Dekodierung von Reed-Solomon Codes

Im diesem Text soll noch eine andere Darstellung der Dekodierung von Reed-Solomon Codes gegeben werden, welche mehr auf allgemeinen Konzepten als direktem Nachrechnen wie in den Skripten basiert.

Sei C ein $[n, k, d]$-Code über $\mathbb{F}_q$. Sei $y \in \mathbb{F}_q^n$ ein empfangenes Wort, welches zum einem bezüglich des Hammingabstands nächstgelegenen Codewort dekodiert werden soll. Sei $t = \lfloor (d-1)/2 \rfloor$. Dann kann prinzipiell C maximal t Fehler korrekt dekodieren. Wir nehmen daher an, daß es $x \in C$ mit $w(x - y) \leq t$ gibt. Mit dieser Bedingung ist x eindeutig durch y bestimmt. Die Dekodierungsaufgabe besteht nun darin, aus dem Vektor y das Codewort x auszurechnen.

Wir definieren $e = y - x$ als den Fehlervektor und $I_y = \{i \,|\, 1 \leq i \leq n \text{ und } e_i \neq 0\}$ die Menge der Fehlerpositionen. Sei $s : \mathbb{F}_q^n \to \mathbb{F}_q^{n-k}$ die Syndromabbildung, daß heißt s ist ein Epimorphismus mit $\ker(s) = C$. Es gilt $s(e) = s(y)$. Wir haben bereits gesehen, daß e leicht aus $s(y)$ und I_y mittels der aus $s(e) = s(y)$ resultierenden linearen Gleichungen für die Koeffizienten von e berechnet werden kann. Die Dekodierung ist damit in gewisser Weise ein vornehmlich kombinatorisches Problem.

Algebraisierung des Dekodierung

Der Ansatz für eine effiziente Dekodoierung liegt in der geeigneten Algebraisierung des unterliegenden kombinatorischen Problems. Wir nehmen zunächst nur an, daß C ein linearer Code wie oben mit $n \leq q - 1$ ist. Wir wählen $a, c \in \mathbb{F}_q^n$ mit $a_i \neq a_j$ für alle $i \neq j$ und $c_i \neq 0$ für alle i, und definieren das Fehlerlokalisierungspolynom

$$\sigma(X) = \prod_{i \in I_y} (1 - a_i X)$$

sowie das Fehlerpolynom

$$\omega(X) = \sum_{i \in I_y} e_i c_i \prod_{j \in I_y,\, j \neq i} (1 - a_j X).$$

Kennen wir $\sigma(X)$, so können wir daraus I_y ausrechnen, denn es gilt

$$I_y = \{i \,|\, 1 \leq i \leq n \text{ und } \sigma(1/a_i) = 0\}.$$

Kennen wir darüberhinaus auch $\omega(X)$, so können wir damit e berechnen, denn es gilt

$$e_i = \omega(1/a_i) \left(c_i \prod_{j \in I_y,\, j \neq i} (1 - a_j/a_i) \right)^{-1}.$$

Wir bemerken, daß $\sigma(X)$ und $\omega(X)$ teilerfremd sind.

Für die zentrale Gleichung rechnen wir in dem formalen Laurentreihenkörper $\mathbb{F}_q((X))$. Wir betten $\mathbb{F}_q(X)$ wie üblich nach $\mathbb{F}_q((X))$ ein und erhalten:

$$\begin{aligned} \frac{\omega(X)}{\sigma(X)} &= \sum_{i \in I_y} e_i c_i \frac{1}{1 - a_i X} = \sum_{i \in I_y} e_i c_i \sum_{j=0}^{\infty} (a_i X)^j = \sum_{j=0}^{\infty} \left(\sum_{i \in I_y} e_i c_i a_i^j \right) X^j \\ &= \sum_{j=0}^{\infty} \left(\sum_{i=1}^{n} e_i c_i a_i^j \right) X^j. \end{aligned}$$

Diese Gleichung besagt, daß wir $\sigma(X)$ und $\omega(X)$ im Prinzip aus den Werten $\sum_{i=1}^{n} e_i c_i a_i^j$ für genügend viele j ausrechnen können.

Es ergeben sich zwei Fragen:

1. Wie erhalten wir die Werte $\sum_{i=1}^{n} e_i c_i a_i^j$?
2. Wieviele benötigen wir davon und wie rechnen wir daraus $\sigma(X)$ und $\omega(X)$ aus?

Zur Beantwortung der ersten Frage nehmen wir an, daß C die Kontrollmatrix

$$H = \left(c_i a_i^j\right)_{\substack{1 \leq i \leq n \\ 0 \leq j \leq n-k-1}}$$

besitzt. Diese Matrix hat Rang $n-k$ (eine nichttriviale Linearkombination der Zeilen würde die Existenz eines Polynoms ungleich Null vom Grad $\leq n-k-1$ mit n Nullstellen implizieren, was einen Widerspruch ergibt). Mit anderen Worten nehmen wir $C = RS_{n-k}(a, c)^{\perp}$ an, werden also auf $C = RS_k(a, b)$ für ein $b \in \mathbb{F}_q^n$ geführt. Wir erhalten damit die ersten $n - k$ Glieder der Potenzreihe durch die Koordinaten des Syndroms $s(e) = s(y)$ (wir zeigen unten, daß diese auch ausreichen).

Zur Beantwortung der zweiten Frage verwenden wir die Padéapproximation.

Padéapproximation

Die Padéapproximation behandelt das folgende allgemeine Problem: Sind $f, g \in \mathbb{F}_q[X]$ teilerfremd und $h \in \mathbb{F}_q[[X]]$ mit $f/g = h$ in $\mathbb{F}_q[[X]]$ (h Taylorreihe von f/g), wieviele Terme von h werden benötigt, um f und g bis auf skalare Vielfache eindeutig zu rekonstruieren?

1 Satz. *Sei $h \in \mathbb{F}_q[[X]]$ und seien $s, r \geq 0$. Dann hat die Gleichung*

$$f \equiv gh \bmod X^{r+s+1}$$

bis auf skalare Vielfache höchstens eine Lösung (f, g) in $\mathbb{F}_q[X]^2$ mit $\deg(f) \leq r$ und $\deg(g) \leq s$ sowie $\gcd(f, g) = 1$.

Die Polynome f und g können mit $O((r+s)^2)$ arithmetischen Operationen in $\mathbb{F}_q$ berechnet werden.

Der Satz kann auf verschiedene Weisen bewiesen werden. Wir verwenden das Analogon des LLL-Algorithmus für Funktionenkörper.

Sei $M \in \mathbb{F}_q[X]^{n \times m}$. Mit M_i bezeichnen wir die Spalten von M und mit $\deg(M_i)$ den in M_i auftretenden Maximalgrad. Wir nennen M spaltenreduziert, wenn $\deg(\sum_{i=1}^n \lambda_i M_i) = \max_{1 \leq i \leq n} \deg(\lambda_i M_i)$ für alle $\lambda_i \in \mathbb{F}_q[X]$ gilt.

2 Satz. *Sei $M \in \mathbb{F}_q[X]^{n \times n}$ mit $\det(M) \neq 0$.*

(*i*) *Es gibt $N, T \in \mathbb{F}_q[X]^{n \times n}$ mit N spaltenreduziert, $\deg(\det(T)) = 0$ und*

$$N = MT.$$

Die Matrix N kann mit dem LLL-Algorithmus für Funktionenkörper in $O(n^3 + n^2 d^2)$ arithmetischen Operationen in $\mathbb{F}_q$ berechnet werden, wobei d den in M auftretenden maximalen Grad bezeichnet.

(*ii*) *M ist genau dann spaltenreduziert, wenn $\deg(\det(M)) = \sum_{i=1}^n \deg(M_i)$ gilt.*

Grobe Beweisskizze. (*i*): Man führt einen verallgemeinerten euklidischen Algorithmus auf den Spalten mit dem Ziel der Verringerung der Zahl $\sum_{i=1}^n \deg(M_i)$ durch. Man kann damit $\sum_{i=1}^n \deg(M_i)$ genau dann verringern, wenn M nicht spaltenreduziert ist.

(*ii*): Man überlegt sich, daß man bei $\deg(\det(M)) < \sum_{i=1}^n \deg(M_i)$ einen $\sum_{i=1}^n \deg(M_i)$ echt verringernden Reduktionsschritt ausführen kann. Die Laufzeit ergibt sich wie folgt: Die Ermittlung, ob und wie ein Reduktionschritt durchgeführt werden kann, benötigt mit dem Gaußalgorithmus n^3 Operationen in $\mathbb{F}_q$. Die Durchführung eines Reduktionsschritts benötigt nd Operationen in $\mathbb{F}_q$. Wir benötigen maximal nd Reduktionsschritte. □

Beweis von Satz 1. Sei $n = r + s + 1$ und $\tilde{h} \in \mathbb{F}_q[X]$ mit $\tilde{h} \equiv h \bmod X^n$. Sei

$$\Lambda_0 = \{(f, g)^t \mid f, g \in \mathbb{F}_q[X] \text{ mit } f \equiv gh \bmod X^n\}.$$

Dann ist Λ_0 ein $\mathbb{F}_q[X]$-Untermodul von $\mathbb{F}_q[X]^2$ und eine Basis von Λ_0 wird durch die Spalten der Matrix

$$M_0 = \begin{pmatrix} X^n & \tilde{h} \\ 0 & 1 \end{pmatrix}$$

gegeben.

Sei nun $\phi : \mathbb{F}_q[X]^2 \to \mathbb{F}_q[X]^2$, $(f, g)^t \mapsto (X^{n-r} f, X^{n-s} g)^t$. Dann ist ϕ injektiv und $\mathbb{F}_q[X]$-linear. Sei

$$\Lambda = \phi(\Lambda_0)$$

und

$$M = \begin{pmatrix} X^{2n-r} & X^{n-r}\tilde{h} \\ 0 & X^{n-s} \end{pmatrix}.$$

Die Spalten von M bilden eine Basis von Λ. Für $(f, g)^t \in \Lambda_0$ gilt $\deg(f) \leq r$ und $\deg(g) \leq s$ genau dann, wenn $\deg(\phi((f, g)^t)) \leq n$ ist.

Nach Satz 2 gibt es eine reduzierte Matrix N, deren Spalten ebenfalls Λ erzeugen und für die $\deg(N_1) + \deg(N_2) = \det(M) = 3n - r - s$ gilt. Wir nehmen ohne Einschränkung $\deg(N_1) \leq \deg(N_2)$ an.

Nach diesen Vorbereitungen sei $(f, g) \in \Lambda_0$ mit $\deg(f) \leq r$, $\deg(g) \leq s$ sowie $\gcd(f, g) = 1$. Sei $v = \phi((f, g)^t)$. Dann gilt $\deg(v) \leq n$. Es gibt $\lambda_i \in \mathbb{F}_q[X]$ mit $v = \lambda_1 N_1 + \lambda_2 N_2$. Wegen $\deg(v) = \max_{1 \leq i \leq 2} \deg(\lambda_i N_i) \leq n < (3n - r - s)/2$ folgt $\deg(N_1) < \deg(N_2)$ und $v = \lambda_1 N_1$. Wegen $\gcd(f, g) = 1$ folgt $\deg(\lambda_1) = 0$. Damit stimmt v bis auf ein skalares Vielfaches mit N_1 überein. Nun wurde N_1 unabhängig von f, g konstruiert, so daß die Eindeutigkeitsaussage des Satzes folgt. □

Anwendung auf die Dekodierung

Aufgrund der eingangs gemachten Annahmen wissen wir $\deg(\sigma(X)) \leq t$ und $\deg(\omega(X)) \leq t-1$. Zur Anwendung von Satz 1 benötigen wir die Potenzreihe modulo X^{2t}, daß heißt wir benötigen die ersten $2t$ Koeffizienten der Potenzreihe. Wie oben dargelegt erhalten wir die ersten $n-k$ Koeffizienten aus dem Syndrom $s(e) = s(y)$. Es gilt $n - k = d + 1 \geq 2t$, so daß diese Koeffizienten bereits ausreichen.

Zur Dekodierung ermitteln wir nach Satz 1 die Polynome $\omega(X)$ und $\sigma(X)$ aus $\sum_{j=0}^{2t-1} \left(\sum_{i=1}^{n} e_i c_i a_i^j\right) X^j$ für $r = t-1$ und $s = t$, und gehen dann wir oben beschrieben zur Berechnung von I_y und e vor.

Bemerkungen

Die Padéapproximation kann auch mit dem Kettenbruchverfahren berechnet werden. Die spezielle Verwendung des LLL-Algorithmus für Funktionenkörper bei der Padéapproximation liefert gerade den Kettenbruchalgorithmus.

Man kann auch für rationale Zahlen fragen, wie man eine Approximation h als f/g mit $f, g \in \mathbb{Z}$ teilerfremd und größenbeschränkt schreiben kann. Die Situation ist dann ganz analog wie bei der Padéapproximation: Man verwendet den Kettenbruchalgorithmus, welcher als Spezialfall des LLL-Algorithmus aufgefaßt werden kann.

Sowohl der LLL-Algorithmus als auch der Kettenbruchalgorithmus können als Verallgemeinerung des euklidischen Algorithmus angesehen werden.

Die angegebenen Laufzeiten beruhen auf der Verwendung relativ trivialer Algorithmen. Der wesentliche Unterschied des hier beschriebenen Dekodierverfahrens zum in der vorigen VL beschriebenen Dekodierverfahren liegt darin, daß hier nur Arithmetik mit Polynomen über $\mathbb{F}_q$ vom Grad $O(t)$ betrieben wird, während letztes Mal lineare Algebra über $\mathbb{F}_q$ in der Dimension $O(t)$ angewendet wurde. Ersteres hat eine Laufzeit von $O(t^2)$ Operationen in $\mathbb{F}_q$, während letzteres eine Laufzeit von $O(t^3)$ Operationen in $\mathbb{F}_q$ aufweist. Ähnlich sieht es mit dem Speicherbedarf aus. Das hier beschriebene Dekodierverfahren ist also bei großen Werten von t effizienter.

Ein Algorithmus zum Lösen des speziellen linearen Gleichungssystems zwischen den Syndromen und der Koeffizienten des (in der vorigen VL) definierten Fehlerlokalisierungspolynoms ist der Berlekamp-Massey Algorithmus für lineare Rekurrenzen. Ich habe die Details nicht nachgeprüft, aber es sieht so aus, als wäre der Berlekamp-Massey Algorithmus zumindest im vorliegenden Fall nur eine Variante des Kettenbruchalgorithmus beziehungsweise des LLL-Algorithmus für Funktionenkörper. Siehe Literaturreferenz im Codierungstheorie-Buch von Willems, Schlußkapitel.