Теория информации, решения задач части 2 (линейные коды)

Table of Contents

1 2.1 Оптимальные коды для длины 6

Итак, для $i \in \{1..6\}$ мы хотим построить коды длины 6 максимизирующие минимальное расстояние d.

Для (6,1) мы хотим попробовать получить $d = r + 1 = 6$ (граница Синглтона) и это в самом деле возможно: воспользуемся код дуальный к коду с проверкой на четность:

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

Очевидно что такой код имеет $d = 6$: любые 5 столбцов – это либо первые 5 (дают в сумме шестой), либо 4 из первых и последний (дают в сумме ровно отсутствующий).

Далее рассмотрим (6,2) код со следующей проверочной матрицей:

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

В этом случае $d = 4$, недостижимость $d = 5$ можно показать прямолинейным перебором (но и интуитивно понятно, что невозможно получить 4 ЛНЗ столбца).

Аналогично для (6,3) получаем $d = 3$.

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

Далее для (6,4) предлагается рассмотреть следующую проверочную матрицу:

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

На самом деле неважно то, как строить матрицу – существует всего 4 разных вектора длины 2, поэтому матрица содержала бы дубликаты – почему бы не заполнить ее одним вектором в таком случае? Кстати, d такого кода равен 2, а еще оно проверяет, что вес кодвого слова четный и что первый бит – всегда ноль.

То же и для (6,5):

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Такой код имеет $d = 1$ и есть кодом с проверкой на четность.

Для кода (6,6) применим пример из учебника: $G = I_6$, а H неопределена поскольку $r = 0$. Такой код имеет $d = 1$.

2 2.3/2.7 Подсчет параметров кода

Дана проверочная матрица с параметрами $r = 4$, $n = 10$, то есть $k = 6$, отсюда $R = 3/5$. Поскольку все столбцы различны и ненулевые, то $d \geq 3$, собственно сумма первого, второго и последнего столбцов равна нулю, отсюда $d = 3$.

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

Несложными преобразованиями (метод Гаусса, выполнил на бумаге) она приводится к следующему виду:

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

Из чего сразу следует (поскольку $G = (I_k P)$):

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

Перейдем к задаче синдромного декодирования.

Реализация алгоритма на haskell (опущена большая часть методов реализации алгебры матриц по полю F_2).

```
syndromDecodeBuild :: [BVector] -> Map BVector BVector
syndromDecodeBuild h = flip execState mempty $ forM_ allEs $ \e -> do
```

```
    let syndrom = e `vMulM` (transpose h)
    let updateSyndrom = at syndrom ?= e
    use (at syndrom) >>= \case
        Just x  -> when (weight x > weight e) updateSyndrom
        Nothing -> updateSyndrom
  where
    n = length h
    allEs :: [BVector]
    allEs = binaryVectors (fromIntegral n)
```

В результате получили следующую таблицу синдромного декодирования:

Синдром	Вектор ошибки
[0,0,0,0]	[0,0,0,0,0,0,0,0,0,0]
[0,0,0,1]	[0,0,1,0,0,0,1,0,0,0]
[0,0,1,0]	[0,0,0,0,0,1,1,0,0,0]
[0,0,1,1]	[0,0,0,0,0,0,0,0,0,1]
[0,1,0,0]	[0,0,0,1,0,0,0,0,0,1]
[0,1,0,1]	[0,0,0,0,0,0,0,0,1,0]
[0,1,1,0]	[0,0,0,0,0,0,0,0,1,1]
[0,1,1,1]	[0,0,0,1,0,0,0,0,0,0]
[1,0,0,0]	[0,0,0,0,0,0,1,0,0,0]
[1,0,0,1]	[0,0,1,0,0,0,0,0,0,0]
[1,0,1,0]	[0,0,0,0,0,1,0,0,0,0]
[1,0,1,1]	[0,0,0,0,0,0,1,0,0,1]
[1,1,0,0]	[0,0,0,0,0,0,0,1,0,1]
[1,1,0,1]	[0,0,0,0,1,0,0,0,0,0]
[1,1,1,0]	[1,0,0,0,0,0,0,0,0,0]
[1,1,1,1]	[0,0,0,0,0,0,0,1,0,0]

3 2.5 Количество линейных кодов

Каждая проверочная матрица порождает какой-то линейный код, плюс имеем тривиальный код (n,n), не имеющий проверочной. Поэтому суммарное количество возможных кодов для (n,k) есть $2^{n(n-k)}$ если $k < n$ и еще какое-то количество способов представить $G = I_n$ в виде n ЛНЗ столбцов.

Но это маловажно. Интереснее обратить внимание на то, что, поскольку $H = (P^T I_r)$, и перестановки строк в H не несут большой разницы с точки зрения эквивалентности кодов (имеем один P^T, значит одна G, значит один базис), то можно считать что все H с одним P эквивалентны. Тогда, поскольку размер P есть $k \times r$, количество различных P есть 2^{kr}.

Если kr достаточно мало, наивный перебор P может быть эффективен для нахождения d кода.

4 2.6 Соответствия синдромы/векторы ошибок

Нетрудно применить решение 2.3/2.7 для того чтобы выявить соответствие между синдромами и векторами ошибок для кодов Хэмминга. Например, рассмотрим $r = 3$ и код (7, 4) и его синдромную таблицу:

Синдром	Вектор ошибки
[0,0,0]	[0,0,0,0,0,0,0]

Синдром	Вектор ошибки
[0,0,1]	[1,0,0,0,0,0,0]
[0,1,0]	[0,1,0,0,0,0,0]
[0,1,1]	[0,0,1,0,0,0,0]
[1,0,0]	[0,0,0,1,0,0,0]
[1,0,1]	[0,0,0,0,1,0,0]
[1,1,0]	[0,0,0,0,0,1,0]
[1,1,1]	[0,0,0,0,0,0,1]

Закономерность очевидна – если рассматривать синдром как двоичное представление десятичного числа, то это число будет индексом ошибки в векторе. Эта закономерность очевидно обобщается на любое r. Она имеет место по построению H – каждый ряд $i \in 1..r$ имеет 2^i "разбиений" на продолжительные последовательности битов (первая бьется на две части, вторая на 4, и так далее). Тогда, при подсчете синдрома $y\dot{H}^T$ мы r раз уточняем где находится единственная ошибка.

5 2.8 Обобщения двоичных полей

Я попытаюсь ответить на этот вопрос, хотя он сложный и требует более внимательного рассмотрения. Во-первых, начиная с определения расстояния между объектами, метрику Хэмминга следует заменить на иную. Интуитивно кажется, что Манхэттэнская метрика (расстояние Минковского с $p = 1$) наиболее близка к расстоянию Хэмминга.

Теорема 2.1 о количестве исправлений ошибок не должна меняться, поскольку ее доказательство оперирует абстрактной метрикой, то есть всё же $\lfloor (d-1)/2 \rfloor$ ошибок код будет исправлять.

Теорема 2.2 также не меняет смысл, если определить вес $w(x)$ как сумму расстояний от 0 до символа (в нашем случае с манхэттэнской метрикой это просто разница, так что w это сумма).

6 2.9 Циклические коды

Порождающая матрица кода полученного из (1101000) будет выглядеть так:

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

Проверочная матрица :

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

Отсюда делаем вывод, что $d = 3$. Увеличение длины g (n) засчет добавления нулей в конец приведет лишь к тому что последние столбцы P будут нулевыми. Например, для $n = 8$:

$$G_8 = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

$$H_8 = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Заметим, что на d это преобразование не повлияет, так как те же столбцы что были независимы в H будут зависимы и в H_8. Понятно, что такое строгое ограничение на форму порождающей матрицы отразится на минимальном расстоянии – циклических кодов меньше и поэтому они в среднем хуже. Лучший линейный код (7,4) имеет $d = 3$ (как рассмотренный из примера), а вот лучший (8,4) имеет $d = 4$, что уже больше.

Author: Волхов Михаил, M4139

Created: 2017-11-14 Tue 09:01

Validate