Karlsruher Institut für Technologie
Institut für Theoretische Informatik

Prof. Dr. P. Sanders, Dr. S. Gog
L. Hübschle-Schneider, T. Maier

14. Übungsblatt zu Theoretische Grundlagen der Informatik im WS 2015/16

http://algo2.iti.kit.edu/TGI2015.php
{sanders,gog,huebschle,t.maier}@kit.edu

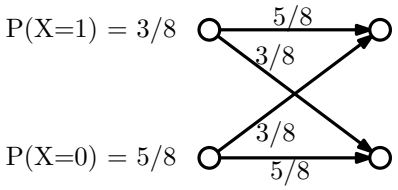
Musterlösungen

Freiwilliges Zusatzblatt, keine Abgabe, keine Korrektur!

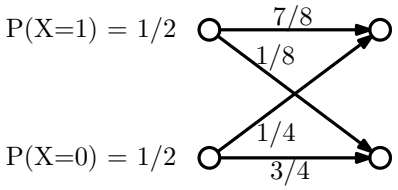
Aufgabe 1 (*Verrauschter Kanal,* $3+4$ *Punkte*)

Berechnen Sie in den folgenden Szenarien die Entropie beim Sender und beim Empfänger. Berechnen Sie außerdem die verloren gegangene Information $H(X|Y)$, die falsch erhaltene Information $H(Y|X)$, und die Transinformation $I(X;Y)$. Betrachten Sie die folgenden Szenarien:

a)



b)



Musterlösung:

a)

$$P(Y=0) = P(Y=0|X=1)\cdot P(X=1) + P(Y=0|X=0)\cdot P(X=0) = (\frac{5}{8})^2 + (\frac{3}{8})^2 = \frac{34}{64} = \frac{17}{32}$$

$$P(Y=1) = 1 - P(Y=0) = \frac{15}{32}$$

$$H(X) = -\sum_{i=0,1} P(X=i)\cdot\log_2(P(X=i)) = -\frac{3}{8}\log_2(\frac{3}{8}) - \frac{5}{8}\log_2(\frac{5}{8}) = 3 - \frac{3}{8}\log_2 3 - \frac{5}{8}\log_2 5$$

$$\approx 0.9544$$

$$H(Y) = -\sum_{i=0,1} P(Y=i)\log_2(P(Y=i))$$

$$= 160 - \frac{17}{32}\log_2(\frac{17}{32}) - \frac{15}{32}\log_2(\frac{15}{32}) = -\frac{17}{32}\log_2 17 - \frac{15}{32}\log_2 15 \approx 0.9972$$

$$H(Y|X) = -\sum_{i=0,1}\sum_{j=0,1} P(X=i,Y=j)\log_2(P(Y=j|X=i))$$

$$= -\frac{25}{64}(\log_2 5 - 3) - \frac{15}{64}(\log_2 3 - 3) - \frac{9}{64}(\log_2 3 - 3) - \frac{15}{64}(\log_2 5 - 3)$$

$$= 3 - \frac{40}{64}\log_2 5 - \frac{24}{64}\log_2 3 \approx 0.9544$$

$$I(X;Y) = H(Y) - H(Y|X) \approx 0.9972 - 0.9544 = 0.0428$$

$$H(X|Y) = H(X) - I(X,Y) \approx 0.9544 - 0.0428 = 0.9116$$

b)

$$
\begin{aligned}
P(Y=0) &= P(Y=0|X=1)\cdot P(X=1) + P(Y=0|X=0)\cdot P(X=0) = \frac{1}{2}\cdot\frac{3}{4} + \frac{1}{2}\cdot\frac{1}{8} = \frac{7}{16}\\
P(Y=1) &= 1 - P(Y=0) = \frac{9}{16}\\
H(X) &= -\sum_{i=0,1} P(X=i)\cdot\log_2(P(X=i)) = -\frac{1}{2}\log_2(\frac{1}{2}) - \frac{1}{2}\log_2(\frac{1}{2}) = 1\\
H(Y) &= -\sum_{i=0,1} P(Y=i)\log_2(P(Y=i))\\
&= -\frac{7}{16}\log_2\frac{7}{16} - \frac{9}{16}\log_2\frac{9}{16} = 4 - \frac{7}{16}\log_2 7 - \frac{9}{16}\log_2 9 \approx 0.9887\\
H(Y|X) &= -\sum_{i=0,1}\sum_{j=0,1} P(X=i,Y=j)\log_2(P(Y=j|X=i))\\
&= -\frac{3}{8}(\log_2 3 - 2) - \frac{1}{8}(\log_2 1 - 2) - \frac{1}{16}(\log_2 1 - 3) - \frac{7}{16}(\log_2 7 - 3)\\
&= 5/2 - \frac{3}{8}\log_2 3 - \frac{7}{16}\log_2 7 \approx 0.6774\\
I(X;Y) &= H(Y) - H(Y|X) \approx 0.9887 - 0.6774 = 0.3113\\
H(X|Y) &= H(X) - I(X,Y) \approx 1 - 0.3113 = 0.6887
\end{aligned}
$$

Aufgabe 2 (*Fehlerkorrigierende Codes,* 3 *Punkte*)

In den Folien über Kreuzsicherung (Folie 53) wurde nicht gesagt, wie sich das unterste rechte Bit der Matrix berechnet. Es ist weder Teil der Daten noch der Längs- oder Querparität und ist daher noch nicht belegt. Wie muss es berechnet werden, damit jeder dreifache Fehler gefunden werden kann? Geben Sie ein Beispiel an, bei dem ein dreifacher Fehler in einem kreuzgesicherten Code *ohne* dieses Bit nicht erkannt werden kann.

Musterlösung:
Das Bit muss auf die Parität der Längs- oder Querparität gesetzt werden. Diese sind zwangsweise gleich, sind sie doch beide ein großes XOR über die ganzen Daten. Die Parität der Längsparität verknüpft die Daten zuerst zeilenweise und dann spaltenweise, die der Querparität genau andersrum. Da XOR aber eine symmetrische Operation ist, muss das Ergebnis das selbe sein.

Wenn das Bit in der n-ten Zeile und m-ten Spalte falsch übertragen wird, und zusätzlich das n-te Bit der Längsparität und das m-te Bit der Querparität unterwegs verfälscht wird, ist der Fehler ohne das untere rechte Paritätsbit nicht bemerkbar. Da dieses sich durch diesen Fehler aber ändern würde, würde der "eigentliche" Code den Fehler bemerken. Erst ein vierfacher Fehler würde hier möglicherweise unbemerkt bleiben.

Aufgabe 3 (*Vergleich verschiedener Kompressionsverfahren,* $1+2+2+2$ *Punkte*)

Wir wollen Schwarz-Weiß-Bilder (bspw. Faxe) komprimieren, und wissen, dass im Schnitt etwa eines von 16 Pixeln schwarz und die anderen 15 weiß sind. In diesem Modell wollen wir jetzt untersuchen, wie hoch die Kompression verschiedener Verfahren ist. Wir betrachten das Bild dazu als Folge von Nullen und Einsen, wobei eine Null einem weißen und eine 1 einem schwarzen Pixel entspricht (wir können auch `s` und `w` schreiben). Wie viele Bits verwenden die folgenden Verfahren im Schnitt pro Pixel? Was ist die Entropie der Quelle? Ignorieren Sie Randfälle am Ende des Worts.

a) Huffman-Coding

b) Blocked Huffman Coding mit Blockgröße 2

c) Blocked Huffman Coding mit Blockgröße 3

d) Elias-Fano-Coding der Positionen der schwarzen Bits

Hinweis: Elias-Fano-Coding ist auch mit $m\lfloor\log\frac{n}{m}\rfloor + \mathbf{2}m + \mathrm{o}(m)$ Bits möglich. Die vereinfachte Version in der Vorlesung verwendet $3m$ Bits für den mittleren Term. Tatsächliche Implementierungen verwenden dafür aber nur $2m$ Bits. Nehmen Sie hier die bessere Schranke an.

Musterlösung:
Die Entropie ist $H_0 = -\frac{15}{16}\cdot\log_2\frac{15}{16} - \frac{1}{16}\cdot\log_2\frac{1}{16} \approx 0.337$ Bit pro Zeichen.

a) Ein normaler Huffman-Code kann das Bild nicht komprimieren, da er einem der Zeichen die 0 und dem anderen die 1 zuweisen muss. Er verwendet also 1 Bit pro Zeichen.

b) Wir gruppieren das Alphabet in Zweiergruppen und erhalten dadurch die neuen Zeichen `ww`, `ws`, `sw` und `ss` mit Wahrscheinlichkeiten $(\frac{15}{16})^2 = \frac{225}{256}$, $\frac{15}{16}\cdot\frac{1}{16} = \frac{15}{256}$ (`ws`, `sw`) und $(\frac{1}{16})^2 = \frac{1}{256}$. Wir erhalten folgende Codes für einen kanonischen Huffman-Code: `0`, `11`, `100` und `101`. Als durchschnittliche Codelänge ergibt sich also $\frac{1\cdot225+2\cdot15+3\cdot15+3\cdot1}{256} = \frac{303}{256}$ Bit pro Zeichen des Blockalphabets, d.h. ungefähr 0.592 Bit pro Zeichen des ursprünglichen Alphabets. Dies ist bereits signifikant besser als der normale Huffman-Code, der keinerlei Kompression erreicht, aber noch ein ganzes Stück von der Entropie der Quelle entfernt.

c) Bei Gruppierung in Dreiergruppen sind die Wahrscheinlichkeiten wie folgt:

- `www`: $(\frac{15}{16}^3) = \frac{3375}{4096}$
- `wws`, `wsw`, `sww`: $(\frac{15}{16})^2\cdot\frac{1}{16} = \frac{225}{4096}$
- `ssw`, `sws`, `wss`: $\frac{15}{16}\cdot(\frac{1}{16})^2 = \frac{16}{4096}$
- `sss`: $(\frac{1}{16})^3 = \frac{1}{4096}$

Der kanonische Huffmancode darüber weißt den Zeichen also (wenn wir sie initial in obiger Reihenfolge sortieren!) folgende Codes zu: `0`, `100`, `101`, `110`, `11100`, `11101`, `11110`, `11111`. Damit ergibt sich eine durchschnittliche Codelänge von $\frac{3375+3\cdot3\cdot225+3\cdot5\cdot16+5}{4096} = \frac{5645}{4096}$ Bit für je drei Zeichen des ursprünglichen Alphabets, also ungefähr 0.459 Bit pro Zeichen. Das ist wiederum etwas besser als Huffman-Coding mit Blockgröße 2 aber immer noch recht weit von der Entropie entfernt.

d) Wir kodieren die $m = \frac{n}{16}$ Positionen der schwarzen Pixel, was eine aufsteigende Folge ist. Daher können wir Elias-Fano-Coding verwenden. Laut Vorlesung existiert eine Darstellung dafür mit $m\lfloor\log\frac{n}{m}\rfloor + 2m + \mathrm{o}(m)$ Bits. Durch Einsetzen erhalten wir

$$\frac{n}{16}\left\lfloor\log\frac{n}{n/16}\right\rfloor + 2\frac{n}{16} + \mathrm{o}\left(\frac{n}{16}\right) = \frac{n}{16}\lfloor\log\rfloor 16 + \frac{n}{8} + \mathrm{o}(n) = \frac{3}{8}n + \mathrm{o}(n)$$

Je länger die Eingabe ist desto unwichtiger wird der sublineare Term, und die Bit pro Zeichen konvergieren gegen 0.375, was ziemlich nahe an der Entropie von 0.337 Bit pro Zeichen liegt. Zusätzlich hat Elias-Fano-Coding den Vorteil, dass wir beliebige Substrings des Codes dekodieren können, d.h. wenn ein Übertragungsfehler passiert, bleibt nur der betroffene Teil des Bildes unvollständig und der Rest ist wieder dekodierbar. Dies ist bei Huffman-Coding nicht gegeben, da man dort wissen muss, wo ein ein Codezeichen beginnt, um es decodieren zu können.

Aufgabe 4 (*Wissensfragen,* 6 *Punkte*)

Sind die folgenden Aussagen wahr oder falsch? Begründen Sie kurz.

Aussage	Wahr	Falsch	Begründung
Die Entropie einer Quelle kann nie größer als 1 werden		×	Dies gilt nur für binäre Quellen, die Entropie einer n-ären Quelle kann bis zu $\log_2 n$ sein
$\frac{1}{\log_{\sqrt{a}}(b^2)} = \frac{1}{4}\log_b a$	×		$\frac{1}{\log_{\sqrt{a}}(b^2)} = \frac{1/2}{\log_{\sqrt{a}} b} = \frac{\log_{\sqrt{a}}\sqrt{a}}{2\log_{\sqrt{a}} b} = \frac{\log_b \sqrt{a}}{2} = \frac{\log_b a}{4}$
Ein Code, der die Kraft-McMillan-Ungleichung erfüllt, ist eindeutig dekodierbar		×	Die Implikation ist andersrum: Jeder eindeutig dekodierbare Code erfüllt die Kraft-McMillan-Ungleichung. Gegenbeispiel in Übung 12
Für Alphabetgröße n wächst die maximale Länge des Huffman-Codes eines Zeichens strikt langsamer als n, liegt also in $\mathrm{o}(n)$		×	Für $p_i = 2^{-i},\ i \in \{1, \ldots, n\}$ hat Zeichen a_i Codelänge $i-1$, insbesondere ist die Codelänge des letzten Zeichens also $n-1 \notin \mathrm{o}(n)$
Wenn für ein Alphabet der Größe n und Codelängen ℓ_i gilt: $\sum_{i=1}^{n} 2^{-\ell_i} \leq 1$, dann gibt es einen Präfixcode mit diesen Codelängen	×		Ja, denn die Kraftsche Ungleichung ist notwendig und hinreichend
Für eine fixe Instanz ist die durchschnittliche Codelänge aller Shannon-Fano-Codes gleich		×	Für die Wkeiten 0.4, 0.2, 0.2, 0.2 sind sowohl 0, 10, 110, 111 als auch 00, 01, 10, 11 gültige SF-Codes, durchschnittliche Länge: 2.2 vs 2.0