Computer Science Fundamentals

Channel Coding – Hamming Codes / Checksums

Technische Hochschule Rosenheim
Winter 2021/22
Prof. Dr. Jochen Schmidt

Overview

- Hamming Codes
- Non-binary checksums

Hamming Codes

Hamming Codes

Technische Hochschule Rosenheim

- Developed by R. Hamming (1950)
  - Binary Hamming codes are 1-error-correcting
  - with a Hamming distance of 3

- Idea
  - Mixes (even) parity bits with the original data at defined positions (powers of two)
  - Uses check bits during decoding, whose binary coding indicates
    - $= 0 \longrightarrow$ error-free transmission
    - $> 0 \longrightarrow$ the location of the erroneous bit
  - All Hamming codes have length $2^r - 1$ bits, with $r \geq 2$
    - r parity bits (which will be used to generate the check bits during decoding) at positions $2^0, 2^1, 2^2, \ldots, 2^{r-1}$.
    - $2^r - 1 - r$ data bits

(7, 4) Hamming Code

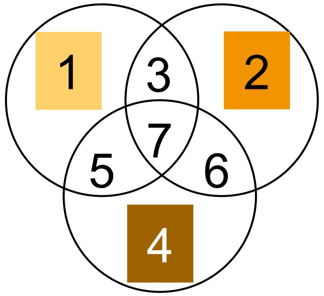
Shortest non-trivial Hamming code

- block code of length $2^3 - 1 = 7$ bits
- 3 parity bits at positions $2^0, 2^1, 2^2 \longrightarrow$ we can encode 7 error positions
- 4 data bits

we count bits right to left as usual
you'll often see this from left to right
this is just a convention

we start counting at 1
(not at 0 as usual – 0 indicates "no error")

7 D	6 D	5 D	4 P	3 D	2 P	1 P	Parity bit at position
D	-	D	-	D	-	P	2^0
D	D	-	-	D	P	-	2^1
D	D	D	P	-	-	-	2^2



D … Data bit
P … Parity bit

(7, 4) Hamming Code – Example

We want to transmit 1101 ⟶ the transmitted bit sequence is 1100110

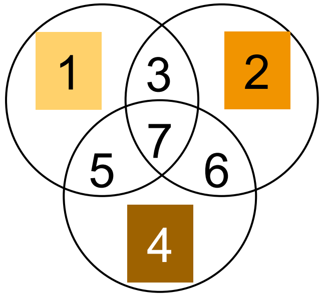
7	6	5	4	3	2	1
1	1	0	0	1	1	0
1	-	0	-	1	-	0
1	1	-	-	1	1	-
1	1	0	0	-	-	-

When one of the bits 1 to 7 is changed ⟶ one or more of the three parity bits are affected

- Error in bit 7 ⟶ affects all three parity bits
- Error in bit 6 ⟶ affects only parity bits 2 and 4
- Error in a parity bit ⟶ affects only this bit

(7, 4) Hamming Code

Decimal	D	D	D	P	D	P	P
0	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1
2	0	0	1	1	0	0	1
3	0	0	1	1	1	1	0
4	0	1	0	1	0	1	0
5	0	1	0	1	1	0	1
6	0	1	1	0	0	1	1
7	0	1	1	0	1	0	0
8	1	0	0	1	0	1	1
9	1	0	0	1	1	0	0
10	1	0	1	0	0	1	0
11	1	0	1	0	1	0	1
12	1	1	0	0	0	0	1
13	1	1	0	0	1	1	0
14	1	1	1	1	0	0	0
15	1	1	1	1	1	1	1



All Hamming codes are perfect

- They are as dense as possible for a 1-error-correcting code and can always be decoded uniquely.
- Every possible word is either actually a code word or
- has a distance of 1 to an actual code word.

(7, 4) Hamming Code – Decoding Example

Technische Hochschule **Rosenheim**

The sent bit sequence was 1100110; bit 6 changes an we receive 1000110

7 1	6 0	5 0	4 0	3 1	2 1	1 0	Parity bit	Check bit
1	-	0	-	1	-	0	correct	0
1	0	-	-	1	1	-	incorrect	1
1	0	0	0	-	-	-	incorrect	1

- The check bits written bottom to top = left to right encode the position of the erroneous bit in binary.
- Here: Binary number $110_2 = 6_{10} \longrightarrow$ error in bit at position 6.
- Just invert this bit $\longrightarrow$ we obtain the nearest correct code word.

Hamming Codes – Exercise

- You receive the following (7, 4) Hamming code word: 1000111
- Was the transmission error-free?
  - If no: correct the code word!
- Decode: Which decimal number was sent?

Hamming Codes are Linear

Using row vectors: $\boldsymbol{c} = \boldsymbol{o}\boldsymbol{G}$

Generator for (7, 4) code: $\boldsymbol{G} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$

maps 4 data bits to 7-bit Hamming code word

7 D	6 D	5 D	4 P	3 D	2 P	1 P
D	-	D	-	D	-	P
D	D	-	-	D	P	-
D	D	D	P	-	-	-

Check matrix for (7, 4) code: $\boldsymbol{H}\boldsymbol{c}^T$

$$\begin{array}{c} \boldsymbol{H} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \\ \quad\quad 7 \quad 6 \quad 5 \quad 4 \quad 3 \quad 2 \quad 1 \end{array}$$

matches the table exactly!

Hamming Codes are Linear – Example

We want to encode 1101

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

The sender receives the (correct) code word 1100110 and checks for errors:

$$\boldsymbol{H}\boldsymbol{c}^T = \begin{pmatrix}1 & 0 & 1 & 0 & 1 & 0 & 1\\ 1 & 1 & 0 & 0 & 1 & 1 & 0\\ 1 & 1 & 1 & 1 & 0 & 0 & 0\end{pmatrix}\begin{pmatrix}1\\1\\0\\0\\1\\1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\0\end{pmatrix} \quad \longrightarrow \text{no error detected}$$

remember: it's all mod 2

Hamming Codes are Linear – Example

The sender receives the (incorrect) code word 1000110 and checks for errors:

$$\boldsymbol{Hc}^T = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \uparrow \qquad \longrightarrow \text{error detected at position 6}$$

remember: it's all mod 2

Hamming Codes are Linear

In fact, we can choose the order of the columns of $\boldsymbol{H}$ any way we like

- this will give us a different, but equivalent Hamming code
- in particular, we can use the standard form
  - this way, the parities are no longer mixed in between the data bits
  - the generator has to be changed accordingly

Example: (7, 4) code:

(negate) &
transpose

$$\boldsymbol{H}_S = \begin{array}{ccccccc} 7 & 6 & 5 & 3 & 1 & 2 & 4 \\ \hline \end{array}$$

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

3x3 Identity matrix

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

fill to correct dimension with identity matrix (here: 4x4)

Hamming Codes are Linear – Example Revisited

We want to encode 1101

$$\boldsymbol{c} = \boldsymbol{o}\boldsymbol{G}_S = \begin{pmatrix}1 & 1 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & 0 & 0 & 1 & 1 & 1\\ 0 & 1 & 0 & 0 & 0 & 1 & 1\\ 0 & 0 & 1 & 0 & 1 & 0 & 1\\ 0 & 0 & 0 & 1 & 1 & 1 & 0\end{pmatrix} = \begin{pmatrix}1 & 1 & 0 & 1 & 0 & 1 & 0\end{pmatrix}$$

original data – much easier to separate from parities by receiver after correction

The sender receives the (correct) code word 1101010 and checks for errors:

$$\boldsymbol{H}_S\boldsymbol{c}^T = \begin{pmatrix}1 & 0 & 1 & 1 & 1 & 0 & 0\\ 1 & 1 & 0 & 1 & 0 & 1 & 0\\ 1 & 1 & 1 & 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1\\1\\0\\1\\0\\1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\0\end{pmatrix}$$

$\longrightarrow$ no error detected

remember: it's all mod 2

Hamming Codes are Linear – Example Revisited

The sender receives the (incorrect) code word 1001010 and checks for errors:

$$\boldsymbol{H}_S\boldsymbol{c}^T = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \uparrow$$

$\longrightarrow$ error detected at position 6

remember: it's all mod 2

(15, 11) Hamming Code

15 bits total

- 4 parities
- 11 data bits

15 D	14 D	13 D	12 D	11 D	10 D	9 D	2^3 8 P	7 D	6 D	5 D	2^2 4 P	3 D	2^1 2 P	2^0 1 P
D	-	D	-	D	-	D	-	D	-	D	-	D	-	P
D	D	-	-	D	D	-	-	D	D	-	-	D	P	-
D	D	D	D	-	-	-	-	D	D	D	P	-	-	-
D	D	D	D	D	D	D	P	-	-	-	-	-	-	-

D … Data bit
P … Parity bit

Checksums

Checksums

- Humans handling sequences of (mostly decimal) digits are prone to make mistakes
  - e.g., when entering order numbers or product codes into computers
- We will look at codes that add checksums to cope with incorrect inputs

- Simple method: **Digit Sum** (*Quersumme*)
  - Add up all digits, reduce modulo 10 to get a single decimal digit (check digit, *Prüfziffer*)
  - Append at the end of the sequence
  - Incorrect inputs can be detected by comparing the resulting check digit with the expected one
  - Drawback: permutations of correct digits result in the same check digit
  - Possible solution: Weigh digits when summing

Checksums – ISBN-10

International Standard Book Number

$$\text{ISBN} = a_1\text{-}\underbrace{a_2a_3a_4}\text{-}\underbrace{a_5a_6a_7a_8a_9}\text{-}p$$

Language group　Publisher number　title number　check digit

Calculation of p:

- weighted sum: $10a_1 + 9a_2 + 8a_3 + 7a_4 + 6a_5 + 5a_6 + 4a_7 + 3a_8 + 2a_9$
- Determine p such that the total sum including p is divisible by 11 without remainder

Result:

- $0 \leq p \leq 10$ – the two-digit remainder 10 is replaced by the single character X
- For a correct ISBN-10 we have: $(10a_1 + 9a_2 + 8a_3 + 7a_4 + 6a_5 + 5a_6 + 4a_7 + 3a_8 + 2a_9 + p) \bmod 11 = 0$
- Both, incorrectly entered digits as well as transposed digits can be detected

ISBN-10 – Exercise

Check the following ISBN: 3-528-25717-2

Is it correct?

Checksums – IBAN

International Bank Account Number

- up to 34 characters, usually shorter
- in Germany: 22 characters

$$\mathrm{DE}\ p_1\ p_2\ b_1b_2b_3b_4b_5b_6b_7b_8\ k_1k_2k_3k_4k_5k_6k_7k_8k_9k_{10}$$

$k_1k_2k_3k_4k_5k_6k_7k_8k_9k_{10}$:	former account number
$b_1b_2b_3b_4b_5b_6b_7b_8$:	former bank code
$p_1\ p_2$:	checksum
DE:	country code

Checksums – IBAN Checksum Calculation

- Initialize $p_1\ p_2$ = 00
- Move country code and $p_1\ p_2$ to the right:

  $b_1b_2b_3b_4b_5b_6b_7b_8\ k_1k_2k_3k_4k_5k_6k_7k_8k_9k_{10}$ **DE 00**

- Replace country code by the positions of the characters in the alphabet + 9 (A=10, B=11, …):

  $b_1b_2b_3b_4b_5b_6b_7b_8\ k_1k_2k_3k_4k_5k_6k_7k_8k_9k_{10}$ **13 14** 00

- Calculate remainder of division by 97
- Choose $p_1\ p_2$ such that the remainder is 1

IBAN Checksum Calculation – Example

Bank code: 711 500 00, account number: 215 632

- Combination, country code/checksum on the right:

  711500000000215632**DE00**

- Replace country code by the positions in alphabet + 9:

  711500000000215632**1314**00

- Remainder of division by 97:
  711500000000215632131400 mod 97 = 49

- Checksum is: 98 – 49 = 49

- IBAN: DE 49 7115 0000 0000 2156 32

IBAN Checksum Validation – Example

IBAN: DE 49 7115 0000 0000 2156 32

- country code/checksum on the right:

  711500000000215632**DE49**

- Replace country code by the positions in alphabet + 9 :

  711500000000215632**1314**49

- Remainder of division by 97 :

  711500000000215632131449 mod 97 = 1 $\longrightarrow$ IBAN correct

IBAN – Questions

1. Is this a weighted sum that we use with IBAN?

2. We use integers with up to 36 decimal digits – with standard data types in typical programming languages, we have up to 64 bits:
   - How many bits would be required for (unsigned) integer arithmetic with 36 decimal places?
   - Nevertheless: How can we perform the calculations using standard types? Or can't we?

3. Why mod 97? And not 98, 99, 100? We would still get two-digit checksums.

Checksums in General

Check the (decimal) sequence $z_n \dots z_i \dots z_0$ (including check digits) using weights g_i

$$\sum_{i=0}^{n} g_i z_i \mod m = 0$$

- Detection of single incorrect digits is guaranteed if all weights g_i and m are relatively prime (*teilerfremd*): $\gcd(g_i, m) = 1$

- Detection of the transposition (*Vertauschung*) of two digits z_i and z_k is guaranteed, if $g_i - g_k$ and m are relatively prime.

$\longrightarrow$ using prime numbers for m makes sense