Lineaire codes

Bij het versturen van digitale informatie worden in principe ketens van bits verstuurd die de waarde 0 of 1 kunnen hebben. Omdat de transmissiekanalen door ruis verstoord zijn, treden er tijdens de transmissie fouten op die sommige van de bits veranderen. Een soortgelijk probleem treedt ook bij het afspelen van CD's en DVD's op, want door beschadigingen kunnen delen van de informatie incorrect of onleesbaar zijn.

De gebruikelijke methode om met dit soort fouten om te gaan is de informatie *redundant* op te slaan of te versturen, d.w.z. in totaal een groter aantal bits te versturen dan nodig. Een eenvoudige manier bestaat bijvoorbeeld erin, iedere bit twee of drie keer te sturen. Het zal duidelijk zijn dat de mogelijkheid tot herkennen en verbeteren van transmissiefouten stijgt naarmate er meer redundante informatie aan een signaal wordt toegevoegd. Het doel is met zo weinig mogelijk additionele informatie zo veel mogelijk fouten te kunnen herstellen.

De tak van de wiskunde die zich met dit soort vraagstukken bezig houdt is de *coderingstheorie*.

1. Binaire lineaire codes

Definitie: Een *binaire lineaire code* van lengte n is een lineaire deelruimte van de vectorruimte $\mathbb{F}_2^n$ van n-tupels over het lichaam $\mathbb{F}_2$ met 2 elementen.

Algemeen definieert men een *lineaire code* als lineaire deelruimte van een vectorruimte over een willekeurig eindig lichaam. Vooral het lichaam met vier elementen speelt soms een rol, maar het lichaam $\mathbb{F}_2$ met twee elementen is bij verre het meest belangrijke geval.

> Er zijn ook codes die niet *lineair* zijn, d.w.z. deelverzamelingen van $\mathbb{F}_2^n$ die geen lineaire deelruimte vormen. Het is echter veel moeilijker, eigenschappen van deze codes te bepalen, daarom zullen we ons hier tot lineaire codes beperken.

We zullen vanaf nu voor het gemak met een *lineaire code* altijd een *binaire* lineaire code bedoelen.

Voorbeelden:

(1) Een flauw voorbeeld van een code is de vectorruimte $\mathbb{F}_2^n$ zelf.

(2) Een ook niet erg interessant voorbeeld is de deelruimte die alleen maar de nulvector en de vector $(1, 1, \ldots, 1)$ bevat.

(3) Het herhalen van bits laat zich met de volgende code realiseren:

$$C_h := \{x = (x_1, \ldots, x_n, y_1, \ldots, y_n) \in \mathbb{F}_2^{2n} \mid x_1 = y_1, \ldots, x_n = y_n\} \subset F_2^{2n}$$

Een keten van n bits wordt gewoon herhaald. Merk op dat dit gunstiger is dan iedere bit rechtstreeks te herhalen, omdat fouten niet gelijkmatig verdeeld optreden maar op zekere plekken cumuleren.

Als we de boodschap `1011` met behulp van deze code willen versturen, moeten we het signaal `10111011` versturen. Merk op dat we hier de vector $(1, 0, 1, 1, 1, 0, 1, 1)$ gewoon als keten van zijn componenten hebben geschreven.

(4) Bij een *parity check code* wordt een controlebit aan de boodschap toegevoegd dat 0 is als de boodschap even veel 1en bevat en 1 als de boodschap oneven veel 1en bevat. De boodschap samen met dit controlebit heeft dus altijd een even aantal 1en. Omdat we over het lichaam $\mathbb{F}_2$ werken, is het aantal 1en in een vector juist even als de som over de componenten van de vector 0 is, daarom laat zich een parity check code als volgt beschrijven:

$$\begin{aligned} C_p &:= \{x = (x_1, \ldots, x_n, x_{n+1}) \in \mathbb{F}_2^{n+1} \mid x_1 + \ldots + x_n + x_{n+1} = 0\} \subset \mathbb{F}_2^{n+1} \\ &= \{x = (x_1, \ldots, x_n, x_{n+1}) \in \mathbb{F}_2^{n+1} \mid x_{n+1} = x_1 + \ldots + x_n\} \end{aligned}$$

Als we de boodschap `1011` met behulp van deze code willen versturen, moeten we het signaal `10111` versturen.

2. Herkennen en verbeteren van fouten

De grap bij het gebruik van een lineaire deelruimte C van $\mathbb{F}_2^n$ voor het versturen van informatie is dat alleen maar de elementen van C geldige signalen vormen. Als bij een element van C een bit wordt veranderd zal het vaak geen element van C meer zijn.

Voorbeeld: Stel bij de herhalingscode van boven wordt de derde bit omgeschakeld, d.w.z. het ontvangen signaal is `10011011`. Dan zien we dat dit geen element van C_h is, want $x_3 \neq y_3$. We kunnen echter niet zeggen of de eerste helft (`1001`) of de tweede helft (`1011`) van het ontvangen signaal de correcte boodschap weergeeft, want we weten natuurlijk niet welke bit veranderd is.

Stel dat ook bij de parity check code de derde bit omgeschakeld wordt, dan is in dit geval het ontvangen signaal `10011`. Dit is geen element van C_p, omdat de som van de componenten 1 en niet 0 geeft. We weten daarom dat er een fout gebeurt is, maar we kunnen niet achterhalen welke bit veranderd is.

Merk op dat we in beide gevallen ervan uit zijn gegaan dat er slechts één fout is gebeurd. Bij twee fouten kan het zijn dat we weer een geldig signaal, d.w.z. een element van C ontvangen.

Voor de herhalingscode zien we bij het ontvangen van `10001011` bijvoorbeeld wel dat er (minstens) twee fouten zijn opgetreden, maar bij `10011001` zou het kunnen dat de boodschap `1001` zonder fout is verstuurd.

Bij de parity check code geldt zelfs dat twee fouten altijd weer een element van C opleveren.

Om systematisch na te gaan hoe we met behulp van een lineaire code fouten kunnen herkennen en verbeteren, hebben we een paar begrippen nodig.

Definitie:

(i) Het *gewicht* $wt(v)$ van een vector $v \in \mathbb{F}_2^n$ is het aantal componenten die niet 0 zijn. Bijvoorbeeld is $wt(\texttt{10111011}) = 6$.

(ii) De *Hamming afstand* $d(v, w)$ (of gewoon *afstand*) van twee vectoren $v, w \in \mathbb{F}_2^n$ is het aantal componenten waarop v en w verschillen. Bijvoorbeeld is $d(\texttt{10111011}, \texttt{10001011}) = 2$.

Men gaat eenvoudig na dat $d(v, w) = wt(v + w)$. Hiermee laat zich het volgende lemma aantonen.

Lemma: De minimale Hamming afstand $d(v, w)$ die twee elementen $v \neq w$ van een lineaire code $C \subset \mathbb{F}_2^n$ hebben, is gelijk aan het minimale gewicht van een element $v \in C, v \neq 0$.

Dit minimale gewicht van een element $0 \neq v \in C$ noemt men ook het *minimale gewicht van de code C*.

Voorbeeld: De herhalingscode en de parity check code van boven hebben beide het minimale gewicht 2:

Voor de herhalingscode geldt dit omdat iedere 1 uit de eerste helft in de tweede helft herhaald wordt, dus is het minimale gewicht minstens 2. De vector $\texttt{10001000} \in C_h$ heeft gewicht 2, dus is het minimale gewicht ook niet groter dan 2.

Voor de parity check code moet het aantal 1en even zijn, dus minstens 2. De vector $\texttt{10001} \in C_p$ heeft gewicht 2, dus is het minimale gewicht inderdaad 2.

Het idee achter het verbeteren van fouten is de aanname dat transmissiefouten slechts met een kleine kans (in relatie tot de lengte n van de vectoren) optreden. Dit maakt het plausibel van zo weinig mogelijk transmissiefouten uit te gaan. Als we dus een vector v' ontvangen die niet in de code C ligt, zoeken we in C een vector v die minimale Hamming afstand van v' heeft. Met andere woorden: we maken van v' weer een geldig signaal v door zo weinig mogelijk bits van v' te veranderen.

Probleem: Als er meerdere vectoren $v \in C$ bestaan die minimale Hamming afstand van v' hebben, kunnen we de fout(en) in v' niet verbeteren.

Voorbeeld: Bij de herhalingscode C_h kunnen we het ontvangen signaal $v' = \texttt{10011011}$ op twee manieren tot een element van C_h veranderen door een enkele bit te veranderen, namelijk tot $v = \texttt{10111011}$ of tot $w = \texttt{10011001}$. Er geldt dus $d(v', v) = d(v', w) = 1$ en we kunnen de opgetreden fout niet herstellen.

Bij de parity check code is de situatie nog onduidelijker, we kunnen namelijk in het ontvangen signaal $v' = \texttt{10011}$ iedere bit veranderen om tot een element van C_p te komen, dit geeft de mogelijke originele signalen $v_1 = \texttt{00011}$, $v_2 = \texttt{11011}$, $v_3 = \texttt{10111}$, $v_4 = \texttt{10001}$ en $v_5 = \texttt{10010}$ met $d(v', v_i) = 1$ voor $i = 1, \ldots, 5$.

Om tot een code te komen die ten minste één fout kan verbeteren, zouden we de boodschap drie keer kunnen herhalen. Voor iedere bit nemen we dan de meerderheidsbeslissing van de drie kopieën en op deze manier laat zich een enkele fout steeds herstellen (omdat de twee andere versies van de bit nog correct zijn).

Maar dit kan beter, we hoeven aan de herhalingscode alleen maar een controlebit toe te voegen:

$$C_{hp} := \{x = (x_1, \ldots, x_n, y_1, \ldots y_n, z) \in \mathbb{F}_2^{2n+1} \mid x_1 = y_1, \ldots, x_n = y_n, \\ x_1 + \ldots + x_n + z = 0\} \subset \mathbb{F}_2^{2n+1}$$

In dit geval zullen we de boodschap `1011` als `101110111` versturen. Als nu weer de derde bit wordt omgeschakeld, ontvangen we `100110111` en we kunnen nu beslissen dat de tweede helft correct is, omdat hiervoor de controlebit klopt. Als alleen maar de laatste bit wordt veranderd, kunnen we dit ook zien, omdat dan de eerste twee blokken nog steeds hetzelfde zijn.

Hoe veel fouten door een lineaire code C kunnen worden herkend en verbeterd hangt alleen maar van het minimale gewicht van C af, er geldt:

Stelling: Zij C een lineaire code met minimaal gewicht d.

(i) Als er bij het versturen van $v \in C$ hoogstens $d - 1$ fouten optreden laat zich dit altijd herkennen.

(ii) Stel bij het versturen van $v \in C$ zijn er k fouten opgetreden. Dan laten zich deze verbeteren als: $\begin{cases} d \text{ even en } k \leq \frac{d}{2} - 1 \\ d \text{ oneven en } k \leq \frac{d-1}{2}. \end{cases}$

Men gaat eenvoudig na dat de boven aangegeven code C_{hp} het minimale gewicht 3 heeft, dus kunnen we het optreden van twee fouten altijd herkennen en één fout altijd verbeteren.

Voorbeeld: Ook als in de twee herhalingen van `1011` dezelfde bit wordt veranderd, kunnen we herkennen dat er fouten zijn opgetreden, want in `100110011` komt de controlebit niet overeen met de twee blokken `1001`.

Men zegt vaak dat een code een zeker aantal van fouten kan herkennen. Merk op dat dit een misleidende formulering is, want de fouten worden niet zelf herkend maar er wordt alleen maar herkend *dat* er fouten zijn opgetreden.

3. Controle cijfers

Soms spelen ook codes over andere lichamen dan $\mathbb{F}_2$ een rol, maar dan meestal als parity check codes. En voor een parity check code is niet eens een onderliggend lichaam nodig, we kunnen met de resten bij delen door een getal n werken dat geen priemgetal hoeft te zijn. We geven hiervoor twee voorbeelden.

3.1 De ISBN-10 code

Het *internationale standaard boeknummer* ISBN was tot voor kort een parity check code van lengte 10 over het lichaam met 11 elementen. Inmiddels is deze code vervangen door een langere code van lengte 13, die afgeleid is uit de EAN code (zie hieronder). De oude boekencode heet daarom nu vaak ISBN-10, de nieuwe ISBN-13.

De ISBN-10 nummers bestaan uit 4 blokken, waarvan de eerste het taalgebied aangeeft, de tweede blok de uitgever binnen het taalgebied en de derde blok het nummer van het boek bij de uitgever. De vierde blok is alleen maar het laatste cijfer en dat is juist een controlecijfer. In dit geval wordt het controlecijfer zo bepaald dat voor een ISBN-10 nummer $a_1 a_2 \ldots a_9 a_{10}$ geldt:

$$10a_1 + 9a_2 + \ldots + 2a_9 + a_{10} \text{ is een 11-voud,}$$

d.w.z. deze som geeft rest 0 bij delen door 11.

Als het controlecijfer 10 moet zijn, wordt dit door het symbool `X` aangegeven (Romeins voor 10).

Voorbeeld: We bepalen het controlecijfer voor het ISBN-10 nummer `3-540-78056-?`: Er geldt $10 \cdot 3 + 9 \cdot 5 + 8 \cdot 4 + 7 \cdot 0 + 6 \cdot 7 + 5 \cdot 8 + 4 \cdot 0 + 3 \cdot 5 + 2 \cdot 6 = 30 + 45 + 32 + 42 + 40 + 15 + 12 = 216 = 7 + 19 \cdot 11$. Zonder het controlecijfer hebben we dus rest 7 bij delen door 11, dus moet het controlecijfer 4 zijn.

De meest gebruikelijke fouten bij het doorgeven of overschrijven van nummers is het vervangen van een cijfer door een andere en het verruilen van twee naburige cijfers. Beide soorten van fouten worden door de ISBN-10 code herkend:

- Als het cijfer a_i door een ander cijfer b_i wordt vervangen, is het verschil van de twee sommen

  $$\begin{aligned} &(10a_1 + \ldots + (11-i)a_i + \ldots + a_{10}) \\ -&(10a_1 + \ldots + (11-i)b_i + \ldots + a_{10}) = (11-i)(a_i - b_i). \end{aligned}$$

  Het nieuwe nummer met b_i in plaats van a_i is alleen maar een geldige ISBN-code als ook dit verschil een 11-voud is. Maar dit kan niet omdat $11 - i$ kleiner is dan 11 en ook a_i en b_i zijn kleiner dan 11. Daarom ligt namelijk $a_i - b_i$ tussen -10 en 10 en is dus geen veelvoud van 11 (behalve als $a_i = b_i$, maar dan is er geen fout gemaakt).

- Als we bijvoorbeeld de cijfers a_2 en a_3 verruilen, is het verschil van de twee sommen

  $$\begin{aligned} &(10a_1 + 9a_2 + 8a_3 + 7a_4 + \ldots + a_{10}) \\ -&(10a_1 + 9a_3 + 8a_2 + 7a_4 + \ldots + a_{10}) = a_2 - a_3. \end{aligned}$$

  Analoog krijgt men bij het verruilen van de cijfers a_i en a_{i+1} het verschil $a_i - a_{i+1}$. Maar ook dit verschil is nooit een 11-voud, dus hebben we na het verruilen geen geldig ISBN nummer meer.

3.2 De EAN code

De *Europese artikelnummering* **EAN** is een code voor bijna alle producten die te koop zijn en is vergezeld met een streepjescode. De meest gebruikelijke vorm van de EAN heeft 13 cijfers, waarvan de laatste een controlecijfer is. Deze wordt als volgt bepaald: De cijfers op de even posities worden bij elkaar opgeteld en met 3 vermenigvuldigd, vervolgens worden de cijfers op de oneven posities erbij opgeteld. Het resultaat moet dan een 10-voud zijn, d.w.z. geeft rest 0 bij delen door 10. M.a.w.: Een code $a_1a_2a_3 \ldots a_{12}a_{13}$ is een geldige EAN als $a_1 + 3a_2 + a_3 + 3a_4 + \ldots + 3a_{12} + a_{13}$ een 10-voud is.

Voorbeeld: Een zeker product heeft de EAN 9 783528 06989? waarvan we het controlecijfer willen bepalen. De som op de even posities is $7 + 3 + 2 + 0 + 9 + 9 = 30$, de som op de oneven posities $9 + 8 + 5 + 8 + 6 + 8 = 44$. Voor het controlecijfer z moet dus gelden dat $3 \cdot 30 + 44 + z$ een 10-voud is, dus moet $z = 6$ zijn.

Men gaat eenvoudig na dat ook de EAN-code fouten door veranderen van een enkele cijfer herkent.

Maar het verruilen van naburige cijfers wordt niet altijd ontdekt, bijvoorbeeld kunnen we in het voorbeeld boven het derde en vierde cijfer verruilen en hebben alsnog een geldig EAN nummer, namelijk 9 738528 069896 (ga dit na). Het is niet moeilijk om in te zien dat het bij het verruilen alleen maar mis gaat als de twee verruilde cijfers verschil 5 hebben.

Merk op: De nieuwe ISBN-13 nummers volgen het schema van de EAN. Omdat de EAN-code niet meer alle fouten door het verruilen van naburige cijfers herkent, is met de overgang van ISBN-10 naar ISBN-13 dus in feite geen vooruitgang geboekt (tenminste wat het herkennen van fouten betreft).

De Amerikaanse UPC zijn codes van 12 cijfers die corresponderen met de EAN-codes waarbij het eerste cijfer 0 is.