Kompression

Tim Kilian

Seminar „Effiziente Programmierung“
Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

18.01.2018

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Gliederung

Was ist Datenkompression?

- Auch Datenkomprimierung genannt.
- Ziel: Eine Repräsation X_c einer Eingabe X, welche möglichst weniger Bits als X zur Darstellung benötigt.
- Ein Algorithmenpaar oder Datenkompressionschema (Codec) besteht aus einem Kompressions- und Dekompressionsalgorithmus
- Zwei Arten der Kompresion
  - Verlustfreie Kompression (Redundanz)
  - Verlustbehaftete Kompression (Irrelevanz)

Definition (verlustfrei, verlustbehaftet)

Sei X eine Eingabe und Y eine Rekonstruktion, so ist ein Datenkompressionschema verlustfrei, wenn $X = Y$ und verlustbehaftet, wenn $X \neq Y$ gilt.

Warum wollt ihr etwas über Kompression erfahren?

- Kompression ist überall!
- Anwendungsbeispiele: digitaler Rundfunk und digitales Fernsehen
- Datenübertragung
- Speicherplatzreduktion

Definition (Kompressionsquotient, compression ratio, Kompressionsfaktor)

Sei X eine Eingabe und X_c eine Repräsentation dieser, so nennt man X_c/X Kompressionsquotient und den Kehrwert Kompressionsfaktor.

- Kompressionsfaktor möglichst groß

Wie kann man Daten komprimieren?

Verlustfreie Kompressionsverfahren (Redundanz)

- Kompression nur bedingt möglich.
- Keinerlei Informationen gehen verloren.

Beispiel:

Text:	Dies ist ein Text, welcher das Wort Text enthält.
Codiert:	Dies ist ein Text, welcher das Wort -4 enthält.
Codiert:	Dies ist ein #0, welcher das Wort #0 enthält.

Verlustbehaftete Kompressionsverfahren (Irrelevanz)

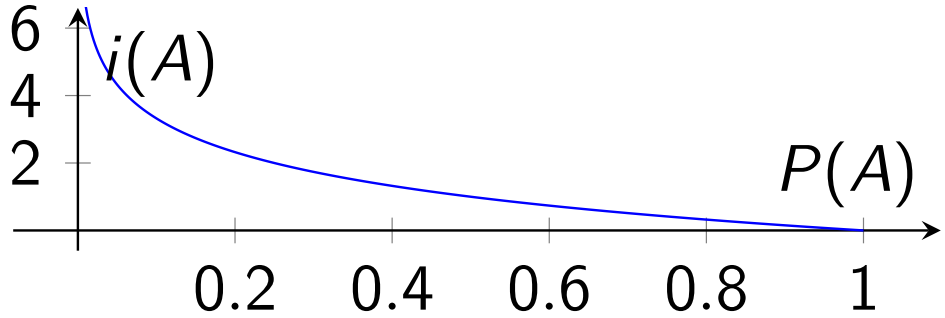
- Kompression ist immer möglich (bis nur noch 1 Bit übrig bleibt).
- Der Algorithmus entscheidet, welche Daten redundant sind und verworfen werden.

Beispiel:

Text:	Dies ist ein Text, welcher das Wort Text enthält.
Codiert:	Dies ist ein Text.

Entropie

Shannon definierte das Maß der Informationen als $i(A) = \log_2(\frac{1}{P(A)})$.



Definition (Entropie)

Sei X eine Eingabe und $\Sigma = \{x_1, \ldots, x_n\}$ das Alphabet der Nachricht.
Sei außerdem A_i ein Ereignis, dass das Symbol x_i vorkommt und
$S = \{A_1, ..., A_n\}$ ein Menge aller Ereignisse der Symbole in Σ,
so lautet der mittlere Informationsgehalt (Entropie) der Nachricht X

$$H(S) = \sum_{i=1}^{n} P(A_i) \cdot i(A_i) = -\sum_{i=1}^{n} P(A_i) \cdot \log_2(P(A_i)).$$

Der Erwartungswert $H(S)$ bezeichnet die mittlere Anzahl von Bits, um eine Nachricht zu codieren.

Variable Codelänge

Definition (Code)

Die Menge aller Codewörter

Definition (Blockcode)

Alle Codewörter haben die selbe Länge

- Minimierung der Datenmenge durch Anpassung an die Symbolhäufigkeiten
- Häufige Symoble bekommen kurze Codewörter, seltene Symbole längere Codewörter.
- Anders als bei Blockcodes, ist die Trennung zwischen Codewörtern nicht mehr durch Abzählen möglich.

Fano-Bedingung

Definition (Präfix-Eigenschaft, Fano-Bedinung)

Kein Wort aus einem Code bildet den Anfang eines anderen Codeworts

- ein Präfix-Code ist eindeutig decodierbar
- Blockcodes sind Präfix-Codes

Definition (Satz von Kraft, Kraftsche Ungleichung)

Für die Existenz eines eindeutig decodierbaren n-elementigen Codes mit Codelängen $l_1, \ldots, l_n$ über einen binären Zeichervorrat Σ ist $\sum_{i=1}^{n} \frac{1}{2^{l_i}} \leq 1$ eine notwendige und hinreichende Bedingung.

Shannon Codierung

Sei $S = \{A_1, \ldots, A_n\}$ eine Ereignismenge und $P(A_i)$ deren Wahrscheinlichkeiten, so sortiere alle Ereignisse in S in absteigender Wahrscheinlichkeit, sodass $P(A_1) \geq \cdots \geq P(A_n)$ gilt.

Definiere für $i = 1: \; P_1 = 0$ und für $i > 1: \; P_i = \sum_{j=1}^{i-1} P(A_j)$.

Sei Außerdem: $l_i = \lceil i(A_i) \rceil = \lceil -\log_2(P(A_i)) \rceil$, so ergibt sich der Code aus den abgelesenen l_i Bits der binären Repräsentation von P_i.

Beispiel:

A	$P(A)$	l_i	P_i	$(P_i)_2$	Code
c	0.4	2	0	$0.0000\ldots$	00
a	0.15	3	0.4	$0.0110\ldots$	011
b	0.15	3	0.55	$0.1000\ldots$	100
d	0.15	3	0.7	$0.1011\ldots$	101
e	0.15	3	0.85	$0.1101\ldots$	110

$H(S) \approx 2.17095$

$L = 2.6$

Der Shannon-Algorithmus generiert einen Präfixcode.

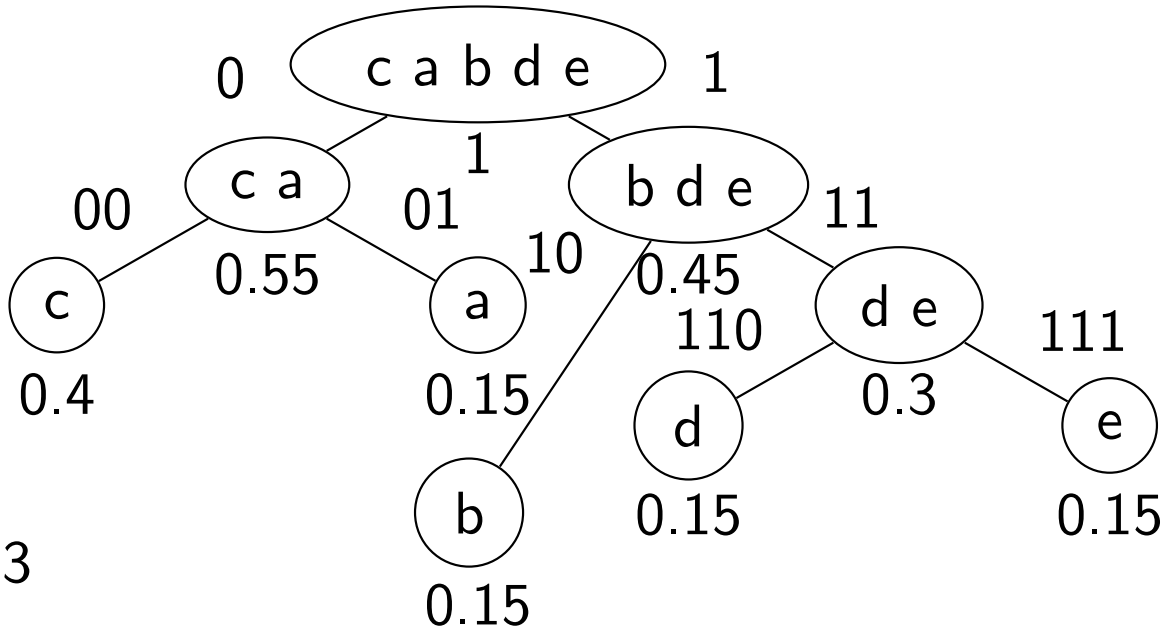
Shannon-Fano Codierung

Sei $V = \{A_1, \ldots, A_n\}$ und $P(A_i)$ deren Wahrscheinlichkeiten, so sortiere die Ereignisse in V, sodass $P(A_1) \geq \cdots \geq P(A_n)$ gilt.

1. Starte mit einem Knoten V der alle sortierten Ereignisse enthält.
2. Falls $|V| = 1$ gebe den Knoten zurück
3. Teile den Knoten in V_1 und V_2, sodass $\sum_{A \in V_1} P(A) \approx \sum_{A \in V_2} P(A)$
4. Starte den Algorithmus rekursiv mit $V = V_1$ und $V = V_2$

Beispiel:

A	$P(A)$	Code
c	0.4	00
a	0.15	01
b	0.15	10
d	0.15	110
e	0.15	111



$H(S) \approx 2.17095$ $L = 2.3$

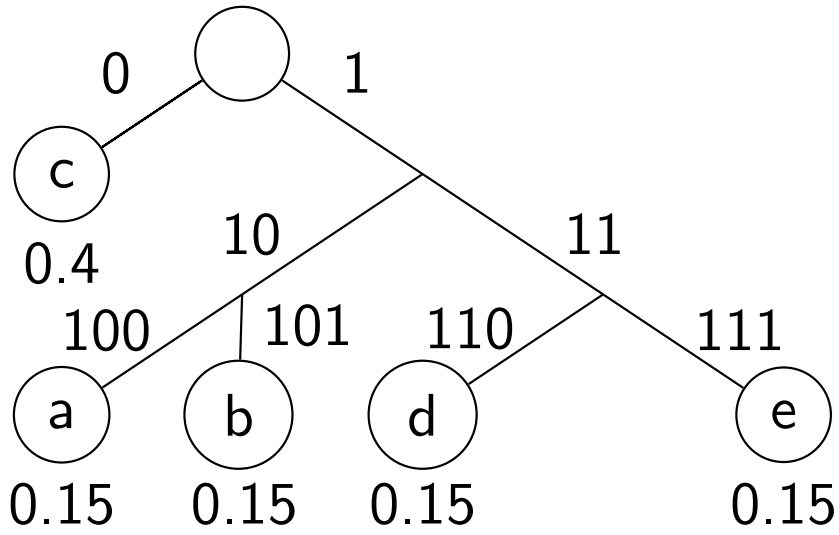
Der Shannon-Fano-Algorithmus generiert einen Präfixcode.

Huffman Codierung

1. Starte mit einem Wald aus Bäumen, in dem jeder Baum ein Symbol darstellt und $w_i = P(A_i)$ das Gewicht des Baumes ist.
2. Wiederhole solange, bis der Wald nur noch aus einem Baum besteht:
   - Wähle die zwei Bäume mit den kleinsten Gewichten w_1 und w_2.
   - Verbinde diese zu einem neuen Baum mit dem Gewicht $w_r = w_1 + w_2$.

Beispiel:

A	$P(A)$	Code
c	0.4	0
a	0.15	100
b	0.15	101
d	0.15	110
e	0.15	111



$H(S) \approx 2.17095$ $L = 2.2$

Die erwartete Codelänge der Huffman-Codierung ist optimal.

LZ77 (LZ1) Gleitfenster-Algorithmus

1. Verschiebe den Zeiger auf das erste Symbol im Absuch-Puffer. Setze $L = 0$ und S dem ersten Symbol im Codier-Puffer.
2. Verschiebe den Zeiger im Absuch-Puffer so weit nach links, bis er das Zeichen an Stelle K findet, das gleich dem Startsymbol ist.
3. Wenn die Grenze im Puffer erreicht wurde, gib den Code (K, L, S) aus.
4. Setze L gleich der Länge der gleichzeitig gelesenen gleichen Zeichen vom Absuch- und Codier-Puffer und fahre mit Schritt 2 fort.

Die Berechnung wird anhand der Zeichenkette `ababcabca` veranschaulicht

	Absuch-Puffer						Codier-Puffer					
Zeile	6	5	4	3	2	1	0	1	2	3	4	Code
1							a	b	a	b	c	(0,0,a)
2						a	b	a	b	c	a	(0,0,b)
3					a	b	a	b	c	a	b	(2,2,c)
4		a	b	a	b	c	a	b	c	a		(3,3,a)
6	b	c	a	b	c	a						(-,-,-)

Übersicht (Zeittafel und bekannte Methoden)[1]

1833–65	Morse-Code
1883	Forsyth-Edwards-Notation
1949	Shannon-Fano-Kodierung
1952	Huffman-Kodierung
1977	LZ77
1982	LZSS (GZip, Zip)
1984	LZW (TIFF, GIF, PNG)
1993	Deflate
1995	ZLib
2011	LZ4
2013	Zopfli
2015	Brotli

verlustbehaftet	beides	verlustfrei
AAC (MPEG)		ALS (MPEG)
Dolby Digital		FLAC
MP3 (MPEG)	JPEG	PNG
	TIFF	
MPEG-4		
WMV		
...	...	...

[1] https://de.wikipedia.org/wiki/Datenkompression

TIFF, BMP, JPEG, PNG, GIF

Abbildung:
Lena[1].tiff (768 KB)

Abbildung:
Lena.bmp (768 KB)

Abbildung:
Lena.jpg (65 KB)

Abbildung:
Lena.png (509 KB)

Abbildung:
Lena.gif (208 KB)

Abbildung:
EP.tiff (49 KB)

Abbildung:
EP.bmp (48 KB)

Abbildung:
EP.jpg (17 KB)

Abbildung:
EP.png (6 KB)

Abbildung:
EP.gif (2 KB)

[1] http://sipi.usc.edu/database/database.php

Weiterführende Inhalte

- Adaptive Huffman-Codierung
  - On-the-fly Codierung
- Arithmetische Codes
  - Folge der Symbole → numerische Representation → binärer Code
- Calgary corpus [1]
  - Standard-Testsuite englischer Texte für Benchmarks
- USC-SIPI Image Database Suite [2]
  - Sammlung von Testbildern
- Kodak Lossless True Color Image Suite[3]
  - Hochauflösende Testbilder
- Hutter Prize[4]
  - Komprimieren von 100 MB natürlicher Sprache

[1] http://www.data-compression.info/Corpora/CalgaryCorpus/
[2] http://sipi.usc.edu/database/database.php
[3] http://r0k.us/graphics/kodak/
[4] http://prize.hutter1.net/

Literatur

[Sa06] Sayood, Khalid: Introduction to Data Compression, Morgan Kaufmann Publishers, 3. Auflage, 2006

[Da06] Dankmeier, Wilfried: Grundkurs Codierung, Vieweg, 3. Auflage, 2006

[LF11] Liskiewicz, Maciej; Fernau, Henning: Datenkompression, Universität Tier, SoSe 2011, https://www.uni-trier.de/fileadmin/fb4/prof/INF/TIN/Folien/DK/script.pdf, letzter Abruf am 17.01.2018

[Mä17] Mäder, Andreas: Codierung, Vorlesung zu Rechnerstrukturen, Kapitel 9, WiSe 17/18, https://tams.informatik.uni-hamburg.de/lectures/2017ws/vorlesung/rs/doc/rsWS17-09.pdf, letzter Abruf am 17.01.2018