Error Detection and Correction

Becker/Molitor, Chapter 13

Jan Reineke
Universität des Saarlandes

Overview: Codes for Error Detection and Correction

- Motivation

- Codes

- Error Detection
  - general results
  - Example of a 1-error-detecting code: Parity code

- Error Correction
  - general results
  - Example of a 1-error-correcting code: Hamming code

Transmission and storage errors

Computers store, process and produce information
→ Information storage and transfer must be exact

Problems: noise, crosstalk, attenuation
→ There is **no** exact data transfer or data storage
→ *Goal*: Coding that is robust against disturbances

Binary codes

Let $A=\{a_1,\ldots,a_m\}$ be a finite alphabet of size m.

A mapping $c : A \rightarrow \{0,1\}^*$ is called **code**, if c is injective.

The set $c(A) := \{ c(a) \mid a\in A \}$ is the set of **codewords.**

A code $c : A \rightarrow \{0,1\}^n$ is called **fixed-length code.**

What is the minimum length n of
a fixed-length code for a set A?

Binary codes

What is the minimum length n of
a fixed-length code for a set A?

For a fixed-length code $c : A \rightarrow \{0,1\}^n$ we have: $n \geq \lceil \log_2 m \rceil$.

If $n = \lceil \log_2 m \rceil + r$ with $r > 0$, then the r additional bits can be used to **detect** and **correct errors**.

Motivation: Transmission errors + storage errors

A **transmission error** (storage error) of a word from $\{0, 1\}^*$ occurs if the received bit sequence differs from the sent (stored) bit sequence.

Transmission error = **Flipping of individual bits** $(0 \rightarrow 1, 1 \rightarrow 0)$

Transmission errors increase the **(Hamming) distance** $dist(v,w)$ between the send bit sequence v and the received bit sequence w.

The **(Hamming) distance** of two bit sequences is the number of places in which the two bit sequences differ.

Hamming distance: Example

$dist(00001101,10001100) = 2$

$dist(00001101,00001101) = 0$

A transmission error is called **simple**, if $dist(v,w)=1$.

Error-detecting Codes

Let $c : A \to \{0,1\}^n$ be a fixed-length code of A.

The code c is **k-error detecting**, if the receiver can always determine whether the sent codeword has been disturbed by flipping up to *k* bits.

The minimal distance

$$\mathrm{dist}(c) := \min \{\mathrm{dist}(c(a_i),c(a_j)) \mid a_i,a_j \in A \text{ with } a_i \neq a_j \}$$

between two codewords is called the **code distance**.

Lemma (Error Detection)

A fixed-length code c is *k*-error detecting *iff* $\mathrm{dist}(c) \geq k+1$.

Repetition code: A 1-error-detecting code

Let $c : A \to \{0,1\}^n$ be a fixed-length code of A.
Consider the repetition code $R2 : A \to \{0,1\}^{2n}$ that arises from c by repeating each bit of a codeword twice.

What is R2's code distance?

R2's code distance is 2

$\Rightarrow$ Code R2 is 1-error-detecting!

Is the repetition code efficient?

Can we do better?

Parity code: A 1-error-detecting code

Parity Check:

A bit sequence $w \in \{0,1\}^n$ passes the **parity check**, if the number of bits that are 1 is even.

Parity Code:

Let $c : A \rightarrow \{0,1\}^n$ be a fixed-length code of A.

Consider the code $C : A \rightarrow \{0,1\}^{n+1}$ that arises from c by adding one bit to each codeword $c(a)$ so that the new code $C(a)$ passes the parity check.

$\Rightarrow$ Code C is 1-error-detecting!

Error-correcting Codes

Let $c : A \rightarrow \{0,1\}^n$ be a fixed-length code of A.

Code c is **k-error correcting**, if the receiver can always determine whether the sent codeword has been disturbed by flipping up to *k* bits,
and is able to restore the sent codeword from the received bit sequence.

Repetition code: A 1-error-correcting code

Let $c : A \rightarrow \{0,1\}^n$ be a fixed-length code of A.
Consider the repetition code $R3 : A \rightarrow \{0,1\}^{3n}$ that arises from c by repeating each bit of a codeword three times.

$\Rightarrow$ Code R3 is 1-error-correcting!

Why?

Is the repetition code efficient?

Can we do better?

Error-correcting Codes

Let $c : A \to \{0,1\}^n$ be a fixed-length code of A.

Code c is **k-error correcting**, if the receiver can always determine whether the sent codeword has been disturbed by flipping up to k bits,
and is able to restore the sent codeword from the received bit sequence.

Lemma (Error Correction)
A fixed-length code c is k-error correcting *iff* $\mathrm{dist}(c) \geq 2k+1$.

Proof of Lemma (Error Correction)

Let $M(c(a_i), k) := \{w \in \{0,1\}^n \mid \mathrm{dist}(c(a_i),w) \leq k\}$
be the sphere around $c(a_i)$ with radius k.

Then we have:

c is *k*-error correcting $\Leftrightarrow$

$\forall a_i, a_j\ i \neq j : M(c(a_i),k) \cap M(c(a_j),k) = \varnothing$

Thus, we need to show:

$[\forall a_i, a_j\ i \neq j : M(c(a_i),k) \cap M(c(a_j),k) = \varnothing] \Leftrightarrow \mathrm{dist}(c) \geq 2k+1$

Proof of Lemma (Error Correction)

To show: $[\forall a_i, a_j \;\; i \neq j : M(c(a_i),k) \cap M(c(a_j),k) = \varnothing]$

$\Leftrightarrow \;\; dist(c) \geq 2k+1$

"$\Rightarrow$" (Proof by contraposition)

Assumption: $dist(c) < 2k+1$
i.e., $\exists a_i, a_j$ with $dist(c(a_i),c(a_j)) = d$ such that $d < 2k+1$;

Thus there is a sequence:
$c(a_i) = b_0, b_1, \ldots, b_{k-1}, b_k, b_{k+1}, \ldots b_{2k} = c(a_j)$

with $dist(b_i, b_{i+1}) = 0$ or $dist(b_i, b_{i+1}) = 1$ $(i=0,\ldots,2k-1)$,

and so $b_k \in M(c(a_i),k) \cap M(c(a_j),k)$.

Proof of Lemma (Error Correction)

To show: $[\forall a_i, a_j\ \ i \neq j : M(c(a_i),k) \cap M(c(a_j),k) = \emptyset]$

$\Leftrightarrow\ dist(c) \geq 2k+1$

"$\Leftarrow$" (Proof by contraposition)

Assumption: $\exists a_i, a_j\ \ i \neq j : M(c(a_i),k) \cap M(c(a_j),k) \neq \emptyset$

Thus there is a b in the intersection such that:

$$\begin{aligned} dist(c) &\leq dist(c(a_i), c(a_j)) \\ &\leq dist(c(a_i), b) + dist(b, c(a_j)) \leq k + k \end{aligned}$$

How many additional bits are required for error correction?

Let $c : A \rightarrow \{0,1\}^{m+r}$ be a 1-error-correcting fixed-length code of A with $|A| = 2^m$.

Theorem (Lower Bound): Then: $r \geq 1 + \lfloor \log_2 m \rfloor$.

Proof:

We must have $M(c(a_1), 1) \cap M(c(a_2), 1) = \emptyset$ for all $a_1, a_2 \in A$ with $a_1 \neq a_2$.

We have $|M(c(a),1)| = m+r+1$ for all $a \in A$ (Why?).

$\Rightarrow 2^m(m+r+1) \leq 2^{m+r}$, from which the claim follows (after simple calculation).

Proof Theorem (Lower Bound)

It remains to show: $m+r+1 \leq 2^r \Rightarrow r \geq 1+\lfloor \log_2 m \rfloor$

Let $m = 2^k + l$ with $l, k \in N, l \geq 0$ and k maximal.
(I.e., k, l are chosen such that $k = \lfloor \log_2 m \rfloor$).
Then we have:

$$
\begin{aligned}
& m+r+1 \leq 2^r \\
\Leftrightarrow\ & 2^k + l + r + 1 \leq 2^r \\
\Rightarrow\ & 2^k + 1 \leq 2^r \\
\Rightarrow\ & k < r \\
\Leftrightarrow\ & 1+k \leq r \\
\Leftrightarrow\ & 1+\lfloor \log_2 m \rfloor \leq r
\end{aligned}
$$

1-error-correcting Code: Lower Bound

The lower bound from the theorem for the number of additional bits is not always exact.
From the proof we can conclude:

Corollary:
Let $c : A \to \{0,1\}^{m+r}$ be a 1-error correcting fixed-length code of A with $|A| = 2^m$. Then: $m+r+1 \le 2^r$.

Intuition:
Error-correcting bits must be able to encode the error location (there are m+r possible locations) or that there is no error (1 possibility).

The corollary may provide a slightly sharper lower bound for the number of additional bits.

- Example: $m = 63$.
  The theorem provides $r \ge 6$, with the corollary we get $r \ge 7$.

1-error-correcting Code: Example

Hamming code:

- is a **1-error-correcting code**
- extends non-error-correcting code by r bits;
  such that the number of additional bits r is minimal
  under the condition $m + r + 1 \leq 2^r$,
- and thus corresponds **exactly** to the condition from the last corollary for the **minimum length** of a 1-error-correcting code!

$\Rightarrow$ The Hamming code is thus **space optimal**.

Hamming code: Idea

Extend non-error-correcting code by r additional bits.

Use the bits at positions 2^0, 2^1, … 2^{r-1} as error-correcting bits. The bit at position 2^j checks the bits at those positions whose binary representations are 1 at the *j*-th digit.

Bit position 2^j is chosen so that an even number of the bits at positions whose binary representations are 1 at the *j*-th digit are set.

Intuition:
Every error-correcting bit contributes a parity test that provides one bit of the binary encoding of the error location.

Hamming code on an example

Input: 0111 0101 0000 1111
➔ m = 16, r = 5

What's the Hamming code of this input?

- The code is extended to 21 bits
- The "power-of-two" positions are used as error-correcting bits (numbering starts on the right with position 1)

  0 1110 _101 0000 _111 _1__

  where the bit at position 2^j checks the bits at those positions whose binary representations have a 1 in the *j*-th digit.

Hamming code on an example

	2^4	2^3	2^2	2^1	2^0	bit sequence to encode
3				x	x	1
5			x		x	1
6			x	x		1
7			x	x	x	1
9		x			x	0
10		x		x		0
11		x		x	x	0
12		x	x			0
13		x	x		x	1
14		x	x	x		0
15		x	x	x	x	1
17	x				x	0
18	x			x		1
19	x			x	x	1
20	x		x			1
21	x		x		x	0

The error-correcting bit 2^j checks the bits whose encoding have a 1 in the *j-th* digit.

Hamming code on an example

	2^4	2^3	2^2	2^1	2^0	bit sequence to encode
3				1	1	1
5			1		1	1
6			1	1		1
7			1	1	1	1
9		0			0	0
10		0		0		0
11		0		0	0	0
12		0	0			0
13		1	1		1	1
14		0	0	0		0
15		1	1	1	1	1
17	0				0	0
18	1			1		1
19	1			1	1	1
20	1		1			1
21	0		0		0	0
	1	0	0	0	0	

The error-correcting bit 2^j checks the bits whose encoding have a 1 in the *j*-th digit.

The error-correcting bit is determined as the **sum modulo 2** of the corresponding column.

Hamming code on an example

The Hamming code of

0111 0101 0000 1111

is thus

0 1110 1101 0000 0111 0100

How to find an error?

The Hamming code of

0111 0101 0000 1111

is thus

0 1110 1101 0000 0111 0100

Assume there is an error in position 13!

How do we find the error location?

How to find an error?

	2^4	2^3	2^2	2^1	2^0	bit sequence to encode
3				1	1	1
5			1		1	1
6			1	1		1
7			1	1	1	1
9		0			0	0
10		0		0		0
11		0		0	0	0
12		0	0			0
13		0	0		0	0
14		0	0	0		0
15		1	1	1	1	1
17	0				0	0
18	1			1		1
19	1			1	1	1
20	1		1			1
21	0		0		0	0
	1	0	0	0	0	

Error must be in row 8+4+1=13!

The columns 8, 4 and 1 do not pass the parity check!

Summary

- Basic definitions for codes
- Error detection, Error correction
- Examples: Parity check, Hamming code