DEKODIERUNG MIT GRÖBNERBASEN

Vortrag im Rahmen des Seminars der WE AlZAGK im
Sommersemester 2008 an der Universität Bremen

Fabian Dreher

22.05.2008

Vorbemerkung

Basierend auf Kapitel 11 (Gröbner Bases für Decoding) und Projekt 7 (The Golay Codes) des Buches *Some Tapas of Computer Algebra* wird im Folgenden erläutert, wie das Dekodieren zyklischer Codes mit Hilfe der Nutzung von Algorithmen zur Bestimmung von Gröbnerbasen möglich ist.

1 Zyklische Codes

Wir beginnen mit einer kurzen Erinnerung an einige grundlegende Eigenschaften zyklischer Codes.

1.1 Definition *Ein linearer Code $C \subset \mathbb{F}_q^n$ heißt **zyklischer Code**, wenn aus $(c_0 \dots, c_{n-1}) \in C$ folgt, dass auch $(c_{n-1}, c_1, c_2 \dots, c_{n-2}) \in C$.*

Wir wollen im Folgenden annehmen, dass n und q teilerfremd sind.

Die Abbildung $\varphi: \mathbb{F}_q^n \to \mathbb{F}_q[X]/\langle X^n - 1\rangle$, $c \mapsto c_0 + c_1 X + \cdots + c_{n-1}X^{n-1}$ ist offenbar eine Bijektion. Da die Multiplikation von $\varphi\big((c_0 \dots, c_{n-1})\big)$ mit X in $\mathbb{F}_q[X]/\langle X^n - 1\rangle$ der zyklischen Vertauschung von $(c_0 \dots, c_{n-1})$ zu $(c_{n-1}, c_1, c_2 \dots, c_{n-2})$ entspricht, gibt φ eine bijektive Zuordnung zwischen Idealen in $\mathbb{F}_q[X]/\langle X^n - 1\rangle$ und zyklischen Codes in $\mathbb{F}_q^n$.

Da Ideale in $\mathbb{F}_q[X]/\langle X^n - 1\rangle$ jeweils von einem Element erzeugt werden, besitzt jeder zyklische Code ein eindeutiges normiertes Erzeugerpolynom $g(X)$ von kleinstem Grad. Insbesondere teilt dieses Erzeugerpolynom $X^n - 1$, da Ideale in $\mathbb{F}_q[X]/\langle X^n - 1\rangle$ Idealen in $\mathbb{F}_q[X]$ entsprechen, die $\langle X^n - 1 \rangle$ enthalten. Gleichwertig zur Beschreibung durch das Erzeugerpolynom kann ein zyklischer Code auch durch die Nullstellen des Erzeugerpolynoms in einem Erweiterungskörper von $\mathbb{F}_q$ beschrieben werden.

1.2 Definition *Sei α eine primitive n-te Einheitswurzel in einem Erweiterungskörper $\mathbb{F}_{q^e}$. Wir nennen eine Menge $J \subset \mathbb{Z}_n$ **definierende Menge** des zyklischen Codes C, wenn $C = \{c(X) \in \mathbb{F}_q[X]/\langle X^n - 1 \rangle \,|\, c(\alpha^j) = 0 \; \forall j \in J\}$.*

2 Dekodieren allgemein

Sei $C \subset \mathbb{F}_q^n$ ein linearer Code. Die allgemeine Situation, die uns zum Problem des Dekodierens führt, ist die Folgende:

Der Sender schickt ein Codewort $c \in C$ über einen Transportkanal zu einem Empfänger, der ein Wort $y \in \mathbb{F}_q^n$ empfängt. Da der Transportkanal gestört sein kann, ist das empfangene Wort von der Form $y = c + e$, wobei $e = y - c$ der aufgetretene Fehler ist. Die Aufgabe des Dekodierers ist es nun, das empfangene Wort zu verarbeiten.

Ein Dekodierer ist eine Abbildung $\mathcal{D}: \mathbb{F}_q^n \to C \cup \{?\}$ mit der Eigenschaft $\mathcal{D}(c) = c$ für alle $c \in C$. Der Dekodierer gibt also entweder ein Codewort aus oder ?, falls ein Dekodierversagen aufgetreten ist. Als einfachster Dekodierer ist ein solcher denkbar, der für ein empfangenes Wort y entweder y selbst zurückgibt, falls $y \in C$, oder andernfalls ein Dekodierversagen meldet.

Um den für einen Dekodierer notwendigen Test, ob ein Wort ein Codewort ist, effektiv durchführen zu können, ist es sinnvoll eine Kontrollmatrix zu verwenden.

2.1 Definition *Eine Matrix H heißt **Kontrollmatrix** für den Code C, wenn gilt* $C = \{c \in \mathbb{F}_q^n | cH^t = 0\}$.

Damit lässt sich die Fehlererkennung mittels Multiplikation des empfangen Wortes y mit H^t durchführen. Wir wollen statt bloßer Fehlererkennung jedoch auch eine Fehlerkorrektur durchführen können, sofern kein „zu großer" Fehler aufgetreten ist.

Aus der Kontrollmatrix H und dem empfangenen Wort $y = c + e$ erhalten wir das **Syndrom** $s = yH^t$. Wegen $s = yH^t = (c + e)H^t = eH^t$ enthält das Syndrom offenbar eine Information über den aufgetretenen Fehler. Der nachfolgende Satz besagt, dass bei hinreichend kleiner Fehlerzahl der Fehlervektor e aus dem Syndrom durch das Lösen linearer Gleichungen berechnet werden kann.

2.2 Satz *Sei C ein linearer Code in $\mathbb{F}_q^n$ mit Kontrollmatrix H und Minimalabstand $d(C)$. Sei y ein empfangenes Wort mit Fehlervektor e und sei J eine Menge mit höchstens $d(C) - 1$ Elementen, die die Menge der Fehlerpositionen enthält. Dann ist der Fehlervektor e die eindeutige Lösung für x der linearen Gleichungen $xH^t = yH^t = s$ und $x_j = 0$ für $j \notin J$.*

Beweis. Zuerst zeigen wir, dass e eine Lösung ist. Offensichtlich erfüllt e die Bedingungen aus $x_j = 0$ für $j \notin J$. Sei $c \in C$, so dass $y = c + e$, damit erhalten wir $yH^t = eH^t$. Also ist e eine Lösung der Gleichungen.
Sei nun e' ein weiterer Vektor, der die Gleichungen erfüllt. Dann gilt $eH^t = yH^t = e'H^t$, also $(e - e')H^t = 0$ und somit folgt $e - e' \in C$. Aus den Gleichungen $e_j = 0 = e_j'$ für $j \notin J$ ergibt sich $d(e - e', 0) \leq d(C) - 1 < d(C)$, also $e = e'$. ■

2.3 Folgerung *Sei $s = yH^t$ das Syndrom eines empfangenen Wortes $y = c + e$ mit höchstens $\lfloor (d(C) - 1)/2 \rfloor$ Fehlern. Dann ist e die einzige Lösung von $s = xH^t$ für $d(x, 0) \leq \lfloor (d(C) - 1)/2 \rfloor$.*

Beweis. Nehmen wir an, es gäbe zwei Fehlervektoren e, e' die beide das gleiche Syndrom $s = yH^t = eH^t = e'H^t$ liefern. Da $d(e - e', 0) \leq d(C) - 1$ folgt nach Satz 2.2, dass $e - e'$ die eindeutige Lösung der Gleichungen $0 = xH^t$ und $x_j = 0$ für $j \notin \{i | e_i \neq e_i'\}$ ist. Da 0 eine Lösung ist, muss gelten $e - e' = 0$, und somit ist das Syndrom eindeutig. ■

Wir nennen $t = \lfloor (d(C) - 1)/2 \rfloor$ auch die Fehlerkorrekturkapazität des Codes C.

Aus Satz 2.2 ergibt sich, dass das Problem der Fehlerkorrektur auf das Auffinden der Fehlerstellen zurückgeführt werden kann, da man aus den Fehlerstellen durch Lösen linearer Gleichungen den Fehlervektor gewinnen kann. Das Ermitteln der Fehlerstellen mittels Gröbnerbasen ist Inhalt des nächsten Abschnitts.

3 Dekodieren mit Gröbnerbasen

Wir betrachten im Folgenden einen zyklischen Code C mit Parametern $[n,k,d]_q$, Erzeugerpolynom $g(X)$ und definierender Menge $J=\{j_1,\dots,j_r\}$. n und q seien teilerfremd.

Sei $\mathbb{F}_{q^e}$ eine Erweiterung von $\mathbb{F}_q$, die eine primitive n-te Einheitswurzel $\alpha \in \mathbb{F}_{q^e}$ enthält. Dann erhalten wir eine Kontrollmatrix für C durch

$$H=\begin{pmatrix} 1 & \alpha^{j_1} & \alpha^{2j_1} & \cdots & \alpha^{(n-1)j_1} \\ 1 & \alpha^{j_2} & \alpha^{2j_2} & \cdots & \alpha^{(n-1)j_2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{j_r} & \alpha^{2j_r} & \cdots & \alpha^{(n-1)j_r} \end{pmatrix}$$

Die Multiplikation eines Wortes mit der Kontrollmatrix entspricht dem Einsetzen der α^{j_i} in das dem Wort zugeordnete Polynom in $\mathbb{F}_q[X]/\langle X^n-1\rangle$. Nach der Definition 1.2 einer definierenden Menge ist ersichtlich, dass H tatsächlich ein Kontrollmatrix ist.

Aus Gründen einer klareren Schreibweise der Indizes ist es sinnvoll, an Stelle von H die erweiterte Matrix $\widehat{H}$ zu betrachten, die eine $n\times n$-Matrix mit i-ter Zeile $(1 \quad \alpha^i \quad \alpha^{2i} \quad \cdots \quad \alpha^{(n-1)i})$ ist.

Sei $y=y(X)$ ein empfangenes Wort mit Fehlervektor $e=e(X)$. Diese Schreibweise soll verdeutlichen, dass wir Vektor- und Polynomdarstellung gleichberechtigt und mit denselben Symbolen benutzen. Analog zum bisherigen Syndrom wählen wir die Bezeichnung $\hat{s}=e\widehat{H}^t$. Die j-te Komponente von $\hat{s}$ ist

$$\hat{s}_j = e\left(\alpha^j\right)=\sum_{i=0}^{n-1} e_i\cdot\alpha^{ij}$$

Für $j\in J$ gilt gemäß den normalen Syndromeigenschaften $\hat{s}_j = e\left(\alpha^j\right)=y\left(\alpha^j\right)$, diese Werte sind uns also bekannt.

Wir bezeichnen im Folgenden $\hat{s}_j$ einfach mit s_j, so dass die alten s_i dann mit s_{j_i} bezeichnet werden.

Sei $e=e(X)$ ein Fehlervektor mit den Fehlerpositionen $i_1,\dots,i_t$ und Fehlerwerten $e_{i_1},\dots,e_{i_t}$. Die bekannten Syndrome sind dann

$$s_j=\sum_{m=1}^{t} e_{i_m}\left(\alpha^{i_m}\right)^j,\ j\in J$$

Um Gröbnerbasen zur Ermittlung von Fehlerstellen und Fehlerwerten einsetzen zu können, müssen wir uns ein geeignetes System von Polynomgleichungen über $\mathbb{F}_{q^e}$ beschaffen. Betrachten wir dazu

$$\mathcal{S}(s,v)=\begin{cases} \sum_{m=1}^{v} Y_m X_m^j = s_j\ , & j\in J \\ Y_m^q = Y_m\ , & m=1,\dots,v \\ X_m^n = 1\ , & m=1,\dots,v \end{cases}$$

Eine Lösung des Gleichungssystems $\mathcal{S}(s,t)$, also für die „richtige“ Fehlerzahl, ist durch $X_m=\alpha^{i_m}$ und $Y_m=e_{i_m}$ für $m=1,\dots,t$ gegeben. Allgemein lässt sich folgender Satz formulieren:

3.1 Satz *Es seien t Fehler aufgetreten und $t \leq (d(C)-1)/2$. Dann hat das System $\mathcal{S}(s,v)$ über $\mathbb{F}_{q^e}$ keine Lösung, wenn $v < t$, und eine, bis auf Permutation eindeutige, Lösung, wenn $v = t$, die dem Fehlervektor e von geringstem Gewicht, der die Syndromgleichungen erfüllt, entspricht. Die X_i der Lösung sind die Fehlerlokalisatoren und die Y_i die entsprechenden Fehlerwerte.*

Beweis. Sei $x_1, \dots, x_v, y_1, \dots, y_v$, $1 \leq v \leq t$ eine Lösung von $\mathcal{S}(s,v)$. Die x_m sind n-te Einheitswurzeln und haben daher eine eindeutige Darstellung als $x_m = \alpha^{k_m}$, $0 \leq k_m \leq n-1$ für $m = 1, \dots, v$.
Wir konstruieren uns einen Vektor $\tilde{e} \in \mathbb{F}_q^n$ durch die Setzung $\tilde{e}_{k_m} := \sum_{i \in \{i \mid k_i = k_m\}} y_i$ für $m = 1, \dots, v$, und 0 für die restlichen Komponenten. Für diesen gilt $\tilde{e}(\alpha^j) = \sum_{i=0}^{n-1} \tilde{e}_i (\alpha^j)^i = \sum_{m=1}^{v} y_m \alpha^{k_m \cdot j} = \sum_{m=1}^{v} y_m x_m^j = s_j$ für $j \in J$. Da $\tilde{e}$ also die Syndromgleichungen löst und $d(\tilde{e}, 0) \leq v \leq (d(C)-1)/2$, muss $\tilde{e}$ nach Folgerung 2.3 gleich dem Fehlervektor sein. Insbesondere folgt $v = t$, da t Fehler aufgetreten waren, und es ist klar, dass die Lösung eine Permutation der sich aus dem Fehlervektor e ergebenden Lösung ist. ■

Unser Ziel ist es nun, eine Möglichkeit zu finden, Lösungen für die X_i zu erhalten, da wir nach Satz 2.2 aus der Kenntnis der Fehlerstellen die Fehlerwerte rekonstruieren können. Das System $\mathcal{S}(s,v)$ definiert ein Ideal im Ring $\mathbb{F}_{q^e}[X_1, \dots, X_v, Y_1, \dots, Y_v]$, welches wir ebenfalls mit $\mathcal{S}(s,v)$ bezeichnen werden. Die Nullstellenmenge dieses Ideals liefert uns für $v = t$, wie in Satz 3.1 beschrieben, den Fehlervektor.

3.2 Satz *Es seien t Fehler aufgetreten und $t \leq (d(C)-1)/2$. Sei $g(X_1)$ der normierte Erzeuger des Ideals $\mathcal{S}(s,t) \cap \mathbb{F}_{q^e}[X_1]$. Dann sind die Nullstellen von g die Fehlerlokalisatoren.*

Beweis. Aus Satz 3.1 wissen wir, dass $\mathcal{S}(s,t)$ nur endlich viele Nullstellen besitzt. Daraus folgt, dass die Nullstellenmenge von $\mathcal{S}(s,t) \cap \mathbb{F}_{q^e}[X_1]$ gleich der Projektion der Nullstellenmenge von $\mathcal{S}(s,t)$ auf die X_1-Komponente ist. Ebenfalls aus Satz 3.1 wissen wir, dass die X_i der Lösungen von $\mathcal{S}(s,t)$ die Fehlerlokalisatoren sind. Da Permutationen einer Lösung ebenfalls Lösungen sind, tritt jeder Fehlerlokalisator auch in der X_1-Komponente einer der Lösungen auf. Somit ist die Nullstellenmenge von $\mathcal{S}(s,t) \cap \mathbb{F}_{q^e}[X_1]$ genau die Menge der Fehlerlokalisatoren. ■

Um das $g(X)$ aus Satz 3.2 berechnen zu können, lassen sich Gröbnerbasen einsetzen, wie folgender Satz zeigt:

3.3 Satz *Sei $I \subset K[Z_1, Z_2, \dots, Z_r]$ ein Ideal und sei G eine Gröbnerbasis von I bezüglich der lexikographischen Ordnung mit $Z_1 < Z_2 < \cdots < Z_r$. Dann ist $G \cap K[Z_1, \dots, Z_i]$ eine Gröbnerbasis von $I \cap K[Z_1, \dots, Z_i]$.*

Wir müssen jedoch noch bedenken, dass Satz 3.2 eine Aussage über das System $\mathcal{S}(s,t)$ trifft und daher vorerst nur nützt, wenn die korrekte Fehlerzahl t bekannt ist. Da die Zahl der Berechnungen zur Lösung des Systems $\mathcal{S}(a,v)$ umso höher ist, je größer v ist, wäre ein Algorithmus, der mit kleinem v startet und sich zu t „hocharbeitet“, hilfreich. Die Grundlage dafür liefert der folgende Satz.

3.4 Satz *Es seien t Fehler aufgetreten, $t \leq (d(C)-1)/2$. Das normierte Fehlerlokalisierungspolynom werde mit $l(X_1)$ bezeichnet, also $l(x)=0$ genau dann, wenn x ein Fehlerlokalisator ist. Sei $g(X_1)$ das normierte Erzeugerpolynom von $\mathcal{S}(s,v) \cap \mathbb{F}_{q^e}[X_1]$. Dann gilt*

$$g(X_1) = \begin{cases} 1 & wenn\ v < t \\ l(X_1) & wenn\ v = t \end{cases}$$

Beweis. Wir wissen aus Satz 3.1, dass $\mathcal{S}(s,v)$ keine Lösung für $v < t$ besitzt. Da die Nullstellenmenge von $\mathcal{S}(s,v) \cap \mathbb{F}_{q^e}[X_1]$ die Projektion der Nullstellenmenge von $\mathcal{S}(s,v)$auf die X_1-Komponente ist, muss auch sie leer sein. Einer leeren Nullstellenmenge entspricht ein von einem konstanten Polynom erzeugtes Ideal. Da $g(X_1)$ normiert ist, ist s für $v < t$ also 1. Für $v = t$ folgt aus Satz 3.2, dass $g(X_1) = l(X_1)$. ■

Dieser Satz ermöglicht es nun, einen Algorithmus zu formulieren, der die Fehlerstellen ermittelt und so zyklische Codes dekodieren kann.

```
Eingabe(y);
s ≔ yĤ^T;
Falls s_j = 0 für alle j ∈ J, dann {
    Ausgabe(y);
    Stop; // Es sind keine Fehler aufgetreten
}
Sonst {
    v ≔ 0;
    G ≔ {1};
    Solange 1 ∈ G, tue {
        v ≔ v + 1;
        S ≔ {∑_{m=1}^{v} Y_m X_m^j − s_j | j ∈ J} ∪ {Y_m^q − Y_m, X_m^n − 1 | m = 1, …, v};
        G ≔ Gröbner(S);
    }
    // 1 ∉ G, es gibt also Lösungen
    g(X_1) ≔ das einzige Element von G ∩ F_{q^e}[X_1];
    Falls deg(g(X_1)) > v, dann {
        Ausgabe(?);
        Stop; // Es sind zu viele Fehler aufgetreten
    }
    Sonst {
        Fehlerlokalisatoren ≔ {Nullstellen von g(X_1)};
        Finde den Fehlervektor e durch Lösen der linearen Gleichungen;
        Ausgabe(y − e);
    }
}
```

Einen die zu lösenden Gleichungen vereinfachenden Sonderfall gibt es, wenn $q = 2$ ist. Das Gleichungssystem

$$\mathcal{S}(s,v) = \begin{cases} \sum_{m=1}^{v} Y_m X_m^j = s_j\ , & j \in J \\ Y_m^q = Y_m\ , & m = 1, \dots, v \\ X_m^n = 1\ , & m = 1, \dots, v \end{cases}$$

vereinfacht sich zu

$$\mathcal{S}_2(s,v) = \begin{cases} \sum_{m=1}^{v} X_m^j = s_j \ , \ \ j \in J \\ X_m^n = 1 \ , \quad m = 1, \dots, v \end{cases}$$

Dies ist möglich, weil der einzige Fehlerwert, der auftreten kann, 1 ist, und daher die Gleichungen $Y_m^q = Y_m$ wegfallen und Y_m in den anderen Gleichungen durch 1 ersetzt werden kann. Im Fall $q = 2$ ist das Problem der Fehlerkorrektur also identisch mit dem Auffinden der Fehlerstellen.

Allgemein ist festzustellen, dass die Leistungsfähigkeit des vorgestellten Algorithmus vor allem von der Geschwindigkeit des Algorithmus zur Ermittlung von Gröbnerbasen abhängt. Dieser findet zwar nur in der einen Zeile „$\mathcal{G} := \mathrm{Gröbner}(\mathcal{S})$;“ Eingang in das Dekodierungsverfahren, dort findet jedoch die Hauptrechenarbeit statt.

4 Information zum Anhang

Im Anhang befindet sich das während des Vortrags ausgegebene Informationsblatt, das den Algorithmus in Pseudocode sowie eine Implementierung in Macaulay2 zeigt. Der vollständige Quelltext der Implementierung findet sich unter http://crypto.math.uni-bremen.de/Ausarbeitungen/SoSe08/GroebnerDekodieren.zip. Dieses konkrete Codebeispiel arbeitet mit dem binären Golaycode und benutzt die erwähnten Vereinfachungen des zu lösenden Gleichungssystems für $q = 2$.

5 Literaturverzeichnis

de Boer, M., & Pellikaan, R. (1999). Gröbner Bases for Decoding. In A. M. Cohen, H. Cuypers, & H. Sterk (Hrsg.), *Some Tapas of Computer Algebra* (S. 260-275). Berlin Heidelberg: Springer.

de Boer, M., & Pellikaan, R. (1999). The Golay Codes. In A. M. Cohen, H. Cuypers, & H. Sterk (Hrsg.), *Some Tapas of Computer Algebra* (S. 338-348). Berlin Heidelberg: Springer.

Dreher, F. (2008). *Gröbnerbasen und Eliminationstheorie.* Von http://crypto.math.uni-bremen.de/Ausarbeitungen/WiSe0708/Groebnerbasen.pdf abgerufen

Der fertige Algorithmus in Pseudocode

```
Eingabe(y);
s ≔ yĤ^T;
Falls s_j = 0 für alle j ∈ J, dann {
      Ausgabe(y);
      Stop; // Es sind keine Fehler aufgetreten
}
Sonst {
      v ≔ 0;
      G ≔ {1};
      Solange 1 ∈ G, tue {
            v ≔ v + 1;

      S ≔ {Σ_{m=1}^{v} Y_m X_m^j − s_j | j ∈ J} ∪ {Y_m^q − Y_m, X_m^n − 1 | m = 1, …, v};
            G ≔ Gröbner(S);
      }
      // 1 ∉ G, es gibt also Lösungen
      g(X_1) ≔ das einzige Element von G ∩ F_{q^e}[X_1];
      Falls deg(g(X_1)) > v, dann {
            Ausgabe(?);
            Stop; // Es sind zu viele Fehler aufgetreten
      }
      Sonst {
            Fehlerlokalisatoren ≔ {Nullstellen von g(X_1)};
       Finde den Fehlervektor e durch Lösen der linearen Gleichungen;
            Ausgabe(y − e);
      }
}
```

Implementierung in Macaulay2 für binäre Codes

```
print ("Empfangenes Wort: " | net(y));
H=map(R^n,R^n,(i,j)->a^(j*i)); --R ist der Erweiterungskörper
s=y*transpose(H);
if (s_J = 0) then (print "Korrektes Codewort empfangen") else (
    print "Falsches Wort empfangen, versuche Korrektur";
    v=0;
    G=matrix {{1}};
    while G_(0,0)==1 do (
        v=v+1;
        S={};
        for j from 0 to (r-1) do (
            f=-s_(0,J#j);
            for m from 1 to v do (f=f+X_m^(J#j););
           S=append(S,f);
        );
        for m from 1 to v do (
            f=X_m^(n)-1;
            S=append(S,f);
        );
        I=ideal(S);
        G=gens(gb(I));
    );
    g=G_(0,0);
    if ((degree g)_0 > v) then (print "Zu viele Fehler";) else (
        E={};
        for i from 0 to n-1 do (
             if (substitute(g,{X_1=>a^i})==0) then (E=append(E,i);)
             );
        print ("Fehlerstellen:    " | net(E));
        e=map(R^1,n,(i,j)->(if positions(E,x->x==j)!={} then 1 else 0));
        print ("Fehlervektor:     " | net(e));
        print ("Korrigiertes Wort:" | net(y-e));
    )
)
```