

Vorlesung

Effiziente Algorithmen und Komplexitätstheorie

Sommersemester 2008

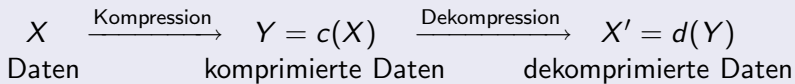
Ingo Wegener; Vertretung: Carsten Witt

7. Juli 2008

Vorlesung am 14.07. (nächste Woche): Raum E04/E05

Datenkompression: Definition und Motivation

Aufgabe: Verkürze Repräsentation von Daten



Datenkompression: Definition und Motivation

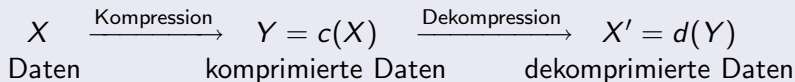
Aufgabe: Verkürze Repräsentation von Daten

$$\begin{array}{ccccc} X & \xrightarrow{\text{Kompression}} & Y = c(X) & \xrightarrow{\text{Dekompression}} & X' = d(Y) \\ \text{Daten} & & \text{komprimierte Daten} & & \text{dekomprimierte Daten} \end{array}$$

Brauchen wir heutzutage noch Datenkompression?

Datenkompression: Definition und Motivation

Aufgabe: Verkürze Repräsentation von Daten



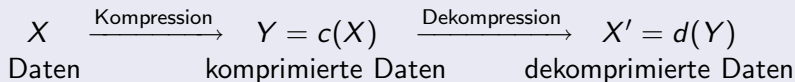
Brauchen wir heutzutage noch Datenkompression?

Anschein: Speicher wächst viel langsamer als Datenmengen

Fakt: Datenkompression allgegenwärtig

Datenkompression: Definition und Motivation

Aufgabe: Verkürze Repräsentation von Daten



Brauchen wir heutzutage noch Datenkompression?

Anschein: Speicher wächst viel langsamer als Datenmengen

Fakt: Datenkompression allgegenwärtig

Klassifikation

verlustfrei (lossless)

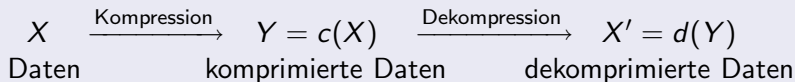
- $\forall X: d(c(X)) = X$
- z. B. bzip2, gif, zip

verlustbehaftet (lossy)

- i. Allg. $d(c(X)) \neq X$
- z. B. jpeg, mpeg, mp3

Datenkompression: Definition und Motivation

Aufgabe: Verkürze Repräsentation von Daten



Brauchen wir heutzutage noch Datenkompression?

Anschein: Speicher wächst viel langsamer als Datenmengen

Fakt: Datenkompression allgegenwärtig

Klassifikation

verlustfrei (lossless)

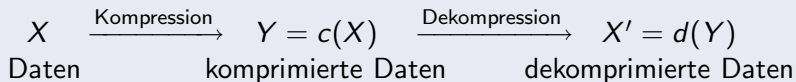
- $\forall X: d(c(X)) = X$
- z. B. bzip2, gif, zip

verlustbehaftet (lossy)

- i. Allg. $d(c(X)) \neq X$
- z. B. jpeg, mpeg, mp3

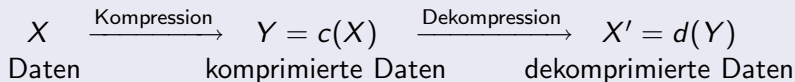
Hier: nur verlustfreie Kompression

Güte von Kompressionen



Konventionen: Daten $X \in \Sigma^*$ für endliches Alphabet Σ
 Komprimierte Daten $c(X) \in \{0, 1\}^+$

Güte von Kompressionen

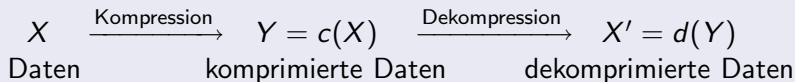


Konventionen: Daten $X \in \Sigma^*$ für endliches Alphabet Σ
 Komprimierte Daten $c(X) \in \{0, 1\}^+$

Maße

- Kompressionsrate = $\frac{|c(X)|}{|X|}$ (Einheit [bpb], „bits per bit“)
- Kompressionsfaktor = $\frac{1}{\text{Kompressionsrate}}$

Güte von Kompressionen



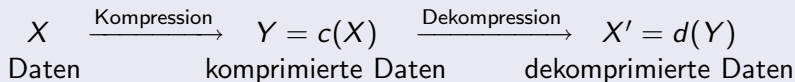
Konventionen: Daten $X \in \Sigma^*$ für endliches Alphabet Σ
 Komprimierte Daten $c(X) \in \{0, 1\}^+$

Maße

- Kompressionsrate = $\frac{|c(X)|}{|X|}$ (Einheit [bpb], „bits per bit“)
- Kompressionsfaktor = $\frac{1}{\text{Kompressionsrate}}$

Schranken für die Maße?

Güte von Kompressionen



Konventionen: Daten $X \in \Sigma^*$ für endliches Alphabet Σ
 Komprimierte Daten $c(X) \in \{0, 1\}^+$

Maße

- Kompressionsrate = $\frac{|c(X)|}{|X|}$ (Einheit [bpb], „bits per bit“)
- Kompressionsfaktor = $\frac{1}{\text{Kompressionsrate}}$

Schranken für die Maße?

Im Allgemeinen nicht: $c(X) = 1$ für beliebig lange X möglich

Aber: für jedes verlustfreie Verfahren gibt es auch X mit $|c(X)| \geq |X|$,
 da c **injektiv** und $c(X) \in \{0, 1\}^+$.

Präfixkodes: Repräsentation komprimierter Daten

Ansatz hier: Kodiere Texte $X \in \Sigma^*$ buchstabenweise

Zunächst $c: \Sigma \rightarrow \{0, 1\}^+$, also Code für jeden Buchstaben

Präfixkodes: Repräsentation komprimierter Daten

Ansatz hier: Kodiere Texte $X \in \Sigma^*$ buchstabenweise

Zunächst $c: \Sigma \rightarrow \{0, 1\}^+$, also Code für jeden Buchstaben

Dann setze c für Texte $X = x_1x_2 \dots x_\ell$ fort:

$c(x_1x_2 \dots x_\ell) = c(x_1)c(x_2) \dots c(x_\ell)$ **ohne Trennzeichen**

Nötig: eindeutige Dekodierbarkeit

Präfixkodes: Repräsentation komprimierter Daten

Ansatz hier: Kodiere Texte $X \in \Sigma^*$ buchstabenweise

Zunächst $c: \Sigma \rightarrow \{0, 1\}^+$, also Code für jeden Buchstaben

Dann setze c für Texte $X = x_1x_2 \dots x_\ell$ fort:

$c(x_1x_2 \dots x_\ell) = c(x_1)c(x_2) \dots c(x_\ell)$ **ohne Trennzeichen**

Nötig: eindeutige Dekodierbarkeit

Lösung: Präfixkodes

Definition

Für $x \in \{0, 1\}^*$ sei $\text{PRE}(x) := \{x' \in \{0, 1\}^* \mid x' \sqsubset x\}$ die Menge aller Präfixe von x . Eine Kodierung $c: \Sigma^* \rightarrow \{0, 1\}^+$ heißt

Präfixkode, wenn

$$\forall Y \neq X: c(X) \notin \text{PRE}(c(Y))$$

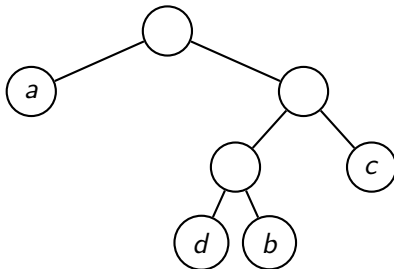
gilt.

Beschreibung von Präfixkodes

Eindeutige Darstellung von Präfixkodes: Binärbaum mit $|\Sigma|$ Blättern

$$\Sigma = \{a, b, c, d\}$$

$s \in \Sigma$	$c(s)$
a	0
b	101
c	11
d	100

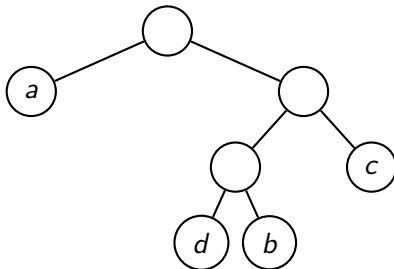


Beschreibung von Präfixkodes

Eindeutige Darstellung von Präfixkodes: Binärbaum mit $|\Sigma|$ Blättern

$$\Sigma = \{a, b, c, d\}$$

$s \in \Sigma$	$c(s)$
a	0
b	101
c	11
d	100



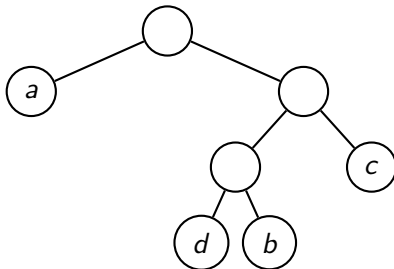
Wann existieren Präfixkodes überhaupt?

Beschreibung von Präfixkodes

Eindeutige Darstellung von Präfixkodes: Binärbaum mit $|\Sigma|$ Blättern

$$\Sigma = \{a, b, c, d\}$$

$s \in \Sigma$	$c(s)$
a	0
b	101
c	11
d	100



Wann existieren Präfixkodes überhaupt?

Theorem (Ungleichung von Kraft/McMillan)

Ein Präfixkode $c: \{s_1, s_2, \dots, s_n\} \rightarrow \{0, 1\}^+$ mit $|c(s_1)| = \ell_1, |c(s_2)| = \ell_2, \dots, |c(s_n)| = \ell_n$ existiert genau dann, wenn $\sum_{i=1}^n 2^{-\ell_i} \leq 1$ gilt.

Beweis der Ungleichung von Kraft/McMillan (\Rightarrow)

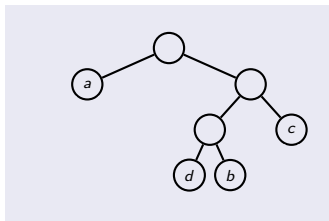
Sei **Präfixkode** $c: \Sigma \rightarrow \{0, 1\}^+$ mit $\ell_i := |c(s_i)|$ gegeben.

Beweis der Ungleichung von Kraft/McMillan (\Rightarrow)

Sei **Präfixkode** $c: \Sigma \rightarrow \{0, 1\}^+$ mit $l_i := |c(s_i)|$ gegeben.

Sei $l := \max\{l_1, \dots, l_n\}$.

Vervollständige Binärbaum für c



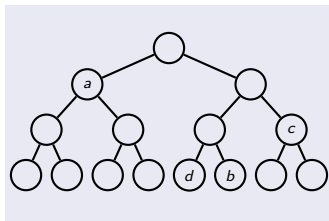
Beweis der Ungleichung von Kraft/McMillan (\Rightarrow)

Sei **Präfixkode** $c: \Sigma \rightarrow \{0, 1\}^+$ mit $\ell_i := |c(s_i)|$ gegeben.

Sei $\ell := \max\{\ell_1, \dots, \ell_n\}$.

Vervollständige Binärbaum für c

→ Binärbaum mit Höhe ℓ
und 2^ℓ Blättern



Darin

- **betrachte** Teilbäume mit Wurzeln s_1, \dots, s_n ,
- **beobachte:** Teilbaum mit Wurzel s_i hat $2^{\ell-\ell_i}$ Blätter.

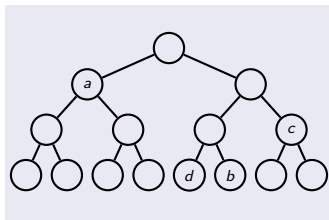
Beweis der Ungleichung von Kraft/McMillan (\Rightarrow)

Sei **Präfixkode** $c: \Sigma \rightarrow \{0, 1\}^+$ mit $\ell_i := |c(s_i)|$ gegeben.

Sei $\ell := \max\{\ell_1, \dots, \ell_n\}$.

Vervollständige Binärbaum für c

→ Binärbaum mit Höhe ℓ
und 2^ℓ Blättern



Darin

- **betrachte** Teilbäume mit Wurzeln s_1, \dots, s_n ,
- **beobachte:** Teilbaum mit Wurzel s_i hat $2^{\ell-\ell_i}$ Blätter.

Also: $\sum_{i=1}^n 2^{\ell-\ell_i} = 2^\ell \Rightarrow \sum_{i=1}^n 2^{-\ell_i} = 1.$



Beweis der Ungleichung von Kraft/McMillan (\Leftrightarrow)

Gegeben: l_1, \dots, l_n mit $S := \sum_{i=1}^n 2^{-l_i} \leq 1$

Ziel: Präfixkode

Beweis der Ungleichung von Kraft/McMillan (\Leftrightarrow)

Gegeben: l_1, \dots, l_n mit $S := \sum_{i=1}^n 2^{-l_i} \leq 1$

Ziel: Präfixkode

O. B. d. A.: $l_1 \leq l_2 \leq \dots \leq l_n =: l.$

Beweis der Ungleichung von Kraft/McMillan (\Leftrightarrow)

Gegeben: l_1, \dots, l_n mit $S := \sum_{i=1}^n 2^{-l_i} \leq 1$

Ziel: Präfixkode

O. B. d. A.: $l_1 \leq l_2 \leq \dots \leq l_n =: l$.

Induktion über l

- $l = 1$: Dann $n \leq 2$ und 0 bzw. $0,1$ sind passende Präfixkodes.

Beweis der Ungleichung von Kraft/McMillan (\Leftrightarrow)

Gegeben: l_1, \dots, l_n mit $S := \sum_{i=1}^n 2^{-l_i} \leq 1$

Ziel: Präfixkode

O. B. d. A.: $l_1 \leq l_2 \leq \dots \leq l_n =: l$.

Induktion über l

- $l = 1$: Dann $n \leq 2$ und 0 bzw. $0, 1$ sind passende Präfixkodes.
- $l > 1$: **Fallunterscheidung** nach S
 - $S \leq 1/2$: Entweder $l_1 = 1$, dann aber $l = 1$ und Präfixkode trivial.
Oder $l_1 \geq 2$, dann benutze $\sum_{i=1}^n 2^{-(l_i-1)} = 2S \leq 1$,
wende IV auf $l_1 - 1, l_2 - 1, \dots, l_n - 1$ an
und hänge vor die erhaltenen Kodes je eine 0 .
→ **Präfixkode** mit gewünschten Längen.

Beweis der Ungleichung von Kraft/McMillan (\Leftrightarrow)

Gegeben: l_1, \dots, l_n mit $S := \sum_{i=1}^n 2^{-l_i} \leq 1$

Ziel: Präfixkode

O. B. d. A.: $l_1 \leq l_2 \leq \dots \leq l_n =: l$.

Induktion über l

- $l = 1$: Dann $n \leq 2$ und 0 bzw. $0, 1$ sind passende Präfixkodes.
- $l > 1$: **Fallunterscheidung** nach S
 - $S \leq 1/2$: Entweder $l_1 = 1$, dann aber $l = 1$ und Präfixkode trivial.
Oder $l_1 \geq 2$, dann benutze $\sum_{i=1}^n 2^{-(l_i-1)} = 2S \leq 1$, wende IV auf $l_1 - 1, l_2 - 1, \dots, l_n - 1$ an und hänge vor die erhaltenen Kodes je eine 0 .
→ **Präfixkode** mit gewünschten Längen.
 - $S > 1/2$: **Beobachtung:** $2^{-l_1} \geq 2^{-l_2} \geq \dots \geq 2^{-l_n}$,
Definiere $m := \min\{k \mid \sum_{i=1}^k 2^{-l_i} \geq 1/2\}$
Beobachtung: $\sum_{i=1}^m 2^{-l_i} = 1/2$,
da $\sum_{i=1}^{m-1} 2^{-l_i} \leq 1/2 - 2^{-l_m}$
IV anwenden auf $\{l_1, \dots, l_m\}$ **und** $\{l_{m+1}, \dots, l_n\}$,
gib erhaltenen **Präfixkodes** neue Präfixe (0 bzw. 1).

Informationstheorie

Hintergrund: probabilistisches Modell

Quelle erzeugt unabh. Ereignisse, **hier** Buchstaben $s \in \Sigma = \{s_1, \dots, s_n\}$
mit Auftrittswahrscheinlichkeiten $\text{Prob}(s)$ für $s \in \Sigma$.

Informationstheorie

Hintergrund: probabilistisches Modell

Quelle erzeugt unabh. Ereignisse, **hier** Buchstaben $s \in \Sigma = \{s_1, \dots, s_n\}$
mit Auftrittswahrscheinlichkeiten $\text{Prob}(s)$ für $s \in \Sigma$.

Selbstinformation für Ereignis s ist $i(s) = \log \frac{1}{\text{Prob}(s)}$,
mindestens 1 für $|\Sigma| \geq 2$.

Informationstheorie

Hintergrund: probabilistisches Modell

Quelle erzeugt unabh. Ereignisse, **hier** Buchstaben $s \in \Sigma = \{s_1, \dots, s_n\}$ mit Auftrittswahrscheinlichkeiten $\text{Prob}(s)$ für $s \in \Sigma$.

Selbstinformation für Ereignis s ist $i(s) = \log \frac{1}{\text{Prob}(s)}$,
mindestens 1 für $|\Sigma| \geq 2$.

Entropie $H(\Sigma) = \sum_{s \in \Sigma} \text{Prob}(s) \cdot i(s) = - \sum_{s \in \Sigma} \text{Prob}(s) \log(\text{Prob}(s))$
als Erwartungswert der Selbstinformation

Informationstheorie

Hintergrund: probabilistisches Modell

Quelle erzeugt unabh. Ereignisse, **hier** Buchstaben $s \in \Sigma = \{s_1, \dots, s_n\}$
mit Auftrittswahrscheinlichkeiten $\text{Prob}(s)$ für $s \in \Sigma$.

Selbstinformation für Ereignis s ist $i(s) = \log \frac{1}{\text{Prob}(s)}$,
mindestens 1 für $|\Sigma| \geq 2$.

Entropie $H(\Sigma) = \sum_{s \in \Sigma} \text{Prob}(s) \cdot i(s) = - \sum_{s \in \Sigma} \text{Prob}(s) \log(\text{Prob}(s))$
als Erwartungswert der Selbstinformation

Satz: durchschnittliche Kodierungslänge \geq Entropie
(Beweis später)

Informationstheorie

Hintergrund: probabilistisches Modell

Quelle erzeugt unabh. Ereignisse, **hier** Buchstaben $s \in \Sigma = \{s_1, \dots, s_n\}$
mit Auftrittswahrscheinlichkeiten $\text{Prob}(s)$ für $s \in \Sigma$.

Selbstinformation für Ereignis s ist $i(s) = \log \frac{1}{\text{Prob}(s)}$,
mindestens 1 für $|\Sigma| \geq 2$.

Entropie $H(\Sigma) = \sum_{s \in \Sigma} \text{Prob}(s) \cdot i(s) = - \sum_{s \in \Sigma} \text{Prob}(s) \log(\text{Prob}(s))$
als Erwartungswert der Selbstinformation

Satz: durchschnittliche Kodierlänge \geq Entropie
(Beweis später)

Problem: im Allgemeinen Entropie unbekannt
→ probabilistisches Modell aufstellen oder schätzen

Erzeugung von Präfixkodes aus probabilistischen Modellen

Einige Eigenschaften

- Präfixkode gegeben: Kodierung und Dekodierung in Linearzeit
- gegenüber fester Kodierung:
Gewinn durch kurze Kodierung häufiger Zeichen
- bei „rein zufälligen“ Texten (Gleichverteilung über Σ) sinnlos

Erzeugung von Präfixkodes aus probabilistischen Modellen

Einige Eigenschaften

- Präfixkode gegeben: Kodierung und Dekodierung in Linearzeit
- gegenüber fester Kodierung:
Gewinn durch kurze Kodierung häufiger Zeichen
- bei „rein zufälligen“ Texten (Gleichverteilung über Σ) sinnlos

Annahmen: Zeichen $s_i \in \Sigma$ **unabhängig**, $\text{Prob}(s_i)$ bekannt

Erwartete Länge eines Kodeworts c : $L_c := \sum_{i=1}^n \text{Prob}(s_i) \cdot |c(s_i)|$

Damit erwartete Kodierungslänge eines Textes X : $|X| \cdot L_c$

Erzeugung von Präfixkodes aus probabilistischen Modellen

Einige Eigenschaften

- Präfixkode gegeben: Kodierung und Dekodierung in Linearzeit
- gegenüber fester Kodierung:
Gewinn durch kurze Kodierung häufiger Zeichen
- bei „rein zufälligen“ Texten (Gleichverteilung über Σ) sinnlos

Annahmen: Zeichen $s_i \in \Sigma$ **unabhängig**, $\text{Prob}(s_i)$ bekannt

Erwartete Länge eines Kodeworts c : $L_c := \sum_{i=1}^n \text{Prob}(s_i) \cdot |c(s_i)|$

Damit erwartete Kodierungslänge eines Textes X : $|X| \cdot L_c$

Gesucht: optimaler Präfixkode c , d. h. mit minimalem L_c

Erzeugung von Präfixkodes aus probabilistischen Modellen

Einige Eigenschaften

- Präfixkode gegeben: Kodierung und Dekodierung in Linearzeit
- gegenüber fester Kodierung:
Gewinn durch kurze Kodierung häufiger Zeichen
- bei „rein zufälligen“ Texten (Gleichverteilung über Σ) sinnlos

Annahmen: Zeichen $s_i \in \Sigma$ **unabhängig**, $\text{Prob}(s_i)$ bekannt

Erwartete Länge eines Kodeworts c : $L_c := \sum_{i=1}^n \text{Prob}(s_i) \cdot |c(s_i)|$

Damit erwartete Kodierungslänge eines Textes X : $|X| \cdot L_c$

Gesucht: optimaler Präfixkode c , d. h. mit minimalem L_c

Im Folgenden: drei Ansätze für kurze Präfixkodes

Shannon-Algorithmus

Notation: $p_i := \text{Prob}(s_i)$, $P_i := \sum_{j=1}^{i-1} p_j$

O. B. d. A.: $p_1 \geq p_2 \geq \dots \geq p_n$

Shannon-Algorithmus

Notation: $p_i := \text{Prob}(s_i)$, $P_i := \sum_{j=1}^{i-1} p_j$

O. B. d. A.: $p_1 \geq p_2 \geq \dots \geq p_n$

Shannon-Algorithmus (1948)

1. Für $i = 1, \dots, n$
2. $\ell_i := \lceil -\log p_i \rceil$
3. $c(s_i) := b_1 b_2 \dots b_{\ell_i}$ nach Binärdarstellung für $P_i = 0, b_1 b_2 b_3 \dots$

Shannon-Algorithmus

Notation: $p_i := \text{Prob}(s_i)$, $P_i := \sum_{j=1}^{i-1} p_j$

O. B. d. A.: $p_1 \geq p_2 \geq \dots \geq p_n$

Shannon-Algorithmus (1948)

1. Für $i = 1, \dots, n$
2. $\ell_i := \lceil -\log p_i \rceil$
3. $c(s_i) := b_1 b_2 \dots b_{\ell_i}$ nach Binärdarstellung für $P_i = 0, b_1 b_2 b_3 \dots$

Beispiel: $\Sigma = \{a, b, c, d, e\}$, Wkten. (0,4; 0,25; 0,15; 0,1; 0,1)

s_i	p_i	ℓ_i	P_i	P_i binär	$c(s_i)$
a	0,4	2	0	0,0000...	00
b	0,25	2	0,4	0,0110...	01
c	0,15	3	0,65	0,1010...	101
d	0,1	4	0,8	0,1100...	1100
e	0,1	4	0,9	0,1110...	1110

Shannon-Algorithmus

Notation: $p_i := \text{Prob}(s_i)$, $P_i := \sum_{j=1}^{i-1} p_j$

O. B. d. A.: $p_1 \geq p_2 \geq \dots \geq p_n$

Shannon-Algorithmus (1948)

1. Für $i = 1, \dots, n$
2. $\ell_i := \lceil -\log p_i \rceil$
3. $c(s_i) := b_1 b_2 \dots b_{\ell_i}$ nach Binärdarstellung für $P_i = 0, b_1 b_2 b_3 \dots$

Beispiel: $\Sigma = \{a, b, c, d, e\}$, Wkten. (0,4; 0,25; 0,15; 0,1; 0,1)

s_i	p_i	ℓ_i	P_i	P_i binär	$c(s_i)$
a	0,4	2	0	0,0000...	00
b	0,25	2	0,4	0,0110...	01
c	0,15	3	0,65	0,1010...	101
d	0,1	4	0,8	0,1100...	1100
e	0,1	4	0,9	0,1110...	1110

Erwartete Kodelänge

$$= 0,4 \cdot 2 + 0,25 \cdot 2 + 0,15 \cdot 3 + 0,1 \cdot 4 + 0,1 \cdot 4 = 2,55$$

korrekt?

Shannon-Algorithmus: Korrektheit

Theorem

Der Shannon-Algorithmus erzeugt einen Präfixkode.

Shannon-Algorithmus: Korrektheit

Theorem

Der Shannon-Algorithmus erzeugt einen Präfixkode.

Beweis

- Algo. setzt $\ell_i := \lceil -\log p_i \rceil$, also $\ell_i \geq \log(1/p_i)$.

Shannon-Algorithmus: Korrektheit

Theorem

Der Shannon-Algorithmus erzeugt einen Präfixkode.

Beweis

- Algo. setzt $\ell_i := \lceil -\log p_i \rceil$, also $\ell_i \geq \log(1/p_i)$.
- Daher: $j > i \Rightarrow P_j - P_i = \sum_{k=i}^{j-1} p_k \geq p_i \geq 2^{-\ell_i}$.

Shannon-Algorithmus: Korrektheit

Theorem

Der Shannon-Algorithmus erzeugt einen Präfixkode.

Beweis

- Algo. setzt $\ell_i := \lceil -\log p_i \rceil$, also $\ell_i \geq \log(1/p_i)$.
- **Daher:** $j > i \Rightarrow P_j - P_i = \sum_{k=i}^{j-1} p_k \geq p_i \geq 2^{-\ell_i}$.
- **Wegen** $p_1 \geq \dots \geq p_n$ **ist** $\ell_1 \leq \dots \leq \ell_n$.
- **Annahme:** kein Präfixkode, dann $\exists i < j$ mit $c(s_i) \sqsubset c(s_j)$.
Sei $c(s_i) = a_1 \dots a_{\ell_i}$ und $c(s_j) = b_1 \dots b_{\ell_j}$.

Shannon-Algorithmus: Korrektheit

Theorem

Der Shannon-Algorithmus erzeugt einen Präfixkode.

Beweis

- Algo. setzt $\ell_i := \lceil -\log p_i \rceil$, also $\ell_i \geq \log(1/p_i)$.
- Daher: $j > i \Rightarrow P_j - P_i = \sum_{k=i}^{j-1} p_k \geq p_i \geq 2^{-\ell_i}$.
- Wegen $p_1 \geq \dots \geq p_n$ ist $\ell_1 \leq \dots \leq \ell_n$.
- **Annahme:** kein Präfixkode, dann $\exists i < j$ mit $c(s_i) \sqsubset c(s_j)$.
Sei $c(s_i) = a_1 \dots a_{\ell_i}$ und $c(s_j) = b_1 \dots b_{\ell_j}$.

- Es folgt

$$\begin{aligned}
 P_j - P_i &= \left(\frac{b_1}{2^1} + \dots + \frac{b_{\ell_i}}{2^{\ell_i}} + \dots \right) - \left(\frac{a_1}{2^1} + \dots + \frac{a_{\ell_i}}{2^{\ell_i}} + \dots \right) \\
 &= \left(\frac{a_1}{2^1} + \dots + \frac{a_{\ell_i}}{2^{\ell_i}} + \frac{b_{\ell_i+1}}{2^{\ell_i+1}} + \dots \right) - \left(\frac{a_1}{2^1} + \dots + \frac{a_{\ell_i}}{2^{\ell_i}} + \dots \right) \\
 &= \left(\frac{b_{\ell_i+1}}{2^{\ell_i+1}} + \dots \right) - \left(\frac{a_{\ell_i+1}}{2^{\ell_i+1}} + \dots \right) < \frac{1}{2^{\ell_i}} \quad \color{red}{\text{⚡}}
 \end{aligned}$$

Shannon-Algorithmus: Güte

Erinnerung: minimale erwartete Kodelänge gesucht,
Entropie untere Schranke

Shannon-Algorithmus: Güte

Erinnerung: minimale erwartete Kodelänge gesucht,
Entropie untere Schranke

Theorem

Der Shannon-Algorithmus berechnet aus einer Quelle Σ einen Kode c mit $L_c^{Shannon} \leq H(\Sigma) + 1$.

Shannon-Algorithmus: Güte

Erinnerung: minimale erwartete Kodelänge gesucht,
Entropie untere Schranke

Theorem

Der Shannon-Algorithmus berechnet aus einer Quelle Σ einen Kode c mit $L_c^{Shannon} \leq H(\Sigma) + 1$.

Beweis

- **Erinnerung:** Kodelänge $c(s_i) = \lceil -\log \text{Prob}(s_i) \rceil$.

Shannon-Algorithmus: Güte

Erinnerung: minimale erwartete Kodelänge gesucht,
Entropie untere Schranke

Theorem

Der Shannon-Algorithmus berechnet aus einer Quelle Σ einen Kode c mit $L_c^{Shannon} \leq H(\Sigma) + 1$.

Beweis

- **Erinnerung:** Kodelänge $c(s_i) = \lceil -\log \text{Prob}(s_i) \rceil$.
- Somit

$$\begin{aligned} L_c &= \sum_{i=1}^n (\text{Prob}(s_i) \lceil -\log(\text{Prob}(s_i)) \rceil) \\ &\leq \sum_{i=1}^n (\text{Prob}(s_i)(-\log(\text{Prob}(s_i)) + 1)) = H(\Sigma) + 1. \quad \checkmark \end{aligned}$$

Shannon-Algorithmus: Güte

Erinnerung: minimale erwartete Kodelänge gesucht,
Entropie untere Schranke

Theorem

Der Shannon-Algorithmus berechnet aus einer Quelle Σ einen Kode c mit $L_c^{Shannon} \leq H(\Sigma) + 1$.

Beweis

- **Erinnerung:** Kodelänge $c(s_i) = \lceil -\log \text{Prob}(s_i) \rceil$.
- Somit

$$\begin{aligned} L_c &= \sum_{i=1}^n (\text{Prob}(s_i) \lceil -\log(\text{Prob}(s_i)) \rceil) \\ &\leq \sum_{i=1}^n (\text{Prob}(s_i)(-\log(\text{Prob}(s_i)) + 1)) = H(\Sigma) + 1. \quad \checkmark \end{aligned}$$

Aber werden sehen: Shannon-Algo. nicht optimal.

Shannon-Fano-Algorithmus

Rekursiv Präfixkodebaum erzeugen

Shannon-Fano-Algorithmus

Rekursiv Präfixcodebaum erzeugen

Eingabe wieder: Σ mit zugehöriger Wahrscheinlichkeitsverteilung

Shannon-Fano-Algorithmus

Rekursiv Präfixcodebaum erzeugen

Eingabe wieder: Σ mit zugehöriger Wahrscheinlichkeitsverteilung

- 1 Falls $|\Sigma| = 1$, erzeuge Baum mit einem Knoten und entsprechender Beschriftung. STOP.

Shannon-Fano-Algorithmus

Rekursiv Präfixkodebaum erzeugen

Eingabe wieder: Σ mit zugehöriger Wahrscheinlichkeitsverteilung

- 1 Falls $|\Sigma| = 1$, erzeuge Baum mit einem Knoten und entsprechender Beschriftung. STOP.
- 2 Teile Σ in Σ_1 und Σ_2 mit $\sum_{s \in \Sigma_1} \text{Prob}(s) \approx \sum_{s \in \Sigma_2} \text{Prob}(s)$.

Shannon-Fano-Algorithmus

Rekursiv Präfixcodebaum erzeugen

Eingabe wieder: Σ mit zugehöriger Wahrscheinlichkeitsverteilung

- 1 Falls $|\Sigma| = 1$, erzeuge Baum mit einem Knoten und entsprechender Beschriftung. STOP.
- 2 Teile Σ in Σ_1 und Σ_2 mit $\sum_{s \in \Sigma_1} \text{Prob}(s) \approx \sum_{s \in \Sigma_2} \text{Prob}(s)$.
- 3 Erzeuge Baum mit unbeschrifteter Wurzel und Shannon-Fano(Σ_1) als linkem sowie Shannon-Fano(Σ_2) als rechtem Teilbaum.

Shannon-Fano-Algorithmus

Rekursiv Präfixcodebaum erzeugen

Eingabe wieder: Σ mit zugehöriger Wahrscheinlichkeitsverteilung

- 1 Falls $|\Sigma| = 1$, erzeuge Baum mit einem Knoten und entsprechender Beschriftung. STOP.
- 2 Teile Σ in Σ_1 und Σ_2 mit $\sum_{s \in \Sigma_1} \text{Prob}(s) \approx \sum_{s \in \Sigma_2} \text{Prob}(s)$.
- 3 Erzeuge Baum mit unbeschrifteter Wurzel und Shannon-Fano(Σ_1) als linkem sowie Shannon-Fano(Σ_2) als rechtem Teilbaum.

Beobachtung: erzeugt Präfixcode

Shannon-Fano-Algorithmus

Rekursiv Präfixcodebaum erzeugen

Eingabe wieder: Σ mit zugehöriger Wahrscheinlichkeitsverteilung

- 1 Falls $|\Sigma| = 1$, erzeuge Baum mit einem Knoten und entsprechender Beschriftung. STOP.
- 2 Teile Σ in Σ_1 und Σ_2 mit $\sum_{s \in \Sigma_1} \text{Prob}(s) \approx \sum_{s \in \Sigma_2} \text{Prob}(s)$.
- 3 Erzeuge Baum mit unbeschrifteter Wurzel und Shannon-Fano(Σ_1) als linkem sowie Shannon-Fano(Σ_2) als rechtem Teilbaum.

Beobachtung: erzeugt Präfixcode

Probleme: algorithmisch schwierig (Partitionierung), mehrdeutig

Shannon-Fano-Algorithmus: Beispiel

Wie oben: $\Sigma = \{a, b, c, d, e\}$, Wkten. (0,4; 0,25; 0,15; 0,1; 0,1)

- $\Sigma_1^1 = \{a, d\}$, $\Sigma_2^1 = \{b, c, e\}$

Shannon-Fano-Algorithmus: Beispiel

Wie oben: $\Sigma = \{a, b, c, d, e\}$, Wkten. (0,4; 0,25; 0,15; 0,1; 0,1)

- $\Sigma_1^1 = \{a, d\}$, $\Sigma_2^1 = \{b, c, e\}$
- $\Sigma_1^2 = \{a\}$, $\Sigma_2^2 = \{d\}$

Shannon-Fano-Algorithmus: Beispiel

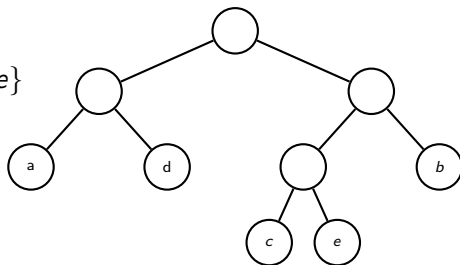
Wie oben: $\Sigma = \{a, b, c, d, e\}$, Wkten. (0,4; 0,25; 0,15; 0,1; 0,1)

- $\Sigma_1^1 = \{a, d\}$, $\Sigma_2^1 = \{b, c, e\}$
- $\Sigma_1^2 = \{a\}$, $\Sigma_2^2 = \{d\}$
- $\Sigma_1^3 = \{c, e\}$, $\Sigma_2^3 = \{b\}$

Shannon-Fano-Algorithmus: Beispiel

Wie oben: $\Sigma = \{a, b, c, d, e\}$, Wkten. (0,4; 0,25; 0,15; 0,1; 0,1)

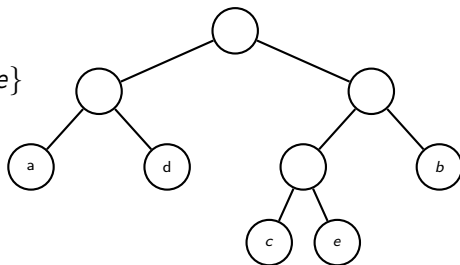
- $\Sigma_1^1 = \{a, d\}$, $\Sigma_2^1 = \{b, c, e\}$
- $\Sigma_1^2 = \{a\}$, $\Sigma_2^2 = \{d\}$
- $\Sigma_1^3 = \{c, e\}$, $\Sigma_2^3 = \{b\}$
- $\Sigma_1^4 = \{c\}$, $\Sigma_2^4 = \{e\}$



Shannon-Fano-Algorithmus: Beispiel

Wie oben: $\Sigma = \{a, b, c, d, e\}$, Wkten. (0,4; 0,25; 0,15; 0,1; 0,1)

- $\Sigma_1^1 = \{a, d\}$, $\Sigma_2^1 = \{b, c, e\}$
- $\Sigma_1^2 = \{a\}$, $\Sigma_2^2 = \{d\}$
- $\Sigma_1^3 = \{c, e\}$, $\Sigma_2^3 = \{b\}$
- $\Sigma_1^4 = \{c\}$, $\Sigma_2^4 = \{e\}$



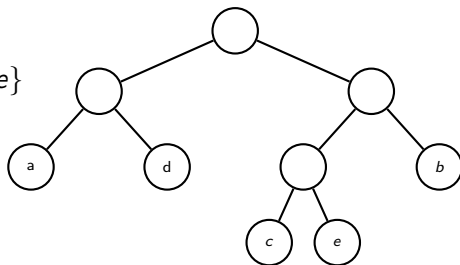
Erwartete Kodelänge

$$= 0,4 \cdot 2 + 0,1 \cdot 2 + 0,15 \cdot 3 + 0,1 \cdot 3 + 0,25 \cdot 2 = 2,25$$

Shannon-Fano-Algorithmus: Beispiel

Wie oben: $\Sigma = \{a, b, c, d, e\}$, Wkten. (0,4; 0,25; 0,15; 0,1; 0,1)

- $\Sigma_1^1 = \{a, d\}$, $\Sigma_2^1 = \{b, c, e\}$
- $\Sigma_1^2 = \{a\}$, $\Sigma_2^2 = \{d\}$
- $\Sigma_1^3 = \{c, e\}$, $\Sigma_2^3 = \{b\}$
- $\Sigma_1^4 = \{c\}$, $\Sigma_2^4 = \{e\}$



Erwartete Kodelänge

$$= 0,4 \cdot 2 + 0,1 \cdot 2 + 0,15 \cdot 3 + 0,1 \cdot 3 + 0,25 \cdot 2 = 2,25$$

Werden sehen: auch nicht optimal

Huffman-Algorithmus

Huffman (1952): **Iterativ** Präfixkodebaum erzeugen

Huffman-Algorithmus

Huffman (1952): **Iterativ** Präfixcodebaum erzeugen

Vorgehen: bottom-up mithilfe von Gewichtswerten

- 1 Erzeuge Wald aus $t := n$ Bäumen, die je nur aus der Wurzel mit Markierung s_i bestehen und Gewicht $\text{Prob}(s_i)$ haben.

Huffman-Algorithmus

Huffman (1952): **Iterativ** Präfixkodebaum erzeugen

Vorgehen: bottom-up mithilfe von Gewichtswerten

- 1 Erzeuge Wald aus $t := n$ Bäumen, die je nur aus der Wurzel mit Markierung s_i bestehen und Gewicht $\text{Prob}(s_i)$ haben.
- 2 Solange $t > 1$
- 3 Wähle Bäume T_1 und T_2 mit min. Gewichten w_1 und w_2 .

Huffman-Algorithmus

Huffman (1952): **Iterativ** Präfixkodebaum erzeugen

Vorgehen: bottom-up mithilfe von Gewichtswerten

- 1 Erzeuge Wald aus $t := n$ Bäumen, die je nur aus der Wurzel mit Markierung s_i bestehen und Gewicht $\text{Prob}(s_i)$ haben.
- 2 Solange $t > 1$
- 3 Wähle Bäume T_1 und T_2 mit min. Gewichten w_1 und w_2 .
- 4 Erzeuge Baum T mit unmarkierter Wurzel, linkem Teilbaum T_1 , rechtem Teilbaum T_2 und Gewicht $w_1 + w_2$.

Huffman-Algorithmus

Huffman (1952): **Iterativ** Präfixkodebaum erzeugen

Vorgehen: bottom-up mithilfe von Gewichtswerten

- 1 Erzeuge Wald aus $t := n$ Bäumen, die je nur aus der Wurzel mit Markierung s_i bestehen und Gewicht $\text{Prob}(s_i)$ haben.
- 2 Solange $t > 1$
- 3 Wähle Bäume T_1 und T_2 mit min. Gewichten w_1 und w_2 .
- 4 Erzeuge Baum T mit unmarkierter Wurzel, linkem Teilbaum T_1 , rechtem Teilbaum T_2 und Gewicht $w_1 + w_2$.
- 5 Entferne T_1 und T_2 .

Huffman-Algorithmus

Huffman (1952): **Iterativ** Präfixkodebaum erzeugen

Vorgehen: bottom-up mithilfe von Gewichtswerten

- 1 Erzeuge Wald aus $t := n$ Bäumen, die je nur aus der Wurzel mit Markierung s_i bestehen und Gewicht $\text{Prob}(s_i)$ haben.
- 2 Solange $t > 1$
- 3 Wähle Bäume T_1 und T_2 mit min. Gewichten w_1 und w_2 .
- 4 Erzeuge Baum T mit unmarkierter Wurzel, linkem Teilbaum T_1 , rechtem Teilbaum T_2 und Gewicht $w_1 + w_2$.
- 5 Entferne T_1 und T_2 .
- 6 $t := t - 1$.

Huffman-Algorithmus

Huffman (1952): **Iterativ** Präfixkobaum erzeugen

Vorgehen: bottom-up mithilfe von Gewichtswerten

- 1 Erzeuge Wald aus $t := n$ Bäumen, die je nur aus der Wurzel mit Markierung s_i bestehen und Gewicht $\text{Prob}(s_i)$ haben.
- 2 Solange $t > 1$
- 3 Wähle Bäume T_1 und T_2 mit min. Gewichten w_1 und w_2 .
- 4 Erzeuge Baum T mit unmarkierter Wurzel, linkem Teilbaum T_1 , rechtem Teilbaum T_2 und Gewicht $w_1 + w_2$.
- 5 Entferne T_1 und T_2 .
- 6 $t := t - 1$.

Beobachtungen: erzeugt Präfixkode, Laufzeit $O(n \log n)$ möglich

Huffman-Algorithmus

Huffman (1952): **Iterativ** Präfixkodebaum erzeugen

Vorgehen: bottom-up mithilfe von Gewichtswerten

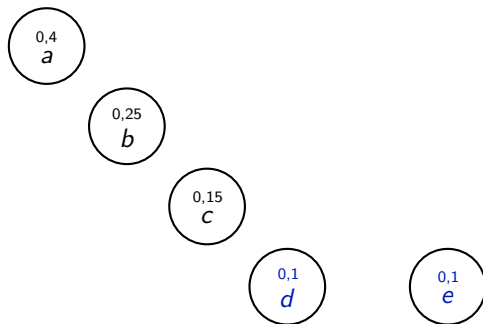
- 1 Erzeuge Wald aus $t := n$ Bäumen, die je nur aus der Wurzel mit Markierung s_i bestehen und Gewicht $\text{Prob}(s_i)$ haben.
- 2 Solange $t > 1$
- 3 Wähle Bäume T_1 und T_2 mit min. Gewichten w_1 und w_2 .
- 4 Erzeuge Baum T mit unmarkierter Wurzel, linkem Teilbaum T_1 , rechtem Teilbaum T_2 und Gewicht $w_1 + w_2$.
- 5 Entferne T_1 und T_2 .
- 6 $t := t - 1$.

Beobachtungen: erzeugt Präfixkode, Laufzeit $O(n \log n)$ möglich

Anwendungen: Übertragung von Faxen (Gruppen 3 und 4)

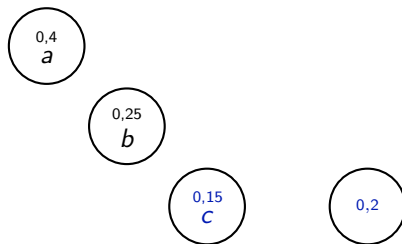
Huffman-Algorithmus: Beispiel

Wieder: $\Sigma = \{a, b, c, d, e\}$, Wkten. (0,4; 0,25; 0,15; 0,1; 0,1)



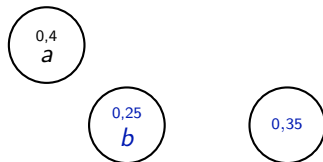
Huffman-Algorithmus: Beispiel

Wieder: $\Sigma = \{a, b, c, d, e\}$, Wkten. $(0,4; 0,25; 0,15; 0,1; 0,1)$



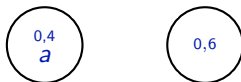
Huffman-Algorithmus: Beispiel

Wieder: $\Sigma = \{a, b, c, d, e\}$, Wkten. $(0,4; 0,25; 0,15; 0,1; 0,1)$



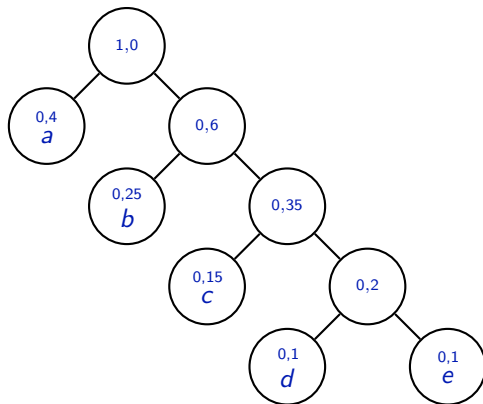
Huffman-Algorithmus: Beispiel

Wieder: $\Sigma = \{a, b, c, d, e\}$, Wkten. $(0,4; 0,25; 0,15; 0,1; 0,1)$



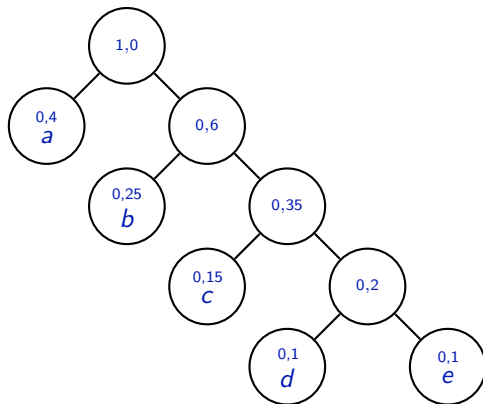
Huffman-Algorithmus: Beispiel

Wieder: $\Sigma = \{a, b, c, d, e\}$, Wkten. (0,4; 0,25; 0,15; 0,1; 0,1)



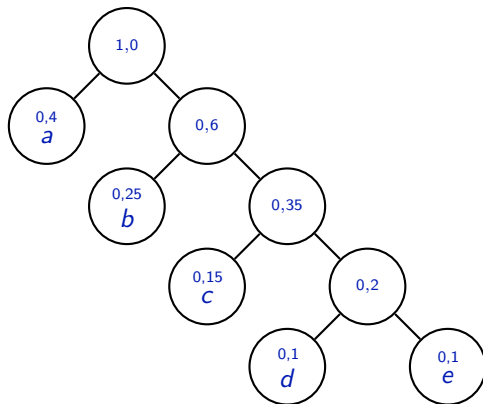
Huffman-Algorithmus: Beispiel

Wieder: $\Sigma = \{a, b, c, d, e\}$, Wkten. (0,4; 0,25; 0,15; 0,1; 0,1)



Huffman-Algorithmus: Beispiel

Wieder: $\Sigma = \{a, b, c, d, e\}$, Wkten. $(0,4; 0,25; 0,15; 0,1; 0,1)$



Erwartete Kodelänge

$$= 0,4 \cdot 1 + 0,25 \cdot 2 + 0,15 \cdot 3 + 0,1 \cdot 4 + 0,1 \cdot 4 = 2,15 \quad \text{optimal?}$$

Optimalität der Huffman-Kodierung

Theorem

Die Huffman-Kodierung hat optimale erwartete Kodelänge.

Optimalität der Huffman-Kodierung

Theorem

Die Huffman-Kodierung hat optimale erwartete Kodelänge.

Beweis per Induktion über $|\Sigma|$.

- Induktionsanfang für $|\Sigma| \leq 2$ klar.

Optimalität der Huffman-Kodierung

Theorem

Die Huffman-Kodierung hat optimale erwartete Kodelänge.

Beweis per Induktion über $|\Sigma|$.

- Induktionsanfang für $|\Sigma| \leq 2$ klar.
- Sei nun $|\Sigma| > 2$ und
 T ein Baum, der einen optimalen Präfixkode für Σ darstellt.

Optimalität der Huffman-Kodierung

Theorem

Die Huffman-Kodierung hat optimale erwartete Kodelänge.

Beweis per Induktion über $|\Sigma|$.

- Induktionsanfang für $|\Sigma| \leq 2$ klar.
- Sei nun $|\Sigma| > 2$ und
 T ein Baum, der einen optimalen Präfixkode für Σ darstellt.
- 1. **Beobachtung:** Jeder innere Knoten in T hat zwei Kinder (sonst Widerspruch zur Optimalität).

Optimalität der Huffman-Kodierung

Theorem

Die Huffman-Kodierung hat optimale erwartete Kodelänge.

Beweis per Induktion über $|\Sigma|$.

- Induktionsanfang für $|\Sigma| \leq 2$ klar.
- Sei nun $|\Sigma| > 2$ und T ein Baum, der einen optimalen Präfixkode für Σ darstellt.
- 1. **Beobachtung:** Jeder innere Knoten in T hat zwei Kinder (sonst Widerspruch zur Optimalität).
- 2. **Beobachtung:** Seien s_i und s_j die Zeichen mit den geringsten Wahrscheinlichkeiten. Dann haben s_i und s_j in T maximale Tiefe (sonst Widerspruch zur Optimalität).

Optimalität der Huffman-Kodierung

Theorem

Die Huffman-Kodierung hat optimale erwartete Kodelänge.

Beweis per Induktion über $|\Sigma|$.

- Induktionsanfang für $|\Sigma| \leq 2$ klar.
- Sei nun $|\Sigma| > 2$ und T ein Baum, der einen optimalen Präfixkode für Σ darstellt.
- 1. **Beobachtung:** Jeder innere Knoten in T hat zwei Kinder (sonst Widerspruch zur Optimalität).
- 2. **Beobachtung:** Seien s_i und s_j die Zeichen mit den geringsten Wahrscheinlichkeiten. Dann haben s_i und s_j in T maximale Tiefe (sonst Widerspruch zur Optimalität).
- **Also:** s_i und s_j in T wie im Huffman-Baum

Optimalität der Huffman-Kodierung

Theorem

Die Huffman-Kodierung hat optimale erwartete Kodelänge.

Beweis per Induktion über $|\Sigma|$.

- Induktionsanfang für $|\Sigma| \leq 2$ klar.
- Sei nun $|\Sigma| > 2$ und T ein Baum, der einen optimalen Präfixkode für Σ darstellt.
- 1. **Beobachtung:** Jeder innere Knoten in T hat zwei Kinder (sonst Widerspruch zur Optimalität).
- 2. **Beobachtung:** Seien s_i und s_j die Zeichen mit den geringsten Wahrscheinlichkeiten. Dann haben s_i und s_j in T maximale Tiefe (sonst Widerspruch zur Optimalität).
- **Also:** s_i und s_j in T wie im Huffman-Baum
- Ersetze s_i und s_j durch neuen Buchstaben s mit $\text{Prob}(s) = \text{Prob}(s_i) + \text{Prob}(s_j)$.

Optimalität der Huffman-Kodierung

Theorem

Die Huffman-Kodierung hat optimale erwartete Kodelänge.

Beweis per Induktion über $|\Sigma|$.

- Induktionsanfang für $|\Sigma| \leq 2$ klar.
- Sei nun $|\Sigma| > 2$ und T ein Baum, der einen optimalen Präfixkode für Σ darstellt.
- 1. **Beobachtung:** Jeder innere Knoten in T hat zwei Kinder (sonst Widerspruch zur Optimalität).
- 2. **Beobachtung:** Seien s_i und s_j die Zeichen mit den geringsten Wahrscheinlichkeiten. Dann haben s_i und s_j in T maximale Tiefe (sonst Widerspruch zur Optimalität).
- **Also:** s_i und s_j in T wie im Huffman-Baum
- Ersetze s_i und s_j durch neuen Buchstaben s mit $\text{Prob}(s) = \text{Prob}(s_i) + \text{Prob}(s_j)$.
- Indukt.-Vor.: Rest-Huffman-Baum für neues Σ optimal

⇒ Induktionsschritt.



Huffman-Kodierung und Entropie (1/2)

Wissen: erwartete Kodelänge L_c^{Huffman} von Huffman-Kode optimal

Vergleich mit Entropie (theoretischem Optimum) gewünscht

Huffman-Kodierung und Entropie (1/2)

Wissen: erwartete Kodelänge L_c^{Huffman} von Huffman-Kode optimal

Vergleich mit Entropie (theoretischem Optimum) gewünscht

Theorem

$$H(\Sigma) \leq L_c^{\text{Huffman}} \leq H(\Sigma) + 1.$$

Huffman-Kodierung und Entropie (1/2)

Wissen: erwartete Kodelänge L_c^{Huffman} von Huffman-Kode optimal

Vergleich mit Entropie (theoretischem Optimum) gewünscht

Theorem

$$H(\Sigma) \leq L_c^{\text{Huffman}} \leq H(\Sigma) + 1.$$

Beweis

Zweite Ungleichung folgt wegen Optimalität von L_c^{Huffman} aus

$$L_c^{\text{Huffman}} \leq L_c^{\text{Shannon}} \leq H(\Sigma) + 1.$$

Huffman-Kodierung und Entropie (1/2)

Wissen: erwartete Kodelänge L_c^{Huffman} von Huffman-Kode optimal

Vergleich mit Entropie (theoretischem Optimum) gewünscht

Theorem

$$H(\Sigma) \leq L_c^{\text{Huffman}} \leq H(\Sigma) + 1.$$

Beweis

Zweite Ungleichung folgt wegen Optimalität von L_c^{Huffman} aus

$$L_c^{\text{Huffman}} \leq L_c^{\text{Shannon}} \leq H(\Sigma) + 1.$$

Erste Ungleichung:

Betrachte

$$\underbrace{\left(- \sum_{s \in \Sigma} \text{Prob}(s) \log(\text{Prob}(s)) \right)}_{H(\Sigma)} - \underbrace{\left(\sum_{s \in \Sigma} \text{Prob}(s) |c(s)| \right)}_{L_c^{\text{Huffman}}}$$

Huffman-Kodierung und Entropie (2/2)

$$\underbrace{\left(- \sum_{s \in \Sigma} \text{Prob}(s) \log(\text{Prob}(s)) \right)}_{H(\Sigma)} - \underbrace{\left(\sum_{s \in \Sigma} \text{Prob}(s) |c(s)| \right)}_{L_c^{\text{Huffman}}}$$

Huffman-Kodierung und Entropie (2/2)

$$\underbrace{\left(- \sum_{s \in \Sigma} \text{Prob}(s) \log(\text{Prob}(s)) \right)}_{H(\Sigma)} - \underbrace{\left(\sum_{s \in \Sigma} \text{Prob}(s) |c(s)| \right)}_{L_c^{\text{Huffman}}}$$
$$= \sum_{s \in \Sigma} \text{Prob}(s) \cdot \left(\log \left(\frac{1}{\text{Prob}(s)} \right) - |c(s)| \right)$$

Huffman-Kodierung und Entropie (2/2)

$$\begin{aligned} & \underbrace{\left(- \sum_{s \in \Sigma} \text{Prob}(s) \log(\text{Prob}(s)) \right)}_{H(\Sigma)} - \underbrace{\left(\sum_{s \in \Sigma} \text{Prob}(s) |c(s)| \right)}_{L_c^{\text{Huffman}}} \\ &= \sum_{s \in \Sigma} \text{Prob}(s) \cdot \left(\log \left(\frac{1}{\text{Prob}(s)} \right) - |c(s)| \right) \\ &= \sum_{s \in \Sigma} \text{Prob}(s) \cdot \left(\log \left(\frac{1}{\text{Prob}(s)} \cdot 2^{-|c(s)|} \right) \right) \end{aligned}$$

Huffman-Kodierung und Entropie (2/2)

$$\begin{aligned} & \underbrace{\left(- \sum_{s \in \Sigma} \text{Prob}(s) \log(\text{Prob}(s)) \right)}_{H(\Sigma)} - \underbrace{\left(\sum_{s \in \Sigma} \text{Prob}(s) |c(s)| \right)}_{L_c^{\text{Huffman}}} \\ &= \sum_{s \in \Sigma} \text{Prob}(s) \cdot \left(\log \left(\frac{1}{\text{Prob}(s)} \right) - |c(s)| \right) \\ &= \sum_{s \in \Sigma} \text{Prob}(s) \cdot \left(\log \left(\frac{1}{\text{Prob}(s)} \cdot 2^{-|c(s)|} \right) \right) \\ &= E(\log(X)) \quad \text{mit Zufallsv. } X = 2^{-|c(s)|} / \text{Prob}(s). \end{aligned}$$

Huffman-Kodierung und Entropie (2/2)

$$\begin{aligned}
 & \underbrace{\left(- \sum_{s \in \Sigma} \text{Prob}(s) \log(\text{Prob}(s)) \right)}_{H(\Sigma)} - \underbrace{\left(\sum_{s \in \Sigma} \text{Prob}(s) |c(s)| \right)}_{L_c^{\text{Huffman}}} \\
 &= \sum_{s \in \Sigma} \text{Prob}(s) \cdot \left(\log \left(\frac{1}{\text{Prob}(s)} \right) - |c(s)| \right) \\
 &= \sum_{s \in \Sigma} \text{Prob}(s) \cdot \left(\log \left(\frac{1}{\text{Prob}(s)} \cdot 2^{-|c(s)|} \right) \right) \\
 &= E(\log(X)) \quad \text{mit Zufallsv. } X = 2^{-|c(s)|} / \text{Prob}(s).
 \end{aligned}$$

Jensensche Ungleichung: $\forall f$ konkav: $E(f(X)) \leq f(E(X))$.

Huffman-Kodierung und Entropie (2/2)

$$\begin{aligned}
 & \underbrace{\left(- \sum_{s \in \Sigma} \text{Prob}(s) \log(\text{Prob}(s)) \right)}_{H(\Sigma)} - \underbrace{\left(\sum_{s \in \Sigma} \text{Prob}(s) |c(s)| \right)}_{L_c^{\text{Huffman}}} \\
 &= \sum_{s \in \Sigma} \text{Prob}(s) \cdot \left(\log \left(\frac{1}{\text{Prob}(s)} \right) - |c(s)| \right) \\
 &= \sum_{s \in \Sigma} \text{Prob}(s) \cdot \left(\log \left(\frac{1}{\text{Prob}(s)} \cdot 2^{-|c(s)|} \right) \right) \\
 &= E(\log(X)) \quad \text{mit Zufallsv. } X = 2^{-|c(s)|} / \text{Prob}(s).
 \end{aligned}$$

Jensensche Ungleichung: $\forall f$ konkav: $E(f(X)) \leq f(E(X))$.

$$\leq \log \left(\sum_{s \in \Sigma} \text{Prob}(s) \frac{2^{-|c(s)|}}{\text{Prob}(s)} \right)$$

Huffman-Kodierung und Entropie (2/2)

$$\begin{aligned}
 & \underbrace{\left(- \sum_{s \in \Sigma} \text{Prob}(s) \log(\text{Prob}(s)) \right)}_{H(\Sigma)} - \underbrace{\left(\sum_{s \in \Sigma} \text{Prob}(s) |c(s)| \right)}_{L_c^{\text{Huffman}}} \\
 &= \sum_{s \in \Sigma} \text{Prob}(s) \cdot \left(\log \left(\frac{1}{\text{Prob}(s)} \right) - |c(s)| \right) \\
 &= \sum_{s \in \Sigma} \text{Prob}(s) \cdot \left(\log \left(\frac{1}{\text{Prob}(s)} \cdot 2^{-|c(s)|} \right) \right) \\
 &= E(\log(X)) \quad \text{mit Zufallsv. } X = 2^{-|c(s)|} / \text{Prob}(s).
 \end{aligned}$$

Jensensche Ungleichung: $\forall f$ konkav: $E(f(X)) \leq f(E(X))$.

$$\leq \log \left(\sum_{s \in \Sigma} \text{Prob}(s) \frac{2^{-|c(s)|}}{\text{Prob}(s)} \right) = \log \left(\sum_{s \in \Sigma} 2^{-|c(s)|} \right)$$

Huffman-Kodierung und Entropie (2/2)

$$\begin{aligned}
 & \underbrace{\left(- \sum_{s \in \Sigma} \text{Prob}(s) \log(\text{Prob}(s)) \right)}_{H(\Sigma)} - \underbrace{\left(\sum_{s \in \Sigma} \text{Prob}(s) |c(s)| \right)}_{L_c^{\text{Huffman}}} \\
 &= \sum_{s \in \Sigma} \text{Prob}(s) \cdot \left(\log \left(\frac{1}{\text{Prob}(s)} \right) - |c(s)| \right) \\
 &= \sum_{s \in \Sigma} \text{Prob}(s) \cdot \left(\log \left(\frac{1}{\text{Prob}(s)} \cdot 2^{-|c(s)|} \right) \right) \\
 &= E(\log(X)) \quad \text{mit Zufallsv. } X = 2^{-|c(s)|} / \text{Prob}(s).
 \end{aligned}$$

Jensensche Ungleichung: $\forall f$ konkav: $E(f(X)) \leq f(E(X))$.

$$\leq \log \left(\sum_{s \in \Sigma} \text{Prob}(s) \frac{2^{-|c(s)|}}{\text{Prob}(s)} \right) = \log \left(\sum_{s \in \Sigma} 2^{-|c(s)|} \right) \leq \log(1) = 0$$

wegen der Ungleichung von Kraft/McMillan.



Huffman-Kodierung: Diskussion

Frage: Stört $+1$ in $L_c^{\text{Huffman}} \leq H(\Sigma) + 1$?

Beispiel: $\Sigma = \{a, b\}$, $\text{Prob}(a) = 0,99$, $\text{Prob}(b) = 0,01$

Huffman-Kodierung: Diskussion

Frage: Stört $+1$ in $L_c^{\text{Huffman}} \leq H(\Sigma) + 1$?

Beispiel: $\Sigma = \{a, b\}$, $\text{Prob}(a) = 0,99$, $\text{Prob}(b) = 0,01$

Klar: Huffman-Kode $c(a) = 0$, $c(b) = 1$ (oder umgekehrt)

erwartete Länge $L_c = 1 \cdot 0,99 + 1 \cdot 0,01 = 1$

Entropie $= 0,99 \cdot \log(1/0,99) + 0,01 \cdot \log(1/0,01) \leq 0,081$

Konsequenzen?

Huffman-Kodierung: Diskussion

Frage: Stört $+1$ in $L_c^{\text{Huffman}} \leq H(\Sigma) + 1$?

Beispiel: $\Sigma = \{a, b\}$, $\text{Prob}(a) = 0,99$, $\text{Prob}(b) = 0,01$

Klar: Huffman-Kode $c(a) = 0$, $c(b) = 1$ (oder umgekehrt)

erwartete Länge $L_c = 1 \cdot 0,99 + 1 \cdot 0,01 = 1$

Entropie $= 0,99 \cdot \log(1/0,99) + 0,01 \cdot \log(1/0,01) \leq 0,081$

Konsequenzen?

Definition: Redundanz = erwartete Länge – Entropie

Hier: Redundanz $\geq 0,919$

Anders gesagt: Kompression mehr als 1134% länger als optimal

Huffman-Kodierung: Diskussion

Frage: Stört $+1$ in $L_c^{\text{Huffman}} \leq H(\Sigma) + 1$?

Beispiel: $\Sigma = \{a, b\}$, $\text{Prob}(a) = 0,99$, $\text{Prob}(b) = 0,01$

Klar: Huffman-Kode $c(a) = 0$, $c(b) = 1$ (oder umgekehrt)

erwartete Länge $L_c = 1 \cdot 0,99 + 1 \cdot 0,01 = 1$

Entropie $= 0,99 \cdot \log(1/0,99) + 0,01 \cdot \log(1/0,01) \leq 0,081$

Konsequenzen?

Definition: Redundanz = erwartete Länge – Entropie

Hier: Redundanz $\geq 0,919$

Anders gesagt: Kompression mehr als 1134% länger als optimal

Problem: a nicht kürzer als 1 Bit kodierbar
unabhängig von $\text{Prob}(a)$

Erweiterte Huffman-Kodierung

Beispiel: $\Sigma = \{a, b\}$, $\text{Prob}(a) = 0,99$, $\text{Prob}(b) = 0,01$

Idee: Zusammenfassung von je 2 Buchstaben zu einem Block

Erweiterte Huffman-Kodierung

Beispiel: $\Sigma = \{a, b\}$, $\text{Prob}(a) = 0,99$, $\text{Prob}(b) = 0,01$

Idee: Zusammenfassung von je 2 Buchstaben zu einem Block

Also: $\Sigma^2 = \{aa, ab, ba, bb\}$ mit $\text{Prob}(aa) = 0,9801$,
 $\text{Prob}(ab) = \text{Prob}(ba) = 0,0099$, $\text{Prob}(bb) = 0,0001$

Erweiterte Huffman-Kodierung

Beispiel: $\Sigma = \{a, b\}$, $\text{Prob}(a) = 0,99$, $\text{Prob}(b) = 0,01$

Idee: Zusammenfassung von je 2 Buchstaben zu einem Block

Also: $\Sigma^2 = \{aa, ab, ba, bb\}$ mit $\text{Prob}(aa) = 0,9801$,
 $\text{Prob}(ab) = \text{Prob}(ba) = 0,0099$, $\text{Prob}(bb) = 0,0001$

Huffman-Kode: $c(aa) = 0$, $c(ab) = 10$, $c(ba) = 110$, $c(bb) = 111$.

Erweiterte Huffman-Kodierung

Beispiel: $\Sigma = \{a, b\}$, $\text{Prob}(a) = 0,99$, $\text{Prob}(b) = 0,01$

Idee: Zusammenfassung von je 2 Buchstaben zu einem Block

Also: $\Sigma^2 = \{aa, ab, ba, bb\}$ mit $\text{Prob}(aa) = 0,9801$,
 $\text{Prob}(ab) = \text{Prob}(ba) = 0,0099$, $\text{Prob}(bb) = 0,0001$

Huffman-Kode: $c(aa) = 0$, $c(ab) = 10$, $c(ba) = 110$, $c(bb) = 111$.

Erwartete Länge

$$= 0,9801 \cdot 1 + 0,0099 \cdot 2 + 0,0099 \cdot 3 + 0,0001 \cdot 3 = 1,0299$$

Entropie $H(\Sigma^2) = 0,9801 \cdot \log(1/0,9801) + 2 \cdot 0,0099 \cdot \log(1/0,0099) + 0,0001 \cdot \log(1/0,0001) = 0,161\dots$

Erweiterte Huffman-Kodierung

Beispiel: $\Sigma = \{a, b\}$, $\text{Prob}(a) = 0,99$, $\text{Prob}(b) = 0,01$

Idee: Zusammenfassung von je 2 Buchstaben zu einem Block

Also: $\Sigma^2 = \{aa, ab, ba, bb\}$ mit $\text{Prob}(aa) = 0,9801$,
 $\text{Prob}(ab) = \text{Prob}(ba) = 0,0099$, $\text{Prob}(bb) = 0,0001$

Huffman-Kode: $c(aa) = 0$, $c(ab) = 10$, $c(ba) = 110$, $c(bb) = 111$.

Erwartete Länge

$$= 0,9801 \cdot 1 + 0,0099 \cdot 2 + 0,0099 \cdot 3 + 0,0001 \cdot 3 = 1,0299$$

Entropie $H(\Sigma^2) = 0,9801 \cdot \log(1/0,9801) + 2 \cdot 0,0099 \cdot \log(1/0,0099) + 0,0001 \cdot \log(1/0,0001) = 0,161 \dots$

Also: Redundanz $\leq 0,869$, Kompression „nur noch“ 537% zu lang

Erweiterte Huffman-Kodierung: weiteres Beispiel

Beispiel: $\Sigma = \{a, b\}$, $\text{Prob}(a) = 0,99$, $\text{Prob}(b) = 0,01$

Idee: Zusammenfassung von je 3 Buchstaben zu einem Block

Erweiterte Huffman-Kodierung: weiteres Beispiel

Beispiel: $\Sigma = \{a, b\}$, $\text{Prob}(a) = 0,99$, $\text{Prob}(b) = 0,01$

Idee: Zusammenfassung von je 3 Buchstaben zu einem Block

Also: $\Sigma^3 = \{aaa, aab, aba, baa, abb, bab, bba, bbb\}$

Erweiterte Huffman-Kodierung: weiteres Beispiel

Beispiel: $\Sigma = \{a, b\}$, $\text{Prob}(a) = 0,99$, $\text{Prob}(b) = 0,01$

Idee: Zusammenfassung von je 3 Buchstaben zu einem Block

Also: $\Sigma^3 = \{aaa, aab, aba, baa, abb, bab, bba, bbb\}$

Huffman-Kode: $c(aaa) = 0$, $c(aab) = 100$, $c(aba) = 101$, $c(baa) = 110$,
 $c(abb) = 11100$, $c(bab) = 11101$, $c(bba) = 11110$, $c(bbb) = 11111$

Erweiterte Huffman-Kodierung: weiteres Beispiel

Beispiel: $\Sigma = \{a, b\}$, $\text{Prob}(a) = 0,99$, $\text{Prob}(b) = 0,01$

Idee: Zusammenfassung von je 3 Buchstaben zu einem Block

Also: $\Sigma^3 = \{aaa, aab, aba, baa, abb, bab, bba, bbb\}$

Huffman-Kode: $c(aaa) = 0$, $c(aab) = 100$, $c(aba) = 101$, $c(baa) = 110$,
 $c(abb) = 11100$, $c(bab) = 11101$, $c(bba) = 11110$, $c(bbb) = 11111$

Erwartete Länge = 1,059998

Entropie $H(\Sigma^3) \geq 0,242$

Erweiterte Huffman-Kodierung: weiteres Beispiel

Beispiel: $\Sigma = \{a, b\}$, $\text{Prob}(a) = 0,99$, $\text{Prob}(b) = 0,01$

Idee: Zusammenfassung von je 3 Buchstaben zu einem Block

Also: $\Sigma^3 = \{aaa, aab, aba, baa, abb, bab, bba, bbb\}$

Huffman-Kode: $c(aaa) = 0$, $c(aab) = 100$, $c(aba) = 101$, $c(baa) = 110$,
 $c(abb) = 11100$, $c(bab) = 11101$, $c(bba) = 11110$, $c(bbb) = 11111$

Erwartete Länge = 1,059998

Entropie $H(\Sigma^3) \geq 0,242$

Also: Redundanz $\leq 0,818$, Kompression „nur noch“ 338% zu lang

Erweiterte Huffman-Kodierung: weiteres Beispiel

Beispiel: $\Sigma = \{a, b\}$, $\text{Prob}(a) = 0,99$, $\text{Prob}(b) = 0,01$

Idee: Zusammenfassung von je 3 Buchstaben zu einem Block

Also: $\Sigma^3 = \{aaa, aab, aba, baa, abb, bab, bba, bbb\}$

Huffman-Kode: $c(aaa) = 0$, $c(aab) = 100$, $c(aba) = 101$, $c(baa) = 110$,
 $c(abb) = 11100$, $c(bab) = 11101$, $c(bba) = 11110$, $c(bbb) = 11111$

Erwartete Länge = 1,059998

Entropie $H(\Sigma^3) \geq 0,242$

Also: Redundanz $\leq 0,818$, Kompression „nur noch“ 338% zu lang

Vergleich der 3 Varianten:

k	$H(\Sigma^k)$	$L_c(\Sigma^k)$	$L_c(\Sigma^k)/k$	$L_c(\Sigma^k)/k - H(\Sigma^k)$	$L_c(\Sigma^k)/k - H(\Sigma)$
1	0,081	1,000	1,000	0,919	0,919
2	0,161	1,030	0,515	0,354	0,434
3	0,242	1,060	0,353	0,111	0,272

Erweiterte Huffman-Kodierung: Analyse

Beobachtung: $L_c(\Sigma^k)/k =$ bei Blocklänge k erwartete Kodierungslänge *eines Zeichens* der erweiterten Huffman-Kodierung

Erweiterte Huffman-Kodierung: Analyse

Beobachtung: $L_c(\Sigma^k)/k =$ bei Blocklänge k erwartete Kodierungslänge *eines Zeichens* der erweiterten Huffman-Kodierung

Theorem

$$L_c(\Sigma^k)/k \leq H(\Sigma) + 1/k.$$