



R-Paket für Kanalkodierung

Blockcodes

Gliederung

1. Einführung in Kanalkodierung

1.1 Grundlagen

2. Blockcodes

2.1 Hamming-Kodes

2.2 BCH-Kodes

3. R





Motivation und Ziel

- ▶ Kodierer sind hauptsächlich in der Hardware realisiert.



Motivation und Ziel

- ▶ Kodierer sind hauptsächlich in der Hardware realisiert.
- ▶ Schlecht nachzuvollziehen, Lehre bisher nur theoretisch.



Motivation und Ziel

- ▶ Kodierer sind hauptsächlich in der Hardware realisiert.
- ▶ Schlecht nachzuvollziehen, Lehre bisher nur theoretisch.
- ▶ **Ziel:** R-Paket zur Simulation und Visualisierung der Kanalkodierung.



Grundlagen

- ▶ Kodierung als Abbildung

$$D^k \rightarrow K^n$$



Grundlagen

- ▶ Kodierung als Abbildung

$$D^k \rightarrow K^n$$

- ▶ Koderate R

$$R = \frac{k}{n}$$



Grundlagen

- ▶ Kodierung als Abbildung

$$D^k \rightarrow K^n$$

- ▶ Koderate R

$$R = \frac{k}{n}$$

- ▶ Hamming Distanz d für $x, y \in K$

$$d(x, y) := | \{ i \in \{1, \dots, n\} \mid x_i \neq y_i \} |$$



Grundlagen

- ▶ Kodierung als Abbildung

$$D^k \rightarrow K^n$$

- ▶ Koderate R

$$R = \frac{k}{n}$$

- ▶ Hamming Distanz d für $x, y \in K$

$$d(x, y) := |\{i \in \{1, \dots, n\} \mid x_i \neq y_i\}|$$

- ▶ Beispiel

$$d(1011, 1010) = 1$$



Grundlagen - Fehlermodell

- ▶ Additive White Gaussian Noise (AWGN)

$$x_i = y_i + Z_i \sim \mathcal{N}(0, \sigma^2)$$



Grundlagen - Fehlermodell

- ▶ Additive White Gaussian Noise (AWGN)

$$x_i = y_i + Z_i \sim \mathcal{N}(0, \sigma^2)$$

- ▶ Signal-Rausch-Verhältnis

$$SNR = 10 \log_{10} \left(\frac{\text{Signalleistung}}{\text{Rauschleistung}} \right) \text{ dB}$$



Grundlagen - Fehlermodell

- ▶ Additive White Gaussian Noise (AWGN)

$$x_i = y_i + Z_i \sim \mathcal{N}(0, \sigma^2)$$

- ▶ Signal-Rausch-Verhältnis

$$SNR = 10 \log_{10} \left(\frac{\text{Signalleistung}}{\text{Rauschleistung}} \right) \text{ dB}$$

- ▶ Shannon Grenze

$$C_{max} \leq B \log_2 \left(1 + \frac{S}{R} \right)$$



Blockcodes - Allgemeines

1. Nachricht in Blöcke aufteilen



Blockcodes - Allgemeines

1. Nachricht in Blöcke aufteilen
2. Jeden Block kodieren / dekodieren



Blockcodes - Allgemeines

1. Nachricht in Blöcke aufteilen
2. Jeden Block kodieren / dekodieren
3. Blöcke zur kodierten / dekodierten Nachricht zusammensetzen



Blockcodes - Buchstabieralphabet

<i>Daten</i>	<i>Kodierung</i>
A	Anton
B	Berta
C	Cäsar
.	.
.	.
.	.
Z	Zeppelin



Hamming-Kodes

- ▶ Hamming-(n,k)-Kode mit $n = 2^m - 1$ und $k = 2^m - m - 1$

Quelle: [1]



Hamming-Kodes

- ▶ Hamming-(n,k)-Kode mit $n = 2^m - 1$ und $k = 2^m - m - 1$
- ▶ Generatormatrix $G = (I_k \mid A)$

Quelle: [1]



Hamming-Kodes

- ▶ Hamming-(n,k)-Kode mit $n = 2^m - 1$ und $k = 2^m - m - 1$
- ▶ Generatormatrix $G = (I_k \mid A)$
- ▶ Kontrollmatrix $H = (A^T \mid I_m)$

Quelle: [1]



Hamming-Kodes - (7,4)

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



Hamming-Kodes - (7,4)

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

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



Hamming-Kodes - (7,4)

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

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 & \mathbf{1} & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & \mathbf{1} & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & \mathbf{1} \end{pmatrix}$$

$$H' = \begin{pmatrix} 0 & 0 & 0 & \mathbf{1} & 1 & 1 & 1 \\ 0 & \mathbf{1} & 1 & 0 & 0 & 1 & 1 \\ \mathbf{1} & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$



Hamming-Kodes - (7,4)

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

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



Hamming - (7,4) - Kodierung

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



Hamming - (7,4) - Dekodierung

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



Hamming - (7,4) - Dekodierung

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

Kodewort: (1,1,1,0,1,0,1)

Fehlerindex: 2

Korrigiertes Kodewort: (1,0,1,0,1,0,1)



- ▶ **B**ose-**C**haudhuri-**H**ocquenghem



BCH-Kodes

- ▶ **B**ose-**C**haudhuri-**H**ocquenghem
- ▶ **(n,k,d)**-Kode



BCH-Kodes

- ▶ **B**ose-**C**haudhuri-**H**ocquenghem
- ▶ **(n,k,d)**-Kode
- ▶ Korrigiert auf n -Bit Kodewörter bis zu $t = \frac{d}{2} - 1$ Fehler



BCH - Kodierung

- ▶ Nachrichten werden als Polynome mit Koeffizienten aus $GF(2)$ interpretiert.



BCH - Kodierung

- ▶ Nachrichten werden als Polynome mit Koeffizienten aus $GF(2)$ interpretiert.
- ▶ Generatorpolynom $g(x)$ kodiert Datenwort $dw(x)$:

$$kw(x) = dw(x)x^{n-k} - (dw(x)x^{n-k} \bmod g(x))$$



BCH - (15,5,7) - Kodierung

Input: (1,0,1,0,1)

Daten Polynom:

$$dw(x) = x^4 + x^2 + 1$$



BCH - (15,5,7) - Kodierung

Input: (1,0,1,0,1)

Daten Polynom:

$$dw(x) = x^4 + x^2 + 1$$

Generator Polynom:

$$g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x^1 + 1$$

$$dw^*(x) = dw(x) * x^{10} = x^{14} + x^{12} + x^{10}$$



BCH - (15,5,7) - Kodierung

Input: (1,0,1,0,1)

Daten Polynom:

$$dw(x) = x^4 + x^2 + 1$$

Generator Polynom:

$$g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x^1 + 1$$

$$dw^*(x) = dw(x) * x^{10} = x^{14} + x^{12} + x^{10}$$

Rest Polynom:

$$r(x) = dw^*(x) \mod g(x) = x^9 + x^6 + x^2 + x^1 + 1$$



BCH - (15,5,7) - Kodierung

Input: (1,0,1,0,1)

Daten Polynom:

$$dw(x) = x^4 + x^2 + 1$$

Generator Polynom:

$$g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x^1 + 1$$

$$dw^*(x) = dw(x) * x^{10} = x^{14} + x^{12} + x^{10}$$

Rest Polynom:

$$r(x) = dw^*(x) \bmod g(x) = x^9 + x^6 + x^2 + x^1 + 1$$

Kode Polynom:

$$kw(x) = c(r(x)|dw(x)) = x^{14} + x^{12} + x^{10} + x^9 + x^6 + x^2 + x^1 + 1$$

Output: (1,1,1,0,0,0,1,0,0,1,1,0,1,0,1)



BCH - Generatorpolynom

Körpererweiterung $GF(2^m)$ konstruieren. Beispiel für $m = 3$:



BCH - Generatorpolynom

Körpererweiterung $GF(2^m)$ konstruieren. Beispiel für $m = 3$:

α^i	Polynom	Vektor	Minimalpolynom
-	0	000	-
α^0	1	100	-
α^1	x	010	$x^3 + x + 1$
α^2	x^2	001	$x^3 + x + 1$
α^3	$x + 1$	110	$x^3 + x^2 + 1$
α^4	$x^2 + x$	011	$x^3 + x + 1$
α^5	$x^2 + x + 1$	111	$x^3 + x^2 + 1$
α^6	$x^2 + 1$	101	$x^3 + x^2 + 1$

BCH - Generatorpolynom

- ▶ Generatorpolynom für t -Fehler korrigierenden Kode hat Nullstellen

$$\alpha^1, \dots, \alpha^{2t}$$



BCH - Generatorpolynom

- ▶ Generatorpolynom für t -Fehler korrigierenden Kode hat Nullstellen

$$\alpha^1, \dots, \alpha^{2t}$$

- ▶ Für Minimalpolynome ϕ_i :

$$g(x) = LCM(\phi_1(x), \phi_2(x), \dots, \phi_{2t}(x))$$



BCH - Dekodierung

1. Berechnen der Syndrome $S_i = kw(\alpha^i) \mid 1 \leq i \leq 2t$.

Quelle: [2]



BCH - Dekodierung

1. Berechnen der Syndrome $S_i = kw(\alpha^i) \mid 1 \leq i \leq 2t$.
2. Bestimmen des Fehlerstellenpolynoms $\sigma(x)$.
(Berlekamp-Massey-Algorithmus)

Quelle: [2]



BCH - Dekodierung

1. Berechnen der Syndrome $S_i = kw(\alpha^i) \mid 1 \leq i \leq 2t$.
2. Bestimmen des Fehlerstellenpolynoms $\sigma(x)$.
(Berlekamp-Massey-Algorithmus)
3. Ermitteln der Inversen der Nullstellen von $\sigma(x)$, diese entsprechen den Fehlerindizes α^i . (Chien's Suche)

Quelle: [2]



BCH - Dekodierung

1. Berechnen der Syndrome $S_i = kw(\alpha^i) \mid 1 \leq i \leq 2t$.
2. Bestimmen des Fehlerstellenpolynoms $\sigma(x)$.
(Berlekamp-Massey-Algorithmus)
3. Ermitteln der Inversen der Nullstellen von $\sigma(x)$, diese entsprechen den Fehlerindizes α^i . (Chien's Suche)
4. Korrigieren der Fehler im Kodewort an den Fehlerindizes.

Quelle: [2]



Programmiersprache R

- Ursprünglich für statistische Zwecke entwickelt.

Quelle: [3]



Programmiersprache R

- ▶ Ursprünglich für statistische Zwecke entwickelt.
- ▶ Über 8000 Pakete auf den CRAN-Servern.

Quelle: [3]



Programmiersprache R

- ▶ Ursprünglich für statistische Zwecke entwickelt.
- ▶ Über 8000 Pakete auf den CRAN-Servern.
- ▶ Bereits in Lehrveranstaltungen verwendet.

Quelle: [3]



Entwicklungsumgebung RStudio

The screenshot displays the RStudio interface with the following components:

- Source Editor:** Shows the `BlockGenerateEncoderBCH` function in `BlockEncodeBCH.Rmd`. The function takes `code.length` and `code.t` as arguments and returns a list of encoder parameters.
- Environment:** Lists the current environment variables, including `x`, `coder`, `decoded`, `encoded`, `message`, and `vec`.
- Files:** Shows the file structure of the `Channelcoding2` package, including `.gitignore`, `.Rbuildignore`, `.RData`, `.Rhistory`, `channelcoding.Rproj`, `DESCRIPTION`, `doc`, `inst`, `man`, `NAMESPACE`, `R`, `README.md`, and `src`.
- Console:** Shows the execution of the `BlockGenerateEncoderBCH` function and the resulting `message` vector.

```
# Source Editor: BlockEncodeBCH.Rmd
86 - BlockGenerateEncoderBCH = function(code.length = 15, code.t = 3){
87
88   if(2*code.t > code.length)
89     stop("Invalid Parameters for BCH, 2*t must be < code.length")
90
91   n = ceiling(log2(code.length))
92   ret = c_getGeneratorPoly(code.length,
93                             n,
94                             code.t)
95
96   block.encoder = list(type = 'BCH',
97                        code.length = code.length,
98                        code.n = n,
99                        data.length = ret$data.length,
100                        code.t = code.t,
101                        gen.poly = ret$gen.poly,
102                        alpha.to = ret$alpha.to,
103                        index.of = ret$index.of)
104
105   1:1 (Top Level)

# Console
> message = c(1,0,1,0,1)
Restarting R session...
> library(channelcoding)
> message = c(1,0,1,0,1)
> coder = BlockGenerateEncoderHamming(7,4)
> encoded = BlockEncode(message,coder)
> encoded
[1] 1 0 1 0 1 0 1 1 1 0 0 0 0 1 1
> decoded = BlockDecode(encoded,coder)
> decoded
[1] 1 0 1 0 1 0 0 0
>
```


- ▶ [1] Cary W. Huffman und Vera Pless. *Fundamentals of error-correcting codes*. Cambridge university press, 2010.
- ▶ [2] Robert H. Morelos-Zaragoza. *The art of error correcting coding*. John Wiley and Sons, 2006.
- ▶ [3] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2016.

