

Codierungstheorie – Teil 1: Fehlererkennung und -behebung

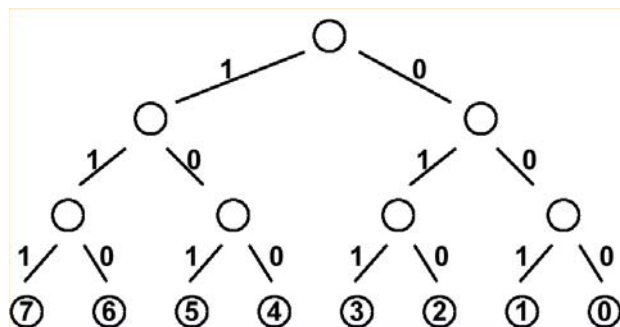
von Manuel Sprock

1 Einleitung

Eine **Codierung** ist eine injektive Abbildung von Wortmengen aus einem Alphabet A in über einem Alphabet B. Jedem Wort aus A wird dabei ein Wort aus B eindeutig zugeordnet. Die Menge aller **Codeworte** nennt man **Code**. Unter der **Decodierung** versteht man die Umkehrabbildung, die einem Codewort wieder das ursprüngliche Wort zuordnet. Man unterscheidet Codes mit fester (Beispiel: ASCII-Code) und variabler Codewortlänge (Beispiel: Morsecode). Codes mit variabler Wortlänge heißen **Präfix-Codes**, falls kein Codewort Anfang eines anderen ist. Nur in diesem Fall ist eine eindeutige Decodierung möglich. Codes mit fester Wortlänge werden **Blockcodes** genannt. Codes unterliegen einer Vielzahl mathematischer Gesetze (Körper- und Vektorraumeigenschaften usw.), auf die hier nicht explizit eingegangen werden soll. Im Folgenden spielen hauptsächlich Binärcodes eine Rolle, Ausgangspunkt der Betrachtung sei der 3-Bit-Binärcode. Er besteht aus acht Codeworten, die man z.B. beim BCD-Code den Dezimalziffern von 0 bis 7 zuordnet.

000
001
010
011
100
101
110
111

3-Bit-Binärcode



Codebaum zum 3-Bit-BCD-Code

Weitere wichtige Begriffe der Codierungstheorie sind der **Hamming-Abstand** und das **Hamming-Gewicht**. Der Hamming-Abstand $d(x,y)$ gibt die Anzahl der Binärstellen an, in denen sich die Codeworte x und y unterscheiden. Unter dem Hamming-Abstand eines Codes versteht man das Minimum der Hamming-Abstände zwischen je zwei Codeworten. Das Hamming-Gewicht $w(x)$ gibt die Anzahl der Einsen in x an. Unter der **Parität** eines Codeworts $p(x)$ versteht man das Ergebnis der XOR-Verknüpfung der Binärstellen, also die Summe der Binärstellen modulo 2.

Beispiele: $d(01101, 00111) = 2,$

$w(0\ 1\ 1\ 0\ 1) = 3,$

$w(1\ 0\ 1\ 1\ 1) = 4$

$p(01101) = 1$

$p(01100) = 0$

2 Fehlererkennung und -behebung

2.1 Anforderungen

Ziel einer Codierung ist die Übertragung oder Speicherung von Informationen. An Codierungen werden verschiedene Anforderungen gestellt, eine davon ist Resistenz gegenüber Fehlern bei der Übertragung oder Speicherung. Unter einem **n-fach-Fehler** versteht man die Verfälschung von n Binärstellen eines Codewortes. Für einen konkreten Anwendungsfall muss man die möglichen Fehler und die Wahrscheinlichkeit, mit der sie auftreten, möglichst genau kennen. Für die Erkennbarkeit von Fehlern ist der Hamming-Abstand von entscheidender Bedeutung. Fehlererkennende Codes bestehen aus Codeworten, die durch bestimmte Fehler zu Worten verfälscht werden, die nicht zum Code gehören („Nicht-Codeworte“). Empfängt der Empfänger ein Nicht-Codewort, kann er sicher sein, dass ein Fehler aufgetreten ist.

2.2 Wiederholungscode

Die einfachste Überlegung zur Erkennung eines Übertragungsfehler ist die Wiederholung jedes Bits in der zu übertragenden Bitfolge: statt des Informationswortes

1 0 0 0 1 1 1 0

überträgt man

11 00 00 00 11 11 11 00.

Dies entspricht einer Erhöhung des Hamming-Abstandes auf $d = 2$. Ein Einfachfehler kann nun erkannt, jedoch nicht behoben werden, da nicht feststellbar ist, an welcher Position innerhalb eines Zweierblocks er aufgetreten ist. Verdreifacht man jedes Bit, so wird ein Einzelfehler korrigierbar. Das Verfahren ist jedoch offensichtlich nicht sehr effizient, da der Informationsgehalt der Codeworte erhalten bleibt, sich die Länge jedoch verdoppelt bzw. verdreifacht.

2.3 Prüfsummen

Effektivere Fehlererkennung kann durch Verwendung von Prüfsummen erreicht werden. Hierbei wird das zu übertragende Informationswort durch eine zuvor festgelegte Zahl dividiert und der Rest der Division wird an das Informationswort angehängt. Bei der Decodierung wird ebenfalls die Division durchgeführt und der errechnete Rest wird mit dem übertragenen verglichen. In der Praxis werden Erweiterungen dieses Verfahrens verwendet, die nach diesem Prinzip funktionieren. Beim EAN-Code (European Article Number) ist die letzte von 13 Ziffern eine Prüfziffer, die sich wie folgt berechnet: $a_{13} = (a_1 + 3a_2 + a_3 + 3a_4 + \dots + a_{11} + 3a_{12}) \bmod 10$. Weitere Beispiele sind ISBN oder Kontonummern.

2.4 Paritätsprüfung

Ein weiteres Verfahren ist die sogenannte Paritätsprüfung. Man fügt an die Bitfolge ein Bit an, sodass für alle Codeworte x gilt: Parität $p(x) = 0$. Dies entspricht ebenfalls der Erhöhung des Hamming-Abstandes auf $d = 2$.

```

000 0
001 1
010 1
011 0
100 1
101 0
110 0
111 1

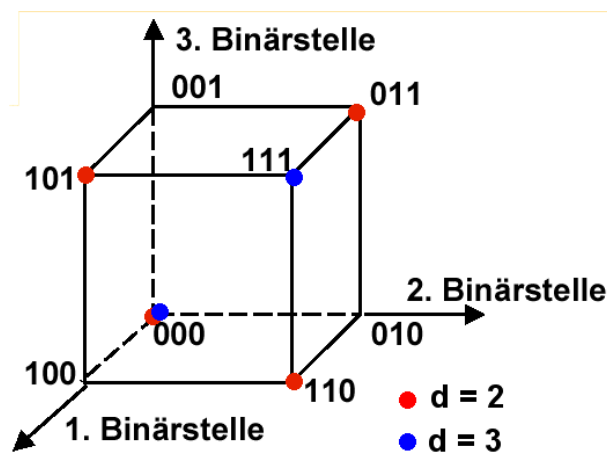
```

3-Bit-Binärcode mit Paritätsbit

Wenn man nun ein beliebiges Codewort an einer beliebigen Binärstelle verfälscht, wird es zu einem Nicht-Codewort. Doppelfehler werden allerdings nicht erkannt, denn eine Verfälschung an 2 Binärstellen macht aus jedem Codewort ein anderes Codewort.

2.5 Fehlerbehebung

Bei einem Code mit einem Hamming-Abstand von $d = 3$ sind Einzelfehler korrigierbar oder Doppelfehler erkennbar, jedoch nicht beides zugleich. Alle Codeworte haben den Mindestabstand $d = 3$, das bedeutet, dass sie sich in mindestens 3 Binärstellen unterscheiden. Ein durch einen Einzelfehler verfälschtes Wort hat zum ursprünglichen Codewort den Abstand $d = 1$, deshalb ist eindeutig erkennbar, aus welchem Codewort es hervorgegangen ist. Beispiel: der Code besteht aus den 2 Worten 000 und 111, die den Abstand $d = 3$ haben. Wird das Codewort 000 an genau einer Stelle verfälscht, so ergeben sich die fehlerbehafteten Worte 001, 010 oder 100. Diese haben zum anderen Codewort 111 (in diesem Falle gibt es ja nur ein weiteres) den Abstand $d = 2$. Dieser Sachverhalt lässt sich graphisch darstellen:



Hamming-Abstände des 3-Bit-Binärcode

Rot eingezeichnet: ein aus 4 Worten bestehender Code mit dem Abstand $d = 2$. In Blau: der o.g. Code aus 2 Worten mit dem Abstand $d = 3$.

Ziel ist es nun, einen Code zu finden, der bei minimaler Länge den maximalen Hamming-Abstand bietet. Man bildet paarweise Hamming-Abstände und trägt sie in eine Matrix ein. Hier wieder am Beispiel des 3-Bit-Binär-codes:

	000	001	010	011	100	101	110	111
000	0	1	1	2	1	2	2	3
001	1	0	2	1	2	1	3	2
010	1	2	0	1	2	3	1	2
011	2	1	1	0	3	2	2	1
100	1	2	2	3	0	1	1	2
101	2	1	3	2	1	0	2	1
110	2	3	1	2	1	2	0	1
111	3	2	2	1	2	1	1	0

Hamming-Abstände des 3-Bit-Binär-codes

Vorgehen zur Bildung eines Codes mit dem Hamming-Abstand $d = 2$:

1. Finden der Paare mit $d > 1$ in der 1. Zeile: (000, 011), (000, 101), (000, 110), (000, 111)
2. Überprüfung der Abstände der gefundenen Worte untereinander: in der Zeile 011 die Abstände zu 101, 110 und 111. Da die Matrix symmetrisch ist, kann man sich bei der Untersuchung auf den Teil oberhalb der Diagonalen beschränken. Ist der paarweise Abstand zweier Worte kleiner als der gewünschte Mindestabstand, kann das Wort nicht verwendet werden. Dies ist der Falls für das Paar (011, 111), 111 scheidet damit als Kandidat aus.
3. Fortsetzung des Verfahrens für die verbliebenden Kandidaten: in diesem Fall in der Zeile 101 den Abstand zu 110 überprüfen. Am Ende verbleiben: 000, 011, 101 und 110.
4. Nun ist eine Teilmenge der Codeworte mit dem Abstand $d > 1$ gefunden. Es gibt jedoch mehrere solcher Teilmengen und wir suchen diejenige mit der größten Anzahl an Codeworten. Deshalb: Fortsetzung des Verfahrens mit Schritt 1, jedoch in der 2. Zeile beginnend.

2.6 Hamming-Code

Das o.g. Verfahren ist theoretisch auch auf längere Codes und größere Abstände anwendbar, doch wächst der Aufwand exponentiell. Herr Hamming ist der Erfinder eines Bauplans für einen Code mit dem Mindestabstand $d = 3$, der die Korrektur eines Einzelfehlers erlaubt. Einem Informationswort werden so viele Prüfstellen hinzugefügt, dass die Position des Fehlers über die Prüfstellen gekennzeichnet werden kann.

Bildung des Codes:

- Die Bits der Codeworte seien von links nach rechts mit 1, 2, ..., n durchnummeriert.
- Die Bits 1, 2, 4, 8, ... (Zweierpotenzen) dienen als Prüfbits.
- Die restlichen Bits enthalten die Nutz-Bits.

- Das Prüfbit Nummer 2^n ist das Paritätsbit für die Menge der Datenbits, die eine Nummer haben, in deren Binärdarstellung der Summand 2^n vorkommt.

Prüfung eines übertragenen Wortes:

- Berechnung der Prüfbits für das empfangene Wort
- Die Summe der Nummern der Prüfbits, deren übertragener Wert nicht mit dem berechneten übereinstimmt, gibt die Nummer des fehlerhaft übertragenen Bits an.

Beispiel: 4 Nutzbits (a), 3 Prüfbits (p)

Anordnung der Bits: **p₁** **p₂** **a₃** **p₄** **a₅** **a₆** **a₇**

$$\begin{aligned}
 p_1 &= a_3 \oplus a_5 \oplus a_7 & \oplus & \text{bezeichnet die} \\
 p_2 &= a_3 \oplus a_6 \oplus a_7 & & \text{Operation zur Bildung} \\
 p_4 &= a_5 \oplus a_6 \oplus a_7 & & \text{des Paritätsbits (XOR)}
 \end{aligned}$$

In folgender Tabelle werden die gebildeten Codeworte dargestellt. Die 1. Spalte enthält den Binärwert der Informationsstellen, die 2. Spalte den Dezimalwert und die 3. Spalte das Codewort:

0000	0	0000000
0001	1	1101001
0010	2	0101010
0011	3	1000011
0100	4	1001100
0101	5	0100101
0110	6	1100110
0111	7	0001111
1000	8	1110000
1001	9	0011001
1010	10	1011010
1011	11	0110011
1100	12	0111100
1101	13	1010101
1110	14	0010110
1111	15	1111111

Wörter eines 1-Bit-Fehlerkorrigierenden (7,4)-Hammingcodes

Beispiel zur Fehlerkorrektur: sei das Codewort 1111111 zu 1111011 verfälscht worden. Die Berechnung der Prüfbits und der Vergleich mit den übertragenen Prüfbits ergibt:

Berechnet	Übertragen
$p_1 = 0$	$p_1 = 1$
$p_2 = 1$	$p_2 = 1$
$p_4 = 0$	$p_4 = 1$

Die Summe der Nummern der nicht übereinstimmenden Prüfbits ergibt 5. Bit Nummer 5 wurde also falsch übertragen! Die Berechnung funktioniert natürlich auch im Falle eines fehlerhaft übertragenen Prüfbits.

Quellen:

Codierung (Wilfried Dankmeier), 2. Auflage, Vieweg-Verlag

Codierungstheorie (Chr. Vogt, FH München): <http://www.cs.fhm.edu/~vogt/gdi/CODIERG.pdf>

Codierungstheorie (FH Flensburg): <http://www.iti.fh-flensburg.de/lang/algorithmen/code/codier.htm>