = 2^{m-r}$ и докажем по индукции.
База: $\mathrm{RM}(0, m)$ — единственный бит повторён 2^m раз.
Очевидно, $w(\underbrace{11...1}_{2^m}) = 2^m = 2^{m-0} \ge 2^{m-r}$.
Гипотеза: Если $v \in \mathrm{RM}(r-1, m-1)$, то $w(v) \ge 2^{m-r}$.
Шаг: Хотим доказать для $c \in \mathrm{RM}(r, m)$.

$$w(c) \overset{(1)}{=} w((u \mid u \oplus v)) \overset{(2)}{=} w(u) + w(u \oplus v) \ge \\ \overset{(3)}{\ge} w(u) + (w(v) - w(u)) = w(v) \overset{IH}{\ge} 2^{m-r} \blacksquare$$

Хотим найти минимальное расстояние для кода $\mathrm{RM}(r, m)$

$$d = \min_{c \in C, c \neq 0} w(c)$$

Предположим, что $d = 2^{m-r}$ и докажем по индукции.
База: $\mathrm{RM}(0, m)$ — единственный бит повторён 2^m раз.
Очевидно, $w(\underbrace{11\ldots1}_{2^m}) = 2^m = 2^{m-0} \geq 2^{m-r}$.
Гипотеза: Если $v \in \mathrm{RM}(r-1, m-1)$, то $w(v) \geq 2^{m-r}$.
Шаг: Хотим доказать для $c \in \mathrm{RM}(r, m)$.

$$w(c) \overset{(1)}{=} w((u \mid u \oplus v)) \overset{(2)}{=} w(u) + w(u \oplus v) \geq$$
$$\overset{(3)}{\geq} w(u) + (w(v) - w(u)) = w(v) \overset{IH}{\geq} 2^{m-r} \blacksquare$$

1. Случай $\mathrm{RM}(0, m)$ мы разбирали раньше, но я напомню. Здесь длина сообщения равна $k = \sum_{i=0}^{r} C_m^i = C_m^0 = 1$, а длина кода $n = 2^m$. Причём мы просто берём один бит и повторяем его 2^m раз (в таблице истинности).
Замечу, что не рассматриваю второй случай $w(\texttt{00...0})$, поскольку он нам не нужен для расчёта минимального расстояния. Вариант с нулевым вектором явно выкидывается, см. определение d выше.
2. Теперь немного объяснений.
Переход (1): используем конструкцию Плоткина, чтобы разбить c на конкатенацию двух кодовых слов поменьше.
Переход (2): $w((x \mid y)) = w(x) + w(y)$. Вес это всего лишь число ненулевых элементов, поэтому нет разницы как мы будем группировать части вектора.
Переход (3): $w(u \oplus v) \geq w(v) - w(u)$. Если у нас в v стоит $w(v)$ бит, то прибавив к нему u, мы сможем изменить (обнулить) не больше $w(u)$ бит. Возможно появится больше единиц, но нас интересует нижняя граница.
Переход (IH): предположение индукции в чистом виде.

Свойства и параметры

Для бинарного кода $\mathrm{RM}(r, m)$:

- $r \leq m$
- Длина кода: 2^m
- Длина сообщения: $k = \sum_{i=0}^{r} C_m^i$
- Минимальное расстояние: $d = 2^{m-r}$
- Корректирующая способность: $t = 2^{m-r-1} - 1$
- Существует порождающая матрица G для кодирования
- Проверочная матрица H совпадает с порождающей для $\mathrm{RM}(m-r-1, m)$

Для бинарного кода $\mathrm{RM}(r, m)$:
- $r \leq m$
- Длина кода: 2^m
- Длина сообщения: $k = \sum_{i=0}^{r} C_m^i$
- Минимальное расстояние: $d = 2^{m-r}$
- Корректирующая способность: $t = 2^{m-r-1} - 1$
- Существует порождающая матрица G для кодирования
- Проверочная матрица H совпадает с порождающей для $\mathrm{RM}(m-r-1, m)$

1. Теперь можно подвести итоги исследования свойств.
2. , поскольку $t = \left\lfloor \frac{d-1}{2} \right\rfloor = \left\lfloor \frac{2^{m-r}}{2} - \frac{1}{2} \right\rfloor = \left\lfloor 2^{m-r-1} - 0.5 \right\rfloor = 2^{m-r-1} - 1$
3. , она позволяет делать так: $C(x) = xG$. Но я, как обычно, её избегаю. Рекомендую почитать «Коды Рида-Маллера: Примеры исправления ошибок», если интересно.
4. , но это я это доказывать не собираюсь. Однако доказательство можно найти в «Reed-Muller Codes: Theory and Algorithms», раздел Duality.

Как линейный код

Этот код является линейным кодом, к нему применимы все обычные (и неэффективные методы):

- Перебор по всему пространству кодовых слов в поисках ближайшего.
- С использованием синдромов: $s = rH^T$.

Этот код является линейным кодом, к нему применимы все обычные (и неэффективные методы):
- Перебор по всему пространству кодовых слов в поисках ближайшего.
- С использованием синдромов: $s = rH^T$.

1. Этот способ применим ко всем кодам, но никто в здравом уме им не пользуется.
2. Здесь s — синдром, r — полученное сообщение, H — проверочная матрица. Этот метод обычен для линейных кодов.
3. Эти способы нужно иметь в виду, но о них было рассказано и без меня, так что я их пропущу.

Синдромы и как их использовать

Пусть у нас в полученном сообщении r есть ошибка e. Тогда $r = v + e$, где v — кодовое слово, которое крайне легко можно декодировать. Получается, что $s = rH^T = (v + e)H^T = vH^T + eH^T = eH^T$, поскольку $vH^T = 0$ (есть такое свойство). Мы можем перебрать всевозможные ошибки (e), для каждой посчитать синдром и записать всё это в таблицу. Тогда, чтобы восстановить сообщение, нужно посчитать синдром, по таблице найти ошибку и исправить её.

Пусть у нас в полученном сообщении r есть ошибка e. Тогда $r = v + e$, где v — кодовое слово, которое крайне легко можно декодировать. Получается, что $s = rH^T = (v+e)H^T = vH^T + eH^T = eH^T$, поскольку $vH^T = 0$ (есть такое свойство). Мы можем перебрать всевозможные ошибки (e), для каждой посчитать синдром и записать всё это в таблицу. Тогда, чтобы восстановить сообщение, нужно посчитать синдром, по таблице найти ошибку и исправить её.

1. Я не стал включать это в презентацию, но вообще-то говоря метод полезный, так что пусть будет здесь.
2. Источник: `https://ru.wikipedia.org/wiki/Линейный_код`

Определения

1. Пусть $A \subseteq \{1, \ldots, m\}$ для $m \in \mathbb{N}$
2. Подпространство $V_A \subseteq \mathbb{F}_2^m$, которое обнуляет все v_i, если $i \notin A$: $V_A = \{v \in \mathbb{F}_2^m : v_i = 0 \ \forall i \notin A\}$
3. Аналогично для $V_{\bar{A}}$, где $\bar{A} = \{1, \ldots, m\} \setminus A$: $V_{\bar{A}} = \{v \in \mathbb{F}_2^m : v_i = 0 \ \forall i \in A\}$

Пример:

- Пусть $m = 3, A = \{1, 2\}$, тогда...
- $\mathbb{F}_2^m = \{\texttt{000}, \texttt{001}, \texttt{010}, \texttt{011}, \texttt{100}, \texttt{101}, \texttt{110}, \texttt{111}\}$
- $V_A = \{\texttt{000}, \texttt{010}, \texttt{100}, \texttt{110}\}$ $(v_3 = 0 \, \forall v)$
- $\bar{A} = \{1, 2, 3\} \setminus A = \{3\}$
- $V_{\bar{A}} = \{\texttt{000}, \texttt{001}\}$ $(v_1 = v_2 = 0 \, \forall v)$

1. Начать стоит с нескольких определений, без которых алгоритм Рида объяснить не получится.
2. — все 8 векторов этого пространства
3. — обнулилась третья позиция, первые две остались
4. — осталась только третья позиция, остальные обнулились.

Смежные классы

Код Рида-Маллера
Введение
Кодирование
Свойства кода
Конструкция Плоткина
Минимальное расстояние
Параметры
Декодирова-ние
Пара слов о синдромах
Алгоритм Рида
Пример
Домашнее задание
Источники

Если фиксировано $V_A \subseteq \mathbb{F}_2^m$, то для каждого $b \in \mathbb{F}_2^m$ существует смежный класс $V_A + b$:

$$(V_A + b) = \{v + b \mid v \in V_A\}$$

Утверждается, что если брать $b \in V_{\bar{A}}$, то полученные смежные классы будут все различны (и это будут все смежные классы).

1. Почему все смежные классы $(V_A + b)$ можно получить именно перебором $b \in V_{\bar{A}}$ можно найти в разделе «Дополнительные доказательства» из пдфки

Алгоритм Рида для кода $\mathrm{RM}(r, m)$

Код Рида-Маллера
Введение
Кодирование
Свойства кода
Конструкция Плоткина
Минимальное расстояние
Параметры
Декодирова-ние
Пара слов о синдромах
Алгоритм Рида
Пример
Домашнее задание
Источники

Декодирует сообщение u, если использовался $\mathrm{RM}(r, m)$.
Для $\mathrm{RM}(2, 2)$: $f(x_1, x_2) = u_{1,2}x_1x_2 + u_1x_1 + u_2x_2 + u_4$.

Data: vector $y = (y_z \in \mathbb{F}_2 \mid z \in \mathbb{F}_2^m)$
$t = r$
while $t \geq 0$
  foreach $A \subseteq \{1, \ldots, m\}$ *with* $|A| = t$
    $c = 0$
    foreach $b \in V_{\bar{A}}$
      $c \mathrel{+}= \left(\sum_{z \in (V_A + b)} y_z \right) \mod 2$
    $u_A \leftarrow \mathbb{1}[c \geq 2^{m-t-1}]$
  $y \mathrel{-}= \mathrm{Eval}\left(\sum_{\substack{A \subseteq \{1,\ldots,m\} \\ |A| = t}} u_A \prod_{i \in A} x_i \right)$
  $t \mathrel{-}= 1$

На вход поступает бинарный вектор y длины 2^m. Это вектор значений функции, возможно с ошибками (но их не больше, чем $t = 2^{m-r-1} - 1$).

1. Теперь, наконец, сам алгоритм Рида с объяснением, что тут происходит. Почему он именно такой и почему это работает — см. раздел (на русском) «Reed's Algorithm: Unique decoding up to half the code distance» [??] в пдфке.
2. Цель — восстановить все коэффициенты при многочлене вида $f(x_1, \dots, x_m) = u_\varnothing + u_1 x_1 + x_2 x_2 + \dots + u_{1,2,\dots,r} x_{1,2,\dots,r}$, где $\deg f \le r$. Обратите внимание, что для индексов при u используются подмножества $A \subseteq \{1, \dots, m\}, |A| \le r$, причём каждый u_A умножается на моном $\prod_{i \in A} x_i$.

Алгоритм Рида для кода $\mathrm{RM}(r, m)$

Декодирует сообщение u, если использовался $\mathrm{RM}(r, m)$.
Для $\mathrm{RM}(2, 2)$: $f(x_1, x_2) = u_{1,2} x_1 x_2 + u_1 x_1 + u_2 x_2 + u_4$.

Data: vector $y = (y_z \in \mathbb{F}_2 \mid z \in \mathbb{F}_2^m)$
$t = r$
while $t \ge 0$
  foreach $A \subseteq \{1, \dots, m\}$ *with* $|A| = t$
    $c = 0$
    foreach $b \in V_{\bar{A}}$
      $c \mathrel{+}= \left(\sum_{z \in (V_A + b)} y_z \right) \bmod 2$
    $u_A \leftarrow \mathbf{1}\left[c \ge 2^{m-t-1}\right]$
  $y \mathrel{-}= \mathrm{Eval}\left(\sum_{\substack{A \subseteq \{1,\dots,m\} \\ |A| = t}} u_A \prod_{i \in A} x_i \right)$
  $t \mathrel{-}= 1$

Будем восстанавливать сначала коэффициенты u_A при старших степенях, потом поменьше и так пока не восстановим их все. Начинаем с $t = r$.

1. Теперь, наконец, сам алгоритм Рида с объяснением, что тут происходит. Почему он именно такой и почему это работает — см. раздел (на русском) «Reed's Algorithm: Unique decoding up to half the code distance» [??] в пдфке.

Алгоритм Рида для кода $\mathrm{RM}(r, m)$

Декодирует сообщение u, если использовался $\mathrm{RM}(r, m)$.
Для $\mathrm{RM}(2, 2)$: $f(x_1, x_2) = u_{1,2} x_1 x_2 + u_1 x_1 + u_2 x_2 + u_4$.

Data: vector $y = (y_z \in \mathbb{F}_2 \mid z \in \mathbb{F}_2^m)$
$t = r$
while $t \ge 0$
  foreach $A \subseteq \{1, \dots, m\}$ *with* $|A| = t$
    $c = 0$
    foreach $b \in V_{\bar{A}}$
      $c \mathrel{+}= \left(\sum_{z \in (V_A + b)} y_z \right) \bmod 2$
    $u_A \leftarrow \mathbf{1}\left[c \ge 2^{m-t-1}\right]$
  $y \mathrel{-}= \mathrm{Eval}\left(\sum_{\substack{A \subseteq \{1,\dots,m\} \\ |A| = t}} u_A \prod_{i \in A} x_i \right)$
  $t \mathrel{-}= 1$

Хотим восстановить все коэффициенты при мономах степени t. Для этого перебираем все $A, |A| = t$ и для каждого восстанавливаем коэффициент u_A при $x_{A_1} x_{A_2} \dots x_{A_t}$.

1. Теперь, наконец, сам алгоритм Рида с объяснением, что тут происходит. Почему он именно такой и почему это работает — см. раздел (на русском) «Reed's Algorithm: Unique decoding up to half the code distance» [??] в пдфке.

Алгоритм Рида для кода $\mathrm{RM}(r, m)$

Декодирует сообщение u, если использовался $\mathrm{RM}(r, m)$.
Для $\mathrm{RM}(2, 2)$: $f(x_1, x_2) = u_{1,2} x_1 x_2 + u_1 x_1 + u_2 x_2 + u_4$.

Data: vector $y = (y_z \in \mathbb{F}_2 \mid z \in \mathbb{F}_2^m)$
$t = r$
while $t \ge 0$
  foreach $A \subseteq \{1, \dots, m\}$ *with* $|A| = t$
    $c = 0$
    foreach $b \in V_{\bar{A}}$
      $c \mathrel{+}= \left(\sum_{z \in (V_A + b)} y_z \right) \bmod 2$
    $u_A \leftarrow \mathbf{1}\left[c \ge 2^{m-t-1}\right]$
  $y \mathrel{-}= \mathrm{Eval}\left(\sum_{\substack{A \subseteq \{1,\dots,m\} \\ |A| = t}} u_A \prod_{i \in A} x_i \right)$
  $t \mathrel{-}= 1$

Чтобы восстановить коэффициент, нужно перебрать все смежные классы вида $(V_A + b)$:

$$V_A = \{v \in \mathbb{F}_2^m : v_i = 0\, \forall i \notin A\}$$
$$b \in \{v \in \mathbb{F}_2^m : v_i = 0\, \forall i \in A\}$$

1. Теперь, наконец, сам алгоритм Рида с объяснением, что тут происходит. Почему он именно такой и почему это работает — см. раздел (на русском) «Reed's Algorithm: Unique decoding up to half the code distance» [??] в пдфке.

Алгоритм Рида для кода $\mathrm{RM}(r, m)$

Декодирует сообщение u, если использовался $\mathrm{RM}(r, m)$.
Для $\mathrm{RM}(2, 2)$: $f(x_1, x_2) = u_{1,2}x_1x_2 + u_1x_1 + u_2x_2 + u_4$.

Data: vector $y = (y_z \in \mathbb{F}_2 \mid z \in \mathbb{F}_2^m)$
$t = r$
while $t \geq 0$
  foreach $A \subseteq \{1, \ldots, m\}$ *with* $|A| = t$
    $c = 0$
    foreach $b \in V_{\bar{A}}$
      $c \mathrel{+}= \left(\sum_{z \in (V_A + b)} y_z\right) \bmod 2$
    $u_A \leftarrow \mathbf{1}[c \geq 2^{m-t-1}]$
  $y \mathrel{-}= \mathrm{Eval}\left(\sum_{\substack{A \subseteq \{1,\ldots,m\} \\ |A| = t}} u_A \prod_{i \in A} x_i\right)$
  $t \mathrel{-}= 1$

Считаем количество (c) смежных классов, в которых $\sum_{z \in (V_A + b)} y_z = 1 \pmod 2$. Пороговое значение ($2^{m-t-1}$) здесь — половина от числа смежных классов. Таким образом, если большинство сумм дало 1, то $u_A = 1$, иначе $u_A = 0$.

1. Теперь, наконец, сам алгоритм Рида с объяснением, что тут происходит. Почему он именно такой и почему это работает — см. раздел (на русском) «Reed's Algorithm: Unique decoding up to half the code distance» [??] в пдфке.
2. Если это количество больше порогового значения, то считаем, что $u_A = 1$, иначе же $u_A = 0$.

Алгоритм Рида для кода $\mathrm{RM}(r, m)$

Декодирует сообщение u, если использовался $\mathrm{RM}(r, m)$.
Для $\mathrm{RM}(2, 2)$: $f(x_1, x_2) = u_{1,2}x_1x_2 + u_1x_1 + u_2x_2 + u_4$.

Data: vector $y = (y_z \in \mathbb{F}_2 \mid z \in \mathbb{F}_2^m)$
$t = r$
while $t \geq 0$
  foreach $A \subseteq \{1, \ldots, m\}$ *with* $|A| = t$
    $c = 0$
    foreach $b \in V_{\bar{A}}$
      $c \mathrel{+}= \left(\sum_{z \in (V_A + b)} y_z\right) \bmod 2$
    $u_A \leftarrow \mathbf{1}[c \geq 2^{m-t-1}]$
  $y \mathrel{-}= \mathrm{Eval}\left(\sum_{\substack{A \subseteq \{1,\ldots,m\} \\ |A| = t}} u_A \prod_{i \in A} x_i\right)$
  $t \mathrel{-}= 1$

Затем мы вычитаем из y (вектор значений функции) всё найденное на этой итерации, после чего переходим к мономам меньшей степени. Повторять до восстановления всех коэффициентов.

1. Теперь, наконец, сам алгоритм Рида с объяснением, что тут происходит. Почему он именно такой и почему это работает — см. раздел (на русском) «Reed's Algorithm: Unique decoding up to half the code distance» [??] в пдфке.

Пример

Ранее: `101` кодируется как `1100` при помощи $\mathrm{RM}(1, 2)$

$$\texttt{101} \rightsquigarrow (f(x_1, x_2) = x_1 + 1) \rightsquigarrow \begin{array}{|cc|c|} x_1 & x_2 & f \\ \hline 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{array} \rightsquigarrow \begin{array}{rl} y_{\texttt{00}} = & 1 \\ y_{\texttt{01}} = & 1 \\ y_{\texttt{10}} = & 0 \\ y_{\texttt{11}} = & 0 \end{array} \rightsquigarrow \texttt{1100}$$

1. Как происходит кодирование, схематически:

Пример

Ранее: `101` кодируется как `1100` при помощи $\mathrm{RM}(1,2)$
Положим $y_{00} = 1, y_{01} = 1, y_{10} = 0, y_{11} = 0$
Здесь $m = 2$, значит $A \subseteq \{1,2\}$. Причём $r = 1$, т.е. $|A| \leq 1$.

Шаг 1/3: $t = 1, A = \{1\}$

- Здесь $V_A = \{00, 10\}$, $V_{\bar{A}} = \{00, 01\}$.
  Нужно рассмотреть два смежных класса.
- $(V_A + 00) = \{00, 10\}$, сумма: $y_{00} + y_{10} = 1 + 0 = 1$
- $(V_A + 01) = \{01, 11\}$, сумма: $y_{01} + y_{11} = 1 + 0 = 1$
- Итого: $u_A = u_{\{1\}} = 1$

1. Теперь начинаем декодирование.
2. — по одному на каждый вектор из $V_{\bar{A}}$

Пример

Ранее: `101` кодируется как `1100` при помощи $\mathrm{RM}(1,2)$
Положим $y_{00} = 1, y_{01} = 1, y_{10} = 0, y_{11} = 0$
Здесь $m = 2$, значит $A \subseteq \{1,2\}$. Причём $r = 1$, т.е. $|A| \leq 1$.

Шаг 2/3: $t = 1, A = \{2\}$

- Здесь $V_A = \{00, 01\}$, $V_{\bar{A}} = \{00, 10\}$.
  Нужно рассмотреть два смежных класса
- $(V_A + 00) = \{00, 01\}$, сумма: $y_{00} + y_{01} = 1 + 1 = 0$
- $(V_A + 10) = \{10, 11\}$, сумма: $y_{10} + y_{11} = 0 + 0 = 0$
- Итого: $u_A = u_{\{2\}} = 0$

1. — по одному на каждый вектор из $V_{\bar{A}}$.

Пример

Ранее: `101` кодируется как `1100` при помощи $\mathrm{RM}(1,2)$
Положим $y_{00} = 1, y_{01} = 1, y_{10} = 0, y_{11} = 0$
Здесь $m = 2$, значит $A \subseteq \{1,2\}$. Причём $r = 1$, т.е. $|A| \leq 1$.

Перед переходом к $t = 0$, нужно вычесть из y вектор значений следующей функции:

$$g(x_1, x_2) = u_{\{1\}} x_1 + u_{\{2\}} x_2 = 1 x_1 + 0 x_2 = x_1$$

Вычислим $\mathrm{Eval}(g)$:

x_1	x_2	$g(x_1,x_2)$
0	0	0
0	1	0
1	0	1
1	1	1

Тогда $y \leftarrow y - \mathrm{Eval}(g) = \texttt{1100} \oplus \texttt{0011} = \texttt{1111}$.

1. Здесь мы берём все u, полученные при $t = 1$, домножаем каждую на соответствущие ей x-ы и получаем функцию от m переменных.
2. Очень важно, чтобы у вас во всех таблицах истинности (в т.ч. той, которая использовалась при кодировании для получения y) был одинаковый порядок строк. Иначе чуда не выйдет.
3. Полезно заметить, что в $\mathbb{F}_2$ сложение и вычитание — одно и то же.

Продолжение примера: $t = 0$

Теперь $y_{00} = 1, y_{01} = 1, y_{10} = 1, y_{11} = 1$

Шаг 3/3: $t = 0, A = \varnothing$

- Здесь $V_A = \{\texttt{00}\}$, но $V_{\bar{A}} = \{\texttt{00}, \texttt{01}, \texttt{10}, \texttt{11}\}$. Нужно рассмотреть **четыре** смежных класса.
- $(V_A + \texttt{00}) = \{\texttt{00}\}$, сумма: $y_{00} = 1$
- $(V_A + \texttt{01}) = \{\texttt{01}\}$, сумма: $y_{01} = 1$
- $(V_A + \texttt{10}) = \{\texttt{10}\}$, сумма: $y_{10} = 1$
- $(V_A + \texttt{11}) = \{\texttt{11}\}$, сумма: $y_{11} = 1$
- Итого: $u_A = u_\varnothing = 1$

Продолжение примера: $t = 0$

Теперь $y_{00} = 1, y_{01} = 1, y_{10} = 1, y_{11} = 1$

Получили $u_{\{1\}} = 1, u_{\{2\}} = 0, u_\varnothing = 1$.
Это значит, что исходный многочлен был таков:

$$f(x_1, x_2) = u_{\{1\}}x_1 + u_{\{2\}}x_2 + u_\varnothing = x_1 + 1,$$

а исходное сообщение: `101`, как и ожидалось.

Время работы

Утверждается, что время работы алгоритма — $O(n \log^r n)$, где $n = 2^m$ — длина кода.

Домашнее задание

Вариант 1

1. Закодировать сообщение: `1001`.
2. Декодировать код, если ошибок нет: `1010`, использовался $\mathrm{RM}(1, 2)$.
3. Декодировать код, полученный с ошибками: `1101 1010`, использовался $\mathrm{RM}(1, 3)$

Вариант 2

1. Закодировать сообщение: `0101`.
2. Декодировать код, если ошибок нет: `0110`, использовался $\mathrm{RM}(1, 2)$.
3. Декодировать код, полученный с ошибками: `1111 0100`, использовался $\mathrm{RM}(1, 3)$

1. Замечание: каких-либо требований на методы решения нет, но если используете код — приложите его. Различных способов решить существует больше одного.
Номер варианта можете определять как $1 + ((5n + 98) \bmod 2)$, но главное напишите его и своё имя.

Источники

1. `https://arxiv.org/pdf/2002.03317.pdf` — великолепный обзор, очень рекомендую.
2. `http://dha.spb.ru/PDF/ReedMullerExamples.pdf` — очень хорошо и подробно, но используется подход через матрицы, а не через полиномы, а это не весело.
3. `https://en.wikipedia.org/wiki/Reed–Muller_code` — кратко, чётко, понятно, но не описано декодирование.
4. `https://ru.bmstu.wiki/Коды_Рида-Маллера` — в целом всё есть, но написано очень непонятно;