Fachbereich Elektrotechnik und Informationstechnik

Informationstheorie

Prof. Dr.-Ing. Snjezana Gligorevic

Vorlesungsskript

Sommersemester 2020

Literaturquellen

- ▷ Salomon, Motta: "Handbook of Data Compression", Springer Verlag 2010
- ▷ Neubauer: "Informationstheorie und Quellencodierung", J. Schlembach Fachverlag, 2006
- ▷ Mengyi Pu: "Fundamental Data Compression", Butterworth-Heinemann Verlag, 2006
- ▷ Sayood: "Introduction to Data Compression", Morgan kaufmann Verlag, 1996
- ▷ Anderson, Johannesson: "Understanding Information Transmission", IEEE Press, Wiley&Sons, Inc., Publication, 2005

Inhaltsverzeichnis

1 Nachricht und Information

Eine Nachricht ist eine Zusammensetzung von Zeichen, die zur Informationsübertragung dient. Dabei kann der am Sender und Empfänger vereinbarte Zeichenset verändert (codiert) werden, wobei dann eine veränderte Nachricht die gleiche Information überträgt.

Ein Signal ist der physikalische Träger einer Nachricht.

Die Information ist der Anteil einer Nachricht, der einen Kenntniszuwachs erzeugt bzw. eine Unsicherheit im Kenntniszustand über die Nachrichtenquelle beseitigt. Dabei hat der Begriff Information viele Aspekte: einen semantischen (Inhalt und Bedeutung), syntaktischen (Grammatik, Formeln, Sprache), pragmatischen (Wert der Information) und statistischen (Wahrscheinlichkeit). Informationstheorie betrachtet nur den statistischen Aspekt und definiert ein quantitatives Maß für Information. Dabei ist die Information ein Maß für die beseitigte Unsicherheit.

Für digitale Nachrichtenübertragung sind die theoretischen Grenzen der Informationsübertragung von Bedeutung: Die Übertragung einer Nachricht soll - unabhängig vom Inhalt der Nachricht - möglichst schnell und fehlerfrei erfolgen.
Die Aufgabe jedes Nachrichtenübertragungssystems ist die empfangsseitige fehlerfreie Rückgewinnung der Originalnachricht, wenn diese durch Störungen auf dem Übertragungskanal beeinträchtigt wurde. Zu diesem Zweck wird die zu übertragene Nachricht unterschiedlich codiert (Quellencodierung $\rightarrow$ Kanalcodierung $\rightarrow$ Modulation).

Codierung ist eine Abbildung einer Zeichen- bzw. Symbolmenge auf eine andere Menge der Zeichen/Symbole.

- ▷ Quellencodierung hat den Zweck der Reduktion der redundanten Nachrichtenteile um die relevante Nachricht auf eine möglichst kleine Größe zu komprimieren.
- ▷ Kanalcodierung hat den Zweck der Fehlererkennung und Korrektur und fügt der Nachricht zusätzliche Redundanz hinzu.
- ▷ Die Modulation hat den Zweck der Anpassung des Nachrichtenträgers (Signals) auf die Eigenschaften des Übertragungskanals.

Die Menge der Information in einer Nachricht wird durch den Informationsgehalt beschrieben. Der Informationsgehalt eines Zeichens bezeichnet die minimale Anzahl von Bits, die benötigt werden, um ein Zeichen (also dessen Information) darzustellen oder zu übertragen.

- ▷ Der Teil der Nachricht, der keinen Kenntniszuwachs erzeugt, wird als *redundant* bezeichnet.
- ▷ Der Teil der Nachricht, der vom Empfänger nicht aufgenommen werden kann, wird als *irrelevant* bezeichnet.

Redundante und irrelevante Anteile der Nachricht können, ohne Folgen für den Empfänger, aus der Nachricht entfernt werden. Entfernen von irrelevanten Nachrichtenanteilen ist eine verlustbehaftete (irreversible) Quellencodierung, da die Nachricht nicht fehlerfrei rekonstruierbar ist. Dagegen ist eine verlustfreie Quellencodierung reversibel.

1.1 Nachrichtenquellen

Eine Nachricht kann als eine Symbolfolge aufgefasst werden, welche eine Aneinanderreihung verschiedener Symbole aus einem festgelegtem Symbolalphabet α darstellt. Die Wahrscheinlichkeit eines einzelnen Symbols x wird durch eine Funktion $p(x)$ definiert.
Beispiele diskreter Nachrichtenquellen sind

- ▷ binäres Alphabet $\{0, 1\}$
- ▷ deutsches Alphabet $\{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}$
- ▷ Augenzahlen eines Würfels $\{1, 2, 3, 4, 5, 6\}$

Allgemein kann jedes Alphabet als eine endliche Menge $\alpha_M = \{x_1, x_2, ..., x_M\}$ dargestellt werden und angenommen werden, dass eine Nachrichtenquelle zu einem Zeitpunkt zufällig ein Element des Alphabets wählt. Das ausgegebene Zeichen entspricht einer Zufallsvariable X. Die Notation $X(i) = x_i$ beschreibt, dass zum Zeitpunkt i als Ergebnis des Zufallsprozesses das Ereignis x_i auftritt. Für ein Ereignis x_i können wir die Wahrscheinlichkeit dessen Auftretens, $p(x_i) \equiv P\{X = x_i\}$ angeben.

- ▷ Ist die Ausgabe von Zeichen des Alphabets zu einem Zeitpunkt unabhängig von der Ausgabe zu jedem anderen Zeitpunkt, dann wird die Nachrichtenquelle als *gedächtnislos* bezeichnet.
- ▷ Eine Quelle ist *gedächtnisbehaftet*, wenn die Ausgabe eines Zeichens von einem oder mehreren vergangenen Zeichen abhängt.

  Ist z.B. die Ausgabe einer Nachrichtenquelle das Ereignis des Würfelns mit einem idealen Würfel, dann ist diese Quelle gedächtnislos. Ein Sprachalphabet ist dagegen das Beispiel einer gedächtnisbehafteten Nachrichtenquelle, da die Wahrscheinlichkeit der Ausgabe eines Zeichens von den vorangegangenen Zeichen abhängt.

1.2 Wahrscheinlichkeiten

ZUR ERINNERUNG:

- ▷ Eine diskrete Zufallsvariable X mit Werten aus α_M ist eine Abbildung von einem Stichprobenraum (Menge aller möglichen Ereignisse eines Zufallsexperiments) auf die Menge α_M.
- ▷ Ein Ereignis ist jede Teilmenge von α_M.
- ▷ Das unmögliche Ereignis ist die leere Menge $\{\} \equiv \emptyset$, $P\{x_i \in \emptyset\} = 0$.
- ▷ Das sichere Ereignis ist α_M, $P\{x_i \in \alpha_M\} = 1$.

- ▷ Die Wahrscheinlichkeitsfunktion $p(x_i) = P\{X = x_i\}$ ist eine Abbildung $p : \alpha_M \to \mathbb{R}$ mit Eigenschaften $0 \leq p \leq 1$ und $\sum_{i=1}^{M} p(x_i) = 1$.
- ▷ Die Wahrscheinlichkeit eines Ereignisses x_i charakterisiert, wie häufig dieses Ereignis bei der Durchführung des Zufallsexperiments auftritt.

  Wenn die Wahrscheinlichkeit eines zufälligen Ereignissen unbekannt ist (bzw. nicht bestimmt werden kann), dann muss man das betreffende Ereignis genügend oft beobachten und die relative Häufigkeit als einen Näherungswerte für die Wahrscheinlichkeit nehmen.
- ▷ Für zwei statistisch unabhängige Zufallsvariablen X und Y (unabhängige Zufallsexperimente, bzw. unabhängig ausgegebene Quellen) gilt $p(x_i, y_j) = p(x_i) \cdot p(y_j)$ (Ergebnistupel $\equiv$ Verbundwahrscheinlichkeit).
- ▷ Zu einer Zufallsvariable X definiert man den Erwartungswert durch

$$E(X) = \sum_{i=1}^{M} x_i \cdot p(x_i)$$

  und die Varianz durch

$$\sigma^2(X) = \sum_{i=1}^{M} (x_i - E(X))^2 \cdot p(x_i).$$

- ▷ Die Standardabweichung von X, $\sigma(X)$, ist ein Maß für die Streuung der Werte.
- ▷ Wenn $E(X)$ und $\sigma^2(X)$ nicht bekannt sind, dann lässt sich durch eine Beobachtung (n Stichproben!) der Mittelwert $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ und die Varianz der Stichproben $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2$ berechnen. Bei einer großen Anzahl n von Stichproben der Zufallsvariable X gilt $\lim_{n\to\infty} \overline{x} = E(X)$ und $\lim_{n\to\infty} s^2 = \sigma^2(X)$.

BEISPIEL 1.1: Wie groß ist die Wahrscheinlichkeit eines Ereignisses (z.B. $x_i = 3$) bei einem idealen Würfel? Wie verändert sich diese Wahrscheinlichkeit, wenn der Würfen so manipuliert wird, dass 3 mit der doppelt so hohen Wahrscheinlichkeit auftritt? LÖSUNG: Beim idealen Würfel ist $p(x_i) = 1/6 \ \forall x_i$ und beim manipulierten $p(3) = 1/3$ und $p(x_i \neq 3) = 2/15$.

BEISPIEL 1.2: Wie groß ist die Wahrscheinlichkeit des Ereignisses A :„Augenzahl ist eine gerade Zahl" bei einem idealen Würfel? Wie groß ist die Wahrscheinlichkeit, dass zwei gerade Zahlen fallen, wenn zwei Spielwürfel gleichzeitig gewürfelt werden? LÖSUNG: Das Ereignis A tritt ein, wenn Zahlen $2, 4$ und 6 fallen. Bei einem idealen Würfel ist $p(A) = 3/6 = 1/2$ und bei zwei (unabhängigen!) Spielwürfeln $p(A) = \frac{1}{2} \cdot \frac{1}{2} = 1/4$.

BEISPIEL 1.3: Ein Verkäufer bekommt eine Lieferung von 100 Rechner vom gleichen Typ. In zwei unabhängigen Tests stellt er fest, dass 17 Rechner ein Problem mit der Festplatte haben, während 9 Rechner mit unvollständige Software ausgeliefert wurden. Wie groß die Wahrscheinlichkeit, dass bei einem zufällig ausgewählten Rechner nur das Festplattenproblem auftritt? Wie groß die Wahrscheinlichkeit, das bei dem ausgewählten Rechner keiner der Fehler auftreten?
LÖSUNG: Die Wahrscheinlichkeit für die Ereignisse
A : "Festplatte defekt" $p(A) = 17/100 = 0.17$,
$\overline{A}$: „Festplatte in Ordnung" $p(\overline{A}) = 1 - p(A) = 0.83$,
B : „Software nicht vollständig" $P(B) = 9/100 = 0.09$
B : „Software vollständig" $p(\overline{B}) = 1 - p(B) = 0.91$
Bei einem zufällig ausgewählten Rechner tritt nur das Festplattenproblem mit der Wahrscheinlichkeit $p(A) \cdot p(\overline{B}) = 0.15$ auf. Keiner der Probleme tritt mit der Wahrscheinlichkeit von $p(\overline{A}) \cdot p(\overline{B}) = 0.755$ auf.

Bedingte Wahrscheinlichkeiten

Ein Wissen über den Ausgang des Zufallsexperiments kann die Wahrscheinlichkeit der Zufallsvariable verändern.

BEISPIEL 1.4: Wie groß ist die Wahrscheinlichkeit des Ereignisses A : „Augenzahl ist eine gerade Zahl" bei einem idealen Würfel, wenn bekannt ist, dass das Ereignis B : "Augenzahl > 3" aufgetreten ist?
LÖSUNG: Weiß man, dass B aufgetreten ist, so gibt es für A nur noch drei mögliche Ereignisse, wovon zwei günstig sind ($x_i = 4$ und $x_i = 6$). Es ist $p(A|B) = 2/3$.

Die Wahrscheinlichkeit von A unter der Bedingung, dass B mit $p(B) \neq 0$ bereits aufgetreten ist, wird mit $p(A|B)$ bezeichnet und heißt die *bedingte Wahrscheinlichkeit* von A unter der Bedingung B.
Die totale Wahrscheinlichkeit für ein Ereignis A, bedingt durch paarweise unvereinbaren zufälligen Ereignisse B_i, $i = 1, .., n$, ist gegeben durch

$$p(A) = \sum_{i=1}^{n} p(A|B_i)p(B_i).$$

Sind zwei zufällige Ereignisse voneinander unabhängig, dann wird das Eintreten von einem nicht von dem anderen beeinflusst und $p(A|B) = p(A)$.
Die Wahrscheinlichkeit für das gleichzeitige Auftreten zweier Ereignisse A und B ist $p(A, B) = p(A|B) \cdot p(B) = p(B|A) \cdot p(A)$. Sind die Ereignisse unabhängig voneinander, dann gilt $p(A, B) = p(A) \cdot p(B)$.
Für zwei Ereignisse A und B mit $p(B) > 0$ lässt sich die Wahrscheinlichkeit von A unter der Bedingung, dass B eingetreten ist, durch die Wahrscheinlichkeit von B unter der Bedingung, dass A eingetreten ist, errechnen:

$$p(A|B) = \frac{p(A,B)}{p(B)} = \frac{p(B|A) \cdot p(A)}{p(B)} \qquad \text{(Satz vom Bayes).}$$

BEISPIEL 1.5: Gegeben seien drei Urnen vom Typ I, mit 2 weißen und 6 schwarzen Kugeln, und. eine Urne vom Typ II, mit einer weißen und 8 schwarzen Kugeln. Es wird zufällig aus einer Urne eine Kugel gezogen.
Wie groß ist die Wahrscheinlichkeit für das Ereignis B : "die gezogene Kugel ist weiß"?
LÖSUNG: Sei A_1 das Ereignis, die Urne vom Typ I wird gewählt und A_2 die Urne vom Typ II wird gewählt. Es wird eine Kugel aus eine Urne gezogen, daher ist $p(A_1) \cdot p(A_2) = 0$ und $p(B) = p(B|A_1) \cdot p(A_1) + p(B|A_2) \cdot p(A_2)$.
Mit $p(A_1) = 3/4$, $p(A_2) = 1/4$, $p(B|A_1) = 2/8$ und $p(B|A_2) = 1/9$ folgt $p(B) = 3/16 + 1/36 = 0.215$.

Die bedingte Wahrscheinlichkeit nach einer Beobachtung wird *a-posteriori* Wahrscheinlichkeit genannt.

BEISPIEL 1.6: Fortsetzung des vorherigen Beispiels: .
Wenn bekannt ist, das die gezogene Kugel weiß ist, wie groß ist die Wahrscheinlichkeit dass die gewählte Urne vom Typ I ist?
LÖSUNG: $p(A_1|B) = \dfrac{p(B|A_1) \cdot p(A_1)}{p(B)} = \frac{0.25 \cdot 0.75}{0.215} = 0.871.$

1.3 Informationsgehalt und Entropie

Umso unsicherer die Ausgabe eines Zeichens ist desto mehr Information über die Quelle beinhaltet diese Ausgabe. Die Überlegung, dass der Informationsgehalt eines Zeichens x_i umso größer ist, je kleiner seiner Wahrscheinlichkeit $p(x_i)$ ist, brachte Shannon 1948 dazu, den Informationsgehalt als eine Funktion $I(x_i)$ zu definieren, die folgende Eigenschaften besitzt:

- ▷ der Informationsgehalt $I(x_i)$ eines Zeichens x_i ist eine monoton fallende Funktion der Wahrscheinlichkeit $p(x_i)$
- ▷ der Informationsgehalt $I(x_i, y_i)$ zweier statistisch unabhängigen Zeichen x_i und y_i ist die Summe der Informationsgehalte der einzelnen Zeichen, $I(x_i) + I(y_i)$

Dazu führte er *bit* als Einheit für den Informationsgehalt eines Zeichens. Dabei soll ein Zeichen, das mit der Wahrscheinlichkeit $p(x_i) = 0.5$ auftritt, den Informationsgehalt von 1 bit haben.

Eine Funktion, die diese Anforderung erfüllt ist

$$I(x_i) = -\log_2 (p(x_i)) \quad \text{[bit]}$$

und wird als Informationsgehalt eines Zeichens x_i bezeichnet.

BEISPIEL 1.7: Wie groß ist der Informationsgehalt eines Ereignisses (z.B. $x_i = 3$) bei einem idealen Würfel? Wie verändert sich dieser Informationsgehalt wenn der Würfen so manipuliert wird, dass 3 mit der doppelt so hohen Wahrscheinlichkeit auftritt?
LÖSUNG: Beim idealen Würfel ist $I(x_i) = 2.5849$ bit $\forall x_i$ und beim manipulierten $I(3) = 1.585$ bit und $I(x_i \neq 3) = 2.9069$ bit.

Um zu wissen, wie viele Bits benötigt werden um nicht nur ein Zeichen, sondern eine Nachricht (Zeichenfolge) darzustellen, benötigen wir den Informationsgehalt der Nachricht.

Eine Nachrichtenquelle liefert im Mittel $E\{I(X)\} = \sum_{i=1}^{M} I(x_i)p(x_i)$ bit/Zeichen an Information.

Dieser mittlere Informationsgehalt $H(X) = -\sum_{i=1}^{M} p(x_i) \cdot \log_2(p(x_i))$ wird *Entropie* genannt.

Entropie ist ein Maß für den mittleren Informationsgehalt pro Zeichen einer Quelle.

BEISPIEL 1.8: Wie groß ist die Entropie eines idealen Würfels? Wie verändert sich die Entropie wenn der Würfen so manipuliert wird, dass 3 mit der doppelt so hohen Wahrscheinlichkeit auftritt?
LÖSUNG: Beim idealen Würfel ist $H(X) = 2.5849$ bit und beim manipulierten $H(X) = 2.4662$ bit.

Es sind also mindestens $H(X)$ Bits für die Darstellung/Übertragung eines Zeichens notwendig.

Es gilt (hier ohne Beweis): $0 \leq H(X) \leq \log_2(M)$.

Maximale Entropie besitzt eine Nachrichtenquelle, bei der die Zeichen des Quellenalphabets gleichwahrscheinlich sind. Es gilt $H(X) \leq H_{max} = \log_2(M)$ [bit]. Die maximale Entropie wird auch als Entscheidungsgehalt bezeichnet. In einem System mit zufälligen Ereignissen kann jedes Ereignis durch, im Mittel, H binäre Entscheidungen bestimmt werden. Soll z.B. das Resultat des Würfelns erraten werden, kann ob kleiner/größer oder ja/nein gefragt werden...
Minimale Entropie $H(x) = 0$ besitzt eine Nachrichtenquelle, bei der ein Zeichen mit der Wahrscheinlichkeit $p(x_i) = 1$ auftritt.
Die *Redundanz* einer Nachrichtenquelle mit der Entropie $H(X)$ ist $R(X) = H_{max} - H(X)$.

Die Entropie einer **gedächtnislosen Binärquelle** bei der $x_1 = 0$ mit Wahrscheinlichkeit p und $x_2 = 1$ mit Wahrscheinlichkeit $1 - p$ auftritt, ist

$$H(X) = -p\log_2(p) - (1-p)\log_2(1-p) \qquad \text{(Shannon'sche Funktion)}$$

Treten 0 und 1 gleich-wahrscheinlich auf, dann ist $H(X) = H_{max} = -\log_2(0.5) = 1$ bit. Man benötigt also 1 Bit pro Zeichen und N Bits um eine Nachricht der Länge N zu übertragen.

Ist z.B. $p(x_1) = 0.89$, dann ist $p(x_2) = 0.11$ und die Entropie beträgt $H(X) = 0.5$bit.

Die Buchstabenhäufigkeit in der deutschen Sprache ist ungleichmäßig. Beispielsweise ist der Buchstabe E sieben Mal so häufig wie M oder O, was zu Redundanz im Alphabet führt. Die Entropie des deutschen Alphabets beträgt $H(X) \sim 4.063$ bit. Die Redundanz ist somit $R(x) = \log_2(26) - 4.063 = 0.37$ bit.

Die Auflösung eines Ereignisses in Teilereignisse führt zu einer Zunahme der Entropie.
Betrachten wir z.B. $\underline{p} = (p_1, p_2, ..., p_N)$ und $p_N = q_1 + q_2$. Zu zeigen ist, dass

$$H_1 = -\sum_{i=1}^{N-1} p_i \log_2 p_i - p_N \log_2 p_N < -\sum_{i=1}^{N-1} p_i \log_2 p_i - q_1 \log_2 q_1 - q_2 \log_2 q_2 = H_2.$$

Der Logarithmus eine monoton steigende Funktion ist, gilt
$p_N \log_2 p_N = (q_1+q_2)\log_2(q_1+q_2) = q_1 \log_2(q_1(1+\frac{q_2}{q_1}))+q_2 \log_2((\frac{q_1}{q_2}+1)q_2) > q_1 \log_2(q_1)+q_2 \log_2(q_2)$.

1.4 Gedächtnisbehaftete Quellen

Bisher hatten wir gedächtnisfreie Quellen betrachtet, deren Ausgangssymbole unabhängige Zufallsvariablen waren.

Bei einer gedächtnisbehafteten Quelle hängt die Auftrittswahrscheinlichkeit der Zeichen von den zuvor ausgegebenen Zeichen ab. Suchen wir z.B. ein passendes Modell für eine Sprachquelle, dann muss die Wahrscheinlichkeit für jedes Symbol von den vorhergehenden Symbolen abhängen.

Einer Nachrichtenquelle mit dem Gedächtnis der Länge L wird als eine MARKOV-QUELLE L-ter Ordnung bezeichnet. Das aktuelle Zeichen hängt statistisch von den L zuvor ausgegebenen Zeichen ab. Das Auftreten des Zeichens $X(t) = x_j$ zum Zeitpunkt t ist beschrieben durch die bedingte Wahrscheinlichkeit $P^{(t)}\{X(t) = x_j | X(t-1) = x_{i_1}, ..., X(t-L) = x_{i_L}\}$, wobei $x_j, x_i, ... \in \alpha_M$, d.h. beliebige Zeichen der Quelle sind.

Ist diese Wahrscheinlichkeit unabhängig vom Zeitpunkt des Auftretens t, dann wird die Quelle als *homogen* bezeichnet. Im folgenden betrachten wir nur homogene Markov Quellen und lassen t in der Bezeichnung der bedingten Wahrscheinlichkeiten weg.

Da jedes beliebige ausgegebene Zeichen von L vorherigen Zeichen abhängt, beschreiben wir diese Abhängigkeit mit der relativen Zeit k. Die Zeitschritte $..., k-1,\ k,\ k+2, ...$ entsprechen dem Takt, mit dem die Quelle die Zeichen ausgibt.
Die L vorherige Zeichen definieren den Zustand $S(k-1) = (x_{i_1}, ..., x_{i_L})$ in dem sich die Quelle zum Zeitpunkt $k-1$, vor dem Aussenden des aktuellen Zeichens x_j zum Zeitpunkt k, befindet. Da die Quelle homogen ist, hängt der aktuelle Zeitpunkt k nicht von der absoluten Zeit t.

Mit der bedingten Wahrscheinlichkeit $p(x_j|x_{i_1}, ..., x_{i_L}) = p(x_j|S(k-1))$ tritt das Zeichen x_j auf und die Quelle geht in den Zustand $S(k) = (x_j, x_{i_1}, ..., x_{i_L-1})$ über. Die Wahrscheinlichkeiten $p(S(k)|S(k-1))$ werden auch *Übergangswahrscheinlichkeiten* genannt.

Markov Quellen können somit durch

- ▷ eine endliche Zustandsmenge $\mathcal{S} = \{s_1, s_2, ..., s_Z\}$ und
- ▷ eine Übergangsmatrix $P_S = [p_{ij}]$, $i, j = 1, 2, ..., Z$, mit Zustandsübergangswahrscheinlichkeiten $p_{ij} = p(s_j|s_i)$, $p_{ij} \geq 0$

beschrieben werden. Dabei ist s_i einer von $Z = M^L$ möglichen Zuständen. In jedem Zustand muss es einen Folgezustand geben, d.h. für die Summe der Elemente in jeder Zeile der Matrix P_S gilt $\sum_{j=1}^{Z} p(s_j|s_i) = \sum_{j=1}^{Z} p_{ij} = 1$.

BEISPIEL 1.9: Eine Markov-Quelle mit 3 Zuständen, s_1, s_2 und s_3, ist charakterisiert durch die Übergangsmatrix
$P_S = \begin{pmatrix} \frac{1}{3} & \frac{2}{3} & 0 \\ \frac{1}{4} & 0 & \frac{3}{4} \\ \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}$. Die Quelle ist binär, d.h. die möglichen Ausgabezeichens, je nach dem in welchem Zustand sich die Quelle befindet, sind 0 und 1.
Eine solche Quelle kann mit einem Zustandsdiagramm beschrieben werden. In einem Zustandsdiagramm stellen Knoten die Zustände dar und die Verbindungen bezeichnen die Übergänge zwischen den Zuständen.
Wie sieht das Zustandsdiagramm der Markov Quelle aus?

Eine stochastische Folge von Zuständen einer homogener Markov Quelle, bei der der aktuelle Zustand nur von dem vorherigen Zustand, aber nicht von weiteren Zuständen in der Vergangenheit abhängt, wird als endliche homogene Markov Kette bezeichnet.

Die Wahrscheinlichkeit, dass sich die Quelle am Anfang, also zum Zeitpunkt $k = 0$, im Zustand $s_i^{(0)}$ befindet sei p_i. Der Vektor $\underline{p}^{(0)} = (p_1^{(0)}, p_2^{(0)}, ..., p_Z^{(0)})$ mit Wahrscheinlichkeiten aller Zustände zum Zeitpunkt $k = 0$ heißt Zustandswahrscheinlichkeitsverteilung der Markov-Quelle zur Zeit 0. Analog entspricht der Vektor $\underline{p}^{(k)} = (p_1^{(k)}, p_2^{(k)}, ..., p_Z^{(k)})$ zur Zeit k der momentanen Zustandswahrscheinlichkeitsverteilung der Markov-Quelle. Im Gegensatz zu den Übergangswahrscheinlichkeiten sind die Zustandswahrscheinlichkeiten zeitlich veränderlich. Die Wahrscheinlichkeit $p_j^{(k)}$ ergibt sich aus $p_j^{(k)} = p_1^{(k-1)} p_{1j} + p_2^{(k-1)} p_{2j} + ... + p_Z^{(k-1)} p_{Zj}$, bzw. in der Matrix Form

$$\underline{p}^{(k)} = \underline{p}^{(k-1)} \cdot P_S.$$

Nach n Zeitschritten folgt $\underline{p}^{(k+n)} = \underline{p}^{(k+n-1)} P_S = \underline{p}^{(k+n-2)} (P_S)^2 = ... = \underline{p}^k (P_S)^n$. In der Regel geht man von einem Anfangszustand $s_i^{(0)}$ an. Dann ist $\underline{p}^{(n)} = \underline{p}^{(0)} (P_S)^n$.

BEISPIEL 1.10: Die Markov-Quelle ist charakterisiert durch die Übergangsmatrix P_S aus dem obigen Beispiel. Nehmen wir an, dass die Wahrscheinlichkeiten der Anfangszustände $p^{(0)}(s_1) = p^{(0)}(s_2) = p^{(0)}(s_3) = 1/3$ sind. Im nächsten Zeitschritt ergibt sich folgende Verteilung:
$p^{(1)}(s_1) = \frac{1}{3}\frac{1}{3} + \frac{1}{3}\frac{1}{4} + \frac{1}{3}\frac{1}{2} = \frac{13}{36}$,
$p^{(1)}(s_2) = \frac{1}{3}\frac{2}{3} + 0 + \frac{1}{3}\frac{1}{2} = \frac{14}{36}$,
$p^{(1)}(s_3) = 0 + \frac{1}{3}\frac{3}{4} + 0 = \frac{9}{36}$.
Weitere Berechnung ergibt: $(P_S)^8 = \begin{pmatrix} 0.3485 & 0.3720 & 0.2794 \\ 0.3491 & 0.3721 & 0.2788 \\ 0.3489 & 0.3722 & 0.2789 \end{pmatrix}$.

Das Beispiel zeigt, dass mit der Zeit die Wahrscheinlichkeit für einen Zustand zu einer Konstante konvergiert, also unabhängig vom Anfangszustand ist. Wenn so eine asymptotische Wahrscheinlichkeitsverteilung $\underline{w} = (w_1, w_2, ..., w_Z)$ existiert, dann wird sie *stationäre* Verteilung genannt.
Für eine homogene Markov-Kette mit der stationären Verteilung gilt $\underline{w} \cdot P_S = \underline{w}$, d.h. die Wahrscheinlichkeit der Zustände ändert sich von einem Zeitpunkt zum nächsten nicht. Konvergiert die Folge der Wahrscheinlichkeitsvektoren $\underline{p}^{(k)}$ gegen eine feste Verteilung $\underline{w}$, dann spricht man von einer *regulären* Markov-Kette.

Die stationäre Verteilung kann als Lösung des Gleichungssystems $\underline{w} \cdot P_S = \underline{w}$ (linear abhängig!) und der Bedingung $w_1 + w_2 + w_3 = 1$ berechnet werden.

BEISPIEL 1.11: Die *stationäre* Wahrscheinlichkeitsverteilung (Zustandsverteilung) der Markov-Quelle aus dem obigen Beispiel ist $w_1 = 0.3488$, $w_2 = 0.3721$und $w_3 = 0.2791$.

Markov Quelle 1. Ordnung Bei einer Markov-Quelle erster Ordnung ist die Wahrscheinlichkeit für das Ereignis $X = x_j$ zu einem Zeitpunkt nur vom dem davor ausgegebenen Zeichen x_i abhängig. Ein Zustand ist nur durch ein Zeichen definiert. Bei einer homogener Quelle ist die Übergangswahrscheinlichkeit $p(x_j|x_i) = p_{ij}$ unabhängig von der absoluten Zeit t. Die Quelle kann mit einer Markov-Matrix P_α beschrieben werden,

$$P_\alpha = \begin{pmatrix} p(x_1|x_1) & p(x_2|x_1) & \cdots & p(x_M|x_1) \\ p(x_1|x_2) & \ddots & & \vdots \\ \\ p(x_1|x_M) & \cdots & & p(x_M|x_M) \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1M} \\ p_{21} & \ddots & & \vdots \\ \\ p_{M1} & \cdots & & p_{MM} \end{pmatrix}.$$

wobei für jede Zeile gilt $\sum_{j=1}^{M} p(x_j|x_i) = \sum_{j=1}^{M} p_{ij} = 1$.

Die Wahrscheinlichkeit für das Ereignis x_j ist gegeben durch

$$p^{(k)}(x_j) = \sum_{i=1}^{M} p(x_j|x_i) \cdot p^{(k-1)}(x_i).$$

Betrachten wir als Beispiel eine binäre Markov-Quelle 1. Ordnung mit $M = 2$ mit der Übergangsmatrix $P_\alpha = \begin{pmatrix} 1 - p_{12} & p_{12} \\ p_{21} & 1 - p_{21} \end{pmatrix}$. Die möglichen Zustände der Quelle sind x_1 und x_2 und somit muss $p(x_1) + p(x_2) = 1$ gelten.

Befindet sich die Quelle zum einem Zeitpunkt k im Zustand x_1 mit der Wahrscheinlichkeit $p^{(k)}(x_1)$, dann lässt sich die Quelle durch den Zustands- bzw. Auftrittswahrscheinlichkeitsvektor $\underline{p}^{(k)} = (p^{(k)}(x_1), 1 - p^{(k)}(x_1))$ beschreiben. Mit Hilfe der Übergangsmatrix berechnen wir die Wahrscheinlichkeitsverteilung zum Zeitpunkt $k + 1$,

$\underline{p}^{(k+1)} = \underline{p}^{(k)} \cdot P_\alpha = (p^{(k+1)}(x_1), p^{(k+1)}(x_2))$.

Es folgt
$p^{(k+1)}(x_1) = (1 - p_{1,2})\, p^{(k)}(x_1) + p_{2,1}\left(1 - p^{(k)}(x_1)\right)$
und
$p^{(k+1)}(x_2) = p_{1,2}\left(1 - p^{(k)}(x_2)\right) + (1 - p_{2,1})p^{(k)}(x_2)$.

BEISPIEL 1.12: Setzen wir in das Beispiel die Werte $P_\alpha = \begin{pmatrix} 0.1 & 0.9 \\ 0.3 & 0.7 \end{pmatrix}$ und nehmen an, dass sich die Quelle am Anfang in dem Zustand x_1 befindet, d.h. $\underline{p}^{(0)} = (1, 0)$. Der Wahrscheinlichkeitsverteilungsvektor der Folgezustände ändert sich folgendermaßen:

$$\begin{array}{ll} \underline{p}^{(1)} = (0.1, 0.9) & \underline{p}^{(4)} = (0.2512, 0.7488) \\ \underline{p}^{(2)} = (0.28, 0.72) & \underline{p}^{(5)} = (0.24976, 0.75024) \\ \underline{p}^{(3)} = (0.244, 0.756) & ... \end{array}$$

Die Wahrscheinlichkeitsverteilung konvergiert auch hier zu den festen Werten. Diese können wir berechnen indem wir

$p^{(k+1)}(x_i) = p^{(k)}(x_i) = w(x_i)$ für $i = 1, 2$, setzen. Dadurch erhalten wir *stationäre* Verteilung $w(x_1) = \dfrac{p_{2,1}}{p_{1.2} + p_{2,1}}$ und $w(x_2) = \dfrac{p_{1,2}}{p_{1.2} + p_{2,1}}$.

BEISPIEL 1.13: In dem konkreten von vorhin Beispiel erhalten wir die Werte $w(x_1) = \frac{0.3}{0.3+0.9} = 0.25$ und $w(x_2) = \frac{0.9}{0.3+0.9} = 0.75$.

Im Falle der symmetrischen Übergangswahrscheinlichkeiten, $p_{1,2} = p_{2,1}$, folgt die Gleichheit der stationären Zustandswahrscheinlichkeiten, $w(x_1) = w(x_2) = 1/2$.

Die Entropie als Maß für die Unbestimmtheit eines Systems ist hier durch die Unbekanntheit des momentanen Zustandes und der Ungewissheit welcher der möglichen Übergänge eintreten wird gegeben.

Die Ungewissheit über die Übergangsmöglichkeiten ausgehend von einem Zustand x_i kann wird mit der, vom Zustand x_i abhängigen Entropie

$$H(X|S = x_i) = -\sum_{j=1}^{M} p(x_j|x_i) \cdot \log_2\left(p(x_j|x_i)\right)$$

bestimmen. Die Entropie der homogenen und regulären Markov Quelle 1. Ordnung entspricht dem Erwartungswert über die Menge aller Zustände $S = x_i$,

$$\begin{aligned} H(X) &= \sum_{i=1}^{M} H(X|S = x_i) \cdot w(x_i) = -\sum_{i=1}^{M}\sum_{j=1}^{M} p(x_j|x_i) \cdot \log_2\left(p(x_j|x_i)\right) \cdot w(x_i) \\ &= -\sum_{i=1}^{M}\sum_{j=1}^{M} p_{ij} \cdot \log_2\left(p_{ij}\right) \cdot w(x_i). \end{aligned}$$

Das Ergebnis ist als mittlere Anzahl von Bits pro Zustand der Quelle zu interpretieren. Bei der Markov Quellen 1. Ordnung entspricht das wieder der mittleren Anzahl von Bits pro Zeichen der Quelle.
Betrachten wir wieder als Beispiel die Markov-Quelle 1. Ordnung mit $M = 2$ und der Übergangsmatrix $P_\alpha = \begin{pmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{pmatrix}$ und stationären Wahrscheinlichkeitsverteilung $\underline{w} = (\frac{p_{21}}{p_{12}+p_{21}}, \frac{p_{12}}{p_{12}+p_{21}})$.

Die Entropie der Quelle resultiert in $H(X) = -\sum_{i=1}^{M}\sum_{j=1}^{M} p_{ij} \cdot \log_2(p_{ij}) \cdot w(x_i)$

$$= -\frac{p_{1,1} \cdot p_{2,1}}{p_{1.2}+p_{2,1}} \cdot \log_2(p_{1,1}) - \frac{p_{1,2} \cdot p_{2,1}}{p_{1.2}+p_{2,1}} \cdot \log_2(p_{1,2}) - \frac{p_{2,1} \cdot p_{1,2}}{p_{1.2}+p_{2,1}} \cdot \log_2(p_{2,1}) - \frac{p_{2,2} \cdot p_{1,2}}{p_{1.2}+p_{2,1}} \cdot \log_2(p_{2,2}).$$

Sind die Übergangswahrscheinlichkeiten symmetrisch, $p_{1,2} = p_{2,1} = q$, und somit $p_{1,1} = p_{2,2} = 1-q$ und $w(x_1) = w(x_2) = 1/2$, dann folgt für die Entropie

$$\begin{aligned} H(X) &= -\frac{1}{2}(1-q) \cdot \log_2(1-q) - \frac{1}{2}q \cdot \log_2(q) - \frac{1}{2}q \cdot \log_2(q) - \frac{1}{2}(1-q) \cdot \log_2(1-q) \\ &= -q \cdot \log_2(q) - (1-q) \cdot \log_2(1-q) \quad \text{[bit]}. \end{aligned}$$

Die resultierende Entropie entspricht der Shannon'schen Formel für die Entropie einer gedächtnislosen binären Quelle. Bei der letzteren ergibt sich für gleiche Auftrittswahrscheinlichkeiten, $p(x_1) = p(x_2) = 1/2$, $H(X) = 1$ bit. Die Entropie einer gedächtnisbehafteten binären Quelle mit symmetrischen Übergangswahrscheinlichkeiten ist immer kleiner als die Entropie einer gedächtnislosen binären Quelle mit denselben Auftrittswahrscheinlichkeiten. Das zeigt, dass die Unbestimmtheit über ein Zeichen größer ist, wenn man aus dem vorigen Zeichen keine Information erhält.

BEISPIEL 1.14: Eine reguläre binäre Markov-Quelle 1.Ordnung mit Übergangswahrscheinlichkeiten $p_{1,2} = p_{2,1} = 1/3$ hat die Entropie $H(X) = 0.918$ bit.

Maximale Entropie folgt für $q = 1/2$, was der Fall einer gedächtnislosen Binärquelle ist.

BEISPIEL 1.15: Eine reguläre ternäre Markov-Quelle 1.Ordnung mit $M = 3$ hat die Übergangsmatrix $P_\alpha = \begin{pmatrix} p_{1,1} & p_{1,2} & p_{1,3} \\ p_{2,1} & p_{2,2} & p_{2,3} \\ p_{3,1} & p_{3,2} & p_{3,3} \end{pmatrix} = \begin{pmatrix} 0.1 & 0.2 & 0.7 \\ 0.1 & 0.8 & 0.1 \\ 0.2 & 0.4 & 0.4 \end{pmatrix}$. Die stationäre Verteilung soll ausgehend vom Anfangszustand $S(0) = x_1$ berechnet werden.
Mit der Berechnung $\underline{p}^{(t)} = \underline{p}^{(t-1)} P_\alpha$ folgt nach einigen Schritten $\underline{w} = (0.125, 0.625, 0.25)$. Die Entropie der einzelnen Zustände ist $H(X|S = x_1) = 1.1568$ bit, $H(X|S = x_2) = 0.9219$ bit und $H(X|S = x_3) = 1.5219$ bit, und die Entropie der Quelle $H(X) = 1.1$ bit. Die Entropie einer gedächtnislosen ternären Quelle mit denselben Auftrittswahrscheinlichkeiten wäre $H(X) = 1.2988$ bit.
Maximale Entropie hat die gedächtnislose ternäre Quelle mit gleichwahrscheinlichen Symbolen, $H_{max} = 1.5849$ bit.

Da die Entropie einer Markov-Quelle kleiner ist als die, der vergleichbaren unabhängigen Quelle, führt die Anwendung des Markov Modells, wenn realisierbar, zu einer Reduzierung der zu verarbeitenden bzw. zu speichernden Information.

1.5 Verbundquellen

Betrachten wir gleichzeitig zwei Quellen X und Y, wobei die Ereignisse innerhalb jeder Einzelquelle voneinander unabhängig sind, aber die Ereignisse beider Quellen voneinander abhängen. Zufallsvariablen X und Y nehmen Werte aus endlichen Zeichenalphabeten $\{x_i\}_{1\leq i\leq M}$ und $\{y_j\}_{1\leq j\leq Q}$ an. Das Auftreten von zwei Ereignissen, $X = x_i, Y = y_j$, bezeichnet man als Verbundereignis (x_i, y_j), das mit der Verbundwahrscheinlichkeit $p(x_i, y_j)$ auftritt. Dabei ist die Verbundquelle (X, Y) ebenfalls eine Zufallsvariable, die $M \cdot Q$ unterschiedliche Werte annehmen kann.

Für die Verbundwahrscheinlichkeit gilt

$$p(x_i, y_j) = p(x_i) \cdot p(y_j|x_i) = p(y_j) \cdot p(x_i|y_j).$$

Die Auftrittswahrscheinlichkeiten $\underline{p}_X = (p(x_1), p(x_2), ..., p(x_M))$ und $\underline{p}_Y = (p(y_1), p(y_2), ..., p(y_Q))$ lassen sich auch aus Verbundwahrscheinlichkeiten mit $p(x_i) = \sum_{j=1}^{Q} p(x_i, y_j)$ und $p(y_j) = \sum_{i=1}^{M} p(x_i, y_j)$ berechnen. Die bedingte Wahrscheinlichkeiten lassen sich mit der REGEL VON BAYES berechnen:

$$p(x_i|y_j) = \frac{p(x_i, y_j)}{p(y_j)} \qquad \text{und} \qquad p(y_j|x_i) = \frac{p(x_i, y_j)}{p(x_i)}.$$

Der Informationsgehalt eines Verbundzeichenpaares (x_i, y_j), welches mit der Wahrscheinlichkeit $p(x_i, y_j)$ auftritt, ist gegeben durch $I(x_i, y_j) = -\log_2 (p(x_i, y_j))$.

Somit ist die Entropie einer Verbundquelle gegeben durch

$$H(X, Y) = E\{I(X, Y)\} = -\sum_{i=1}^{M} \sum_{j=1}^{Q} p(x_i, y_j) \cdot \log_2 (p(x_i, y_j)).$$

Durch Einsetzen von obigen Formeln folgt

$$\begin{aligned} H(X, Y) &= -\sum_{i=1}^{M} \sum_{j=1}^{Q} p(y_j|x_i)p(x_i) \cdot \log_2 (p(y_j|x_i)p(x_i)) \\ &= -\sum_{i=1}^{M} p(x_i) \cdot \log_2 (p(x_i)) - \sum_{i=1}^{M} \sum_{j=1}^{Q} p(y_j|x_i)p(x_i) \cdot \log_2 (p(y_j|x_i)) \\ &= H(X) + H(Y|X). \end{aligned}$$

Der erste Term entspricht der Entropie einer gedächtnisfreien Quelle, und der zweite ähnelt der Entropie einer gedächtnisbehafteten Quelle. Da aber X und Y hier unterschiedliche Quellen beschreiben,

entspricht $H(Y|\mathrm{X})$ der bedingten Entropie, also der Ungewissheit über Y die bleibt, nachdem man X beobachtet hat.

Die bedingten Entropie $H(Y|X)$ berechnet sich durch das Mitteln über das Sendealphabet $\{x_i\}_{1\leq i\leq M}$:

$$\begin{aligned} H(Y|X) &= \sum_{i=1}^{M} p(x_i)\cdot H(Y|x_i) = -\sum_{i=1}^{M}\sum_{j=1}^{Q} p(x_i)\cdot p(y_j|x_i)\cdot \log_2\left(p(y_j|x_i)\right) \\ &= -\sum_{i=1}^{M}\sum_{j=1}^{Q} p(x_i,y_j)\cdot \log_2\left(p(y_j|x_i)\right). \end{aligned}$$

Ähnlich lässt sich zeigen, dass auch

$$H(X,Y) = H(Y) + H(X|Y)$$

gilt.

Für beliebige Zufallsvariablen X und Y gilt

$$H(X|Y) \leq H(X),$$

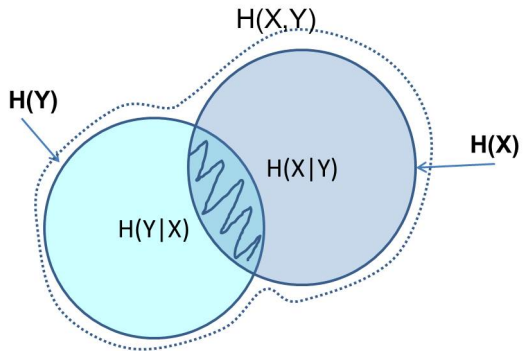
wobei die Gleichheit gegeben ist, wenn X und Y unabhängig sind. Auch wenn der mathematische Beweis nicht allzu schwer ist, lässt sich diese Ungleichheit leicht argumentieren: Wenn die Zufallsvariablen X und Y nicht unabhängig voneinander sind, dann kann die Kenntnis über Y die Ungewissheit über X nur verringern.

Die Entropie einer Verbundquelle kann mit einem VENN-Diagramm, wie in Abbildung rechts, dargestellt werden. Aus der Abbildung ist ersichtlich das folgendes gilt:

$$H(X|Y) \leq H(X)$$

und

$$H(Y|X) \leq H(Y).$$



Als Grenzfälle gelten

- ▷ vollständige Unabhängigkeit: $p(y_j|x_i) = p(y_j) \Rightarrow H(Y|X) = H(Y)$ und $H(X,Y) = H(X) + H(Y)$;
- ▷ vollständige Abhängigkeit: $p(y_j|x_i) = 1 \Rightarrow H(Y|X) = 0$ und $H(X,Y) = H(X)$.

Die Reduktion der Ungewissheit über X durch die Kenntnis von Y, $H(X) - H(X|Y)$ ist ein Maß dafür, wie viel Information sie übereinander beinhalten und wird *wechselseitige Information* genannt. Im Zusammenhang mit dem Nachrichtenkanal im nächsten Kapitel wird sie auch Transinformation genannt und mit $T(X,Y)$ beschrieben. Die *wechselseitige Information* ist symmetrisch, d.h.

$$T(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X).$$

BEISPIEL 1.16: Betrachten wir wieder das Würfeln, eine Zufallsvariable X, die einen Werte aus der Menge $\{1, 6\}$ annehmen kann, wobei die Zufallsvariable Y als Ereignis gerade oder ungerade sein kann.
Es ist also $H(X) = ld6 = 2.58$bit und $H(Y) = 1$bit. Wenn das Ereignis y bekannt ist, dann reduziert sich die Ungewissheit über X zu $H(X|Y) = ld3$bit. Ist aber das Ereignis x bekannt, dann ist $H(Y|X) = 0$.
Die Verbund-Zufallsvariable (X, Y) kann 12 Werte annehmen, wobei 6 nie auftreten können. Somit ist die Entropie $H(X, Y) = ld6$bit
Das gleiche Ergebnis folgt mit den Gleichungen:
$H(X, Y) = H(Y) + H(X|Y) = 1 + ld3 = ld2 + ld3 = ld6$bit
oder auch mit
$H(X, Y) = H(X) + H(Y|X) = 2.58 + 0 = ld6$bit
Die *wechselseitige Information* zwischen X und Y berechnet sich zu
$T(X, Y) = H(X) - H(X|Y) = ld6 - ld3 = ld2 = 1$bit, bzw.
$T(X, Y) = H(Y) - H(Y|X) = 1 - 0 = 1$bit.
Durch die Kenntnis von X (bzw. Y) reduzieren wir die Ungewissheit über Y (X) um 1bit.

BEISPIEL 1.17: Gegeben seien zwei diskrete Quellen mit folgenden Verbundwahrscheinlichkeiten:

$$P_{(X,Y)} = \begin{pmatrix} p(x_1, y_1) & \dots & p(x_1, y_3) \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ p(x_4, y_1) & \dots & p(x_4, y_3) \end{pmatrix} = \begin{pmatrix} \frac{1}{8} & 0 & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{32} & \frac{1}{32} \\ 0 & \frac{1}{8} & 0 \\ \frac{1}{8} & \frac{1}{8} & \frac{1}{4} \end{pmatrix}.$$ Zu berechnen sind

1. Einzelwahrscheinlichkeit $p(x_i)$ und $p(y_i)$,
2. bedingte Wahrscheinlichkeiten $p(y_j|x_i)$,
3. Entropien $H(X)$, $H(Y)$, $H(Y|X)$ und $H(X, Y)$.

LÖSUNG:

1.) $p(x_1) = \frac{1}{4}$, $p(x_2) = p(x_3) = \frac{1}{8}$, $p(x_4) = \frac{1}{2}$. 2.) $P_{ch} = \begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ 0 & 1 & 0 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \end{pmatrix}$.

3.) $H(X) = 1.75$bit, $H(Y) = 1.57$bit, $H(Y|X) = 1.19$bit , $H(X, Y) = 2.94$bit.

BEMERKUNG: Mit Verbundquellen lassen sich Nachrichtenkanäle beschreiben und modellieren, wobei Transmission eines Nachrichtenkanals beschreibt wie viel Information über einen gestörten Kanal übertragen werden kann, dazu aber mehr im nächsten Kapitel.

Die Erweiterung auf N Quellen bzw. einer Zufallssequenz mit N Komponenten ergibt die Kettenregel für die Verbundentropie:

$$H(X_1, X_2, ..., X_N) = H(X_1) + H(X_2|X_1) + ... + H(X_N|X_1, X_2, ..., X_{N-1})$$

gilt. Da Entropie als Maß für die Ungewissheit einer Quelle durch Bedingungen nicht zunehmen kann, gilt auch

$$H(X_1) \geq H(X_1|X_2) \geq H(X_1|X_2, X_3)...$$

Hauptsatz der Datenverarbeitung

Gegeben seien zwei seriell verkettete Datenprozessoren mit dem Eingang vom ersten X und dem Ausgang vom zweiten Z. Der Ausgang Y vom ersten Prozessor stellt den Eingang vom zweiten Prozessors dar. Der Ausgang Z des zweiten Prozessors kann nur indirekt vom Eingang X über Y beeinflusst werden, d.h. $P(z|x, y) = P(z|y)$,$\forall x, y, z$. Für solche seriell verkettete Zufallsvariablen X, Y und Z gilt

$$T(X, Z) = H(X) - H(X|Z) \leq H(X) - H(X|Y, Z) = H(X) - H(X|Y) = T(X, Y)$$

und

$$T(X, Z) = H(Z) - H(Z|X) \leq H(Z) - H(Z|X, Y) = H(Z) - H(Z|Y) = T(Y, Z)$$

Aus $T(X, Z) \leq T(X, Y)$ und $T(X, Z) \leq T(Y, Z)$ folgt der Hauptsatz der Datenverarbeitung, der besagt, dass durch eine solche sequentielle Datenverarbeitung der Informationsgewinn nicht erhöht werden kann. Das gleiche gilt auch für die serielle Datenübertragung. Der Satz gilt allerdings nicht, wenn eine a-priori Information in die Datenverarbeitung einfließt.

2 Nachrichtenkanal und Kanalcodierung

In Kommunikationssystemen werden oft Vorgänge beobachtet, um Schlüsse auf andere Ereignisse zu ziehen. Ein Nachrichtenkanal entspricht auch einer Verbundquelle die Zeichenpaare (X, Y) ausgibt. Dabei beschreibt X das Ereignis, das von einer Nachrichtenquelle ausgesandt und über einen Nachrichtenkanal übertragen wird, und Y das am Kanalausgang beobachtete Ereignis. Somit wird das Ereignis der Quelle Y immer von einem vorausgegangenen Ereignis der Quelle X bestimmt. Dabei löst z.B. das Ereignis $X = x_i$ an der Quelle X das Ereignis $Y = y_j$ in der Quelle Y mit der bedingten Wahrscheinlichkeit $p(y_j|x_i)$ aus.

2.1 Nachrichtenkanal als Verbundquelle

Nachdem an der Quelle X ein Ereignis stattfindet, tritt am Ausgangs des Nachrichtenkanals, der hier als Quelle Y betrachtet wird, ein bedingtes Ereignis mit der Wahrscheinlichkeit $p(y_j|x_i)$ auf. Erfolgt die Übertragung über den Nachrichtenkanal fehlerfrei, dann ist $Y = X$. Bei einer fehlerbehafteten Übertragung ist $Y \neq X$. Der Empfänger der Nachricht kann die Entscheidung, welches X gesendet worden ist, treffen, indem er Y beobachtet.

Wenn die Übertragung vom bekannten Sendezeichen x_i über einen Kanal fehlerbehaftet ist, interessiert uns der Informationsgehalt des Kanalausgangs $Y \neq X$.

Die über den Nachrichtenkanal übertragene Information ist

$$T(X, Y) = H(Y) - H(Y|X),$$

also die *wechselseitige Information* zwischen X und Y. Diese beschreibt die Information über Y, die aus der Kenntnis von X gewonnen wird. Dabei beschreibt die bedingte Entropie $H(Y|X) = H(X, Y) - H(X)$ die Unsicherheit, die durch die Störung auf dem Kanal verursacht wird und enthält keine Information über X. Analog beschreibt $H(X|Y) = H(X, Y) - H(Y)$ die verbleibende Unsicherheit über X wenn Y beobachtet wird.

Es gilt

$$\begin{aligned} T(X, Y) &= H(Y) - H(Y|X) = H(X, Y) - H(X|Y) - (H(X, Y) - H(X)) \\ &= H(X) - H(X|Y) = T(Y, X). \end{aligned}$$

Mit dem Ausdruck $T(Y, X) = H(X) - H(X|Y)$ wird die Information beschrieben, die man über das Ereignis X enthält, indem man das Ereignis Y beobachtet. Da diese Information symmetrisch ist, d.h. $T(X, Y) = T(Y, X)$, wird sie auch *Transinformation* des Nachrichtenkanals genannt.

Die Transinformation ist die Informationsmenge, die in Mittel durch ein Kanalzeichen vom Sender zum Empfänger übertragen wird. Aus $H(X|Y) \leq H(X)$, mit dem Gleichheitszeichen wenn X und Y unabhängig sind, folgt dass $T(X,Y) \geq 0$. Man kann also die wechselseitige Information auffassen, als ein Maß für die statistische Abhängigkeit von X und Y.

Bei einem Nachrichtenkanal beschreiben bedingte Wahrscheinlichkeiten $p_{i,j} \equiv p(y_j|x_i)$ die Übergangswahrscheinlichkeiten des Kanals und P_{ch} die Kanalmatrix mit

$$P_{ch} = \begin{pmatrix} p(y_1|x_1) & p(y_2|x_1) & \cdots & p(y_Q|x_1) \\ p(y_1|x_2) & \ddots & & \vdots \\ & & & \\ p(y_1|x_M) & \cdots & & p(y_Q|x_M) \end{pmatrix} = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,Q} \\ p_{2,1} & \ddots & & \vdots \\ & & & \\ p_{M,1} & \cdots & & p_{M,Q} \end{pmatrix}.$$

Aufgrund von der Formel $p(y_j) = \sum_{i=1}^{M} p(y_j|x_i) \cdot p(x_i)$ folgt

$\underline{p}_Y = (p(y_1), p(y_2), ..., p(y_Q)) = \underline{p}_X \cdot P_{ch} = (p(x_1), p(x_2), ..., p(x_M)) \cdot P_{ch}$.

Betrachten wir als Beispiel den **binären symmetrischen Kanal** (BSC, *Binary Symmetric Channel*). Der 'BSC' Kanal ist ein gedächtnisfreier Kanal mit einer konstanten Übergangswahrscheinlichkeit (konst. Fehlerwahrscheinlichkeit dass eine gesendete 0 als 1 empfangen wird und andersherum). Die Eingangs- und die Ausgangssymbole sind binär, daher $M = Q = 2$.

Die Übergangswahrscheinlichkeit des Kanals ist gegeben durch

$$p(y_i|x_i) = \begin{cases} 1-p & \text{für } y_i = x_i \\ p & \text{für } y_i \neq x_i \end{cases}.$$

Somit ist die Kanalmatrix $P_{bsc} = \begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix}$.

Aufgrund der Kenntnis der Kanalmatrix berechnen wir den mittleren Informationsgehalt des Ausgangszeichens Y unter der Voraussetzung dass am Kanaleingang das Ereignis $X = x_i$ beobachtet wurde:

$$H(Y|x_i) = -\sum_{j=1}^{2} p(y_j|x_i) \cdot \log_2 (p(y_j|x_i)) = -p \cdot \log_2(p) - (1-p) \cdot \log_2(1-p).$$

Aufgrund von Symmetrie der Kanalmatrix folgt

$$H(Y|X) = \sum_{i=1}^{2} p(x_i) \cdot H(Y|x_i) = -p \cdot \log_2(p) - (1-p) \cdot \log_2(1-p).$$

Die wechselseitige Information des BSC Kanals ist somit

$$T(X,Y) = H(Y) - H(Y|X) = H(Y) + p \cdot \log_2(p) + (1-p) \cdot \log_2(1-p).$$

BEISPIEL 2.1: Die Fehlerwahrscheinlichkeit des BSC Kanals beträgt $p = 0.3$.
Aus der Wahrscheinlichkeitsverteilung der Eingangssymbole $(p(x_1), 1 - p(x_1))$ können wir $p(y_1)$ und $p(y_2)$ berechnen:

$$\begin{aligned} p(y_1) &= \sum_{i=1}^{2} p(y_1|x_i) \cdot p(x_i) = (1-p) \cdot p(x_1) + p \cdot (1 - p(x_1)) \\ &= 0.4 \cdot p(x_1) + 0.3 \end{aligned}$$

und $p(y_2) = 1 - p(y_1) = 0.7 - 0.4 \cdot p(x_1)$. Sind beide Sendezeichen gleichwahrscheinlich folgt das auch für die Ausgangszeichen $p(y_1) = p(y_2) = 0.5$.
Die Ausgangsentropie ist in dem Fall $H(Y) = -\sum_{j=1}^{2} p(y_j) \cdot \log_2 (p(y_j)) = 1$, die bedingte Entropie
$H(Y|X) = -p \cdot \log_2(p) - (1-p) \cdot \log_2(1-p) = -0.3 \log_2(0.3) - 0.7 \cdot \log_2(0.7) = 0.88$ bit. Die wechselseitige Information beträgt dann
$T(X, Y) = H(Y) - H(Y|X) = 1 - 0.88 = 0.12$ bit.

Für einen BSC Kanal mit der Fehlerwahrscheinlichkeit von $p = 0.5$ folgt $H(Y) = 1$, $H(Y|X) = 1$ und $T(X, Y) = 0$. D.h. der Kanalausgang ist statistisch unabhängig vom Kanaleingang und durch die Beobachtung von X kann keine Aussage über Y getroffen werden.

BEISPIEL 2.2: Die Fehlerwahrscheinlichkeit des BSC Kanals beträgt $p = 0.3$. Die Sendequelle sendet x_1 mit der Wahrscheinlichkeit $p(x_1) = 0.4$ und damit x_2 mit der Wahrscheinlichkeit $p(x_2) = 0.6$.
Aus der Wahrscheinlichkeitsverteilung der Eingangssymbole $(p(x_1), 1 - p(x_1)) = (0.4, 0.6)$ berechnen wir jetzt
$p(y_1) = \sum_{i=1}^{2} p(y_1|x_i) \cdot p(x_i) = 0.7 \cdot 0.4 + 0.3 \cdot 0.6 = 0.46$ und $p(y_2) = 1 - p(y_1) = 0.54$.
Die Ausgangsentropie ist in dem Fall $H(Y) = -\sum_{j=1}^{2} p(y_j) \cdot \log_2 (p(y_j)) = 0.995$ bit, $H(Y|X) = 0.88$ bit und die wechselseitige Information 0.115 bit.
Für einen BSC Kanal mit der Fehlerwahrscheinlichkeit von $p = 0.5$ ergibt sich allerdings $p(y_1) = p(y_2) = 0.5$, $H(Y) = 1$, $H(Y|X) = 1$ und $T(X, Y) = 0$.

2.2 Kanalkapazität

Die Kanalkapazität eines diskreten gedächtnisfreien Kanals lautet

$$C = \max_{\{p(x)\}} \{T(X, Y)\},$$

d.h. die Kanalkapazität ist die maximale wechselseitige Information zwischen den Eingangssymbolen und Ausgangssymbolen des Kanals. Die Maximierung erfolgt über die Wahrscheinlichkeitsverteilung der Symbole am Kanaleingang.

Betrachten wir wieder den binären symmetrischen Kanal. Zu maximieren ist die wechselseitige Information des BSC Kanals $T(X,Y) = H(Y) + p \cdot \log_2(p) + (1-p) \cdot \log_2(1-p)$ bezüglich $p(x_i), 1 \leq i \leq M$. Die Kanalfehlerwahrscheinlichkeit p ist konstant und die Ausgangsentropie $H(Y)$ ist maximal wenn alle Ausgangszeichen gleich wahrscheinlich sind, d.h. $p(y_1) = p(y_2) = 1/2$ und damit $H_{max}(Y) = 1$. Die Ausgangszeichen sind dann gleich-wahrscheinlich, wenn das auch für die Eingangszeichen gilt, $p(x_1) = p(x_2) = 1/2$. Die resultierende Kanalkapazität des BSC Kanals ist

$$C_{BSC} = \max_{\{p(x_i)\}_{1 \leq i \leq M}} \{T(X,Y)\} = 1 + p \cdot \log_2(p) + (1-p) \cdot \log_2(1-p).$$

Für den Fall dass $p = 0$, der Kanal also fehlerfrei ist, gilt $C = 1$. Man spricht auch vom verlustlosen Kanal, da der maximale mittlere Informationsfluss von 1 bit pro Kanalsymbol erreicht wird.

Dagegen, wenn $C = 0$ ist, kann über den Kanal keine Information übertragen werden. Dieser Fall triff ein wenn $p = 0.5$ ist, d.h. wenn binären Zeichen mit gleicher Wahrscheinlichkeit verfälscht oder unverfälscht sind.

BEISPIEL 2.3: Wie groß ist die Kanalkapazität des BSC Kanals mit der Übergangsfehlerwahrscheinlichkeit $p = 1$?
LÖSUNG: Wie bei $p = 0$, ist auch hier die Kanalkapazität $C = 1$ bit pro Kanalsymbol.

BEISPIEL 2.4: Bei einem binären Auslöschungskanal (BEC, *binary erasure channel*) wird mit der Wahrscheinlichkeit $\delta > 0$ das gesendete Symbol ausgelöscht und damit ε ('erasure') empfangen. D.h. mit der Wahrscheinlichkeit $1 - \delta$ wird das Symbol fehlerfrei übertragen. Wie groß ist die Kanalkapazität des BEC Kanals für $p(x_0 = 0) = p(x_1 = 1) = 0.5$?
LÖSUNG: $H(Y) = (1 - \delta) \log_2 \left(\frac{2}{1-\delta}\right) + \delta \log_2 \left(\frac{1}{\delta}\right)$ und $H(Y|X) = (1-\delta) \log_2 \left(\frac{1}{1-\delta}\right) + \delta \log_2 \left(\frac{1}{\delta}\right)$. Daraus folgt $C = 1 - \delta$.

Das <u>Kanalkapazitätstheorem vom Shannon</u> besagt, dass man über einen Kanal mit der Kanalkapazität C mit einer beliebig kleinen Fehlerwahrscheinlichkeit übertragen kann, solange der mittlere Eingangsinformationsfluss $R = \frac{H(X)}{N}$ kleiner ist als die Kanalkapazität, $R < C$. Hier ist N z.B. die Anzahl der Bits, die für die Übertragung eines Quellenzeichens verwendet werden, wobei $K = H(X)$ die notwendige Anzahl der Bits darstellt.

Das Quellenzeichen X muss also mit $N > K = H(X)$ Bits übertragen werden damit eine fehlerfreie Übertragung trotz $p > 0$ erreicht werden kann. Fehlerfreie Übertragung bedeutet, dass übertragene Zeichen fehlerfrei rekonstruiert werden können, bzw. von die vom Kanal verursachten Fehler korrigiert werden. Das geschickte Hinzufügen von der Redundanz mit dem Ziel der Fehlerkorrektur am Kanalausgang, wird als Kanalcodierung bezeichnet und bedeutet, dass eine längere Zeichenfolge als von der Quelle ausgesandt übertragen wird.

Kanalcodierung fügt also $N - K$ redundante Bits hinzu, wobei diese nicht beliebig gewählt werden, sondern durch Codierung abhängig von Informationsbits erzeugt werden.

BEMERKUNG: Aus dem Hauptsatz der Datenverarbeitung folgt weiterhin auch, dass die Gesamtkapazität einer seriellen Verkettung zweier Nachrichtenkanäle nicht größer sein kann als die Kapazität der Teilkanäle, $C \leq C_1$ und $C \leq C_2$. Für die Gesamtkapazität des Nachrichtenkanals gilt dann $C = \min\{C_1, C_2\}$.

2.3 Codierung zur Fehlerkorrektur

Codierung ist eine Abbildung von einer Menge S auf eine Menge Z, wobei jedem Informationswort $\underline{s} \in S$ ein Codewort $\underline{c} \in Z$ zugeordnet wird.

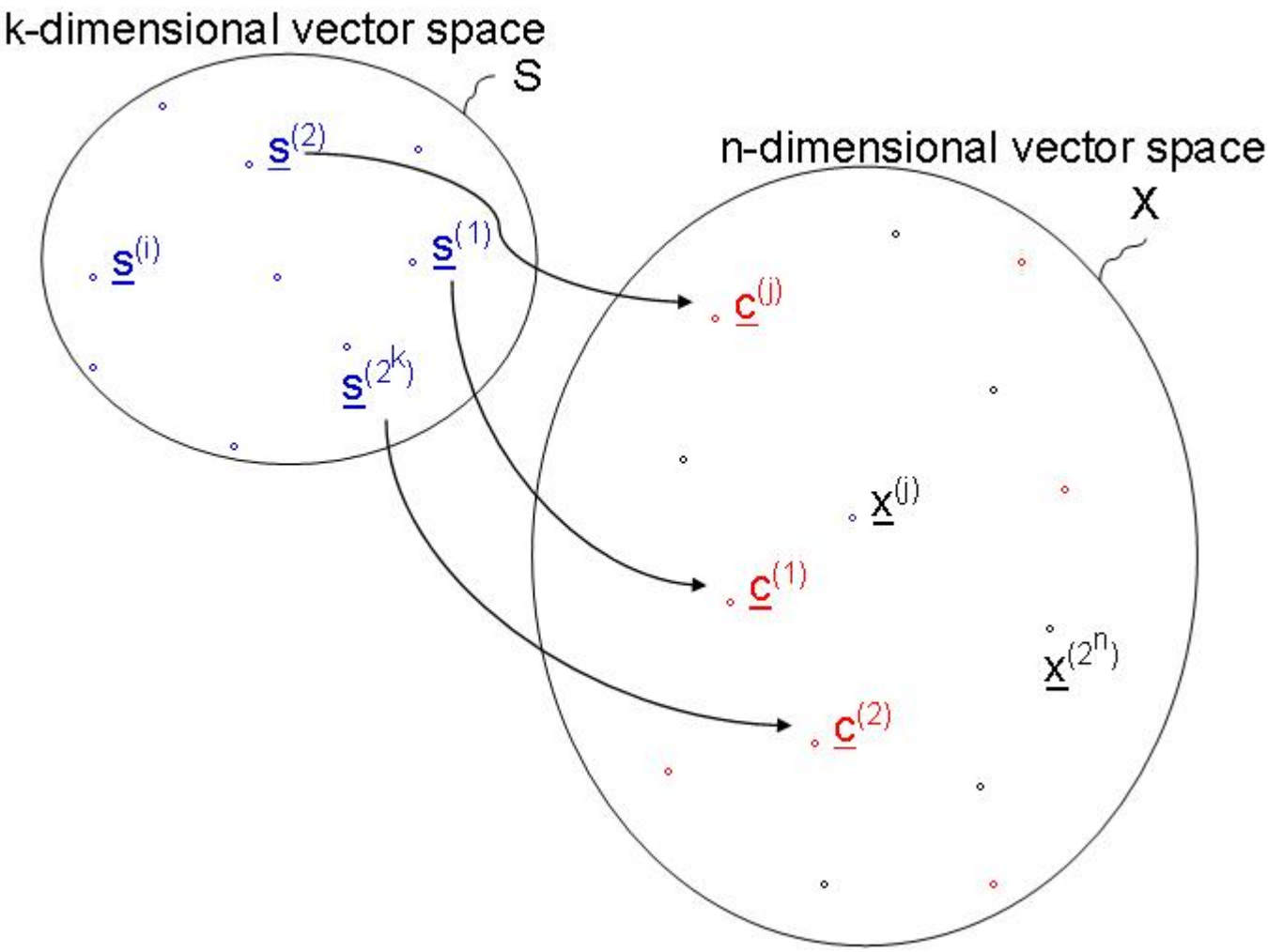
S ist die Menge aller **Informationswörter** der Länge K und $\underline{s} = (s_1, \cdots, s_K)$ ist einer der α^K möglichen Vektoren (Wörter) aus S. Dabei ist α die Größe des Alphabets und $|S| = \alpha^K$ die Kardinalität von S.

Für das binäre Alphabet gilt $\alpha = 2$, $s_i \in \{0, 1\}$ und $|S| = 2^K$.

Ein **Codewort** der Länge N, wobei $N > K : \underline{c} = (c_1, \cdots, c_N)$ ist ein Vektor aus dem Code $C \subset Z$. Hier stellt Z die Menge aller möglichen Vektoren (Wörter) der Länge N mit $|Z| = \alpha^N$.

Ein **Code** C der Länge N ist also eine Untermenge der Menge aller möglichen Vektoren (Wörter) der Länge N, $C \subset Z$, und $|C| = 2^K$.

Ein Encoder führt die Abbildung $\underline{s} \mapsto \underline{c}$ nach einer vorgegebenen Vorschrift durch. Ein Decoder hat die Aufgabe das am wahrscheinlichsten gesendete Codewort zu finden und das zugehörige Informationswort auszugeben.



Dabei betrachten wir immer den gestörten Fall, d.h. codierte Information $\underline{c}$ wird über einen Kanal übertragen, der Fehler verursacht. Bei einer binären Codierung nehmen wir als einfaches Beispiel einen binären Störkanal, wie BSC.

Der Störkanal ist ebenfalls eine Abbildung $\underline{c} \mapsto \underline{y} = \underline{c} + \underline{e}$, wobei $\underline{e}$ das Fehlerwort darstellt.

Ein Decoder decodiert $\underline{y}$ indem er das Codewort sucht, das dem Empfangswort $\underline{y}$ am nächsten ist. Betrachten wir als Beispiel den BSC Kanal, dann unterscheidet sich das am wahrscheinlichsten gesendete Codewort $\tilde{\underline{c}} \in C$ in wenigsten Stellen vom Empfangswort $\underline{y} \in Z$. Decodierung ist also nicht die reine Abbildung $\tilde{\underline{c}} \mapsto \tilde{\underline{s}}$, sondern die Abbildung $\underline{y} \mapsto \tilde{\underline{c}}$, wobei $\tilde{\underline{c}}$ das am wahrscheinlichsten gesendeten Codewort darstellt.

2.3.1 Lineare Binäre Codes

Codierungstheorie verwendet viele elegante mathematische Konstruktionen, wobei wir uns hier auf einfache Beispiele von linearen Codes beschränken um wichtige Grundbegriffe und Eigenschaften zu zeigen.

In den Beispielen werden nur **binäre Codes** mit Elementen 0 und 1 verwendet. Die Überlegungen gelten jedoch grundsätzlich auch für beliebige Zahlenkörper (endliche und unendliche).

Ein binärer Code C hat Codesymbole $c_i \in \{0, 1\}$. Die Kardinalität (Größe, bzw. Anzahl der Elemente) des Codes ist somit $|C| = 2^K$. Die Addition und die Multiplikation werden modulo 2 durchgeführt, d.h. $1 + 1 = 0 \Rightarrow c_i - c_k = c_i + c_k$ bzw. für Codewörter gilt $\underline{c}^{(j)} - \underline{c}^{(\ell)} = \underline{c}^{(j)} + \underline{c}^{(\ell)}$.

Für einen **linearen Code** gilt, dass die Summe zwei Codewörter wieder ein Codewort aus dem Code C ergibt:

$$\underline{c}^{(j)}, \underline{c}^{(\ell)} \in C \;\Rightarrow \underline{c}^{(j)} + \underline{c}^{(\ell)} \in C.$$

Ein Codewort eines Blockcodes habe dabei immer eine Länge $N = K + M$, wobei K die Anzahl der Informations– und M die Anzahl der Prüfstellen angeben soll.

Das Verhältnis $R = K/N$ nennt man **Code Rate.**

BEISPIEL 2.5: Das einfachste Beispiel eines binären Blockcodes ist der *Parity Check Code,* der einer beliebigen Anzahl von Informationsstellen K nur eine einzige Prüfstelle, $M = 1$, hinzufügt.
Die Prüfstelle sei z.B. so gewählt, dass die Gesamtanzahl der Einsen in einem Codewort der Länge $N = K + 1$ geradzahlig wird (*even parity*). Für $K = 3$ sind damit folgende $2^3 = 8$ Codewörter der Länge $N = 4$ möglich:

k_1	k_2	k_3	m_4
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
$\vdots$			

Der Empfänger, der zwar die Codier-Regel kennt, aber nicht das gesendete Codewort, kann damit einen einzelnen Übertragungsfehler (ebenso jede ungerade Anzahl von Übertragungsfehlern) erkennen, da die Gesamtanzahl der Einsen $\left(\sum_{i=1}^{N} c_i\right)$ dann ungeradzahlig wird. Dabei ist es gleichgültig, ob die Fehler in den Informations- oder Prüfstellen aufgetreten sind. Er kann aber einen Fehler nicht korrigieren, da nicht festzustellen ist, wo ein Fehler aufgetreten ist.

BEISPIEL 2.6: Ebenso trivial ist ein Code mit nur einer Informationsstelle $K = 1$, die eine ungerade Anzahl N mal gesendet wird (*Repetition Code*), d.h. jedes Codewort hat $M = N - 1$ Prüfstellen. Dabei gibt es nur $2^K = 2$ Codewörter.
Für z.B. $N = 5$ sind das $\underline{c}^{(1)} = (00000)$ und $\underline{c}^{(2)} = (11111)$. Solange weniger als $e = 2$ Fehler auftreten, kann der Empfänger durch Mehrheitsentscheidung das richtige Codewort finden.
Damit sind bis zu $(N - 1)/2$ Fehler korrigierbar. Bei mehr Fehlern wird eine unerkannte Falschkorrektur durchgeführt. Ohne Fehlerkorrektur sind dagegen bis zu $N - 1$ Fehler erkennbar.

BEISPIEL 2.7: Wir ordnen $K = K_1 \cdot K_2$ Informationsstellen in eine $(K_1 \times K_2)$-Matrix an und fügen in jeder Zeile und Spalte je eine Prüfstelle hinzu (*Product Code*), so dass die Quersumme jeder Zeile und Spalte geradzahlig wird ($= 0 \, mod \, 2$). Insgesamt benötigt man dabei $M = K_1 + K_2 + 1$ Prüfstellen.
Für $K = 4 = 2 \cdot 2$ Informationsstellen benötigt man $M = 2 + 2 + 1 = 5$ Prüfstellen, z.B.

$$\begin{array}{ccc|c} 1 & 1 & & 0 \\ 0 & 1 & & 1 \\ \hline 1 & 0 & & 1 \end{array}$$

Falls nur ein einzelner Fehler auftritt, sind die Quersummen in der zugehörigen Zeile und Spalte nicht gerade. Durch ändern der Binärstelle im Kreuzungspunkt kann ein einzelner Fehler korrigiert werden. Zwei Fehler können erkannt, aber nicht korrigiert werden.

Allgemein gilt, dass Prüf- und Informationsstellen bei der Fehlererkennung oder –korrektur als gleichwertig behandelt werden.

Man bezeichnet als **Hamming Gewicht** $w_H(\underline{c})$ eines Codeworts die Anzahl der Einsen (bzw. Stellen ungleich 0) des Codewortes.

Anzahl der Stellen, in denen sich zwei Codewörter $\underline{c}^{(j)}$ und $\underline{c}^{(\ell)}$ unterscheiden, ist **Hamming Distanz** $d_H(\underline{c}^{(j)}, \underline{c}^{(\ell)})$, wobei für binäre Codes gilt:

$$d_H(\underline{c}^{(j)}, \underline{c}^{(\ell)}) = w_H\left(\underline{c}^{(j)} + \underline{c}^{(\ell)}\right).$$

Von einer Minimal-(Hamming-) Distanz d_{min} eines Code spricht man, wenn sich alle möglichen Codewörter in mindestens d_{min} Stellen voneinander unterscheiden.

Sind $\underline{c}^{(j)}$ und $\underline{c}^{(\ell)}$ Codewörter aus dem linearen Code C, so ist auch $\underline{c}^{(k)} = \underline{c}^{(j)} + \underline{c}^{(\ell)}$ ein Codewort. Dabei unterscheidet sich $\underline{c}^{(k)}$ von $\underline{c}^{(j)}$ genau an den Stellen, wo $c_i^{(k)} \neq 0$ ist. D.h. die Distanz $d_H(\underline{c}^{(j)}, \underline{c}^{(k)})$ ist so groß wie das Gewicht $w_H\left(\underline{c}^{(\ell)}\right)$. Durchläuft bei einem festen Codewort $\underline{c}^{(j)}$ nun $\underline{c}^{(\ell)}$ alle möglichen 2^K Codewörter, so durchläuft auch $\underline{c}^{(k)} = \underline{c}^{(j)} + \underline{c}^{(\ell)}$ alle 2^K Codewörter. D.h. es gibt genau so viele Codewörter $\underline{c}^{(k)}$ im Abstand $d_H(\underline{c}^{(j)}, \underline{c}^{(k)})$ wie es Codewörter $\underline{c}^{(\ell)}$ vom Gewicht $w_H\left(\underline{c}^{(\ell)}\right)$ gibt. Somit gelten folgende Sätze:

Satz. *In einem linearen Code C gibt es zu jedem Codewort $\underline{c}^{(j)}$ genau so viele Codewörter $\underline{c}^{(k)}$ im Abstand $d_H(\underline{c}^{(j)}, \underline{c}^{(k)}) = d$ wie es Codewörter vom Gewicht d gibt, $d = 0, 1, \ldots, N$.*

Satz. *Die minimale (Hamming-) Distanz eines Codes entspricht dem minimalen Hamming Gewicht aller Codewörter des Codes,*

$$d_{min} = \min_{\forall \underline{c}^{(j)}, \underline{c}^{(\ell)} \in C} d_H(\underline{c}^{(j)}, \underline{c}^{(\ell)}) = \min_{\forall \underline{c} \in C, \, \underline{c} \neq \underline{0}} w_H(\underline{c}).$$

Minimale Distanz eines Codes ist der entscheidende Parameter für die Fehlerkorrekturfähigkeit: Man kann allgemein zeigen, dass ein Code eine Mindestdistanz d_{min} bis zu $t \leq \left\lfloor \frac{d_{min}-1}{2} \right\rfloor$ Fehler korrigieren kann. Jedoch kann d_{min} nicht beliebig erhöht werden. Es lässt sich zeigen, dass für jeden linearen

(N, K, d_{min}) Code gilt

$$N - K \geq d_{min} - 1.$$

Diese Ungleichung ist als **Singleton Bound** bekannt. Um $d_{min} \leq N - K + 1$ zu erhöhen, müsste also $N - K$ erhöht werden, wodurch die Code Rate $R = \frac{K}{N}$ kleiner wird. Somit ist die Wahl eines Codes immer ein Kompromiss zwischen der Rate und der Distanz bzw. der Bitfehler-Performance.

Linear Blockcodes werden oft mit Parametern (N, K, d_{min}) angegeben.

Beispiel 2.5 stellt einen **Single Parity Check Code** dar, der zu den K Informationsbits ein Paritätsbit hinzufügt, d.h. $N = K + 1$. Es handelt sich um einen $(N, N-1, 2)$ Code, der keinen Fehler korrigieren, aber einen Fehler erkennen kann.

Bei einem "*even parity check*" gilt $\sum_{i=1}^{N} c_i = 0$ und bei einem *odd parity check*" $\sum_{i=1}^{N} c_i = 1$.

BEISPIEL 2.8: Ein Single Parity Check Code der Länge $N = 3$ hat (da $K = 2$) folgende Codewörter: $(0,0,0)$ $(0,1,1)$ $(1,0,1)$ $(1,1,0)$. Wenn $\underline{y} = (1,0,0)$ empfangen worden ist, sind $(0,0,0)$, $(1,0,1)$ und $(1,1,0)$ gleich-wahrscheinliche Sendewörter. 1 Bitfehler wird also erkannt, kann aber nicht korrigiert werden.
Was passiert wenn zwei Bits verfälscht werden?

Beim **Repetition Code** (Wiederholungscode) wird ein Informationsbit N-mal wiederholt. Es handelt sich also um ein $(N, 1, N)$ Code der $\left\lfloor \frac{N-1}{2} \right\rfloor$ Fehler korrigieren kann.

BEISPIEL 2.9: Codewörter des Repetition Codes der Länge $N = 3$ sind $(0,0,0)$ und $(1,1,1)$. Wenn $\underline{y} = (1,0,0)$ empfangen worden ist, entscheidet sich der Decoder für $\underline{s} = (0,0,0)$. Ein Bitfehler wird also korrigiert, 2 Bit nicht.

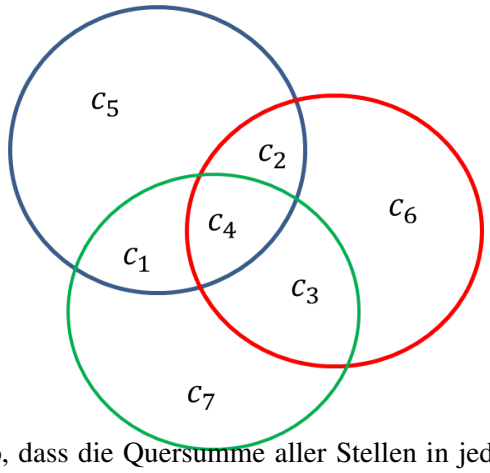
BEISPIEL 2.10: Eine binäre Datenfolge der Länge $K = 7$ soll mit einem binären Code C der Länge $N = 10$ codiert werden.

1. Wie viele Codewörter existieren in dem Code C?
2. Der Übertragungskanal verfälscht einige Bits. Aus wie vielen möglichen Empfangswörter muss der Decoder den wahrscheinlichsten auswählen?

LÖSUNG: 1. 2^7 2. 2^{10}

BEISPIEL 2.11: Was ist die maximale Coderate und die max. Dimension eines linearen Blockcodes der Länge $N = 21$ der bis zu 3 Fehler korrigieren kann?
LÖSUNG: Aus $K \leq N - d_{min} + 1 = 15$ folgt die max. Codedimension $= 2^{15}$ und max. Coderate $= 15/21$.

2.3.2 Codierung und Decodierung am Beispiel vom Hamming Code



In der Abbildung rechts ist der sogenannte **Hamming Code** der Länge $N = 7$ dargestellt, der mit nur $M = 3$ Prüfstellen ($K = N - M = 4$) einen Fehler korrigieren kann. In einem Codewort $\underline{c} = (c_1, c_2, ..., c_7)$ wollen wir z.B. die $K = 4$ Stellen $c_1, ..., c_4$ als Informations- und die $M = 3$ Stellen c_5, c_6, c_7 als die daraus abgeleiteten Prüfstellen betrachten. In der Abbildung bilden die $M = 3$ überschneidenden Kreise insgesamt $N = 7$ Felder. In die inneren $K = 4$ Felder schreibe man je eine binäre Informationsstelle, in die äußeren $M = 3$ Felder je eine Prüfstelle, und zwar so, dass die Quersumme aller Stellen in jedem Kreis (= Prüfgleichung) $0 mod 2$ ergibt.

Bei einem einzigen Fehler stimmt die Quersumme genau in denjenigen Kreisen nicht, welche die falsche Stelle enthalten. Für jede der $N = 7$ möglichen Positionen eines Fehlers erhält man dabei andere Prüfergebnisse: Bei einem Fehler in den Stellen c_5, c_6, c_7 stimmt jeweils genau eine Prüfgleichung nicht, in den Stellen c_1, c_2, c_3 je zwei und in der Stelle c_4 alle drei. Aus den Ergebnissen der drei Prüfgleichungen lässt sich so die Position des Fehlers bestimmen und ein einzelner Fehler somit korrigieren.

Der dargestellte **Hamming Code** ist also ein $(N = 7, K = 4, d_{min} = 3)$ Code mit der Rate $R = 4/7$, der 1 Bitfehler korrigiert.

Zu dem abgebildeten Beispiel zugehörige Prüfgleichungen sind

$$c_i = s_i, i = 1, \cdots, 4$$

$$c_5 = c_1 + c_2 + c_3$$

$$c_6 = c_1 + c_2 + c_4$$

$$c_7 = c_1 + c_3 + c_4$$

Rechts sind alle mögliche 2^4 Informationswörter und die resultierende Codewörter aufgelistet. Dieser Code heißt **systematisch**, d.h. Informationszeichen werden unverändert (d.h. ohne Unterbrechung und in der selben Reihenfolge) auf ein Teil des Codewortes abgebildet werden.

0000	0000 000
0001	0001 011
0010	0010 101
0011	0011 110
0100	0100 110
0101	0101 101
0110	0110 011
0111	0111 000
1000	1000 111
1001	1001 100
1010	1010 010
1011	1011 001
1100	1100 001
1101	1101 010
1110	1110 100
1111 $\rightarrow$	1111 111

Da die Informations- und die Redundanz-Bits im Codewort 'getrennt' sind, lässt sich die Information aus dem Codewort direkt ablesen. Im Allgemeinen kann ein Hamming Code mit Parametern $(2^Q - 1, N - Q, 3)$ und beliebigem $Q \in \mathbb{N}$ generiert werden. Unabhängig von der Codelänge, kann der Code nur 1 Bitfehler korrigieren.

Es existieren viele unterschiedliche Code Konstruktionen, mit denen bessere Codes generiert werden, und passende gute Decodieralgorithmen, die die Fehlerkorrektur vornehmen. Wir bleiben weiterhin beim Beispiel des Hamming Codes, um zu sehen wie mit Hilfe der Linearen Algebra Codierung und Fehlerkorrektur effektiv durchgeführt werden kann. Dafür müssen wir vorerst die Prüf- und die Generatormatrix eines Codes einführen.

Codedarstellung durch eine Prüfmatrix

Schreibt man die Prüfgleichungen des Hamming Codes in der Form

$$1 \cdot c_1 + 1 \cdot c_2 + 1 \cdot c_3 + 0 \cdot c_4 + 1 \cdot c_5 + 0 \cdot c_6 + 0 \cdot c_7 = 0 mod 2$$

$$1 \cdot c_1 + 1 \cdot c_2 + 0 \cdot c_3 + 1 \cdot c_4 + 0 \cdot c_5 + 1 \cdot c_6 + 0 \cdot c_7 = 0 mod 2$$

$$1 \cdot c_1 + 0 \cdot c_2 + 1 \cdot c_3 + 1 \cdot c_4 + 0 \cdot c_5 + 0 \cdot c_6 + 1 \cdot c_7 = 0 mod 2$$

mit der Koeffizienten-Matrix

$$H = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

mit M Zeilen und N Spalten, so besteht der Code aus all jenen Codewörtern (Binärvektoren) $\underline{c} = (c_1, c_2, ..., c_7)$, deren Skalarprodukt mit jeder Zeile von H Null ergibt modulo 2. Jede der M Prüfstellen und damit jede Prüfgleichung entspricht dabei einer Zeile von H. Man nennt deshalb $[H]_{M \times N}$ die **Prüfmatrix** des Codes. In kurzer Form als Matrix-Produkt geschrieben lautet das Gleichungssystem

$$H \cdot \underline{c}^T = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

Da jede Zeile in H im Skalarprodukt Null ergibt, gilt dies auch für jede Linearkombination von Zeilen von H. Jede $M \times N$ Matrix H' mit M linear unabhängigen Zeilen, die durch Linearkombinationen der Zeilen von H entstanden sind, definiert deshalb den gleichen Code wie H.

Für jeden linearen (N, K, d_{min}) Code C existiert eine Matrix H für die gilt: $H \cdot \underline{c}^T = \underline{0}, \forall \underline{c} \in C$. Ein $\underline{c}$ ist genau dann ein Codewort, wenn $H \cdot \underline{c}^T = \underline{0}$ ist.

BEISPIEL 2.12: Wie viele Codewörter existieren in einem linearen binären Code mit einer 7×13 Prüfmatrix mit linear unabhängigen Zeilen?
LÖSUNG: 2^6

Codedarstellung durch eine Generatormatrix

Jedes Codewort $\underline{c}$ eines linearen (N, K, d_{min}) Codes entspricht einer linearen Kombination von K linear unabhängigen Codewörter $\underline{g}_1$, ..., $\underline{g}_K \in C$. Eine Matrix, deren Zeilen K linear unabhängige Codewörter eines Codes darstellen, generiert durch die lineare Abbildung

$$C: \quad \underline{s} \mapsto \underline{c} = \underline{s} \cdot G$$

aus 2^K möglichen Informationsworten der Länge K, alle 2^K mögliche Codewörter des Codes. Die Matrix $G = \begin{pmatrix} \underline{g}_1 \\ \vdots \\ \underline{g}_K \end{pmatrix}$ wird **Generatormatrix** $[G]_{K\times N}$ des Codes genant.

BEISPIEL 2.13: Die Generatormatrix $G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$ generiert den $(7,4,3)$ Hamming Code.
Für den Encoder mit $\underline{c} = \underline{s} \cdot G$ folgt z.B. $(0100) \cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} = (0100110)$.

Aus $\underline{c} = \underline{s} \cdot G$ und $H \cdot \underline{c}^T = \underline{0}$ folgt der Zusammenhang zwischen der H und der G Matrix, $H \cdot G^T = 0$:

$$H \cdot \underline{c}^T = H \cdot (\underline{s} \cdot G)^T = H \cdot G^T \underline{s}^T = 0 \; \forall \underline{s} \quad \Rightarrow \; H \cdot G^T = 0.$$

Die Prüfmatrix lässt sich aus der G Matrix einfach berechnen, wenn G Matrix in der systematischen Form, $G = [I|P]$ gegeben ist. Hier ist P eine $K \times M$ Matrix und I die $K \times K$ Einheitsmatrix. Die zugehörige H Matrix hat die Form $\left[-P^T|I\right]$, wobei P^T die Dimension $M \times K$ hat und I hier eine $M \times M$ Einheitsmatrix ist.

BEISPIEL 2.14: Für die Generatormatrix $G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$ des Hamming Codes $(7,4,3)$ ist die zugehörige Prüfmatrix $H = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$.
Bsp. $\underline{c} = (1001100) = (1001) \cdot G$ und

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Decodierung mit HIlfe des Syndroms

Das **Code Syndrom** ist das Ergebnis der Multiplikation der Prüfmatrix mit dem Empfangswort,

$$H \cdot \underline{y}^T = \underline{z}.$$

Ein Syndrom $\underline{z} \neq \underline{0}$ weist auf ein Fehler bei der Übertragung hin.

Ein Bitfehler im Hamming Code lässt sich mit Hilfe der Syndrom-Decodierung korrigieren. Bei einer fehlerbehafteten Übertragung, lässt sich das Empfangswort $\underline{y}$ als Summe aus dem Codewort und einem Fehlerwort schreiben, $\underline{y} = \underline{c} + \underline{e}$. Die Multiplikation des Empfangswortes mit der Prüfmatrix ergibt

$$H \cdot \underline{y}^T = H \cdot (\underline{c} + \underline{e})^T = H \cdot \underline{c}^T + H \cdot \underline{e}^T = H \cdot \underline{e}^T$$

(weil per Definition $H \cdot \underline{c}^T = \underline{0}$ gilt). Das Syndrom hängt also nur vom Fehlerwort ab. Wenn nur ein Bit an der Stelle k in dem Codewort verfälscht wird, d.h. $\underline{e} = (e_1, e_2, ..., e_N)$ mit $e_k = 1, e_i = 0 \; \forall i \neq k$, ergibt $\underline{z} = H \cdot \underline{e}^T$ genau die $k-$te Spalte der H Matrix.

BEISPIEL 2.15: Das gesendete Codewort $\underline{c}$ soll mit Hilfe von Syndrom-Decodierung bestimmt werden. Gegeben sind die Prüfmatrix des Codes H und das Empfangswort $\underline{y}$:

$$H = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \text{ und } \underline{y} = (1001101)$$

Aus $H \cdot \underline{y}^T = \underline{z}$ folgt das Syndrom $\underline{z}^T = (001)$, d.h. $\underline{y}$ ist kein gültiges Codewort des Codes.
Das Syndrom entspricht der $7-$ten Spalte der H Matrix. Unter der Annahme dass nur 1 Bitfehler aufgetreten ist, entscheiden wir aus dem Syndrom dass $\underline{e} = (0000001)$ ist und $\underline{c} = (1001100)$ übertragen worden ist.
Für $w_H(\underline{e}) > 1$ ist eine solche Entscheidung falsch!

2.4 Fehlererkennung am Beispiel vom CRC Code

Cyclic Redundancy Check (CRC) Code ist ein zyklischer binärer Blockcode, der nicht zur Fehlerkorrektur, sondern nur zur Fehlerdetektion eingesetzt wird. Der CRC Code besteht aus Paritätsbits, die mit Hilfe eines **Generatorpolynoms** $g(x)$, auch CRC Polynom genannt, für eine gegebene Informationssequenz generiert werden. Die zu übertragene Information wird (block- oder *frame*-weise) mit diesen Paritätsbits erweitert und gegebenenfalls anschließend mit einem fehlerkorrigierenden Code codiert. Im Empfänger wird mittels des Generatorpolynoms geprüft, ob die bereits decodierte Information noch fehlerbehaftet ist. Die Fehlerdetektion entspricht der Berechnung der Restpolynoms bei der Division der Codesequenz mit $g(x)$. Wird ein Fehler in einer Datensequenz detektiert, dann wird diese noch einmal übertragen (see z.B. ARQ). Der Vorteil der Fehlerdetektion mittels CRC Codes ist die sehr einfache Realisierung, hard- wie softwaremäßig.

Beispiele für Generatorpolynome der CRC Codes:

- ▷ ISDN Header Error Control: CRC-8(ITU-T) $g(x) = x^8 + x^2 + x + 1$

- ▷ Ethernet: CRC-32 $g(x) = x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$
- ▷ USB: CRC-5 $g(x) = x^5 + x^2 + 1$

Für CRC Codes benötigen wir nur die polynomiale Darstellung eines Vektors in $GF_{2^N}[x]$. Das bedeutet, das ein binärer Vektor $(c_0, c_1, \ldots, c_{N-2}, c_{N-1}),\ c_i \in \{0,1\}$ als ein Polynom mit Koeffizienten c_i dargestellt wird:

$(c_0, c_1, \ldots, c_{N-2}, c_{N-1}),\ c_i \in \{0,1\} \quad \Leftrightarrow \quad c(x) = \sum_{i=0}^{N-1} c_i x^i \ \in GF_{2^N}[x]$

Z.B:
$$\begin{aligned}
&0\ldots000 \rightarrow 0\\
&0\ldots001 \rightarrow 1\\
&0\ldots010 \rightarrow x\\
&0\ldots100 \rightarrow x^2\\
&\quad\vdots\\
&1\ldots111 \rightarrow x^{N-1} + \cdots + x^2 + x + 1 = \tfrac{x^N-1}{x-1}
\end{aligned}$$

Der CRC Code ist ein systematischer Code, der aus der Informationssequenz und zusätzlichen Prüfbits besteht. Die Anzahl der Prüfbits entspricht dem Grad des Generatorpolynoms.

Sei $g(x) = g_0+g_1x+\cdots+g_{n-k}x^{N-K}$ das Generatorpolynom und $s(x) = s_0+s_1x+\cdots+s_{k-1}x^{K-1}$ das Informationspolynom bzw. $\underline{s} = (s_0, \ldots, s_{K-1})$ der Informationsvektor, wobei $s_i, g_i \in \{0,1\}$. Ein systematischer Codes kann dadurch generiert werden, dass jeder Informationsvektor um $N-K$ Prüfstellen erweitert wird. Eine Erweiterung mit $N-K$ Nullen, bzw. Verschiebung von K Informationsbits an die Stellen $N-K$ bis $N-1$, lässt sich als Multiplikation $s(x) \cdot x^{N-K}$ realisieren.

Um den CRC Codevektor zu generieren, werden an den ersten $N-K$ Stellen Prüfbits p_i hinzugefügt. Der resultierende Codevektor ist dann $\underline{c} = (p_0, \ldots, p_{N-K-1}, s_0, \ldots, s_{K-1})$, bzw. das Codepolynom $c(x) = p(x) + x^{N-K}s(x)$.

Die Bestimmung der Prüfbits bzw. des Prüfpolynoms $p(x)$ erfolgt durch die Polynomdivision

$$p(x) = \left(x^{N-K}s(x)\right) \text{mod} g(x),$$

denn für das Codepolynom $c(x)$ gilt dann

$$c(x) \text{mod} g(x) = 0.$$

BEMERKUNG: Da $1+1=0$, gilt für ein Polynom mit binären Koeffizienten $p(x)+p(x)=0$ und damit auch

$$\begin{aligned}
c(x)\text{mod}g(x) &= \left(p(x) + x^{N-K}s(x)\right) \text{mod} g(x)\\
&= \left(\left(x^{N-K}s(x)\right) \text{mod} g(x) + x^{N-K}s(x)\right) \text{mod} g(x)\\
&= 0.
\end{aligned}$$

D.h., eine zu überprüfende, mit dem CRC Code geschützte Datensequenz $\underline{y}$, ist fehlerfrei, wenn die Polynomdivision $y(x)\text{mod}g(x)$ den Rest $= 0$ ergibt.

Fehlererkennung Empfangene Daten $\underline{y}$ werden mittels der Polynomdivision $y(x)/g(x)$ auf Fehler geprüft. Ist der Rest der Division (CRC Wert) ungleich Null, dann ist die die Datensequenz fehlerhaft.

Es gilt $y(x)$ mod $g(x) = (c(x) + e(x))$ mod $g(x) =$ c(x) mod $g(x)$+$e(x)$ mod $g(x) = e(x)$ mod $g(x)$.

Ist aber das CRC Polynom $g(x)$ ein Teiler des Fehlerpolynoms $e(x)$, d.h. $e(x)$ mod $g(x) = 0$, dann werden fehlerhafte Daten als fehlerfrei erkannt. D.h., wenn der Rest der Polynomdivision gleich Null ist, dann ist entweder

- ▷ kein Fehler aufgetreten, oder
- ▷ der Fehlervektor hat das CRC Polynom als Faktor (nicht erkennbare Fehler).

Durch eine geschickte Auswahl des CRC Polynoms lässt sich die Wahrscheinlichkeit, dass der Fehler nicht detektiert wird, zusätzlich reduzieren. Folgende Aussagen gelten (hier ohne Beweis):

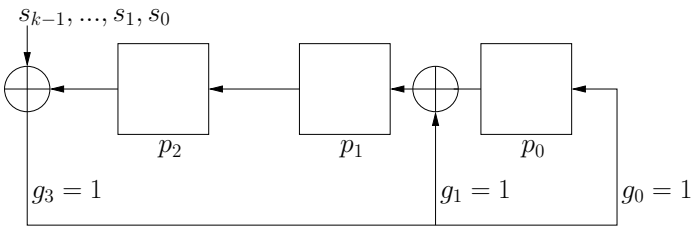
- ▷ Wenn das CRC Polynom zwei oder mehr Terme enthält, wird jeder 1-Bit-Fehler erkannt.
- ▷ Zwei Fehler im Abstand M werden erkannt wenn das CRC Polynom kein Teiler von $x^M + 1$ ist.
- ▷ Alle Fehlervektoren mit einer ungeraden Anzahl von verfälschten Bits werden erkannt, wenn das CRC Polynom $(x + 1)$ als Faktor hat.
- ▷ Alle Bündelfehler der Länge $\ell \leq N - K =$ deg $(g(x))$ werden erkannt.

BEISPIEL 2.16: Die Informationssequenz $\underline{s} = (s_0 ... s_{12}) = (1001011010101)$ soll mit dem CRC Code mit dem Generatorpolynom $g(x) = x^3 + x + 1$ geschützt werden.

- ▷ Polynomdivision: $p(x) = x^{n-k} s(x)$ mod $g(x) =$
  $x^3(x^{12} + x^{10} + x^8 + x^6 + x^5 + x^3 + 1)$ mod $(x^3 + x + 1) = 1$,
  d.h. $\underline{p} = (100)$ und $\underline{c} = (1001001011010101)$ der Codevektor.

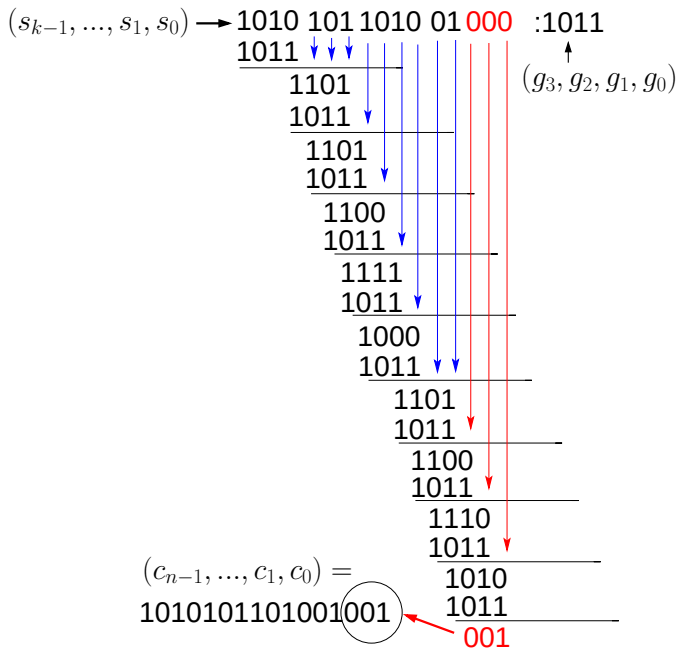
BEISPIEL 2.17: Die Informationssequenz $\underline{s} = (s_0 ... s_{12}) = (1001011010101)$ soll mit dem CRC Code mit dem Generatorpolynom $g(x) = x^3 + x + 1$ geschützt werden.

- ▷ Lösung mit dem rückgekoppelten Schieberegister
  (LFSR≡'*Linear Feedback Shift Register*'):



BEISPIEL 2.18: Die Informationssequenz $\underline{s} = (s_0 ... s_{12}) = (1001011010101)$ soll mit dem CRC Code mit dem Generatorpolynom $g(x) = x^3 + x + 1$ geschützt werden.

▷ Lösung mit der langen Division:



3 Quellencodierung

Quellencodierung ordnet jedem Zeichen x_i des Quellenalphabets ein Codewort $\underline{a}^{(i)} = (a_1^{(i)}, a_2^{(i)}, ..., a_{\ell_i}^{(i)})$ der Länge ℓ_i zu. Bei der Binärcodierung gilt $a_j \in \{0, 1\}$. Alle Codewörter, die dem Quellenalphabets zugeordnet sind, bilden den Code $\mathcal{A}$. Man unterscheidet zwischen den gleichmäßigen Codes, in den alle Codewörter gleich lang sind, und ungleichmäßigen Codes, mit variablen Codewortlängen.
Die Zuordnung vom Quellenzeichen zum Codewort muss eindeutig sein. Damit der Code eindeutig decodiert werden kann, muss die codierte binäre Folge eindeutig in Codewörter zerlegt werden. Die ursprüngliche Zeichenfolge wird dann durch die Abbildung von einem Codewort zum Alphabet-Zeichen gewonnen.

Kurze Codewörter sind vom Vorteil, da sie weniger Speicherplatz und kürzere Übertragungszeit benötigen. Die Länge der binäre Codewörter hängt aber von dem mittleren Informationsgehalt der Quelle und kann nicht beliebig klein gewählt werden.

Die Differenz zwischen der Codewortlänge und der Entropie der Quelle wird als Coderedundanz bezeichnet. Dabei ist der Ziel der Quellencodierung einen redundanzfreien oder möglichst redundanzarmen Quellencode zu realisieren.

Betrachten wir im Folgenden gedächtnisfreie Informationsquellen und binäre Codes mit variablen Codewortlängen.

3.1 Codierung mit Kenntnis der Quellenstatistik

3.1.1 Präfix(freie) Codierung

Jedes Zeichen x_i der Nachrichtenquelle, bzw. eine Zufallsvariable aus dem Alphabet $\alpha_M = \{x_1, x_2, ..., x_M\}$, wird auf ein Codewort $\underline{a}^{(i)} = (a_1^{(i)}, a_2^{(i)}, ..., a_{\ell_i}^{(i)})$ der Länge ℓ_i abgebildet. Die mittlere Codewortlänge,

$$\ell_C = \sum_{i=1}^{M} \ell_i \cdot p(x_i)$$

ist ein Maß für die Effizienz der Codierung. Umso kleiner die Redundanz,

$$R_C(X) = \ell_C - H(X),$$

desto effizienter die Quellencodierung.
Damit ein Code mit ungleichen Codewortlängen eindeutig decodiert werden kann, dürfen keine zwei Codewörter identisch sein und kein Codewort darf den Präfix (Anfang) eines anderen Codewortes darstellen.

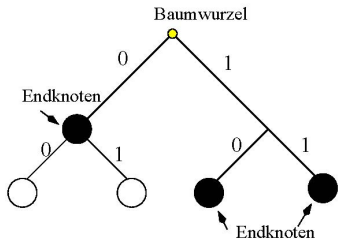
BEISPIEL 3.1: Drei Zeichen werden folgendermaßen binär codiert:

- ▷ Code 1: $x_1 \to 0$, $x_2 \to 10$, $x_3 \to 11$
- ▷ Code 2: $x_1 \to 1$, $x_2 \to 10$, $x_3 \to 11$

Der Code 1 ist ein Präfix-freier Code während der Code 2 keiner ist, da x_1 der Präfix von x_2 und x_3 ist.

Ein Code kann in Form eines Codebaumes dargestellt werden. Dabei wird jedes Codewort durch einen Pfad, der vom Baumwurzel zum Endknoten verläuft, bestimmt.
Die Stufen des Codebaumes bestimmen die verschiedenen Codewortlängen. Die letzte Stufe entspricht der maximalen Codewortlänge, $\ell_{max} = \max\limits_{i=1..M}(\ell_i)$. Auf der Stufe ℓ_i gibt es 2^{ℓ_i} Knoten. Wenn sich ein Endknoten auf einer Stufe $\ell_i < \ell_{max}$ befindet, dann bleiben auf der letzten Stufe $2^{\ell_{max}-\ell_i}$ Knoten ungenutzt.

Der Codebaum des Codes 1 aus dem Beispiel oben:



Beim ungleichmäßigen Code wird die Anzahl zulässiger Codewörter durch jeden Endknoten reduziert, der auf einer Stufe $\ell < \ell_{max}$ liegt. Damit der Code existiert, muss die Anzahl der ungenutzten Knoten, $\sum\limits_{i=1}^{M} 2^{\ell_{max}-\ell_i}$, kleiner sein als die Anzahl der Endknoten insgesamt, $2^{\ell^{max}}$. Daraus folgt die sogenannte KRAFT-UNGLEICHUNG:

$$\sum_{i=1}^{M} 2^{-\ell_i} \leq 1.$$

Während die Präfix-Eigenschaft eine hinreichende Bedingung ist, ist die Kraft-Ungleichung nur eine notwendige Bedingung. Sie besagt, dass es einen Code mit den gewählten Codewortlängen gibt, aber nicht wie der Code ist.

BEISPIEL 3.2: Betrachten wir 4 unterschiedliche Codes mit 6 Codewörtern:

- ▷ $C_1 = \{00, 01, 10, 110, 111, 1101\}$ erfüllt die Präfix-Bedingung nicht.
  Die Kraft-Ungleichung ist ebenfalls nicht erfüllt: $3 \cdot 2^{-2} + 2 \cdot 2^{-3} + 1 \cdot 2^{-4} > 1$.
- ▷ $C_2 = \{00, 01, 10, 110, 1110, 1101\}$ erfüllt die Präfix-Bedingung nicht.
  Die Kraft-Ungleichung ist aber erfüllt: $3 \cdot 2^{-2} + 1 \cdot 2^{-3} + 2 \cdot 2^{-4} = 1$.
- ▷ $C_3 = \{00, 01, 10, 110, 1110, 1111\}$ erfüllt die Präfix-Bedingung.
  Die Kraft-Ungleichung ist wie im Fall von C_2 erfüllt.
- ▷ $C_4 = \{0, 100, 101, 110, 1110, 1111\}$ erfüllt die Präfix-Bedingung.
  Die Kraft-Ungleichung ist ebenfalls erfüllt: $1 \cdot 2^{-1} + 3 \cdot 2^{-3} + 2 \cdot 2^{-4} = 1$.

Eindeutig decodierbar sind somit nur C_3 und C_4.

Wenn die Kraft-Ungleichung für gewählte Codelängen erfüllt ist, dann lässt sich ein Code durch schrittweise Verlängerung des Baumes konstruieren.

BEISPIEL 3.3: Es soll ein Code mit den Längen $\ell_1 = \ell_2 = \ell_3 = 2$, $\ell_4 = 3$ und $\ell_5 = 4$ konstruiert werden. Wegen $\sum_{i=1}^{5} 2^{-\ell_i} = \frac{3}{4} + \frac{1}{8} + \frac{1}{16} = \frac{15}{16} < 1$ wissen wir dass ein solcher präfixfreier Code existiert. Wir lassen den Baum schrittweise wachsen und bekommen Codewörter $00, 01, 10, 110, 1110$.

BEISPIEL 3.4: Ein Code mit den Längen $\ell_1 = 1,\ \ell_2 = \ell_3 = 2$, $\ell_4 = 3$ und $\ell_5 = 4$ kann nicht konstruiert werden, da
$\sum_{i=1}^{5} 2^{-\ell_i} = \frac{1}{2} + \frac{2}{4} + \frac{1}{8} + \frac{1}{16} = \frac{19}{16} > 1.$

Ein gleichmäßiger Code wird mit einem vollständigen Baum beschrieben, d.h. alle Endknoten bei $\ell = \ell_{max}$ sind besetzt, wobei ℓ die Länge aller Codewörter darstellt. Wenn M Zeichen des Quellenalphabets mit einem gleichmäßigen Code codiert werden, ist die Codelänge $\ell \geq \log_2 M$.
Es kann gezeigt werden, dass für einen ungleichmäßigen Code für die mittlere Codelänge die untere Schranke

$$\ell_C \geq H(X)$$

gilt. Das bedeutet für die Coderedundanz $R_C(X) \geq 0$. Ein Code wird als *optimal* bezeichnet, wenn er kleinst mögliche Redundanz aufweist.

BEISPIEL 3.5: Codes $C_3 = \{00, 01, 10, 110, 1110, 1111\}$ und $C_4 = \{0, 100, 101, 110, 1110, 1111\}$ sind eindeutig decodierbar. Angenommen, die Quellenstatistik der Quelle sei bekannt und die Wahrscheinlichkeitsverteilung ist $\underline{p} = (0.6, 0.1, 0.1, 0.1, 0.05, 0.05)$.
Welcher der beiden Codes beinhaltet weniger Redundanz?
Lösung: $H(X) = 1.87$ bit, $R_{C3} = 0.43$ bit $> R_{C4} = 0.03$ bit.

Für einen redundanzfreien Code müsste die Codewortlänge für jedes Zeichen entsprechend seiner Auftrittswahrscheinlichkeit gewählt werden, $\ell_i = \log_2 \frac{1}{p_i} \Rightarrow 2^{-\ell_i} = p_i$. Wenn diese Gleichung für keine natürliche Zahl ℓ_i erfüllt werden kann, dann ergibt sich folgende Schranke für einen annähernd redundanzfreien Code:

$$2^{-\ell_i} \leq p_i < 2^{-\ell_i+1}.$$

Aus $\log_2 p_i < -\ell_i + 1$ folgt $\ell_i < -\log_2 p_i + 1$. Das Gewichten mit p_i und Aufsummieren über alle i ergibt $\ell_C < H(X) + 1$. Ausgehend von

$$H(X) \leq \ell_C < H(X) + 1$$

zeigte Shannon, dass jede diskrete Quelle prinzipiell völlig redundanzfrei codiert werden kann, auch wenn keine 'passende' Wahrscheinlichkeiten vorliegen. Wenn Quellenzeichen in Blöcken von k Zeichen codiert werden, dann gilt analog zu der obigen Gleichung

$$kH(x) \leq k\ell_C < kH(X) + 1 \Rightarrow H(x) \leq \ell_C < H(X) + \frac{1}{k}.$$

Durch das Zusammenfassen von mehreren Zeichen, kann die mittlere Codelänge beliebig nah an das mittlere Informationsgehalt der Quelle kommen. Allerdings verzögert eine solche Codierung die Decodierung der einzelnen Quellenzeichen, da die Entscheidung erst nach k Zeichen getroffen werden kann.

3.1.2 Codierverfahren (Entropiecodierung)

Unter Entropiecodierungsalgorithmen werden Verfahren zusammengefasst, die auf Kenntnis der Quellenstatistik beruhen. Bei den Präfix-freien Codes werden die Codewortlängen anhand der Auftrittswahrscheinlichkeiten des Zeichens bestimmt. Umso höher die Auftrittswahrscheinlichkeit eines Quellzeichens, desto kleiner die entsprechende Codelänge. Damit sind häufig auftretende Codewörter kurz und seltene lang. Die mittlere Codelänge wird somit minimiert. Im folgenden sind unterschiedliche Algorithmen vorgestellt, wobei die Huffman Codierung die kleinste Redundanz aufweist und damit optimal ist.

3.1.2.1 Shannon-Codierung

Algorithmus:

1. Sortiere die M Zeichen der Nachrichtenquelle X nach ihren Auftrittswahrscheinlichkeiten, $p_1 \geq p_2 \geq \ldots \geq p_M$.
2. Bestimme die Codewortlängen nach $\ell_i = \lceil -\log_2 p(x_i) \rceil$.
3. Berechne die kumulativen Wahrscheinlichkeiten $P_i = p_1 + p_2 + ... + p_{i-1}$ für $1 < i \leq M$.
4. Ordne den kumulativen Wahrscheinlichkeiten die größte mögliche binäre Zahl $\underline{b}_i = (b_{i,1}, ..., b_{i,\ell})$,$b_{i,j} \in \{0, 1\}$ zu, so dass

$$\sum_{j=1}^{\ell_i} b_{i,j} \cdot 2^{-j} \leq P_i.$$

BEISPIEL 3.6: Eine Nachrichtenquelle sendet eine Zeichenfolge, mit Zeichen aus dem Alphabet $\{a, b, c, d, e, f, g\}$ und der Wahrscheinlichkeitsverteilung $\{0.05, 0.03, 0.17, 0.23, 0.01, 0.32, 0.19\}$, aus. Generiere den Shannon Code, berechne die resultierende mittlere Codelänge und die resultierende Redundanz.

BEISPIEL 3.6: (Fortsetzung)

LÖSUNG:

p_i	x_i	ℓ_i	P_i	$\underline{b}$
0.32	f	2	0	00
0.23	d	3	0.32	010
0.19	g	3	0.55	100
0.17	c	3	0.74	101
0.05	a	5	0.91	11101
0.03	b	6	0.96	111101
0.01	e	7	0.99	1111110

$a \to 11101$, $b \to 111101$, $c \to 101$, $d \to 010$, $e \to 1111110$, $f \to 00$, $g \to 100$.
Die mittlere Codelänge ist $\ell_C(X) = 2.91$bit. Mit der Entropie $H(X) = 2.3378$ bit folgt die Redundanz $R_C(X) = 0.57218$bit.

In dem obigen Beispiel könnten die Codewörter für a, b, d, e um die letzte Bit-stelle gekürzt werden und der neue Code $\mathcal{A}'$ wäre ebenfalls eindeutig decodierbar. Die mittlere Codelänge vom modifizierten Code ist kleiner, $\ell_{C'}(X) = 2.59$ bit und somit auch die Redundanz der Quellencodierung. Der Shannon Algorithmus liefert nicht immer den optimalen Code.

3.1.2.2 Fano-Codierung

Algorithmus:

1. Sortiere die M Zeichen x_i der Nachrichtenquelle X werden nach ihrer Auftrittswahrscheinlichkeit,

   $p(x_1) \geq p(x_2) \geq \ldots \geq p(x_M)$.

2. Teile die Wahrscheinlichkeiten in zwei geordnete Teilmengen, so dass die Summe der Auftrittswahrscheinlichkeiten in beiden Mengen möglichst gleich sind, $\sum_{j=1}^{k} p(x_j) \approx \sum_{j=k+1}^{M} p(x_j)$.

3. Ordne den Codewörtern der ersten Menge das Bit 0 und den Codewörtern der zweiten Menge das Bit 1 zu.

4. Wiederhole die Schritte 2 und 3 bis die Teilmengen die Kardinalität 1 haben.

BEISPIEL 3.7: Eine Nachrichtenquelle sendet Zeichen $\{a, b, c, d, e, f, g\}$ mit den Auftrittswahrscheinlichkeiten $\{0.05, 0.03, 0.17, 0.23, 0.01, 0.32, 0.19\}$ aus. Generiere den Fano Code, berechne die resultierende mittlere Codelänge und die resultierende Redundanz.

LÖSUNG: Wenn die erste Teilung: $\{0.32, 0.23\}$ und $\{0.19, 0.17, 0.05, 0.03, 0.01\}$ ist, die zweite Menge in $\{0.19\}$ und $\{0.17, 0.05, 0.03, 0.01\}$ geteilt wird, danach $\{0.17\}$ und $\{0.05, 0.03, 0.01\}$, $\{0.05\}$ und $\{0.03, 0.01\}$, folgt der Code
$a \to 1110$, $b \to 11110$, $c \to 110$, $d \to 01$, $e \to 11111$, $f \to 00$, $g \to 10$.
Die mittlere Codelänge ist $\ell_C(X) = 2.390$ bit. Mit der Entropie $H(X) = 2.3378$ bit folgt die Redundanz $R_C(X) = 0.0522$ bit.
Die Aufteilung der Wahrscheinlichkeiten in zwei Mengen ist nicht immer eindeutig. Es können sich also unterschiedliche aber gleichwertige Codes ergeben. Die Coderedundanz ist kleiner als beim Shannon Code, jedoch noch immer nicht optimal.

3.1.2.3 Huffman-Codierung

Die Konstruktion eines optimalen (Huffman) Codes basiert darauf, dass

- ▷ der binäre Baum eines optimalen Codes keine ungenützten Endknoten hat und
- ▷ zwei am wenigsten wahrscheinliche Codewörter unterscheiden sich nur in der letzten Stelle, d.h. habe einen gemeinsamen Präfix.

Während Shannon und Fano Codierung nach dem „Top-to-Bottom" Algorithmus arbeiten, d.h. das Codewort wird ausgehend vom ersten Bit aufgebaut, fängt der Aufbau der Codewörter im Huffman Algorithmus mit dem letzten Bit („Bottom-to-Top" Methode).

Algorithmus:

1. Sortiere die M Zeichen x_i der Nachrichtenquelle X werden nach ihrer Auftrittswahrscheinlichkeit,

   $p(x_1) \geq p(x_2) \geq \ldots \geq p(x_M)$.
2. Die zwei kleinsten Wahrscheinlichkeiten stellen zwei Endknoten des Codebaumes dar. Fasse sie zu einem neuen Wert zusammen und sortiere die Wahrscheinlichkeiten erneut absteigend.
3. Wiederhole den letzten Schritt bis die resultierende Menge aus zwei Wahrscheinlichkeitswerten besteht, dessen Summe 1 ergibt.

Mit diesem Iterationsverfahren wird ein binärer Codebaum aufgestellt. Den einzelnen Zweigen des Codebaums werden die Bits 0 und 1 zugeordnet. Die resultierende Codewörter entsprechen den Bits gelesen ausgehend von dem ersten gemeinsamen Knoten bis in die Endknoten.

BEISPIEL 3.8: Eine Nachrichtenquelle sendet Zeichen $\{a, b, c, d, e, f, g\}$ mit den Auftrittswahrscheinlichkeiten $\{0.05, 0.03, 0.17, 0.23, 0.01, 0.32, 0.19\}$ aus. Generiere den Huffman Code, berechne die resultierende mittlere Codelänge und die resultierende Redundanz.

LÖSUNG: $a \to 0110$, $b \to 01110$, $c \to 010$, $d \to 10$, $e \to 01111$, $f \to 00$, $g \to 11$.
Die mittlere Codelänge ist $\ell_C(X) = 2.39$ bit. Mit der Entropie $H(X) = 2.3378$ bit folgt die Redundanz $R_C(X) = 0.0522$ bit.
In diesem Beispiel liefert die Huffman Codierung dieselbe Codewortlängenverteilung und Redundanz als Fano Codierung. In der Regel besitzt Huffman Codierung die minimale Redundanz.

BEISPIEL 3.9: Für eine Nachrichtenquelle mit der gegebenen diskreten Wahrscheinlichkeitsverteilung ihrer Zeichen, $\underline{p}(X) = (0.18, 0.10, 0.40, 0.08, 0.05, 0.05, 0.14)$ sind der Codes nach Shannon, Fano und Huffman Verfahren zu entwerfen und bezüglich der Coderedundanz zu vergleichen.

Allerdings erzielt die Huffman-Codierung die maximale Kompression, wenn alle Wahrscheinlichkeiten ganzzahlige Potenzen von ½ sind. Der ungünstigste Fall für die Huffman-Codierung tritt ein, wenn ein Symbol eine Wahrscheinlichkeit von ca. 1 hat. Zum Beispiel hat ein Zeichen mit einer Wahrscheinlichkeit von ca. 90% eine optimale Codelänge von 0,15 Bit. Mit der Huffman-Codierung würde dieses Symbol jedoch durch einen Code mit der einer ganzzahligen Länge von 1 Bit codiert werden. Dies wäre 6x mehr als das Optimum.

3.1.3 Codierung erweiterter Quellen

Wenn eine Quelle nur zwei Zeichen aussendet (binäre Quelle), dann müssten die Zeichen zu Symbolen zusammengefasst werden, damit die Huffman Codierung angewendet werden kann.

BEISPIEL 3.10: Zeichen der binären Nachrichtenquelle $\alpha = \{a, b\}$ mit $p_a = 0.15$ und $p_b = 0.85$ werden zu 2-bit Symbolen aa, ab, ba und bb zusammengefasst. Wie sieht der Code Baum der Huffman Codierung aus? Es sind die mittlere Codelänge und die Redundanz zu berechnen.
LÖSUNG: Mittlere Codelänge pro Symbol ist $\ell_K = 1.4275$ bit und damit pro binäres Zeichen der Quelle $l_C = 0.714$ bit. Das entspricht einer Kompressionsrate von 71.4%.

Ohne Zusammenfassung mehrere Bits zu einem Symbol wäre hier keine Kompression möglich. Durch das Zusammenfassen zu Symbolen wird die Quelle 'erweitert', d.h. das Quellenalphabet vergrößert. Sendet eine Quelle immer 2 Zeichen (=1 Symbol) gleichzeitig, dann ist die Entropie pro Symbol $2 \times H(X)$, mit $H(X)$ die Entropie pro Zeichen der Quelle.
Im obigen Beispiel ist $H(X) \approx 0.6$ bit pro Zeichen. Berechnen wir aber die Entropie der neuen Quelle, $\alpha_2 = \{aa,\ ab,\ ba,\ bb\}$, dann ergibt sich die Entropie von $H(X) \approx 1.2$ bit pro Symbol.

Als Güte der Quellencodierung wird oft die Kompressionsrate (anstatt Redundanz) angegeben. Die Kompressionsrate ist gegeben als das Verhältnis von der Anzahl der Bits pro Symbol nach und vor der Codierung.

Es ist zu erwarten, dass umso mehr Bits zu einem Symbol zusammengefasst werden, desto besser die Kompressionsrate. Dabei können die zu codierende Symbole aus unterschiedlichen Anzahl von Bits bestehen.

BEISPIEL 3.11: Zeichen der binären Nachrichtenquelle $\alpha = \{a, b\}$ mit $p_a = 0.15$ und $p_b = 0.85$ werden zu Symbolen a, ba, bba und bbb zusammengefasst. Wie sieht der Code Baum der Huffman Codierung aus? Es sind die mittlere Codelänge und die Kompressionsrate zu berechnen.
LÖSUNG: Mittlere Codelänge pro Symbol ist $\ell_K = 1.66$ bit. In etwa 100 Symbolen befinden sich $61\times bbb$, $13\times\ bba$, $11\times ba$ und $15\times a$. Uncodiert wäre der Text 257 bit lang, codiert $100 \cdot 1.66$ bit. Die Kompressionsrate ist demnach $R_K = 166/257 \approx 0.646$, also besser als im vorherigen Beispiel.

Werden immer K Zeichen einer Nachrichtenquelle gleichzeitig ausgesandt, dann lassen sich diese für die Entropiecodierung zu K-Zeichen Symbolen zusammenfassen. Das entspricht einer $K-$fachen Erweiterung einer <u>gedächtnislosen</u> Nachrichtenquelle, $X \to X^K$. Die Anzahl der unterschiedlichen Symbole der erweiterten Quelle ist dann M^K (für binäre Quellen: 2^K). Die Entropie einer solchen erweiterten Quelle wird wie die Entropie einer Verbundquelle berechnet.
Wenn zwei beliebige Symbole ($K = 2$) einer gedächtnislosen Nachrichtenquelle aufeinander folgend ausgesandt werden, dann gilt für die Wahrscheinlichkeit des k-ten Symbols $\underline{x}^{(k)} = (x_i, x_j)$: $p(\underline{x}^{(k)}) = p(x_i, x_j) = p(x_i) \cdot p(x_j)$. Insgesamt müssten dann M^2 Symbole $\left(x_i^{(k)}, x_j^{(k)}\right)$,

$k = 1, ..., M^2$, betrachtet werden. Für die Entropie folgt dann

$$\begin{aligned} H(X^2) &= -\sum_{i=1}^{M}\sum_{j=1}^{M} p(x_i, x_j) \cdot \log_2 (p(x_i, x_j)) \\ &= -\sum_{i=1}^{M}\sum_{j=1}^{M} p(x_i)p(x_j) \cdot (\log_2 (p(x_i)) + \log_2 (p(x_j))) \\ &= -\sum_{i=1}^{M} p(x_i) \log_2 (p(x_i)) \sum_{j=1}^{M} p(x_j) - \sum_{j=1}^{M} p(x_j) \cdot \log_2 (p(x_j)) \sum_{i=1}^{M} p(x_i) \\ &= H(X) + H(X) = 2H(X). \end{aligned}$$

Im Allgemein gilt für die Entropie der erweiterten Quelle $H(X^K) = K \cdot H(X)$. Die mittlere Codewortlänge wird nach $\ell_K = \sum_{i=1}^{M^K} p(\underline{x}^{(i)}) \cdot \ell_i$ berechnet. Bezogen auf die ursprüngliche Quelle gilt dann $\ell_C = \frac{\ell_K}{K}$. Gemäß dem Codiertheorem von Shannon wird die Coderedundanz solcher erweiterten Quellen umso kleiner, je größer K gewählt wird.

BEISPIEL 3.12: Betrachtet wird eine binäre Quelle, die x_1 und x_2 mit Wahrscheinlichkeiten $p_1 = 0.2$ und $p_2 = 0.8$ aussendet.
Zu vergleichen sind die Redundanzen der Codierung wenn

1. jedes Zeichen unabhängig codiert wird,
2. 2 Zeichen zusammen codiert werden (2−fache Erweiterung) mit der Huffman Codierung,
3. 3 Zeichen zusammen codiert werden (3−fache Erweiterung) mit der Huffman Codierung.

LÖSUNG:

1. Die Entropie der Quelle ist $H(X) = -\sum_{i=1}^{2} p(x_i) \log_2(p(x_i)) = 0.7219$ bit. Mit der Codierung 1 Bit pro Zeichen ist die resultierende Coderedundanz ist dann $R_C = 0.2781$ bit.
2. Mittlere Codelänge pro Symbol ist $\ell_K = 1.56$ bit, pro Zeichen $\ell_C = 0.78$ bit, die Kompressionsrate 78% und $R_C = 0.0581$ bit.
3. Mittlere Codelänge pro Symbol ist $\ell_K = 2.184$ bit, pro Zeichen $\ell_C = 0.728$ bit, die Kompressionsrate $72,8\%$ und $R_C = 0.0061$bit.

3.1.4 Codierung von Markov-Quellen

Bei der gedächtnisbehafteten (Markov) Quellen kann die mittlere Codelänge und damit auch die Redundanz durch Berücksichtigung der statistischen Bindungen nacheinander folgenden Zeichen

reduziert werden. Markov Quellen können als erweiterte Quellen betrachtet werden, wobei die statistische Unabhängigkeit der ausgesandten Zeichen nicht gegeben ist, sondern die Kenntnis der Übergangswahrscheinlichkeiten vorausgesetzt wird. Demnach müssen alle mögliche Übergänge betrachtet und getrennt codiert werden. Bei einer Markov Quellen erster Ordnung wird der Code anhand von bedingten Wahrscheinlichkeiten gebildet. Für jeden aktuellen Zustand ergibt sich einen Code mit einer mittleren Codewortlänge. Die resultierende mittlere Codewortlänge ist deutlich kleiner als wenn man die statistische Abhängigkeiten von aneinander folgenden Zeichen vernachlässigt.

Zu Decodierung auf der Empfängerseite muss der Huffman (bzw. Fano oder Shannon) Code bekannt sein. In Anwendungen wird der Codebaum mit übertragen oder empfänger-seitig erzeugt, wenn die Statistik der Quelle bekannt ist. Ebenfalls kann die Codierung an die aktuelle Statistik der Quelle angepasst werden ($\rightarrow$adaptive Huffman Codierung).

3.2 Arithmetische Codierung

Arithmetischen Codierung gehört zu den Verfahren, die auf Kenntnis der Quellenstatistik basieren. Dabei werden die Zeichenfolgen nicht auf Codewörter abgebildet, sondern der zu codierenden Zeichenfolge wird eine rationale Zahl zugeordnet. Das Codewort wird erst ausgegeben, nachdem die gesamte Zeichenfolge am Eingang eingelesen wird.

Proportional zu der statistischen Verteilung der Zeichen im Quellenalphabet wird das Ausgangsintervall $[0, 1)$ in Unterintervalle aufgeteilt, deren Bereiche dann das jeweils zugehörige Zeichen repräsentieren. Für das nächste Zeichen wird das aktuelle Intervall herangezogen und wiederum proportional zu den auftretenden Häufigkeiten unterteilt. Diese rekursive Intervall-Teilung und Zuweisung wird mit jedem weiteren Zeichen fortgesetzt, d.h. das Intervall des $n-$ten Zeichens ergibt sich aus der Unterteilung des Intervalls des $(n-1)-$ten Zeichens entsprechend der relativen Häufigkeiten.

BEISPIEL 3.13: Betrachtet wird eine gedächtnislose binäre Quelle mit dem Alphabet $\{A, B\}$ und der Wahrscheinlichkeitsverteilung $(\frac{1}{4}, \frac{3}{4})$. Es sollen die Sequenzen ABA und ABB codiert werden.

- ▷ Beim Codieren des ersten Zeichens wird das Intervall $[0, 1)$ entsprechend der Auftrittswahrscheinlichkeiten in Intervalle $A \in \left[0, \frac{1}{4}\right)$ und $B \in \left[\frac{1}{4}, 1\right)$ geteilt.
- ▷ Da das erste Zeichen A ist, wird im zweiten Schritt das Intervall $\left[0, \frac{1}{4}\right)$ geteilt in $AA \in \left[0, \frac{1}{16}\right)$ und $AB \in \left[\frac{1}{16}, \frac{1}{4}\right)$.
- ▷ Für das zweite Zeichen B wird das Intervall $\left[\frac{1}{16}, \frac{1}{4}\right)$, der Breite $\frac{3}{16}$, in $ABA \in \left[\frac{1}{16}, \frac{1}{16} + \frac{3}{16} \cdot \frac{1}{4}\right) = \left[\frac{4}{64}, \frac{7}{64}\right)$ und $ABB \in \left[\frac{7}{64}, \frac{16}{64}\right)$ geteilt.
- ▷ ABA wird mit einer Zahl aus dem Intervall $\left[\frac{4}{64}, \frac{7}{64}\right) = [0.0625, 0.109375)$ dargestellt. Eine geeignete Wahl wäre z.B. 0.1. ABB könnte entsprechend mit $0.2 \in [0.109375, 0.25)$ dargestellt werden.

Der Empfänger benötigt zur Decodierung die übertragene Zahl und dieselbe Intervall-Teilung, die er aus der Häufigkeitsverteilung erstellen kann. Zusätzlich muss entweder die Wortlänge bekannt sein, oder ein spezielles Zeichen das Wort-Ende deuten.

Die Decodierung funktioniert analog zu Codierung. Im ersten Schritt wird überprüft, in welches Intervall die übertragene Zahl fällt, und daraus das erste Zeichen bestimmt. Danach wird das gewählte Intervall erneut skaliert und überprüft in welches Intervall die Zahl dann fällt. Dieser Vorgang wird bis zum Ende des Wortes wiederholt.

BEISPIEL 3.14: Die Wahrscheinlichkeitsverteilung $(\frac{1}{4}, \frac{3}{4})$ der gedächtnislose binäre Quelle mit dem Alphabet $\{A, B\}$ sei bekannt. Die komprimierte Sequenz wird als $y = 0.1$ gelesen.

- ▷ Die erste Intervall-Teilung führt zu $A \in \left[0, \frac{1}{4}\right)$ und $B \in \left[\frac{1}{4}, 1\right)$. Da y in dem ersten Intervall liegt, wird A als erstes Zeichen ausgegeben.
- ▷ Mit dem ersten Zeichen A folgt die weitere Intervall-Teilung $AA \in [0, 0.0625)$ und $AB \in [0.0625, 0.25)$. Da y jetzt im zweiten Intervall liegt, wird als zweites Zeichen B ausgegeben.
- ▷ Für das dritte Zeichen wird das Intervall $[0.0625, 0.25)$ in $ABA \in [0.0625, 0.109375)$ und $ABB \in [0.109375, 0.25)$ geteilt. Da y jetzt im ersten Intervall liegt, wird als drittes Zeichen A ausgegeben. So kann man z. B. mit der JPEG-Komprimierung 50 % der Datenmenge einsparen.

Offensichtlich ist, dass umso kleiner das errechnete Intervall, desto mehr Bits sind für die Codierung notwendig.

Im Allgemeinen kann die Arithmetische Codierung näher an die Entropie kommen als die Huffman Codierung. Sie ist insbesondere vom Vorteil bei einem kleinen Alphabet mit unterschiedlichen Wahrscheinlichkeiten für einzelne Zeichen. Allerdings werden bei langen Nachrichten Codeintervalle mit sehr vielen Nachkommastellen erreicht. Das Problem der Genauigkeit der Darstellung kann zwar umgegangen werden, das Problem bleibt die Geschwindigkeit des Decoders. Außerdem muss die gesamte Nachricht am Sender/Empfänger bereit stehen, bevor sie codiert/decodiert werden kann.

BEISPIEL 3.15: Das Alphabet besteht aus zwei Symbolen x_1, x_2 mit $p_1 = 0.4$, $p_2 = 0.5$ wobei 'end of file' mit der Wahrscheinlichkeit $p(eof) = 0.1$ auftritt. Es soll die Sequenz $x_2x_2x_2eof$ codiert werden.
Lösung: $[0, 1) \rightarrow [0.4, 0.9) \rightarrow [0.6, 0.85) \rightarrow [0.7, 0.825) \rightarrow [0.8125, 0.825)$. Die Wahrscheinlichkeit für $x_2x_2x_2eof$ ist 0.0125 und $-\log_2 0.0125 \approx 6.322$. Binäre Darstellung der Intervall-Grenzen führt somit zu der Codierung mit 7 Bits: 1101000.

3.3 Verfahren ohne Kenntnis der Quellenstatistik

Bisher vorgestellte Verfahren generieren ein Codebuch, der Statistik-basiert und daher unveränderbar ist. Das verwendete Codebuch ist sender- und empfangs-seitig bekannt. Wenn die Statistik der Quelle nicht bekannt ist, dann können 'Wörterbuch-Methoden' verwendet werden, wobei das Wörterbuch während des Codiervorgangs erstellt wird. Solche Methoden suchen nach wiederholt auftretenden Zeichenketten in einer Nachricht (z.B. Text) und codieren diese mit dem entsprechenden Index aus dem Wörterbuch. Die resultierende Codefolge besteht aus Indexen und und (Original-)Zeichen(-folgen). In der Regel wird die Redundanz beim Codieren von Zeichenketten kleiner als bei Codieren einzelner Symbole. Die Reduktion der Redundanz wird dadurch erzielt, dass möglichst lange Zeichenketten in das Codebuch eingetragen werden, was erst bei langen Texten passiert.

Unter dem Begriff Lempel-Ziv (LZ) verbergen sich unterschiedlichen Varianten eines solches Verfahrens, die sich vor allem dadurch unterscheiden, wie das Codebuch aufgebaut und genutzt wird.

3.3.1 Willems Algorithmus

Der Algorithmus von Willems codiert eine feste Quellenwortlänge auf eine variable Codewortlänge mit einem präfixfreien Code, ohne die Kenntnis der Quellenstatistik. Die Nachrichtenfolgen der Quelle sind beliebig lang, die Symbole aus einem endlichen Alphabet.
Die Nachricht wird in Teilfolgen konstanter Länge K unterteilt, wobei $K = 2^p - 1$ gewählt wird mit $p \geq 2$. Für jede Teilsequenz wird die Zeit t bestimmt, wann die Sequenz zuletzt aufgetreten war. Das wesentliche Prinzip besteht darin, den Sequenzen mit kleinen Wiederholungszeiten eine kurze Codewortlänge zuzuordnen. In einer Codetabelle werden die binäre Repräsentationen (Codewörter) der Wiederholungszeiten generiert.

Es wird ein Puffer der Länge $2^K - 1$ angelegt und mit einer beliebigen (dem Decoder bekannten) Symbolfolge initialisiert. Die Nachricht der Länge N wird in $k = N/K$ Teilsequenzen unterteilt, die dann nacheinander von rechts in den Puffer geschoben werden. Der Encoder bestimmt die Wiederholungszeit t der aktuellen Teilsequenz. Ist diese kleiner gleich Pufferlänge, so liest man aus dem Codetabele ein Codewort aus, welches die Wiederholungszeit repräsentiert. Ist die aktuelle Teilsequenz nicht im Puffer enthalten, dann wird sie mit einem Präfix versehen ausgegeben. Der Pufferinhalt wird um K Stellen nach links verschoben. Dieser Vorgang wird so lange wiederholt bis die Nachrichtenfolge zu Ende ist.

Wiederholungszeiten in der Codetabelle werden ein Präfix i und ein Suffix j zugeordnet, wobei $0 \leq i, j \leq K$, $i \leq \log_2(t) < i + 1$ und $j = t - 2^i$ gilt. Für $t \geq 2^K$ setzt man $i = K$, die aktuelle Teilsequenz wird als Suffix genommen. Präfix und Suffix werden binär codiert, wobei i mit $\lceil \log_2(K + 1) \rceil$ Bits und j mit i Bits codiert wird.

Für z.B. $K = 3$ ergibt sich folgende Codetabelle:

t_r	i	j	Präfix binär	Suffix binär	Codewortlänge
1	0	0	00	-	2
2	1	0	01	0	3
3	1	1	01	1	3
4	2	0	10	00	4
5	2	1	10	01	4
6	2	2	10	10	4
7	2	3	10	11	4
≥ 8	3	-	11	Teilsequenz	2+Suffixlänge

BEISPIEL 3.16:

Betrachtet wird das Bild unten bestehend aus $3 \times 5 = 15$ Pixeln. Jeder Pixel stellt eine von 4 möglichen Graustufen dar: $A = 00$, $B = 01$, $C = 10$ und $D = 11$. Das Bild wird somit ohne Quellencodierung mit 30 Bits dargestellt. Das Bild wird zeilenweise gelesen, was der Nachricht $AABAAAAAAACCACC$ entspricht. Teilen wir diese Folge in Teilsequenzen der Länge $K = 2^2 - 1 = 3$ und initialisieren den Puffer der Länge $2^K - 1 = 7$ mit $[AAAAAAA]$.

A	A	B	A	A
A	A	A	A	A
C	C	A	C	C

Mit Hilfe der Tabelle im vorigen Beispiel ergeben sich folgende Codierungsschritte:

1. [Puffer]Teilsequenz: $[AAAAAAA]\,AAB \Rightarrow t > 7 \Rightarrow i = 3,\ j = 000001,$ $\underline{s} = 11000001$
2. [Puffer]Teilsequenz: $[AAAAAAB]\,AAA \Rightarrow t = 4 \Rightarrow i = 2,\ j = 0, \underline{s} = 1000$
3. [Puffer]Teilsequenz: $[AAABAAA]\,AAA \Rightarrow t = 1 \Rightarrow i = 0$, kein Suffix, $\underline{s} = 00$
4. [Puffer]Teilsequenz: $[BAAAAAA]\,ACC \Rightarrow t > 7 \Rightarrow i = 3,\ j = 001010,$ $\underline{s} = 11001010$
5. [Puffer]Teilsequenz: $[AAAAACC]\,ACC \Rightarrow t = 3 \Rightarrow i = 1,\ j = 1, \underline{s} = 011$

Die resultierende codierte binäre Nachricht ist 25 Bit lang: 1100000110000011001010011.

Durch eine Vergrößerung des Puffers kann die Effizient der Kompression verbessert werden. Es lässt sich auch zeigen, dass der Willems Algorithmus asymptotisch (bei unendlich großer Pufferlänge) optimal ist und die Entropierate erreicht. Die Alternative zum Willems Algorithmus ist der Lempel-Ziv Algorithmus, der eine variable Quellenwortlänge auf einen feste Codewortlänge abbildet.

3.3.2 Lempel-Ziv-Welch Algorithmus (LZW)

LZW ist eine Variante vom LZ78 Algorithmus, wobei der Unterschied zum LZ78 darin besteht, dass das Codebuch am Anfang nicht leer ist, sondern mit 2^8 ASCII Zeichen initialisiert wird (Index $1...256$). Jedem weiteren Wörterbucheintrag wird ein Index i und ein Binärcode der Länge 12 Bit eindeutig zugeordnet. Der LZW Algorithmus arbeitet die eingegebene Zeichenfolge sequentiell ab und sucht dabei nach der jeweils längsten Zeichenkette, die bereits im Codebuch eingetragen ist. Neue Zeichenketten werden durch das Anhängen des nächsten Zeichen an eine bereits eingetragene Zeichenkette und den Eintrag in das Codebuch mit dem entsprechenden Index codiert.

BEISPIEL 3.17: Das Quellenalphabet besteht aus Zeichen $\{a, b, c\}$. Die Nachricht $baacbacbaacba$ soll mit dem LZW Algorithmus codiert werden.

Das Initial-Codebuch beinhaltet die Einträge

i	z_i
1	a
2	b
3	c

.

$\overbrace{b\ a}\,acbacbaacba$: Das erste Eingabezeichen b entspricht dem Eintrag Nr. 2, die Zeichenkette ba ist noch nicht vorhanden und wird dann unter Index $i = 4$ eingetragen. Die aktuelle Ausgabe (als Index) ist 2, wobei a als Präfix für nächste Zeichenkette betrachtet wird.

$b\overbrace{a\ a}\,cbacbaacba$: aa wird in das Codebuch mit dem fortlaufenden Index $i = 5$ eingetragen und für das zweite Eingabezeichen das Codewort $i = 1$ binär ausgegeben. Das dritte Zeichen (das zweite a) ist jetzt der Präfix für die nächste suchende Zeichenkette.
$ba\overbrace{a\ c}\,bacbaacba$: ac ist im Codebuch nicht vorhanden und wird eingetragen mit $i = 6$, die Codefolge wird mit $i = 1$ für a fortgesetzt.
$baa\overbrace{c\ b}\,acbaacba$: Da cb noch nicht im Codebuch vorhanden ist, wird $i = 3$ für c ausgegeben und cb mit $i = 7$ in das Codebuch eingetragen.
$baac\overbrace{ba\ c}\,baacba$: ba wird als $4-$ter Eintrag ausgegeben, bac ist der neue $8-$te Eintrag in das Codebuch.
$baacba\overbrace{cb\ a}\,acba$: cb wird als $7-$ter Eintrag ausgegeben, cba ist der neue $9-$te Eintrag in das Codebuch.
$baacbacb\overbrace{aa\ c}\,ba$: aa wird als $5-$ter Eintrag ausgegeben, aac ist der neue $10-$te Eintrag in das Codebuch.
$baacbacbaa\overbrace{cba}$: cba wird als der 9. Codebucheintrag ausgegeben.

Das Codebuch des Encoders:

i	z_i
4	ba
5	aa
6	ac
7	cb
8	bac
9	cba
10	aac

In dem obigen Beispiel resultiert die Codierung in einer mittleren Codelänge von $96/13 \sim 7.4$ bit/Zeichen bzw. einer Kompressionsrate von $K_C = \frac{96}{104} = 0.923$. Offensichtlich ist der Text zu kurz um noch besser komprimiert zu werden.

LZW Codierung wird erst mit größeren Nachrichten (Texten) effektiv. Grundsätzlich gilt, dass Texte mit einer kleinen Anzahl von oft wiederholenden Zeichenmustern besser komprimiert werden.

Das Problem der LZW Codierung wird die Suche in dem Codebuch, wenn die Anzahl der Einträge groß wird. Anderseits wird mit einem kleinen Codebuch (unzureichende Kapazität) die Datenkompression schlecht.

Um die komprimierte Daten wieder zu rekonstruieren, müsste der Empfänger (Decoder) das verwendete Codebuch besitzen. Der Decoder kann allerdings die Systematik der Codebucheinträge ausnutzen um während des Decodiervorgangs das Codebuch zu generieren. Die Systematik besteht darin, dass das letzte Zeichen eines Eintrags immer dem ersten Zeichen des folgenden Eintrags entspricht.

Beim Decodieren wird also jedes empfangene Codewort als eine im Codebuch gefundenen Zeichenkette interpretiert und ausgegeben. Diese wird gleichzeitig gespeichert und als Präfix mit dem ersten Zeichen der folgende Zeichenkette verknüpft und in das Codebuch eingetragen.

BEISPIEL 3.18: Die binäre Codefolge wird in die Indexfolge $2-1-1-3-4-7-5-9$ umgewandelt und soll decodiert werden. Nur das Initial-Codebuch ist bekannt:

i	z_i
1	a
2	b
3	c

.

2 wird an b interpretiert, ausgegeben und zwischengespeichert.
1 wird als a interpretiert und ausgegeben, ba wird mit $i = 4$ in das Codebuch eingetragen, a zwischengespeichert.
1 wird als a interpretiert und ausgegeben, aa wird mit $i = 5$ in das Codebuch eingetragen, a zwischengespeichert.
3 wird als c interpretiert und ausgegeben, ac wird mit $i = 6$ in das Codebuch eingetragen, c zwischengespeichert.
4 wird als ba interpretiert und ausgegeben, cb wird mit $i = 7$ in das Codebuch eingetragen, ba zwischengespeichert.
7 wird als cb interpretiert und ausgegeben, bac wird mit $i = 8$ in das Codebuch eingetragen, cb zwischengespeichert.
5 wird als aa interpretiert und ausgegeben, cba wird mit $i = 9$ in das Codebuch eingetragen, aa zwischengespeichert.
9 wird als cba interpretiert und ausgegeben, aac wird mit $i = 10$ in das Codebuch eingetragen, das Ende der Codesequenz erkannt und Decodiervorgang gestoppt.

Ausgabe: $baacbacbaacba$.

Das Codebuch des Decoders:

i	z_i
4	ba
5	aa
6	ac
7	cb
8	bac
9	cba
10	aac

4 Verlustbehaftete Kompression

Im Vergleich zu einem Buch, benötigt eine Sprach-/Musikaufnahme oder ein Bild deutlich mehr Speicherplatz, wenn sie auf einem Rechner verarbeitet oder gespeichert werden soll.

Akustische Signale, wie Sprache und Musik, werden bei einer Aufnahme, z.B. über einen Mikrofon, in Audiosignale, d.h. elektrische Signale umgewandelt. Um sie dann zu komprimieren und speichern müssen sie erstmals digitalisiert werden.

Das menschliche Ohr kann akustische Ereignisse nur innerhalb eines bestimmten Frequenz- und Schalldruckpegel-Bereichs wahrnehmen, der auch als Hörfläche bezeichnet wird. Die wahrnehmbare Lautstärke, zwischen der unteren und der obere Hörschwelle, liegt zwischen den Punkten der tiefsten hörbaren Frequenz von 20 Hertz und der höchsten hörbaren Frequenz, die je nach Alter bis maximal 22 kHz beträgt. Dabei verläuft die Hörschwelle des Menschen nicht linear. Den Punkt der höchsten Wahrnehmungsempfindlichkeit ist z.B. bei etwa 4 kHz. Menschliche Stimme liegt wiederum im Bereich zwischen 500Hz und 2kHz. Diese und weitere akustische Effekte, wie z.B. Maskierungseffekte beim menschlichen Gehör, können bei der Audiokompression ausgenutzt werden, indem von Menschen nicht wahrnehmbare Signale eliminiert werden.

Ähnlich ist es mit der Bildkompression. Im Allgemeinen bestehen digitale Bilddaten aus RGB oder Luminanz/Chrominanz[1] Information für jeden Pixeln, wobei RGB aus LC Werten berechnet werden können und andersherum. Da ein Bild eine subjektive Wahrnehmung ist, können die vom menschlichen Auge nicht wahrnehmbare und somit irrelevante oder wenig relevante Informationsanteile entfernt werden um bessere Kompressionsrate zu erzielen. So wird z.B. die Helligkeitsinformation vom menschlichen Auges in einer höheren Auflösung wahrgenommen als die Farbe, so dass viele auf dem LC-Farbmodell aufbauende Formate eine Reduzierung in der Auflösung der Chrominanz vornehmen. Es handelt sich hierbei um einen verlustbehafteten bzw. irreversiblen Quellencodierung.

Audio- und Bildkompression kann auch verlustlos sein. GIF und PNG sind z.B. Bildformaten, die LZW Codierung verwenden. Verlustbehaftete Verfahren erzielen allerdings besser Kompressionsraten. Sie basieren in der Regel auf der Datenreduktion im Frequenzbereich, d.h. verwenden eine Transformationscodierung des Originalsignals und eine Quantisierung im Frequenzbereich.

4.1 Ansätze zu Bildkompression

Bildkompression basiert auf der Tatsache, dass benachbarte Pixel ähnlich, also korreliert sind. Unterschiedliche Bildtypen werden allerdings unterschiedlich codiert.

[1] L/C basierte Farbmodelle sind

- ▷ YCbCr, für das Digitalfernsehen, digitale Bild- und Videoaufzeichnung, bei JPEG-Bildern, MPEG-Videos und damit auch bei DVDs, Video CDs sowie den meisten anderen digitalen Videoformaten verwendet.
- ▷ YPbPr, für analoge Übertragung von Videosignalen
- ▷ YUV für die analoge Fehrnsehtechnik

1. In schwarz-weißen Bildern entspricht ein Pixel einem Bit (s/w bzw. $0/1$). Angenommen, dass viele benachbarte Pixel gleich sind, wird die Run Length Codierung (RLE) angewendet. Die resultierende Längen der 'Runs' können dann mit einem Code variabler Länge codiert werden und übertragen.
2. Andere Methode für Kompression von Schwarz-Weiß Bildern ist es, n benachbarte Pixel als eine $n-$bit Zahl aufzufassen, die den Inhalt des Pixels beschreibt. Dies resultiert in 2^n möglichen Pixelinhalten mit unterschiedlichen Auftrittswahrscheinlichkeiten. Anschließend kann z.B. die arithmetische Codierung, basierend auf Wahrscheinlichkeiten der Pixel, angewandt werden.
3. In graustufigen Bildern wird jedes Pixel mit n Bits dargestellt werden, d.h. es gibt 2^n Graustufen. So ein Bild kann in n zweistufigen Bilder zerlegt werden, die dann wie unter 1. weiter komprimiert werden. Damit aber die binäre Darstellung der benachbarten Pixeln ähnlich bleibt (wenig unterschiedliche Bits) wird für die binäre Darstellung der *Gray Code*[2] verwendet.
4. In graustufigen Bildern kann der Inhalt eines Pixels A aus einer gewählten Gewichtung und Mitteilung von einer vorbestimmten Anzahl der benachbarten Pixeln bestimmt werden. Dieser Wert soll dem Pixel Wert P ähneln, d.h. die Differenz $\Delta = P - A$ ist eine kleine Zahl. Die Differenz Δ wird dann mit einem präfixfreien Code codiert und übertragen.
5. In bisher vorgestellten Verfahren wird die Nachricht im sogenannten Originalbereich (Zeitbereich) codiert. Bei einigen Verfahren zu Bildkompression wird jedoch die Codierung im Spektralbereich durchgeführt, d.h. das Signal wird einer Transformation unterzogen. Bei Bilddaten ist die Information der benachbarten Pixeln typischerweise korreliert. Betrachtet man die Information im spektralen (Frequenz-) Bereich, stellt man fest dass viele spektrale Koeffizienten mit Nullen besetzt sind und damit hier eine effizientere Entropie-Codierung durchgeführt werden kann. Vor der Codierung wird durch eine Quantisierung im Spektralbereich die wenig relevante (irrelevante) Information reduziert (eliminiert). Derart durchgeführte Codierung ist irreversibel.
6. Farbige Bilder können drei Farbkomponenten, rot, grün und blau, zerlegt werden, wobei jede Farbkomponente dann als ein Graustufen-Bild komprimiert wird.
7. Farbige Bilder mit diskrete Farb-Übergängen (z.B. Text + Symbole) werden in (wiederholende) Bereiche unterteilt. Jeder Bereich wird einmal (beim ersten Auftreten) übertragen. Bei der Wiederholung wir nur der Zeiger auf den übertragenen Bereich ausgegeben.

Im folgenden betrachten wir die Transformationscodierung, die zur verlustbehafteten Bildkompression verwendet wird.

4.2 Transformationscodierung

Für die Abbildung im Spektralbereich werden diskrete Transformationen endlicher wert-diskreter Abtastfolgen verwendet. Die Transformationscodierung basiert auf Reduktion der statistischen Bindungen zwischen benachbarten Werten. Eine gedächtnisbehaftete Quelle mit einer nachgeschalteten unitären diskreten Transformation modelliert eine gedächtnislose Nachrichtenquelle. Allerdings, um die Werte optimal zu de-korrelieren, müssten die Korrelationen der aktuellen Quellenzeichen bekannt sein, was nie der Fall ist. Daher wird normalerweise, unabhängig von der Statistik der Quelle, eine bekannt unitäre Transformationsmatrix verwendet.

[2] Gray Code ist ein dualer Code, bei dem sich zwei benachbarte Codewörter nur an einer Stelle unterscheiden.

Meistens wird der Spektralbereich mit der Diskreten FOURIER Transformation (DFT) beschrieben. Die spektralen Koeffizienten beschreiben die diskrete Frequenzen, die im Signal vorkommen. In der Bild- und Audiocodierung kommt jedoch oft die *Diskrete Cosinus Transformation* (DCT) zum Einsatz, da sie bessere Dekorrelationseigenschaften besitzt. Da beide Transformationen verwandt sind, führen wir die DCT als ein Sonderfall der DFT ein.

Ein Bild kann als eine Matrix mit Pixeln beschrieben werden, wobei benachbarten Zeilen/Spalten korreliert (ähnlich) sind. Die Transformationscodierung wird dann nacheinander in beiden Richtungen durchgeführt. Die resultierende Matrix mit spektralen Koeffizienten ist schwach besetzt bzw. besteht im unteren Teil aus kleinen Werte, die durch Quantisierung auf 0 gesetzt werden. D.h. die eigentliche Kompression entsteht erst durch die Quantisierung der Werte im Spektralbereich.

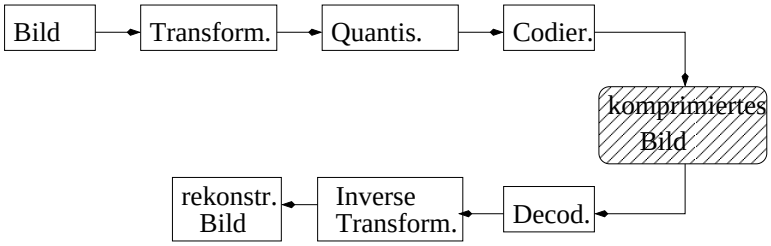


Bild: Prinzip der verlustbehafteten Codierung

Bei der verlustbehafteten JPEG Kompression werden z.B. Bilddaten in 8×8 Pixel-Blöcke geteilt und jeder Block wird mit DCT transformiert. Die Spektralkoeffizienten werden anschließend quantisiert, womit die Anzahl der Bits benötigt für deren Darstellung reduziert wird. Anschließend wird eine Entropie-Codierung der binären Daten vorgenommen.

4.3 Abtastung und Quantisierung

Erfassung der Information von einer kontinuierliche Quelle kann nur zu diskreten Zeiten und in diskreten Werten erfolgen. Zeitliche Diskretisierung (Abtastung) bedeutet, dass wir die Amplituden eines Signals in bestimmten konstanten Abständen, d.h. zu diskreten Zeitpunkten $t_n = nT$, betrachten. Wir speichern also nur die momentane Amplitudenwerte $A_n = A(t_n)$. Teilen wir den Amplitudenbereich, der alle Amplituden des betrachteten Signals umfasst, in Q Intervalle und ordnen jedem Intervall einen festen Wert a_q zu, dann können wir den gesamten Amplitudenbereich mit Q Werten beschreiben. Alle Amplituden, die im selben Intervall liegen, werden mit dem gleichen quantisierten Wert dargestellt. Aus einer kontinuierlichen Quelle ist eine diskrete Quelle mit Q Quellenzeichen geworden. Wenn die Amplitudenwerte a_q, $q = 0, .., Q - 1$, dann noch mit einem Binärcode dargestellt werden, sprechen wir von einem digitalen Signal.

Die gleichmäßige Abtastung, Quantisierung und Darstellung des Wertebereiches mit einem binären Code sind auch als Pulse Coded Modulation (PCM) bekannt. Offensichtlich ist dass umso höher die Abtastrate $\frac{1}{T}$, desto mehr Abtastwerte müssen gespeichert werden, aber umso besser kann auch das

rekonstruierte analoge Signal wiedergegeben werden. Allerdings kann die Abtastrate nicht beliebig klein gewählt werden.

Damit das analoge Signal wieder rekonstruiert wird, muss die Abtastrate, $\frac{1}{T}$, größer oder gleich sein der doppelten maximalen im Signal vorkommenden Frequenz f_{max}. Die Abtastfrequenz $f_a = 2 \cdot f_{max}$ wird auch die *Nyquist Rate* genannt. Für eine gute Audiowiedergabe sollte das digitale Signal mit einer Abtastfrequenz von mindestens 44 kHz generiert werden. Sprache wird allerdings auch bei einer Abtastung mit 8 kHz verständlich.

Bei einer gleichmäßigen Quantisierung wird der quantisierte Wert $[A_n]_q = a_q(n)$, d.h. $\forall A_n \in \left[\frac{(q-1)\cdot A_{max}}{Q}, \frac{q \cdot A_{max}}{Q}\right)$ ist $a_q(n) = \lfloor \frac{A_n}{\Delta} + \frac{1}{2} \rfloor$, wobei $\Delta = \frac{A_{max}}{Q}$ die feste Quantisierungsstufe darstellt.

Die Anzahl der möglichen Quantisierungsstufen Q ergibt sich bei dem Binärcode aus der Anzahl b der Bits, die ein Codewort aufweist: $Q = 2^b$. Die Zahl der Quantisierungsstufen bestimmt wesentlich das Quantisierungsrauschen. Je größer die Quantisierungsstufen werden, desto größer ist der resultierende Quantisierungsfehler e_q, mit $-\frac{\Delta}{2} \leq e_q(n) \leq \frac{\Delta}{2}$. Häufig wird angenommen, dass der Quantisierungsfehler einer gleichverteilten mittelwertfreien Zufallsvariable mit der Varianz $\sigma_q^2 = \frac{\Delta^2}{12}$ entspricht.

Die Quantisierung mit gleichmäßig großen Wertebereichen wird als linear bezeichnet. Bei einer nichtlinearen Quantisierung werden größere Amplituden gröber aufgelöst und kleine Signalamplituden mit einer höheren Auflösung quantisiert. Der Vorteil besteht darin, dass mit weniger Bit pro Abtastwert ein geringeres Quantisierungsrauschen als bei linearer Quantisierung erzielt werden kann.

Bei der Sprachübertragung, also in Telefonie und Mobilfunk, wird die gesprochene Sprache, mittels einen Vocoders codiert und komprimiert, als analoges Signal übertragen, um am Zielort wieder synthetisiert. Solche Sprachvocoder verwenden die Vektorquantizierung. Bei der Vektorquantizierung werden die Datensätze in Merkmalsvektoren zusammengefasst. Dabei wird jedem Merkmalsvektor ein von Q vordefinierten (in einer Tabelle bzw. einem Codebuch gespeicherten) Vektoren nach dem Ähnlichkeitsprinzip zugeordnet. Statt alle Daten des Merkmalsvektors zu speichern, wird nur der Index dieses ähnlichsten Vektors benötigt.

Zur Datenübertragung wird nur der Index des Codebuchvektors benötigt. Der korrespondierende Decoder muss über das gleiche Codebuch verfügen und kann dann aus dem Index eine Approximation des ursprünglichen Vektors erzeugen.

5 Anhang

5.1 Diskrete Fourier Transformation

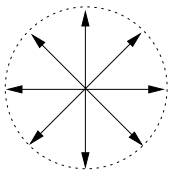
ZUR ERINNERUNG:

▷ Fourier Transformation $s(t) \circ\!\!-\!\!\bullet S(f)$: $S(f) = \int_{-\infty}^{\infty} s(t) exp(-j2\pi ft)dt$

▷ Inverse Fourier Transformation $S(f) \bullet\!\!-\!\!\circ s(t)$: $s(t) = \int_{-\infty}^{\infty} S(f) exp(j2\pi ft)df$

▷ Komplexer Drehzeiger:

$e^{j2\pi f_0 t} = \cos(2\pi f_0 t) + j\sin(2\pi f_0 t)$
$\left|e^{j2\pi f_0 t}\right| = 1$
$e^{j2\pi f_0 t} \circ\!\!-\!\!\bullet \delta(f - f_0)$



Die Diskrete Fourier Transformation (DFT) bildet eine endliche Abtastfolge $\underline{s} = (s_1, s_2, ..., s_n, ..., s_N)$, die das zeitdiskrete Signal mit $t_n = nT$ darstellt, auf eine diskrete Spektralfolge $\underline{S} = (S_1, S_2, ..., S_k, ..., S_N)$, mit diskreten Frequenzen $f_k = k \cdot \frac{1}{T}$, eindeutig gemäß der Transformationsvorschrift

$$S_k = \sum_{n=0}^{N-1} s_n exp(-j2\pi \frac{nk}{N}).$$

Die inverse Diskrete Fourier Transformation (IDFT), $F^{-1}: \underline{S} \bullet\!\!-\!\!\circ \underline{s}$ erfolgt durch

$$s_n = \frac{1}{N} \sum_{k=0}^{N-1} S_k exp(j2\pi \frac{nk}{N}).$$

▷ Es gilt $\frac{1}{N} \sum_{n=0}^{N-1} exp(j2\pi \frac{n(k-m)}{N}) = \delta_{k-m} = \begin{cases} 1 & \text{für } k - m = \ell \cdot N \\ 0 & \text{sonst} \end{cases}$

Kronecker Delta: $\delta_i = \begin{cases} 1 & \text{für } i = 0 \\ 0 & \text{für } i \neq 0 \end{cases}$

▷ $IDFT(DFT(\underline{s})) = \underline{s}$:

$$s_n = \frac{1}{N} \sum_{k=0}^{N-1} S_k exp(j2\pi \frac{nk}{N}) = \sum_{k=0}^{N-1} \frac{1}{N} \sum_{\ell=0}^{N-1} s_\ell exp(-j2\pi \frac{\ell k}{N}) exp(j2\pi \frac{nk}{N}) =$$

$$\sum_{\ell=0}^{N-1} s_\ell \frac{1}{N} \sum_{k=0}^{N-1} exp(-j2\pi \frac{(\ell-n)k}{N}) = \sum_{\ell=0}^{N-1} s_\ell \delta_{\ell-n} = s_n$$

▷ DFT ist immer periodisch mit N: $exp(-j2\pi n \frac{(k+N)}{N}) = exp(-j2\pi n \frac{k}{N}) \Rightarrow \underline{S}_k = \underline{S}_{k+N}$

▷ Allgemein gilt $S_{-k} = S_{N-k}$:

$$S_{N-k} = \sum_{n=0}^{N-1} s_n exp\left(-j2\pi \frac{n(N-k)}{N}\right) = \sum_{n=0}^{N-1} s_n exp\left(-j2\pi \frac{n(-k)}{N}\right) = S_{-k}$$

Führen wir den Drehfaktor $w = exp(-j\frac{2\pi}{N})$ ein. Es gilt $w^N = 1$, d.h. w ist die $N-$te Einheitswurzel, und $w^{-1} = w^*$.

Die DFT und IDFT lassen sich dann folgendermaßen darstellen:

$$S_k = \sum_{n=0}^{N-1} s_n w^{nk} \text{ und } s_n = \frac{1}{N} \sum_{k=0}^{N-1} S_k w^{-nk},$$

bzw. in der Matrixdarstellung, Beispiel für $N = 4$:

$$\begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{bmatrix} = \begin{bmatrix} w^0 & w^0 & w^0 & w^0 \\ w^0 & w^1 & w^2 & w^3 \\ w^0 & w^2 & w^4 & w^6 \\ w^0 & w^3 & w^6 & w^9 \end{bmatrix} \cdot \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & w^1 & w^2 & w^3 \\ 1 & w^2 & 1 & w^2 \\ 1 & w^3 & w^2 & w^1 \end{bmatrix} \cdot \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix}.$$

DFT ist also eine lineare Abbildung $DFT: \ \underline{s} \mapsto \underline{S} = \underline{W}\underline{s}$, mit der $N \times N$ Matrix $\underline{W}$, wobei $\underline{s}$ und $\underline{S}$ die Spaltenvektoren der Abtastwerte im Zeit- und Frequenzbereich darstellen.

Die Rücktransformation ist in dem Beispiel mit $N = 4$ dann

$$\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} w^0 & w^0 & w^0 & w^0 \\ w^0 & w^{-1} & w^{-2} & w^{-3} \\ w^0 & w^{-2} & w^{-4} & w^{-6} \\ w^0 & w^{-3} & w^{-6} & w^{-9} \end{bmatrix} \cdot \begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & w^{-1} & w^{-2} & w^{-3} \\ 1 & w^{-2} & 1 & w^{-2} \\ 1 & w^{-3} & w^{-2} & w^{-1} \end{bmatrix} \cdot \begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{bmatrix}$$

bzw. IDFT: $\underline{S} \longmapsto \underline{s} = \frac{1}{N}\underline{W}^*\underline{S}$. Es folgt dass $\underline{W}^{-1} = \frac{1}{N}\underline{W}^*$.

Berechnen wir die Energie des diskreten Signals im Zeitbereich, $\sum_{n=0}^{N-1} |s_n|^2 = \underline{s}^T \cdot \underline{s}$, und die Energie des diskreten Signals im Frequenzbereich, $\sum_{k=0}^{N-1} |S_k|^2 = \underline{S}^T \cdot \underline{S}$, werden wir feststellen, dass sie sich um den Faktor N unterscheiden: $\frac{\sum_{k=0}^{N-1} |S_k|^2}{\sum_{n=0}^{N-1} |s_n|^2} = N$.

Normalerweise werden für Kompressionsverfahren orthogonale diskrete Transformationen verwendet, bei denen die Transformationsmatrix eine unitäre Matrix ist, d.h. $\underline{W}^* \cdot \underline{W} = I$, mit der Einheitsmatrix I. Das bedeutet, dass die Energie im Original- und Bildbereich, hier also im Zeit- und Frequenzbereich gleich bleibt. Die DFT wird unitär, wenn die Transformationsbeziehungen

$$S_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} s_n exp(-j2\pi \frac{nk}{N}) \text{ und } s_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} S_k exp(j2\pi \frac{nk}{N})$$

verwendet werden. Die Transformationsmatrizen sind dann

$$\underline{W} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ w^0 & w^{1\cdot 1} & \ddots & w^{(N-1)\cdot 1} \\ \vdots & \vdots & \ddots & \vdots \\ w^0 & w^{(N-1)\cdot 1} & \cdots & w^{(N-1)\cdot(N-1)} \end{bmatrix} \text{ und}$$

$$\underline{W}^{-1} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ w^0 & w^{-1\cdot 1} & \ddots & w^{-(N-1)\cdot 1} \\ \vdots & \vdots & \ddots & \vdots \\ w^0 & w^{-(N-1)\cdot 1} & \cdots & w^{-(N-1)\cdot(N-1)} \end{bmatrix}.$$

DFT erzeugt aus N Datensamples N Spektralkoeffizienten, unter der Annahme dass die Daten, wie auch das Spektrum, periodisch sind mit der Periode N. Nach N Samples werden die Daten also wiederholt. Sind die Datensamples rein reell, dann lässt sich zeigen, dass das Spektrum symmetrisch ist, wobei sein Realteil eine gerade und der Imaginärteil eine ungerade Funktion darstellt. Ein rein reelles Spektrum bedeutet, das die Zeitfunktion geraden Realteil und ungeraden Imaginärteil besitzt.

DCT ist verwandt mit der DFT. DCT kann durch Spiegelung von N Zeitbereichssamples und eine DFT der Länge $2N$ generiert werden. Dabei nimmt DCT die ungeraden Samples und transformiert sie auf die ersten N Koeffizienten der $2N-$DFT.

5.2 Diskrete Cosinus Transformation

Für die DCT gilt folgende Transformationsbeziehung

$$S_0 = \sqrt{\frac{1}{N}} \sum_{n=0}^{N-1} s_n \text{ und } S_k = \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} s_n \cos(\frac{(2n+1)k\pi}{2N}) \text{ für } 0 < k \leq N-1.$$

Die Rücktransformation, IDCT, ergibt sich dann aus

$$s_n = \frac{1}{\sqrt{N}} S(0) + \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} S_k \cos(\frac{(2n+1)k\pi}{2N}) \text{ für } 0 \leq k \leq N-1.$$

Die Transformationsmatrix ist eine unitäre Matrix, die im Beispiel für $N = 4$ folgendermaßen aussieht:

$$\underline{W} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \sqrt{2}\cos(\frac{\pi}{2N}) & \sqrt{2}\cos(\frac{3\pi}{2N}) & \ddots & \sqrt{2}\cos(\frac{(2n+1)\pi}{2N}) \\ \vdots & \vdots & \ddots & \vdots \\ \sqrt{2}\cos(\frac{(2n+1)k\pi}{2N}) & \sqrt{2}\cos(\frac{(2n+1)k\pi}{2N}) & \cdots & \sqrt{2}\cos(\frac{(2n+1)k\pi}{2N}) \end{bmatrix}.$$

DFT geht von der Periodizität des Signals und Spektrums und erzeugt Spektrale Anteile hoher Frequenz, die Unstetigkeiten an Übergängen zwischen zwei Signalperioden repräsentieren. DCT erzeugt ein reelles Spektrum, geht also davon aus, dass das reelle Signal eine gerade Funktion ist. Wie bei der DFT erhöht sich die Frequenz in den Matrixkoeffizienten von links oben bis rechts unten.

Dadurch dass DCT von einer gerader, also gespiegelten Funktion im Bereich $2N$ ausgeht, treten an den Übergängen zwischen zwei Signalperioden keine Sprünge. Die hochfrequente Spektrale Anteile sind vernachlässigbar klein im Vergleich zur DFT. Dadurch eignet sich DCT besser zur Kompression als DFT.