Код Рида-Маллера

Илья Коннов

Факультет компьютерных наук

Высшая Школа Экономики

11 февраля 2022 г.

Код Рида-Маллера

Илья Коннов

Факультет компьютерных наук

Высшая Школа Экономики

11 февраля 2022 г.

1. Если вы смотрите презентацию, то на сером фоне справа иногда видны некоторые ценные комментарии, для которых поля слайда оказались слишком узки. Если вы читаете pdf-ку, то эти комментарии уже находятся в самом подходящем для них месте в тексте (а в правых полях видны заголовки слайдов). Если вы смотрите мой доклад и видите этот текст, то что-то пошло серьёзно не так. Да, у этого одного файла есть три разные версии.

Введение

Описаны Дэвидом Маллером (автор идеи) и Ирвингом Ридом (автор метода декодирования) в сентябре 1954 года. Обозначаются 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. $\mathbb{Z}_2$.
Соглашение: сложение векторов $u, v \in \mathbb{Z}_2^n$ будем обозначать как $u \oplus v = (u_1 + v_1, u_2 + v_2, ..., u_n + v_n)$.

Код Рида-Маллера
Введение
2022-02-11
Введение
Описаны Дэвидом Маллером (автор идеи) и Ирвингом Ридом (автор метода декодирования) в сентябре 1954 года. Обозначаются 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. $\mathbb{Z}_2$.
Соглашение: сложение векторов $u, v \in \mathbb{Z}_2^n$ 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

И при помощи многочлена Жегалкина:

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

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

И 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 $m = 2$:
$f(x_1, x_2) = c_1 \cdot x_1 x_2 + c_2 \cdot x_1 + c_3 \cdot x_2 + c_4 \cdot 1$
Всего $n = 2^m$ коэффициентов для описания каждой функции.

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 $m = 2$:
$f(x_1, x_2) = c_1 \cdot x_1 x_2 + c_2 \cdot x_1 + c_3 \cdot x_2 + c_4 \cdot 1$
Всего $n = 2^m$ коэффициентов для описания каждой функции.

Функции небольшой степени

Рассмотрим функции, степень многочленов которых не больше r:

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

Каждую можно записать следующим образом:

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

В каждом произведении используется не больше r переменных.

Сколько тогда всего коэффициентов используется?

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

Рассмотрим функции, степень многочленов которых не больше r:

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

Каждую можно записать следующим образом:

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

В каждом произведении используется не больше r переменных.

Сколько тогда всего коэффициентов используется?

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

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 $(x + y + z)$, затем произведения одночленов $(xy + yz + xz)$ и т.д. вплоть до r множителей. Тогда легко видеть, почему k именно такое: мы складываем все возможные перестановки сначала для 0 переменных, потом для одной, двух, и так до всех r

Идея кодирования

Пусть каждое сообщение (длины k) — коэффициенты некоторого многочлена от m переменных степени не больше r.
Тогда мы можем его представить при помощи 2^n бит, подставив все возможные комбинации переменных (ведь рассматриваем многочлены над $\mathbb{Z}_2$).
Таким образом получим таблицу истинности, из которой позднее сможем восстановить исходный многочлен, а вместе с ним и сообщение.

Пусть каждое сообщение (длины k) — коэффициенты некоторого многочлена от m переменных степени не больше r.
Тогда мы можем его представить при помощи 2^n бит, подставив все возможные комбинации переменных (ведь рассматриваем многочлены над $\mathbb{Z}_2$).
Таким образом получим таблицу истинности, из которой позднее сможем восстановить исходный многочлен, а вместе с ним и сообщение.

Пример

- $r = 1$ (степень многочлена), $m = 2$ (переменных). Это $\mathrm{RM}(1, 2)$.
- Тогда наш многочлен: $f(x, y) = c_1 x + c_2 y + c_3$.
- Сообщение: `101`, тогда $f(x, y) = x + 0 + 1$.
- Подставим всевозможные комбинации:

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

- Получили код: `1100`.

- $r = 1$ (степень многочлена), $m = 2$ (переменных). Это $\mathrm{RM}(1, 2)$.
- Тогда наш многочлен: $f(x, y) = c_1 x + c_2 y + c_3$.
- Сообщение: `101`, тогда $f(x, y) = x + 0 + 1$.
- Подставим всевозможные комбинации:

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

- Получили код: `1100`.

Декодирование когда потерь нет

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

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

- Подстановками в $f(x,y) = c_1 x + c_2 y + c_3$ получим СЛАУ.

$$\begin{cases} c_3 = 1 \\ c_2 + c_3 = 1 \\ c_1 + c_3 = 0 \\ c_1 + c_2 + c_3 = 0 \end{cases}$$

- $c_1 = 1, c_2 = 0, c_3 = 1$, исходное сообщение: `101`.

- Мы получили код: 1100
- Представим таблицу истинности.
- Подстановками в $f(x,y) = c_1 x + c_2 y + c_3$ получим СЛАУ.
- $c_1 = 1, c_2 = 0, c_3 = 1$, исходное сообщение: 101.

1. Или как можно декодировать код, в самых простых ситуациях. Этот пример — продолжение предыдущего.

Доказательство линейности

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

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

где $p_x(a_i)$ — соответствующий сообщению a_i многочлен. Перебирая все a_i получаем упорядоченный набор его значений. Это и будет кодом.
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.

Пусть $C(x)$ кодирует сообщение $x \in \mathbb{Z}_2^k$ в код $C(x) \in \mathbb{Z}_2^m$.
$C(x) = (p_x(a_i) \mid a_i \in \mathbb{Z}_2^m)$
где $p_x(a_i)$ — соответствующий сообщению a_i многочлен. Перебирая все a_i получаем упорядоченный набор его значений. Это и будет кодом.
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 ($\mathbb{Z}_2^k$) в пространство слов ($\mathbb{Z}_2^m$).
   Для этого необходимо немного формализовать всё описанное раньше.
2. Напомню, что базис пространства многочленов выглядит примерно так: $1, x_1, x_2, x_1x_2$ (для двух переменных, степени не выше 2). Поскольку мы работает в поле $\mathbb{Z}_2$, здесь нету x_1^2 и x_2^2 ($a^2 = a$).
   Чтобы преобразовать сообщение в многочлен, мы берём каждый бит сообщения и умножаем его на соответствующий базисный вектор.
   Очевидно, такое преобразование будет изоморфизмом. Именно поэтому $p_{(x+y)} = p_x + p_y$.
   Обратите внимание, что 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 ($\oplus$).
3. Для краткости, 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 растояние 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. Существует порождающая матрица G.
$C(x) = x_{1\times k}G_{k\times n} = c_{1\times n}$
2. Минимальное растояние будет равно минимальному весу Хемминга среди всех кодов.
$d = \min_{c\in C, c\neq 0} w(c)$
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 Хэмминга 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 — таблица истинности функции $f(x_1, ..., x_m) \in \mathrm{RM}(r, m)$, причём $\deg f \leq r$.
- Разделим функцию по x_1: $f(x_1, ..., x_m) = g(x_2, ..., x_m) + x_1 h(x_2, ..., x_m)$.
- Заметим, что $\deg f \leq r$, а значит $\deg g \leq r$ и $\deg h \leq r - 1$.

Хотим понять как выглядят кодовые слова.
Код — таблица истинности функции $f(x_1, ..., x_m) \in \mathrm{RM}(r, m)$, причём $\deg f \leq r$.
Разделим функцию по x_1: $f(x_1, ..., x_m) = g(x_2, ..., x_m) + x_1 h(x_2, ..., x_m)$.
Заметим, что $\deg f \leq r$, а значит $\deg g \leq r$ и $\deg h \leq r - 1$.

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

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

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

- 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. Здесь я очень резко ввожу обозначения для таблицы истинности, но они больше не пригодятся. Вообще-то говоря, это не матрица, а вектор длины 2^m (число аргументов), поскольку порядок аргументов всегда фиксирован и нам его хранить не нужно. В любом случае, $\mathrm{Eval}(f)$ — таблица для всей функции, $\mathrm{Eval}^{[x_1=0]}(f)$ — кусок таблицы при $x_1 = 0$, $\mathrm{Eval}^{[x_1=1]}(f)$ — кусок таблицы при $x_1 = 1$.
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 $\mathbb{Z}_2$) — получили второе равенство.

Конструкция Плоткина: вывод

Если дана $f(x_1, ..., x_m)$, причём $\deg f \leq r$, 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, h). Тогда:

$c = \mathrm{Eval}(f) \in \mathrm{RM}(r, m)$ (т.к. $\deg f \leq r$)
$u = \mathrm{Eval}(g) \in \mathrm{RM}(r, m-1)$ (т.к. $\deg g \leq r$)
$v = \mathrm{Eval}(h) \in \mathrm{RM}(r-1, m-1)$ (т.к. $\deg h \leq r-1$)

Утверждение: 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 $c = (u \mid u + v)$.

Если дана $f(x_1, ..., x_m)$, причём $\deg f \leq r$, 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, h). Тогда:
$c = \mathrm{Eval}(f) \in \mathrm{RM}(r, m)$ (т.к. $\deg f \leq r$)
$u = \mathrm{Eval}(g) \in \mathrm{RM}(r, m-1)$ (т.к. $\deg g \leq r$)
$v = \mathrm{Eval}(h) \in \mathrm{RM}(r-1, m-1)$ (т.к. $\deg h \leq r-1$)
Утверждение: 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. Теперь собираем всё это в одно важное утверждение.
2. Причём мы уже знаем, что $\deg g \leq r$ и $\deg h \leq r-1$, если $\deg f \leq r$
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.
4. Что здесь важно отметить — оба наших новых кодовых слова u, v получились «меньше», чем исходное c.
   Это позволяет, во-первых, устраивать индукцию по m, чем мы скоро и займёмся. Во-вторых, это позволяет легко строить большие порождающие матрицы, но мы этим не будем заниматься.

Минимальное расстояние

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 потворён 2^m раз.
Очевидно, $w(\underbrace{11...1}_{2^m}) = 2^m = 2^{m-0} \geq 2^{m-r}$.

Гипотеза: Если $v \in \mathrm{RM}(r-1, m-1)$, то $w(v) \geq 2^{m-r}$.
Шаг: Хотим доказать для $c \in \mathrm{RM}(r, m)$.

$$w(c) = w((u \mid u \oplus v)) \overset{(1)}{=} w(u) + w(u \oplus v) \geq$$

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

1. Случай $\mathrm{RM}(0, m)$ — очень скучный. 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 (соответсвует функции $f(x_1, ..., x_m) = 0$ или $f(x_1, ..., x_m) = 1$) 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): $w((x \mid y)) = w(x) + w(y)$. Вес это всего лишь число ненулевых элементов, поэтому нет разницы как мы будем группировать части вектора.
   Переход (2): $w(u \oplus v) \geq w(v) - w(u)$. 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 $\mathrm{RM}(r, m)$:

- $r \le m$
- 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

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

- $r \le m$
- 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

1. поскольку $t = \left\lfloor \frac{d-1}{2} \right\rfloor = \left\lfloor \frac{2^{m-r}}{2} - \frac{1}{2} \right\rfloor = \left\lfloor 2^{m-r-1} - 0.5 \right\rfloor = 2^{m-r-1} - 0.5$

Если потери есть

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

-

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