

fehlerkorrigierende Codes

Thomas Irgang

Katholische Universitt Eichsätt

28. Februar 2007

Was sind fehlerkorrigierende Codes ?

Mathematik und fehlerkorrigierende Codes

Was fehlerkorrigierende Codes nicht sind

- ▶ es geht nicht um Verschlüsselung

Was fehlerkorrigierende Codes nicht sind

- ▶ es geht nicht um Verschlüsselung
- ▶ Daten sollen trotz Fehlern wieder herstellbar sein

Definition fehlerkorrigierender Codes

Fehlerkorrigierende Codes sind Codes, die es ermöglichen trotz Fehlern, die bei der Übertragung oder Speicherung von Daten auftreten, die Daten wieder herzustellen.

Wozu braucht man fehlerkorrigierende Codes

- ▶ Internetverbindung

Wozu braucht man fehlerkorrigierende Codes

- ▶ Internetverbindung
- ▶ DVB-T

Wozu braucht man fehlerkorrigierende Codes

- ▶ Internetverbindung
- ▶ DVB-T
- ▶ CD, DVD...

Modellvorstellung

- ▶ ein Sender sendet eine Nachricht
- ▶ die Nachricht wird über einen Kanal übertragen
- ▶ beim übertragen treten Fehler auf
- ▶ der Empfänger versucht die Nachricht wiederherzustellen

Problemstellung

- ▶ Kann man an Hand der Nachricht erkennen ob ein Fehler aufgetreten ist?

Problemstellung

- ▶ Kann man an Hand der Nachricht erkennen ob ein Fehler aufgetreten ist?
- ▶ Es müssen zusätzliche Daten übertragen werden um die „Nutzdaten“ validieren zu können.

Problemstellung

- ▶ Kann man an Hand der Nachricht erkennen ob ein Fehler aufgetreten ist?
- ▶ Es müssen zusätzliche Daten übertragen werden um die „Nutzdaten“ validieren zu können.
- ▶ Was kann man machen wenn Fehler festgestellt werden?

Problemstellung

- ▶ Kann man an Hand der Nachricht erkennen ob ein Fehler aufgetreten ist?
- ▶ Es müssen zusätzliche Daten übertragen werden um die „Nutzdaten“ validieren zu können.
- ▶ Was kann man machen wenn Fehler festgestellt werden?
- ▶ Die einfachse Möglichkeit ist es das Datenpaket nochmal zu senden. Dies funktioniert aber nicht immer!
Welche Möglichkeiten gibt es wenn die Nachricht nicht nochmal gesendet werden kann (z.B. verkratzte CD). Wir brauchen Verfahren, die die Daten wiederherstellen können.

Ein Ansatz

Die Codewörter müssen sich paarweise jeweils in mehreren Bits unterscheiden! Was hilft das?

Ein Ansatz

Die Codewörter müssen sich paarweise jeweils in mehreren Bits unterscheiden! Was hilft das?

Dadurch gibt es zwischen je zwei Codewörtern einen Bereich in dem kein Codewort liegt. Wenn jetzt weniger Fehler als dieser Minimalabstand von zwei Codewörtern auftreten, ergibt sich ein ungültiges Codewort.

⇒ *es kann mehr als ein Fehler erkannt werden.*

Ein Ansatz

Die Codewörter müssen sich paarweise jeweils in mehreren Bits unterscheiden! Was hilft das?

Dadurch gibt es zwischen je zwei Codewörtern einen Bereich in dem kein Codewort liegt. Wenn jetzt weniger Fehler als dieser Minimalabstand von zwei Codewörtern auftreten, ergibt sich ein ungültiges Codewort.

⇒ *es kann mehr als ein Fehler erkannt werden.*

Wie können mit diesem Ansatz Fehler korrigiert werden?

Die Idee

Wenn der Code einen Minimalabstand von ≥ 3 hat, und nur ein Fehler auftritt, können wir das veränderte Codewort eindeutig einem Codewort zuweisen und damit einen Fehler korrigieren.

Ein Beispiel

Gegeben sind die Codewörter a, b die jeweils aus einer Folge von vier Einsen und Nullen bestehen: $a := 0011$, $b := 1110$

a und b unterscheiden sich in: $\left. \begin{array}{l} 0011 \\ 1110 \end{array} \right\}$ drei Stellen.

Wenn wir jetzt ein verändertes Codewort x empfangen, z.B. $x = 0110$, können wir es mit a und b vergleichen

a und x : $\left. \begin{array}{l} 0011 \\ 0110 \end{array} \right\}$ unterscheiden sich in zwei Stellen.

b und x : $\left. \begin{array}{l} 1110 \\ 0110 \end{array} \right\}$ unterscheiden sich in einer Stelle.

Damit können wir, unter der Annahme einer minimalen Fehlerzahl, x als b decodieren.

Der Hammingabstand

Die Anzahl der Stellen in denen sich zwei Codewörter unterscheiden bezeichnet man als *Hammingabstand*.

Ein Code ist eine Menge von Codewörtern. Der minimale Hammingabstand eines Codes ist das Minimum des Abstands das zwei beliebige Codewörter des Codes haben.

Aufgaben

$c(0)=00011$, $c(1)=00110$, $c(2)=01100$, $c(3)=11000$

1. Bestimme den minimalen Hamming-Abstand der Codewörter $c(0)$, $c(1)$, $c(2)$, $c(3)$!
2. Kann man das veränderte Codewort $x=00010$ eindeutig decodieren?
3. Beschreiben Sie den Zusammenhang zwischen dem minimalen Hamming-Abstand eines Codes und seiner Eigenschaften bezüglich des Erkennens und Korrigierens von Fehlern?

Minimalabstand und Fehlereigenschaften

Der Code im Beispiel hat einen Minimalabstand von 2. Das hat zur Folge, dass es passieren kann, dass wenn nur ein Bit im übertragenen Wort verändert ist, das veränderte Wort Abstand 1 zu zwei Codewörtern hat. Daraus folgt: Ein Code C ist ein *t-fehlerkorrigierender* Code falls der Minimalabstand zwischen zwei Codewörtern $\geq 2t + 1$ ist. Ein solcher kann dann Code kann $2t$ Fehler erkennen.

Es muss also um jedes Codewort eine Umgebung mit Radius t geben in der in der jedes Wort eindeutig dem Codewort zugeordnet werden kann.

Was hat das ganze mit Mathe zu tun?

um Codes anwenden zu können muss es möglich sein effektiv mit ihnen umzugehen! D.h.:

- ▶ Daten müssen schnell und einfach codiert werden können und

Was hat das ganze mit Mathe zu tun?

um Codes anwenden zu können muss es möglich sein effektiv mit ihnen umzugehen! D.h.:

- ▶ Daten müssen schnell und einfach codiert werden können und
- ▶ Daten müssen schnell und einfach wiederhergestellt und decodiert werden können.

An dieser Stelle kommt die lineare Algebra ins Spiel.

Vektoren und Codes

Wir können die Codewörter als Vektoren auffassen, dazu brauchen wir aber einige neue Regeln wie die Multiplikation und Addition mit 0 und 1 funktioniert. Dazu wird folgendes definiert:

- ▶ $0 + 0 = 0$, $0 + 1 = 1$, $1 + 1 = 0$ und

Vektoren und Codes

Wir können die Codewörter als Vektoren auffassen, dazu brauchen wir aber einige neue Regeln wie die Multiplikation und Addition mit 0 und 1 funktioniert. Dazu wird folgendes definiert:

- ▶ $0 + 0 = 0$, $0 + 1 = 1$, $1 + 1 = 0$ und
- ▶ $0 \cdot 0 = 0$, $0 \cdot 1 = 0$, $1 \cdot 1 = 1$

Die Vektoren haben dabei beliebige Länge, die Addition von zwei Vektoren und die Multiplikation mit Skalaren, in diesem Fall nur 0 und 1, ändert sich dabei nicht.

linearer Codes

Als linearer Code wird ein Code bezeichnet, bei dem sich die Codewörter mit Hilfe von wenigen Codewörtern erzeugen lassen. D.h. wenn wir Codewörter a , b , c haben sind auch $a+b$, $a+c$, $b+c$, $a+b+c$, ... Codewörter.

Vorteile linearer Codes

Daraus ergeben sich einige Vorteile:

- ▶ Man kann mit Hilfe von wenigen Codewörtern den ganzen Code speichern

Vorteile linearer Codes

Daraus ergeben sich einige Vorteile:

- ▶ Man kann mit Hilfe von wenigen Codewörtern den ganzen Code speichern
- ▶ Der Code kann mit Hilfe von Matrizen “verarbeitet” werden.

Verarbeitet bedeutet in diesem Zusammenhang dass Daten codiert, decodiert und Fehler korrigiert werden.

Matrizenmultiplikation

Im folgenden brauchen wir die Multiplikation von Vektoren mit Matrizen. Diese Multiplikation ist allgemein definiert durch:

$$\begin{pmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & \dots & a_{2n} \\ \dots & \dots & \dots \\ a_{m1} & \dots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \\ \dots \\ v_4 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n a_{1i} \cdot v_i \\ \sum_{i=1}^n a_{2i} \cdot v_i \\ \dots \\ \sum_{i=1}^n a_{mi} \cdot v_i \end{pmatrix}$$

(also: Zeile \cdot Spalte)

Ein Beispiel dazu:

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \cdot 1 + 0 \cdot 1 + 0 \cdot 0 \\ 1 \cdot 1 + 1 \cdot 1 + 0 \cdot 0 \\ 0 \cdot 1 + 1 \cdot 1 + 1 \cdot 0 \\ 0 \cdot 1 + 0 \cdot 1 + 0 \cdot 0 \\ 0 \cdot 1 + 0 \cdot 1 + 0 \cdot 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Ein Beispiel

Die Matrix $G :=$
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$
 (G für Generatormatrix)

erzeugt einen linearen 1-fehlerkorrigierenden Code.

Ein Beispiel

Wenn jetzt z.B. das Datenwort $x := \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$ codiert werden soll

rechnet man einfach $G \cdot x = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$. Dadurch muss man statt 16

(= 2^4) Codewörtern nur 4 speichern und kann die Daten mit Hilfe der Matrix schnell codieren.

Ein Beispiel

Damit können wir Daten effektiv codieren und Codes effektiv speichern. Um auch effektiv decodieren zu können brauchen wir noch etwas mehr Mathematik.

Minimalgewicht

Als *Gewicht* eines Vektors wird seine Anzahl der von 0 verschiedenen Stellen bezeichnet.

Das Minimalgewicht eines Codes ist, analog zum Hammingabstand, das Minimum über das Gewicht über alle Codewörter.

Minimalgewicht = Minimalabstand

Es ergibt sich, dass bei linearen Codes das Minimalgewicht gleich dem Minimalabstand ist.

Daraus ergibt sich ein großer Vorteil: statt $|C|$ Codewörter miteinander zu vergleichen muss man nur das Minimalgewicht der $|C|$ Codewörter bestimmen.

Dadurch lassen sich die Fehlerkorrektureigenschaften von einem linearen Code schnell erkennen, aber Vorsicht, man muss dennoch alle Codewörter betrachten denn z.B. die Codewörter $a := 011100$ und $b := 111000$ haben beide Gewicht 3, der lineare Code zu dem diese gehören muss aber auch $a + b$ enthalten. Damit ergibt sich:

$$\left. \begin{array}{r} 111000 \\ + 011100 \end{array} \right\} = 100100 \text{ und damit hat der Code ein}$$

Minimalgewicht von ≤ 2 .

Aufgabe

Wie viele Fehler kann der folgende lineare Code erkennen/korrigieren?

0000000	0001110	0010111	0011001
0100101	0101011	0110010	0111100
1111111	1110001	1101000	1100110
1011010	1010100	1001101	1000011

Wie kann man effektiv decodieren?

Wir können jetzt Datenwörter einfach codieren und einen Code mit Hilfe von 2^k geschickt gewählten Codewörter darstellen und die Eigenschaften eines solchen linearen Codes schnell einschätzen. Aber wie können wir einfach fehlerkorrigieren und die Codewörter wieder decodieren?

der duale Code

Um einen linearen Code effektiv zu decodieren brauchen wir den dualen Code C^\perp . Was ist das?

V ist die Menge aller möglichen Codewörter mit n Zeichen. Der duale Code enthält die Codewörter für die gilt $c \cdot v = 0$. $c \cdot v$ bedeutet dabei $c \cdot v = \sum_{i=1}^n c_i \cdot v_i$

die Kontrollmatrix

Wenn wir den dualen Code haben können wir die Kontrollmatrix angeben und sind damit wieder einen Schritt näher am Ziel. Eine Matrix deren Zeilen eine Basis (d.h. der duale Code lässt sich mit diesen Vektoren erzeugen und die Anzahl dieser Vektoren ist Minimal) des dualen Codes sind nennt man *Kontrollmatrix* des linearen Codes, das Produkt $H \cdot s$ nennt man das *Syndrom*. Mit Hilfe des Syndroms können wir nun endlich einen linearen Code effektiv decodieren.

der Decodieralgorithmus

Jetzt kann man mit Hilfe einer Liste der Nebenklassenanhufer folgendermaen Fehler korrigieren:

1. Syndrom vom empfangen Wort x berechnen
2. Fehlervektor e mit Hilfe der Liste der Syndrome feststellen
3. Codewort c durch $c = x + e$ berechnen

vollständiges Beispiel : Generatormatrix

Diesen Algorithmus können wir jetzt mal an einem Beispiel durchspielen. Wir benutzen wieder die Generatormatrix

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

vollständiges Beispiel : Kontrollmatrix I

Als erstes brauchen wir wieder eine Kontrollmatrix, also brauchen wir den dualen Code. Wir brauchen also drei Vektoren die den dualen Code erzeugen.

Dies ist z.B. bei den Vektoren $\begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$, $\begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$, $\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}$ der Fall.

Dies können wir nachrechnen.

vollständiges Beispiel : Kontrollmatrix II

Die Matrix $K := \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$ ist also eine
Kontrollmatrix zu G .

vollständiges Beispiel : Syndrome

Mit Hilfe der Kontrollmatrix können wir jetzt die Syndrome berechnen. Wir berechnen die Syndrome der Einheitsvektoren, da dies alle möglichen Fehlervektoren sind. Dadurch erhalten wir:

0000000	000
0000001	111
0000010	011
0000100	101
0001000	110
0010000	001
0100000	010
1000000	100

vollständiges Beispiel : Codewörter

Jetzt müssen wir nur noch die Codewörter ihren Datenwörtern zuordnen. Das ist aber einfach, denn wenn wir die Generatormatrix betrachten, sehen wir, dass die ersten 4 Bit des Codeworts das Datenwort sind. Es reicht also, die letzten drei Bit abzuschneiden. Jetzt haben wir alles um den Algorithmus anzuwenden.

vollständiges Beispiel : Codierung

Nehmen wir uns ein Datenwort z.B. $x = 0110$. Als erstes codieren wir das Datenwort durch Multiplikation mit der Generatormatrix.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

vollständiges Beispiel : Übertragung

Wenn jetzt bei der Übertragung ein Fehler passiert und aus dem

Codewort z.B. $\begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$ wird

vollständiges Beispiel : Decodierung I

ergibt die Multiplikation mit der Kontrollmatrix

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} =$$

$$\begin{pmatrix} 1+0+0+0+0+0+0 \\ 0+1+0+0+0+1+0 \\ 0+0+1+0+0+1+0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

vollständiges Beispiel : Decodierung II

Dadurch erhalten wir den Fehlervektor $\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ und

$$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}. \text{ Das Datenwort ist demnach } \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

was korrekt ist. Die Korrektur funktioniert also.