4. BCH-Codes

Codierungstheorie
Anton Malevich
22.02.2018

Definitionen

BCH-Codes

BCH-Codes sind Verallgemeinerungen von Hamming-Codes, die mehr als einen Fehler korrigieren.
Der Hamming-Code $C = \mathrm{Ham}(k)$ hat Länge $2^k - 1$, Dimension $\dim C = 2^k - 1 - k$ und korrigiert einen Fehler. Die Idee ist, einen $\left[2^k - 1,\ 2^k - 1 - 2\,k\right]$-Code zu konstruieren, indem weitere k Zeilen zur Matrix H hinzugefügt werden, um 2 Fehler zu korrigieren, usw.

Die Konstruktion erfolgt leicht, wenn man die Kontrollmatrizen der Codes als Matrizen über dem Körper $\mathbb{F}$ mit 2^k Elementen (siehe Aufgabe 1) auffasst.

So ist $H = \begin{pmatrix} 1 & \alpha & \alpha^2 & \dots & \alpha^{2^k-2} \end{pmatrix}$ die Kontrollmatrix des Hamming-Codes $\mathrm{Ham}(k)$, $H' = \begin{pmatrix} 1 & \alpha & \alpha^2 & \dots \\ 1 & \alpha^3 & \alpha^6 & \dots \end{pmatrix}$ die Kontrollmatrix des binären 2-fehlerkorrigierenden BCH-Codes, $H'' = \begin{pmatrix} 1 & \alpha & \alpha^2 & \dots \\ 1 & \alpha^3 & \alpha^6 & \dots \\ 1 & \alpha^5 & \alpha^{10} & \dots \end{pmatrix}$ die Kontrollmatrix des binären 3-fehlerkorrigierenden BCH-Codes usw.

Decodieren der BCH-Codes

Es sei C ein binärer t-fehlerkorrigierender BCH-Code mit Kontrollmatrix H. Es sei ferner $c \in C$ versendet und $y = c + f$ mit $\mathrm{wt}(f) \leqslant t$ empfangen.

Die folgenden berechnungen sind im großen Körper $\mathbb{F}$ durchzuführen!

- Berechne das Syndrom $H\,y^{\intercal} = (\,s_1 \quad s_3 \quad s_5 \quad \dots\,)^{\intercal}$.
  Berechne s_2, s_4 usw. nach der Formel $s_{2\,i} = s_i^2$.
  Setze $r_0(x) = s_1 + s_2\,x + s_3\,x^2 + \dots + s_{2\,t}\,x^{2\,t-1}$.
- Dividiere $x^{2\,t}$ durch $r_0(x)$ mit Rest, dann $r_0(x)$ durch den Rest $r_1(x)$ usw.

$$x^{2t} = q_1(x)\,r_0(x) + r_1(x)$$
$$r_0(x) = q_2(x)\,r_1(x) + r_2(x)$$
$$r_1(x) = q_3(x)\,r_2(x) + r_3(x)$$
$$\vdots$$

- Bestimme den Index $k \geqslant 0$ mit $\mathrm{Grad}\ r_k(x) < t$ und $\mathrm{Grad}\ r_{k-1}(x) \geqslant t$.
- Nach dem erweiterten Euklidischen Algorithmus gilt $r_k(x) = a_k(x)\,x^{2\,t} + b_k(x)\,r_0(x)$.
  Finde $b_k(x)$ rekursiv als $b_k(x) = b_{k-2}(x) + q_k(x)\,b_{k-1}(x)$ mit Anfangswerten $b_{-1}(x) = 0$, $b_0(x) = 1$.
- Die Nullstellen von $b_k(x)$ entsprechen nun der Fehlerpositionen:
  Ist $b_k\left(\alpha^{-i}\right) = 0$, so ist $f_{i+1} = 1$.

Aufgaben

Aufgabe 1

Es sei $m(x)$ ein irreduzibles Polynom. Dann ist die Menge $\mathbb{F} = {}^{K[x]}/_{\langle m(x)\rangle} = \{v(x) \in K[x] \mid \text{Grad } v(x) < \text{Grad } f(x)\}$ ein Körper. Die Elemente des Körpers $\mathbb{F}$ sind Polynome vom Grad $<$ Grad $m(x)$ und Addition und Multiplikation in $\mathbb{F}$ werden modulo $m(x)$ durchgeführt.

```
m[x_] := x^4 + x + 1;
```

Man kann die Elemente von $\mathbb{F}$ auch als Vektoren der Länge Grad $m(x)$ auffassen. Die Addition ist die übliche Vektoraddition und die Multiplikation kann leicht anhand einer Tabelle durchgeführt werden. Dabei müssen die Vektoren nur richtig sortiert und als Potenzen eines Elementen (genannt *primitiv*) geschrieben werden.

```
AlphaPowers = {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0},
    {0, 0, 0, 1}, {1, 1, 0, 0}, {0, 1, 1, 0}, {0, 0, 1, 1}, {1, 1, 0, 1}, {1, 0, 1, 0},
    {0, 1, 0, 1}, {1, 1, 1, 0}, {0, 1, 1, 1}, {1, 1, 1, 1}, {1, 0, 1, 1}, {1, 0, 0, 1}};
MapIndexed[{α^#2[[1]]-1, "↔", #1, "↔", Expand[FromDigits[Reverse@#1, x]]} &, AlphaPowers];
Grid[%, Dividers → {False, All}, Alignment → {Left, Baseline}]
```

1	↔	{1, 0, 0, 0}	↔	1
α	↔	{0, 1, 0, 0}	↔	x
α^2	↔	{0, 0, 1, 0}	↔	x^2
α^3	↔	{0, 0, 0, 1}	↔	x^3
α^4	↔	{1, 1, 0, 0}	↔	$1 + x$
α^5	↔	{0, 1, 1, 0}	↔	$x + x^2$
α^6	↔	{0, 0, 1, 1}	↔	$x^2 + x^3$
α^7	↔	{1, 1, 0, 1}	↔	$1 + x + x^3$
α^8	↔	{1, 0, 1, 0}	↔	$1 + x^2$
α^9	↔	{0, 1, 0, 1}	↔	$x + x^3$
α^{10}	↔	{1, 1, 1, 0}	↔	$1 + x + x^2$
α^{11}	↔	{0, 1, 1, 1}	↔	$x + x^2 + x^3$
α^{12}	↔	{1, 1, 1, 1}	↔	$1 + x + x^2 + x^3$
α^{13}	↔	{1, 0, 1, 1}	↔	$1 + x^2 + x^3$
α^{14}	↔	{1, 0, 0, 1}	↔	$1 + x^3$

Die Regel $\alpha^4 = 1 + \alpha$ entspricht dabei der Polynomialgleichung $m(x) \equiv 0 \bmod m(x)$. Die anderen Potenzen findet man rekursiv: $\alpha^i \equiv \alpha\, \alpha^{i-1} \bmod m(x)$.

Mathematica kann auch anhand dieser Tabelle rechen.

```
<< FiniteFields`
𝔽 = PowerListToField[AlphaPowers];
```

Hier sind Beispiele einiger wichtigen Rechenoperationen.

```
𝔽[{0, 0, 1, 1}] + 𝔽[{1, 0, 1, 0}]
𝔽[{0, 0, 1, 1}] * 𝔽[{1, 0, 1, 0}]
```

$\{1, 0, 0, 1\}_2$

$\{1, 0, 0, 1\}_2$

```
FieldInd[%]
FieldExp[𝔽, 14]
```

14

$\{1, 0, 0, 1\}_2$

Überprüfen Sie per Hand, dass $\alpha^6 + \alpha^8 = \alpha^{14} = \alpha^6 * \alpha^8$ in $\mathbb{F}$ ist.

Die Kontrollmatrix des Hamming-Codes C = Ham(k) kann wie folgt aufgefasst werden.

```
Ham = ( 1  α  α^2  α^3  α^4  α^5  α^6  α^7  α^8  α^9  α^10  α^11  α^12  α^13  α^14 );
Ham /. (MapIndexed[α^(#2[[1]]-1) → #1 &, AlphaPowers]);
MatrixForm[%[[1]]^T]
```

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

Der binäre 2-fehlerkorrigierende BCH-Code C ist der Code mit der Kontrollmatrix

```
H16 = ( 1  α  α^2  α^3  α^4  α^5  α^6  α^7  α^8  α^9  α^10  α^11  α^12  α^13  α^14
        1  α^3  α^6  α^9  α^12  1  α^3  α^6  α^9  α^12  1  α^3  α^6  α^9  α^12 );
H16 /. (MapIndexed[α^(#2[[1]]-1) → #1 &, AlphaPowers]);
H2 = Join[%[[1]]^T, %[[2]]^T];
MatrixForm@%
```

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

Berechnen Sie die Minimaldistanz von C.

Die Kontrollmatrix über $\mathbb{F}$ (in der Form, die Mathematica versteht) ist

```
H = (H16 /. {1 → 𝔽[{1, 0, 0, 0}], α → 𝔽[{0, 1, 0, 0}], α^i_ :> FieldExp[𝔽, i]});
```

Es sei der Codewort c versendet, aber der Vektor y empfangen worden:

```
c = {0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0};
y = c + {0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0}
```

```
{0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0}
```

Zunächst berechnen wir das Syndrom und konstruieren das Polynom $r_0(x)$

```
{S1, S3} = H.y; S2 = S1^2; S4 = S2^2;
S = {S1, S2, S3, S4};
r[0, x_] = S.x^Range[0,3]
```

```
x^2 {0, 1, 0, 0}_2 + x^3 {0, 1, 1, 1}_2 + {1, 0, 0, 1}_2 + x {1, 0, 1, 1}_2
```

Nun dividieren wir x^4 (Dabei ist es wichtig für Mathematica klar zu machen, dass x^4 über dem Körper $\mathbb{F}$ zu betrachten ist!) durch das Syndrompolynom $r_0(x)$ mit Rest, dann $r_0(x)$ durch $r_1(x)$ usw. bis wir den Index k mit Grad $r_k(x) < t$ und Grad $r_{k-1}(x) \geqslant t$ gefunden haben.

```
{q[1, x_], r[1, x_]} = PolynomialQuotientRemainder[𝔽[{1, 0, 0, 0}] x^4, r[0, x], x]
Exponent[r[1, x], x] < 2
```

```
{{0, 1, 0, 1}_2 + x {1, 1, 0, 0}_2, {1, 0, 1, 0}_2 + x {1, 1, 0, 0}_2 + x^2 {1, 1, 0, 0}_2}
```

```
False
```

```
{q[2, x_], r[2, x_]} = PolynomialQuotientRemainder[r[0, x], r[1, x], x]
Exponent[r[2, x], x] < 2
```

$\{\{0, 0, 1, 0\}_2 + x \{1, 1, 0, 1\}_2, \{0, 1, 1, 1\}_2\}$

True

Nun müssen wir $b_2(x)$ finden. $b_k(x)$ ist rekursiv definiert

```
ClearAll[b];
b[-1, x_] = 0;
b[0, x_] = 𝔽[{1, 0, 0, 0}];
b[k_?Positive, x_] := Expand[b[k - 2, x] + q[k, x] b[k - 1, x]]
```

```
b[2, x]
```

$x \{0, 1, 1, 1\}_2 + x^2 \{0, 1, 1, 1\}_2 + \{1, 1, 1, 1\}_2$

Nun suchen wir die Nullstellen von $b_2(x)$. Das sind α^7 und α^9.

```
zerosPositions = Position[Table[b[2, FieldExp[𝔽, i]], {i, 0, 14}], 0, {1}]
```

{{8}, {10}}

Die Fehlerpositionen finden wir durch Kehrwerte der Nullstellen. Somit haben wir den Fehlervektor bestimmt.

```
FieldInd[FieldExp[𝔽, 1 - #[[1]]]] & /@ zerosPositions;
pos = Sort[% + 1]
f = ReplacePart[Array[0 &, 15], Thread[pos → 1]]
```

{7, 9}

{0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0}

Das gesendete Codewort wird richtig decodiert:

```
Mod[y - f, 2] == c
```

True

Benutzen Sie den BCH-Decodierer um die Worten {1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0} und {1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0} zu decodieren.

Programmieren Sie den BCH-Decodierer für den Code C.

Aufgabe 2

Wir betrachten den binären [7, 4, 3] Hamming-Code $C = \mathrm{Ham}_2(3)$ als 1-fehlerkorrigierenden BCH-Code. Die Kontrollmatrix ist gegeben durch $H = (\,1 \;\; \alpha \;\; \alpha^2 \;\; \alpha^3 \;\; \alpha^4 \;\; \alpha^5 \;\; \alpha^6\,)$ über dem Körper $\mathbb{F}_8$ mit 8 Elementen (mit der Regel $\alpha^3 = \alpha + 1$).

```
𝔽8 = GF[2, {1, 1, 0, 1}];
MapIndexed[{α^#2[[1]]-1, "↔", #1, "↔", Expand[FromDigits[Reverse@#1, x]]} &, PowerList[𝔽8]];
Grid[%, Dividers → {False, All}]
```

1	↔	{1, 0, 0}	↔	1
α	↔	{0, 1, 0}	↔	x
α^2	↔	{0, 0, 1}	↔	x^2
α^3	↔	{1, 1, 0}	↔	$1 + x$
α^4	↔	{0, 1, 1}	↔	$x + x^2$
α^5	↔	{1, 1, 1}	↔	$1 + x + x^2$
α^6	↔	{1, 0, 1}	↔	$1 + x^2$

Benutzen Sie den BCH-Decodierer um die Wörter (0 101 010) und (1 010 111) zu decodieren.

Implementieren Sie den BCH-Decodierer für den Code C.

Aufgabe 3*

Konstruieren Sie den binären 2-fehlerkorrigierenden BCH-Code C mit Länge $n = 31$. Dabei soll die Kontrollmatrix H über $\mathbb{F}_{32}$ (mit der Regel $\alpha^5 = \alpha^2 + 1$ gegeben sein.

```
𝔽32 = GF[2, {1, 0, 1, 0, 0, 1}];
```

Was ist die Dimension des Codes?

Implementieren Sie den BCH-Decodierer für den Code C.