Домашнее задание №1

Лев Довжик, M3439

Вариант №62

Начальные условия

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

№1 Скорость кода

Матрица H имеет $n = 10$ столбцов и $r = n - k = 4$ строк, а значит $n = 10, k = 6, r = 4$, и скорость кода $R = \frac{k}{n} = \frac{3}{5} = 0.6\ \frac{bit}{symbol}$

№2 Минимальное расстояние кода

Заметим, что любые два столбца проверочной матрицы независимы (т.к. все столбцы различны). При этом первый, четвёртый и десятый столбец в сумме дают нулевой вектор: $\begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$.

Откуда по теореме 2.4 получаем, что минимальное расстояние $d = 3$.

№3 Расстрояние кода

3.1 Граница Хэмминга

Для нашего $(10, 6, 3)$ кода $M = 2^k,\ q = 2$ и граница Хэмминга принимает вид: $2^k \le \dfrac{2^n}{\sum\limits_{i=0}^{t} C_i^n \left(2-1\right)^i} \Rightarrow 2^k \le \dfrac{2^n}{\sum\limits_{i=0}^{t} C_i^n} \Rightarrow \sum\limits_{i=0}^{t} C_i^n \le 2^{n-k} \Rightarrow \sum\limits_{i=0}^{t} C_i^{10} \le 16$

Найдём такое максимальное d, для которого выполняется данное соотношение при $t = \lfloor \dfrac{d-1}{2} \rfloor$:

```
from scipy import special


def check_hamming(n, k, d):
    t = (d - 1) // 2
    total = 0
    for i in range(t + 1):
        total += special.comb(n, i)
    return total <= 2 ** (n - k)


d = 1
n = 10
k = 6
while check_hamming(n, k, d):
    d += 1
d -= 1
print(d)
# 4
```

Таким образом получаем, что верхняя граница для d равна 4. То есть не существует никакого двоичного линейного $(10, 6)$-кода с минимальным расстоянием больше 4.

3.2 Граница Варшамова-Гилберта

При $q = 2$ граница Варшамова-Гилберта принимает вид: $2^{n-k} > \sum\limits_{i=0}^{d-2} C_i^{n-1}(2-1)^i$. Что в нашем случае превращается в $\sum\limits_{i=0}^{d-2} C_i^9 < 16$.

Найдём максимальное d, что которого выполняется данное соотношение, что будет означать существование $(10, 6)$-кода с таким минимальным расстоянием:

```
from scipy import special


def check_varshamov(n, k, d):
    total = 0
    for i in range(d - 1):
        total += special.comb(n - 1, i)
    return total < 2 ** (n - k)


d = 1
n = 10
k = 6
while check_varshamov(n, k, d):
    d += 1
d -= 1
print(d)
# 3
```

Таким образом существует двоичный линейный $(10, 6)$-код с минимальным расстоянием $d = 3$.

3.3 Проверка оптимальности

Из вышеоисанного имеем, что для минимального расстояние оптимальное кода стоит искать в диапазоне от 3 до 4. Исследуемый код попадает в данный диапазон. К тому же по данным сайта codetables.de, а также таблицы 3.2, лучший двоичный линейный $(10, 6)$-код имеет минимальное расстояние $d = 3$, следовательно наш код является оптимальным.

№4 Порождающая матрица

Приведём проверочную матрицу к систематическому виду.

Исходная матрица: $\begin{pmatrix} 1\,1\,1\,0\,1\,0\,0\,1\,0\,1 \\ 1\,0\,0\,0\,0\,0\,1\,1\,1\,1 \\ 1\,1\,0\,1\,0\,1\,1\,1\,0\,0 \\ 0\,1\,1\,0\,0\,1\,0\,1\,0\,0 \end{pmatrix}$

Меняем 2 и 4 строки местами: $\begin{pmatrix} 1\,1\,1\,0\,1\,0\,0\,1\,0\,1 \\ 0\,1\,1\,0\,0\,1\,0\,1\,0\,0 \\ 1\,1\,0\,1\,0\,1\,1\,1\,0\,0 \\ 1\,0\,0\,0\,0\,0\,1\,1\,1\,1 \end{pmatrix}$

Прибавим 2 строку к 1 и 3: $\begin{pmatrix} 1\,0\,0\,0\,1\,1\,0\,0\,0\,1 \\ 0\,1\,1\,0\,0\,1\,0\,1\,0\,0 \\ 1\,0\,1\,1\,0\,0\,1\,0\,0\,0 \\ 1\,0\,0\,0\,0\,0\,1\,1\,1\,1 \end{pmatrix}$

Прибавим к 4 строке первые три: $\begin{pmatrix} 1\,0\,0\,0\,1\,1\,0\,0\,0\,1 \\ 0\,1\,1\,0\,0\,1\,0\,1\,0\,0 \\ 1\,0\,1\,1\,0\,0\,1\,0\,0\,0 \\ 1\,1\,0\,1\,1\,0\,0\,0\,1\,0 \end{pmatrix}$

Переупорядочим стоки, приведя к систематическому виду: $\begin{pmatrix} 1\,0\,1\,1\,0\,0\,1\,0\,0\,0 \\ 0\,1\,1\,0\,0\,1\,0\,1\,0\,0 \\ 1\,1\,0\,1\,1\,0\,0\,0\,1\,0 \\ 1\,0\,0\,0\,1\,1\,0\,0\,0\,1 \end{pmatrix}$

Значит $P^T = \begin{pmatrix} 1\,0\,1\,1\,0\,0 \\ 0\,1\,1\,0\,0\,1 \\ 1\,1\,0\,1\,1\,0 \\ 1\,0\,0\,0\,1\,1 \end{pmatrix}$, $P = \begin{pmatrix} 1\,0\,1\,1 \\ 0\,1\,1\,0 \\ 1\,1\,0\,0 \\ 1\,0\,1\,0 \\ 0\,0\,1\,1 \\ 0\,1\,0\,1 \end{pmatrix}$, а $G = \begin{pmatrix} 1\,0\,0\,0\,0\,0\,1\,0\,1\,1 \\ 0\,1\,0\,0\,0\,0\,0\,1\,1\,0 \\ 0\,0\,1\,0\,0\,0\,1\,1\,0\,0 \\ 0\,0\,0\,1\,0\,0\,1\,0\,1\,0 \\ 0\,0\,0\,0\,1\,0\,0\,0\,1\,1 \\ 0\,0\,0\,0\,0\,1\,0\,1\,0\,1 \end{pmatrix}$

В конце проверяем, что $G \cdot P^T = O_{k \times n}$

```
import numpy as np

G = np.array([[1, 0, 0, 0, 0, 0, 1, 0, 1, 1],
              [0, 1, 0, 0, 0, 0, 0, 1, 1, 0],
              [0, 0, 1, 0, 0, 0, 1, 1, 0, 0],
              [0, 0, 0, 1, 0, 0, 1, 0, 1, 0],
              [0, 0, 0, 0, 1, 0, 0, 0, 1, 1],
              [0, 0, 0, 0, 0, 1, 0, 1, 0, 1]])

H = np.array([[1, 1, 1, 0, 1, 0, 0, 1, 0, 1],
              [1, 0, 0, 0, 0, 0, 1, 1, 1, 1],
              [1, 1, 0, 1, 0, 1, 1, 1, 0, 0],
              [0, 1, 1, 0, 0, 1, 0, 1, 0, 0]])

H_t = H.transpose()
res = np.matmul(G, H_t) % 2
print(res)
# [[0 0 0 0]
#  [0 0 0 0]
#  [0 0 0 0]
#  [0 0 0 0]
#  [0 0 0 0]
```

```
#   [0  0  0  0]]
```

№5 Кодировка сообщения

Пусть $m = (1\ 1\ 0\ 1\ 0\ 0)$, тогда $c = m \cdot G = (1\ 1\ 0\ 1\ 0\ 0) \cdot \begin{pmatrix} 1\,0\,0\,0\,0\,0\,1\,0\,1\,1 \\ 0\,1\,0\,0\,0\,0\,0\,1\,1\,0 \\ 0\,0\,1\,0\,0\,0\,1\,1\,0\,0 \\ 0\,0\,0\,1\,0\,0\,1\,0\,1\,0 \\ 0\,0\,0\,0\,1\,0\,0\,0\,1\,1 \\ 0\,0\,0\,0\,0\,1\,0\,1\,0\,1 \end{pmatrix}$.

То есть $c = (1\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 1)$

№6 Информационная совокупность

Заметим, что при «систематизации» матрицы H мы нигде не перестовляли столбцы, значит нам не нужна перенумеровка стобцов в G. При этом первый $k = 6$ столбцов матрицы G образают единичную подматрицу, а следовательно линейно назависимы. Из всего вышесказанного получаем, что информационная совокупность равна $(1, 2, 3, 4, 5, 6)$

№11 Таблица синдромного декодирования

Всего у нас существует $2^r = 2^4 = 16$ синдромов. Нулевому синдрому, очевидно, соотвествует нулевой вектор ошибок. А так же, в силу того что строки H^T различны, каждому вектору ошибок веса 1 будет соотвествовать свой уникальный синдром. Итого мы построили таблицу для 11 синдромов из 16:

$$H^T = \begin{pmatrix} 1\,1\,1\,0 \\ 1\,0\,1\,1 \\ 1\,0\,0\,1 \\ 0\,0\,1\,0 \\ 1\,0\,0\,0 \\ 0\,0\,1\,1 \\ 0\,1\,1\,0 \\ 1\,1\,1\,1 \\ 0\,1\,0\,0 \\ 1\,1\,0\,0 \end{pmatrix}$$

s	e
0 0 0 0	0 0 0 0 0 0 0 0 0 0
0 0 0 1	?
0 0 1 0	0 0 0 1 0 0 0 0 0 0
0 0 1 1	0 0 0 0 0 1 0 0 0 0
0 1 0 0	0 0 0 0 0 0 0 0 1 0
0 1 0 1	?
0 1 1 0	0 0 0 0 0 0 1 0 0 0
0 1 1 1	?
1 0 0 0	0 0 0 0 1 0 0 0 0 0
1 0 0 1	0 0 1 0 0 0 0 0 0 0
1 0 1 0	?
1 0 1 1	0 1 0 0 0 0 0 0 0 0
1 1 0 0	0 0 0 0 0 0 0 0 0 1
1 1 0 1	?
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

Оставшиеся синдромы будут иметь соотвествующие им вектора ошибок веса хотя бы 2. Заметим, что синдром (0 0 0 1) получается складыванием 1 и 8 строк, (0 1 0 1) — 6 и 7, (0 1 1 1) — 5 и 8, (1 0 1 0) — 1 и 9, (1 0 1 1) — 8 и 9. Вес меньше двух они иметь не могут, а значит мы нашли минимальные вектора для данных синдромов. Итоговая таблица синдромного декодирования:

s	e
0 0 0 0	0 0 0 0 0 0 0 0 0 0
0 0 0 1	1 0 0 0 0 0 0 1 0 0
0 0 1 0	0 0 0 1 0 0 0 0 0 0
0 0 1 1	0 0 0 0 0 1 0 0 0 0
0 1 0 0	0 0 0 0 0 0 0 0 1 0
0 1 0 1	0 0 0 0 0 1 1 0 0 0
0 1 1 0	0 0 0 0 0 0 1 0 0 0
0 1 1 1	0 0 0 0 1 0 0 1 0 0
1 0 0 0	0 0 0 0 1 0 0 0 0 0
1 0 0 1	0 0 1 0 0 0 0 0 0 0
1 0 1 0	1 0 0 0 0 0 0 0 1 0
1 0 1 1	0 1 0 0 0 0 0 0 0 0
1 1 0 0	0 0 0 0 0 0 0 0 0 1
1 1 0 1	0 0 0 0 0 0 0 1 1 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

Задача 3.3

Выберем $n = 23,\ d = 13$.

Граница Хэмминга

При $t = \lfloor\frac{d-1}{2}\rfloor = 6,\ q = 2,\ M = 2^k$ граница Хэмминга принимает следующий вид: $2^k \leq \frac{2^{23}}{\sum\limits_{i=0}^{6} C_i^{23}} \Rightarrow k \leq 23 - \log_2\left(\sum\limits_{i=0}^{6} C_i^{23}\right) \Rightarrow k \leq \lfloor 23 - \log_2\left(\sum\limits_{i=0}^{6} C_i^{23}\right)\rfloor = 5$

Значит ни для какого кода с $k > 5$ не существует двоичного линейного кода с заданными параметрами

Граница Варшамова-Гилберта

В нашем случае граница Варшамова-Гилберта имеет вид: $2^{23-k} > \sum\limits_{i=0}^{13-2} C_i^{23-1}$. Преобразовав данное неравенство, получаем: $23 - k > \log_2 \left(\sum\limits_{i=0}^{11} C_i^{22} \right)$. Или же $k \leq 23 - \log_2 \left(\sum\limits_{i=0}^{11} C_i^{22} \right) \Rightarrow k \leq \lfloor 23 - \log_2 \left(\sum\limits_{i=0}^{11} C_i^{22} \right) \rfloor = 1$

Получаем для кода с $k \leq 1$ (что неудивитильно) мы точно можем построить двоичный линейный код с заданными параметрами.

Определение оптимального кода

Из всего вышесказанного получаем, что оптимальный код имеет $1 \leq k \leq 5$. При этом по данным сайта codetables.de и таблицы 3.2 при $k = 3$ минимальное расстояние $d = 12$, а при $k = 2$ имеем $d = 15$. Следовательно оптимальный код подходящий под данные парметры имеет $k = 2$.

Задача 3.4

Выберем $n = 23,\ k = 6$.

Граница Хэмминга

Для нашего $(23, 6)$-кода $M = 2^k,\ q = 2$ и граница Хэмминга принимает вид:

$$2^k \leq \frac{2^n}{\sum\limits_{i=0}^{t} C_i^n \left(2 - 1\right)^i} \Rightarrow 2^k \leq \frac{2^n}{\sum\limits_{i=0}^{t} C_i^n} \Rightarrow \sum_{i=0}^{t} C_i^n \leq 2^{n-k} \Rightarrow \sum_{i=0}^{t} C_i^{23} \leq 2^{17}$$

Найдём такое максимальное d, для которого выполняется данное соотношение при $t = \lfloor \frac{d-1}{2} \rfloor$:

```
from scipy import special
```

```
def check_hamming(n, k, d):
    t = (d - 1) // 2
    total = 0
    for i in range(t + 1):
        total += special.comb(n, i)
    return total <= 2 ** (n - k)



d = 1
n = 23
k = 6
while check_hamming(n, k, d):
    d += 1
d -= 1
print(d)
# 12
```

Значит 12 является верхней границей на минимальное расстояние расстояние для двочиного линейного $(23, 6)$-кода.

Граница Варшамова-Гилберта

При $q = 2$ граница Варшамова-Гилберта принимает вид: $2^{n-k} > \sum\limits_{i=0}^{d-2} C_i^{n-1}(2-1)^i$.

Что в нашем случае превращается в $\sum\limits_{i=0}^{d-2} C_i^{22} < 2^{17}$.

Найдём максимальное d, что которого выполняется данное соотношение, что будет означать существование $(23, 6)$-кода с таким минимальным расстоянием:

```
from scipy import special
```

```python
def check_varshamov(n, k, d):
    total = 0
    for i in range(d - 1):
        total += special.comb(n - 1, i)
    return total < 2 ** (n - k)



d = 1
n = 23
k = 6
while check_varshamov(n, k, d):
    d += 1
d -= 1
print(d)
# 8
```

Таким образом существует двоичный линейный $(23, 6)$-код с минимальным расстоянием $d = 8$.

Определение оптимального кода

Из всего вышесказанного получаем, что оптимальный код имеет $8 \leq d \leq 12$. При этом по данным сайта codetables.de и таблицы 3.2 оптимальный код имеет $d = 10$, что попадает в наши границы.