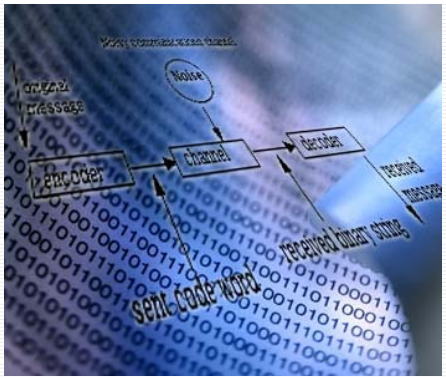
Kapitel 7: Optimalcodierung und Huffman Coding



Ziele des Kapitels

ETH

- Auftreten von Fehlern bei zu starker Kompression
- Konstruktion optimaler Codes
- Huffman Coding

Bisher

ETH

- Theorem (Shannon I): Die mittlere Codewortlänge $E[l_C(X)]$ eines optimalen präfixfreien Codes C ist begrenzt durch die Entropie

$$H(X) \le E[l_C(X)] < H(X)+1$$

- Das Shannon'sche Codierungstheorem gibt also eine obere und untere Schranke für die Konstruktion eines optimalen Codes
- Diese Schranke wird im Mittel erfüllt
- Wir wollen nun die Konstruktion von Optimalcodes untersuchen

Code-Redundanz

ETH

- Definition: Die Redundanz R_C eines Codes C für eine Quelle X ist wie folgt definiert:

$$R_C = E[l_C(X)] - H(X) \ge 0$$

Ein optimaler Code besitzt minimale Coderedundanz. Dennoch kann diese grösser als 0 sein.

Redundanzfreie Codierung

- Es stellt sich die Frage nach einer völlig redundanzfreien Codierung
- Definition: Unter einer erweiterten Quelle versteht man eine Ersatzquelle, welche durch Zusammenfassung von je *m* Zeichen der Ursprungsquelle mit *L* Symbolen zu einem Ersatzzeichen entsteht.
  - Anzahl der Ersatzzeichen ist L^m
- Werden die Zeichen nicht einzeln, sondern in Blöcken der Grösse *m* codiert, so gilt

$$H(X) \leq E[l_C(X)] < H(X) + \frac{1}{m}$$

Redundanzfreie Codierung

- Sowie

$$m \cdot H(X) \leq m \cdot E[l_C(X)] < m \cdot H(X) + 1$$

- Die Coderedundanz eines solchen Codes ist

$$R_C < \frac{1}{m}$$

- Solche Blockcodes sind also generell redundanzärmer, als Codes für Einzelzeichen
- Nachteil: nicht verzögerungsfreie Decodierbarkeit
- Der Empfänger muss also auf den gesamten Block warten, bevor er mit der Decodierung beginnen kann
- Je nach Datentyp können Blockcodes sehr sinnvoll sein (Beispiel: Bilder)

Nicht perfekte Kompression

- Bisher: Codes, welche die Daten eindeutig und fehlerfrei decodieren
- Entropie stellt Untergrenze der Komprimierbarkeit dar
- Komprimiert man stärker, so wird man Information verlieren
- Frage nach dem Informationsverlust bei zu starker Kompression
- Können wir bei gegebener Kompressionsrate den Fehler minimieren?

Nicht perfekte Kompression

- Theorem (Fano-Ungleichung): Für die Fehlerwahrscheinlichkeit P_e bei der Schätzung von *X* auf Grund von *Y* gilt für jedes Schätzverfahren

$$h(P_e) + P_e \log(|\chi| - 1) \geq H(X|Y)$$

- Insbesondere gilt für binäre Zustandsvariablen *X*:

$$h(P_e) \geq H(X|Y)$$

- Beweis ist im Skript

Die bedingte Entropie (Information), die *Y* über *X* enthält, setzt der Fehlerwahrscheinlichkeit eines jeden Schätzers also eine Untergrenze! Eigentlich logisch ☺

Nicht perfekte Kompression

- Gegeben sei ein N-Bit String der Entropie K Bits ($K <= N$)
- Sei $X^N = [X_1 \ldots X_N]$ eine entsprechende Zufallsvariable für diesen String und C ein beliebiger, binärer Code
- Die mittlere Codewortlänge ist $E[l_c(X^N)]$
- Es gilt

$$H(C(\mathrm{X}^N)) \le E[l_C(\mathrm{X}^N)]$$

- Und somit

$$H(\mathrm{X}^N \mid C(\mathrm{X}^N)) = H(\mathrm{X}^N, C(\mathrm{X}^N)) - H(C(\mathrm{X}^N)) = H(\mathrm{X}^N) - H(C(\mathrm{X}^N)) \ge H(\mathrm{X}^N) - E[l_C(\mathrm{X}^N)]$$

Nicht perfekte Kompression

- Bezeichnen wir mit $P_{e,i}$ die Fehlerwahrscheinlichkeit der Schätzung des i-ten Bits des Strings X^N
- Sei ferner $P_e = \frac{1}{N}\sum_{i=1}^{N} P_{e,i}$
- Theorem: Für die mittlere Bitfehlerwahrscheinlichkeit P_e bei der Decodierung eines N-Bit Strings X^N gilt:

$$P_e \ge h^{-1}\left(\frac{H(C(\mathrm{X}^N)) - E[l_C(\mathrm{X}^N)]}{N}\right)$$

In der Praxis sind derartiges Codes für die Kompression von Bild- und Audiodaten relevant. Ein Beispiel dafür ist JPEG.

Typen von Optimalcodes

- Grundsätzlich unterscheidet man Codes für bekannte Quellenstatistik (Shannon-Fano, Huffman) sowie Codes für unbekannte Quellenstatistik (Ziv-Lempel)
- Ist die Statistik bekannt, kann man Codebäume über gegebene Verteilungen konstruieren.
  - Es gilt: Je grösser die Zeichenwahrscheinlichkeit, desto kürzer muss das Codezeichen sein
- Ist die Statistik nicht bekannt, kann man ein dynamisches Wörterbuch aufbauen und surjektiv Zeichen auf Codewörter abbilden

Lemmas

- Lemma: Ein Baum eines optimal präfixfreien Codes ist ausgefüllt, d.h. er besitzt keine unbesetzten Blätter
- Beweis: ($D=2$) Skizze: Wir unterscheiden zwei Fälle bei Annahme eines optimalen Codes
  - Ist b ein unbesetztes Blatt und ist der Bruder von b, b' auch ein Blatt, so können wir b' direkt dem Vaterknoten zuweisen, ohne die mittlere Codelänge zu erhöhen
  - Ist der Bruder von b ein Teilbaum, so kann eines der am tiefsten liegenden Blätter mit b vertauscht werden -> Fall 1
  - Elimination aller freien Knoten im Baum

Lemmas

- Annahme: Zustandsvariable X mit $|\chi| = L$, $\chi = \{x_1 \ldots x_L\}$ sowie diskrete Verteilung $p_X(x_i) = p_i$ geordnet, $p_1 >= \ldots >= p_L$
- Lemma: Es gibt einen optimalen präfixfreien Code für X, in dem sich die beiden Codewörter für x_{L-1} und x_L nur im letzten Bit unterscheiden, d.h. im Codebaum zwei Geschwisterblättern zugeordnet sind
- Beweis: Trivial
  - Gemäss vorherigem Lemma gibt es mindestens 2 Codewörter maximaler Länge. Diese können ohne Verlust der Optimalität den beiden Werten x_{L-1} und x_L zugeordnet werden

Lemmas

- Aus einem binären präfixfreien Code C für die Liste $[p_1 \ldots p_L]$, $p_1 >= \ldots . >= p_L$ kann rekursiv ein Code für die Liste $[p_1 \ldots p_{L-2}, p_{L-1} + p_L]$ konstruiert werden
- Wir entfernen die Blätter für p_{L-1} und p_L und ersetzen sie durch ihren gemeinsamen Vorfahren
- Lemma: C ist ein optimaler, präfixfreier Code für $[p_1 \ldots p_L]$ genau dann, wenn C' optimal ist für $[p_1 \ldots p_{L-2}, p_{L-1} + p_L]$
- Beweis:
  - Siehe Skript

Huffman-Coding

- Theorem (Huffman): Der folgende Algorithmus liefert einen optimalen binären Code für die Liste $[p_1 \ldots p_L]$ von Codewort-Wahrscheinlichkeiten:
  1. Zeichne L Blätter mit Wahrscheinlichkeiten $p_1 \ldots p_L$ ohne Baum und markiere sie als aktiv.
  2. Führe folgenden Schritt (L-1) mal rekursiv aus:
     1. Fasse zwei Knoten mit den kleinsten Wahrscheinlichkeiten zu einem Zweig zusammen
     2. Aktiviere die Wurzel der entsprechenden Gabel
     3. Weise ihr die Summe der beiden Wahrscheinlichkeiten zu
     4. Deaktiviere die beiden zusammengefassten Knoten
     5. Stelle Codebaum auf und weise Symbole 0 und 1 zu

Huffman-Coding ist eines der bedeutendsten klassischen Codierungsverfahren. Gut merken ☺

Huffman-Coding

- Gegeben eine diskrete Quelle mit der Verteilung

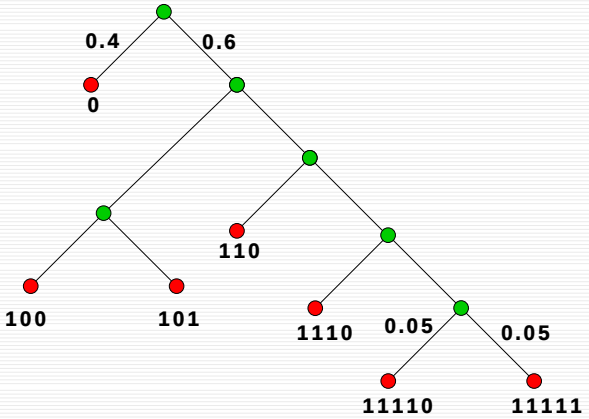
$$P_X(x| = [0.4; 0.18; 0.14; 0.1; 0.08; 0.05; 0.05])$$

- Sortieren und rekursiv aufbauen

0.4 0.18 0.14 0.1 0.08 0.05 0.05
0.4 0.18 0.14 0.1 0.1 0.08
0.4 0.18 0.18 0.14 0.1
0.4 0.24 0.18 0.18
0.4 0.36 0.24
0.6 0.4
1.0

Huffman-Coding

- Code-Baum dazu



Huffman-Coding

- mittlere Codelänge:

$$E[l_C(X)] = 2.48 \text{ Bit /QZ}$$

- Entropie

$$H(X) = 2.43 \text{ Bit /QZ}$$

- Coderedundanz

$$R_C = 0.05 \text{ Bit /QZ}$$

Applet

Huffman-Coding

- Beweis: Aus dem beschriebenen Algorithmus entsteht ein einzelner Baum
  - Die Anzahl der nicht verbundenen Komponenten nimmt in jedem Schritt um 1 ab
  - Die Optimalität erfolgt direkt aus dem letzten Lemma
  - Wir betrachten die Folge von Codes, die aus der Wurzel durch sukzessive Erweiterung in umgekehrter Reihenfolge entstehen
  - Die Wurzel ein Optimalcode für ein einzelnes Zeichen darstellt und der Code bleibt optimal, wenn der vorherige Code optimal ist

Es ist überraschend, dass wir mit einem Greedy-Verfahren in O(*L*) Schritten ein global optimales Ergebnis erreichen.