Thema:
„Hamming-Codes“

Titelblatt
anonymisiert

Hamming-Codes

1 Einführung in die Kodierungstheorie (aus Quelle: [Q1])

Ein Code, dahinter steckt nichts anderes als eine Information, die geschickt verschlüsselt wurde. Man benutzt Codes an sehr vielen Stellen im Alltag, bei PCs, beim Telefon, beim Einkaufen, selbst das Lesen und Schreiben sind Kodierungs- und Dekodierungsvorgänge.

Doch wieso brauchen wir überhaupt Codes? Was steckt hinter diesem "Verpacken" von Informationen?
Nun, ein Code kann je nach Anwendungsgebiet einen anderen Zweck erfüllen, jedoch bleibt der Grundgedanke immer derselbe: Man will eine **Information von A nach B** senden.
Dies soll nach Möglichkeit so geschehen, dass diese Information für unbefugte Personen nicht lesbar oder verstehbar ist.

Beispielsweise wird jedem Haushalt in Deutschland eine Telefonnummer zugeteilt, aus der man nicht sofort erkennen kann, welche Person sich hinter dieser Nummer verbirgt, sodass die Privatsphäre jedes Haushaltes geschützt wird.
Daraus ergibt sich ein weiterer großer Vorteil, nämlich verkürzt man die Informationen. Der Alltag wird dadurch stark vereinfacht. Müssten wir bei jedem Telefoniervorgang den kompletten Namen, Adresse, Geburtstag und weitere Informationen eingeben, um zur genau richtigen Person zu gelangen, so wäre das doch viel langwieriger, als einfach eine kurze Nummer in das Telefon einzugeben.

Ein weiterer, großer Vorteil von Codes ist es, dass man die Möglichkeit hat, Übertragungsfehler zu erkennen und sogar zu korrigieren. Und einer der bekanntesten fehlerkorrigierenden Codes ist der sogenannte "Hamming-Code".

1.1 Grundlegendes über Codewörter

Um Hamming-Codes nun genau verstehen zu können, muss man zuerst die grundlegenden Ideen der Kodierungstheorie verstehen. Ein Code ist eine Menge von Codewörtern.
C bezeichnet den gesamten Code und beinhaltet alle Codewörter, C ist ein Codewort.
c ist die y-te Stelle im Codewort (c wird auch manchmal einfach mit x bezeichnet).

Diese Codewörter sind alle definiert durch drei Eigenschaften, die mit n, i und d abgekürzt werden.

n bezeichnet die **Länge des Codewortes**. Diese kann innerhalb eines Codes variieren, ist jedoch in den meisten Fällen für alle Codewörter dieselbe. Telefonnummern beispielsweise können verschiedene Längen haben, während eine ISBN immer genau 10 Zeichen lang ist.

i steht für die **Größe des Alphabets** des Codes. Das Alphabet ist die Menge aller Zeichen, die in den Codewörtern vorkommen können. Eine Telefonnummer besteht aus Zeichen von 0 bis 9, also handelt es sich um einen Code mit i=10.
Die meisten Codes der Kodierungstheorie beinhalten allerdings nur Nullen und Einsen, also insgesamt 2 Zeichen, und werden auch binäre Codes genannt.

d ist der sogennante **Hamming-Abstand**. Er regelt, in wievielen Stellen sich die einzelnen Codewörter mindestens unterscheiden. In einem Code C mit Hamming-Abstand d=3 könnten die Codewörter C=10001 und C=10111 nicht gleichzeitig vorkommen, da sie sich in nur 2 Stellen unterscheiden. Ein solches neben C wäre beispielsweise C=01101.

1.2 Matrizen

Jeder Code ist definiert durch eine sogenannte **Prüfmatrix**.
Eine Matrix ist eine mathematische Form einer Tabelle, mit der man rechnen kann.

Eine **Prüfmatrix** (oder Kontrollmatrix) ist ein **Gleichungssystem**.
Jede Zeile steht für eine Gleichung, die für modulo Z/iZ in der Summe gleich Null sein muss.
Jede Gleichung muss für jedes Codewort richtig sein, um ein Element von C zu sein.
Die einzelnen Bestandteile jeder Zeile sind die Faktoren, mit denen die jeweilige Stelle im Codewort multipliziert werden muss.

Beispiel:

bei i=3

Ist ein Gleichungssystem:
I = 0c+ 2c+1c=0 Z/3Z
I = 1c+1c+1c =0 Z/3Z

Eine weitere Form einer Matrix in der Kodierungstheorie ist die Erzeugermatrix.
Diese ist jedoch für den Hamming-Code unwichtig und wird deshalb nicht erklärt.

1.3 Die "maßgebliche Stelle"

Die maßgebliche Stelle ist kein mathematischer Begriff, er wird von einigen Quellen zur Vereinfachung eingeführt, um bessere Erklärungen zu einigen Theorien liefern zu können (zum Beispiel in Quelle [I1]).

Die **"maßgebliche Stelle"** m ist die erste Stelle eines Wortes, die nicht Null ist.
Beispiel: X = 00210 # m=3, da die dritte Stelle die erste Stelle ist, die nicht Null ist.

Der Wert, der an dieser Stelle steht, wird als **"maßgeblicher Wert"** bezeichnet.
Der maßgebliche Wert von X ist somit 2.

2 Grundlegende Ideen **(aus Quelle: [H1- H6])**

Der Hamming-Code ist eine der bekanntesten Erfindungen des amerikanischen Mathematikers **Richard Wesley Hamming**(1925-1998). Durch den Hamming-Code, das Hamming-Fenster, den Hamming-Abstand und die Hamming-Ähnlichkeit gilt er als einer der Begründer der Codierungstheorie und hatte großen Einfluß auf die heutige Informatik und die Telekommunikationstechnik.

Hamming-Codes haben einen **Hamming-Abstand von 3**.
Ein Hamming-Code ist ein **perfekter Code**, das heißt, dass jeder Fehler an einer Stelle korrigiert werden kann.

Treten Fehler an 2 Stellen auf, können diese erkannt und nur unter Umständen korrigiert werden. Treten Fehler an mehr als 2 Stellen auf, so können diese nicht mehr korrigiert werden. Dies ist wahrscheinlich die größte Schwäche des Hamming-Codes.

3 Die Hamming-Matrix **(aus Quelle: [I1])**

Der Hamming-Code ist wohl einer der elegantesten Wege, eine Botschaft zu verschlüsseln. Der Grund für diese Eleganz ist der geschickte Aufbau der Prüfmatrix, der sogenannten **Hamming-Matrix**. Sie wurde 1949 von **Marcel Golay** und 1950 von **Richard Hamming** entdeckt.
Leider ist der Aufbau bei nicht-binären Codes etwas problematisch. Dies ist auch der Grund dafür, wieso der Hamming-Code nur in der binären Ausführung so handsam ist.
In diesem Kapitel soll der Aufbau der Hamming-Matrix geschildert werden.

3.1 Die Hamming-Matrix für binäre Codes

Die Hamming-Matrix für binäre Codes, die zwar nicht weit von der Hamming-Matrix für nicht-binäre Codes abweicht, allerdings entscheidende Vorteile hat, entsteht durch das **Darstellen aller natürlichen Zahlen** von 1 bis n in ihrer Form **im Dualsystem**.

Wir schreiben also die natürlichen Zahlen von 1 bis n ins Dualsystem um.

1 : 001 6 : 110
2: 010 7 : 111
3: 011 8 : 1000
4: 100 9 : 1001
5: 101 10: 1010
...

Wir wollen als erstes eine Matrix mit 3 Zeilen erstellen. Wir verwenden dabei alle Zahlen, die im Dualsystem höchstens 3 Stellen beanspruchen. Ab der Zahl 8 benötigen alle

darauffolgenden Zahlen mindestens 4 Stellen, deswegen können wir für 3 Spalten genau 7 Zahlen abbilden:

1 2 3 4 5 6 7

Hamming-Codes werden durch die Bezeichnung **H(h)** abgekürzt (h = Anzahl der Zeilen). Dies ist die Hamming-Matrix für den H(3)-Code, also ein binärer Code mit 3 Zeilen und 7 Spalten, wobei jede Spalte eine Zahl im Dualsystem darstellt.
Die Codewörter des H(3)-Codes sind demnach 7 Zeichen lang (n=7).

Natürlich gibt es auch Hamming-Matrizen mit mehr als 3 Zeilen.
Hamming-Codes gibt es für jede beliebige Zeilenanzahl h > 0.

Eine solche Matrix, für h=4, sieht folgendermaßen aus:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$ Der H(4) hat eine Länge von n=15.

Hamming-Codes haben eine **bestimmte Länge**, die in der Regel eingehalten wird.
Theoretisch könne man für H(3) auch nur die Zahlen von 1 bis 5 abbilden:

Warum man dies nicht macht, hat einen praktischen Grund. Man will mit einer Matrix möglichst lange Codewörter haben, da man mit längeren Codewörtern auch längere Daten kodieren kann, und somit sind längere Hamming-Matrizen "universeller" einsetzbar.

3.2 Die Hamming-Matrix für nicht-binäre Codes

Der Aufbau einer nicht-binären Hamming-Matrix ist dem einer binären sehr ähnlich.
Wieder werden die natürlichen Zahlen in ein anderes Zahlensystem umgeschrieben.
Für einen i-ären Code werden die natürlichen Zahlen in das Zahlensystem für i umgeschrieben.

Somit benutzen wir für tertiäre Codes nicht mehr das Dualsystem, sondern das 3-System.
Die Matrix für einen tertiären Code mit 3 Zeilen als Beispiel für alle nicht-binären Codes sieht wie folgt aus:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 & 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 & 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \end{pmatrix}$$

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26

Hier lassen sich alle Zahlen von 1 bis 26 darstellen.
Wir müssen nun alle Spalten, deren maßgeblicher Wert größer als 1 ist, löschen.
Dies gilt für alle Hamming-Codes mit i > 2.

Für den Hamming-Code H(3) sieht die fertiggestellte Matrix nun so aus:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \end{pmatrix}$$

Der Grund, wieso hier Spalten gelöscht werden, hängt mit den Prüfziffern zusammen.
Der genaue Zusammenhang wird in Kapitel 4 genauer erklärt.

4 Die Hamming-Prüfstellen **(verschiedene Quellen)**

Das Besondere am Hamming-Code sind die Prüfstellen. Sie resultieren aus dem speziellen Aufbau der Prüfmatrix. In diesem Kapitel wird gezeigt, wie Hamming-Codewörter aufgebaut sind, wie die Prüfstellen definiert sind und wo Probleme auftreten.

4.1 ISBN **(aus Quelle: [Q1])**

Um den Aufbau der Hamming-Codewörter leichter verstehen zu können, kann man einen Vergleich zur **ISBN-Nummer** ziehen. Die ISBN ist eine Identifikationsnummer für Bücher. Man findet sie auf Rücken eines jeden Buches.
Die ISBN besteht aus **10 Zeichen**. In den ersten 9 sind Informationen enthalten.
Die erste Information ist die Kennzahl für die Sprache des Buches, sie wird auch Länderzahl genannt. Sie kann mehrere Ziffern belegen.
Die nächste Information in der ISBN ist die Nummer des Verlags. So wie die Länderzahl kann auch die Verlagsnummer mehrere Stellen haben.
Dann folgt die Zahl, die dem Buch im Verlag gegeben wurde. Jeder Verlag hat solch eine Durchnummerierung, über die sie frei bestimmen können.

Abb.1: ISBN

Die letzte Ziffer ist die Prüfziffer. Durch ein bestimmtes Schema kann man mit den 9 Informationstellen die Prüfstelle errechnen.
Diese Formel kann man nun auch zum Fehlererkennen benutzen.
Dieses Verfahren wird zum Beispiel in Bücherläden verwendet. Alle Buchtitel und Informationen sind in einem Computer abgespeichert. Ein Vorteil ist, dass man bestimmte Bücher schneller finden kann. Oft verfügen Bücherläden auch über Kundencomputer, die der Kunde selbst bedienen kann.
Wenn sie als Kunde die ISBN eines Buches wissen, das sie suchen, und diese ISBN in den Kundencomputer eingeben, so zeigt ihnen der Computer neben vielen weiteren Informationen auch, wo dieses Buch im Buchladen zu finden ist.
Geben sie allerdings eine falsche Nummer ein, so kann der Computer durch Benutzen des Prüfverfahrens erkennen, das sie eine falsche Nummer eingegeben haben.
Der Computer wird dem Kunden mitteilen, dass in der eingegebenen ISBN ein Fehler vorliegt. So kann der Kunde die eingegebene ISBN noch einmal überprüfen.

3	4	9	9	1	3	4	6	2	4

Tab.1:ISBN-Code

c bis c, hier blau eingefärbt, sind Informationsstellen. c10 ist die Prüfstelle, hier grün eingefärbt.

4.2 Prüfstellen von binären Hamming-Codes (aus Quelle: [H1-H6])

Die ISBN hat 9 Informationsstellen und eine Prüfstelle.
Beim Hamming-Code gibt es nun **nichtmehr nur eine Prüfstelle, sondern mehrere**, wobei man bei Hamming-Codes nicht mehr von Stellen spricht, sondern von **Bits**, nämlich Informations- oder Datenbits und Prüfbits.

Dabei werden Datenbits werden mit **k** abgekürzt, während Prüfbits mit **r** gekennzeichnet werden.
Stellen allgemein bezeichnet man wie zuvor mit n.

Ein sieben-stelliger, binärer Hamming-Code ist nun allgemein so aufgebaut:

n_1	n_2	n_3	n_4	n_5	n_6	n_7

Tab.2:Schema Hamming-Code

Sehen wir uns noch einmal die Hamming-Matrix für n=7 an, um herauszufinden, welche der 7 Stellen sich nun am besten für die Prüfstelle des Hamming-Codes eignet:

Bei Untersuchung der Spalten erkennt man, dass die erste, die zweite und die vierte Spalte jeweils nur **einfach besetzt** sind.
Das bedeutet, c, c und c kommen im Gleichungssystem nur einmal vor, und zwar in unterschiedlichen Zeilen.

Im Gleichungssystem: **c**+c+c+c=0

c+c+c+c=0

c+c+c+c=0

Somit sind c, cund c voneinander unabhängig und kommen nur einmal im Gleichungssystem vor.
Anders ausgedrückt bedeutet das Gleichungssystem auch:

c= -(c+c+c)
c= -(c+c+c)
c= -(c+c+c)

Man hat also 3 Stellen gefunden, die wie die 10.Stelle der ISBN **durch alle anderen unabhängig berechnet werden können**. Diese Stellen kann man als Prüfstellen verwenden.

Prüfstellen bei binären Hamming-Codes sind diejenigen Stellen, welche **Potenzen von 2** sind.

Das heißt :

Die 2^0. Stelle ist Prüfbit. --- Die erste Stelle ist Prüfbit.	1
Die 2^1. Stelle ist Prüfbit. --- Die zweite Stelle ist Prüfbit.	2
Die 2^2. Stelle ist Prüfbit. --- Die vierte Stelle ist Prüfbit.	4
Die 2^3. Stelle ist Prüfbit. --- Die achte Stelle ist Prüfbit.	8
Die 2^4. Stelle ist Prüfbit. --- Die sechtzehnte Stelle ist Prüfbit.	16
Die 2^5. Stelle ist Prüfbit. --- Die zweiunddreisigste Stelle ist Prüfbit.	32
Die 2^6. Stelle ist Prüfbit. --- Die vierundsechzigste Stelle ist Prüfbit.	64

usw.

Für den Beispielscode, der aus 7 Stellen besteht, heißt das:

Die 1., 2. und 4.Stelle sind Prüfbits:

r_1	r_2	k_3	r_4	k_5	k_6	k_7

Tab.3: Beispiel 7-stelliger Code

Ein 7stelliger Code mit 3 Prüfbits und 4 Datenbits.

Insgesamte Anzahl der Stellen: n = 7

Anzahl der Datenbits: k = 4

Anzahl der Prüfbits: r = 3

Ein Hamming-Codewort besteht nur aus Datenbits und Prüfbits.

Logischerweise kann man daraus folgern, dass **n = k + r** gelten muss.

4.3 Prüfstellen von nicht-binären Codes (aus Quelle: [I1] und [H7])

Wir können nun erklären, wieso man beim Aufbau der nicht-binären Hamming-Matrix Spalten löschen muss.

Für eine Prüfziffer brauchen wir immer eine einfach besetzte Spalte.

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 & 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 & 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \end{pmatrix}$$

1 **2** **3** 4 5 **6** 7 8 **9** 10 11 12 13 14 15 16 17 **18** 19 20 21 22 23 24 25 26

Man sieht, dass es auch hier einfach besetzte Spalten gibt.

Die 1., 3. und 9. Spalte sind einfach besetzt mit einer 1.

Die 2., 6. und 18. Spalte sind mit einer 2 einfach besetzt.

Man hätte hier also insgesamt 6 Prüfziffern.

Je größer i wird, desto mehr Prüfziffern hat man für eine gleichbleibende Zeilenanzahl.

Da man auch Hamming-Codes mit großen i einfach gestalten will, und auch die Zahl der Prüfstellen überschaubar machen will, **streicht man alle Spalten mit einem maßgeblichen Wert größer als 1 weg.**

Dies vereinfacht den Code.

In jedem Fall einer 3-zeiligen Matrix gibt es nämlich dann 3 Prüfziffern, und auch i-äre Codes mit großem Alphabet werden noch überschaubar.

Indem wir genau die Spalten mit maßgeblichem Wert > 1 wegstreichen, wird auch das Gleichungssystem einfacher, da dadurch sehr viele Zahlen, die größer als 1 sind, gelöscht werden, und mit vielen Einsern lässt es sich bequemer rechnen.

Allerdings entsteht auch ein **Problem:**

Beim binären Hamming-Code ist jede 2-te Stelle Prüfbit (für a $\in$ N).

Dies ist eine sehr regelmäßige Anordnung, die wir mit dem Löchen von Spalten bei nicht-binären Codes nicht beibehalten können.

Ohne das Löschen wären unsere Prüfstellen bei 1, 3, 9 und 2, 6 und 18, also bei 3 und $2 \cdot 3$.

Mit dem Löschen sind die einfach besetzten Zeilen verrutscht, sie liegen nun bei 1, 2 und 5, die nächste wäre bei 14. Die schöne Periodizität ist verloren gegangen:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \end{pmatrix}$$

1 **2** 3 4 **5** 6 7 8 9 10 11 12 13

Neben mehreren Vorteilen hat das Löschen von einigen Spalten auch einen Nachteil.

Der Code wird vereinfacht, allerdings tut man sich nun schwerer, die Prüfbits zu erkennen.

Der Hamming-Code ist also nur in der binären Ausführung absolut fehlerfrei und einfach.

4.4 Prüfbits und Zeilenanzahl (aus Quelle: [H1-H6])

Eine weitere Eigenschaft des Hamming-Codes, die man nicht sofort erkennen kann, ist der **Zusammenhang zwischen den Prüfbits und der Anzahl der Zeilen der Hamming-Matrix.**

Vergleichen wir die Matrix des binären Hamming-Codes mit 3 Zeilen und der des Codes mit 4 Zeilen. Die binäre Ansicht dienen hierbei nur als Beispiel. Der Zusammenhang gilt auch für jeden nicht-binären Hamming-Code.

für h=3

und

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$ für h=4

Was man nun schön erkennen kann, ist Folgendes:
Immer, wenn man von einer Matrix mit h Zeilen zur nächsten Matrix mit (h+1)-Zeilen geht, wird der Matrix neben anderen auch eine neue, einfach besetzte Spalte hinzugefügt.

Wie wir in 4.2 gesehen haben, sind c, cund c bei binären Hamming-Codes nur deswegen Prüfstellen, weil die Spalten, in denen sie vorkommen (die 1., 2. und 4. Spalte), nur einfach besetzt sind.

Erhöhen wir nun die Zeilenanzahl um 1, so gibt es **genau eine weitere Möglichkeit, eine Spalte nur einfach zu besetzen.**
Dies wäre in diesem Fall die 8. Spalte (8 entspricht 1000) und c wäre bei einem binären Code mit n>7 eine Prüfstelle.

Somit kann man sagen, erhöht man die Anzahl der Zeilen um 1, so erhöht man gleichzeitig auch die Anzahl der Prüfziffern.

Auch kann man sagen, da eine Matrix mit h=3 genau 3 Prüfstellen hat und eine Matrix mit h=4 genau 4 Prüfstellen hat, **dass die Anzahl der Zeilen auch gleichzeitig die Anzahl der Prüfstellen ist.**

Also gilt: **Zeilenanzahl = Prüfstellen ; h = r**

4.5 Berechnung der Codewortlänge (aus Quelle: [I1])

Je länger das Codewort wird, umso größer wird auch die Anzahl der Prüfstellen.
Die Anzahl der Prüfstellen steigt allerdings viel weniger als die Länge:

Prüfstellen sind die 1., 2., 4., 8., 16., 32., 64., ... Stellen für binäre Codes.
Das bedeutet:
Für einen Code mit 7 Stellen brauche ich 3 Prüfstellen, nämlich die 1., 2. und 4.
Für einen Code mit n=63 brauche ich hingegen nur 6 Prüfstellen.
Und für einen Code mit n=255 brauche ich nur noch 8 Prüfstellen.

Das bedeutet, j**e länger der Code wird, umso kleiner wird das Verhältnis .**

Wir wollen berechnen, wie lang der Code mit zunehmendem r wird.
Um n zu berechnen, berechnen wir die Anzahl der Spalten einer Hamming-Matrix.
Auch hier ist der H(3)-Code wieder nur das Beispiel. Die Berechnung gilt auch für alle nicht-binären Hamming-Codes.

Eine Zeile hat n Symbole, eine Spalte hat r Symbole.
Die Größe der Matrix ist somit r • n.
Man kann die Anzahl der Zeilen n in Abhängigkeit von r berechnen.

Man nehme eine Matrix, von der man weiß, sie habe r Zeilen.
In jeder Spalte kann nun die maßgebliche Stelle irgendeine zwischen 1 und r sein.
Ist die maßgebliche Stelle einer Spalte m, so erscheinen vor dieser Stelle m-1 Nullen und nach dieser Stelle r-m weitere Stellen.

Beispiel:
Eine Spalte einer Hamming-Matrix ist: H = (bei i=5)
Die maßgebliche Stelle m von H ist 3.
Vor dieser Stelle sind m-1 Zeichen, hier 2.
Und nach dieser Stelle kommen noch r-m weitere Zeichen, also 5-3 = 2

Für diese weiteren Stellen gibt es i Möglichkeiten:

Im Beispiel:
Nach der maßgeblichen Stelle kommen noch 2 weitere Stellen, die beide mit den Zahlen 0 bis 4 belegt werden können.
Also gibt es für die erste der weiteren Stellen 5 Möglichkeiten und für die zweite ebenfalls 5 Möglichkeiten. Also gibt es $5 \cdot 5$ Möglichkeiten, also 5^2 Möglichkeiten, die weiteren Stellen zu besetzen.

Somit gibt es auch i Spalten, in denen die maßgebliche Stelle m ist, da jede der i Möglichkeiten in der Matrix vorkommen.
m kann die Werte von 1 bis r annehmen.
m kann die Werte von 1 bis r annehmen.
Somit lassen sich alle Spalten aufsummieren, indem man für m die Werte von 1 bis r einsetzt:

$n = i + i + \ldots + i$

Es handelt sich hierbei um eine Summe, die durch Polynomdivision entstanden ist.
Multipliziert man diesen Summenterm mit i-1, so erhält man den Term i-1.
Deswegen gilt:

$n = i + i + \ldots + i =$

Somit hat man nun eine Formel, mit der man bei gegebener Zeilenanzahl (oder Prüfzifferanzahl) die Länge der Codewörter des Hamming-Codes berechnen kann.

Ein Rechenbeispiel: Beim H(3)-Code sind die Codewörter 7 Zeichen lang.
Beweis:

n = = = 7

5 Kodierung und Dekodierung (aus Quelle: [Q1] und [H1-H6])

Da die Theorie nun verstanden ist, kommt jetzt der praktische Teil.
Der praktische Teil ist sehr leicht durchführbar und einfach zu verstehen. Der Hamming-Code ist auch für seine leichte, praktische Seite bekannt, allerdings ist diese praktische Seite nur durch eine **komplexe Hintergrundtheorie** möglich, die ein Anwender nicht unbedingt verstehen muss.

Die **Aufgabe der Kodierungstheorie** ist es**, Informationen zu verschlüsseln** und dann zu **übertragen**. Man spricht nun hier von einer Person **ALICE**, die den Code übertragen will, einem Empfänger **BOB**, dem die Informationen übertragen werden, und einer dritten Person **EVE**. EVE ist eine unbefugte Person, die die Informationen stehlen will.
ALICE und BOB müssen ihre Informationen also gut verschlüsseln, damit EVE, falls die Daten abgefangen werden sollten, die Informationen nicht verstehen kann.

ALICE und BOB benutzen den Hamming-Code.
Sollte es gelingen, die Daten abzufangen, kann EVE die Daten trotzdem nicht verstehen.
Ein Hamming-Code ist für dritte Personen nur eine Zahlenreihe. Unbefugte wissen nicht, welche Stellen der Codewörter Informationen enthalten und welche nur zur Überprüfung dienen.

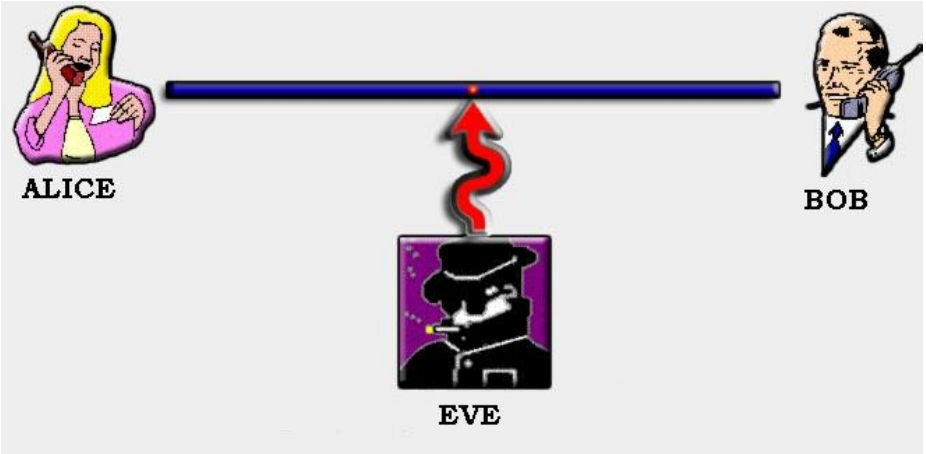

Abb.2: ALICE/BOB/EVE

5.1 Kodierung

ALICE will im ersten Fall eine Zahl Z = 10 übertragen.

ALICE schreibt diese Zahl ins Dualsystem um:

16	8	4	2	1
-	1	0	1	0

Tab.4: Dualsystem

ALICE und BOB entschließen sich in diesem Fall für den H(3)-Code, da dieser genau 4 Datenstellen hat.

Die Information Z=10 wird in die Datenstellen eines Codewortes eingetragen:

r	r	k	r	k	k	k
c	c	1	c	0	1	0

Tab.5: Codewort mit Datenbits

Die Errechnung der Prüfstellen erfolgt durch das Einsetzen in die Hamming-Matrix.

c1, c2 und c4 sind die Prüfstellen:

c= -(c+ c+ c) = c+ c+ c Z/2Z
c= c+ c+ c Z/2Z
c1= c+ c_5+ c Z/2Z

c= 0 + 1 + 0 = 1 Z/2Z
c= 1 + 1 + 0 = 0 Z/2Z
c= 1 + 0 + 0 = 1 Z/2Z

Somit hat ALICE nun die Prüfstellen berechnet und kann den Code vervollständigen:

r	r	k	r	k	k	k
1	0	1	1	0	1	0

Tab.6: 7-stelliges Codewort fertig

Die Information Z=10 wurde erfolgreich verschlüsselt und kann nun übertragen werden.

Für den nicht-binären Fall:

Für die Kodierung ändert sich nichts; nur der Rechenaufwand wird größer.
Im Gleichungssystem kommen nun auch höhere Summanden als 1 vor und die Gleichungen werden länger; dementsprechend wird die Arbeit mehr.
Für einen tertiären H(3)-Code, bei dem die Information Z = 1222222221 verschickt werden soll:

r	r	k	k	r	k	k	k	k	k	k	k	k

r	r	1	2	r	2	2	2	2	2	2	2	1

Tab.7: 15-stelliges Codewort mit Datenbits

Matrix H(3):

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \end{pmatrix}$$

Errechnen der Prüfstellen:

c = - (c+c+c+c+c+c+c+c) = - (2+2+2+2+2+2+2+1) = -15 = 0 Z/3Z

c = - (c+c+c+c+c+2c+2c+2c) = - (1+2+2+2+2+4+4+2) = -19 = 2 Z/3Z

c = - (c+2c+c+2c+c+2c+c+2c) = - (1+4+2+4+2+4+2+2) = -21 = 0 Z/3Z

Codewort ist somit : 0212022222221

5.2 Dekodierung

Der Dekodierungsvorgang schließt sich nach der Überprüfung des Codewortes auf Fehler an, was in Kapitel 6 behandelt wird. **Zum Dekodieren muss das Codewort fehlerfrei sein.**

BOB hat ein Codewort empfangen und hat es bereits auf Fehler überprüft.
Das Codewort 1010101 wurde empfangen.

Zum Dekodieren lässt man nur die Prüfstellen des Codewortes weg, **sodass nur noch die Datenbits übrig bleiben.**
Ein sehr trivialer Vorgang, der schnell vollzogen ist:

1	0	1	0	1	0	1
--	--	1	--	1	0	1

Tab.8: Dekodiertes Codewort

Bei der Dekodierung von binären und nicht-binären Codewörtern gibt es keinen Unterschied, außer dass bei nicht-binären die Prüfbits natürlich an anderer Stelle sind.

Wie man sieht, macht der Aufbau des Hamming-Codes die Kodierung und Dekodierung relativ einfach und kurz.

6. Fehlerkorrektur (aus Quelle: [I1] und [H7])

Die Fehlerkorrektur des Hamming-Codes ist die einfachste und eleganteste Methode für Fehlerkorrektur überhaupt. Hamming-Codewörter können **Fehler an einer Stelle perfekt korrigieren.** Bei **Fehlern an 2 Stellen wird die Korrektur fehlerhaft**, es wird jedoch registriert, dass ein Fehler vorhanden sein muss.

6.1 Binäre Codes

ALICE überträgt ein Codewort C = 1010101. Durch einen Übertragungsfehler wurde ein Bit verfälscht, BOB erhält ein fehlerhaftes Codewort C = 1000101, von dem er nicht weiß, ob es fehlerhaft ist oder nicht.
BOB muss das Codewort auf Fehler überprüfen:

1	0	0	0	1	0	1

Tab.9: Codewort mit Fehler in c

Grundsätzlich hat er dazu 2 verschiedene Möglichkeiten:

1.Methode : Überprüfung der Prüfgleichungen:

BOB rechnet die Prüfstellen nach:

c= c+ c+ c Z/2Z
c= c+ c+ c Z/2Z
c= c+ c_5+ c Z/2Z

c= 1+0+1= 0 Z/2Z

c= 0+0+1= 1 Z/2Z ▪

c= 0+1+1= 0 Z/2Z ▪

Die Gleichungen von c und c sind falsch.
Nun, wir wissen leider nicht, welche Stelle im Codewort nun falsch ist.
Zu diesem Zweck sehen wir uns noch einmal die Prüfgleichungen genauer an.

Wenn man genau hinsieht, so erkennt man, dass die Stelle x eines Datenbit im Codewort charakterisiert ist durch die Stelle der Prüfziffern, in deren Prüfgleichung die Datenbit vorkommt.
Das bedeutet, dass c als einzige Stelle aller c in allen 3 Gleichungen vorkommt, nämlich in c, c und c. Man könnte c auch durch c = c ausdrücken.
Oder man könnte c durch c ausdrücken, da c nur in der c- und c-Gleichung vorkommt.

Dies ist kein Zufall, denn diese Rechnung funktioniert für alle c eines binären Codewortes.
Dabei ist es egal, ob es sich um ein Daten- oder ein Prüfbit handelt, denn Prüfbits sind genauso charakterisiert durch ihre jeweilige Prüfgleichung.

Man kann sogar so weit gehen, dass ein Fehler in c von allen drei Gleichungen registriert werden, während ein Fehler in c nur von den Prüfgleichungen von c und c registriert werden.
Nun, wir haben berechnet, dass die Gleichungen für c und c falsch sind.
c ist charakteristisch für diese beiden Gleichungen. c kommt ausschließlich in den Gleichungen für c und c vor.
Somit muss der Fehler in c = c liegen.

Allgemein gilt für binäre Codewörter:
Sind die Prüfgleichungen für die Prüfbits c, c, c, ... falsch,
so liegt der Fehler bei c.
Dies gilt für beliebig viele, falsche Prüfgleichungen.

Da wir hier im binären System rechnen, müssen wir uns keine Gedanken darüber machen, welche Zahl anstelle der 0 in c richtig ist. Es gibt nur eine Möglichkeit, und das ist 1.

Somit heißt das korrigierte Codewort C = 1010101 = C

2.Möglichkeit: Skalarmultiplikation mit der Prüfmatrix:

Die zweite Möglichkeit ist die eleganteste **Variante, Ein-Bit-Fehler in Hamming-Codewörtern zu korrigieren.**

Wir benutzen hierfür das sogenannte **"Syndrom S".**
Das Syndrom entsteht, wenn wir ein Codewort mit der spalten- und zeilenvertauschten Prüfmatrix H skalarmultipliziert.

Aus der Prüfmatrix H: **wird die umgekehrte Matrix H^T:**

In H sind die Zeilen die Zahlen von 1 bis 7 im Dualsystem dargestellt.

C sei hierbei ein beliebiges Codewort aus C.
Das Syndrom S errechnet sich nun aus dem Term **C x H.**
Das Syndrom für C x H ist 000.

1010101 x = 000

Zum Rechenvorgang:
Man multipliziert c des Codewortes mit der ersten Ziffer der ersten Spalte, c mit der zweiten Ziffer der ersten Spalte, c mit der dritten Ziffer der ersten Spalte...
Diese Produkte summiert man auf und werden die erste Ziffer des Syndroms in Z/2Z.

Danach multipliziert man jede Stelle c mit den Werten der 2. Spalte, danach mit der dritten Spalte und summiert diese auf.
So entsteht das Syndrom S(C).

Da jedes richtige Codewort aus der Prüfmatrix H (bzw. H entsteht), muss das Syndrom bei der Multiplikation von einem richtigen Codewort C und H Null sein.
Eine 1 im Syndrom bedeutet also einen Fehler.

Man bemerkt beim genaueren Überlegen, dass wenn wir an die 1.Stelle des Codewortes C einen Fehler setzen, dann erkennt nur die Dritte Spalte von H einen Fehler (für die erste und zweite Spalte bleibt das Produkt von c mit der ersten Zahl der ersten und zweiten Spalte immer null, egal ob mit oder ohne Fehler). Dieser Fehler lässt sich im Syndrom erkennen, dass dann 001 wäre.
Wenn wir den Fehler an die 2.Stelle setzen, erkennt nur die 2.Spalte einen Fehler (S=010).
Setzen wir den Fehler nun an die Dritte Stelle, so erkennen die zweite und die dritte Spalte einen Fehler (011).
Allgemein lässt sich sagen, dass nur Spalten, die an der x-ten Position eine 1 haben, einen Fehler im Codewort an der x-ten Stelle erkennen.
Somit lässt sich erkennen, **dass das Syndrom S die binäre Darstellung derjenigen Stelle ist, in der der Fehler liegt.**
Ist nämlich die erste Stelle falsch, so ist das Syndrom S=001, was die duale Darstellung von 1 ist. Ist der Fehler an der 2. Stelle, so ist das Syndrom S=010 (=2). Ist der Fehler an Stelle drei, so ist das Syndrom S=011, die 3.

Beweis:

Einen Beweis für diese Tatsache lässt sich finden, indem man die Codewörter als **Vektoren** sieht:
Sei C ein richtiges Codewort 1010101, und sei der zugehörige Vektor = .

Und sei ein Vektor mit der Länge eins, der eine 1 an der x-ten Position hat (alle anderen Positionen 0).

So ist + = , ein Vektor, der ein Codewort mit einem Fehler an der x-ten Stelle darstellt.

Da Hamming-Codes perfekt sind, gilt - Є C.
Da der Hamming-Abstand beim Hamming-Code 3 ist, gibt es für jeden Vektor genau ein Codewort , dass sich von ihm nur um unterscheidet.

Das Syndrom berechnet sich nun so:

S() = x H = (+) x H = (x H) + (x H) = **x H**, da x H = 0.
Somit wäre bewiesen, dass das Syndrom nur von dem Fehler an der x-ten Stelle abhängt.

S() ist die x-te Spalte von H. Diese repräsentiert die Nummer x. Rechnet man das Syndrom aus, so kann die fehlerhafte Stelle leichtgefunden werden.

Empfängt man also ein fehlerhaftes Codewort, so kann mithilfe der Skalarmultiplikation die fehlerhafte Stelle berechnen und korrigieren:

1	0	0	0	1	0	1

Tab.10: Codewort mit Fehler in c

1000101 x = 011 (= 3)

Da wir einen Binärcode behandelt, kann die richtige Ziffer an der dritten Position des Codewortes nur 1 lauten.

6.2 Nicht-binäre Codes

Als Beispiel für alle nicht-binären Codewörter wird der **H(3)** verwendet.

ALICE überträgt ein Codewort C= 1101110211201
Leider ist die 7.Stelle fehlerhaft übertragen worden:

r	r	k	k	r	k	k	k	k	k	k	k	k
1	1	0	1	1	1	2	2	1	1	2	0	1

Tab.11: Codewort mit Fehler bei c

Für nicht-binäre Codes ist die Prüfgleichungs-Methode zu zeitintensiv und kompliziert, deswegen ist bei nicht-binären Codes die **Skalarmultiplikation die bevorzugte Methode**, die ich hier ausführe.

Doch nicht nur die Prüfgleichungen werden schwerer, auch die Skalarmultiplikation wird schwerer.
Denn:
Ein Syndrom S() = 201, sollte es aus irgendeinem falschen Codewort berechnet worden sein, **kommt nicht in der Prüfmatrix H vor, und somit stellt S = 201 keine Zeile in H dar.**

Allgemein gilt:
Sei V ein fehlerhaftes Codewort, S() das Syndrom von x H, und α der maßgebliche Wert von S(). Ist S() nun ein Term, der nicht in H vorkommt, so ist der Fehler an der x-ten Stelle, wenn $\alpha^{-1} \bullet$ S() die x-te Spalte in H ist.

Für einen fehlerhaften Code V = 1101112211201 gilt dann:
x H = 201 = S()

α = 2 ist der maßgebliche Wert von S(), somit kann das Syndrom umgeschrieben werden in
$\alpha^{-1} \bullet$ S() = $\bullet$ (201) = (102)
(102) ist die 7.Spalte von H, somit ist der Fehler an der 7. Stelle des Codewortes.

Doch es schließt sich ein weiteres Problem an:
An der siebten Stelle von V befindet sich eine 2.
Da man einen tertiären Code hat, kann man nun nicht so einfach sagen, ob die richtige Zahl eine 0 oder eine 1 ist.

Hierfür gilt:

- α • = C mit als Fehlervektor mit Länge 1.

Somit gilt für das Beispiel:

- 2 • =

in Rechnung:
- 2 • =

Somit wäre das falsche Codewort **fehlerfrei und fertig zum Dekodieren.**

6.3 2-Bit-Fehler

Wieso 2-Bit-Fehler nicht korrigert werden können, liegt am Hamming-Abstand.
Der ist beim Hamming-Code 3.

Das bedeutet, wenn wir ein Codewort C übertragen wollen, und es geschehen Fehler in 2 Stellen des Codewortes, so zeigt uns das Skalarverfahren (oder das Gleichungsverfahren), dass genau **ein** Fehler vorliegt. Korrigieren wir diesen, so haben wir ein **völlig neues Codewort** C aus C, das mit dem ursprünglichen Codewort nichts mehr zu tun hat.

Ein Beispiel:
C = 1010101 soll übertragen werden.
V = 0110101 kommt bei BOB an und enthält 2 Fehler (1. und 2. Stelle).

Beim Skalarverfahren kommt BOB auf ein Syndrom S(V) = 011, somit wird BOB ein Fehler an der 3.Stelle angezeigt.
Er korrigiert den Fehler und kommt auf ein Codewort C = 0100101, das ein völlig anderes als C ist.
Dies geschieht, weil die **Prüfverfahren immer nach 1-Bit-Fehler zum nächsten, ähnlichsten Codewort suchen**. Da der Hamming-Abstand 3 ist, **können 2-Bit-Fehler nicht korrigiert werden.**

7.Schluss

Der Hamming-Code ist der beste Code, um Fehler an einer Stelle auszubessern.
Der Code lässt sich sehr einfach kodieren und dekodieren und Fehler lassen sich mit einem relativ einfachen Verfahren sehr gut ausbessern.
Der Grund für diese anwenderfreundlichen Verfahren ist die Hamming-Matrix, die einen sehr einfachen Aufbau hat.
Das schwierige beim Hamming-Code ist jedoch zu verstehen, wieso der Code so gut funktioniert. Die theoretischen Anteile in dieser Arbeit überwiegen, weil die theoretische Komponente des Hamming-Code sehr schwer verständlich zu erklären ist.
Das gute jedoch ist, dass ein Anwender diese Theorie nicht verstehen muss.
Man zeigt dem Anwender ein einfaches Verfahren und er kann in kürzester Zeit jeden Fehler berechnen.
Und genau deshalb ist der Hamming-Code **einer der bekanntesten Codes der Kodierungstheorie.**

Der Nachteil des Hamming-Code ist, dass er nur eine Stelle korrigieren kann. Hamming-Codes können immer nur einen Fehler korrigieren, auch falls das Codewort 1000 Stellen hat. Somit ist die Fehler-Korrektur-Rate für lange Codewörter sehr schlecht.

Literaturverzeichnis

[Q1] Mathe-Unterricht der 11.Klasse bei Markus Meiringer

[H1] http://de.wikipedia.org/wiki/Hamming-Code
(Titel: Hamming-Code
Autor: unbekannt
Letzter Aufruf: 02. Juni 2007)

[H2] http://en.wikipedia.org/wiki/Hamming_code
(Titel: Hamming-Code
Autor: unbekannt
Letzter Aufruf: 02. Juni 2007)

[H3] http://www.computing.dcu.ie/~humphrys/Notes/
Networks/data.hamming.html
(Titel: Hamming-Code 1 bit error correcting
Autor: unbekannt
Letzter Aufruf: 07. Juni 2007)

[H4] http://www.cs.fiu.edu/~downeyt/cop3402/hamming.html
(Titel: Calculating the Hamming-Code
Autor: unbekannt
Letzter Aufruf: 16. Juni 2007)

[H5] http://einstein.informatik.uni-oldenburg.de/rechnernetze/hamming-.htm
(Titel: Hamming-Codes
Autor: unbekannt
Letzter Aufruf: 16. Juni 2007)

[H6] http://www.inf.fh-flensburg.de/lang/algorithmen/code/hamming.htm
(Titel: Codierungstheorie : Hamming-Code
Autor: H.W.Lang
Letzter Aufruf: 16. Juni 2007)

[H7] http://www.mdstud.chalmers.se/~md7sharo/coding/main/node32.html
(Titel: Hamming-Code
Autor: unbekannt
Letzter Aufruf: 25. Januar 2009)

[I1] Introduction to Coding and Information Theory
(Steven Roman, Springer Verlag, 2000)

Erklärung

"Ich erkläre hiermit, dass ich die Facharbeit ohne fremde Hilfe angefertigt und nur die im Literaturverzeichnis angeführten Quellen und Hilfsmittel benutzt habe."

……..	…….	…….
Ort	Datum	Unterschrift des Schülers