Seminar Kompressionsalgorithmen
Huffman-Codierung, arithmetische Codierung

Junior Lekane Nimpa

Theoretische Informatik
RWTH-Aachen

4. April 2012

Übersicht

Einführung

Datenkompression

- Disziplin,die Kompressionsalgorithmen entwirft
- Ziel: effiziente Speicherung und Übertragung von Daten

Einführung

Datenkompression

- Disziplin,die Kompressionsalgorithmen entwirft
- Ziel: effiziente Speicherung und Übertragung von Daten

Kompressionsalgorithmus

- Kompressionsalgo A^* = Kompressionsalgo A' und Dekompressionsalgo A"
- A'(x) = y
- A"(y) = z

Einführung

Klassen von Kompressionsalgorithmen

- verlustfreie Kompressionsalgorithmen. Es gilt: $z = x$
- verlustbehaftete Kompressionsalgorithmen. Im Allgemeinen $z \neq x$

Einführung

Klassen von Kompressionsalgorithmen

- verlustfreie Kompressionsalgorithmen. Es gilt: $z = x$
- verlustbehaftete Kompressionsalgorithmen. Im Allgemeinen $z \neq x$

Gliederung des Kompressionsvorgangs

- Modellierung
- Codierung

Grundlagen

Alphabet, Symbol, Sequenz

- $A = \{a_1, a_2, \ldots, a_n\}$ ist ein Alphabet
- $a_i \in A$ ist ein Symbol
- $s = (a_{i1} a_{i2} \cdots a_{im}), m \geq 1$, Sequenz über A der Länge m
- $A^m = \{(a_{i1} a_{i2} \cdots a_{im}) | a_{ij} \in A\}$, $m \geq 1$
- $A^* = \bigcup_{m=1}^{\infty} A^m$ ist der *Kleenesche Abschluss von A*

Grundlagen

Datenquelle

- Datenquelle Q: Folge von Symbolen aus A
- Q diskret $\iff$ A endlich oder abzählbar unendlich
- Q gedächtnislos $\iff$ Auftreten eines Symbol unabhängig von den anderen
- $p_i = p(a_i)$: Auftretenswahrscheinlichkeit vom Symbol a_i
- $p = (p_1, \ldots, p_n)$: Wahrscheinlichkeitsverteilung der Symbole aus A

Grundlagen

Modell

$$M : A \rightarrow [0, 1); a_i \rightarrow p_M(a_i) = p_i$$

Grundlagen

Modell

$$M : A \to [0, 1); a_i \to p_M(a_i) = p_i$$

Code

A: Quellenalphabet und B ein endliches Alphabet.
$C : A \to B^*, a_i \to C(a_i)$
$l_i :=$ Länge von $C(a_i)$. Im Allgemeinen $B = \{0, 1\}$

Grundlagen

Modell

$$M : A \to [0,1); a_i \to p_M(a_i) = p_i$$

Code

A: Quellenalphabet und B ein endliches Alphabet.
$C : A \to B^*, a_i \to C(a_i)$
$l_i :=$ Länge von $C(a_i)$. Im Allgemeinen $B = \{0,1\}$

Erweiterung eines Codes C

$C^* : A^* \to B^*$,
$s = (a_{i1}a_{i2}\cdots a_{im}) \to C(s) = C(a_{i1})C(a_{i2})\cdots C(a_{im})$

Nützliche Eigenschaften eines Codes

Sei $C : A \rightarrow B^*$ ein Code

Eindeutige Dekodierbarkeit

C eindeutig decodierbar $\iff$
$\forall s_1, s_2 \in A^* : s_1 \neq s_2 \implies C^*(s_1) \neq C^*(s_2)$

Nützliche Eigenschaften eines Codes

Sei $C : A \rightarrow B^*$ ein Code

Eindeutige Dekodierbarkeit

C eindeutig decodierbar $\iff$
$\forall s_1, s_2 \in A^* : s_1 \neq s_2 \implies C^*(s_1) \neq C^*(s_2)$

Präfixfreier Code

C präfixfrei, wenn kein Codewort präfix eines anderen ist.

Nützliche Eigenschaften eines Codes

Sei $C : A \rightarrow B^*$ ein Code

Eindeutige Dekodierbarkeit

C eindeutig decodierbar $\iff$
$\forall s_1, s_2 \in A^* : s_1 \neq s_2 \implies C^*(s_1) \neq C^*(s_2)$

Präfixfreier Code

C präfixfrei, wenn kein Codewort präfix eines anderen ist.

Fano-Bedingung

C präfixfrei $\implies$ C eindeutig decodierbar

Nützliche Eigenschaften eines Codes

Sei $C : A \rightarrow B^*$ ein Code

Eindeutige Dekodierbarkeit

C eindeutig decodierbar $\iff$
$\forall s_1, s_2 \in A^* : s_1 \neq s_2 \implies C^*(s_1) \neq C^*(s_2)$

Präfixfreier Code

C präfixfrei, wenn kein Codewort präfix eines anderen ist.

Fano-Bedingung

C präfixfrei $\implies$ C eindeutig decodierbar

Nützliche Eigenschaften eines Codes

Beispiel: Code C_1

Symbol	Codewort
A	0
B	01
C	10

Beispiel: Code C_2

Symbol	Codewort
A	0
B	10
C	110

Nützliche Eigenschaften eines Codes

Beispiel: Code C_1

Symbol	Codewort
A	0
B	01
C	10

Beispiel: Code C_2

Symbol	Codewort
A	0
B	10
C	110

Example

Zu decodieren: 001010.

Nützliche Eigenschaften eines Codes

Beispiel: Code C_1

Symbol	Codewort
A	0
B	01
C	10

Beispiel: Code C_2

Symbol	Codewort
A	0
B	10
C	110

Example

Zu decodieren: 001010.

$$001010 \xrightarrow{C_1} \begin{cases} 0 & 0 & 10 & 10 \\ 0 & 01 & 0 & 10 \\ 0 & 01 & 01 & 0 \end{cases} \xrightarrow{C_1} \begin{cases} AACC \\ ABAC \\ ABBA \end{cases}$$

Nützliche Eigenschaften eines Codes

Beispiel: Code C_1

Symbol	Codewort
A	0
B	01
C	10

Beispiel: Code C_2

Symbol	Codewort
A	0
B	10
C	110

Example

Zu decodieren: 001010.

$$001010 \xrightarrow{C_1} \begin{cases} 0 & 0 & 10 & 10 \\ 0 & 01 & 0 & 10 \\ 0 & 01 & 01 & 0 \end{cases} \qquad \xrightarrow{C_1} \begin{cases} AACC \\ ABAC \\ ABBA \end{cases}$$

$$001010 \xrightarrow{C_2} 0 \quad 0 \quad 10 \quad 10 \qquad \xrightarrow{C_2} \quad AABB.$$

Nützliche Ergebnisse aus der Informationstheorie

Ausgangspunkt

Datenquelle Q

- über $A = \{a_1, a_2, \ldots, a_n\}$ mit $p = (p_1, p_2, \ldots, p_n)$

Nützliche Ergebnisse aus der Informationstheorie

Ausgangspunkt

Datenquelle Q

- über $A = \{a_1, a_2, \ldots, a_n\}$ mit $p = (p_1, p_2, \ldots, p_n)$

Informationsgehalt eines Symbols

- $I(a_i) = I(p_i) = log_2(\frac{1}{p_i}) = -log_2(p_i)$ [Bits]

Nützliche Ergebnisse aus der Informationstheorie

Ausgangspunkt

Datenquelle Q

- über $A = \{a_1, a_2, \ldots, a_n\}$ mit $p = (p_1, p_2, \ldots, p_n)$

Informationsgehalt eines Symbols

- $I(a_i) = I(p_i) = log_2(\frac{1}{p_i}) = -log_2(p_i)$ [Bits]

Informationsgehalt einer Sequenz

Sei $s = (a_{i1} a_{i2} \cdots a_{im})$ eine Sequenz.
$p(s) = p(a_{i1}) \cdot p(a_{i2}) \cdots p(a_{im})$

- $I(s) = I(p(s)) = \sum_{j=1}^{m} I(p_{ij})$ [Bits]

Nützliche Ergebnisse aus der Informationstheorie

Definition

Entropie von Q: durchschnittlicher Informationsgehalt aller a_i

Nützliche Ergebnisse aus der Informationstheorie

Definition

Entropie von Q: durchschnittlicher Informationsgehalt aller a_i

formal

Die Entropie $H(p) = \sum_{i=1}^{n} p_i \cdot log_2(\frac{1}{p_i}) = -\sum_{i=1}^{n} p_i \cdot log_2(p_i)$
Einheit : [Bits/Symbol]

Nützliche Ergebnisse aus der Informationstheorie

Definition

Entropie von Q: durchschnittlicher Informationsgehalt aller a_i

formal

Die Entropie $H(p) = \sum_{i=1}^{n} p_i \cdot log_2(\frac{1}{p_i}) = -\sum_{i=1}^{n} p_i \cdot log_2(p_i)$
Einheit : [Bits/Symbol]

Entropie erweiterter Alphabete

A^m , Alphabet mit Wahrscheinlichkeitsverteilung p^m.

- $H(p^m) = m \cdot H(p)$

Nützliche Ergebnisse aus der Informationstheorie

sei $C : A \rightarrow B^*$ ein präfixfreier Code.

Mittlere Länge L_C von C

$$L_C = \sum_{j=1}^{n} l_j \cdot p_j$$

Nützliche Ergebnisse aus der Informationstheorie

sei $C : A \to B^*$ ein präfixfreier Code.

Mittlere Länge L_C von C

$$L_C = \sum_{j=1}^{n} l_j \cdot p_j$$

Satz ("Noiseless Coding theorem "von Claude Shannon)

$$H(p) \leq L_C$$

Das Huffman-Verfahren

- 1952 von *David Albert Huffman* veröffentlicht.
- sehr schnell wegen mathematischer Einfachheit beliebt.
- generiert präfixfreien Code

Das Huffman-Verfahren

- 1952 von *David Albert Huffman* veröffentlicht.
- sehr schnell wegen mathematischer Einfachheit beliebt.
- generiert präfixfreien Code

Eigenschaften eines optimalen Codes C

- $P_j > P_k \implies l_j \leq l_k$
- a_i und a_j zwei Symbole mit den kleinsten Auftretenswahrscheinlichkeiten.
  Dann : $l_i = l_j$,
  $C(a_i)$ und $C(a_j)$ bis auf das letzte Bit identisch

Das Huffman-Verfahren

Huffman-Baum

- binärer Baum, Knoten mit Gewichten.

Das Huffman-Verfahren

Huffman-Baum

- binärer Baum, Knoten mit Gewichten.

Algorithmus: Generierung des Huffman-Baumes

Sei L eine leere Liste.

- Für jedes a_i, ein Blatt mit Gewicht p_i
- wiederhole,bis L nur einen Knoten enthält.
  - Sortiere L absteigend nach den Gewichten
  - wähle 2 Knoten u, v mit kleinsten Gewichten p(u) und p(v)
  - Erzeuge w mit Gewicht p(w) = p(u) + p(v), w Vater von u und v.
  - Entferne u und v aus L und füge w in L ein.

Beispiel: Huffman-Baum

Example

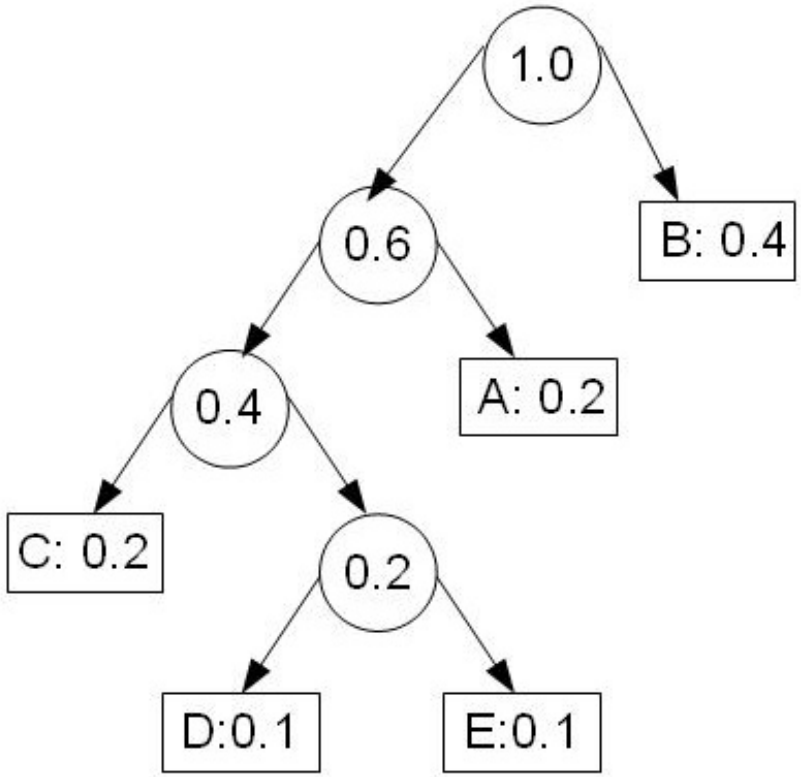
a_i	p_i
A	0.2
B	0.4
C	0.2
D	0.1
E	0.1

Beispiel: Huffman-Baum

Example

a_i	p_i
A	0.2
B	0.4
C	0.2
D	0.1
E	0.1

Huffman-Baum



Code-Generierung

Code-Generierung

- Für jeden inneren Knoten u, markiere die linke Kante aus u mit 0 und die rechte mit 1
- Codewort eines Symbols als Folge der Kantenmarkierungen von der Wurzel bis zum Blatt.

Code-Generierung

Code-Generierung

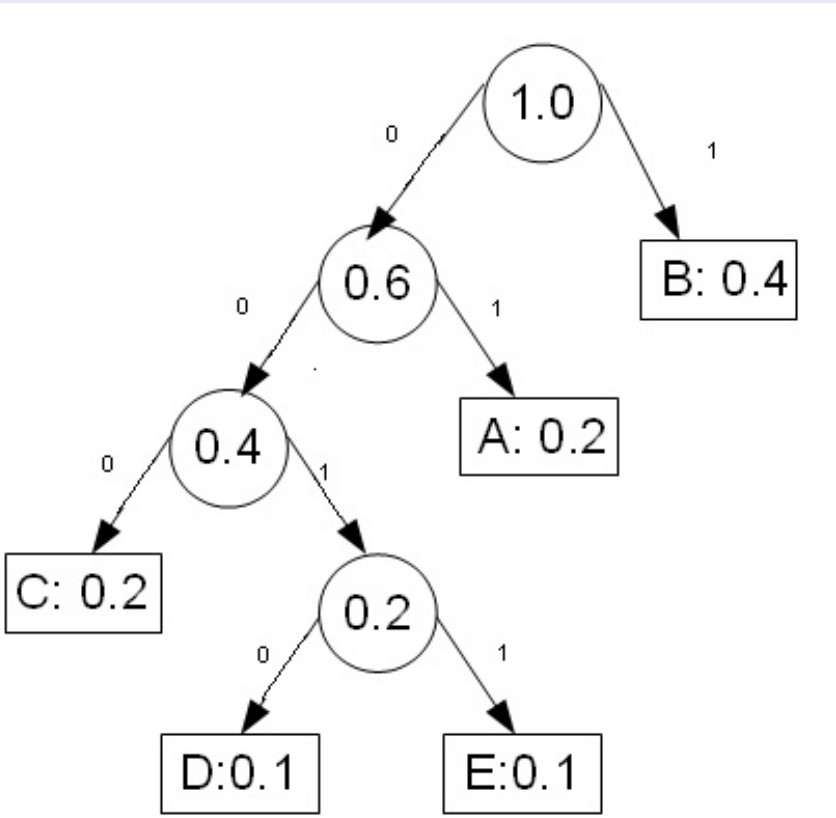
- Für jeden inneren Knoten u, markiere die linke Kante aus u mit 0 und die rechte mit 1
- Codewort eines Symbols als Folge der Kantenmarkierungen von der Wurzel bis zum Blatt.

Bemerkung

- Huffman-Code besitzt die obigen Eigenschaften eines optimalen Codes.

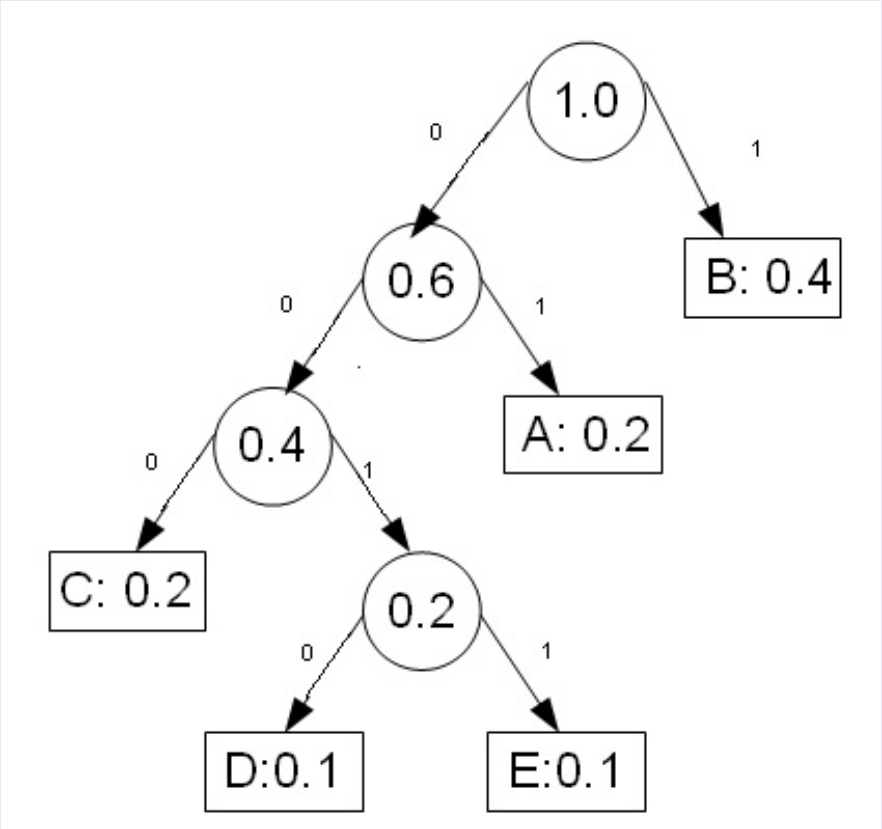
Beispiel: Huffman-Code

markierter Huffman-Baum



Beispiel: Huffman-Code

markierter Huffman-Baum



Code-Tabelle

a_i	p_i	$C(a_i)$
A	0.2	01
B	0.4	1
C	0.2	000
D	0.1	0010
E	0.1	0011

Analyse des Huffman-Codes

Seien C_H ein Huffman-Code mit mittlerer Länge L_H und C ein präfixfreier Code mit mittlerer Länge L.

Optimalität-Eigenschaft

- C_H ist optimal.
- $L_H \leq L$

Analyse des Huffman-Codes

Seien C_H ein Huffman-Code mit mittlerer Länge L_H und C ein präfixfreier Code mit mittlerer Länge L.

Optimalität-Eigenschaft

- C_H ist optimal.
- $L_H \leq L$

Schranken für L_H

$H(p) \leq L_H < H(p) + 1$

Bessere Annäherung der Entropie

Betrachte A^m als Alphabet mit p^m .
C_H ein Huffman-Code für A^m mit L_m. Dann gilt:

Bessere Annäherung der Entropie

Betrachte A^m als Alphabet mit p^m .
C_H ein Huffman-Code für A^m mit L_m. Dann gilt:

Schranken für L_m

$$H(p^m) \leq L_m < H(p^m) + 1$$

Bessere Annäherung der Entropie

Betrachte A^m als Alphabet mit p^m .
C_H ein Huffman-Code für A^m mit L_m. Dann gilt:

Schranken für L_m

$$H(p^m) \leq L_m < H(p^m) + 1$$

- $H(p^m) = m \cdot H(p)$ und $L_H = \frac{L_m}{m}$

Bessere Annäherung der Entropie

Betrachte A^m als Alphabet mit p^m .
C_H ein Huffman-Code für A^m mit L_m. Dann gilt:

Schranken für L_m

$$H(p^m) \leq L_m < H(p^m) + 1$$

- $H(p^m) = m \cdot H(p)$ und $L_H = \frac{L_m}{m}$

Schranken für L_H

$$H(p) \leq L_H < H(p) + \frac{1}{m}$$

Arithmetische Codierung

Erinnerung: Schranken für L_H

$$H(p) \leq L_H < H(p) + \frac{1}{m}$$

Arithmetische Codierung

Erinnerung: Schranken für L_H

$$H(p) \leq L_H < H(p) + \frac{1}{m}$$

Nachteile

- Codewörter aller Sequenzen der Länge m.
- exponentieller Wachstum der Codetabelle mit wachsender m
- Beispiel : $m = 10$ und $|A| = 10 \implies 10^{10}$ Codewörter

Arithmetische Codierung

Erinnerung: Schranken für L_H

$$H(p) \leq L_H < H(p) + \frac{1}{m}$$

Nachteile

- Codewörter aller Sequenzen der Länge m.
- exponentieller Wachstum der Codetabelle mit wachsender m
- Beispiel : $m = 10$ und $|A| = 10 \implies 10^{10}$ Codewörter

Lösung : Arithmetische Codierung

- verlustfrei
- Codewort für eine ganze Sequenz

Codierung als reelle Zahl

Ausgangspunkt

- Datenquelle über $A = \{a_1, \ldots, a_n\}$ und $p = (p_1, \ldots, p_n)$

Codierung als reelle Zahl

Ausgangspunkt

- Datenquelle über $A = \{a_1, \ldots, a_n\}$ und $p = (p_1, \ldots, p_n)$

Definiere folgende Funktion F

$$F : \{0, 1, \ldots, n\} \to [0, 1],\ F(i) = \begin{cases} 0 & \text{wenn } i = 0 \\ \sum_{j=1}^{i} p_j & \text{sonst} \end{cases}$$

Codierung als reelle Zahl

Ausgangspunkt

- Datenquelle über $A = \{a_1, \ldots, a_n\}$ und $p = (p_1, \ldots, p_n)$

Definiere folgende Funktion F

$$F : \{0, 1, \ldots, n\} \to [0, 1],\ F(i) = \begin{cases} 0 & \text{wenn } i = 0 \\ \sum_{j=1}^{i} p_j & \text{sonst} \end{cases}$$

Eigenschaften von F

- $F(0) = 0$, $F(n) = 1$
- $\nexists i \in \{1, \ldots, n\} : p_i = 0 \implies \forall i, j \in \{0, 1, \ldots, n\}, j \neq i :$ $F(i) \neq F(j)$.

Codierung als reelle Zahl

Prinzip der arithmetischen Codierung

- $s \in A^* \xrightarrow{AC} [l_s, u_s) \subseteq [0, 1)$
- $[l_s, u_s)$ eindeutig für s
- $T(s) = \frac{l_s + u_s}{2}$ als Code

Codierung als reelle Zahl

Vorgehen

- Init: $l = 0$ und $u = 1$
- wiederhole, bis ganze Sequenz bearbeitet
  - partitioniere $[l, u)$:
    $\forall a_i \in A \implies [l + (u - l) \cdot F(i - 1), l + (u - l) \cdot F(i))$

Codierung als reelle Zahl

Vorgehen

- Init: $l = 0$ und $u = 1$
- wiederhole, bis ganze Sequenz bearbeitet
  - partitioniere $[l, u)$:
    $\forall a_i \in A \implies [l + (u - l) \cdot F(i - 1), l + (u - l) \cdot F(i))$
  - a_j, $j - te$ Symbol der Sequenz. Setze:
    $l = l + (u - l) \cdot F(j - 1)$ und $u = l + (u - l) \cdot F(j)$
- $T(s) = \frac{l+u}{2}$

Codierung als reelle Zahl

Vorgehen

- Init: $l = 0$ und $u = 1$
- wiederhole, bis ganze Sequenz bearbeitet
  - partitioniere $[l, u)$:
    $\forall a_i \in A \implies [l + (u - l) \cdot F(i - 1), l + (u - l) \cdot F(i))$
  - a_j, $j - te$ Symbol der Sequenz. Setze:
    $l = l + (u - l) \cdot F(j - 1)$ und $u = l + (u - l) \cdot F(j)$
- $T(s) = \frac{l+u}{2}$

Example

siehe Projektor..

binärer arithmetischer Code

Definition

seien $s \in A^*$ mit Wahrscheinlichkeit $p(s)$ und x, y zwei Zahlen.

- $l(s) := \lceil log_2(\frac{1}{p(s)}) \rceil + 1$
- $\lfloor bin(x) \rfloor_y :=$ erste y-Bits der binären Darstellung von x.

binärer arithmetischer Code

Definition

seien $s \in A^*$ mit Wahrscheinlichkeit $p(s)$ und x, y zwei Zahlen.

- $l(s) := \lceil log_2(\frac{1}{p(s)}) \rceil + 1$
- $\lfloor bin(x) \rfloor_y :=$ erste y-Bits der binären Darstellung von x.

binärer arithmetischer Code

$C(s) = \lfloor bin(T(s)) \rfloor_{l(s)}$

Beispiel: binärer arithmetischer Code

Example

Für $s = abc$ hatten wir T(s) = 0.553.

Mit $p(s) = p(a) \cdot P(b) \cdot p(c) = 0.014$ erhalten wir:

$$l(s) = \lceil log_2(\frac{1}{0.014}) \rceil + 1 = 8.$$

$$T(s) = (0.553)_{10} = (0.100011011001000\ldots)_2.$$

$$C(s) = \lfloor bin(0.553) \rfloor_8 = 10001101.$$

Decodierung einer reellen Zahl

Ausgangspunkt

$A = \{a_1, \ldots, a_n\}$, $p = (p_1, \ldots, p_n)$
s eine Sequenz mit Codewort T(s) der Länge k.

Technik

Imitiere den Codierer.

Decodierung einer reellen Zahl

Algorithmus

$l = 0$; $u = 1$; s= ;
For i = 1 To k Do
finde a_{j_i} so, dass für das zugehörige Teilintervall [a, b) $\subseteq$ [l , u) gilt: $T(s) \in [a, b)$.
$s = s \cdot a_{j_i}$;
$l = a$; u = b;
End

Decodierung einer reellen Zahl

Algorithmus

$l = 0$; $u = 1$; s= ;
For i = 1 To k Do
finde a_{j_i} so, dass für das zugehörige Teilintervall [a, b) $\subseteq$ [l , u) gilt: $T(s) \in [a, b)$.
$s = s \cdot a_{j_i}$;
$l = a$; u = b;
End

Example

siehe Projektor...

Terminierung

zwei Möglichkeiten zur Terminierung.

- Länge der Sequenz bekannt
- *end-of-transmission Symbol*

Analyse des arithmetischen Codes

$s \in A^*$ mit Codewort $C(s)$ der Länge $l(s)$

Eindeutigkeit und eindeutige Decodierbarkeit

- C(s) identifiziert s eindeutig
- C(s) präfixfrei $\Longrightarrow$ C(s) eindeutig decodierbar

Analyse des arithmetischen Codes

$s \in A^*$ mit Codewort $C(s)$ der Länge $l(s)$

Eindeutigkeit und eindeutige Decodierbarkeit

- C(s) identifiziert s eindeutig
- C(s) präfixfrei $\Longrightarrow$ C(s) eindeutig decodierbar

mittlere Länge L_{A^m} eines arithmetischen Codes

- $L_{A^m} = \sum_{s \in A^m} p(s) \cdot l(s)$
- Es gilt: $L_A = \frac{L_{A^m}}{m}$

Analyse des arithmetischen Codes

Schranken für L_{A^m}

$$H(p^m) \leq L_{A^m} < H(p^m) + 2$$

Analyse des arithmetischen Codes

Schranken für L_{A^m}

$$H(p^m) \leq L_{A^m} < H(p^m) + 2$$

Schranken für L_A

$$H(p) \leq L_A < H(p) + \frac{2}{m}$$

arithmetische Codierung ist asymptotisch optimal.

Skalierungsfunktionen

Motivation

- Inkrementelle De-/Codierung
- Endliche Genauigkeit von Rechnern

Skalierungsfunktionen

Motivation

- Inkrementelle De-/Codierung
- Endliche Genauigkeit von Rechnern

Für kleines $[l, u)$ unterscheidet man:

- $[l, u) \subseteq [0, 0.5)$
- $[l, u) \subseteq [0.5, 1)$
- $[l, u) \subseteq [0.25, 0.75)$

Skalierungsfunktionen

Skalierungsfunktionen E_1 und E_2

$E_1 : [0, 0.5) \rightarrow [0, 1), E_1(x) = 2 \cdot x$
$E_2 : [0.5, 1) \rightarrow [0, 1) , E_2(x) = 2 \cdot (x - 0.5)$

Skalierungsfunktionen

Skalierungsfunktionen E_1 und E_2

$E_1 : [0, 0.5) \rightarrow [0, 1), E_1(x) = 2 \cdot x$
$E_2 : [0.5, 1) \rightarrow [0, 1) , E_2(x) = 2 \cdot (x - 0.5)$

Die Skalierungsfunktion E_3

$E_3 : [0.25, 0.75) \rightarrow [0, 1), E_3(x) = 2 \cdot (x - 0.25)$

Skalierungsfunktionen

Skalierungsfunktionen E_1 und E_2

$E_1 : [0, 0.5) \rightarrow [0, 1), E_1(x) = 2 \cdot x$
$E_2 : [0.5, 1) \rightarrow [0, 1) , E_2(x) = 2 \cdot (x - 0.5)$

Die Skalierungsfunktion E_3

$E_3 : [0.25, 0.75) \rightarrow [0, 1), E_3(x) = 2 \cdot (x - 0.25)$

Eigenschaften von E_3

- $E_1 \circ (E_3)^n = (E_2)^n \circ E_1$
- $E_2 \circ (E_3)^n = (E_1)^n \circ E_2$

Beispiel: Codierung mit Skalierung

Example

Siehe Projektor...

Huffman- vs arithmetische Codierung

Laufzeit

- Huffman-Verfahren schneller
- arithmetische Codierung rechenaufwändig

Huffman- vs arithmetische Codierung

Laufzeit

- Huffman-Verfahren schneller
- arithmetische Codierung rechenaufwändig

Effizienz

Schranken für die mittlere Länge

- eines Huffman-Codes für Sequenzen der Länge m:

$$H(p) \leq L_H < H(p) + \frac{1}{m}$$

- eines arithmetischen Codes:

$$H(p) \leq L_A < H(p) + \frac{2}{m}$$

Anwendungsfelder

Huffman-Codierung

- Textkomprimierung
- Fax-Übertragung
- ZIP, GZIP, JPEG, MP3,...

Anwendungsfelder

Huffman-Codierung

- Textkomprimierung
- Fax-Übertragung
- ZIP, GZIP, JPEG, MP3,...

Arithmetische Codierung

- Textkomprimierung, JPEG
- Gültige Patente hindern verbreiteten Einsatz