Diskrete Mathematik II

Alexander May

Fakultät für Mathematik
Ruhr-Universität Bochum

Sommersemester 2008

Organisatorisches

- Vorlesung: **Mo 12-14** in HIA , **Di 10-11** in NA 6/99
  (3+1 SWS, 6.75 CP)
- Übung: **Di 11-13** in ND2/99
  - Assistent: **Mathias Herrmann**, Korrektor: **M. Mansour Al-Sawadi**
  - Übung ist zweiwöchentlich: gerade/ungerade Woche
  - Übungsaufgaben werden korrigiert.
  - Gruppenabgaben bis 4 Personen
  - Bonussystem:
    1/3-Notenstufe für 50%, 2/3-Notenstufe für 75%
    Gilt nur, wenn man die Klausur besteht!
  - Musterlösungen
- Klausur: Ende Juli

Abstimmungen

- Zusammensetzung des Auditoriums
- Vorlesungsform
  - Folienunterstützter Tafelanschrieb
  - Nur Tafelanschrieb (Folien als Skript)

Themengebiete

1. Kodierungstheorie
   - Komprimierende Codes
   - Beispiel Anwendungen: Kommunikation (Mobilfunk, Internet), Speicher (MP3)
   - Fehlererkennende Codes
   - Ausfalltolerante Codes
   - Beispiel Anwendungen: Mobilfunk, Internet, CD, Secret Sharing, Kryptosystem
2. Algorithmische Zahlentheorie
   - Quadratische Reste
   - Beispiel Anwendungen: Zufallszahlengenerator, Identity-Based Encryption
   - Elliptische Kurven
   - Beispiel Anwendungen: Kryptosystem, Primzahltest
3. Komplexitätstheorie
   - Klassen P und NP
   - Reduktionen
   - Anwendung: Sicherheitsbeweise in der Kryptographie

Weiterführende Referenzen Kodierungstheorie

- M. Sudan, “Algorithmic Introduction to Coding Theory”, Skript MIT
- W. Trappe, L. Washington, “Introduction to Cryptography with Coding Theory”, Pearson 2006
- J. Blömer, “Algorithmische Codierungstheorie”, Skript Uni Paderborn
- J.H. van Lint, “Introduction to Coding Theory”, Springer 1991
- T. Ihringer, “Diskrete Mathematik”, Teubner 1994

Unser Modell

- Shannon 1948: Informationstheorie und Mathematik der Kommunikation
- Hamming 1950: Erste Arbeit über fehlerkorrigierende Codes

Modell:
Sender $\longrightarrow$ Kodierer $\longrightarrow$ Kanal $\longrightarrow$ Dekodierer $\longrightarrow$ Empfänger

- Kanal ist bandbreitenbeschränkt (Kompression)
- Kanal ist fehleranfällig (Fehlerkorrektur)
  - Bits können ausfallen: $0 \longrightarrow \epsilon$, $1 \longrightarrow \epsilon$
  - Bits können kippen: $0 \longrightarrow 1$, $1 \longrightarrow 0$

Motivierendes Bsp: Datenkompression

Szenario:

- Kanal ist **fehlerfrei**.
- Übertragen gescannte Nachricht:
  Wahrscheinlichkeiten: 99% weißer, 1% schwarzer Punkt.
- Weiße Punkte werden mit 0 kodiert, schwarze mit 1.

Kodierer:

- Splitten Nachricht in Blocks der Größe 10.
- Wenn Block x=0000000000, kodiere mit 0, sonst mit 1x.
- 1 dient als Trennzeichen beim Dekodieren.

Dekodierer:

- Lese den Code von links nach rechts.
- Falls 0, dekodiere 0000000000.
- Falls 1, übernehme die folgenden 10 Symbole.

Erwartete Codelänge

Sei $q := \Pr[\text{Block ist } 0000000000] = (0.99)^{10} \geq 0.9$.
Sei Y Zufallsvariable für die Codelänge eines 10-Bit Blocks:

$$E[Y] = \sum_{y \in \{0,1x\}} |y| \cdot Pr(Y = y) = 1 \cdot q + 11 \cdot (1 - q) = 11 - 10q.$$

- D.h. erwartete Länge der Kodierung eines 10-Bit Blocks ist
$$11 - 10q \leq 2 \text{ Bit}.$$
- Datenkompression der Nachricht auf 20%.
- Können wir noch stärker komprimieren?
- Entropie wird uns Schranke für Komprimierbarkeit liefern.

Ausblick: fehlerkorrigierende Codes

Szenario: Binärer symmetrischer Kanal

- Bits 0,1 kippen mit Ws $p, p < \frac{1}{2}$ zu 1,0. (Warum $< \frac{1}{2}$?)
- Korrekte Übertragung $0 \mapsto 0$, $1 \mapsto 1$ mit Ws $1 - p$.
- In unserem Beispiel $p = 0.1$.

Kodierer:

- Verdreifache jedes Symbol, d.h. $0 \mapsto 000$, $1 \mapsto 111$
- Repetitionscode der Länge 3.

Dekodierer:

- Lese den Code in 3er-Blöcken.
- Falls mindestens zwei Symbole 0 sind, dekodiere zu 0.
- Sonst dekodiere zu 1.

Ws Dekodierfehler

Symbol wird falsch dekodiert, falls mind. zwei der drei Bits kippen.

$$\begin{aligned} & \text{Ws}(\text{Bit wird falsch dekodiert}) \\ = \; & \text{Ws}(\text{genau 2 Bits kippen}) + \text{Ws}(\text{genau 3 Bits kippen}) \\ = \; & 3 * p^2 * (1-p) + p^3 = 3 * 10^{-2} * (1 - 10^{-1}) + 10^{-3} \end{aligned}$$

- Ohne Kodierung Fehlerws von 0.1.
- Mit Repetitionskode Fehlerws von ≈ 0.03.
- Nachteil: Codierung ist dreimal so lang wie Nachricht.
- **Ziel:**
  Finde guten Tradeoff zwischen Fehlerws und Codelänge.

Ausblick: fehlertolerante Codes

Szenario: Binärer Ausfallkanal

- Bits 0,1 gehen mit Ws $p, p < \frac{1}{2}$ verloren, d.h. $0 \mapsto \epsilon$ bzw. $1 \mapsto \epsilon$.
- Korrekte Übertragung $0 \mapsto 0$, $1 \mapsto 1$ mit Ws $1 - p$.
- In unserem Beispiel $p = 0.1$.

Verwenden wieder Repetitionskode der Länge 3.

- Fehler beim Dekodieren: Alle drei Symbole gehen verloren.
- $\text{Ws}(\text{Bit kann nicht dekodiert werden}) = p^3 = 0.001$.
- Fehlerws kleiner beim Ausfallkanal als beim symmetrischen Kanal.

Definition Code

- Alphabet $A = \{a_1, \ldots, a_n\}$, Menge von Symbolen a_i
- Nachricht $m \in A^*$

Definition Code

Sei A ein Alphabet. Eine (binäre) *Codierung C des Alphabets A* ist eine injektive Abbildung

$$\begin{aligned} C: \quad & A \to \{0,1\}^* \\ & a_i \mapsto C(a_i). \end{aligned}$$

Die *Codierung einer Nachricht* $m = a_{i_1} \ldots a_{i_\ell} \in A^*$ definieren wir

$$C(m) = C(a_{i_1}) \ldots C(a_{i_\ell}) \quad \text{(Erweiterung von } C \text{ auf } A^*).$$

Die Abbildung C heißt *Code*.

Bezeichnungen Code

- Die Elemente $c_i := C(a_i)$ bezeichnen wir als *Codeworte*.
- Wir bezeichnen sowohl die Abbildung von Nachrichten auf Codeworte als auch die *Menge der Codeworte* mit dem Buchstaben C.
- Falls $C \subseteq \{0,1\}^n$ spricht man von einem *Blockcode* der Länge n. In einem Blockcode haben alle Codeworte die gleiche Länge.

Entschlüsselbarkeit von Codes

Szenario: Datenkompression in fehlerfreiem Kanal

Definition eindeutig entschlüsselbar

Ein Code heißt eindeutig entschlüsselbar, falls jedes Element aus $\{0,1\}^*$ Bild höchstens einer Nachricht ist. D.h. die Erweiterung der Abbildung C auf A^* muss injektiv sein.

Definition Präfixcode

Ein Code $C = \{c_1, \ldots, c_n\}$ heißt Präfixcode, falls es keine zwei Codeworte $c_i \neq c_j$ gibt mit

$$c_i \text{ ist Präfix (Wortanfang) von } c_j.$$

Beispiel

	a_1	a_2	a_3
C_1	0	0	1
C_2	0	1	00
C_3	0	01	011
C_4	0	10	11

- C_1 ist kein Code, da $C : A \to \{0,1\}^*$ nicht injektiv.
- C_2 ist nicht eindeutig entschlüsselbar, da $C : A^* \to \{0,1\}^*$ nicht injektiv.
- C_3 ist eindeutig entschlüsselbar, aber kein Präfixcode.
- C_4 ist ein Präfixcode.

Präfixcodes sind eindeutig entschlüsselbar.

Satz: Präfixcode eindeutig entschlüsselbar

Sei $C = \{c_1, \ldots, c_n\}$ ein Präfixcode. Dann kann jede Nachricht $C(m)$ in Zeit $\mathcal{O}(|C(m)|)$ eindeutig zu m decodiert werden.

- Zeichne binären Baum
  - Kanten erhalten Label 0 für linkes Kind, 1 für rechtes Kind.
  - Codewort $c_i = c_{i_1} \ldots c_{i_k}$ ist Label des Endknoten des Pfads von der Wurzel mit den Kantenlabeln $i_1, \ldots, i_n$
- **Präfixeigenschaft:** Kein einfacher Pfad von der Wurzel enthält zwei Knoten, die mit Codeworten gelabelt sind.
- Codewort c_i ist Blatt in Tiefe c_i

Algorithmus Dekodierung Präfix

Algorithmus Dekodierung Präfix

1. Lese $C(m)$ von links nach rechts.
2. Starte bei der Wurzel. Falls 0, gehe nach links. Falls 1, gehe nach rechts.
3. Falls Blatt mit Codewort $c_i = C(a_i)$ erreicht, gib a_i aus und iteriere.

Laufzeit: $\mathcal{O}(|C(m)|)$

Woher kommen die Nachrichtensymbole?

Modell

- *Quelle Q* liefert Strom von Symbolen aus A.
- Quellwahrscheinlichkeit: $\mathrm{Ws}(\text{ Quelle liefert } a_i) = p_i$
- Ws p_i ist unabhängig von der Zeit und vom bisher produzierten Strom (erinnerungslose Quelle)
- X_i: Zufallsvariable für die i-te Position im Strom, d.h.

$$\mathrm{Ws}(X_i = a_j) = p_j \quad \text{für } j = 1, \ldots, n \text{ und alle } i.$$

Ziel: Kodiere Elemente a_j mit großer Ws p_j mit kleiner Codewortlänge.

Kompakte Codes

Definition Erwartete Codewortlänge

Sei Q eine Quelle mit Alphabet $A = a_1, \ldots, a_n$ und Quellwahrscheinlichkeiten $p_1, \ldots, p_n$. Die Größe

$$E(C) := \sum_{i=1}^{n} p_i |C(a_i)|$$

bezeichne die erwartete Codewortlänge.

Definition Kompakter Code

Ein Code C heißt kompakt bezüglich einer Quelle Q, falls er *minimale erwartete Codewortlänge* besitzt.